Canonical

# Ubuntu Server

© 2024 Canonical Ltd.
All rights reserved.

Canonical

# Contents


**1 Ubuntu Server tutorial** **3**
- 1.1 Getting started . . . 3
- 1.2 Next steps . . . 3

**2 Ubuntu Server how-to guides** **30**
- 2.1 Server installation . . . 30
- 2.2 Security . . . 111
- 2.3 Networking . . . 225
- 2.4 Managing software . . . 326
- 2.5 Data and storage . . . 362
- 2.6 Mail services . . . 454
- 2.7 Web services . . . 472
- 2.8 Graphics . . . 496
- 2.9 Virtualisation . . . 508
- 2.10 Containers . . . 544
- 2.11 High Availability . . . 569
- 2.12 Observability . . . 572

**3 Ubuntu Server reference** **606**
- 3.1 Glossary . . . 606
- 3.2 Server installation . . . 647
- 3.3 Data and storage . . . 648
- 3.4 High Availability . . . 651
- 3.5 Other tools . . . 652

**4 Ubuntu Server explanation guides** **657**
- 4.1 Security . . . 657
- 4.2 Networking . . . 721
- 4.3 Managing software . . . 753
- 4.4 Storage . . . 770
- 4.5 Web services . . . 801
- 4.6 Virtualisation and containers . . . 804
- 4.7 Clouds . . . 827
- 4.8 System tuning . . . 829
- 4.9 High Availability . . . 854
- 4.10 Debugging . . . 869

**5 Contribute to this documentation** **874**
- 5.1 The Open Documentation Academy . . . 874
- 5.2 Prerequisites . . . 874
- 5.3 The Ubuntu Server docs overview . . . 874
- 5.4 Contribution types . . . 874
- 5.5 General workflow . . . 875
- 5.6 Thank you! . . . 875

## Overview

**Ubuntu Server** is a version of the Ubuntu operating system designed and engineered as a backbone for the internet.

Ubuntu Server brings economic and technical scalability to your datacenter, public or private. Whether you want to deploy an OpenStack cloud, a Kubernetes cluster or a 50,000-node render farm, Ubuntu Server delivers the best value scale-out performance available.

This documentation always targets the latest LTS version of Ubuntu. When there are differences towards older LTS releases, they are noted in the guide.

## In this documentation

*Tutorial* Get started - a hands-on introduction to Ubuntu Server for new users

*How-to guides* Step-by-step guides covering key operations and common tasks

*Explanation* Concepts - Overviews of key topics, as well as more in-depth discussion and clarification

*Reference* Technical information - *system requirements*, package specifications and architecture

## Project and community

Ubuntu Server is a member of the Ubuntu family. It's an open source project that welcomes community projects, contributions, suggestions, fixes and constructive feedback.

If you find any errors or have suggestions for improvements, please use the "Give feedback" button at the top of every page. This will take you to GitHub where you can share your comments or let us know about bugs with any page.

- Read our Code of Conduct
- Get support
- Join the Discourse forum
- Get in touch via the `#ubuntu-server` IRC channel on Libera
- Download Ubuntu Server
- Find out how to *contribute to the Server Guide*

Thinking about using Ubuntu Server for your next project? Get in touch!

## PDF versions

- Current PDF (for Ubuntu 20.04 LTS onwards)
- Ubuntu 18.04 LTS PDF

# 1. Ubuntu Server tutorial

This tutorial should be a good place to start learning about Ubuntu Server in general, how it works, and what it's capable of.

In our "Getting started" tutorial you will learn how to set up an Ubuntu Server; from installing using a bootable USB device, to navigating the Server installer menu.

## 1.1. Getting started

- *Basic installation*

**The Server installer**

The Ubuntu installer has its own documentation. For more guidance on installing Ubuntu Server with the installer, you can also see these guides in the Ubuntu installer documentation (note: these pages will redirect you).

- Operating the Server installer
- Screen-by-screen installer guide
- Configuring storage
- Troubleshooting the installer

## 1.2. Next steps

Once your Ubuntu Server is up and running, you may be interested in this collection of related tutorials and topics that will help you learn more about how it works and what's available to you. These tutorials can be viewed in any order.

- *Managing software in Ubuntu Server*
- *Attach your Ubuntu Pro subscription*

If you have a specific goal, and are already familiar with Ubuntu Server, take a look at our *How-to guides*. These have more in-depth detail and can be applied to a broader set of applications.

Take a look at our *Reference section* when you need to determine what commands are available, and how to interact with various tools.

Finally, for a better understanding of how Ubuntu Server works and how it can be used and configured, our *Explanation section* enables you to expand your knowledge of the operating system and additional software.

### 1.2.1. Basic installation

This chapter provides an overview of how to install Ubuntu Server Edition. You can also refer to this guide on how to operate the installer for more information on using the installer, and to this screen-by-screen reference guide for more information about each of the installer screens.

## Preparing to install

This section explains various aspects to consider before starting the installation.

### System requirements

Ubuntu Server Edition provides a common, minimalist base for a variety of server applications, such as file/print services, web hosting, email hosting, etc. This version supports four 64-bit architectures:

- amd64 (Intel/*AMD* 64-bit)
- arm64 (64-bit ARM)
- ppc64el (POWER8 and POWER9)
- s390x (IBM Z and LinuxONE)

The recommended minimal system requirements for this tutorial are:

- RAM: 2 *GB* or more
- Disk: 5 GB or more

If you are looking for more general system requirements, *refer to this page*.

### Perform a system backup

Before installing Ubuntu Server Edition you should make sure all data on the system is backed up.

If this is not the first time an operating system has been installed on your computer, it is likely you will need to re-partition your disk to make room for Ubuntu.

Any time you partition your disk, you should be prepared to lose all data on the disk should you make a mistake or something goes wrong during partitioning. The programs used in installation are quite reliable, most have seen years of use, but they also perform destructive actions.

### Download the server ISO

You can obtain the amd64 server download from https://releases.ubuntu.com/. Select the version you wish to install and select the "server install image" download. Note that the server download includes the installer.

There are platform specific how-to guides for installations on:

- *s390x LPAR*
- *z/VM*
- *ppc64el*

### Create a bootable USB

There are many ways to boot the installer but the simplest and most common way is to create a bootable USB stick to boot from (tutorials for other operating systems are also available).

## Boot the installer

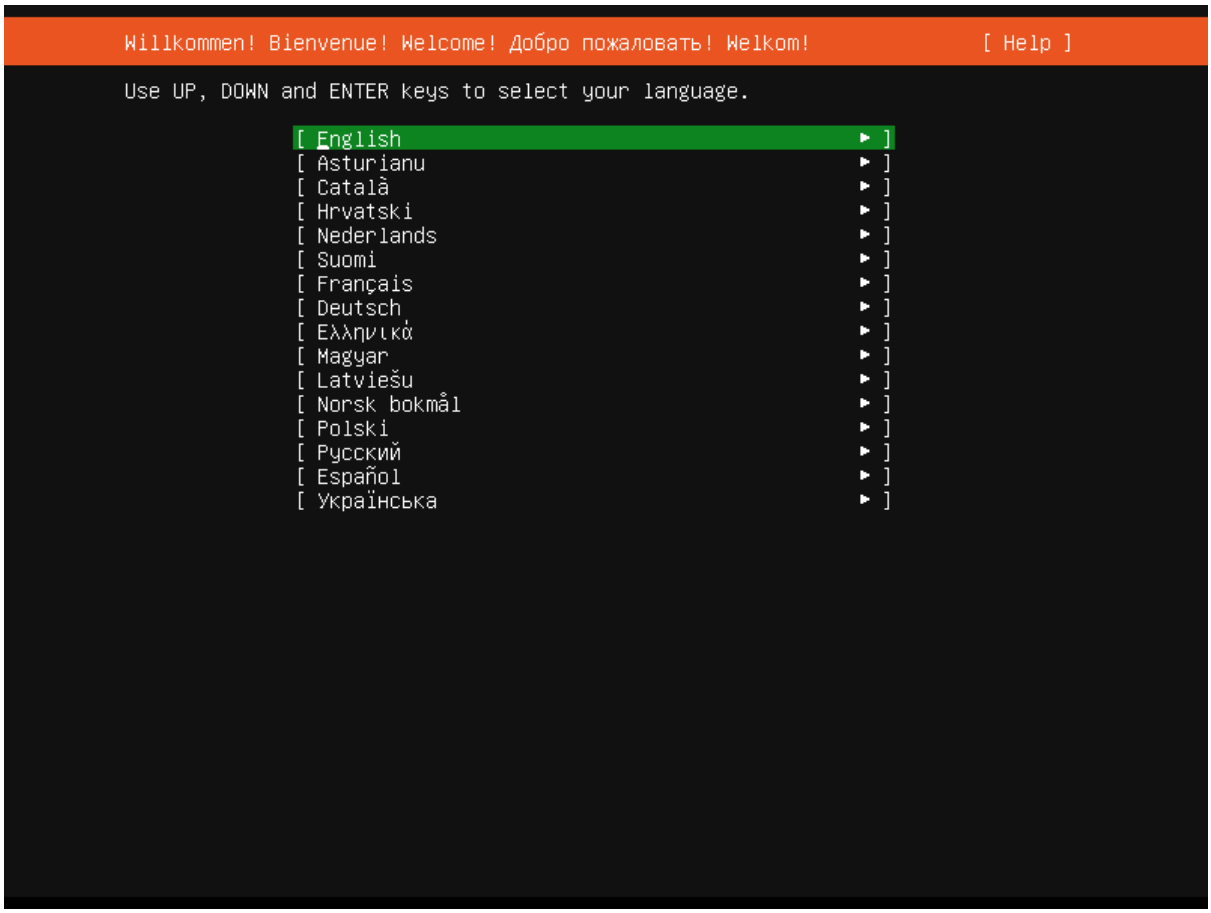
Plug the USB stick into the system to be installed and start it.

Most computers will automatically boot from USB or DVD, though in some cases manufacturers disable this feature to improve boot times. If you don't see the boot message and the "Welcome" screen which should appear after it, you will need to set your computer to boot from the install media.

There should be an on-screen message when the computer starts telling you what key to press for settings or a boot menu. Depending on the manufacturer, this could be Escape, F2, F10 or F12. Simply restart your computer and hold down this key until the boot menu appears, then select the drive with the Ubuntu install media.

If you are still having problems, check out the Ubuntu Community documentation on booting from CD/DVD.

After a few moments, the installer will start in its language selection screen.




## Using the installer

The installer is designed to be easy to use and have sensible defaults so for a first install you can mostly just accept the defaults for the most straightforward install:

- Choose your language
- Update the installer (if offered)
- Select your keyboard layout

- Do not configure networking (the installer attempts to configure wired network interfaces via *DHCP*, but you can continue without networking if this fails)
- Do not configure a proxy or custom mirror unless you have to in your network
- For storage, leave "use an entire disk" checked, and choose a disk to install to, then select "Done" on the configuration screen and confirm the install
- Enter a username, hostname and password
- On the SSH and snap screens, select "Done"
- You will now see log messages as the install is completed
- Select restart when this is complete, and log in using the username and password provided

If you need more details, however, the Ubuntu installer has its own documentation, including a screen-by-screen guide.

### 1.2.2. Managing your software

If you are new to Ubuntu, you may be wondering what to do after installation. After all, Ubuntu is endlessly customisable according to your needs. There are two types of software found in Ubuntu: **Debian packages** and **snaps** – we will learn about both!

To help you get the most from your Ubuntu experience, this tutorial will walk you through managing the software on your Ubuntu machine. This tutorial can be completed using either Ubuntu Server or Ubuntu Desktop.

To avoid making changes to your computer we will set up a virtual machine (VM), which will provide us with a safe environment to run the commands in. Multipass is great for quickly creating Ubuntu virtual machines, so we will use that.

#### Prerequisites

- **Knowledge:**

  None! You don't even need to use an Ubuntu machine – Multipass will give us an Ubuntu environment to play with.

- **Hardware:**

  The default Multipass VM will need **5** *GB* **of disk space**, and **1 GiB of memory**.

- **Software: – Multipass**

  - On Ubuntu, you can install Multipass by running the following command in your terminal (you can open a terminal window by pressing `Ctrl` + `Alt` + `T` together):

    ```
    sudo snap install multipass
    ```

    Or you can install it directly from the Multipass page in the online snap store (make sure to select the "latest/stable" version from the dropdown menu next to the install button).

  - Multipass can be installed on Windows, Mac and other Linux distributions using these instructions.

### Create the virtual machine

Once you have installed and run Multipass, it is straightforward to launch a new VM. Let us launch a VM using the Ubuntu 24.04 LTS release (codename `noble`), and let's give our VM the name `tutorial` using the following command in our terminal window:

```
multipass launch noble --name tutorial
```

Multipass will download the most recent daily **image** and create the VM for us. It may take a little time, depending on the speed of your internet connection.

An Ubuntu **image** is a collection of files we need to install and run Ubuntu. We don't need to specify "server" or "desktop" anywhere in our command, because the image is the same for both. The only difference between Ubuntu Server and Ubuntu Desktop is the subset of software packages we use from the Ubuntu Archive - we will see this later!

Now we can access the VM by running:

```
multipass shell tutorial
```

We will get a "Welcome to Ubuntu" message. Notice that when we run this command, the terminal username changes to `ubuntu` and the *hostname* changes to `tutorial`:

```
ubuntu@tutorial
```

This shows that we are inside the VM, and this is where we will run all our commands.

### Updating the system with APT

The first thing we always want to do with a new system (whether a VM, container, bare metal, or cloud instance) is to make sure we have the latest versions of all the pre-installed software.

Debian packages, commonly referred to as **debs**, are the standard software package format in Ubuntu. They can be identified by the `.deb` file extension.

Every Linux distribution has their own preferred **package manager** for installing, updating and removing packages. In Ubuntu, the default package manager is Advanced Packaging Tool (or APT, for short).

APT handles all of your system software (and other deb packages). It provides an interface to the Ubuntu Archive *repository*, so it can access both the **database** of all the packages available in Ubuntu, and the means to handle the **packages** themselves.

There are two APT commands we need to update our system: `update` and `upgrade`, which we will always run in that order.

### apt update

The `apt update` command is about the **database**. Any bug fixes in a package (or new versions since your last update) will be stored in the metadata about that package in the database (the **package index**).

When we run the `update` command it updates the APT database on our machine, fetching the newest available metadata from the package index:

```
sudo apt update
```

We will see an output like this:

```
Hit:1 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:2 http://archive.ubuntu.com/ubuntu noble InRelease
Hit:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
88 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

As we can see, it checks ("hits") the various archives (**pockets**) that updates can come from for the 24.04 LTS release (`noble-security`, `noble`, `noble-updates` and `noble-backports` – remember these, as we'll come back to them later). It has found some packages that can be upgraded to newer versions. If we want to see which packages those are, we can run the command hinted in the output:

```
apt list --upgradable
```

The output tells us:

- the package name and where the update will come from (e.g. `base-files/noble-updates`),
- the most up-to-date package version available (e.g. `13ubuntu10.1`)
- the hardware version the update is for (e.g. `amd64`), and
- what package version is currently installed (e.g. `13ubuntu10`)

The specific packages included in this list changes over time, so the exact packages shown will be different, but the output will be structured like this:

```
Listing... Done
base-files/noble-updates 13ubuntu10.1 amd64 [upgradable from: 13ubuntu10]
bsdextrautils/noble-updates 2.39.3-9ubuntu6.1 amd64 [upgradable from: 2.39.3-
9ubuntu6]
bsdutils/noble-updates 1:2.39.3-9ubuntu6.1 amd64 [upgradable from: 1:2.39.3-
9ubuntu6]
cloud-init/noble-updates 24.2-0ubuntu1~24.04.2 all [upgradable from: 24.1.3-
0ubuntu3.3]
[...]
```

### apt upgrade

The `apt upgrade` command is about the **packages** on your system. It looks at the metadata in the package index we just updated, finds the packages with available upgrades, and lists them for us. Once we've checked the proposed upgrade and are happy to proceed, it will then install the newer versions for us.

After we have updated the database (which we did by running `apt update`) we can then upgrade the packages to their newest versions by running:

```
sudo apt upgrade
```

When we run this command, it will ask us to confirm if the summary of proposed changes that will be made to our system is what we want.

Let's type `Y`, then press `Enter` to confirm that yes, we do want that, and then the upgrade will proceed. This may take a few minutes.

> **Tip**
>
> You can use the `-y` flag, which is a shorthand for `--assume-yes`. If we ran the command `sudo apt upgrade -y` it would proceed with the upgrade without asking us to confirm. Shorthand versions of flags are common – for most packages, you can check which flags are equivalent using the manual pages or using the `man` command, as we'll see later.

In the output, we'll see where `apt upgrade` is fetching the upgrade from for each package. For example:

```
Get:1 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 libopeniscsiusr
amd64 2.1.9-3ubuntu5.1 [49.1 kB]
```

APT combines the various elements; the package name (`libopeniscsiusr`), version (`2.1.9-3ubuntu5.1`), source (`noble-updates/main`), etc into a single URL that it can use for the download. The package is then unpacked, and the upgrade applied to the system.

> **Note**
>
> These commands only upgrade the packages for the release of Ubuntu that we are using (24.04 LTS). If we wanted to upgrade the entire system to the next release of Ubuntu (e.g. from 22.04 LTS to 24.04 LTS), we would use the `do-release-upgrade` command. See this guide on *how to upgrade your release* for more information.

It's important to know that `apt upgrade` will only handle packages that can be straightforwardly upgraded. If the package has **dependency** issues (i.e., the version you have "depends" on other packages that also need to be added, upgraded or removed), you would need to use `sudo apt dist-upgrade` instead. The `dist-upgrade` command is able to resolve conflicts between package versions, but it *could* end up removing some packages – so although `apt upgrade` is safe to use unattended (in a script, for example), you should only use `dist-upgrade` when you can pay attention to it.

### Searching with APT

Now we're up-to-date, we can start exploring! As with any other database, we can search the list of available packages using APT in order to find software. Let's say that we want to find a webserver, for example. We can run the following command:

```
apt search webserver
```

This will return us a long list of all "webserver" packages it can find. But some of the descriptions don't actually contain the text "webserver" – like in this section of the list:

```
inotify-tools/noble 3.22.6.0-4 amd64
  command-line programs providing a simple interface to inotify

ipcalc/noble 0.51-1 all
  parameter calculator for IPv4 addresses

iwatch/noble 0.2.2-10 all
  realtime filesystem monitoring program using inotify
```

We can use `apt show` to inspect the description and summary details of any package, so let's take a closer look at `ipcalc` from our list:

```
apt show ipcalc
```

The summary has been replaced with `[...]` for brevity, but we can see that the text "web-server" is in the long description of the "Description" field.

```
Package: ipcalc
Version: 0.51-1
[...]
APT-Sources: http://archive.ubuntu.com/ubuntu noble/universe amd64 Packages
Description: parameter calculator for IPv4 addresses
 ipcalc takes an IPv4 address and netmask and calculates the resulting
 broadcast, network, Cisco wildcard mask, and host range. By giving a
 second netmask, you can design sub- and supernetworks. It is also
 intended to be a teaching tool and presents the results as
 easy-to-understand binary values.
 .
 Originally, ipcalc was intended for use from the shell prompt, but a
 CGI wrapper is provided to enable colorful HTML display through a
 webserver.
 You can find it in /usr/share/doc/ipcalc/examples directory.
```

In many places, you will see reference to `apt-get` and `apt-cache` instead of `apt`. Historically, the *database* part of APT was accessed using `apt-cache` (e.g. `apt-cache show ipcalc`), and the *packages* part of APT used `apt-get` (e.g. `apt-get install ipcalc`).

APT has recently been streamlined, so although it uses `apt-get` and `apt-cache` "behind the scenes" (and these commands do still work), we don't need to worry about remembering which command to use – we can use the more convenient `apt` directly. To find out more about these packages and how to use them (or indeed, any package in Ubuntu!) we can refer to the manual pages.

Run `man apt`, `man apt-get` or `man apt-cache` in the terminal to access the manuals for these packages on the command line, or view the same content in the online manual pages.

### Installing deb packages

For the examples for this section, we're going to use the popular webserver package, Apache2.

APT gives us a lot of details about what will be included in the installation, and it's always important to understand the implications of a command *before* we run it. We'll be taking a

close look at the details APT gives us, so we need to be careful in this section.

When we run a command that asks us "`Do you want to continue? [Y/n]`", make sure to type `N` for "no" and then press `Enter` unless instructed otherwise – this will let us see the output of the commands without making changes that then need to be undone.

Installing deb packages using APT is done using the `apt install` command. We can install either a single package, or a list of packages at once, by including their names in a space-separated list after the `install` command, in this format:

```
sudo apt install <package 1> <package 2> <package 3>
```

### About `sudo`

We've seen the `sudo` prefix in a couple of commands already, and you may be wondering what that's about. In Linux, system tasks (like installing software) need elevated administrator permissions. These permissions are often called "root access", and a user with root access is called a "root user".

However, it can be dangerous to operate your machine as a root user – since root access gives you full system control the whole time, it allows you to change or delete important system files. It's very easy to accidentally break your system in root mode!

Instead, we use `sudo` (which is short for `superuser do`). This command is a safety feature that grants regular users *temporary* (per command) admin privileges to make system changes. It's still important for us to always understand what a command does before we run it, but using `sudo` means we purposefully limit any potential mistakes to a single command.

### About dependencies

As we hinted earlier, packages often come with **dependencies** – other packages that *your* package needs so it can function. Sometimes, a package might depend on a specific version of another package. If a package has dependencies, then installing a package via `apt` will also install any dependencies, which ensures the software can function properly.

APT tells us how it will resolve any dependency conflicts or issues when we run the `install` command. Let's try this for ourselves, but remember, we **don't** want to proceed with the install yet, so let's type `N` when it asks us if we want to continue:

```
sudo apt install apache2
```

The output should be similar to the below. It tells us:

- which packages we have but don't need (we'll talk about that in the "autoremove" section),
- additional packages that will be installed (these are our dependencies),
- suggested packages (which we'll discuss in the next section), and
- a summary of which *new* packages will be present on the system after the install is done (which in this case is `apache2` itself, and all its dependencies).

```
Reading package lists... Done
Building dependency tree... Done
```

(continues on next page)

(continued from previous page)

```
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1t64 libaprutil1-dbd-sqlite3
libaprutil1-ldap libaprutil1t64 liblua5.4-0 ssl-cert
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapr1t64 libaprutil1-dbd-
sqlite3 libaprutil1-ldap libaprutil1t64 liblua5.4-0 ssl-cert
0 upgraded, 10 newly installed, 0 to remove and 2 not upgraded.
Need to get 2084 kB of archives.
After this operation, 8094 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Let's try and make sense of this output.

### Types of dependencies

The relationship between a package and any other packages follows the Debian policy on binary dependencies, which we'll briefly look at here. The most common ones you might come across are: `depends`, `recommends`, and `suggests` (although there are others!), so we'll take a look at these three.

- **depends**: Absolutely required, the package won't work without it. If we try to remove a package that is depended on by another, both will be removed!
- **recommends**: Strongly dependent, but not absolutely necessary (which means the package will work better with it, but can still function without it)
- **suggests**: Not needed, but may enhance the usefulness of the package in some way.

We can see, using `apt show`, exactly which packages fall into each of these categories. Let's use Apache2 as our example again:

```
apt show apache2
```

If we look only at the sections on dependencies, we can see that `ssl-cert` is a recommended package:

```
[...]
Provides: httpd, httpd-cgi
Pre-Depends: init-system-helpers (>= 1.54~)
Depends: apache2-bin (= 2.4.58-1ubuntu8.4), apache2-data (= 2.4.58-1ubuntu8.4),
apache2-utils (= 2.4.58-1ubuntu8.4), media-types, perl:any, procps
Recommends: ssl-cert
Suggests: apache2-doc, apache2-suexec-pristine | apache2-suexec-custom, www-
browser, ufw
[...]
```

In Ubuntu, the default configuration of `apt install` is set to install recommended packages alongside `depends`, so when we ran the `apt install apache2` command, `ssl-cert` was included in the proposed packages to be installed (even though it's only recommended, not strictly needed).

We can override this behaviour by passing the `--no-install-recommends` flag to our command, like this:

```
sudo apt install apache2 --no-install-recommends
```

Then the output becomes the following (type `N` at the prompt again to avoid installing for now):

```
[...]
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1t64 libaprutil1-dbd-sqlite3
libaprutil1-ldap libaprutil1t64 liblua5.4-0
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser
Recommended packages:
  ssl-cert
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapr1t64 libaprutil1-dbd-
sqlite3 libaprutil1-ldap libaprutil1t64 liblua5.4-0
0 upgraded, 9 newly installed, 0 to remove and 25 not upgraded.
[...]
```

Now, we see that `ssl-cert` is only mentioned as a recommended package, but is excluded from the list of packages to be installed.

There is a second flag we could pass – the `--install-suggests` flag. This will not only install the strict dependencies and recommended packages, but *also* the suggested packages. From our previous output, it doesn't look like too much, right? It's only four additional packages.

But actually, if we run this command:

```
sudo apt install apache2 --install-suggests
```

There is now an extremely long list of suggested packages (which I will not output here, but you can try it for yourself!). In fact, the number of suggested packages is so long that there is not enough space in this VM to install them all, so it won't even give us the option to proceed:

```
[...]
0 upgraded, 4598 newly installed, 2 to remove and 0 not upgraded.
Need to get 7415 MB of archives.
After this operation, 19.6 GB of additional disk space will be used.
E: You don't have enough free space in /var/cache/apt/archives/.
```

This is because each of these suggested packages also comes with their own lists of dependencies, including suggested packages, all of which would *also* be installed. It's perhaps clear to see why this is not the default setting!

### What if we remove a dependency?

We'll go into more detail about removing packages later, but for now, let's see what happens if we remove a required *dependency*. First, we should (finally!) install the `apache2` package. Let's run the following command again, but this time when we are asked whether we want to continue, let's press `Y` and then `Enter` to confirm, and APT will install the package:

```
sudo apt install apache2
```

One of the required dependencies is the `apache2-data` package. Let's try to remove it using `apt remove`:

```
sudo apt remove apache2-data
```

Once again, `apt` won't proceed without confirmation, so we get the following output – let's take a look before choose anything:

```
[...]
The following packages were automatically installed and are no longer required:
  apache2-bin apache2-utils libapr1t64 libaprutil1-dbd-sqlite3 libaprutil1-ldap
libaprutil1t64 liblua5.4-0 ssl-cert
Use 'sudo apt autoremove' to remove them.
The following packages will be REMOVED:
  apache2 apache2-data
0 upgraded, 0 newly installed, 2 to remove and 2 not upgraded.
After this operation, 1342 kB disk space will be freed.
Do you want to continue? [Y/n]
```

Let's break this down a little bit, because there are some subtle differences here that we want to understand before we proceed.

- "The following packages were automatically installed and are no longer required"

  These were other dependencies that `apache2` needed, but none of them depend upon `apache2-data`, so even if we remove `apache2` and `apache2-data` they would still be functional – they just aren't used by any other installed packages…and so have no reason to be there anymore. They won't be removed, APT is helpfully telling us so we're aware of them.

- "The following packages will be REMOVED"

  These are the packages that will be removed directly - we've told APT we want to remove `apache2-data`, so we expect that to be included, but it will also remove `apache2` itself! This is because `apache2-data` is a required dependency, and `apache2` won't function *at all* without it.

Let's now choose `Y` to confirm we want to remove this dependency.

**Warning**

Removing dependencies can, at worst, cause a system to become unusable – you should always be careful when doing so. If you remove a dependency that is part of a chain, the removals will cascade up the chain as each dependency and the package that depends on it are removed. You can end up removing more than you originally anticipated!

### Autoremove dependencies

So, we have removed the `apache2` and `apache2-data` packages, but the other dependencies that were installed alongside `apache2` are still there. The output of our `remove` command gave us the hint about how to deal with these redundant packages – the `autoremove` command:

```
sudo apt autoremove
```

When we run this command, `apt` once again gives us a summary of the operation we requested, but let's choose `N` for now when it asks if we want to continue:

```
[...]
The following packages will be REMOVED:
  apache2-bin apache2-utils libapr1t64 libaprutil1-dbd-sqlite3 libaprutil1-ldap
libaprutil1t64 liblua5.4-0 ssl-cert
0 upgraded, 0 newly installed, 8 to remove and 2 not upgraded.
After this operation, 6751 kB disk space will be freed.
Do you want to continue? [Y/n]
```

You may be wondering why we don't need to specify any packages when we call the `autoremove` command – after all, we've just been dealing with packages related to `apache2`. This is because `apt` will check all the packages on your system. It examines the dependency tree, and if the original reason for the package to be installed no longer exists (i.e., it isn't needed by anything), it will be flagged for autoremoval.

But!

We might, in the future, uninstall Apache2 without uninstalling the redundant packages at the time. We might have found another use for `ssl-cert`, perhaps in a script that makes use of SSL certificates. So how can we keep the `ssl-cert` package, even though it's flagged for autoremoval?

We can solve this problem, and un-flag the `ssl-cert` package for removal, by *manually* installing it:

```
sudo apt install ssl-cert
```

This sets `ssl-cert` to **manually installed**. We might well wonder "why didn't APT didn't ask us to confirm anything this time?". In this case, it's because `ssl-cert` is already present on the system so APT doesn't need to install anything new.

```
[...]
ssl-cert is already the newest version (1.1.2ubuntu1).
ssl-cert set to manually installed.
The following packages were automatically installed and are no longer required:
  apache2-bin apache2-utils libapr1t64 libaprutil1-dbd-sqlite3 libaprutil1-ldap
libaprutil1t64 liblua5.4-0
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 2 not upgraded.
```

If the `ssl-cert` package is manually installed on our system, by us, then `apt` knows the package is wanted, and we can see that it has been removed from the autoremove list so our next autoremove will not uninstall it. Let's test this, just to make sure!

```
sudo apt autoremove
```

This time we'll select `Y` when prompted, and then we can run `apt list ssl-cert` to quickly see if our `ssl-cert` package is still on the system:

```
apt list ssl-cert
```

Which gives us this output, confirming that `ssl-cert` is currently installed:

```
Listing... Done
ssl-cert/noble,now 1.1.2ubuntu1 all [installed]
```

If you're curious, you can also run `apt list apache2` to see how the output differs for a package that was once installed and then removed!

Anyway, we're not quite finished with the Apache2 package, so let's reinstall it:

```
sudo apt install apache2
```

And this time select `Y` to confirm when it asks.

### Customise configuration

In general, the default package configuration should just work well, and work "out of the box" when it's installed. But it's almost inevitable that, sooner or later, we'll want to customise the package so that it better fits our own purposes.

Before we try to customise the package, we should probably look at what files are included in it. We can check this using `dpkg(1)`, which is the Debian package manager. Although APT is now more commonly used for basic package handling, `dpkg` retains some really helpful commands for examining files and finding out package information. It's installed by default on Ubuntu systems so we can use it directly:

```
dpkg --listfiles ssl-cert
```

This gives us the following list of files and their directory structure (the end of the list is truncated for brevity):

```
/.
/etc
/etc/ssl
/etc/ssl/certs
/etc/ssl/private
/lib
diverted by base-files to: /lib.usr-is-merged
/lib/systemd
/lib/systemd/system
/lib/systemd/system/ssl-cert.service
/usr
/usr/sbin
/usr/sbin/make-ssl-cert
/usr/share
/usr/share/doc
```

(continues on next page)

(continued from previous page)

```
/usr/share/doc/ssl-cert
/usr/share/doc/ssl-cert/README
[...]
```

If we find a file but we're not sure what package it comes from, `dpkg` can help us there too! Let's use the example of one of the files from the previous output: `/usr/share/ssl-cert/ssleay.cnf` and do a search for it using `dpkg`:

```
dpkg --search /usr/share/ssl-cert/ssleay.cnf
```

This will provide us with the package name for the given file:

```
ssl-cert: /usr/share/ssl-cert/ssleay.cnf
```

Although this seems obvious to us, because we already know the source of this file, the `dpkg` search function is really useful for tracking down the sources of files we don't know about!

### Conffiles

Most of a package's configuration is handled through configuration files (often known as **conffiles**). Conffiles often contain things like file paths, logs and debugging configuration, kernel parameters (which can be changed to optimise system performance), access control, and other configuration settings. The actual parameters available will vary from one package to another.

Package conffiles are different from all other files delivered in a package. A package may have any number of conffiles (including none!). Conffiles are explicitly marked by the package maintainer during development to protect local configuration from being overwritten during upgrades so that your changes are saved. This is not the case for any other types of files – changes you make to regular files in that package *will be overwritten* during an upgrade.

### How upgrades are handled

Since a conffile can be changed by us, we might end up with conflicts when the package maintainer changes those same files. Therefore, it's important to understand how such conflicts are handled.

We can show the four possible upgrade scenarios using the following table. What happens during an upgrade depends on whether the conffile on our system has been changed by us ("changed/not changed by user"), and whether the version's default content has been changed by the package maintainer ("changed/not changed by maintainer"):

| The conffile is... | **not changed by maintainer** | **changed by maintainer** |
|---|---|---|
| **...changed by user** | Keep user's changes | Ask user |
| **...not changed by user** | No changes to make | Apply changes from update |

So we can see that if we do make changes to a conffile, APT will never overwrite our changes without asking us first.

### Identifying conffiles

Out of the list of files in a package, how do we know which ones are the conffiles?

After all, they are not marked by any particular file extension, and although they are often found in the `/etc/` directory, they don't *have* to be there. As we saw before, the only thing conffiles have in common is that the package maintainer decided to mark them as such.

But that's our clue! So once more, `dpkg` can come to our rescue. The following command will show us (`--show`) the subset of files in the `apache2` package that have been marked as "`Conffiles`" (`-f='${Conffiles}\n'`) by the maintainer and shows each on a new line (`\n`) in the output:

```
dpkg-query --show -f='${Conffiles}\n' apache2
```

If you want to understand more about what this command does, you can refer to the manual page by typing `man dpkg-query --show`, and it will talk you through all the options.

Unlike `dpkg --listfiles`, `dpkg-query` *also* gives us a string of letters and numbers. This string is known as the **"MD5 checksum"** or **"MD5 hash"**.

```
 /etc/apache2/apache2.conf 354c9e6d2b88a0a3e0548f853840674c
 /etc/apache2/conf-available/charset.conf e6fbb8adf631932851d6cc522c1e48d7
 /etc/apache2/conf-available/security.conf 332668933023a463046fa90d9b057193
 /etc/apache2/envvars e4431a53c868ae0dfcde68564f3ce6a7
 /etc/apache2/magic a6d370833a02f53db6a0a30800704994
[...]
```

We can see the checksum of a specific file by running this command:

```
md5sum /etc/apache2/apache2.conf
```

Which returns us the checksum followed by the file and its location:

```
354c9e6d2b88a0a3e0548f853840674c  /etc/apache2/apache2.conf
```

You might well be wondering "why do we care about that?" since they match (in this example).

The checksum is like a fingerprint - it's unique for every *version* of a file, so any time the file is changed it will get a new checksum – which allows us to see **if a file has been changed** from the default.

### Verifying checksums

Let's set up a situation so we can poke a bit at this idea. We can start by making some changes to a conffile. In Apache2, the main conffile is `/etc/apache2/apache2.conf`, so let's use that. In a situation where we are setting up a new webserver, we might reasonably want to increase the `LogLevel` from "warn" to "debug" to get more debugging messages, so let's run this command and use `sed` to make that change in the conffile:

```
sudo sed -e 's/LogLevel warn/LogLevel debug/' -i /etc/apache2/apache2.conf
```

We won't be prompted to confirm if we want to make these changes – but we do need root access so we use `sudo` in our command. As we hinted in the section about `sudo`, the fact that we can make these changes without needing to confirm is why it can be so easy to break your

system when you're operating as root! Try running the command without the `sudo`, and you will get a "permission denied" error.

Next, we'll restart our Apache2 server so that we can activate our configuration changes:

```
sudo systemctl restart apache2
```

Now if we run the `md5sum` command again, we can see the hash changed:

```
ubuntu@tutorial:~$ md5sum /etc/apache2/apache2.conf

1109a77001754a836fb4a1378f740702  /etc/apache2/apache2.conf
```

This works great if we know that there's a file *we* changed, but what about if someone else tampered with a file, and we don't know which one? In that case, we can use:

```
dpkg --verify apache2
```

This will verify the checksums of the files on our system against those held in the package index for `apache2`, and return a rather strange looking result if (or when) it finds a mismatch:

```
??5?????? c /etc/apache2/apache2.conf
```

Which is exactly what we were expecting to see, since we know we changed this file.

But what if something else was messed with…something that shouldn't be, and something not changed by us? Let's make a "silly" change to a different file to test this – in this case, changing all instances of the word "warning" to "silly" in a random package file:

```
sudo sed -e 's/warning/silly/' -i /usr/sbin/a2enmod
```

And then run the verification again with:

```
dpkg --verify apache2
```

We now see something that looks like this:

```
??5?????? c /etc/apache2/apache2.conf
??5??????   /usr/sbin/a2enmod
```

**Note**

You might have noticed there's a "c" next to the top line but not the bottom – the "c" shows the file is a conffile.

`dpkg` can tell that the file has been changed, but won't tell us what the change was. However, since the file in question is not a conffile, we know that the change *won't be preserved* if we upgrade the package. This means that we can overwrite the changes and restore the default package content by "reinstalling" Apache2:

```
sudo apt install --reinstall apache2
```

By using the `--reinstall` flag, we can force `apt` to re-unpack all of the default content. If we then verify once more…

```
dpkg --verify apache2
```

Then we'll get this output:

```
??5?????? c /etc/apache2/apache2.conf
```

…so we can see that our change to the conffile has been preserved because the checksums are different, but the `a2enmod` file isn't listed anymore because it has been restored to the default. Phew!

> **Note**
>
> We can use `sudo apt install <package>` to upgrade an installed package, but this will only upgrade to the latest version. In our case, we were already on the latest version of Apache2, so we needed to force APT to re-unpack the content to overwrite our "silly" changes.

### Removing packages

Since we have just reinstalled the Apache2 package, we know it is in good shape. But what if we decide we're done with it and just want to remove it? Then we can run:

```
sudo apt remove apache2
```

Which will give us an output like this:

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  apache2-bin apache2-data apache2-utils libapr1t64 libaprutil1-dbd-sqlite3
  libaprutil1-ldap libaprutil1t64 liblua5.4-0
Use 'sudo apt autoremove' to remove them.
The following packages will be REMOVED:
  apache2
0 upgraded, 0 newly installed, 1 to remove and 44 not upgraded.
After this operation, 465 kB disk space will be freed.
Do you want to continue? [Y/n]
```

Let's type `Y` to proceed.

As before, we see that the dependencies will still be there even when `apache2` has been removed. Let's check with `dpkg`…

```
dpkg --listfiles apache2
```

…and see what else might be left behind…

```
/etc
/etc/apache2
/etc/apache2/apache2.conf
```

(continues on next page)

(continued from previous page)

```
/etc/apache2/conf-available
/etc/apache2/conf-available/charset.conf
/etc/apache2/conf-available/localized-error-pages.conf
/etc/apache2/conf-available/other-vhosts-access-log.conf
/etc/apache2/conf-available/security.conf
/etc/apache2/conf-available/serve-cgi-bin.conf
/etc/apache2/conf-enabled
/etc/apache2/envvars
/etc/apache2/magic
/etc/apache2/mods-available
/etc/apache2/mods-available/access_compat.load
/etc/apache2/mods-available/actions.conf
[...]
```

This looks suspiciously like the list of conffiles we saw earlier, right?

### Also removing configuration

As it turns out, removing a package doesn't automatically remove the conffiles. But – this is intentional, for our convenience.

By leaving the conffiles in place, if we decide to reinstall `apache2` again in the future, we don't need to spend time setting up all our configuration again.

Let's see the difference in installing `apache2` after it has been installed (and removed) compared to the first time we installed it:

```
sudo apt install apache2
```

Notice that it did not ask us to confirm if we wanted to proceed this time. Why not? As we saw earlier, the "Y/n" confirmation is shown when there are dependencies, and we know that Apache2 *has* dependencies.

…Ah! But this time, we didn't run `autoremove` when we uninstalled Apache2, so the dependencies are still installed on our system. This means that when we ask `apt` to install `apache2` now, there is nothing missing and we are getting *exactly* what we are asking for.

Since the dependencies and conffiles are still there, we can use our former config immediately. It even retains the changes we made before, which we can verify by looking at the checksum again:

```
md5sum /etc/apache2/apache2.conf
```

### Removing and purging

What if we decide that we don't want the changed conffiles? Perhaps we want to go back to the default installation, or we know we won't want to use the package ever again – how can we ensure that all the conffiles are removed at the same time as we remove the package?

In that case, we can use the `--purge` option of the `remove` command:

```
sudo apt remove --purge apache2
```

Which will give us this output:

```
[...]
The following packages were automatically installed and are no longer required:
  apache2-bin apache2-data apache2-utils libapr1t64 libaprutil1-dbd-sqlite3
  libaprutil1-ldap libaprutil1t64 liblua5.4-0
Use 'sudo apt autoremove' to remove them.
The following packages will be REMOVED:
  apache2*
0 upgraded, 0 newly installed, 1 to remove and 9 not upgraded.
After this operation, 465 kB disk space will be freed.
Do you want to continue? [Y/n]
```

If we look very carefully, we see a little asterisk (*) in the output.

```
The following packages will be REMOVED:
  apache2*
```

This tiny indicator tells us that the package will be removed AND purged. However, it still does not remove the dependencies (or the conffiles of those dependencies).

Let's type `Y` again to confirm we want to proceed. Then, once the removal is complete, we can check the list once more:

```
dpkg --listfiles apache2
```

And this time, the output is very different!

```
dpkg-query: package 'apache2' is not installed
Use dpkg --contents (= dpkg-deb --contents) to list archive files contents.
```

> **Note**
>
> We could also use the `dpkg-query --show -f='${Conffiles}\n' apache2` command from earlier, and `dpkg-query` will find no packages matching `apache2`.

There are other ways to change package files. If you would like to read more, check out our *guide to changing package files*.

### What else is on our system?

As we saw earlier, we can search the APT package database for keywords using `apt search <keyword>` to find software we might want to install. We can also see all the packages we already have using `apt list`, although it can be easier to navigate and more informative if we use `dpkg -l` instead – then we can use the up and down arrow keys on our keyboard to scroll (or press `Q` to return to our terminal prompt).

For every package, we can see what versions of it exist in the database:

```
apt policy apache2
```

This will return a summary of all the versions that exist on our particular Ubuntu release, ordered by "most recent" first:

```
apache2:
  Installed: (none)
  Candidate: 2.4.58-1ubuntu8.4
  Version table:
     2.4.58-1ubuntu8.4 500
        500 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages
        500 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages
        100 /var/lib/dpkg/status
     2.4.58-1ubuntu8 500
        500 http://archive.ubuntu.com/ubuntu noble/main amd64 Packages
```

We know that Apache2 isn't installed right now, because we removed and purged it, which is why the installed version shows as "none":

```
Installed: (none)
```

If we were to install the default package, we would get this one:

```
Candidate: 2.4.58-1ubuntu8.4
```

Under each version we are also shown the **source**. The newest version (`2.4.58-1ubuntu8.4`) comes from `noble-updates` (main) and `noble-security` (main). The *original* version (`2.4.58-1ubuntu8`) comes from `noble` (main). This tells us that this was the version released with the with 24.04 LTS (Noble Numbat).

### Installing older package versions

We can install specific older versions if we want to, for example, to satisfy dependency requirements of another package. We can do that by specifying the package name and version:

```
sudo apt install <package=version>
```

However, this can be tricky and often leads to conflicts in dependency versions as APT always wants to install the most recent version. We can see an example of this if we run the following command:

```
sudo apt install apache2=2.4.58-1ubuntu8
```

APT warns us that the version of apache2 we want to install depends on earlier versions of the dependencies, but it helpfully tells us which dependency versions we need to successfully install the package we want.

```
[...]
Some packages could not be installed. This may mean that you have
requested an impossible situation or if you are using the unstable
distribution that some required packages have not yet been created
or been moved out of Incoming.
The following information may help to resolve the situation:

The following packages have unmet dependencies:
 apache2 : Depends: apache2-bin (= 2.4.58-1ubuntu8) but 2.4.58-1ubuntu8.4 is to be
```

(continues on next page)

(continued from previous page)

```
installed
           Depends: apache2-data (= 2.4.58-1ubuntu8) but 2.4.58-1ubuntu8.4 is to
be installed
           Depends: apache2-utils (= 2.4.58-1ubuntu8) but 2.4.58-1ubuntu8.4 is to
be installed
E: Unable to correct problems, you have held broken packages.
```

So, all we need to do is first install the dependencies, and then run the install command again. Remember that we can install multiple packages at once by separating them with spaces:

```
sudo apt install apache2-bin=2.4.58-1ubuntu8 \
  apache2-data=2.4.58-1ubuntu8 \
  apache2-utils=2.4.58-1ubuntu8 \
  apache2=2.4.58-1ubuntu8
```

In this case we're also breaking the command over multiple lines using backslashes (\) to make it easier to read, but it will still be run as a single command.

APT will warn us that we are downgrading the package, but let us press `Y` to confirm (when prompted), and it will go ahead and downgrade us anyway. Let's run the following command again:

```
apt policy apache2
```

And we'll get confirmation that we're running on an older version:

```
apache2:
  Installed: 2.4.58-1ubuntu8
  Candidate: 2.4.58-1ubuntu8.4
  Version table:
     2.4.58-1ubuntu8.4 500
        500 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages
        500 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages
 *** 2.4.58-1ubuntu8 500
        500 http://archive.ubuntu.com/ubuntu noble/main amd64 Packages
        100 /var/lib/dpkg/status
```

### Where do packages come from?

You may be wondering by now "where exactly do all these packages come from?". We've spotted a few sources very briefly throughout this tutorial, but haven't paid direct attention to them yet. Let's take a little time now to define what we mean by all these different sources that APT can pull packages from.

The source behind APT is the **Ubuntu Package Archive**. This Archive splits into many layers, each with its own terminology. The different terminology is quite confusing at first, but we've seen a few of the terms already. So if we take a look, layer-by-layer, we'll see not just what all the terms mean, but how they all fit together.

Let's have a quick overview with this diagram. The general flow is that the Archive splits into **Ubuntu series**. Each series is split up into **pockets**, and then each pocket contains four

**components**. If we tried to show all of this on one diagram, it would be quite extensive, so let's take a look through a single path.

### Series

The **series** is a set of packages that are released with a specific version of Ubuntu – they're usually referred to by their codename (e.g., `mantic`, `noble` and `oracular` in our diagram). Each version of Ubuntu may have multiple releases (for example, an LTS will have an initial release when it launches (e.g. 24.04 LTS), and then "subsequent point releases" (e.g. 24.04.1 LTS) – these are all part of the same series (`noble`).

In practice, people often use the term "Ubuntu release" and "Ubuntu series" interchangeably.

### Pockets

Every Ubuntu series (`noble`, `jammy`, etc) is split into **pockets**, which are related to the software development/release lifecycle:

- **-release** contains the packages as they are at release time.
- **-proposed** contains package updates while they are being tested.
- Once an update is released, they come from either **-security** or **-updates** depending on whether they are a security-related update or not.
- And **-backports**, which contains packages that were not available at release time.

This is why earlier, we saw that some updates came from `noble-updates` or `noble-security`. These refer to updates and security updates from the noble series (respectively). Pockets are usually appended to the end of the series, and it's quite common to see the hyphen (`-`) included when referring to pockets.

Remember – the original version of the `apache2` package we saw came from `noble`. The `-release` pocket only includes the software that was part of the original LTS release, and so it takes the name of the Ubuntu series by default (i.e., the `-release` pocket is implied).

### Components

Each pocket is split into four **components**, depending on whether the packages they contain are *open source* or *closed source*, and whether they are officially supported by Canonical or are maintained by the Ubuntu Community:

| | Open source | Closed source |
|---|---|---|
| **Officially supported** | main | restricted |
| **Community supported** | universe | multiverse |

- **main** contains the open-source packages that are officially supported by Canonical. These packages are either installed on every Ubuntu machine, or are very widely used for various types of systems and use-cases.
- **universe** holds all other open-source packages in Ubuntu, which are typically maintained by the Debian and Ubuntu communities, but may also include additional security coverage from Canonical under Ubuntu Pro, which is available free for personal use on up to five machines.

- **restricted** contains the packages that are officially supported by Canonical but are not available under a completely free license.
- **multiverse** contains community-maintained proprietary software – these packages are completely unsupported by Canonical.

If you would like more information about the Ubuntu release process, how packages are produced, or to learn more about the sort of terminology you might come across, you may be interested in the Ubuntu Packaging Guide, which is a great resource containing all this information (and much more!).

### Installing a .deb file

Although APT is the preferred way to install packages on your system, due to its ability to handle depedencies and keep software up-to-date, not every package is available in the APT repository – especially if they are so old they are no longer maintained, or conversely, are the newest version still in development!

We can install .deb files that aren't in the APT repository using `dpkg` – all we need is to download the .deb file, and we can run a command like this to install it:

```
sudo dpkg -i <file-name.deb>
```

But – APT is helpful here too. Even if we get a .deb file that isn't from the Ubuntu Archive, we can still install it with APT so that if there are dependencies that can be resolved automatically from the Archive – they will be!

```
sudo apt install ./file-name.deb
```

If we ever do want to install a .deb file, APT is definitely the most convenient way to do it. We may still need to handle *some* dependencies manually, but now we have the knowledge to be able to do that.

Luckily, most of the packages you will ever need are likely to be found through APT. If it's not, it's worth checking if the software is available as a **snap** instead.

### Snaps

Snaps are a newer, self-contained software format that were developed to be a more portable and easy-to-use alternative to debs. They come with all their dependencies pre-bundled so that there is no need for a package management tool to track dependencies, and they run inside sandboxed environments that limit their interactions with the rest of the system.

Instead of **versions** as we have them in debs, snaps use the concept of channels to define which release of a snap is installed.

By default, snaps are kept automatically up-to-date, so we don't need to remember to update and upgrade them. There are times on a live system, such as a server in a production environment, where we might not want to have updates automatically applied. In those cases, we can turn off automatic updates and refresh the system snaps when it's convenient (for example, during a maintenance window).

If you would like to try out snaps, we recommend the excellent quickstart tour tutorial in the snap documentation. Feel free to continue using the VM we've been using in this tutorial while exploring!

### Completion!

Once you are finished and want to leave the tutorial, you can run:

```
exit
```

This will take you out of the VM and back to your live machine. Then, you can run the following commands to delete the VM and remove it completely from your machine:

```
multipass delete tutorial
multipass purge
```

### Summary

Congratulations, we made it to the end! We've covered a lot of material in this tutorial, so let's do a quick recap of what we've learned:

**Finding, installing and removing packages**

- How to update and upgrade all our system's software with APT: * `sudo apt update && sudo apt upgrade`
- How to search for software using keywords or strings: * `apt search <keyword>` or `apt search "some content string"`
- How to see the description of a package, including what dependencies it has: * `apt show <package name>`

  Or how to check what package versions are available: * `apt policy <package>`
- How to install packages… * `sudo apt install <package1> <package2>`
- How to see all the files a package contains * `dpkg --listfiles <package>`
- How to find out what package a file belongs to: * `dpkg --search <path/to/file>`
- …And how to remove packages again! As well as the difference between removing and purging. * `sudo apt remove <package>`
- We even learned how to downgrade to older versions of APT packages, and all about APT sources.

**Customising package configuration**

- How to find the conffiles in a package: * `dpkg-query --show -f='${Conffiles}\n' <package>`
- How to see if package files have been changed: * `dpkg --verify <package>`
- …And if a non-conffile has been changed by accident, we can fix it with: * `sudo apt install --reinstall <package>`
- We know that our changes to conffiles are always safely preserved, while changes to non-conffiles are reverted at the next upgrade or security fix.
- Importantly, we know how to verify checksums with `md5sum` or similar tools, which helps us to more safely build packages from source.
- And finally, we learned about snaps!

### 1.2.3. Attach your Ubuntu Pro subscription

Attaching the Ubuntu Pro subscription to Ubuntu brings you the enterprise lifecycle, including Linux kernel livepatching, access to FIPS-validated packages, and compliance with security profiles such as CIS. This is not required for Ubuntu Pro instances through public clouds such as AWS, Azure or GCP, since these are automatically attached from launch.

> **Note**
>
> Subscriptions are not just for enterprise customers. Anyone can get a personal subscription for free on up to 5 machines, or 50 if you are an official Ubuntu Community member.

The following instructions explain how to attach your subscription to your Ubuntu systems.

#### Step 1: Install the Ubuntu Pro Client

This step is necessary for Ubuntu Pro users or holders of personal subscriptions. If you are an Ubuntu Pro user through a public cloud offering, your subscription is already attached and you can skip these instructions.

First, make sure that you have the latest version of the Ubuntu Pro Client running. The package used to access the Pro Client (`pro`) is `ubuntu-advantage-tools`:

```
sudo apt update && sudo apt install ubuntu-advantage-tools
```

If you already have `ubuntu-advantage-tools` installed, this install command will upgrade the package to the latest version.

#### Step 2: Attach your subscription

To attach your machine to a subscription, run the following command in your terminal:

```
sudo pro attach
```

You should see output like this, giving you a link and a code:

```
ubuntu@test:~$ sudo pro attach
Initiating attach operation...

Please sign in to your Ubuntu Pro account at this link:
https://ubuntu.com/pro/attach
And provide the following code: H31JIV
```

Open the link without closing your terminal window.

In the field that asks you to enter your code, copy and paste the code shown in the terminal. Then, choose which subscription you want to attach to. By default, the Free Personal Token will be selected.

If you have a paid subscription and want to attach to a different token, you may want to log in first so that your additional tokens will appear.

Once you have pasted your code and chosen the subscription you want to attach your machine to, click on the “Submit” button.

The attach process will then continue in the terminal window, and you should eventually be presented with the following message:

```
Attaching the machine...
Enabling default service esm-apps
Updating Ubuntu Pro: ESM Apps package lists
Ubuntu Pro: ESM Apps enabled
Enabling default service esm-infra
Updating Ubuntu Pro: ESM Infra package lists
Ubuntu Pro: ESM Infra enabled
Enabling default service livepatch
Installing snapd snap
Installing canonical-livepatch snap
Canonical Livepatch enabled
This machine is now attached to 'Ubuntu Pro - free personal subscription'
```

When the machine has successfully been attached, you will also see a summary of which services are enabled and information about your subscription.

Available services can be enabled or disabled on the command line with `pro enable <service name>` and `pro disable <service name>` after you have attached.

### Next steps

- For more information about the Ubuntu Pro Client, you can read our documentation.
- For a guided tour through the most commonly-used commands available through the Ubuntu Pro Client, check out this tutorial.

# 2. Ubuntu Server how-to guides

If you have a specific goal, but are already familiar with Ubuntu Server, our how-to guides have more in-depth detail than our tutorials and can be applied to a broader set of applications. They'll help you achieve an end result but may require you to understand and adapt the steps to fit your specific requirements.

## 2.1. Server installation

### 2.1.1. Server installation

If you are new to Ubuntu, we recommend our *basic installation* tutorial to get you started.

#### Automatic install

The Ubuntu Installer has its own documentation for automatic (or "hands off") installations. These guides from the Ubuntu Installer documentation are available for automatic installations.

- Introduction to Automated Server installer
- Autoinstall quickstart
- Autoinstall quickstart on s390x

#### Advanced install

This list of guides contains installation instructions for architecture-specific and more advanced setups. Select your preferred architecture to see which guides are available.

**amd64**

- *Netboot install*

**arm64**

- *Netboot install*
- *Choose between the arm64 and arm64+largemem installer options*

**ppc64el**

- *Netboot install*
- *Virtual CD-ROM and Petitboot install*

**s390x**

- *Install via z/VM*
- *Non-interactive IBM z/VM autoinstall*
- *Install via LPAR*
- *Non-interactive IBM Z LPAR autoinstall*

### See also

- Reference: *System requirements*

### How to netboot the server installer on amd64

*amd*64 systems boot in either UEFI or legacy ("BIOS") mode, and many systems can be configured to boot in either mode. The precise details depend on the system *firmware*, but both modes usually support the "Preboot eXecution Environment" (PXE) specification, which allows the provisioning of a bootloader over the network.

### Steps needed

The process for network booting the live server installer is similar for both modes and goes like this:

1. The to-be-installed machine boots, and is directed to network boot.
2. The *DHCP*/BOOTP server tells the machine its network configuration and where to get the bootloader.
3. The machine's firmware downloads the bootloader over TFTP and executes it.
4. The bootloader downloads configuration, also over TFTP, telling it where to download the kernel, RAM Disk and kernel command line to use.
5. The RAM Disk looks at the kernel command line to learn how to configure the network and where to download the server ISO from.
6. The RAM Disk downloads the ISO and mounts it as a loop device.
7. From this point on the install follows the same path as if the ISO was on a local block device.

The difference between UEFI and legacy modes is that in UEFI mode the bootloader is an *EFI* executable, signed so that is accepted by Secure Boot, and in legacy mode it is PXELINUX. Most DHCP/BOOTP servers can be configured to serve the right bootloader to a particular machine.

### Configure DHCP/BOOTP and TFTP

There are several implementations of the DHCP/BOOTP and TFTP protocols available. This document will briefly describe how to configure *dnsmasq* to perform both of these roles.

1. Install `dnsmasq` with:

```
sudo apt install dnsmasq
```

2. Put something like this in `/etc/dnsmasq.d/pxe.conf`:

```
interface=<your interface>,lo
bind-interfaces
dhcp-range=<your interface>,192.168.0.100,192.168.0.200
dhcp-boot=pxelinux.0
dhcp-match=set:efi-x86_64,option:client-arch,7
dhcp-boot=tag:efi-x86_64,bootx64.efi
```

(continues on next page)

(continued from previous page)

```
enable-tftp
tftp-root=/srv/tftp
```

**Note**

This assumes several things about your network; read `man dnsmasq` or the default `/etc/dnsmasq.conf` for many more options.

1. Restart `dnsmasq` with:

```
sudo systemctl restart dnsmasq.service
```

### Mode-independent set up

1. Download the latest live server ISO for the release you want to install:

```
wget http://cdimage.ubuntu.com/ubuntu-server/noble/daily-live/current/noble-live-server-amd64.iso
```

2. Mount it:

```
sudo mount noble-live-server-amd64.iso /mnt
```

3. Copy the kernel and `initrd` from it to where the `dnsmasq` serves TFTP from:

```
sudo cp /mnt/casper/{vmlinuz,initrd} /srv/tftp/
```

### Set up the files for UEFI booting

1. Copy the signed shim binary into place:

```
apt download shim-signed
dpkg-deb --fsys-tarfile shim-signed*deb | tar x ./usr/lib/shim/shimx64.efi.signed.latest -O | sudo tee /srv/tftp/bootx64.efi >/dev/null
```

2. Copy the signed GRUB binary into place:

```
apt download grub-efi-amd64-signed
dpkg-deb --fsys-tarfile grub-efi-amd64-signed*deb | tar x ./usr/lib/grub/x86_64-efi-signed/grubnetx64.efi.signed -O | sudo tee /srv/tftp/grubx64.efi >/dev/null
```

3. GRUB also needs a font to be available over TFTP:

```
apt download grub-common
dpkg-deb --fsys-tarfile grub-common*deb | tar x ./usr/share/grub/unicode.pf2 -O | sudo tee /srv/tftp/unicode.pf2 >/dev/null
```

4. Create `/srv/tftp/grub/grub.cfg` that contains:

```
sudo mkdir -p /srv/tftp/grub/

sudo tee /srv/tftp/grub/grub.cfg >/dev/null <<EOF
set default="0"
set timeout=-1

if loadfont unicode ; then
   set gfxmode=auto
   set locale_dir=$prefix/locale
   set lang=en_US
fi
terminal_output gfxterm

set menu_color_normal=white/black
set menu_color_highlight=black/light-gray
if background_color 44,0,30; then
   clear
fi

function gfxmode {
   set gfxpayload="${1}"
   if [ "${1}" = "keep" ]; then
      set vt_handoff=vt.handoff=7
   else
      set vt_handoff=
   fi
}

set linux_gfx_mode=keep

export linux_gfx_mode

menuentry 'Ubuntu 24.04' {
   gfxmode $linux_gfx_mode
   linux /vmlinuz $vt_handoff quiet splash root=/dev/ram0 ramdisk_
size=1500000 cloud-config-url=/dev/null ip=dhcp url=http://cdimage.ubuntu.
com/ubuntu-server/noble/daily-live/current/noble-live-server-amd64.iso
   initrd /initrd
}
EOF
```

### Set up the files for legacy boot

1. Download `pxelinux.0` and put it into place:

```
wget http://archive.ubuntu.com/ubuntu/dists/focal/main/installer-amd64/
current/legacy-images/netboot/pxelinux.0
mkdir -p /srv/tftp
mv pxelinux.0 /srv/tftp/
```

2. Make sure to have installed package `syslinux-common` and then:

```
cp /usr/lib/syslinux/modules/bios/ldlinux.c32 /srv/tftp/
```

3. Create `/srv/tftp/pxelinux.cfg/default` containing:

```
 DEFAULT install
 LABEL install
   KERNEL vmlinuz
   INITRD initrd
   APPEND root=/dev/ram0 ramdisk_size=1500000 cloud-config-url=/dev/null
ip=dhcp url=http://cdimage.ubuntu.com/ubuntu-server/noble/daily-live/current/
noble-live-server-amd64.iso
```

> **Note**
>
> Setting `cloud-config-url=/dev/null` on the kernel command line prevents cloud-init from downloading the ISO twice.

As you can see, this downloads the ISO from Ubuntu's servers. You may want to host it somewhere on your infrastructure and change the URL to match.

This configuration is very simple. PXELINUX has many, many options, and you can consult its documentation for more.

### Netboot the server installer via UEFI PXE on ARM (aarch64, arm64) and x86_64 (amd64)

This document provides the steps needed to install a system via netbooting and the live server installer (Subiquity) in UEFI mode with Ubuntu 20.04 (or later).

The process described here is applicable to both arm64 and *amd*64 architectures. The process is inspired by this Ubuntu Discourse post for **legacy mode**, which is UEFI's predecessor. Focal (20.04, 20.04.5) and Groovy (20.10) have been tested with the following method.

### Configure TFTP

This article assumes that you have set up your TFTP (and/or *DHCP*/bootp if necessary, depending on your LAN configuration) by following the method described here. You could also build your own TFTP in this way if your *DNS* and DHCP are already well configured:

```
$ sudo apt install tftpd-hpa
```

If the installation is successful, check that the corresponding TFTP service is active using this command:

```
$ systemctl status tftpd-hpa.service
```

It is expected to show **active (running)** in the output messages. We will also assume your TFTP root path is `/var/lib/tftpboot` for the remainder of this article.

### How to serve files

You can **skip the whole section** of the following manual set up instruction by using this non-official tool. The tool will setup your TFTP server to serve necessary files for netbooting.

### Necessary files

The following files are needed for this process:

- Ubuntu live server image:
  - For arm64 architectures, the image name has the suffix `-arm64`. For example, `ubuntu-20.04.5-live-server-arm64.iso`.
  - For amd64 architectures, the image name has the suxxif `-amd64`. For example, `ubuntu-20.04.5-live-server-amd64.iso`.
- GRUB *EFI* binary (and the corresponding `grub.cfg` text file):
  - For arm64 architectures, this is called `grubnetaa64.efi.signed`.
  - For amd64 architectures, this is called `grubnetx64.efi.signed`.
- `initrd` extracted from your target Ubuntu live server image (use `hwe-initrd` instead if you want to boot with the HWE kernel).
- `vmlinuz` extracted from your target Ubuntu live server image (use `hwe-vmlinuz` instead if you want to boot with the HWE kernel).

### Examples

In the following sections, we will use an arm64 image as an example. This means the following files are used:

- Ubuntu 20.04.5 live server image ubuntu-20.04.5-live-server-arm64.iso
- GRUB EFI binary grubnetaa64.efi.signed
- `initrd` extracted from `ubuntu-20.04.5-live-server-arm64.iso`
- `vmlinuz` extracted from `ubuntu-20.04.5-live-server-arm64.iso`

Replace the corresponding files if you want to work on an amd64 image. For example, your files may be:

- Ubuntu 20.04.5 live server image ubuntu-20.04.5-live-server-amd64.iso
- GRUB EFI binary grubnetx64.efi.signed
- `initrd` extracted from `ubuntu-20.04.5-live-server-amd64.iso`
- `vmlinuz` extracted from `ubuntu-20.04.5-live-server-amd64.iso`

### Download and serve the GRUB EFI binary

The GRUB binary helps us redirect the download path to the target files via `grub.cfg`. Refer to the instructions here to get more information about the PXE process and why we need this binary.

```
$ sudo wget http://ports.ubuntu.com/ubuntu-ports/dists/focal/main/uefi/grub2-
arm64/current/grubnetaa64.efi.signed -O /var/lib/tftpboot/grubnetaa64.efi.signed
```

> **Note**
>
> You may need to change **the archive distribution's name** from `Focal` to your target distribution name.

### Download and serve more files

Fetch the installer by downloading an Ubuntu ARM server ISO, e.g. the 20.04.5 live server arm64 ISO. Note that the prefix "live" is significant. We will need the files available only in the live version.

Mount the ISO and copy the target files we need over to the TFTP folder:

```
$ sudo mount ./ubuntu-20.04.5-live-server-arm64.iso /mnt
$ sudo mkdir /var/lib/tftpboot/grub /var/lib/tftpboot/casper
$ sudo cp /mnt/boot/grub/grub.cfg /var/lib/tftpboot/grub/
$ sudo cp /mnt/casper/initrd /var/lib/tftpboot/casper/
$ sudo cp /mnt/casper/vmlinuz /var/lib/tftpboot/casper/
```

So, the TFTP root folder should look like this now:

```
$ find /var/lib/tftpboot/
/var/lib/tftpboot/
/var/lib/tftpboot/grub
/var/lib/tftpboot/grub/grub.cfg
/var/lib/tftpboot/grubnetaa64.efi.signed
/var/lib/tftpboot/casper
/var/lib/tftpboot/casper/initrd
/var/lib/tftpboot/casper/vmlinuz
```

Finally, let's customise the GRUB menu so we can install our target image by fetching it directly over the internet.

```
$ sudo chmod +w /var/lib/tftpboot/grub/grub.cfg
$ sudo vi /var/lib/tftpboot/grub/grub.cfg
```

Add a new entry:

```
menuentry "Install Ubuntu Server (Focal 20.04.5) (Pull the iso from web)" {
        set gfxpayload=keep
        linux   /casper/vmlinuz url=http://cdimage.ubuntu.com/ubuntu/releases/20.
04.5/release/ubuntu-20.04.5-live-server-arm64.iso only-ubiquity ip=dhcp ---
        initrd  /casper/initrd
}
```

Note that here:

- `ip=dhcp` is for the DHCP management setup in the lab.

- `url` is used to point to your target image download URL. Remember to change them according to your scenario.

If everything goes well, you should get into the expected GRUB menu of the ephemeral live prompt. Select the entry you just put in `grub.cfg`, which is `Install Ubuntu Server (Focal 20.04.5) (Pull the ISO from web)` in our example. Wait for the ISO to download, and then you will see the Subiquity welcome message. Enjoy the installation!

## Appendix

### Always check the serving file names

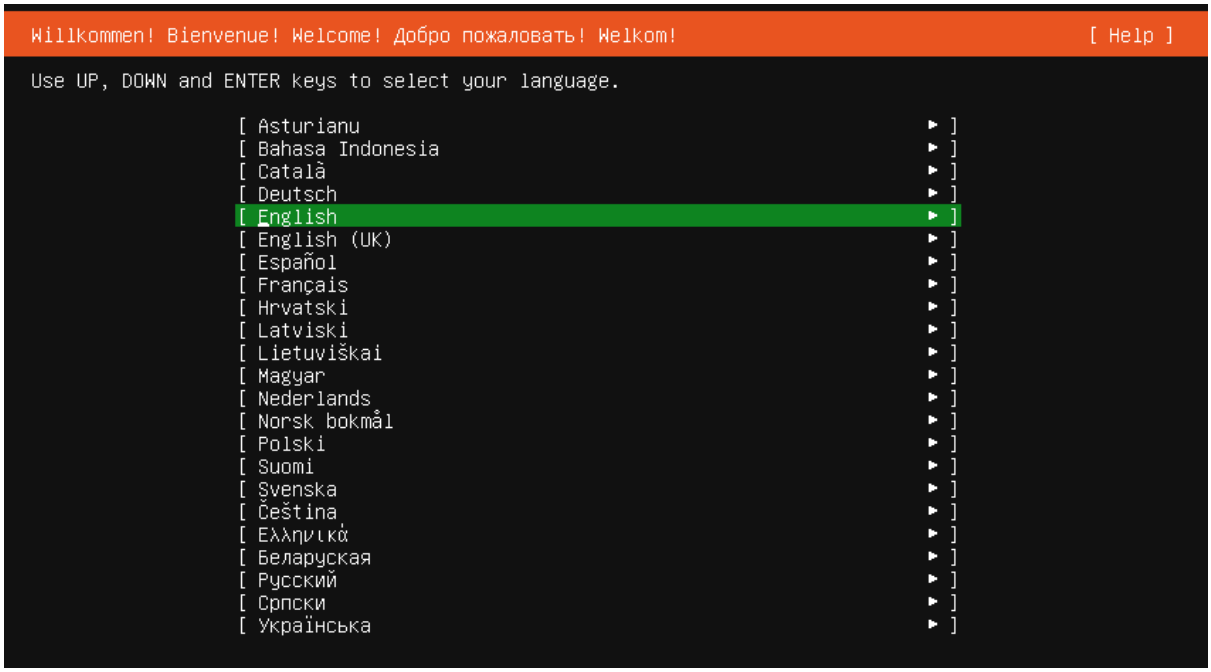
For example, always make sure the target file name for `linux` and `initrd` is correct. For example, the default `initrd` binary file name of 20.04.5 is `initrd`, and it is `initrd.lz` for 20.10. Always make sure you serve the correct file names, since this is a frequent troubleshooting issue. Paying attention to this detail could save you a lot of time.

### Booting screenshots

If your setup is correct, your `grub.cfg` should redirect the process to an ephemeral environment where your target image is downloaded and assigned in the GRUB entry of `grub.cfg`. You will see a screen like this if you are able to access the console or monitor device of your target machine:

```
Listening on LPF/ens3/02:56:e6:08:2d:e3
Sending on   LPF/ens3/02:56:e6:08:2d:e3
Sending on   Socket/fallback
DHCPDISCOVER on ens3 to 255.255.255.255 port 67 interval 3 (xid=0x83e74430)
DHCPOFFER of 10.228.68.73 from 10.228.68.5
DHCPREQUEST for 10.228.68.73 on ens3 to 255.255.255.255 port 67 (xid=0x3044e783)
DHCPACK of 10.228.68.73 from 10.228.68.5 (xid=0x83e74430)
bound to 10.228.68.73 -- renewal in 286 seconds.
Connecting to 10.228.68.91 (10.228.68.91:80)
ubuntu-20.04.1-live-   0% |                                | 462k  0:34:48 ETA
ubuntu-20.04.1-live-   2% |                                | 22.2M  0:01:20 ETA
ubuntu-20.04.1-live-   4% |*                               | 39.7M  0:01:05 ETA
ubuntu-20.04.1-live-   6% |**                              | 59.6M  0:00:57 ETA
ubuntu-20.04.1-live-   8% |**                              | 77.7M  0:00:53 ETA
ubuntu-20.04.1-live-  10% |***                             | 97.7M  0:00:50 ETA
ubuntu-20.04.1-live-  12% |***                             |  114M  0:00:49 ETA
ubuntu-20.04.1-live-  14% |****                            |  132M  0:00:47 ETA
ubuntu-20.04.1-live-  16% |*****                           |  149M  0:00:45 ETA
ubuntu-20.04.1-live-  18% |*****                           |  166M  0:00:45 ETA
ubuntu-20.04.1-live-  20% |******                          |  183M  0:00:43 ETA
ubuntu-20.04.1-live-  21% |*******                         |  200M  0:00:42 ETA
ubuntu-20.04.1-live-  23% |*******                         |  214M  0:00:42 ETA
```

Wait for the download to complete. If you see this Subiquity welcome page, the installer successfully launched via your UEFI PXE setup. Congratulations!




### Choose between the arm64 and arm64+largemem installer options

From 22.04.4 onwards, Ubuntu will provide both 4k and 64k page size kernel ISOs for ARM servers.

The default **arm64** ISO will still use a 4k page size kernel, while the new 64k page size kernel ISO is named **arm64+largemem**.

- arm64 4k ISO download
- arm64+largemem ISO download

### The default arm64 (4k) option

The 4k page size is the default in our arm64 ISO. It is suitable for workloads with many small processes, or environments with tight memory constraints. Typical use cases include (but are not limited to):

- Web servers
- Embedded devices
- General purpose/build systems

### The arm64+largemem (64k) option

Our new **arm64+largemem** ISO includes a kernel with 64k page size. A larger page size can increase throughput, but comes at the cost of increased memory use, making this option more suitable for servers with plenty of memory. Typical use cases for this ISO include:

- Machine learning
- Databases with many large entries
- *High performance computing*
- etc.

> **Note**
>
> It is possible to switch between these kernel options after installation by installing the other kernel alternative, rebooting, and selecting the new kernel from the GRUB menu.

### Switching kernels post-installation

To switch between the two kernels after the initial installation you can run the following commands, replacing `<desired-kernel>` with `linux-generic-64k` when swapping to 64k, or `linux-generic` when swapping to the default 4k kernel:

```
sudo apt update
sudo apt install <desired-kernel>
sudo reboot
```

Upon reboot you will be greeted with the GRUB menu (you may need to hold down the Shift key during the reboot for it to appear). Select “Advanced Options for Ubuntu”, then select your desired kernel to boot into Ubuntu.

To permanently change the default to your `<desired-flavour>`, replace `<desired-flavour>` with `generic` or `generic-64k` and then run the following command:

```
echo "GRUB_FLAVOUR_ORDER=<desired-flavour>" | sudo tee /etc/default/grub.d/local-order.cfg
```

To apply your change run:

```
sudo update-grub
```

Future boots will automatically use your new desired kernel flavour. You can verify this by rebooting using:

```
sudo reboot
```

And then running the following command to display the active kernel:

```
uname -r
```

### Netboot the live server installer on IBM Power (ppc64el) with Petitboot

Open a terminal window on your workstation and make sure the ‘ipmitool’ package is installed.

Verify if you can reach the BMC of the IBM Power system via `ipmitool` with a simple `ipmitool` call like:

```
$ ipmitool -I lanplus -H Power9Box -U <user> -P <password> power status
Chassis Power is off
```

or:

```
$ ipmitool -I lanplus -H Power9Box -U <user> -P <password> fru print 47
  Product Name          : OpenPOWER Firmware
  Product Version       : open-power-SUPERMICRO-P9DSU-V2.12-20190404-prod
  Product Extra         :         op-build-1b9269e
  Product Extra         :         buildroot-2018.11.3-12-g222837a
  Product Extra         :         skiboot-v6.0.19
  Product Extra         :         hostboot-c00d44a-pb1307d7
  Product Extra         :         occ-8fa3854
  Product Extra         :         linux-4.19.30-openpower1-p22d1df8
  Product Extra         :         petitboot-v1.7.5-p8f5fc86
```

or:

```
$ ipmitool -I lanplus -H Power9Box -U <user> -P <password> sol info
  Set in progress                 : set-complete
  Enabled                         : true
  Force Encryption                : false
  Force Authentication            : false
  Privilege Level                 : OPERATOR
  Character Accumulate Level (ms) : 0
  Character Send Threshold        : 0
  Retry Count                     : 0
  Retry Interval (ms)             : 0
  Volatile Bit Rate (kbps)        : 115.2
  Non-Volatile Bit Rate (kbps)    : 115.2
  Payload Channel                 : 1 (0x01)
  Payload Port                    : 623
```

### Activate serial-over-LAN

Open a second terminal and activate serial-over-LAN (SOL), so that you have two terminal windows open:

1) to control the BMC via IPMI
2) for the serial-over-LAN console

Activate serial-over-LAN:

```
$ ipmitool -I lanplus -H Power9Box -U <user> -P <password> sol activate
...
```

Power the system on in the 'control terminal' and watch the SOL console:

```
$ ipmitool -I lanplus -H Power9Box -U <user> -P <password> power on
...
```

It takes some time to see the first lines in the SOL console:

```
[SOL Session operational.  Use ~? for help]
--== Welcome to Hostboot  ==--
```

(continues on next page)

(continued from previous page)

```
  2.77131|secure|SecureROM valid - enabling functionality
  3.15860|secure|Booting in secure mode.
  5.59684|Booting from SBE side 0 on master proc=00050000
  5.60502|ISTEP  6. 5 - host_init_fsi
  5.87228|ISTEP  6. 6 - host_set_ipl_parms
  6.11032|ISTEP  6. 7 - host_discover_targets
  6.67868|HWAS|PRESENT> DIMM[03]=A0A0000000000000
  6.67870|HWAS|PRESENT> Proc[05]=8800000000000000
  6.67871|HWAS|PRESENT> Core[07]=3FFF0C33FFC30000
  6.98988|ISTEP  6. 8 - host_update_master_tpm
  7.22711|SECURE|Security Access Bit> 0xC000000000000000
  7.22711|SECURE|Secure Mode Disable (via Jumper)> 0x0000000000000000
  7.22731|ISTEP  6. 9 - host_gard
  7.43353|HWAS|FUNCTIONAL> DIMM[03]=A0A0000000000000
  7.43354|HWAS|FUNCTIONAL> Proc[05]=8800000000000000
  7.43356|HWAS|FUNCTIONAL> Core[07]=3FFF0C33FFC30000
  7.44509|ISTEP  6.10 - host_revert_sbe_mcs_setup
…
```

After a moment the system reaches the Petitboot screen:

```
Petitboot (v1.7.5-p8f5fc86)                                   9006-12P 1302NXA
  ──────────────────────────────────────────────────
 [Network: enP2p1s0f0 / 0c:c4:7a:87:04:d8]
   Execute
   netboot enP2p1s0f0 (pxelinux.0)
 [CD/DVD: sr0 / 2019-10-17-13-35-12-00]
   Install Ubuntu Server
 [Disk: sda2 / 295f571b-b731-4ebb-b752-60aadc80fc1b]
   Ubuntu, with Linux 5.4.0-14-generic (recovery mode)
   Ubuntu, with Linux 5.4.0-14-generic
   Ubuntu

 System information
 System configuration
 System status log
 Language
 Rescan devices
 Retrieve config from URL
 Plugins (0)

   *Exit to shell
   ──────────────────────────────────────────────────
Enter=accept, e=edit, n=new, x=exit, l=language, g=log, h=help
```

Select “*Exit to shell”.

**Note**

Make sure you really watch the SOL, since the Petitboot screen (above) has a time out (usu-

ally 10 or 30 seconds) and afterwards it automatically proceeds and tries to boot from the configured devices (usually disk). This can be prevented by just navigating in Petitboot. The petitboot shell is small Linux based OS:

```
...
Exiting petitboot. Type 'exit' to return.
You may run 'pb-sos' to gather diagnostic data
```

**Note**

In case one needs to gather system details and diagnostic data for IBM support, this can be done here by running 'pb-sos' (see `msg`).

### Download the ISO

Now download the 'live-server' ISO image (notice that 'focal-live-server-ppc64el.iso' uses the liver server installer, Subiquity, while 'focal-server-s390x.iso' uses `d-i`).

Again, for certain web locations a proxy needs to be used:

```
/ # export http_proxy=http://squid.proxy:3128   # in case a proxy is required
/ #
/ # wget http://cdimage.ubuntu.com/ubuntu-server/daily-live/current/focal-live-
server-ppc64el.iso
Connecting to <proxy-ip>:3128 (<proxy-ip>:3128)
focal-live-server-pp 100% |.....................|  922M  0:00:00 ETA
```

Next we need to loop-back mount the ISO:

```
/ # mkdir iso
/ # mount -o loop focal-live-server-ppc64el.iso iso
```

Or, in case autodetect of type iso9660 is not supported or not working, you should explicitly specify the 'iso9660' type:

```
/ # mount -t iso9660 -o loop focal-live-server-ppc64el.iso iso
```

Now load kernel and initrd from the loop-back mount, specify any needed kernel parameters and get it going:

```
/ # kexec -l ./iso/casper/vmlinux --initrd=./iso/casper/initrd.gz --append=
"ip=dhcp url=http://cdimage.ubuntu.com/ubuntu-server/daily-live/current/focal-
live-server-ppc64el.iso http_proxy=http://squid.proxy:3128 --- quiet"
/ # kexec -e
The system is going down NOW!
Sent SIGTERM to all processes
Sent SIGKILL to all processes
...
```

**Note**

In order to boot with and install the HWE kernel (if available), substitute `vmlinux` with `vmlinux-hwe` in the first `kexec` line.

The system now performs the initial boot of the installer:

```
[ 1200.687004] kexec_core: Starting new kernel
[ 1277.493883374,5] OPAL: Switch to big-endian OS
[ 1280.465061219,5] OPAL: Switch to little-endian OS
ln: /tmp/mountroot-fail-hooks.d//scripts/init-premount/lvm2: No such file or
directory
Internet Systems Consortium DHCP Client 4.4.1
Copyright 2004-2018 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/
Listening on LPF/enP2p1s0f3/0c:c4:7a:87:04:db
Sending on   LPF/enP2p1s0f3/0c:c4:7a:87:04:db
Listening on LPF/enP2p1s0f2/0c:c4:7a:87:04:da
Sending on   LPF/enP2p1s0f2/0c:c4:7a:87:04:da
Listening on LPF/enP2p1s0f1/0c:c4:7a:87:04:d9
Sending on   LPF/enP2p1s0f1/0c:c4:7a:87:04:d9
Listening on LPF/enP2p1s0f0/0c:c4:7a:87:04:d8
Sending on   LPF/enP2p1s0f0/0c:c4:7a:87:04:d8
Sending on   Socket/fallback
DHCPDISCOVER on enP2p1s0f3 to 255.255.255.255 port 67 interval 3
(xid=0x8d5704c)
DHCPDISCOVER on enP2p1s0f2 to 255.255.255.255 port 67 interval 3
(xid=0x94b25b28)
DHCPDISCOVER on enP2p1s0f1 to 255.255.255.255 port 67 interval 3
(xid=0x4edd0558)
DHCPDISCOVER on enP2p1s0f0 to 255.255.255.255 port 67 interval 3
(xid=0x61c90d28)
DHCPOFFER of 10.245.71.102 from 10.245.71.3
DHCPREQUEST for 10.245.71.102 on enP2p1s0f0 to 255.255.255.255 port 67
(xid=0x280dc961)
DHCPACK of 10.245.71.102 from 10.245.71.3 (xid=0x61c90d28)
bound to 10.245.71.102 -- renewal in 236 seconds.
Connecting to 91.189.89.11:3128 (91.189.89.11:3128)
focal-live-server-pp   1% |                                | 14.0M  0:01:04 ETA
focal-live-server-pp   4% |*                               | 45.1M  0:00:38 ETA
focal-live-server-pp   8% |**                              | 76.7M  0:00:33 ETA
focal-live-server-pp  11% |***                             |  105M  0:00:31 ETA
focal-live-server-pp  14% |****                            |  133M  0:00:29 ETA
focal-live-server-pp  17% |*****                           |  163M  0:00:27 ETA
focal-live-server-pp  20% |******                          |  190M  0:00:26 ETA
focal-live-server-pp  24% |*******                         |  222M  0:00:25 ETA
focal-live-server-pp  27% |********                        |  253M  0:00:23 ETA
focal-live-server-pp  30% |*********                       |  283M  0:00:22 ETA
focal-live-server-pp  34% |**********                      |  315M  0:00:21 ETA
```

(continues on next page)

(continued from previous page)

```
focal-live-server-pp  37% |***********                     |  343M  0:00:20 ETA
focal-live-server-pp  39% |************                    |  367M  0:00:19 ETA
focal-live-server-pp  42% |*************                   |  392M  0:00:18 ETA
focal-live-server-pp  45% |**************                  |  420M  0:00:17 ETA
focal-live-server-pp  48% |***************               |  451M  0:00:16 ETA
focal-live-server-pp  52% |****************              |  482M  0:00:15 ETA
focal-live-server-pp  55% |*****************             |  514M  0:00:14 ETA
focal-live-server-pp  59% |******************            |  546M  0:00:13 ETA
focal-live-server-pp  62% |*******************           |  578M  0:00:11 ETA
focal-live-server-pp  65% |********************          |  607M  0:00:10 ETA
focal-live-server-pp  69% |*********************         |  637M  0:00:09 ETA
focal-live-server-pp  72% |**********************        |  669M  0:00:08 ETA
focal-live-server-pp  75% |***********************       |  700M  0:00:07 ETA
focal-live-server-pp  79% |************************      |  729M  0:00:06 ETA
focal-live-server-pp  82% |*************************     |  758M  0:00:05 ETA
focal-live-server-pp  85% |**************************    |  789M  0:00:04 ETA
focal-live-server-pp  88% |***************************   |  817M  0:00:03 ETA
focal-live-server-pp  91% |****************************  |  842M  0:00:02 ETA
focal-live-server-pp  93% |***************************** |  867M  0:00:01 ETA
focal-live-server-pp  97% |******************************|  897M  0:00:00 ETA
focal-live-server-pp 100% |******************************|  922M  0:00:00 ETA
mount: mounting /cow on /root/cow failed: No such file or directory
Connecting to plymouth: Connection refused
passwd: password expiry information changed.
[   47.202736] /dev/loop3: Can't open blockdev
[   52.672550] cloud-init[3759]: Cloud-init v. 20.1-10-g71af48df-0ubuntu1 running
 'init-local' at Wed, 18 Mar 2020 15:18:07 +0000. Up 51.87 seconds.
...
```

Once it has completed, you will reach the initial Subiquity installer screen:

```
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
    Willkommen! Bienvenue! Welcome! Добро пожаловать! Welkom
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
    Use UP, DOWN and ENTER keys to select your language.

                  [ English                                        □ ]
                  [ Asturianu                                      □ ]
                  [ Català                                         □ ]
                  [ Hrvatski                                       □ ]
                  [ Nederlands                                     □ ]
                  [ Suomi                                          □ ]
                  [ Français                                       □ ]
                  [ Deutsch                                        □ ]
                  [ Ελληνικά                                       □ ]
                  [ Magyar                                         □ ]
                  [ Latviešu                                       □ ]
                  [ Norsk bokmål                                   □ ]
                  [ Polski                                         □ ]
```

(continues on next page)

(continued from previous page)

```
                    [ Русский                                  ]
                    [ Español                                  ]
                    [ Українська                               ]
```

From this point, you can follow the normal Subiquity installation. For more details, refer to the Subquity installer documentation.

### How to start a live server installation on IBM Power (ppc64el) with a virtual CD-ROM and Petitboot

> **Note**
>
> Not all IBM Power machines come with the capability of installing via a virtual CD-ROM! However, it is also possible to *boot the installer over the network*.

A separate system (ideally in the same network, because of `ipmitool`) is needed to host the ppc64el ISO image file, which is later used as the virtual CD-ROM.

#### Install ipmitool and Samba

Log in to this separate host and make sure that the `ipmitool` package is installed:

```
$ sudo apt install ipmitool
```

as well as Samba:

```
$ sudo apt install samba
```

#### Configure Samba

Next, set up and configure Samba:

```
$ sudo touch /etc/samba/smb.conf && sudo tee -a /etc/samba/smb.conf <<EOF
[winshare]
  path=/var/winshare
  browseable = yes
  read only = no
  guest ok = yes
EOF
```

And do a quick verification that the required lines are present:

```
$ tail -n 5 /etc/samba/smb.conf
[winshare]
  path=/var/winshare
  browseable = yes
  read only = no
  guest ok = yes
```

**Optional step** For downloading the image you may have to use a proxy server:

```
$ sudo touch ~/.wgetrc && sudo tee -a ~/.wgetrc <<EOF
use_proxy=yes
http_proxy=squid.proxy:3128
https_proxy=squid.proxy:3128
EOF
```

### Download the ISO image

The ISO image needs to be downloaded now:

```
$ wget http://cdimage.ubuntu.com/ubuntu/releases/focal/release/ubuntu-20.04-live-server-ppc64el.iso --directory-prefix=/var/winshare
```

The proxy can also be passed over as a `wget` argument, like this:

```
$ wget -e use_proxy=yes -e http_proxy=squid.proxy:3128 http://cdimage.ubuntu.com/ubuntu/releases/focal/release/ubuntu-20.04-live-server-ppc64el.iso --directory-prefix=/var/winshare
```

Change the file mode of the ISO image file:

```
$ sudo chmod -R 755 /var/winshare/
$ ls -l /var/winshare/
-rwxr-xr-x 1 ubuntu ubuntu  972500992 Mar 23 08:02 focal-live-server-ppc64el.iso
```

### Restart and check Samba

Next we need to restart and check the Samba service:

```
$ sudo service smbd restart
$ sudo service smbd status
● smbd.service - Samba SMB Daemon
   Loaded: loaded (/lib/systemd/system/smbd.service; enabled; vendor
preset: ena
   Active: active (running) since Tue 2020-02-04 15:17:12 UTC; 4s ago
     Docs: man:smbd(8)
           man:samba(7)
           man:smb.conf(5)
 Main PID: 6198 (smbd)
   Status: "smbd: ready to serve connections..."
    Tasks: 4 (limit: 19660)
   CGroup: /system.slice/smbd.service
           ├─6198 /usr/sbin/smbd --foreground --no-process-group
           ├─6214 /usr/sbin/smbd --foreground --no-process-group
           ├─6215 /usr/sbin/smbd --foreground --no-process-group
           └─6220 /usr/sbin/smbd --foreground --no-process-group
Feb 04 15:17:12 host systemd[1]: Starting Samba SMB Daemon…
Feb 04 15:17:12 host systemd[1]: Started Samba SMB Daemon.
```

Test Samba share:

```
ubuntu@host:~$ smbclient -L localhost
WARNING: The "syslog" option is deprecated
Enter WORKGROUP\ubuntu's password:
        Sharename       Type      Comment
        ---------       ----      -------
        print$          Disk      Printer Drivers
        winshare        Disk
        IPC$            IPC       IPC Service (host server (Samba, Ubuntu))
        Reconnecting with SMB1 for workgroup listing.
        Server               Comment
        ---------            -------
        Workgroup            Master
        ---------            -------
        WORKGROUP            host
```

Get the IP address of the Samba host:

```
$ ip -4 -brief address show
lo               UNKNOWN        127.0.0.1/8
ibmveth2         UNKNOWN        10.245.246.42/24
```

> **Optional step**: Additional testing to make certain the Samba share is accessible from remote:
>
> ```
> user@workstation:~$ mkdir -p /tmp/test
> user@workstation:~$ sudo mount -t cifs -o
> username=guest,password=guest //10.245.246.42/winshare /tmp/test/
> user@workstation:~$ ls -la /tmp/test/
> total 1014784
> drwxr-xr-x  2 root root          0 May  4 15:46 .
> drwxrwxrwt 18 root root        420 May  4 19:25 ..
> -rwxr-xr-x  1 root root 1038249984 May  3 19:37 ubuntu-20.04-live-
> server-ppc64el.iso
> ```

Now use a browser and navigate to the BMC of the Power system, which should be installed (let's assume the BMC's IP address is `10.245.246.247`):

```
firefox http://10.245.246.247/
```

Log into the BMC, then find and select:

> Virtual Media –> CD-ROM

Enter the IP address of the Samba share:

> 10.245.246.42

and the path to the Samba share:

```
\winshare\focal-live-server-ppc64el.iso
```

Click "Save and Mount". Make sure that the virtual CD-ROM is really properly mounted!

```
CD-ROM Image:

This option allows you to share a CD-ROM image over a Windows Share with a
maximum size of 4.7GB. This image will be emulated to the host as USB device.

Device 1        There is an iso file mounted.
Device 2        No disk emulation set.
Device 3        No disk emulation set.
<Refresh Status>
```

```
Share host: 10.245.246.42
Path to image: \winshare\focal-live-server-ppc64el.iso
User (optional):
Password (optional):
<Save> <Mount> <Unmount>
```

**Note**

It's important that you see a status like this:

```
Device 1 There is an iso file mounted
```

Then the virtual CD-ROM is properly mounted and you will see the boot/install from CD-ROM entry in Petitboot:

```
[CD/DVD: sr0 / 2020-03-23-08-02-42-00]
  Install Ubuntu Server
```

### Boot into the Petitboot loader

Now use `ipmitool` to boot the system into the Petitboot loader:

```
$ ipmitool -I lanplus -H 10.245.246.247 -U ADMIN -P <password> power status
$ ipmitool -I lanplus -H 10.245.246.247 -U ADMIN -P <password> sol activate
$ ipmitool -I lanplus -H 10.245.246.247 -U ADMIN -P <password> power on
Chassis Power Control: Up/On
```

And reach the Petitboot screen:

```
Petitboot (v1.7.5-p8f5fc86)                                    9006-12C BOS0026
 ──────────────────────────────────────────────
  [Network: enP2p1s0f0 / ac:1f:6b:09:c0:52]
    execute
    netboot enP2p1s0f0 (pxelinux.0)

  System information
  System configuration
  System status log
  Language
  Rescan devices
```

(continues on next page)

(continued from previous page)

```
 Retrieve config from URL
*Plugins (0)
 Exit to shell
_____________________________________________

 Enter=accept, e=edit, n=new, x=exit, l=language, g=log, h=help
 Default boot cancelled
```

Make sure that booting from CD-ROM is enabled:

```
Petitboot (v1.7.5-p8f5fc86)                                    9006-12C BOS0026
_____________________________________________

  [Network: enP2p1s0f0 / ac:1f:6b:09:c0:52]
    Execute
    netboot enP2p1s0f0 (pxelinux.0)
  [Disk: sda2 / ebdb022b-96b2-4f4f-ae63-69300ded13f4]
    Ubuntu, with Linux 5.4.0-12-generic (recovery mode)
    Ubuntu, with Linux 5.4.0-12-generic
    Ubuntu

  System information
  System configuration
  System status log
  Language
  Rescan devices
  Retrieve config from URL
 *Plugins (0)
  Exit to shell

_____________________________________________

 Enter=accept, e=edit, n=new, x=exit, l=language, g=log, h=help
 [sda3] Processing new Disk device
```

```
Petitboot System Configuration

_____________________________________________

  Autoboot:      ( ) Disabled
                 (*) Enabled

  Boot Order:    (0) Any CD/DVD device
                 (1) disk: sda2 [uuid: ebdb022b-96b2-4f4f-ae63-69300ded13f4]
                 (2) net:  enP2p1s0f0 [mac: ac:1f:6b:09:c0:52]

                 [     Add Device     ]
                 [  Clear & Boot Any  ]
                 [       Clear        ]

  Timeout:       30    seconds
```

(continues on next page)

(continued from previous page)

```
  Network:       (*) DHCP on all active interfaces
                 ( ) DHCP on a specific interface
                 ( ) Static IP configuration


_____________________________________________
 tab=next, shift+tab=previous, x=exit, h=help
```

```
Petitboot System Configuration
_____________________________________________
  Network:       (*) DHCP on all active interfaces
                 ( ) DHCP on a specific interface
                 ( ) Static IP configuration

  DNS Server(s):                                  (eg. 192.168.0.2)
                 (if not provided by DHCP server)
  HTTP Proxy:
  HTTPS Proxy:

  Disk R/W:      ( ) Prevent all writes to disk
                 (*) Allow bootloader scripts to modify disks

  Boot console:  (*) /dev/hvc0 [IPMI / Serial]
                 ( ) /dev/tty1 [VGA]
                 Current interface: /dev/hvc0

                 [    OK    ]  [   Help   ]  [  Cancel  ]

_____________________________________________
 tab=next, shift+tab=previous, x=exit, h=help
```

Now select the 'Install Ubuntu Server' entry below the CD/DVD entry:

```
  [CD/DVD: sr0 / 2020-03-23-08-02-42-00]
  *  Install Ubuntu Server
```

And let Petitboot boot from the (virtual) CD-ROM image:

```
Sent SIGKILL to all processes
[  119.355371] kexec_core: Starting new kernel
[  194.483947394,5] OPAL: Switch to big-endian OS
[  197.454615202,5] OPAL: Switch to little-endian OS
```

The initial Subiquity installer screen will show up in the console:

```
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
  Willkommen! Bienvenue! Welcome! Добро пожаловать! Welkom
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
  Use UP, DOWN and ENTER keys to select your language.
```

(continues on next page)

(continued from previous page)

```
                [ English                                        ]
                [ Asturianu                                      ]
                [ Català                                         ]
                [ Hrvatski                                       ]
                [ Nederlands                                     ]
                [ Suomi                                          ]
                [ Français                                       ]
                [ Deutsch                                        ]
                [ Ελληνικά                                       ]
                [ Magyar                                         ]
                [ Latviešu                                       ]
                [ Norsk bokmål                                   ]
                [ Polski                                         ]
                [ Русский                                        ]
                [ Español                                        ]
                [ Українська                                     ]
```

From this point, you can follow the normal Subiquity installation. For more details, refer to the Subiquity installer documentation.

### Interactive live server installation on IBM z/VM (s390x)

Doing an interactive (manual) live installation as described here - meaning without specifying a parmfile - has been supported in Ubuntu Server since LTS 20.04.5 ('Focal').

The following guide assumes that a z/VM guest has been defined, and that it is able to either reach the public `cdimage.ubuntu.com` server or an internal FTP or HTTP server that hosts an Ubuntu Server 20.04 installer image, like this 20.04 (a.k.a. Focal) daily live image

Find a place to download the installer image:

```
user@workstation:~$ wget
http://cdimage.ubuntu.com/ubuntu/releases/20.04.5/release/ubuntu-20.04.5-live-
server-s390x.iso
--2020-08-08 16:01:52--
http://cdimage.ubuntu.com/ubuntu/releases/20.04.5/release/ubuntu-20.04.5-live-
server-s390x.iso
Resolving cdimage.ubuntu.com (cdimage.ubuntu.com)... 2001:67c:1560:8001::1d,
2001:67c:1360:8001::27, 2001:67c:1360:8001::28, ...
Connecting to cdimage.ubuntu.com
(cdimage.ubuntu.com)|2001:67c:1560:8001::1d|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 705628160 (673M) [application/x-iso9660-image]
Saving to: 'ubuntu-20.04.5-live-server-s390x.iso'

ubuntu-20.04.5-live 100%[===================>] 672.94M  37.1MB/s    in
17s

2020-08-08 16:02:10 (38.8 MB/s) - 'ubuntu-20.04.5-live-server-s390x.iso' saved
[705628160/705628160]
```

Now loop-back mount the ISO to extract four files that are needed for a z/VM guest installation:

```
user@workstation:~$ mkdir iso
user@workstation:~$ sudo mount -o loop ubuntu-20.04.5-live-server-s390x.iso iso
user@workstation:~$

user@workstation:~$ ls -1 ./iso/boot/{ubuntu.exec,parmfile.*,kernel.u*,initrd.u*}
./iso/boot/initrd.ubuntu
./iso/boot/kernel.ubuntu
./iso/boot/parmfile.ubuntu
./iso/boot/ubuntu.exec
```

Now transfer these four files to your z/VM guest (for example to its 'A' file mode), using either the 3270 terminal emulator or FTP.

Then log on to the z/VM guest that you want to use for the installation. In this example it will be guest '10.222.111.24'.

Execute the `ubuntu` REXX script to kick-off the installation:

```
ubuntu
00: 0000004 FILES PURGED
00: RDR FILE 0125 SENT FROM 10.222.111.24  PUN WAS 0125 RECS 101K CPY  001 A
NOHOLD NO
KEEP
00: RDR FILE 0129 SENT FROM 10.222.111.24  PUN WAS 0129 RECS 0001 CPY  001 A
NOHOLD NO
KEEP
00: RDR FILE 0133 SENT FROM 10.222.111.24  PUN WAS 0133 RECS 334K CPY  001 A
NOHOLD NO
KEEP
00: 0000003 FILES CHANGED
00: 0000003 FILES CHANGED
01: HCPGSP2627I The virtual machine is placed in CP mode due to a SIGP initial CPU
reset from CPU 00.
02: HCPGSP2627I The virtual machine is placed in CP mode due to a SIGP initial CPU
reset from CPU 00.
03: HCPGSP2627I The virtual machine is placed in CP mode due to a SIGP initial CPU
reset from CPU 00.
¬    0.390935| Initramfs unpacking failed: Decoding failed
Unable to find a medium container a live file system
```

In the usual case that no parmfile was configured, the installation system now offers to interactively configure the basic network:

```
Attempt interactive netboot from a URL?
yes no (default yes): yes
Available qeth devices:
0.0.0600 0.0.0603
zdev to activate (comma separated, optional): 0600
QETH device 0.0.0600:0.0.0601:0.0.0602 configured
```

(continues on next page)

(continued from previous page)

```
Two methods available for IP configuration:
  * static: for static IP configuration
  * dhcp: for automatic IP configuration
static dhcp (default 'dhcp'): static
ip: 10.222.111.24
gateway (default 10.222.111.1): .
dns (default .):
vlan id (optional):
 http://cdimage.ubuntu.com/ubuntu/releases/20.04.5/release/ubuntu-20.04.5-live-
server-s390x.iso (default)
url: ftp://10.11.12.2:21/ubuntu-live-server-20.04.5/ubuntu-20.04.5-live-server-
s390x.iso
http_proxy (optional):
```

Ensure that the same version of the ISO image that was used to extract the installer files – kernel and initrd – is referenced at the '`url:`' setting. It can be at a different location, for example directly referencing the public `cdimage.ubuntu.com` server: `http://cdimage.ubuntu.com/ubuntu/releases/20.04.5/release/ubuntu-20.04.5-live-server-s390x.iso`

The boot-up of the live-server installation now completes:

```
Configuring networking...
QETH device 0.0.0600:0.0.0601:0.0.0602 already configured
IP-Config: enc600 hardware address 02:28:0a:00:00:39 mtu 1500
IP-Config: enc600 guessed broadcast address 10.222.111255
IP-Config: enc600 complete:
 address: 10.222.111.24    broadcast: 10.222.111255   netmask: 255.255.255.0

 gateway: 10.222.111.1     dns0     : 10.222.111.1     dns1   : 0.0.0.0

 rootserver: 0.0.0.0 rootpath:
 filename  :
Connecting to 10.11.12.2:21 (10.11.12.2:21)
focal-live-server-s   5% !*                                ! 35.9M  0:00:17 ETA
focal-live-server-s  19% !******                           !  129M  0:00:08 ETA
focal-live-server-s  33% !**********                       !  225M  0:00:05 ETA
focal-live-server-s  49% !***************                  !  330M  0:00:04 ETA
focal-live-server-s  60% !******************               !  403M  0:00:03 ETA
focal-live-server-s  76% !************************         !  506M  0:00:01 ETA
focal-live-server-s  89% !****************************     !  594M  0:00:00 ETA
focal-live-server-s 100% !********************************!  663M  0:00:00 ETA
passwd: password expiry information changed.
QETH device 0.0.0600:0.0.0601:0.0.0602 already configured
no search or nameservers found in /run/net-enc600.conf /run/net-*.conf /run/net6
-*.conf
¬  594.766372| /dev/loop3: Can't open blockdev
¬  595.610434| systemd¬1|: multi-user.target: Job getty.target/start deleted to
break ordering cycle starting with multi-user.target/start
¬ ¬0;1;31m SKIP  ¬0m| Ordering cycle found, skipping  ¬0;1;39mLogin Prompts ¬0m
¬  595.623027| systemd¬1|: Failed unmounting /cdrom.
```

(continues on next page)

(continued from previous page)

```
¬ ¬0;1;31mFAILED ¬0m| Failed unmounting  ¬0;1;39m/cdrom ¬0m.

¬  598.973538| cloud-init¬1256|: Cloud-init v. 20.2-45-g5f7825e2-0ubuntu1 runnin
g 'init-local' at Thu, 04 Jun 2020 12:06:46 +0000. Up 598.72 seconds.
¬  599.829069| cloud-init¬1288|: Cloud-init v. 20.2-45-g5f7825e2-0ubuntu1 runnin
g 'init' at Thu, 04 Jun 2020 12:06:47 +0000. Up 599.64 seconds.
¬  599.829182| cloud-init¬1288|: ci-info: ++++++++++++++++++++++++++++++++++++Ne
t device info++++++++++++++++++++++++++++++++++
¬  599.829218| cloud-init¬1288|: ci-info: +--------+------+--------------------
----+---------------+--------+-------------------+
¬  599.829255| cloud-init¬1288|: ci-info: ! Device !  Up  !         Address
    !      Mask     ! Scope  !     Hw-Address    !
¬  599.829292| cloud-init¬1288|: ci-info: +--------+------+--------------------
----+---------------+--------+-------------------+
¬  599.829333| cloud-init¬1288|: ci-info: ! enc600 ! True !      10.222.111.24
    ! 255.255.255.0 ! global ! 02:28:0a:00:00:39 !
¬  599.829376| cloud-init¬1288|: ci-info: ! enc600 ! True ! fe80::28:aff:fe00:3
/64 !       .       !  link  ! 02:28:0a:00:00:39 !
¬  599.829416| cloud-init¬1288|: ci-info: !   lo   ! True !        127.0.0.1
    !   255.0.0.0   !  host  !         .         !
¬  599.829606| cloud-init¬1288|: ci-info: !   lo   ! True !         ::1/128
    !       .       !  host  !         .         !
¬  599.829684| cloud-init¬1288|: ci-info: +--------+------+--------------------
----+---------------+--------+-------------------+
¬  599.829721| cloud-init¬1288|: ci-info: +++++++++++++++++++++++++++++Route IP
v4 info+++++++++++++++++++++++++++++
¬  599.829754| cloud-init¬1288|: ci-info: +-------+-------------+-------------
+---------------+-----------+-------+
¬  599.829789| cloud-init¬1288|: ci-info: ! Route ! Destination  !   Gateway
!    Genmask    ! Interface ! Flags !
¬  599.829822| cloud-init¬1288|: ci-info: +-------+-------------+-------------
+---------------+-----------+-------+
¬  599.829858| cloud-init¬1288|: ci-info: !   0   !   0.0.0.0    ! 10.222.111.1
!    0.0.0.0    !   enc600  !   UG  !
¬  599.829896| cloud-init¬1288|: ci-info: !   1   ! 10.222.1110 !   0.0.0.0
! 255.255.255.0 !   enc600  !   U   !
¬  599.829930| cloud-init¬1288|: ci-info: +-------+-------------+-------------
+---------------+-----------+-------+
¬  599.829962| cloud-init¬1288|: ci-info: +++++++++++++++++++Route IPv6 info++++
+++++++++++++++
¬  599.829998| cloud-init¬1288|: ci-info: +-------+-------------+---------+-----
------+-------+
¬  599.830031| cloud-init¬1288|: ci-info: ! Route ! Destination ! Gateway ! Inte
rface ! Flags !
¬  599.830064| cloud-init¬1288|: ci-info: +-------+-------------+---------+-----
------+-------+
¬  599.830096| cloud-init¬1288|: ci-info: !   1   !  fe80::/64  !    ::   !   en
c600  !   U   !
¬  599.830131| cloud-init¬1288|: ci-info: !   3   !    local    !    ::   !   en
```

(continues on next page)

(continued from previous page)

```
c600  !   U   !
¬  599.830164| cloud-init¬1288|: ci-info: !   4   !   ff00::/8  !    ::   !   en
c600  !   U   !
¬  599.830212| cloud-init¬1288|: ci-info: +-------+-------------+---------+-----
------+-------+
¬  601.077953| cloud-init¬1288|: Generating public/private rsa key pair.
¬  601.078101| cloud-init¬1288|: Your identification has been saved in /etc/ssh/
ssh_host_rsa_key
¬  601.078136| cloud-init¬1288|: Your public key has been saved in /etc/ssh/ssh
host_rsa_key.pub
¬  601.078170| cloud-init¬1288|: The key fingerprint is:
¬  601.078203| cloud-init¬1288|: SHA256:kHtkABZwk8AE80fy0KPzTRcYpht4iXdZmJ37Cgi3
fJ0 root§ubuntu-server
¬  601.078236| cloud-init¬1288|: The key's randomart image is:
¬  601.078274| cloud-init¬1288|: +---¬RSA 3072|----+
¬  601.078307| cloud-init¬1288|: !o+*+B++*..       !
¬  601.078340| cloud-init¬1288|: ! o.X+=+=+        !
¬  601.078373| cloud-init¬1288|: !  +.O.= oo       !
¬  601.078406| cloud-init¬1288|: !  ++.+.=o        !
¬  601.078439| cloud-init¬1288|: !   *.=.oSo       !
¬  601.078471| cloud-init¬1288|: !    = +.E .      !
¬  601.078503| cloud-init¬1288|: !     . . .       !
¬  601.078537| cloud-init¬1288|: !        .        !
¬  601.078570| cloud-init¬1288|: !                 !
¬  601.078602| cloud-init¬1288|: +----¬SHA256|-----+
¬  601.078635| cloud-init¬1288|: Generating public/private dsa key pair.
¬  601.078671| cloud-init¬1288|: Your identification has been saved in /etc/ssh/
ssh_host_dsa_key
¬  601.078704| cloud-init¬1288|: Your public key has been saved in /etc/ssh/ssh_
host_dsa_key.pub
¬  601.078736| cloud-init¬1288|: The key fingerprint is:
¬  601.078767| cloud-init¬1288|: SHA256:ZBNyksVVYZVhKJeL+PWKpsdUcn21yiceX/DboXQd
Pq0 root§ubuntu-server
¬  601.078800| cloud-init¬1288|: The key's randomart image is:
¬  601.078835| cloud-init¬1288|: +---¬DSA 1024|----+
¬  601.078867| cloud-init¬1288|: !      o++...+=+o !
¬  601.078899| cloud-init¬1288|: !      .+....+.. .!
¬  601.078932| cloud-init¬1288|: !        +. + o  o!
¬  601.078964| cloud-init¬1288|: !       o..o = oo.!
¬  601.078996| cloud-init¬1288|: !        S. =..o++!
¬  601.079029| cloud-init¬1288|: !          o  *.*=!
¬  601.079061| cloud-init¬1288|: !         o .o.B.*!
¬  601.079094| cloud-init¬1288|: !          = .oEo.!
¬  601.079135| cloud-init¬1288|: !         .+       !
¬  601.079167| cloud-init¬1288|: +----¬SHA256|-----+
¬  601.079199| cloud-init¬1288|: Generating public/private ecdsa key pair.
¬  601.079231| cloud-init¬1288|: Your identification has been saved in /etc/ssh/
ssh_host_ecdsa_key
¬  601.079263| cloud-init¬1288|: Your public key has been saved in /etc/ssh/ssh_
```

(continues on next page)

(continued from previous page)

```
host_ecdsa_key.pub
¬  601.079295| cloud-init¬1288|: The key fingerprint is:
¬  601.079327| cloud-init¬1288|: SHA256:Bitar9fVHUH2FnYVSJJnldprdAcM5Est0dmRWFTU
i8k root§ubuntu-server
¬  601.079362| cloud-init¬1288|: The key's randomart image is:
¬  601.079394| cloud-init¬1288|: +---¬ECDSA 256|---+
¬  601.079426| cloud-init¬1288|: !           o**O%&!
¬  601.079458| cloud-init¬1288|: !           o.OB+=!
¬  601.079491| cloud-init¬1288|: !      .     B *o+!
¬  601.079525| cloud-init¬1288|: !       o   . E.=o!
¬  601.079557| cloud-init¬1288|: !    o . S  .....+!
¬  601.079589| cloud-init¬1288|: !   o o .  . . .o !
¬  601.079621| cloud-init¬1288|: !  .   .. .    .  !
¬  601.079653| cloud-init¬1288|: !     .. .        !
¬  601.079685| cloud-init¬1288|: !    ..           !
¬  601.079717| cloud-init¬1288|: +----¬SHA256|-----+
¬  601.079748| cloud-init¬1288|: Generating public/private ed25519 key pair.
¬  601.079782| cloud-init¬1288|: Your identification has been saved in /etc/ssh/
ssh_host_ed25519_key
¬  601.079814| cloud-init¬1288|: Your public key has been saved in /etc/ssh/ssh_
host_ed25519_key.pub
¬  601.079847| cloud-init¬1288|: The key fingerprint is:
¬  601.079879| cloud-init¬1288|: SHA256:yWsZ/5+7u7D3SIcd7HYnyajXyeWnt5nQ+ZI3So3b
eN8 root§ubuntu-server
¬  601.079911| cloud-init¬1288|: The key's randomart image is:
¬  601.079942| cloud-init¬1288|: +--¬ED25519 256|--+
¬  601.079974| cloud-init¬1288|: !                 !
¬  601.080010| cloud-init¬1288|: !                 !
¬  601.080042| cloud-init¬1288|: !                 !
¬  601.080076| cloud-init¬1288|: !       . .    .  !
¬  601.080107| cloud-init¬1288|: !        S      o !
¬  601.080139| cloud-init¬1288|: !         =   o=++!
¬  601.080179| cloud-init¬1288|: !        + . o**§=!
¬  601.080210| cloud-init¬1288|: !       .   oo+&B%!
¬  601.080244| cloud-init¬1288|: !          ..o*%/E!
¬  601.080289| cloud-init¬1288|: +----¬SHA256|-----+
¬  612.293731| cloud-init¬2027|: Cloud-init v. 20.2-45-g5f7825e2-0ubuntu1 runnin
g 'modules:config' at Thu, 04 Jun 2020 12:06:59 +0000. Up 612.11 seconds.
¬  612.293866| cloud-init¬2027|: Set the following 'random' passwords
¬  612.293940| cloud-init¬2027|: installer:wgYsAPzYQbFYqU2X2hYm
ci-info: no authorized SSH keys fingerprints found for user installer.
<14>Jun  4 12:07:00 ec2:
<14>Jun  4 12:07:00 ec2: ##########################################################
######
<14>Jun  4 12:07:00 ec2: -----BEGIN SSH HOST KEY FINGERPRINTS-----
<14>Jun  4 12:07:00 ec2: 1024 SHA256:ZBNyksVVYZVhKJeL+PWKpsdUcn21yiceX/DboXQdPq0
 root§ubuntu-server (DSA)
<14>Jun  4 12:07:00 ec2: 256 SHA256:Bitar9fVHUH2FnYVSJJnldprdAcM5Est0dmRWFTUi8k
root§ubuntu-server (ECDSA)
```

(continues on next page)

(continued from previous page)

```
<14>Jun  4 12:07:00 ec2: 256 SHA256:yWsZ/5+7u7D3SIcd7HYnyajXyeWnt5nQ+ZI3So3beN8
root§ubuntu-server (ED25519)
<14>Jun  4 12:07:00 ec2: 3072 SHA256:kHtkABZwk8AE80fy0KPzTRcYpht4iXdZmJ37Cgi3fJ0
 root§ubuntu-server (RSA)
<14>Jun  4 12:07:00 ec2: -----END SSH HOST KEY FINGERPRINTS-----
<14>Jun  4 12:07:00 ec2: ############################################################
######
-----BEGIN SSH HOST KEY KEYS-----
ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBIXM6t1/
35ot/aPI59ThIJBzg+qGJJ17+1ZVHfzMEDbsTwpM7e9pstPZUM7W1IHWqDvLQDBm/hGg4u8ZGEqmIMI=
 root§ubuntu-server
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIN7QtU+en+RGruj2zuxWgkMqLmh+35/GR/OEOD16k4nA
 root§ubuntu-server
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDJdKT7iUAvSjkUqI1l3fHysE+Gj7ulwGgGjYh639px
kcHEbbS3V48eROY9BmDISEHfjYXGY2wEH0tGJjNRROGJhZJVNR+qAqJBioj9d/TwXEgwLP8eAy9aVtJB
K1rIylnMQltx/SIhgiymjHLCtKlVoIS4l0frT9FiF54Qi/JeJlwGJIW3W2XgcY9ODT0Q5g3PSmlZ8KTR
imTf9Fy7WJEPA08b3fimYWsz9enuS/gECEUGV3M1MvrzpAQju27NUEOpSMZHR62IMxGvIjYIu3dUkAzm
MBdwxHdLMQ8rI8PehyHDiFr6g2Ifxoy5QLmb3hISKlq/R6pLLeXbb748gN2i8WCvK0AEGfa/kJDW3RNU
VYd+ACBBzyhVbiw7W1CQW/ohik3wyosUyi9nJq2IqOA7kkGH+1XoYq/e4/MoqxhIK/oaiudYAkaCWmP1
r/fBa3hlf0f7mVHvxA3tWZc2wYUxFPTmePvpydP2PSctHMhgboaHrGIY2CdSqg8SUdPKrOE= root§ub
untu-server
-----END SSH HOST KEY KEYS-----
¬  612.877357| cloud-init¬2045|: Cloud-init v. 20.2-45-g5f7825e2-0ubuntu1 runnin
g 'modules:final' at Thu, 04 Jun 2020 12:07:00 +0000. Up 612.79 seconds.
¬  612.877426| cloud-init¬2045|: ci-info: no authorized SSH keys fingerprints fo
und for user installer.
¬  612.877468| cloud-init¬2045|: Cloud-init v. 20.2-45-g5f7825e2-0ubuntu1 finish
ed at Thu, 04 Jun 2020 12:07:00 +0000. Datasource DataSourceNoCloud ¬seed=/var/l
ib/cloud/seed/nocloud|¬dsmode=net|.  Up 612.87 seconds
¬  612.877509| cloud-init¬2045|: Welcome to Ubuntu Server InstallerÜ
¬  612.877551| cloud-init¬2045|: Above you will find SSH host keys and a random
password set for the `installer` user. You can use these credentials to ssh-in a
nd complete the installation. If you provided SSH keys in the cloud-init datasou
rce, they were also provisioned to the installer user.
¬  612.877634| cloud-init¬2045|: If you have access to the graphical console, li
ke TTY1 or HMC ASCII terminal you can complete the installation there too.

It is possible to connect to the installer over the network, which
might allow the use of a more capable terminal.

To connect, SSH to installer§10.222.111.24.

The password you should use is "KRuXtz5dURAyPkcjcUvA".

The host key fingerprints are:

    RSA     SHA256:3IvYMkU05lQSKBxOVZUJMzdtXpz3RJl3dEQsg3UWc54
    ECDSA   SHA256:xd1xnkBpn49DUbuP8uWro2mu1GM4MtnqR2WEWg1fS3o
    ED25519 SHA256:Hk3+/4+X7NJBHl6/e/6xFhNXsbHBsOvt6i8YEFUepko
```

(continues on next page)

(continued from previous page)

```
Ubuntu Focal Fossa (development branch) ubuntu-server ttyS0
```

The next step is to remotely connect to the install system and to proceed with the Subiquity installer.

Notice that at the end of the installer boot-up process, all necessary data is provided to proceed with running the installer in a remote SSH shell. The command to execute locally is:

```
user@workstation:~$ ssh installer@10.222.111.24
```

A temporary random password for the installation was created and shared as well, which should be used without the leading and trailing double quotes:

```
  "KRuXtz5dURAyPkcjcUvA"
```

```
user@workstation:~$ ssh installer@10.222.111.24
The authenticity of host '10.222.111.24 (10.222.111.24)' can't be established.
ECDSA key fingerprint is
SHA256:xd1xnkBpn49DUbuP8uWro2mu1GM4MtnqR2WEWg1fS3o.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.222.111.24' (ECDSA) to the list of known hosts.
installer@10.222.111.24's password: KRuXtz5dURAyPkcjcUvA
```

You may now temporarily see some login messages like these:

```
Welcome to Ubuntu Focal Fossa (development branch) (GNU/Linux 5.4.0-42-generic
s390x)

  * Documentation:  https://help.ubuntu.com
  * Management:     https://landscape.canonical.com
  * Support:        https://ubuntu.com/pro

  System information as of Wed Jun  3 17:32:10 UTC 2020

  System load:    0.0       Memory usage: 2%   Processes:       146
  Usage of /home: unknown   Swap usage:   0%   Users logged in: 0

0 updates can be installed immediately.
0 of these updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable
law.
```

Eventually, the initial Subiquity installer screen appears:

```
================================================================================
  Willkommen! Bienvenue! Welcome! ????? ??????????! Welkom!
================================================================================
  Use UP, DOWN and ENTER keys to select your language.

                [ English                                       > ]
                [ Asturianu                                     > ]
                [ Cataln                                        > ]
                [ Hrvatski                                      > ]
                [ Nederlands                                    > ]
                [ Suomi                                         > ]
                [ Francais                                      > ]
                [ Deutsch                                       > ]
                [ Magyar                                        > ]
                [ Latvie?u                                      > ]
                [ Norsk bokm?l                                  > ]
                [ Polski                                        > ]
                [ Espanol                                       > ]
```

From this point, you can follow the normal Subiquity installation. For more details, refer to the Subquity installer documentation.

```
================================================================================
  Keyboard configuration
================================================================================
  Please select your keyboard layout below, or select "Identify keyboard" to
  detect your layout automatically.

                Layout:  [ English (US)                    v ]


               Variant:  [ English (US)                    v ]


                            [ Identify keyboard ]









                                  [ Done       ]
                                  [ Back       ]
```

```
================================================================================
  Zdev setup
================================================================================
  ID                           ONLINE  NAMES                                  ^
```

(continues on next page)

(continued from previous page)

```
                                                                    |
  generic-ccw                                                       |
  0.0.0009                                  >                       |
  0.0.000c                                  >                       |
  0.0.000d                                  >                       |
  0.0.000e                                  >                       |
                                                                    |
  dasd-eckd                                                         |
  0.0.0190                                  >                       |
  0.0.0191                                  >                       |
  0.0.019d                                  >                       |
  0.0.019e                                  >
  0.0.0200                                  >
  0.0.0300                                  >
  0.0.0400                                  >
  0.0.0592                                  >                     v

                              [ Continue    ]
                              [ Back        ]
```

- If the list is long, hit the End key that will automatically scroll you down to the bottom of the Z devices list and screen.

```
=====================================================================
  Zdev setup
=====================================================================
  ID                           ONLINE  NAMES                        ^
                                                                    |
  generic-ccw                                                       |
  0.0.0009                                  >                       |
  0.0.000c                                  >                       |
  0.0.000d                                  >                       |
  0.0.000e                                  >                       |
                                                                    |
  dasd-eckd                                                         |
  0.0.0190                                  >                       |
  0.0.0191                                  >                       |
  0.0.019d                                  >                       |
  0.0.019e                                  >┌──────────────┐
  0.0.0200                                  >|< (close)     |
  0.0.0300                                  >|  Enable      |
  0.0.0400                                  >|  Disable     |
  0.0.0592                                  >└──────────────┘     v

                              [ Continue    ]
                              [ Back        ]
```

```
=====================================================================
  Zdev setup
```

(continues on next page)

(continued from previous page)

```
================================================================
  ID                          ONLINE  NAMES                    ^
                                                               |
  generic-ccw                                                  |
  0.0.0009                                    >                |
  0.0.000c                                    >                |
  0.0.000d                                    >                |
  0.0.000e                                    >                |
                                                               |
  dasd-eckd                                                    |
  0.0.0190                                    >                |
  0.0.0191                                    >                |
  0.0.019d                                    >                |
  0.0.019e                                    >
  0.0.0200                    online  dasda   >
  0.0.0300                                    >
  0.0.0400                                    >
  0.0.0592                                    >                v

                                [ Continue   ]
                                [ Back       ]
```

```
================================================================
  Zdev setup
================================================================
                                                               ^
  dasd-eckd
  0.0.0190                                    >
  0.0.0191                                    >
  0.0.019d                                    >
  0.0.019e                                    >                |
  0.0.0200                    online  dasda   >                |
  0.0.0300                                    >                |
  0.0.0400                                    >                |
  0.0.0592                                    >                |
                                                               |
  qeth                                                         |
  0.0.0600:0.0.0601:0.0.0602          enc600  >                |
  0.0.0603:0.0.0604:0.0.0605                  >                |
                                                               |
  dasd-eckd                                                    |
  0.0.1607                                    >                v

                                [ Continue   ]
                                [ Back       ]
```

```
================================================================
  Network connections
================================================================
```

(continues on next page)

(continued from previous page)

```
  Configure at least one interface this server can use to talk to other
  machines, and which preferably provides sufficient access for updates.

    NAME    TYPE  NOTES
  [ enc600  eth   -               > ]
    static  10.222.111.24/24
    02:28:0a:00:00:39 / Unknown Vendor / Unknown Model

  [ Create bond > ]









                                  [ Done       ]
                                  [ Back       ]
```

```
==========================================================================
  Configure proxy
==========================================================================
  If this system requires a proxy to connect to the internet, enter its
  details here.

  Proxy address:
                  If you need to use a HTTP proxy to access the outside world,
                  enter the proxy information here. Otherwise, leave this
                  blank.

                  The proxy information should be given in the standard form
                  of "http://[[user][:pass]@]host[:port]/".









                                  [ Done       ]
                                  [ Back       ]
```

```
==========================================================================
  Configure Ubuntu archive mirror
==========================================================================
  If you use an alternative mirror for Ubuntu, enter its details here.
```

(continues on next page)

(continued from previous page)

```
  Mirror address:  http://ports.ubuntu.com/ubuntu-ports
                   You may provide an archive mirror that will be used instead
                   of the default.









                                    [ Done       ]
                                    [ Back       ]
```

```
================================================================================
  Guided storage configuration
================================================================================
  Configure a guided storage layout, or create a custom one:

  (X)  Use an entire disk

       [ 0X0200       local disk 6.876G                                    v ]

       [ ]  Set up this disk as an LVM group

            [ ]  Encrypt the LVM group with LUKS

                         Passphrase:


                    Confirm passphrase:


  ( )  Custom storage layout

                                    [ Done       ]
                                    [ Back       ]
```

```
================================================================================
  Storage configuration
================================================================================
  FILE SYSTEM SUMMARY                                                         ^
                                                                              |
```

(continues on next page)

(continued from previous page)

```
  MOUNT POINT     SIZE    TYPE      DEVICE TYPE                                  |
[ /               6.875G  new ext4  new partition of local disk > ]              |
                                                                                 |
                                                                                 |
AVAILABLE DEVICES                                                                |
                                                                                 |
  No available devices                                                           |
                                                                                 |
[ Create software RAID (md) > ]                                                  |
[ Create volume group (LVM) > ]                                                  |


USED DEVICES
                                                                               v

                                [ Done       ]
                                [ Reset      ]
                                [ Back       ]
```

```
================================================================================
  Storage configuration
================================================================================
  FILE SYSTEM SUMMARY                                                          ^
                                                                               |

   ┌──────────────────────── Confirm destructive action ────────────────────────┐
   │                                                                            │
   │  Selecting Continue below will begin the installation process and          │
   │  result in the loss of data on the disks selected to be formatted.         │
   │                                                                            │
   │  You will not be able to return to this or a previous screen once the      │
   │  installation has started.                                                 │
   │                                                                            │
   │  Are you sure you want to continue?                                        │
   │                                                                            │
   │                             [ No         ]                                 │
   │                             [ Continue   ]                                 │
   │                                                                            │
   └────────────────────────────────────────────────────────────────────────────┘

                                [ Reset      ]
                                [ Back       ]
```

```
================================================================================
  Profile setup
================================================================================
  Enter the username and password you will use to log in to the system. You
  can configure SSH access on the next screen but a password is still needed
  for sudo.
```

(continues on next page)

(continued from previous page)

```
             Your name:  Ed Example


   Your server's name:  10.222.111.24
                        The name it uses when it talks to other computers.

      Pick a username:  ubuntu


    Choose a password:  ********


 Confirm your password:  ********


                                [ Done       ]
```

```
======================================================================
  SSH Setup
======================================================================
  You can choose to install the OpenSSH server package to enable secure remote
  access to your server.

                  [ ]  Install OpenSSH server


  Import SSH identity:  [ No             v ]
                        You can import your SSH keys from Github or Launchpad.

      Import Username:


                  [X]  Allow password authentication over SSH




                                [ Done       ]
                                [ Back       ]
```

- It's a nice and convenient new feature to add the user's SSH keys during the installation to the system, since that makes the system login password-less on the initial login!

```
======================================================================
  SSH Setup
======================================================================
  You can choose to install the OpenSSH server package to enable secure remote
```

(continues on next page)

(continued from previous page)

```
  access to your server.

                  [X]  Install OpenSSH server


  Import SSH identity:  [ from Launchpad v ]
                        You can import your SSH keys from Github or Launchpad.

   Launchpad Username:  user
                        Enter your Launchpad username.

                  [X]  Allow password authentication over SSH




                                  [ Done       ]
                                  [ Back       ]
```

```
================================================================================
  SSH Setup
================================================================================
  You can choose to install the OpenSSH server package to enable secure remote
  access to your server.

    ┌──────────────────────────── Confirm SSH keys ─────────────────────────────┐
    │                                                                           │
    │  Keys with the following fingerprints were fetched. Do you want to        │
    │  use them?                                                                │
    │                                                                           │
    │  2048 SHA256:joGsdfW7NbJRkg17sRyXaegoR0iZEdDWdR9Hpbc2KIw user@W520  |
    │   (RSA)                                                                   │
    │  521 SHA256:T3JzxvB6K1GzXJpP5NFgX4yXvk0jhhgvbw01F7/fZ2c                   │
    │  frank.heimes@canonical.com  (ECDSA)                                      │
    │                                                                           │
    │                              [ Yes        ]                               │
    │                              [ No         ]                               │
    │                                                                           │
    └───────────────────────────────────────────────────────────────────────────┘


                                  [ Done       ]
                                  [ Back       ]
```

```
================================================================================
  Featured Server Snaps
================================================================================
  These are popular snaps in server environments. Select or deselect with
  SPACE, press ENTER to see more details of the package, publisher and
```

(continues on next page)

(continued from previous page)

```
  versions available.

  [ ] kata-containers Lightweight virtual machines that seamlessly plug into >
  [ ] docker          Docker container runtime                                 >
  [ ] mosquitto       Eclipse Mosquitto MQTT broker                            >
  [ ] etcd            Resilient key-value store by CoreOS                      >
  [ ] stress-ng       A tool to load, stress test and benchmark a computer s >
  [ ] sabnzbd         SABnzbd                                                  >
  [ ] wormhole        get things from one computer to another, safely          >
  [ ] slcli           Python based SoftLayer API Tool.                         >
  [ ] doctl           DigitalOcean command line tool                           >
  [ ] keepalived      High availability VRRP/BFD and load-balancing for Linu >
  [ ] juju            Simple, secure and stable devops. Juju keeps complexit >



                                [ Done       ]
                                [ Back       ]
```

```
================================================================================
  Install complete!
================================================================================
  ┌──────────────────────────────────────────────────────────────────────────────┐
  │          configuring raid (mdadm) service                                  ^|
  │          installing kernel                                                  |
  │          setting up swap                                                    |
  │          apply networking config                                           |
  │          writing etc/fstab                                                 |
  │          configuring multipath                                              |
  │          updating packages on target system                                 |
  │          configuring pollinate user-agent on target                         |
  │          updating initramfs configuration                                   |
  │    finalizing installation                                                  |
  │      running 'curtin hook'                                                  |
  │        curtin command hook                                                  |
  │    executing late commands                                                  |
  |final system configuration                                                   |
  |  configuring cloud-init                                                    ||
  |  installing openssh-server |                                              v|
  └──────────────────────────────────────────────────────────────────────────────┘


                                [ View full log ]
```

```
================================================================================
  Installation complete!
================================================================================
  ┌──────────────────────────── Finished install! ─────────────────────────────┐
  │          apply networking config                                          ^|
  │          writing etc/fstab                                                 |
```

(continues on next page)

(continued from previous page)

```
  |          configuring multipath                                          |
  |          updating packages on target system                             |
  |          configuring pollinate user-agent on target                     |
  |          updating initramfs configuration                               |
  |    finalizing installation                                              |
  |      running 'curtin hook'                                              |
  |        curtin command hook                                              |
  |    executing late commands                                              |
  |final system configuration                                               |
  |  configuring cloud-init                                                 |
  |  installing openssh-server                                              |
  |  restoring apt configuration                                           ||
  |downloading and installing security updates                             v|
  └─────────────────────────────────────────────────────────────────────────┘

                                [ View full log ]
                                [ Reboot        ]
```

```
  Installation complete!
=======================================================================
  ┌──────────────────────────── Finished install! ────────────────────────────┐
  |          apply networking config                                       ^|
  |          writing etc/fstab                                              |
  |          configuring multipath                                          |
  |          updating packages on target system                             |
  |          configuring pollinate user-agent on target                     |
  |          updating initramfs configuration                               |
  |    finalizing installation                                              |
  |      running 'curtin hook'                                              |
  |        curtin command hook                                              |
  |    executing late commands                                              |
  |final system configuration                                               |
  |  configuring cloud-init                                                 |
  |  installing openssh-server                                              |
  |  restoring apt configuration                                           ||
  |downloading and installing security updates                             v|
  └─────────────────────────────────────────────────────────────────────────┘

                                [ Connection to 10.222.111.24 closed by remote
host.                             [ Rebooting...  ]
Connection to 10.222.111.24 closed.
user@workstation:~$
```

Type `reset` to clear the screen and to revert it back to the defaults.

Now remove the old host key, since the system got a new one during the installation:

```
user@workstation:~$ ssh-keygen -f "/home/user/.ssh/known_hosts" -R "10.222.111.24"
# Host 10.222.111.24 found: line 159
/home/user/.ssh/known_hosts updated.
```

(continues on next page)

(continued from previous page)

```
Original contents retained as /home/user/.ssh/known_hosts.old
user@workstation:~$
```

And finally login to the newly installed z/VM guest:

```
user@workstation:~$ ssh ubuntu@10.222.111.24
Warning: Permanently added the ECDSA host key for IP address
'10.222.111.24' to the list of known hosts.
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.4.0-42-generic s390x)

  * Documentation:  https://help.ubuntu.com
  * Management:     https://landscape.canonical.com
  * Support:        https://ubuntu.com/pro

  System information as of Wed 03 Jun 2020 05:50:05 PM UTC

  System load: 0.08              Memory usage: 2%   Processes:       157
  Usage of /:  18.7% of 6.70GB   Swap usage:   0%   Users logged in: 0

0 updates can be installed immediately.
0 of these updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the individual
files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable
law.

To run a command as administrator (user "root"), use `sudo <command>`.
See `man sudo_root` for details.

ubuntu@10.222.111.24:~$ uptime
 17:50:09 up 1 min,  1 user,  load average: 0.08, 0.11, 0.05
ubuntu@10.222.111.24:~$ lsb_release -a
No LSB modules are available.
Distributor ID:        Ubuntu
Description: Ubuntu 20.04.5 LTS
Release:        20.04
Codename:        focal
ubuntu@10.222.111.24:~$ uname -a
Linux 10.222.111.24 5.4.0-42-generic #30-Ubuntu SMP Wed Aug 05 16:57:22 UTC 2020
s390x s390x s390x GNU/Linux
ubuntu@10.222.111.24:~$ exit
logout
Connection to 10.222.111.24 closed.
user@workstation:~$
```

Done!

### Non-interactive IBM z/VM autoinstall (s390x)

This non-interactive installation uses 'autoinstall', which can be considered the successor to the Debian installer (d-i) and preseed on Ubuntu. This is a detailed step-by-step guide, including output and logs (which are partially a bit shortened, as indicated by '…', to limit the size of this document).

The example z/VM guest here uses a direct-access storage device (*DASD*) and is connected to a regular (non-VLAN) network.

For a zFCP and a VLAN network example, please see the non-interactive IBM LPAR (s390x) installation using autoinstall guide.

- Start with the preparation of the (FTP) install server (if it doesn't already exist).

```
user@local:~$ ssh admin@installserver.local
admin@installserver:~$ mkdir -p /srv/ftp/ubuntu-daily-live-server-20.04
admin@installserver:~$ wget http://cdimage.ubuntu.com/ubuntu-server/focal/
daily-live/current/focal-live-server-s390x.iso --directory-prefix=/srv/ftp/
ubuntu-daily-live-server-20.04
--2020-06-26 12:18:48--  http://cdimage.ubuntu.com/ubuntu-server/focal/daily-
live/current/focal-live-server-s390x.iso
Resolving cdimage.ubuntu.com (cdimage.ubuntu.com)... 2001:67c:1560:8001::1d,
2001:67c:1360:8001::28, 2001:67c:1360:8001::27,
 ...
Connecting to cdimage.ubuntu.com (cdimage.ubuntu.
com)|2001:67c:1560:8001::1d|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 700952576 (668M) [application/x-iso9660-image]
Saving to: 'focal-live-server-s390x.iso'

focal-live-server-s 100%[===================>] 668.48M  2.73MB/s    in 4m 54s

2020-06-26 12:23:42 (2.27 MB/s) - 'focal-live-server-s390x.iso' saved
[700952576/700952576]
admin@installserver:~$
```

- The ISO image needs to be extracted now.  Since files in the boot folder need to be modified, loopback mount is not an option here:

```
admin@installserver:~$ cd /srv/ftp/ubuntu-daily-live-server-20.04
admin@installserver:/srv/ftp/ubuntu-daily-live-server-20.04$ mkdir iso
admin@installserver:/srv/ftp/ubuntu-daily-live-server-20.04$ sudo mount -o
loop ./focal-live-server-s390x.iso ./iso
[sudo] password for admin:
mount: /home/user/iso-test/iso: WARNING: device write-protected, mounted
read-only.
admin@installserver:/srv/ftp/ubuntu-daily-live-server-20.04$ ls -l
total 684530
-rw-rw-r--  1 user user 700952576 Jun 26 10:12 focal-live-server-s390x.iso
dr-xr-xr-x 10 root    root         2048 Jun 26 10:12 iso
admin@installserver:/srv/ftp/ubuntu-daily-live-server-20.04$
```

- Now make sure an FTP server is running in the *installserver* with `/srv/ftp` as ftp-server

root (as used in this example).

- Next, prepare an *autoinstall* (HTTP) server. This hosts the configuration data for the non-interactive installation.

```
admin@installserver:/srv/ftp/ubuntu-daily-live-server-20.04$ mkdir -p /srv/
www/autoinstall/zvmguest
admin@installserver:/srv/ftp/ubuntu-daily-live-server-20.04$ cd /srv/www/
autoinstall/zvmguest
admin@installserver:/srv/www/autoinstall/zvmguest$
admin@installserver:/srv/www/autoinstall/zvmguest$ echo "instance-id:
$(uuidgen || openssl rand -base64 8)" > meta-data
admin@installserver:/srv/www/autoinstall/zvmguest$ cat meta-data
instance-id: 2c2215fb-6a38-417f-b72f-376b1cc44f01
admin@installserver:/srv/www/autoinstall/zvmguest$
```

```
admin@installserver:/srv/www/autoinstall/zvmguest$ vi user-data
admin@installserver:/srv/www/autoinstall/zvmguest$ cat user-data
#cloud-config
autoinstall:
  version: 1
  refresh-installer:
    update: yes
  reporting:
    builtin:
      type: print
  apt:
    preserve_sources_list: false
    primary:
    - arches: [amd64, i386]
      uri: http://archive.ubuntu.com/ubuntu
    - arches: [default]
      uri: http://ports.ubuntu.com/ubuntu-ports
  keyboard:
    layout: en
    variant: us
  locale: en_US
  identity:
    hostname: zvmguest
    password:
"$6$ebJ1f8wxED22bTL4F46P0"
      username: ubuntu
    user-data:
      timezone: America/Boston
      users:
        - name: ubuntu
          password:
"$6$KwuxED22bTL4F46P0"
          lock_passwd: false
    early-commands:
      - touch /tmp/lets_activate_the_s390x_devices
```

(continues on next page)

(continued from previous page)

```
      - chzdev dasd -e 1f00
      - touch /tmp/s390x_devices_activation_done
    network:
      version: 2
      ethernets:
        enc600:
          addresses: [10.11.12.23/24]
          gateway4: 10.11.12.1
          nameservers:
            addresses: [10.11.12.1]
    ssh:
      install-server: true
      allow-pw: true
      authorized-keys: ['ssh-rsa  meQwtZ user@workstation # ssh-import-id
lp:user']
  admin@installserver:~$
```

- For s390x installations, the `early-commands` section is the interesting part:

```
early-commands:
  - touch /tmp/lets_activate_the_s390x_devices
  - chzdev dasd -e 1f00
  - touch /tmp/s390x_devices_activation_done
```

  The first and last `early-commands` are optional; they only frame and indicate the real s390x command activation.

  In this particular example a single *DASD ECKD* disk with the address `1f00` is enabled. zFCP disk storage can be enabled via their host (*host-bus-adapters*) addresses, for example *e000* (`chzdev zfcp -e e000`) and *e100* (`chzdev zfcp -e e000`). These have certain Logical Unit Numbers (LUNs) assigned, which are all automatically discovered and activated by `chzdev zfcp-lun -e --online`. Activation of a qeth device would look like this: `chzdev qeth -e 0600`.

- For more details about the autoinstall config options, please have a look at the autoinstall reference and autoinstall schema page.
- Now make sure a HTTP server is running with `/srv/www` as web-server root (in this particular example).
- Log in to your z/VM system using your preferred 3270 client – for example x3270 or c3270.
- Transfer the installer kernel, initrd, parmfile and exec file to the z/VM system that is used for the installation. Put these files (for example) on File Mode (Fm) *A* (a.k.a disk *A*):

```
listfiles
UBUNTU    EXEC     A1
KERNEL    UBUNTU   A1
INITRD    UBUNTU   A1
PARMFILE  UBUNTU   A1
```

- Now specify the necessary autoinstall parameters in the parmfile:

```
xedit PARMFILE  UBUNTU   A
 PARMFILE UBUNTU   O1  F 80  Trunc=80 Size=3 Line=0 Col=1 Alt=0
00000 * * * Top of File * * *
00001 ip=10.11.12.23::10.11.12.1:255.255.255.0:zvmguest:enc600:none:10.11.12.
1
00002 url=ftp://installserver.local:21/ubuntu-daily-live-server-20.04/focal-
li
ve-ser
00003 ver-s390x.iso autoinstall ds=nocloud-net;s=http://installserver.
local:80
00004 /autoinstall/zvmguest/ --- quiet
00005 * * * End of File * * *
```

**Note**

In case of any issues hitting the 80-character-per-line limit of the file, you can write parameters across two lines as long as there are no unwanted white spaces. To view all 80 characters in one line, disable the prefix area on the left. "prefix off | on" will be your friend – use it in the command area.

- You can now start the z/VM installation by executing the `UBUNTU REXX` script with `UBUNTU`.
- Now monitor the initial program load (IPL) – a.k.a. the boot-up process – of the install system. This is quite crucial, because during this process a temporary installation password is generated and displayed. The line(s) look similar to this:

```
...
|37.487141| cloud-init¬1873|: Set the following 'random' passwords
|37.487176| cloud-init¬1873|: installer: **i7UFdP8fhiVVMme3qqH8**
...
```

  This password is needed for remotely connecting to the installer via SSH in the next step.

- So, start the `REXX` script:

```
UBUNTU
00: 0000004 FILES PURGED
00: RDR FILE 1254 SENT FROM zvmguest  PUN WAS 1254 RECS 102K CPY
001 A NOHOLD NO
KEEP
00: RDR FILE 1258 SENT FROM zvmguest  PUN WAS 1258 RECS 0003 CPY
001 A NOHOLD NO
KEEP
00: RDR FILE 1262 SENT FROM zvmguest  PUN WAS 1262 RECS 303K CPY
001 A NOHOLD NO
KEEP
00: 0000003 FILES CHANGED
00: 0000003 FILES CHANGED
01: HCPGSP2627I The virtual machine is placed in CP mode due to a SIGP
```

(continues on next page)

(continued from previous page)

```
initial C
PU reset from CPU 00.
02: HCPGSP2627I The virtual machine is placed in CP mode due to a SIGP
initial C
PU reset from CPU 00.
03: HCPGSP2627I The virtual machine is placed in CP mode due to a SIGP
initial C
PU reset from CPU 00.
¬    0.403380| Initramfs unpacking failed: Decoding failed
ln: /tmp/mountroot-fail-hooks.d//scripts/init-premount/lvm2: No such file or
dir
ectory
QETH device 0.0.0600:0.0.0601:0.0.0602 configured
IP-Config: enc600 hardware address 02:28:0b:00:00:51 mtu 1500
IP-Config: enc600 guessed broadcast address 10.11.12.255
IP-Config: enc600 complete:
 address: 10.11.12.23    broadcast: 10.210.210.255   netmask: 255.255.255.0

 gateway: 10.11.12.1     dns0     : 10.11.12.1     dns1   : 0.0.0.0

 host   : zvmguest
 rootserver: 0.0.0.0 rootpath:
 filename  :
Connecting to installserver.local:21 (10.11.12.2:21)
focal-live-server-s3   2% !                                ! 15.2M  0:00:48
ETA
focal-live-server-s3  16% !*****                           !  126M  0:00:09
ETA
focal-live-server-s3  31% !**********                      !  236M  0:00:06
ETA
focal-live-server-s3  46% !**************                  !  347M  0:00:04
ETA
focal-live-server-s3  60% !******************              !  456M  0:00:03
ETA
focal-live-server-s3  74% !***********************         !  563M  0:00:02
ETA
focal-live-server-s3  88% !***************************     !  667M  0:00:00
ETA
focal-live-server-s3 100% !********************************!  752M  0:00:00
ETA
mount: mounting /cow on /root/cow failed: No such file or directory
Connecting to plymouth: Connection refused
passwd: password expiry information changed.
¬   16.748137| /dev/loop3: Can't open blockdev
¬   17.908156| systemd¬1|: Failed unmounting /cdrom.
¬ ¬0;1;31mFAILED ¬0m| Failed unmounting  ¬0;1;39m/cdrom ¬0m.
¬ ¬0;32m  OK   ¬0m| Listening on  ¬0;1;39mJournal Socket ¬0m.
         Mounting  ¬0;1;39mHuge Pages File System ¬0m...
         Mounting  ¬0;1;39mPOSIX Message Queue File System ¬0m...
```

(continues on next page)

(continued from previous page)

```
         Mounting  ¬0;1;39mKernel Debug File System ¬0m...
         Starting  ¬0;1;39mJournal Service ¬0m...
...
[   61.190916] cloud-init[2076]: Cloud-init v. 20.1-10-g71af48df-0ubuntu5
running
'modules:final' at Fri, 26 Jun 2020 11:02:01 +0000. Up 61.09 seconds.
[   61.191002] cloud-init[2076]: ci-info: no authorized SSH keys fingerprints
fo
und for user installer.
[   61.191071] cloud-init[2076]: Cloud-init v. 20.1-10-g71af48df-0ubuntu5
finished at Fri, 26 Jun 2020 11:02:01 +0000.
Datasource DataSourceNoCloudNet [seed=cmdline,
/var/lib/cloud/seed/nocloud,
http://installserver.local:80/autoinstall/zvmguest/]
[dsmode=net].  Up 61.18 seconds
[   61.191136] cloud-init[2076]: Welcome to Ubuntu Server Installer!
[   61.191202] cloud-init[2076]: Above you will find SSH host keys and a
random
password set for the `installer` user. You can use these credentials to ssh-
in
and complete the installation. If you provided SSH keys in the cloud-init
datasource,
they were also provisioned to the installer user.
[   61.191273] cloud-init[2076]: If you have access to the graphical console,
like TTY1 or HMC ASCII terminal you can complete the installation there too.

It is possible to connect to the installer over the network, which
might allow the use of a more capable terminal.

To connect, SSH to installer@10.11.12.23.

The password you should use is ''i7UFdP8fhiVVMme3qqH8''.

The host key fingerprints are:

RSA     SHA256:rBXLeUke3D4gKdsruKEajHjocxc9hr3PI
ECDSA   SHA256:KZZYFswtKxFXWQPuQS9QpOBUoS6RHswis
ED25519 SHA256:s+5tZfagx0zffC6gYRGW3t1KcBH6f+Vt0

Ubuntu 20.04 LTS ubuntu-server sclp_line0
```

- At short notice, you can even log in to the system with the user ‘installer’ and the temporary password that was given at the end of the boot-up process (see above) of the installation system:

```
user@workstation:~$ ssh installer@zvmguest
The authenticity of host 'zvmguest (10.11.12.23)' can't be established.
ECDSA key fingerprint is SHA256:O/dU/D8jJAEGQcbqKGE9La24IRxUPLpzzs5li9F6Vvk.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

(continues on next page)

(continued from previous page)

```
Warning: Permanently added 'zvmguest,10.11.12.23' (ECDSA) to the list of
known hosts.
installer@zvmguest's password:
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-37-generic s390x)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

  System information as of Fri Jun 26 11:08:18 UTC 2020

  System load:    1.25      Memory usage: 4%   Processes:       192
  Usage of /home: unknown   Swap usage:   0%   Users logged in: 0

0 updates can be installed immediately.
0 of these updates are security updates.


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

the installer running on /dev/tty1 will perform the autoinstall

press enter to start a shell
```

- Please note that it informs you about a currently-running autoinstall process:

```
the installer running on /dev/tty1 will perform the autoinstall
```

- Nevertheless, we can quickly check some things – although, only until the autoinstall process is finished and the post-install reboot has been triggered:

```
root@ubuntu-server:/# ls -l /tmp/lets_activate_the_s390x_devices
-rw-r--r-- 1 root root 0 Jun 26 11:08 /tmp/lets_activate_the_s390x_devices
-rw-r--r-- 1 root root 0 Jun 26 11:09 /tmp/s390x_devices_activation_done

root@ubuntu-server:/# lszdev | grep yes
dasd-eckd    0.0.1f00                                       yes  yes
qeth         0.0.0600:0.0.0601:0.0.0602                     yes  no
enc600
root@ubuntu-server:/#
```

- If you wait long enough, you'll see that the remote session gets closed:

```
root@ubuntu-server:/# Connection to zvmguest closed by remote host.
Connection to zvmguest closed.
user@workstation:~$
```

- As well as at the console:

```
ubuntu-server login:

[[0;1;31mFAILED[0m] Failed unmounting [0;1;39m/cdrom[0m.
[  169.161139] sd-umoun[15600]: Failed to unmount /oldroot: Device or
resource busy
[  169.161550] sd-umoun[15601]: Failed to unmount /oldroot/cdrom: Device or
resource busy
[  169.168118] shutdown[1]: Failed to finalize  file systems, loop devices,
ignoring
Total: 282 Selected: 0

Command:
```

- …and that the post-install reboot got triggered:

```
Message
Mounting [0;1;39mKernel Configuration File System[0m...
Starting [0;1;39mApply Kernel Variables[0m...
[[0;32m  OK  [0m] Finished [0;1;39mRemount Root and Kernel File Systems[0m.
[[0;32m  OK  [0m] Finished [0;1;39mUncomplicated firewall[0m.
[[0;32m  OK  [0m] Mounted [0;1;39mFUSE Control File System[0m.
[[0;32m  OK  [0m] Mounted [0;1;39mKernel Configuration File System[0m.
...
[   35.378928] cloud-init[2565]: Cloud-init v. 20.1-10-g71af48df-0ubuntu5
runnin
g 'modules:final' at Fri, 26 Jun 2020 11:10:44 +0000. Up 35.29 seconds.
[   35.378978] cloud-init[2565]: Cloud-init v. 20.1-10-g71af48df-0ubuntu5
finish
ed at Fri, 26 Jun 2020 11:10:44 +0000. Datasource DataSourceNone.  Up 35.37
seconds
[   35.379008] cloud-init[2565]: 2020-06-26 11:10:44,359 - cc_final_message.
py[W
ARNING]: Used fallback datasource
[[0;32m  OK  [0m] Finished [0;1;39mExecute cloud user/final scripts[0m.
[[0;32m  OK  [0m] Reached target [0;1;39mCloud-init target[0m.

zvmguest login:
```

- With the completion of the reboot the autoinstall is finished and the z/VM guest is ready to use:

```
user@workstation:~$ ssh-keygen -f "/home/user/.ssh/known_hosts" -R "zvmguest"
# Host zvmguest found: line 163
/home/user/.ssh/known_hosts updated.
Original contents retained as /home/user/.ssh/known_hosts.old
```

(continues on next page)

(continued from previous page)

```
user@workstation:~$ ssh ubuntu@zvmguest
The authenticity of host 'zvmguest (10.11.12.23)' can't be established.
ECDSA key fingerprint is SHA256:iGtCArEg+ZnoZlgtOkvmyy0gPY8UEI+f7zoISOF+m/0.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'zvmguest,10.11.12.23' (ECDSA) to the list of
known hosts.
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-39-generic s390x)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

  System information as of Fri 26 Jun 2020 11:12:23 AM UTC

  System load: 0.21               Memory usage: 3%   Processes:       189
  Usage of /:  28.2% of 30.88GB   Swap usage:   0%   Users logged in: 0

10 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY,
to the extent permitted by applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@zvmguest:~$ uptime
 11:12:35 up 2 min,  1 user,  load average: 0.18, 0.17, 0.08
ubuntu@zvmguest:~$ lsb_release -a
No LSB modules are available.
Distributor ID:        Ubuntu
Description:        Ubuntu 20.04 LTS
Release:        20.04
Codename:        focal
ubuntu@zvmguest:~$ uname -a
Linux zvmguest 5.4.0-39-generic #43-Ubuntu SMP Fri Jun 19 10:27:17
UTC 2020 s390x s390x s390x
GNU/Linux
ubuntu@zvmguest:~$ lszdev | grep yes
dasd-eckd    0.0.1f00                                       yes  yes
qeth         0.0.0600:0.0.0601:0.0.0602                     yes  yes
enc600
ubuntu@zvmguest:~$ exit
logout
```

(continues on next page)

(continued from previous page)

```
Connection to zvmguest closed.
user@workstation:~$
```

### Some closing notes

- It's always best to use the latest installer and autoinstall components: either make sure the installer gets updated to the latest level, or just use a current daily live-server image.
- The ISO image specified with the kernel parameters needs to fit in the boot folder. Its kernel and initrd are specified in the 'Load from Removable Media and Server' task at the *hardware management console (HMC)*.
- In addition to activating disk storage resources in `early-commands`, other devices like OSA/qeth can be added and activated there, too. This is not needed for the basic network device, as specified in the kernel parameters that are used for the installation (that one is automatically handled).
- If everything is properly set up – FTP server for the image, HTTP server for the autoinstall config files – the installation can be as quick as 2 to 3 minutes (depending on the complexity of the autoinstall YAML file).
- There is a simple way of generating a sample autoinstall YAML file: one can perform an interactive Subiquity installation, grab the file `/var/log/installer/autoinstall-user-data`, and use it as an example – but beware that the `early-commands` entries to activate the s390x-specific devices need to be added manually!

### Interactive live server installation on IBM Z LPAR (s390x)

Doing an interactive (manual) live installation as described here - meaning without specifying a parmfile - has been supported in Ubuntu Server since LTS 20.04.5 ('Focal').

The following guide assumes that an FTP server to host the installation files is in place, which can be used by the 'Load from Removable Media and Server' task of the *Hardware Management Console (HMC)*.

### Download and mount the ISO

Download the 'focal daily live image' (later 20.04.5 image).

Now loop-back mount the ISO to extract the four files that are needed for the installation:

```
user@workstation:~$ mkdir iso
user@workstation:~$ sudo mount -o loop ubuntu-20.04.5-live-server-s390x.iso iso
user@workstation:~$

user@workstation:~$ ls -1 ./iso/boot/{ubuntu.exec,parmfile.*,kernel.u*,initrd.u*}
./iso/boot/initrd.ubuntu
./iso/boot/kernel.ubuntu
./iso/boot/parmfile.ubuntu
./iso/boot/ubuntu.exec
```

### Make the files available via your FTP server.

- Open the IBM Z HMC and navigate to ‘Systems Management’ on your machine.
- Select the LPAR that you are going to install Ubuntu Server on. In this example we use LPAR `s1lp11`.
- Now select menu: ‘Recovery’ –> ‘Load from Removable Media or Server’ task.
- Fill out the ‘Load from Removable Media or Server’ form as follows (adapt the settings to your particular installation environment):

```
Load from Removable Media, or Server - <machine>:s1lp11
Use this task to load operating system software or utility programs
from a CD / DVD-ROM or a server that can be accessed using FTP.
Select the source of the software:
o Hardware Management Console CD / DVD-ROM
o Hardware Management Console CD / DVD-ROM and assign for operating system
use
o Hardware Management Console USB flash memory drive
o Hardware Management Console USB flash memory drive and assign for operating
system use
* FTP Source
   Host computer:        install-server
   User ID:        ftpuser
   Password: ********
   Account (optional):
   File location (optional): ubuntu-live-server-20.04.5/boot
```

  You may need to adjust the file’s location according to your install server environment.

- Confirm the entered data:

```
Load from Removable Media or Server - Select Software to Install -
<machine>:s1lp11
Select the software to install.
Select   Name                                          Description
*  ubuntu-live-server-20.04.5/boot/ubuntu.ins     Ubuntu for IBM Z (default
kernel)
```

- Confirm again that jobs might be cancelled if proceeding:

```
Load from Removable Media or Server Task Confirmation -
<machine>:s1lp11
Load will cause jobs to be cancelled.
Do you want to continue with this task?
ACT33501
```

- And confirm a last time that it’s understood that the task is disruptive:

```
Disruptive Task Confirmation : Load from Removable Media or Server -
<machine>:s1lp11

Attention: The Load from Removable Media or Server task is disruptive.
```

(continues on next page)

(continued from previous page)

```
Executing the Load from Removable Media or Server task may
adversely affect the objects listed below. Review the confirmation text
for each object before continuing with the Load from Removable Media
or Server task.

Objects that will be affected by the Load from Removable Media or
Server task

System Name       Type     OS Name     Status      Confirmation Text
<machine>:s1lp11   Image                Operating   Load from Removable Media
or Server causes operations to be disrupted, since the target is
currently in use and operating normally.

Do you want to execute the Load from Removable Media or Server task?
```

- The 'Load from Removable media or Server' task is now executed:

```
Load from Removable media or Server Progress - P00B8F67:S1LPB
Turn on context sensitive help.
Function duration time:        00:55:00
Elapsed time: 00:00:04
Select      Object Name       Status
*           <machine> s1lp11   Please wait while the image is being loaded.
```

- This may take a moment, but you will soon see:

```
Load from Removable media or Server Progress - <machine>:s1lp11
Function duration time:        00:55:00
Elapsed time:        00:00:21
Select      Object Name       Status
*           <machine> s1lp11   Success
```

- Close the 'Load from Removable media or Server' task and open the console a.k.a. 'Operating System Messages' instead. If no parmfile was configured or provided, you will find the following lines in the 'Operating System Messages' task:

```
Operating System Messages - <machine>:s1lp11

Message

Unable to find a medium container a live file system
Attempt interactive netboot from a URL?
yes no (default yes):
```

- By default, you will now see the interactive network configuration menu (again, only if no parmfile was prepared with sufficient network configuration information).
- Proceed with the interactive network configuration – in this case in a VLAN environment:

```
Unable to find a medium container a live file system
Attempt interactive netboot from a URL?
yes no (default yes):
yes
Available qeth devices:
0.0.c000 0.0.c003 0.0.c006 0.0.c009 0.0.c00c 0.0.c00f
zdev to activate (comma separated, optional):
0.0.c000
QETH device 0.0.c000:0.0.c001:0.0.c002 configured
Two methods available for IP configuration:
* static: for static IP configuration
* dhcp: for automatic IP configuration
static dhcp (default 'dhcp'):
static
ip:
10.222.111.11
gateway (default 10.222.111.1):
10.222.111.1
dns (default 10.222.111.1):
10.222.111.1
vlan id (optional):
1234
http://cdimage.ubuntu.com/ubuntu/releases/20.04.5/release/ubuntu-20.04.5-
live-server-s390x.iso (default)
url:
ftp://10.11.12.2:21/ubuntu-live-server-20.04.5/ubuntu-20.04.5-live-server-
s390x.iso
http_proxy (optional):
```

- After the last interactive step here (that this is about an optional proxy configuration), the installer will complete its boot-up process:

```
Configuring networking...
IP-Config: encc000.1234 hardware address 3e:00:10:55:00:ff mtu 1500
IP-Config: encc000.1234 guessed broadcast address 10.222.111.255
IP-Config: encc000.1234 complete:
address: 10.222.111.11    broadcast: 10.222.111.255   netmask: 255.255.255.0

gateway: 10.222.111.1     dns0     : 10.222.111.1     dns1   : 0.0.0.0

rootserver: 0.0.0.0 rootpath:
filename  :
Connecting to 10.11.12.2:21 (10.11.12.2:21)
focal-live-server-s  10% |***                             | 72.9M  0:00:08 ETA
focal-live-server-s  25% |********                        |  168M  0:00:05 ETA
focal-live-server-s  42% |*************                   |  279M  0:00:04 ETA
focal-live-server-s  58% |******************              |  390M  0:00:02 ETA
focal-live-server-s  75% |************************        |  501M  0:00:01 ETA
focal-live-server-s  89% |****************************    |  595M  0:00:00 ETA
focal-live-server-s  99% |******************************* |  662M  0:00:00 ETA
```

(continues on next page)

(continued from previous page)

```
focal-live-server-s 100% |*******************************|  663M  0:00:00 ETA
ip: RTNETLINK answers: File exists
no search or nameservers found in /run/net-encc000.1234.conf / run/net-*.conf /
run/net6-*.conf
[  399.808930] /dev/loop3: Can't open blockdev
[[0;1;31m SKIP [0m] Ordering cycle found, skipping [0;1;39mLogin Prompts[0m
[  401.547705] systemd[1]: multi-user.target: Job getty.target/start deleted to
break ordering cycle starting with multi-user.target/start
[  406.241972] cloud-init[1321]: Cloud-init v. 20.2-45-g5f7825e2-0ubuntu1 running
'init-local' at Wed, 03 Jun 2020 17:07:39 +0000. Up 406.00 seconds.
[  407.025557] cloud-init[1348]: Cloud-init v. 20.2-45-g5f7825e2-0ubuntu1 running
'init' at Wed, 03 Jun 2020 17:07:40 +0000. Up 406.87 seconds.
[  407.025618] cloud-init[1348]: ci-info:   ++++Net device info++++
[  407.025658] cloud-init[1348]: ci-info: +--------------+------+--------------
--------------+--------------+--------+-------------------+
[  407.025696] cloud-init[1348]: ci-info: |    Device    |  Up  |           Addr
ess            |      Mask     | Scope  |     Hw-Address    |
[  407.025731] cloud-init[1348]: ci-info: +--------------+------+--------------
--------------+--------------+--------+-------------------+
[  407.025766] cloud-init[1348]: ci-info: |   encc000    | True | fe80::3ca7:10f
f:fea5:c69e/64 |       .       |  link  | 72:5d:0d:09:ea:76 |
[  407.025802] cloud-init[1348]: ci-info: | encc000.1234 | True |        10.245.
236.11         | 255.255.255.0 | global | 72:5d:0d:09:ea:76 |
[  407.025837] cloud-init[1348]: ci-info: | encc000.1234 | True | fe80::3ca7:10f
f:fea5:c69e/64 |       .       |  link  | 72:5d:0d:09:ea:76 |
[  407.025874] cloud-init[1348]: ci-info: |      lo      | True |          127.0
.0.1           |   255.0.0.0   |  host  |         .         |
[  407.025909] cloud-init[1348]: ci-info: |      lo      | True |           ::1/
128            |       .       |  host  |         .         |
[  407.025944] cloud-init[1348]: ci-info: +--------------+------+--------------
--------------+--------------+--------+-------------------+
[  407.025982] cloud-init[1348]: ci-info: ++++++++++++++Route I
Pv4 info+++++++++++++++
[  407.026017] cloud-init[1348]: ci-info: +-------+-------------+-------------
+--------------+-------------+-------+
[  407.026072] cloud-init[1348]: ci-info: | Route | Destination  |   Gateway
|    Genmask    |  Interface   | Flags |
[  407.026107] cloud-init[1348]: ci-info: +-------+-------------+-------------
+--------------+-------------+-------+
[  407.026141] cloud-init[1348]: ci-info: |   0   |   0.0.0.0    | 10.222.111.1
|    0.0.0.0    | encc000.1234 |   UG  |
[  407.026176] cloud-init[1348]: ci-info: |   1   | 10.222.111.0 |   0.0.0.0
| 255.255.255.0 | encc000.1234 |   U   |
[  407.026212] cloud-init[1348]: ci-info: +-------+-------------+-------------
+--------------+-------------+-------+
[  407.026246] cloud-init[1348]: ci-info: ++++++++++++++++++++Route IPv6 info+++
++++++++++++++++++
[  407.026280] cloud-init[1348]: ci-info: +-------+-------------+---------+-----
---------+-------+
```

(continues on next page)

(continued from previous page)

```
[  407.026315] cloud-init[1348]: ci-info: | Route | Destination | Gateway |  Int
erface   | Flags |
[  407.026355] cloud-init[1348]: ci-info: +-------+-------------+---------+-----
---------+-------+
[  407.026390] cloud-init[1348]: ci-info: |   1   |  fe80::/64  |    ::   |   en
cc000    |   U   |
[  407.026424] cloud-init[1348]: ci-info: |   2   |  fe80::/64  |    ::   | encc
000.1234 |   U   |
[  407.026458] cloud-init[1348]: ci-info: |   4   |    local    |    ::   |   en
cc000    |   U   |
[  407.026495] cloud-init[1348]: ci-info: |   5   |    local    |    ::   | encc
000.1234 |   U   |
[  407.026531] cloud-init[1348]: ci-info: |   6   |   ff00::/8  |    ::   |   en
cc000    |   U   |
[  407.026566] cloud-init[1348]: ci-info: |   7   |   ff00::/8  |    ::   | encc
000.1234 |   U   |
[  407.026600] cloud-init[1348]: ci-info: +-------+-------------+---------+-----
---------+-------+
[  407.883058] cloud-init[1348]: Generating public/private rsa key pair.
[  407.883117] cloud-init[1348]: Your identification has been saved in /etc/ssh/
ssh_host_rsa_key
[  407.883154] cloud-init[1348]: Your public key has been saved in /etc/ssh/ssh_
host_rsa_key.pub
[  407.883190] cloud-init[1348]: The key fingerprint is:
[  407.883232] cloud-init[1348]: SHA256:KX5cHC4YL9dXpvhnP6eSfS+J/zmKgg9zdlEzaEb+
RTA root@ubuntu-server
[  407.883267] cloud-init[1348]: The key's randomart image is:
[  407.883302] cloud-init[1348]: +---[RSA 3072]----+
[  407.883338] cloud-init[1348]: |           . E.. |
[  407.883374] cloud-init[1348]: |          o . o  |
[  407.883408] cloud-init[1348]: |      .   .= +o. |
[  407.883443] cloud-init[1348]: |       + =ooo++  |
[  407.883478] cloud-init[1348]: |      + S *.o.   |
[  407.883512] cloud-init[1348]: |     . = o o.    |
[  407.883546] cloud-init[1348]: |      .o+o ..+o. |
[  407.883579] cloud-init[1348]: |       o=.. =o+++|
[  407.883613] cloud-init[1348]: |        .... ++*O|
[  407.883648] cloud-init[1348]: +----[SHA256]-----+
[  407.883682] cloud-init[1348]: Generating public/private dsa key pair.
[  407.883716] cloud-init[1348]: Your identification has been saved in /etc/ssh/
ssh_host_dsa_key
[  407.883750] cloud-init[1348]: Your public key has been saved in /etc/ssh/ssh_
host_dsa_key.pub
[  407.883784] cloud-init[1348]: The key fingerprint is:
[  407.883817] cloud-init[1348]: SHA256:xu3vlG1BReKDy3DsuMZc/lg5y/+nhzlEmLDk/qFZ
Am0 root@ubuntu-server
[  407.883851] cloud-init[1348]: The key's randomart image is:
[  407.883905] cloud-init[1348]: +---[DSA 1024]----+
[  407.883941] cloud-init[1348]: |              ..o|
```

(continues on next page)

(continued from previous page)

```
[  407.883975] cloud-init[1348]: |          o. o o |
[  407.884008] cloud-init[1348]: |         +.o+o+  |
[  407.884042] cloud-init[1348]: |       ...E*oo.. |
[  407.884076] cloud-init[1348]: |        S+o =..  |
[  407.884112] cloud-init[1348]: |       . +o+oo.o |
[  407.884145] cloud-init[1348]: |          **+o*o |
[  407.884179] cloud-init[1348]: |         .oo.*+oo|
[  407.884212] cloud-init[1348]: |           .+ ===|
[  407.884246] cloud-init[1348]: +----[SHA256]-----+
[  407.884280] cloud-init[1348]: Generating public/private ecdsa key pair.
[  407.884315] cloud-init[1348]: Your identification has been saved in /etc/ssh/
ssh_host_ecdsa_key
[  407.884352] cloud-init[1348]: Your public key has been saved in /etc/ssh/ssh_
host_ecdsa_key.pub
[  407.884388] cloud-init[1348]: The key fingerprint is:
[  407.884422] cloud-init[1348]: SHA256:P+hBF3fj/pu6+0KaywUYii3Lyuc09Za9/a2elCDO
gdE root@ubuntu-server
[  407.884456] cloud-init[1348]: The key's randomart image is:
[  407.884490] cloud-init[1348]: +---[ECDSA 256]---+
[  407.884524] cloud-init[1348]: |                 |
[  407.884558] cloud-init[1348]: |         .       |
[  407.884591] cloud-init[1348]: |        ..E . o  |
[  407.884625] cloud-init[1348]: |      o .ooo o . |
[  407.884660] cloud-init[1348]: |     o +S.+.. .  |
[  407.884694] cloud-init[1348]: |    . +..*oo.+ . |
[  407.884728] cloud-init[1348]: |     =  o+=.+.+  |
[  407.884762] cloud-init[1348]: |  . o......++o oo|
[  407.884795] cloud-init[1348]: |   oo.  .  +.*@*+|
[  407.884829] cloud-init[1348]: +----[SHA256]-----+
[  407.884862] cloud-init[1348]: Generating public/private ed25519 key pair.
[  407.884896] cloud-init[1348]: Your identification has been saved in /etc/ssh/
ssh_host_ed25519_key
[  407.884930] cloud-init[1348]: Your public key has been saved in /etc/ssh/ssh_
host_ed25519_key.pub
[  407.884966] cloud-init[1348]: The key fingerprint is:
[  407.884999] cloud-init[1348]: SHA256:CbZpkR9eFHuB1sCDZwSdSdwJzy9FpsIWRIyc9ers
hZ0 root@ubuntu-server
[  407.885033] cloud-init[1348]: The key's randomart image is:
[  407.885066] cloud-init[1348]: +--[ED25519 256]--+
[  407.885100] cloud-init[1348]: |       ../%X..o  |
[  407.885133] cloud-init[1348]: |       .=o&*+=   |
[  407.885167] cloud-init[1348]: |      = .+*.* .  |
[  407.885200] cloud-init[1348]: |     . B = + o   |
[  407.885238] cloud-init[1348]: |      + S . . .  |
[  407.885274] cloud-init[1348]: |     .   o o o   |
[  407.885308] cloud-init[1348]: |          + E    |
[  407.885345] cloud-init[1348]: |         . .     |
[  407.885378] cloud-init[1348]: |          .      |
[  407.885420] cloud-init[1348]: +----[SHA256]-----+
```

(continues on next page)

(continued from previous page)

```
[  418.521933] cloud-init[2185]: Cloud-init v. 20.2-45-g5f7825e2-0ubuntu1 runnin
g 'modules:config' at Wed, 03 Jun 2020 17:07:52 +0000. Up 418.40 seconds.
[  418.522012] cloud-init[2185]: Set the following 'random' passwords
[  418.522053] cloud-init[2185]: installer:C7BZrW76s4mJzmpf4eUy
ci-info: no authorized SSH keys fingerprints found for user installer.
<14>Jun  3 17:07:52 ec2:
<14>Jun  3 17:07:52 ec2: ##############################################################
######
<14>Jun  3 17:07:52 ec2: -----BEGIN SSH HOST KEY FINGERPRINTS-----
<14>Jun  3 17:07:52 ec2: 1024 SHA256:xu3vlG1BReKDy3DsuMZc/lg5y/+nhzlEmLDk/qFZAm0
root@ubuntu-server (DSA)
<14>Jun  3 17:07:52 ec2: 256 SHA256:P+hBF3fj/pu6+0KaywUYii3Lyuc09Za9/a2elCDOgdE
root@ubuntu-server (ECDSA)
<14>Jun  3 17:07:52 ec2: 256 SHA256:CbZpkR9eFHuB1sCDZwSdSdwJzy9FpsIWRIyc9ershZ0
root@ubuntu-server (ED25519)
<14>Jun  3 17:07:52 ec2: 3072 SHA256:KX5cHC4YL9dXpvhnP6eSfS+J/zmKgg9zdlEzaEb+RTA
root@ubuntu-server (RSA)
<14>Jun  3 17:07:52 ec2: -----END SSH HOST KEY FINGERPRINTS-----
<14>Jun  3 17:07:52 ec2: ##############################################################
######
-----BEGIN SSH HOST KEY KEYS-----
ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBC2zp4Fq
r1+NJOIEQIISbX+EzeJ6ucXSLi2xEvurgwq8iMYT6yYOXBOPc/XzeFa6vBCDZk3SSSW6Lq83y7VmdRQ=
root@ubuntu-server
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIJFzgips94nJNoR4QumiyqlJoSlZ48P+NVrd7zgD5k4T
root@ubuntu-server
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQChKo06O715FAjd6ImK7qZbWnL/cpgQ2A2gQEqFNO+1
joF/41ygxuw5aG0IQObWFpV9jDsMF5z4qHKzX8tFCpKC0s4uR8QBxh1dDm4wcwcgtAfVLqh7S4/R9Sqa
IFnkCzxThhNeMarcRrutY0mIzspmCg/QvfE1wrXJzl+RtOJ7GiuHHqpm76fX+6ZF1BYhkA87dXQiID2R
yUubSXKGg0NtzlgSzPqD3GB+HxRHHHLT5/Xq+njPq8jIUpqSoHtkBupsyVmcD9gDbz6vng2PuBHwZP9X
17QtyOwxddxk4xIXaTup4g8bH1oF/czsWqVxNdfB7XqzROFUOD9rMIB+DwBihsmH1kRik4wwLi6IH4hu
xrykKvfb1xcZe65kR42oDI7JbBwxvxGrOKx8DrEXnBpOWozS0IDm2ZPh3ci/0uCJ4LTItByyCfAe/gyR
5si4SkmXrIXf5BnErZRgyJnfxKXmsFaSh7wf15w6GmsgzyD9sI2jES9+4By32ZzYOlDpi0s= root@ub
untu-server
-----END SSH HOST KEY KEYS-----
[  418.872320] cloud-init[2203]: Cloud-init v. 20.2-45-g5f7825e2-0ubuntu1 runnin
g 'modules:final' at Wed, 03 Jun 2020 17:07:52 +0000. Up 418.79 seconds.
[  418.872385] cloud-init[2203]: ci-info: no authorized SSH keys fingerprints fo
und for user installer.
[  418.872433] cloud-init[2203]: Cloud-init v. 20.2-45-g5f7825e2-0ubuntu1 finish
ed at Wed, 03 Jun 2020 17:07:52 +0000. Datasource DataSourceNoCloud [seed=/var/l
ib/cloud/seed/nocloud][dsmode=net].  Up 418.86 seconds
[  418.872484] cloud-init[2203]: Welcome to Ubuntu Server Installer!
[  418.872529] cloud-init[2203]: Above you will find SSH host keys and a random
password set for the `installer` user. You can use these credentials to ssh-in a
nd complete the installation. If you provided SSH keys in the cloud-init datasou
rce, they were also provisioned to the installer user.
[  418.872578] cloud-init[2203]: If you have access to the graphical console, li
ke TTY1 or HMC ASCII terminal you can complete the installation there too.
```

(continues on next page)

(continued from previous page)

```
It is possible to connect to the installer over the network, which
might allow the use of a more capable terminal.

To connect, SSH to installer@10.222.111.11.

The password you should use is "C7BZrW76s4mJzmpf4eUy".

The host key fingerprints are:

RSA     SHA256:KX5cHC4YL9dXpvhnP6eSfS+J/zmKgg9zdlEzaEb+RTA
ECDSA   SHA256:P+hBF3fj/pu6+0KaywUYii3Lyuc09Za9/a2elCDOgdE
ED25519 SHA256:CbZpkR9eFHuB1sCDZwSdSdwJzy9FpsIWRIyc9ershZ0

Ubuntu Focal Fossa (development branch) ubuntu-server sclp_line0
ubuntu-server login:
```

- At this point you can proceed with the regular installation either by using 'Recovery' –> 'Integrated ASCII Console' or with a remote SSH session.
- If the 'Integrated ASCII Console' was opened (you can hit F3to refresh the task), the initial Subiquity installation screen is presented, which looks like this:

```
================================================================================
  Willkommen! Bienvenue! Welcome! ????? ??????????! Welkom!           [ Help ]
================================================================================
  Use UP, DOWN and ENTER keys to select your language.
                [ English                                          > ]
                [ Asturianu                                        > ]
                [ Cataln                                           > ]
                [ Hrvatski                                         > ]
                [ Nederlands                                       > ]
                [ Suomi                                            > ]
                [ Francais                                         > ]
                [ Deutsch                                          > ]
                [ Magyar                                           > ]
                [ Latvie?u                                         > ]
                [ Norsk bokm?l                                     > ]
                [ Polski                                           > ]
                [ Espanol                                          > ]
```

- Since the user experience is nicer in a remote SSH session, we recommend using that. However, with certain network environments it's just not possible to use a remote shell, and the 'Integrated ASCII Console' will be the only option.

**Note**

At the end of the installer boot-up process, **all** necessary information is provided to proceed with a remote shell.

- The command to execute locally is:

```
user@workstation:~$ ssh installer@10.222.111.11
```

- A temporary random password for the installation was created and shared as well, which you should use without the leading and trailing double quotes:

```
"C7BZrW76s4mJzmpf4eUy"
```

- Hence the remote session for the installer can be opened by:

```
user@workstation:~$ ssh installer@10.222.111.11
The authenticity of host '10.222.111.11 (10.222.111.11)' can't be established.
ECDSA key fingerprint is SHA256:P+hBF3fj/pu6+0KaywUYii3Lyuc09Za9/a2elCDOgdE.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.222.111.11' (ECDSA) to the list of known hosts.
installer@10.222.111.11's password: C7BZrW76s4mJzmpf4eUy
```

- One may swiftly see some login messages like the following ones:

```
Welcome to Ubuntu Focal Fossa (development branch) (GNU/Linux 5.4.0-42-generic
s390x)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

  System information as of Wed Jun  3 17:32:10 UTC 2020

  System load:    0.0       Memory usage: 2%   Processes:       146
  Usage of /home: unknown   Swap usage:   0%   Users logged in: 0

0 updates can be installed immediately.
0 of these updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
```

Eventually you'll reach the initial Subiquity installer screen:

```
================================================================================
  Willkommen! Bienvenue! Welcome! ????? ??????????! Welkom!
================================================================================
  Use UP, DOWN and ENTER keys to select your language.

                [ English                                       > ]
                [ Asturianu                                     > ]
                [ Cataln                                        > ]
                [ Hrvatski                                      > ]
```

(continues on next page)

(continued from previous page)

```
                [ Nederlands                                    > ]
                [ Suomi                                         > ]
                [ Francais                                      > ]
                [ Deutsch                                       > ]
                [ Magyar                                        > ]
                [ Latvie?u                                      > ]
                [ Norsk bokm?l                                  > ]
                [ Polski                                        > ]
                [ Espanol                                       > ]
```

- From this point, you can follow the normal Subiquity installation. For more details, refer to the Subquity installer documentation.

(I'm leaving some pretty standard screenshots here just to give an example for a basic installation …)

```
================================================================================
  Keyboard configuration
================================================================================
  Please select your keyboard layout below, or select "Identify keyboard" to
  detect your layout automatically.

                 Layout:  [ English (US)                    v ]


                Variant:  [ English (US)                    v ]


                             [ Identify keyboard ]
```

```
                                [ Done       ]
                                [ Back       ]
```

```
================================================================================
  Zdev setup
================================================================================
  ID                           ONLINE  NAMES                                  ^
                                                                              |
  dasd-eckd
  0.0.1600                                      >
  0.0.1601                                      >
  0.0.1602                                      >
  0.0.1603                                      >
```

(continues on next page)

(continued from previous page)

```
  0.0.1604                                  >
  0.0.1605                                  >
  0.0.1606                                  >
  0.0.1607                                  >
  0.0.1608                                  >
  0.0.1609                                  >
  0.0.160a                                  >
  0.0.160b                                  >
  0.0.160c                                  >
  0.0.160d                                  >                              v

                              [ Continue    ]
                              [ Back        ]
```

```
================================================================
  Zdev setup
================================================================
  ID                          ONLINE  NAMES                              ^
                                                                         |
  dasd-eckd
  0.0.1600                                  >┌──────────────┐
  0.0.1601                                  >|< (close)    |
  0.0.1602                                  >|  Enable     |
  0.0.1603                                  >|  Disable    |
  0.0.1604                                  >└──────────────┘
  0.0.1605                                  >
  0.0.1606                                  >
  0.0.1607                                  >
  0.0.1608                                  >
  0.0.1609                                  >
  0.0.160a                                  >
  0.0.160b                                  >
  0.0.160c                                  >
  0.0.160d                                  >                              v

                              [ Continue    ]
                              [ Back        ]
```

```
================================================================
  Zdev setup
================================================================
  ID                          ONLINE  NAMES                              ^
                                                                         |
  dasd-eckd
  0.0.1600                                  >
  0.0.1601                    online  dasda >
  0.0.1602                                  >
  0.0.1603                                  >
  0.0.1604                                  >
```

(continues on next page)

(continued from previous page)

```
  0.0.1605                                  >
  0.0.1606                                  >
  0.0.1607                                  >
  0.0.1608                                  >
  0.0.1609                                  >
  0.0.160a                                  >
  0.0.160b                                  >
  0.0.160c                                  >
  0.0.160d                                  >                              v

                                [ Continue    ]
                                [ Back        ]
```

- One may hit the End key here – that will automatically scroll down to the bottom of the Z devices list and screen:

```
================================================================================
  Zdev setup
================================================================================
  0.0.f1de:0.0.f1df                         >                              ^
  0.0.f1e0:0.0.f1e1                         >
  0.0.f1e2:0.0.f1e3                         >
  0.0.f1e4:0.0.f1e5                         >
  0.0.f1e6:0.0.f1e7                         >
  0.0.f1e8:0.0.f1e9                         >
  0.0.f1ea:0.0.f1eb                         >
  0.0.f1ec:0.0.f1ed                         >
  0.0.f1ee:0.0.f1ef                         >
  0.0.f1f0:0.0.f1f1                         >
  0.0.f1f2:0.0.f1f3                         >
  0.0.f1f4:0.0.f1f5                         >
  0.0.f1f6:0.0.f1f7                         >
  0.0.f1f8:0.0.f1f9                         >
  0.0.f1fa:0.0.f1fb                         >
  0.0.f1fc:0.0.f1fd                         >                              |
  0.0.f1fe:0.0.f1ff                         >                              v

                                [ Continue    ]
                                [ Back        ]
```

```
================================================================================
  Network connections
================================================================================
  Configure at least one interface this server can use to talk to other
  machines, and which preferably provides sufficient access for updates.

    NAME          TYPE  NOTES
  [ encc000       eth   -                 > ]
    72:00:bb:00:aa:11 / Unknown Vendor / Unknown Model
```

(continues on next page)

(continued from previous page)

```
  [ encc000.1234  vlan  -                > ]
    static        10.222.111.11/24
    VLAN 1234 on interface encc000

  [ Create bond > ]




                                  [ Continue   ]
                                  [ Back       ]
```

- Depending on the installer version you are using you may face a little bug here. In that case the button will be named 'Continue without network', but the network is there. If you see that, just ignore it and continue … (If you wait long enough the label will be refreshed and corrected.)

```
=============================================================================
  Configure proxy
=============================================================================
  If this system requires a proxy to connect to the internet, enter its
  details here.

  Proxy address:
                  If you need to use a HTTP proxy to access the outside world,
                  enter the proxy information here. Otherwise, leave this
                  blank.

                  The proxy information should be given in the standard form
                  of "http://[[user][:pass]@]host[:port]/".









                                  [ Done       ]
                                  [ Back       ]
```

```
=============================================================================
  Configure Ubuntu archive mirror
=============================================================================
  If you use an alternative mirror for Ubuntu, enter its details here.
```

(continues on next page)

(continued from previous page)

```
  Mirror address:  http://ports.ubuntu.com/ubuntu-ports
                   You may provide an archive mirror that will be used instead
                   of the default.









                                  [ Done       ]
                                  [ Back       ]
```

```
================================================================================
  Guided storage configuration
================================================================================
  Configure a guided storage layout, or create a custom one:

  (X)  Use an entire disk

       [ 0X1601       local disk 6.877G                                    v ]

       [ ]  Set up this disk as an LVM group

            [ ]  Encrypt the LVM group with LUKS

                         Passphrase:


                    Confirm passphrase:


  ( )  Custom storage layout

                                  [ Done       ]
                                  [ Back       ]
```

```
================================================================================
  Storage configuration
================================================================================
  FILE SYSTEM SUMMARY                                                        ^
                                                                             |
    MOUNT POINT     SIZE    TYPE      DEVICE TYPE                            |
```

(continues on next page)

(continued from previous page)

```
  [ /              6.875G  new ext4  new partition of local disk > ]        |
                                                                            |
                                                                            |
  AVAILABLE DEVICES                                                         |
                                                                            |
    No available devices                                                    |
                                                                            |
  [ Create software RAID (md) > ]                                           |
  [ Create volume group (LVM) > ]                                           |


  USED DEVICES
                                                                          v

                                  [ Done       ]
                                  [ Reset      ]
                                  [ Back       ]
```

```
================================================================================
  Storage configuration
================================================================================
  FILE SYSTEM SUMMARY                                                       ^
                                                                            |

    ┌──────────────────────── Confirm destructive action ────────────────────────┐
    │                                                                            │
    │  Selecting Continue below will begin the installation process and          │
    │  result in the loss of data on the disks selected to be formatted.         │
    │                                                                            │
    │  You will not be able to return to this or a previous screen once the      │
    │  installation has started.                                                 │
    │                                                                            │
    │  Are you sure you want to continue?                                        │
    │                                                                            │
    │                               [ No         ]                               │
    │                               [ Continue   ]                               │
    │                                                                            │
    └────────────────────────────────────────────────────────────────────────────┘

                                  [ Reset      ]
                                  [ Back       ]
```

```
================================================================================
  Profile setup
================================================================================
  Enter the username and password you will use to log in to the system. You
  can configure SSH access on the next screen but a password is still needed
  for sudo.
```

(continues on next page)

(continued from previous page)

```
             Your name:  Ed Example


   Your server's name:  s1lp11
                        The name it uses when it talks to other computers.

      Pick a username:  ubuntu


    Choose a password:  ********


Confirm your password:  ********


                                [ Done       ]
```

```
======================================================================
  SSH Setup
======================================================================
  You can choose to install the OpenSSH server package to enable secure remote
  access to your server.

                  [ ]  Install OpenSSH server


  Import SSH identity:  [ No             v ]
                        You can import your SSH keys from Github or Launchpad.

      Import Username:


                  [X]  Allow password authentication over SSH




                                [ Done       ]
                                [ Back       ]
```

- It's a nice and convenient new feature to add the user's SSH keys during the installation to the system, since that makes the system login password-less for the initial login!

```
======================================================================
  SSH Setup
======================================================================
  You can choose to install the OpenSSH server package to enable secure remote
  access to your server.
```

(continues on next page)

(continued from previous page)

```
                  [X]  Install OpenSSH server


  Import SSH identity:  [ from Launchpad v ]
                        You can import your SSH keys from Github or Launchpad.

   Launchpad Username:  user
                        Enter your Launchpad username.

                  [X]  Allow password authentication over SSH




                                  [ Done       ]
                                  [ Back       ]
```

```
================================================================================
  SSH Setup
================================================================================
  You can choose to install the OpenSSH server package to enable secure remote
  access to your server.

    ┌──────────────────────────── Confirm SSH keys ─────────────────────────────┐
    │                                                                           │
    │  Keys with the following fingerprints were fetched. Do you want to        │
    │  use them?                                                                │
    │                                                                           │
    │  2048 SHA256:joGscmiamcaoincinaäonnväineorviZEdDWdR9Hpbc2KIw user@W520  |
    │   (RSA)                                                                   │
    │  521 SHA256:T3JzxvB6K1Gzidvoidhoidsaoicak0jhhgvbw01F7/fZ2c                  |
    │  ed.example@acme.com  (ECDSA)                                        |
    │                                                                           │
    │                               [ Yes        ]                              │
    │                               [ No         ]                              │
    │                                                                           │
    └───────────────────────────────────────────────────────────────────────────┘

                                  [ Done       ]
                                  [ Back       ]
```

```
================================================================================
  Featured Server Snaps
================================================================================
  These are popular snaps in server environments. Select or deselect with
  SPACE, press ENTER to see more details of the package, publisher and
  versions available.
```

(continues on next page)

(continued from previous page)

```
  [ ] kata-containers Lightweight virtual machines that seamlessly plug into >
  [ ] docker          Docker container runtime                                >
  [ ] mosquitto       Eclipse Mosquitto MQTT broker                           >
  [ ] etcd            Resilient key-value store by CoreOS                     >
  [ ] stress-ng       A tool to load, stress test and benchmark a computer s >
  [ ] sabnzbd         SABnzbd                                                 >
  [ ] wormhole        get things from one computer to another, safely         >
  [ ] slcli           Python based SoftLayer API Tool.                        >
  [ ] doctl           DigitalOcean command line tool                          >
  [ ] keepalived      High availability VRRP/BFD and load-balancing for Linu >
  [ ] juju            Simple, secure and stable devops. Juju keeps complexit >



                                [ Done       ]
                                [ Back       ]
```

```
=============================================================================
  Install complete!
=============================================================================
  ┌──────────────────────────────────────────────────────────────────────────┐
  │          configuring raid (mdadm) service                               ^│
  │          installing kernel                                               │
  │          setting up swap                                                 │
  │          apply networking config                                        │
  │          writing etc/fstab                                              │
  │          configuring multipath                                           │
  │          updating packages on target system                              │
  │          configuring pollinate user-agent on target                      │
  │          updating initramfs configuration                                │
  │    finalizing installation                                               │
  │      running 'curtin hook'                                               │
  │        curtin command hook                                               │
  │    executing late commands                                               │
  │final system configuration                                                │
  │  configuring cloud-init                                                 ||
  │  installing openssh-server \                                            v│
  └──────────────────────────────────────────────────────────────────────────┘



                              [ View full log ]
```

```
=============================================================================
  Installation complete!
=============================================================================
  ┌──────────────────────────── Finished install! ─────────────────────────────┐
  │          apply networking config                                        ^│
  │          writing etc/fstab                                               │
  │          configuring multipath                                           │
```

(continues on next page)

(continued from previous page)

```
  |          updating packages on target system                                |
  |          configuring pollinate user-agent on target                        |
  |          updating initramfs configuration                                  |
  |    finalizing installation                                                 |
  |      running 'curtin hook'                                                 |
  |        curtin command hook                                                 |
  |    executing late commands                                                 |
  |final system configuration                                                  |
  |  configuring cloud-init                                                    |
  |  installing openssh-server                                                 |
  |  restoring apt configuration                                              ||
  |downloading and installing security updates                                v|
  └────────────────────────────────────────────────────────────────────────────┘

                                [ View full log ]
                                [ Reboot        ]
```

```
  Installation complete!
=========================================================================
  ┌──────────────────────────── Finished install! ─────────────────────────────┐
  |          apply networking config                                          ^|
  |          writing etc/fstab                                                 |
  |          configuring multipath                                             |
  |          updating packages on target system                                |
  |          configuring pollinate user-agent on target                        |
  |          updating initramfs configuration                                  |
  |    finalizing installation                                                 |
  |      running 'curtin hook'                                                 |
  |        curtin command hook                                                 |
  |    executing late commands                                                 |
  |final system configuration                                                  |
  |  configuring cloud-init                                                    |
  |  installing openssh-server                                                 |
  |  restoring apt configuration                                              ||
  |downloading and installing security updates                                v|
  └────────────────────────────────────────────────────────────────────────────┘

                                [ Connection to 10.222.111.11 closed by remote
host.                             [ Rebooting...  ]
Connection to 10.222.111.11 closed.
user@workstation:~$
```

- Now type `reset` to clear the screen and reset it to its defaults.
- Before proceeding one needs to remove the old, temporary host key of the target system, since it was only for use during the installation:

```
user@workstation:~$ ssh-keygen -f "/home/user/.ssh/known_hosts" -R "s1lp11"
# Host s1lp11 found: line 159
/home/user/.ssh/known_hosts updated.
```

(continues on next page)

(continued from previous page)

```
Original contents retained as /home/user/.ssh/known_hosts.old
user@workstation:~$
```

- And assuming the post-installation reboot is done, one can now login:

```
user@workstation:~$ ssh ubuntu@s1lp11
Warning: Permanently added the ECDSA host key for IP address
'10.222.111.11' to the list of known hosts.
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.4.0-42-generic s390x)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

  System information as of Wed 03 Jun 2020 05:50:05 PM UTC

  System load: 0.08              Memory usage: 2%   Processes:       157
  Usage of /:  18.7% of 6.70GB   Swap usage:   0%   Users logged in: 0

0 updates can be installed immediately.
0 of these updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@s1lp11:~$ uptime
 17:50:09 up 1 min,  1 user,  load average: 0.08, 0.11, 0.05
ubuntu@s1lp11:~$ lsb_release -a
No LSB modules are available.
Distributor ID:        Ubuntu
Description:        Ubuntu 20.04.5 LTS
Release:        20.04
Codename:        focal
ubuntu@s1lp11:~$ uname -a
Linux s1lp11 5.4.0-42-generic #30-Ubuntu SMP Wed Aug 05 16:57:22 UTC 2020
s390x s390x s390x GNU/Linux
ubuntu@s1lp11:~$ exit
logout
Connection to s1lp11 closed.
user@workstation:~$
```

Done !

### Non-interactive IBM Z LPAR autoinstall (s390x)

This non-interactive installation uses 'autoinstall', which can be considered the successor to the Debian installer (d-i) and preseed on Ubuntu. This is a detailed step-by-step guide, including output and logs (which are partially a bit shortened, as indicated by '…', to limit the size of this document).

The example logical partition (LPAR) here uses zFCP storage and is connected to a VLAN network. For a *DASD* and a non-VLAN network example, please see the non-interactive IBM z/VM (s390x) autoinstallation guide.

- Start with the preparation of the (FTP) install server (if it doesn't already exist).

```
user@local:~$ ssh admin@installserver.local
admin@installserver:~$ mkdir -p /srv/ftp/ubuntu-daily-live-server-20.04
admin@installserver:~$ wget http://cdimage.ubuntu.com/ubuntu-server/focal/
daily-live/current/focal-live-server-s390x.iso --directory-prefix=/srv/ftp/
ubuntu-daily-live-server-20.04
--2020-06-26 12:18:48--  http://cdimage.ubuntu.com/ubuntu-server/focal/daily-
live/current/focal-live-server-s390x.iso
Resolving cdimage.ubuntu.com (cdimage.ubuntu.com)... 2001:67c:1560:8001::1d,
2001:67c:1360:8001::28, 2001:67c:1360:8001::27,
 ...
Connecting to cdimage.ubuntu.com (cdimage.ubuntu.
com)|2001:67c:1560:8001::1d|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 700952576 (668M) [application/x-iso9660-image]
Saving to: 'focal-live-server-s390x.iso'

focal-live-server-s 100%[===================>] 668.48M  2.73MB/s    in 4m 54s

2020-06-26 12:23:42 (2.27 MB/s) - 'focal-live-server-s390x.iso' saved
[700952576/700952576]
admin@installserver:~$
```

- The ISO image needs to be extracted now. Since files in its boot folder need to be modified, loopback mount is not an option here:

```
admin@installserver:~$ cd /srv/ftp/ubuntu-daily-live-server-20.04
admin@installserver:/srv/ftp/ubuntu-daily-live-server-20.04$ mkdir iso
admin@installserver:/srv/ftp/ubuntu-daily-live-server-20.04$ sudo mount -o
loop ./focal-live-server-s390x.iso ./iso
[sudo] password for admin:
mount: /home/user/iso-test/iso: WARNING: device write-protected, mounted
read-only.
admin@installserver:/srv/ftp/ubuntu-daily-live-server-20.04$ ls -l
total 684530
-rw-rw-r--  1 user user 700952576 Jun 26 10:12 focal-live-server-s390x.iso
dr-xr-xr-x 10 root    root         2048 Jun 26 10:12 iso
admin@installserver:/srv/ftp/ubuntu-daily-live-server-20.04$ rsync -rtvz ./
iso/ . && sync
sending incremental file list
```

(continues on next page)

(continued from previous page)

```
skipping non-regular file "ubuntu"
skipping non-regular file "ubuntu-ports"
./
README.diskdefines
boot.catalog
md5sum.txt
ubuntu.ins
skipping non-regular file "dists/stable"
skipping non-regular file "dists/unstable"
.disk/
.disk/base_installable
.disk/casper-uuid-generic
.disk/cd_type
.disk/info
boot/
boot/README.boot
boot/initrd.off
boot/initrd.siz
boot/initrd.ubuntu
boot/kernel.ubuntu
boot/parmfile.ubuntu
boot/ubuntu.exec
boot/ubuntu.ikr
boot/ubuntu.ins
casper/
  ...
sent 681,509,758 bytes  received 1,857 bytes  22,344,643.11 bytes/sec
total size is 700,317,941  speedup is 1.03
admin@installserver:/srv/ftp/ubuntu-daily-live-server-20.04$ ls -l
total 684578
dr-xr-xr-x  2 user user      4096 Jun 26 10:12 boot
-r--r--r--  1 user user      2048 Jun 26 10:12 boot.catalog
dr-xr-xr-x  3 user user      4096 Jun 26 10:12 casper
dr-xr-xr-x  3 user user      4096 Jun 26 10:11 dists
-rw-rw-r--  1 user user 700952576 Jun 26 10:12 focal-live-server-s390x.iso
dr-xr-xr-x  2 user user      4096 Jun 26 10:11 install
dr-xr-xr-x 10 root    root         2048 Jun 26 10:12 iso
-r--r--r--  1 user user      4944 Jun 26 10:12 md5sum.txt
dr-xr-xr-x  2 user user      4096 Jun 26 10:11 pics
dr-xr-xr-x  3 user user      4096 Jun 26 10:11 pool
dr-xr-xr-x  2 user user      4096 Jun 26 10:11 preseed
-r--r--r--  1 user user       236 Jun 26 10:11 README.diskdefines
-r--r--r--  1 user user       185 Jun 26 10:12 ubuntu.ins
```

- Now create ins and parm files dedicated to the LPAR that will be installed (here `zlinl-par`), based on the default ins and parm files that are shipped with the ISO image:

```
admin@installserver:/srv/ftp/ubuntu-daily-live-server-20.04$ chmod -R +rw ./
boot
```

(continues on next page)

(continued from previous page)

```
admin@installserver:/srv/ftp/ubuntu-daily-live-server-20.04$ cp ./boot/
ubuntu.ins ./boot/ubuntu_zlinlpar.ins
admin@installserver:/srv/ftp/ubuntu-daily-live-server-20.04$ cp ./boot/
parmfile.ubuntu ./boot/parmfile.zlinlpar
admin@installserver:/srv/ftp/ubuntu-daily-live-server-20.04$
```

```
admin@installserver:/srv/ftp/ubuntu-daily-live-server-20.04$ vi ./boot/
ubuntu_zlinlpar.ins
admin@installserver:/srv/ftp/ubuntu-daily-live-server-20.04$ cat ./boot/
ubuntu_zlinlpar.ins
* Ubuntu for z Series (default kernel)
kernel.ubuntu 0x00000000
initrd.off 0x0001040c
initrd.siz 0x00010414
parmfile.zlinlpar 0x00010480
initrd.ubuntu 0x01000000
admin@installserver:~$
```

```
admin@installserver:/srv/ftp/ubuntu-daily-live-server-20.04$ vi ./boot/
parmfile.zlinlpar
admin@installserver:/srv/ftp/ubuntu-daily-live-server-20.04$ cat ./boot/
parmfile.zlinlpar
```

```
ip=10.11.12.42::10.11.12.1:255.255.255.0:zlinlpar:encc000.4711:none:10.11.12.
1 vlan=encc000.4711:encc000 url=http://installserver.local:80/ubuntu-daily-
live-server-20.04/focal-live-server-s390x.iso autoinstall ds=nocloud-net;
s=http://installserver.local:80/autoinstall/zlinlpar/ --- quiet
```

- Now make sure an FTP server is running in the *installserver* with `/srv/ftp` as ftp-server root (as used in this example).
- Now prepare an *autoinstall* (HTTP) server, which hosts the configuration data for the non-interactive installation.

```
admin@installserver:/srv/ftp/ubuntu-daily-live-server-20.04$ mkdir -p /srv/
www/autoinstall/zlinlpar
admin@installserver:/srv/ftp/ubuntu-daily-live-server-20.04$ cd /srv/www/
autoinstall/zlinlpar
admin@installserver:/srv/www/autoinstall/zlinlpar$
admin@installserver:/srv/www/autoinstall/zlinlpar$ echo "instance-id:
$(uuidgen || openssl rand -base64 8)" > meta-data
admin@installserver:/srv/www/autoinstall/zlinlpar$ cat meta-data
instance-id: 2c2215fb-6a38-417f-b72f-376b1cc44f01
admin@installserver:/srv/www/autoinstall/zlinlpar$
```

```
admin@installserver:/srv/www/autoinstall/zlinlpar$ vi user-data
admin@installserver:/srv/www/autoinstall/zlinlpar$ cat user-data
#cloud-config
autoinstall:
```

(continues on next page)

(continued from previous page)

```
  version: 1
  refresh-installer:
    update: yes
  reporting:
    builtin:
      type: print
  apt:
    preserve_sources_list: false
    primary:
    - arches: [amd64, i386]
      uri: http://archive.ubuntu.com/ubuntu
    - arches: [default]
      uri: http://ports.ubuntu.com/ubuntu-ports
  keyboard:
    layout: en
    variant: us
  locale: en_US
  identity:
    hostname: zlinlpar
    password:
"$6$ebJ1f8wxED22bTL4F46P0"
      username: ubuntu
    user-data:
      timezone: America/Boston
      users:
        - name: ubuntu
          password:
"$6$KwuxED22bTL4F46P0"
          lock_passwd: false
    early-commands:
      - touch /tmp/lets_activate_the_s390x_devices
      - chzdev zfcp -e e000
      - chzdev zfcp -e e100
      - chzdev zfcp-lun -e --online
      - touch /tmp/s390x_devices_activation_done
    network:
      ethernets:
        encc000: {}
      version: 2
      vlans:
        encc000.4711:
          addresses: [10.11.12.42/24]
          gateway4: 10.11.12.1
          id: 4711
          link: encc000
          nameservers:
            addresses: [10.11.12.1]
    ssh:
      install-server: true
```

(continues on next page)

(continued from previous page)

```
      allow-pw: true
      authorized-keys: ['ssh-rsa  meQwtZ user@workstation # ssh-import-id
lp:user']
  admin@installserver:~$
```

- For s390x installations the `early-commands` section is the interesting part:

```
early-commands:
  - touch /tmp/lets_activate_the_s390x_devices
  - chzdev zfcp -e e000
  - chzdev zfcp -e e100
  - chzdev zfcp-lun -e --online
  - touch /tmp/s390x_devices_activation_done
```

  The first and last `early-commands` are optional; they only frame and indicate the real s390x command activation.

  In this particular example, two zFCP hosts (host-bus-adapters) are enabled via their addresses *e000* (`chzdev zfcp -e e000`) and *e100* (`chzdev zfcp -e e000`). These have certain logical unit numbers (LUNs) assigned that are all automatically discovered and activated by `chzdev zfcp-lun -e --online`.

  Activation of a direct-access storage device (DASD) would look like this: `chzdev dasd -e 1f00`, and a qeth device activation looks like: `chzdev qeth -e c000`.

**See also**

For more details about the autoinstall config options, please have a look at the autoinstall reference and autoinstall schema pages.

- Now make sure a HTTP server is running with `/srv/www` as web-server root (in this particular example).
- Next steps need to be done at the *hardware management console (HMC)*. First, connect to the HMC and proceed with the 'Load From Removable Media and Server' task.
- Then, start the 'Load from Removable Media or Server' task under 'Recovery' –> 'Load from Removable Media or Server' on your specific LPAR that you are going to install, and fill out the following fields (the contents will be of course different on your system):

```
FTP Source
Host computer:        installserver.local
User ID:        ftpuser
Password: ********
Account (optional):
File location (optional): ubuntu-daily-live-server-20.04/boot
```

- Now confirm the entered data and click 'OK'.
- At the 'Load from Removable Media or Server - Select Software to Install' screen, choose the LPAR that is going to be installed, here:

```
ubuntu-daily-live-server-20.04/boot/ubuntu_zlinlpar.ins   Ubuntu for z Series
(default kernel)
```

- Confirm again with 'OK'.
- And another confirmation about the 'Load will cause jobs to be cancelled'.
- Then, another 'Yes' – understanding that it's a disruptive task:

```
Disruptive Task Confirmation : Load from Removable Media or Server
```

- Now monitor the 'Load from Removable media or Server Progress' screen and confirm it once again when the status changes from 'Please wait while the image is being loaded.' to 'Success'.
- Then navigate to 'Daily' –> 'Operating System Messages' to monitor the initial program load (IPL) of the install system …

```
Message
chzdev: Unknown device type or device ID format: c000.4711
Use 'chzdev --help' for more information
QETH device 0.0.c000:0.0.c001:0.0.c002 configured
IP-Config: encc000.4711 hardware address 1a:3c:99:55:2a:ef mtu 1500
IP-Config: encc000.4711 guessed broadcast address 10.11.12.255
IP-Config: encc000.4711 complete:
address: 10.11.12.42    broadcast: 10.11.12.255   netmask: 255.255.255.0

gateway: 10.11.12.1     dns0     : 10.11.12.1     dns1   : 0.0.0.0

host   : zlinlpar
rootserver: 0.0.0.0 rootpath:
filename  :
Connecting to installserver.local:80 (installserver.local:80)
focal-live-server-s3   5% |*                               | 39.9M  0:00:15
ETA
focal-live-server-s3  22% |*******                         |  147M  0:00:07
ETA
focal-live-server-s3  38% |************                    |  254M  0:00:04
ETA
focal-live-server-s3  53% |*****************               |  355M  0:00:03
ETA
focal-live-server-s3  67% |*********************           |  453M  0:00:02
ETA
focal-live-server-s3  81% |*************************       |  545M  0:00:01
ETA
focal-live-server-s3  94% |******************************  |  633M  0:00:00
ETA
focal-live-server-s3 100% |********************************|  668M  0:00:00 ETA
mount: mounting /cow on /root/cow failed: No such file or directory
Connecting to plymouth: Connection refused
passwd: password expiry information changed.
Using CD-ROM mount point /cdrom/
```

(continues on next page)

(continued from previous page)

```
Identifying... [5d25356068b713167814807dd678c261-2]
Scanning disc for index files...
Found 2 package indexes, 0 source indexes, 0 translation indexes and 1
signature
Found label 'Ubuntu-Server 20.04 LTS _Focal Fossa_ - Release s390x (20200616)
'
This disc is called:
'Ubuntu-Server 20.04 LTS _Focal Fossa_ - Release s390x (20200616)'
...
[   61.190916] cloud-init[2076]: Cloud-init v. 20.1-10-g71af48df-0ubuntu5
running
'modules:final' at Fri, 26 Jun 2020 11:02:01 +0000. Up 61.09 seconds.
[   61.191002] cloud-init[2076]: ci-info: no authorized SSH keys fingerprints
fo
und for user installer.
[   61.191071] cloud-init[2076]: Cloud-init v. 20.1-10-g71af48df-0ubuntu5
finished at Fri, 26 Jun 2020 11:02:01 +0000.
Datasource DataSourceNoCloudNet [seed=cmdline,
/var/lib/cloud/seed/nocloud,
http://installserver.local:80/autoinstall/zlinlpar/]
[dsmode=net].  Up 61.18 seconds
[   61.191136] cloud-init[2076]: Welcome to Ubuntu Server Installer!
[   61.191202] cloud-init[2076]: Above you will find SSH host keys and a
random
password set for the `installer` user. You can use these credentials to ssh-
in
and complete the installation. If you provided SSH keys in the cloud-init
datasource,
they were also provisioned to the installer user.
[   61.191273] cloud-init[2076]: If you have access to the graphical console,
like TTY1 or HMC ASCII terminal you can complete the installation there too.

It is possible to connect to the installer over the network, which
might allow the use of a more capable terminal.

To connect, SSH to installer@10.11.12.42.

The password you should use is ''i7UFdP8fhiVVMme3qqH8''.

The host key fingerprints are:

RSA     SHA256:rBXLeUke3D4gKdsruKEajHjocxc9hr3PI
ECDSA   SHA256:KZZYFswtKxFXWQPuQS9QpOBUoS6RHswis
ED25519 SHA256:s+5tZfagx0zffC6gYRGW3t1KcBH6f+Vt0

Ubuntu 20.04 LTS ubuntu-server sclp_line0
```

- At short notice, you can even log in to the system with the user ‘installer’ and the temporary password that was given at the end of the boot-up process (see above) of the installation system:

```
user@workstation:~$ ssh installer@zlinlpar
The authenticity of host 'zlinlpar (10.11.12.42)' can't be established.
ECDSA key fingerprint is SHA256:O/dU/D8jJAEGQcbqKGE9La24IRxUPLpzzs5li9F6Vvk.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'zlinlpar,10.11.12.42' (ECDSA) to the list of
known hosts.
installer@zlinlpar's password:
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-37-generic s390x)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

  System information as of Fri Jun 26 11:08:18 UTC 2020

  System load:    1.25      Memory usage: 4%   Processes:       192
  Usage of /home: unknown   Swap usage:   0%   Users logged in: 0

0 updates can be installed immediately.
0 of these updates are security updates.


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

the installer running on /dev/tty1 will perform the autoinstall

press enter to start a shell
```

- Notice that it informs you about a currently-running autoinstall process:

```
the installer running on /dev/tty1 will perform the autoinstall
```

- Nevertheless, we can quickly check some things – though only until the autoinstall process has finished and the post-install reboot has been triggered:

```
root@ubuntu-server:/# ls -l /tmp/lets_activate_the_s390x_devices
-rw-r--r-- 1 root root 0 Jun 26 11:08 /tmp/lets_activate_the_s390x_devices
-rw-r--r-- 1 root root 0 Jun 26 11:09 /tmp/s390x_devices_activation_done

root@ubuntu-server:/# lszdev | grep yes
zfcp-host    0.0.e000                                       yes  yes
zfcp-host    0.0.e100                                       yes  yes
```

(continues on next page)

(continued from previous page)

```
zfcp-lun     0.0.e000:0x50050763060b16b6:0x4026400200000000  yes  yes   sdb
sg1
zfcp-lun     0.0.e000:0x50050763061b16b6:0x4026400200000000  yes  yes   sda
sg0
zfcp-lun     0.0.e100:0x50050763060b16b6:0x4026400200000000  yes  yes   sdd
sg3
zfcp-lun     0.0.e100:0x50050763061b16b6:0x4026400200000000  yes  yes   sdc
sg2
qeth         0.0.c000:0.0.c001:0.0.c002                      yes  no
encc000
root@ubuntu-server:/#
```

- If you wait long enough you'll see the remote session get closed:

```
root@ubuntu-server:/# Connection to zlinlpar closed by remote host.
Connection to zlinlpar closed.
user@workstation:~$
```

- As well as at the console:

```
ubuntu-server login:

[[0;1;31mFAILED[0m] Failed unmounting [0;1;39m/cdrom[0m.
[  169.161139] sd-umoun[15600]: Failed to unmount /oldroot: Device or
resource busy
[  169.161550] sd-umoun[15601]: Failed to unmount /oldroot/cdrom: Device or
resource busy
[  169.168118] shutdown[1]: Failed to finalize  file systems, loop devices,
ignoring
Total: 282 Selected: 0

Command:
```

- …and that the post-install reboot gets triggered:

```
Message
Mounting [0;1;39mKernel Configuration File System[0m...
Starting [0;1;39mApply Kernel Variables[0m...
[[0;32m  OK  [0m] Finished [0;1;39mRemount Root and Kernel File Systems[0m.
[[0;32m  OK  [0m] Finished [0;1;39mUncomplicated firewall[0m.
[[0;32m  OK  [0m] Mounted [0;1;39mFUSE Control File System[0m.
[[0;32m  OK  [0m] Mounted [0;1;39mKernel Configuration File System[0m.
...
[   35.378928] cloud-init[2565]: Cloud-init v. 20.1-10-g71af48df-0ubuntu5
runnin
g 'modules:final' at Fri, 26 Jun 2020 11:10:44 +0000. Up 35.29 seconds.
[   35.378978] cloud-init[2565]: Cloud-init v. 20.1-10-g71af48df-0ubuntu5
finish
ed at Fri, 26 Jun 2020 11:10:44 +0000. Datasource DataSourceNone.  Up 35.37
seconds
```

(continues on next page)

(continued from previous page)

```
[   35.379008] cloud-init[2565]: 2020-06-26 11:10:44,359 - cc_final_message.
py[W
ARNING]: Used fallback datasource
[[0;32m  OK  [0m] Finished [0;1;39mExecute cloud user/final scripts[0m.
[[0;32m  OK  [0m] Reached target [0;1;39mCloud-init target[0m.

zlinlpar login:
```

- With the completion of the reboot, the autoinstall is finished and the LPAR is ready to use:

```
user@workstation:~$ ssh-keygen -f "/home/user/.ssh/known_hosts" -R "zlinlpar"
# Host zlinlpar found: line 163
/home/user/.ssh/known_hosts updated.
Original contents retained as /home/user/.ssh/known_hosts.old
user@workstation:~$ ssh ubuntu@zlinlpar
The authenticity of host 'zlinlpar (10.11.12.42)' can't be established.
ECDSA key fingerprint is SHA256:iGtCArEg+ZnoZlgtOkvmyy0gPY8UEI+f7zoISOF+m/0.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'zlinlpar,10.11.12.42' (ECDSA) to the list of
known hosts.
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-39-generic s390x)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

  System information as of Fri 26 Jun 2020 11:12:23 AM UTC

  System load: 0.21               Memory usage: 3%   Processes:       189
  Usage of /:  28.2% of 30.88GB   Swap usage:   0%   Users logged in: 0

10 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY,
to the extent permitted by applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@zlinlpar:~$ uptime
 11:12:35 up 2 min,  1 user,  load average: 0.18, 0.17, 0.08
ubuntu@zlinlpar:~$ lsb_release -a
```

(continues on next page)

(continued from previous page)

```
No LSB modules are available.
Distributor ID:        Ubuntu
Description:        Ubuntu 20.04 LTS
Release:        20.04
Codename:        focal
ubuntu@zlinlpar:~$ uname -a
Linux zlinlpar 5.4.0-39-generic #43-Ubuntu SMP Fri Jun 19 10:27:17
UTC 2020 s390x s390x s390x
GNU/Linux
ubuntu@zlinlpar:~$ lszdev | grep yes
zfcp-host    0.0.e000                                          yes  yes
zfcp-host    0.0.e100                                          yes  yes
zfcp-lun     0.0.e000:0x50050763060b16b6:0x4026400200000000  yes  yes
sdb sg1
zfcp-lun     0.0.e000:0x50050763061b16b6:0x4026400200000000  yes  yes
sda sg0
zfcp-lun     0.0.e100:0x50050763060b16b6:0x4026400200000000  yes  yes
sdc sg2
zfcp-lun     0.0.e100:0x50050763061b16b6:0x4026400200000000  yes  yes
sdd sg3
qeth         0.0.c000:0.0.c001:0.0.c002                        yes  yes
encc000
ubuntu@zlinlpar:~$ exit
logout
Connection to zlinlpar closed.
user@workstation:~$
```

### Some closing notes

- It's always best to use the latest installer and autoinstall components. Be sure to update the installer to the most recent version, or just use a current daily live-server image.
- The ISO image specified with the kernel parameters needs to fit in the boot folder. Its kernel and initrd are specified in the 'Load from Removable Media and Server' task at the hardware management console (HMC).
- In addition to activating disk storage resources in `early-commands`, other devices like OSA/qeth can be added and activated there, too. This is not needed for the basic network device, as specified in the kernel parameters used for the installation (that one is automatically handled).
- If everything is properly set up – FTP server for the image, HTTP server for the autoinstall config files – the installation can be as quick as 2 to 3 minutes. Of course this depends on the complexity of the autoinstall YAML file.
- There is a simple way of generating a sample autoinstall YAML file; one can perform an interactive Subiquity installation, grab the file `/var/log/installer/autoinstall-user-data`, and use it as an example – but beware that the `early-commands` entries to activate the s390x-specific devices need to be added manually!

Our list of *installation guides* contains installation instructions for a variety of architecture-specific and advanced setups. For a general installation, or if you're just getting started with

Ubuntu, you may prefer to use our *basic installation* tutorial.

# 2.2. Security

## 2.2.1. Security

While a fresh Ubuntu installation is usually safe for immediate use, there are some additional steps you can take to introduce a layered approach to your system's security. If you are new to Ubuntu, you may want to refer to our *Introduction to security* first for a general overview.

### General configuration

- *Users and groups management* for setting up user accounts, permissions and password policies
- *Firewalls* are recommended for network security
- *AppArmor* limits permissions and access for the software running on your system
- *Console security* for an additional physical security barrier

### User management

User management is a critical part of maintaining a secure system. Ineffective user and privilege management often leads to a system being compromised. Therefore, it is important that you understand how to protect your server through simple and effective user account management techniques.

### Where is root?

Ubuntu developers decided to disable the administrative root account by default in all Ubuntu installations. This does not mean that the root account has been deleted, or that it may not be accessed. Instead, it has been given a password hash that matches no possible value, and so may not log in directly by itself.

Instead, the `sudo` utility ("superuser do") is used to carry out system administrative duties. `sudo` allows an authorised user to temporarily elevate their privileges using their own password instead of having to know the password belonging to the root account. This provides accountability for all user actions, and gives the administrator control over which actions a user can perform with said privileges.

### sudo-rs

From Ubuntu 25.10 (Questing Quokka) onward, `sudo` is provided by the `sudo-rs` package (a rust implementation). While `sudo-rs` is not 100% compatible with `sudo.ws`, the majority of common use cases are supported and the change should be invisible to most users.

A list of key differences is *maintained as a reference*

The original `sudo` utility (maintained by Todd C. Miller), `sudo.ws` remains supported in the 25.10 and the subsequent 26.04 LTS. You will find these utilities installed with `.ws` suffix (for example `sudo.ws`, `visudo.ws`, etc.). Switching the default back to `sudo.ws` is not recommended, but if you need to do so, follow these steps:

Interactive:

```
# update-alternatives --config sudo
```

Non-interactive:

```
# update-alternatives --set sudo /usr/bin/sudo.ws
```

You can always switch back to `sudo-rs` using:

```
# update-alternatives --set sudo /usr/lib/cargo/bin/sudo
```

You can learn more about the motivation for this change in the blog post Adopting sudo-rs By Default in Ubuntu 25.10

The rest of the article should work the same with both `sudo.ws` and `sudo-rs`.

### Enabling the root account

If for some reason you wish to enable the root account, you will need to give it a password:

```
sudo passwd
```

`sudo` will prompt you for your password, and then ask you to supply a new password for `root` as shown below:

```
[sudo: authenticate] Password: (enter your own password)
New password: (enter a new password for root)
Retype new password: (repeat new password for root)
passwd: password updated successfully
```

### Disabling the root account password

To disable the root account password, use the following `passwd` syntax:

```
sudo passwd -l root
```

You can learn more about `sudo` by reading the `sudo(8)` manual page: `man sudo`

By default, the initial user created by the Ubuntu installer is a member of the group `sudo` which is added to the file `/etc/sudoers` as an authorised `sudo` user. To give any other account full root access through `sudo`, add them to the `sudo` group.

### Listing, adding, and deleting local users

Managing local users and groups differs very little from most other *GNU*/Linux operating systems. Ubuntu and other Debian-based distributions encourage the use of the `adduser` package for account management.

Local users and groups are those defined in the `/etc/passwd` and `/etc/group` files, respectively. Via the Name Service Switch (*NSS*) mechanism, however, user and group accounts can also come from other sources, like remote Active Directory or Samba servers, OpenLDAP directory services, and more. Managing accounts in those services in an entirely different matter, and not covered in this document.

### Listing local users

The local users are defined one per line in the `/etc/passwd` text file. The `cat /etc/passwd` command can be used to list its contents (only a few lines shown below, for brevity):

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
...
ubuntu:x:1000:1000:Ubuntu:/home/ubuntu:/bin/bash
```

> **Tip**
>
> The structure of the `/etc/passwd` file is explained in the `passwd(5)` manual page.

There are basically three types of local users:

- system pre-installed users: These are system users essential to a working Linux system, like `root`, `man`, `lp`, and others. Their *UID*s are in the range between 0 and 99.
- system users: These users are created dynamically, as services are installed. The UID range here is between 100 and 999.
- Regular users: Finally, the remaining range is usable for "real" users, generally representing persons using the system. This UID range starts at 1000 and, by convention, ends at 59999.

> **Tip**
>
> For more details about these UID ranges, including groups and other configuration options for adding users, check the `adduser.conf.5` manual page.

To list all local usernames, we can list the contents of `/etc/passwd` with some string manipulation. For example:

```
cat /etc/passwd | cut -d : -f 1
```

This command will cut each line at the first column, using the ":" character as the delimiter, and result in an output showing all local users:

```
root
daemon
bin
sys
...
```

If you want to only show regular users, that is, users with an UID greater than 1000, we can use a regular expression for this filtering:

```
cat /etc/passwd | cut -d : -f 1,3 | grep -E ':[0-9]{4}+'
```

The regular expression `:[0-9]{4}+` means all digits, repeated 4 times or more. The output will be similar to this:

```
nobody:65534
ubuntu:1000
```

> **Tip**
>
> By convention, the `nobody` users has an UID of 65534. That originated back when the user range was limited to 16 bits (so, 0-65535), and is kept for compatibility reasons.

### A note about remote users

There is another command that, at first glance, looks like it lists all local users as well: `getent passwd`. What this tool is doing, however, is listing all entries from the `passwd` database, and this will include any NSS modules that might be installed and configured.

On a default installation of Ubuntu, with no extra configuration, that will indeed list only the local users, but not in the general case. If you have, for example, `libnss-ldap` installed and configured, the output of `getent passwd` will list all local users, **plus all the remote users** it finds in the LDAP server that it was configured with.

Performing such an exhaustive enumeration is frowned upon, because that can incur a large amount of network traffic, and place an unreasonable burden on the remote server. In fact, most NSS services block such enumeration actions by default.

For more information on the Name Service Switch (NSS) mechanism, please consult the `nss(5)` and `nsswitch.conf(5)` manual pages.

### Add a user

To add a user account, use the following syntax, and follow the prompts to give the account a password and identifiable characteristics, such as a full name, phone number, etc:

```
sudo adduser username
```

### Delete a user

To delete a user account and its primary group, use the following syntax:

```
sudo deluser username
```

Deleting an account does not remove their respective home folder. You must decide whether or not to delete the folder manually, or to keep it according to your desired retention policies.

Remember, any user added later with the same UID/GID as the previous owner will now have access to this folder if you have not taken the necessary precautions.

You may want to change the UID/GID values to something more appropriate, such as the root account, and perhaps even relocate the folder to avoid future conflicts:

```
sudo chown -R root:root /home/username/
sudo mkdir /home/archived_users/
sudo mv /home/username /home/archived_users/
```

### Lock or unlock a password

To temporarily lock a user password, use the following syntax:

```
sudo passwd -l username
```

Similarly, to unlock a user password:

```
sudo passwd -u username
```

### Add or delete a group

To add or delete a personalised group, use the following syntax, respectively:

```
sudo addgroup groupname
sudo delgroup groupname
```

### Add a user to a group

To add a user to a group, use the following syntax:

```
sudo adduser username groupname
```

### User profile security

When a new user is created, the `adduser` utility creates a brand new home directory named `/home/username`. The default profile is modelled after the contents found in the directory of `/etc/skel`, which includes all profile basics.

If your server will be home to multiple users, you should pay close attention to the user home directory permissions to ensure confidentiality. By default, user home directories in Ubuntu are created with world read/execute permissions. This means that all users can browse and access the contents of other users home directories, which may not be suitable for your environment.

To verify your current user home directory permissions, use the following syntax:

```
ls -ld /home/username
```

The following output shows that the directory `/home/username` has world-readable permissions:

```
drwxr-xr-x  2 username username    4096 2007-10-02 20:03 username
```

You can remove the world readable-permissions using the following command:

```
sudo chmod 0750 /home/username
```

**Note**

Some people use the recursive option (`-R`) indiscriminately, which modifies all child folders and files. However, this is not necessary and may have undesirable/unintended consequences. Modifying only the parent directory is enough to prevent unauthorised access to anything below the parent.

A more efficient approach would be to modify the `adduser` global default permissions when creating user home folders. To do this, edit the `/etc/adduser.conf` file and modify the `DIR_MODE` variable to something appropriate, so that all new home directories will receive the correct permissions.

```
DIR_MODE=0750
```

After correcting the directory permissions using any of the previously mentioned techniques, verify the results as follows:

```
ls -ld /home/username
```

The output below shows that world-readable permissions have been removed:

```
drwxr-x---   2 username username    4096 2007-10-02 20:03 username
```

### Password policy

A strong password policy is one of the most important aspects of your security posture. Many successful security breaches involve simple **brute force** and **dictionary** attacks against weak passwords.

If you intend to offer any form of remote access involving your local password system, make sure you address minimum password complexity requirements, maximum password lifetimes, and frequent audits of your authentication systems.

### Minimum password length

By default, Ubuntu requires a minimum password length of 6 characters, as well as some basic entropy checks. These values are controlled in the file `/etc/pam.d/common-password`, which is outlined below.

```
password        [success=1 default=ignore]      pam_unix.so obscure sha512
```

To adjust the minimum length to 8 characters, change the appropriate variable to `minlen=8`. The modification is outlined below:

```
password        [success=1 default=ignore]      pam_unix.so obscure sha512
minlen=8
```

**Note**

Basic password entropy checks and minimum length rules do not apply to the administrator using `sudo`-level commands to setup a new user.

### Password expiration

When creating user accounts, you should make it a policy to have a minimum and maximum password age, forcing users to change their passwords when they expire.

To view the current status of a user account:

```
sudo chage -l username
```

The output below shows interesting facts about the user account, namely that there are no policies applied:

```
Last password change                                    : Jan 20, 2015
Password expires                                        : never
Password inactive                                       : never
Account expires                                         : never
Minimum number of days between password change          : 0
Maximum number of days between password change          : 99999
Number of days of warning before password expires       : 7
```

To set any of these values, use the `chage` ("change age") command, and follow the interactive prompts:

```
sudo chage username
```

The following is also an example of how you can manually change the explicit expiration date (`-E`) to 01/31/2015, minimum password age (`-m`) of 5 days, maximum password age (`-M`) of 90 days, inactivity period (`-I`) of 30 days after password expiration, and a warning time period (`-W`) of 14 days before password expiration:

sudo chage -E 01/31/2015 -m 5 -M 90 -I 30 -W 14 username

To verify changes, use the same syntax as mentioned previously:

```
sudo chage -l username
```

The output below shows the new policies that have been established for the account:

```
Last password change                                    : Jan 20, 2015
Password expires                                        : Apr 19, 2015
Password inactive                                       : May 19, 2015
Account expires                                         : Jan 31, 2015
Minimum number of days between password change          : 5
Maximum number of days between password change          : 90
Number of days of warning before password expires       : 14
```

### Other security considerations

Many applications use alternate authentication mechanisms that can be easily overlooked by even experienced system administrators. Therefore, it is important to understand and control how users authenticate and gain access to services and applications on your server.

### SSH access by disabled passwords

Disabling or locking a user password will not prevent a user from logging into your server remotely if they have previously set up SSH public key authentication. They will still be able to gain shell access to the server, without the need for any password. Remember to check the user's home directory for files that will allow for this type of authenticated SSH access, e.g. `/home/username/.ssh/authorized_keys`.

Remove or rename the directory `.ssh/` in the user's home folder to prevent further SSH authentication access.

Be sure to check for any established SSH connections by the disabled account, as it is possible they may have existing inbound or outbound connections – then `pkill` any that are found.

```
who | grep username  (to get the pts/# terminal)
sudo pkill -f pts/#
```

Restrict SSH access to only user accounts that should have it. For example, you may create a group called `sshlogin` and add the group name as the value associated with the `AllowGroups` variable located in the file `/etc/ssh/sshd_config`:

```
AllowGroups sshlogin
```

Then add your permitted SSH users to the group `sshlogin`, and restart the SSH service.

```
sudo adduser username sshlogin
sudo systemctl restart ssh.service
```

### External user database authentication

Most enterprise networks require centralised authentication and access controls for all system resources. If you have configured your server to authenticate users against external databases, be sure to disable the user accounts both externally and locally. This way you ensure that local *fallbacks* authentication is not possible.

### Firewall

The Linux kernel includes the **Netfilter** subsystem, which is used to manipulate or decide the fate of network traffic headed into or through your server. All modern Linux firewall solutions use this system for packet filtering.

The kernel's packet filtering system would be of little use to administrators without a userspace interface to manage it. This is the purpose of the `iptables` utility: when a packet reaches your server, it will be handed off to the Netfilter subsystem for acceptance, manipulation, or rejection based on the rules supplied to it from the userspace (via `iptables`). Thus, `iptables` is all you need to manage your firewall, if you're familiar with it, but many *frontends* are available to simplify the task. We'll take a look at the default frontend used in Ubuntu here.

### ufw - Uncomplicated Firewall

The default firewall configuration tool for Ubuntu is `ufw`. Developed to ease `iptables` firewall configuration, `ufw` provides a user-friendly way to create an IPv4 or IPv6 host-based firewall.

`ufw` by default is initially disabled. From the `ufw(8)` manual page:

> ufw is not intended to provide complete firewall functionality via its command interface, but instead provides an easy way to add or remove simple rules. It is currently mainly used for host-based firewalls.

#### Enable or disable ufw

To enable `ufw`, run the following command from a terminal prompt:

```
sudo ufw enable
```

To disable it, you can use the following command:

```
sudo ufw disable
```

#### Open or close a port

To open a port (SSH in this case):

```
sudo ufw allow 22
```

Similarly, to close an opened port:

```
sudo ufw deny 22
```

#### Add or remove a rule

Rules can also be added using a **numbered** format:

```
sudo ufw insert 1 allow 80
```

To view the numbered format:

```
sudo ufw status numbered
```

To remove a rule, use `delete` followed by the rule:

```
sudo ufw delete deny 22
```

#### Allow access from specific hosts

It is possible to allow access from specific hosts or networks to a port. The following example allows SSH access from host `192.168.0.2` to any IP address on this host:

```
sudo ufw allow proto tcp from 192.168.0.2 to any port 22
```

Replace `192.168.0.2` with `192.168.0.0/24` to allow SSH access from the entire subnet.

### The `--dry-run` option

Adding the `--dry-run` option to a `ufw` command will output the resulting rules, but not apply them. For example, the following is what would be applied if opening the HTTP port:

```
sudo ufw --dry-run allow http

    *filter
    :ufw-user-input - [0:0]
    :ufw-user-output - [0:0]
    :ufw-user-forward - [0:0]
    :ufw-user-limit - [0:0]
    :ufw-user-limit-accept - [0:0]
    ### RULES ###

    ### tuple ### allow tcp 80 0.0.0.0/0 any 0.0.0.0/0
    -A ufw-user-input -p tcp --dport 80 -j ACCEPT

    ### END RULES ###
    -A ufw-user-input -j RETURN
    -A ufw-user-output -j RETURN
    -A ufw-user-forward -j RETURN
    -A ufw-user-limit -m limit --limit 3/minute -j LOG --log-prefix "[UFW LIMIT]:
"
    -A ufw-user-limit -j REJECT
    -A ufw-user-limit-accept -j ACCEPT
    COMMIT
    Rules updated
```

### Check the status

To see the firewall status, enter:

```
sudo ufw status
```

And for more verbose status information use:

```
sudo ufw status verbose
```

> **Note**
>
> If the port you want to open or close is defined in `/etc/services`, you can use the port name instead of the number. In the above examples, replace `22` with `ssh`.

This is a quick introduction to using `ufw`. Please refer to the `ufw(8)` manual page for more information.

### ufw application integration

Applications that open ports can include a `ufw` profile, which details the ports needed for the application to function properly. The profiles are kept in `/etc/ufw/applications.d`, and can be edited if the default ports have been changed.

To view which applications have installed a profile, run the following in a terminal:

```
sudo ufw app list
```

Similar to allowing traffic to a port, using an application profile is accomplished by entering:

```
sudo ufw allow Samba
```

An extended syntax is available as well:

```
ufw allow from 192.168.0.0/24 to any app Samba
```

Replace `Samba` and `192.168.0.0/24` with the application profile you are using and the IP range for your network.

> **Note**
>
> There is no need to specify the **protocol** for the application, because that information is detailed in the profile. Also, note that the `app` name replaces the `port` number.

To view details about which ports and protocols, and so on, are defined for an application, enter:

```
sudo ufw app info Samba
```

Not all applications that require opening a network port come with `ufw` profiles, but if you have profiled an application and want the file to be included with the package, please file a bug against the package in Launchpad.

```
ubuntu-bug nameofpackage
```

### IP masquerading

The purpose of IP masquerading is to allow machines with private, non-routable IP addresses on your network to access the Internet through the machine doing the masquerading. Traffic from your private network destined for the Internet must be manipulated for replies to be routable back to the machine that made the request.

To do this, the kernel must modify the **source** IP address of each packet so that replies will be routed back to it, rather than to the private IP address that made the request, which is impossible over the Internet. Linux uses **Connection Tracking** (`conntrack(8)`) to keep track of which connections belong to which machines and reroute each return packet accordingly. Traffic leaving your private network is thus “masqueraded” as having originated from your Ubuntu gateway machine. This process is referred to in Microsoft documentation as “Internet Connection Sharing”.

### IP masquerading with ufw

IP masquerading can be achieved using custom `ufw` rules. This is possible because the current back-end for `ufw` is `iptables-restore` with the rules files located in `/etc/ufw/*.rules`. These files are a great place to add legacy `iptables` rules used without `ufw`, and rules that are more network gateway or bridge related.

The rules are split into two different files; rules that should be executed before `ufw` command line rules, and rules that are executed after `ufw` command line rules.

### Enable packet forwarding

First, packet forwarding needs to be enabled in `ufw`. Two configuration files will need to be adjusted, so first, in `/etc/default/ufw` change the `DEFAULT_FORWARD_POLICY` to "`ACCEPT`":

```
DEFAULT_FORWARD_POLICY="ACCEPT"
```

Then, edit `/etc/ufw/sysctl.conf` and uncomment:

```
net/ipv4/ip_forward=1
```

Similarly, for IPv6 forwarding, uncomment:

```
net/ipv6/conf/default/forwarding=1
```

### Add the configuration

Now add rules to the `/etc/ufw/before.rules` file. The default rules only configure the **filter** table, and to enable masquerading the `nat` table will need to be configured. Add the following to the top of the file, just after the header comments:

```
# nat Table rules
*nat
:POSTROUTING ACCEPT [0:0]

# Forward traffic from eth1 through eth0.
-A POSTROUTING -s 192.168.0.0/24 -o eth0 -j MASQUERADE

# don't delete the 'COMMIT' line or these nat table rules won't be processed
COMMIT
```

The comments are not strictly necessary, but it is considered good practice to document your configuration. Also, when modifying any of the `rules` files in `/etc/ufw`, make sure these lines are the last line for each table modified:

```
# don't delete the 'COMMIT' line or these rules won't be processed
COMMIT
```

For each **Table**, a corresponding `COMMIT` statement is required. In these examples only the `nat` and `filter` tables are shown, but you can also add rules for the `raw` and `mangle` tables.

**Note**

In the above example, replace `eth0`, `eth1`, and `192.168.0.0/24` with the appropriate interfaces and IP range for your network.

### Restart ufw

Finally, disable and re-enable `ufw` to apply the changes:

```
sudo ufw disable && sudo ufw enable
```

IP masquerading should now be enabled. You can also add any additional `FORWARD` rules to the `/etc/ufw/before.rules`. It is recommended that these additional rules be added to the `ufw-before-forward` chain.

### IP masquerading with iptables

`iptables` can also be used to enable masquerading.

Similarly to `ufw`, the first step is to enable IPv4 packet forwarding by editing `/etc/sysctl.conf` and uncomment the following line:

```
net.ipv4.ip_forward=1
```

If you wish to enable IPv6 forwarding also uncomment:

```
net.ipv6.conf.default.forwarding=1
```

Next, run the `sysctl` command to enable the new settings in the configuration file:

```
sudo sysctl -p
```

IP masquerading can now be accomplished with a single `iptables` rule, which may differ slightly based on your network configuration:

```
sudo iptables -t nat -A POSTROUTING -s 192.168.0.0/16 -o ppp0 -j MASQUERADE
```

The above command assumes that your private address space is `192.168.0.0/16` and that your Internet-facing device is `ppp0`. The syntax is broken down as follows:

- `-t nat` – the rule is to go into the NAT table
- `-A POSTROUTING` – the rule is to be appended (`-A`) to the `POSTROUTING` chain
- `-s 192.168.0.0/16` – the rule applies to traffic originating from the specified address space
- `-o ppp0` – the rule applies to traffic scheduled to be routed through the specified network device
- `-j MASQUERADE` – traffic matching this rule is to "jump" (`-j`) to the `MASQUERADE` target to be manipulated as described above

Also, each chain in the filter table (the default table, and where most – or all – packet filtering occurs) has a default **policy** of `ACCEPT`, but if you are creating a firewall in addition to a gate-

way device, you may have set the policies to DROP or REJECT, in which case your masqueraded traffic needs to be allowed through the FORWARD chain for the above rule to work:

```
sudo iptables -A FORWARD -s 192.168.0.0/16 -o ppp0 -j ACCEPT
sudo iptables -A FORWARD -d 192.168.0.0/16 -m state \
--state ESTABLISHED,RELATED -i ppp0 -j ACCEPT
```

The above commands will allow all connections from your local network to the Internet and all traffic related to those connections to return to the machine that initiated them.

If you want masquerading to be enabled on reboot, which you probably do, edit /etc/rc.local and add any commands used above. For example add the first command with no filtering:

```
iptables -t nat -A POSTROUTING -s 192.168.0.0/16 -o ppp0 -j MASQUERADE
```

### Logs

Firewall logs are essential for recognising attacks, troubleshooting your firewall rules, and noticing unusual activity on your network. You must include logging rules in your firewall for them to be generated, though, and logging rules must come before any applicable terminating rule (a rule with a target that decides the fate of the packet, such as ACCEPT, DROP, or REJECT).

If you are using ufw, you can turn on logging by entering the following in a terminal:

```
sudo ufw logging on
```

To turn logging off in ufw, replace on with off in the above command:

If you are using iptables instead of ufw, run:

```
sudo iptables -A INPUT -m state --state NEW -p tcp --dport 80 \
 -j LOG --log-prefix "NEW_HTTP_CONN: "
```

A request on port 80 from the local machine, then, would generate a log in *dmesg* that looks like this (single line split into 3 to fit this document):

```
[4304885.870000] NEW_HTTP_CONN: IN=lo OUT=
MAC=00:00:00:00:00:00:00:00:00:00:00:00:08:00
SRC=127.0.0.1 DST=127.0.0.1 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=58288 DF PROTO=TCP
SPT=53981 DPT=80 WINDOW=32767 RES=0x00 SYN URGP=0
```

The above log will also appear in /var/log/messages, /var/log/syslog, and /var/log/kern.log. This behavior can be modified by editing /etc/syslog.conf appropriately or by installing and configuring ulogd and using the ULOG target instead of LOG. The ulogd daemon is a userspace server that listens for logging instructions from the kernel – specifically for firewalls – and can log to any file you like, or even to a PostgreSQL or MySQL database. Making sense of your firewall logs can be simplified by using a log analysing tool such as logwatch, fwanalog, fwlogwatch, or lire.

### Other tools

There are many tools available to help you construct a complete firewall without intimate knowledge of `iptables`. A command-line tool with plain-text configuration files, for example, is Shorewall; a powerful solution to help you configure an advanced firewall for any network.

### Further reading

- The Ubuntu Firewall wiki page contains information on the development of ufw
- Also, the `ufw(8)` manual page contains some very useful information: `man ufw`
- See the packet-filtering-HOWTO for more information on using `iptables`
- The nat-HOWTO contains further details on masquerading
- The IPTables HowTo in the Ubuntu wiki is a great resource

### AppArmor

AppArmor is an easy-to-use Linux Security Module implementation that restricts applications' capabilities and permissions with **profiles** that are set per-program. It provides mandatory access control (MAC) to supplement the more traditional UNIX model of discretionary access control (*DAC*).

In Ubuntu, AppArmor is installed and loaded by default – you can check this by running `aa-status`.

It uses **profiles** of an application to determine what files and permissions the application requires. Some packages will install their own profiles, and additional profiles can be found in the `apparmor-profiles` package.

### Install AppArmor Profiles

To install the `apparmor-profiles` package from a terminal prompt:

```
sudo apt install apparmor-profiles
```

AppArmor profiles have two modes of operation:

- **Complaining/Learning**: profile violations are permitted and logged. This is useful for testing and developing new profiles.
- **Enforced/Confined**: enforces profile policy in addition to logging the violation.

### Using AppArmor

The optional `apparmor-utils` package contains command-line utilities you can use to change the AppArmor operation mode, find the status of a profile, create new profiles, etc.

AppArmor profiles are located in the `/etc/apparmor.d` directory. It also stores **abstractions** that can simplify profile authoring, such as `abstractions/base` that allows many shared libraries, writing logs to the journal, many pseudo-devices, receiving signals from unconfined processes, and many more things.

### Common commands

- `apparmor_status` is used to view the current status of AppArmor profiles:

```
sudo apparmor_status
```

- `aa-complain` places a profile into `complain` mode:

```
sudo aa-complain /path/to/bin
```

- `aa-enforce` places a profile into `enforce` mode:

```
sudo aa-enforce /path/to/bin
```

- `apparmor_parser` is used to load a profile into the kernel. It can also be used to reload a currently-loaded profile using the `-r` option after modifying it to have the changes take effect. To reload a profile:

```
sudo apparmor_parser -r /etc/apparmor.d/profile.name
```

- `systemctl` can be used to reload all profiles:

```
sudo systemctl reload apparmor.service
```

### Disabling or re-enabling a profile

The `/etc/apparmor.d/disable` directory can be used along with the `apparmor_parser -R` option to disable a profile:

```
sudo ln -s /etc/apparmor.d/profile.name /etc/apparmor.d/disable/
sudo apparmor_parser -R /etc/apparmor.d/profile.name
```

To re-enable a disabled profile, remove the symbolic link to the profile in `/etc/apparmor.d/disable/`, then load the profile using the `-a` option:

```
sudo rm /etc/apparmor.d/disable/profile.name
cat /etc/apparmor.d/profile.name | sudo apparmor_parser -a
```

> **Note**: Replace `profile.name` with the name of the profile you want to manipulate. Also, replace `/path/to/bin/` with the actual executable file path. For example, for the `ping` command use `/bin/ping`.

### Profiles

AppArmor profiles are simple text files located in `/etc/apparmor.d/`. The files are named after the full path to the executable they profile, replacing the "/" with ".".

For example `/etc/apparmor.d/bin.ping` is the AppArmor profile for the `/bin/ping` command.

There are two main type of rules used in profiles:

- **Path entries**, detailing which files an application can access in the file system.
- **Capability entries**, which determine what privileges a confined process is allowed to use.

As an example, take a look at `/etc/apparmor.d/bin.ping`:

```
#include <tunables/global>
/bin/ping flags=(complain) {
  #include <abstractions/base>
  #include <abstractions/consoles>
  #include <abstractions/nameservice>

  capability net_raw,
  capability setuid,
  network inet raw,

  /bin/ping mixr,
  /etc/modules.conf r,
}
```

Which can be broken down as follows:

- `#include <tunables/global>`: include statements from other files. This allows statements pertaining to multiple applications to be placed in a common file.
- `/bin/ping flags=(complain)`: path to the profiled program, also setting the mode to `complain`.
- `capability net_raw,`: allows the application access to the `CAP_NET_RAW Posix.1e` capability.
- `/bin/ping mixr,`: allows the application read and execute access to the file.

> **Note**
>
> After editing a profile file the profile must be reloaded.

### Create a Profile

- **Design a test plan**: Try to think about how the application should be exercised. The test plan should be divided into small test cases. Each test case should have a small description and list the steps to follow.

  Some standard test cases are:

  - Starting the program
  - Stopping the program
  - Reloading the program
  - Testing all the commands supported by the init script

- **Generate the new profile**: Use `aa-genprof` to generate a new profile. From a terminal:

  ```
  sudo aa-genprof executable
  ```

  For example:

```
sudo aa-genprof slapd
```

- To get your new profile included in the `apparmor-profiles` package, file a bug in Launchpad against the AppArmor package:
  - Include your test plan and test cases
  - Attach your new profile to the bug

### Updating profiles

When the program is misbehaving, audit messages are sent to the log files. The program `aa-logprof` can be used to scan log files for AppArmor audit messages, review them and update the profiles. From a terminal:

```
sudo aa-logprof
```

### Further pre-existing profiles

The packages `apport-profiles` and `apparmor-profiles-extra` ship some experimental profiles for AppArmor security policies. Do not expect these profiles to work out-of-the-box, but they can give you a head start when trying to create a new profile by starting off with a base that already exists.

These profiles are not considered mature enough to be shipped in `enforce` mode by default. Therefore, they are shipped in `complain` mode so that users can test them, choose which are desired, and help improve them upstream if needed.

Some even more experimental profiles carried by the package are placed in `/usr/share/doc/apparmor-profiles/extras/`

### Checking and debugging denies

You will see in *dmesg* (and any log that collects kernel messages) if you have hit a **deny**. It is worth knowing that this will cover any access that was denied `because it was not allowed`, but `explicit denies` will put no message in your logs at all.

Examples might look like:

```
[1521056.552037] audit: type=1400 audit(1571868402.378:24425): apparmor="DENIED"
operation="open" profile="/usr/sbin/cups-browsed" name="/var/lib/libvirt/dnsmasq/"
pid=1128 comm="cups-browsed" requested_mask="r" denied_mask="r" fsuid=0 ouid=0
[1482106.651527] audit: type=1400 audit(1571829452.330:24323): apparmor="DENIED"
operation="sendmsg" profile="snap.lxd.lxc" pid=24115 comm="lxc" laddr=10.7.0.69
lport=48796 faddr=10.7.0.231 fport=445 family="inet" sock_type="stream" protocol=6
requested_mask="send" denied_mask="send"
```

That follows a generic structure starting with a timestamp, an audit tag and the category `apparmor="DENIED"`. From the following fields you can derive what was going on and why it was failing.

In the examples above that would be:

First example:

- `operation`: `open` (program tried to open a file)
- `profile`: `/usr/sbin/cups-browsed` (you'll find `/etc/apparmor.d/usr.bin.cups-browsed`)
- `name`: `/var/lib/libvirt/dnsmasq` (what it wanted to access)
- `pid/comm`: the program that triggered the access
- `requested_mask/denied_mask/fsuid/ouid`: parameters of that open call

Second example:

- `operation`: `sendmsg` (program tried send via network)
- `profile`: `snap.lxd.lxc` (snaps are special, you'll find `/var/lib/snapd/apparmor/profiles/snap.lxd.lxc`)
- `pid/comm`: the program that triggered the access
- `laddr/lport/faddr/fport/family/sock_type/protocol`: parameters of the `sendmsg` call

That way you know in which profile and at what action you have to start if you consider either debugging or adapting the profiles.

### Profile customisation

Profiles are meant to provide security and so can't be too permissive. But often, a very special setup would work with a profile if it wold *just allow this one extra access*. To handle that situation, there are three options:

- Modify the profile itself:
  - Always works, but has the drawback that profiles are in `/etc` and considered `conffiles`. So after modification on a related package update you might get a `conffile` prompt. Worst case; depending on configuration, automatic updates might even override it and your custom rule is gone.
- Use tunables:
  - These provide variables that can be used in templates, for example if you want a custom `dir` considered as it would be a home directory. You could modify `/etc/apparmor.d/tunables/home`, which defines the base path rules used for home directories.
  - By design, these variables will only influence profiles that use them.
- Modify a local override:
  - To mitigate the drawbacks of above approaches, **local includes** were introduced, adding the ability to write arbitrary rules that not run into issues during upgrades that modify the packaged rule.
  - The files can be found in `/etc/apparmor.d/local/` and exist for the packages that are known to sometimes need slight tweaks for special setups.

### Disabling or Re-enabling AppArmor

Starting with Ubuntu 24.04 and later, the AppArmor services are baked into the Ubuntu Kernel. In earlier versions of Ubuntu, you could disable AppArmor by not loading the service.

However, it now requires setting a module parameter on the kernel command line to fully disable or re-enable AppArmor.

### Disable AppArmor

> WARNING! Disabling AppArmor reduces the security of your system! You should only disable apparmor if you understand the security implications of disabling the service!

To disable AppArmor, you must do the following:

- Edit `/etc/default/grub` and add `apparmor=0` to `GRUB_CMDLINE_LINUX` in `/etc/default/grub`
- Run `sudo update-grub` to refresh the boot configuration
- Reboot the system.

Once rebooted, you can check the status of AppArmor with the following commands, and should see similar output to this:

```
$ sudo aa-status
apparmor module is loaded.
apparmor filesystem is not mounted.
```

```
$ systemctl status apparmor
● apparmor.service - Load AppArmor profiles
     Loaded: loaded (/usr/lib/systemd/system/apparmor.service; enabled; preset:
enabled)
     Active: inactive (dead)
  Condition: start condition unmet at Tue 2025-01-14 08:45:04 UTC; 1min 20s ago
             └─ ConditionSecurity=apparmor was not met
       Docs: man:apparmor(7)
             https://gitlab.com/apparmor/apparmor/wikis/home/

Jan 14 08:45:04 n systemd[1]: apparmor.service - Load AppArmor profiles was
skipped because of an unmet condition check (ConditionSecurity=apparmor).
```

### Re-enable AppArmor

- Remove the `apparmor=0` item from `GRUB_CMDLINE_LINUX` in `/etc/default/grub`.
- Run `sudo update-grub` to update your system boot configuration.
- Reboot your system.

You can check if AppArmor is then re-enabled with the following command, and you should seee output similar to this:

```
$ sudo aa-status
apparmor module is loaded.
119 profiles are loaded.
24 profiles are in enforce mode.
   /usr/bin/man
...
```

### Further reading

- See the AppArmor Administration Guide for advanced configuration options.
- For details using AppArmor with other Ubuntu releases see the AppArmor Community Wiki page.
- The OpenSUSE AppArmor page is another introduction to AppArmor.
- (https://wiki.debian.org/AppArmor) is another introduction and basic how-to for AppArmor.
- A great place to get involved with the Ubuntu Server community and to ask for AppArmor assistance is the `\#ubuntu-server` IRC channel on Libera. The `\#ubuntu-security` IRC channel may also be of use.

### Console security

It is difficult to defend against the damage that can be caused by someone with physical access to your environment, for example, due to theft of hard drives, power or service disruption, and so on.

Console security should be addressed as one component of your overall physical security strategy. A locked "screen door" may deter a casual criminal, or at the very least slow down a determined one, so it is still advisable to perform basic precautions with regard to console security.

### Disable Ctrl+Alt+Delete

Anyone with physical access to the keyboard can use the Ctrl + Alt + Delete key combination to reboot the server without having to log on. While someone could also simply unplug the power source, you should still prevent the use of this key combination on a production server. This forces an attacker to take more drastic measures to reboot the server, and will prevent accidental reboots at the same time.

To disable the reboot action taken by pressing the Ctrl + Alt + Delete key combination, run the following two commands:

```
sudo systemctl mask ctrl-alt-del.target
sudo systemctl daemon-reload
```

### Authentication

These tools are particularly useful for more advanced or complex setups.

- *Kerberos* is a network authentication protocol providing identity verification for distributed systems
- *Network user authentication with SSSD* handles authentication, user/group information and authorisation from disparate network sources
- *Smart card authentication* provides a physical authentication method

## Kerberos

This section assumes you have some familiarity with the terms and concepts used in Kerberos.

You will need a properly configured DNS server set up before you begin. See the *Domain Name Service (DNS)* page for instructions on how to set this up.

We recommend following this set of how-to guides in the order presented for best results.

### How to install a Kerberos server

For this discussion, we will create an MIT Kerberos domain with the following features (edit them to fit your needs):

- **Realm**: `EXAMPLE.COM`
- **Primary KDC**: `kdc01.example.com`
- **Secondary KDC**: `kdc02.example.com`
- **User principal**: `ubuntu`
- **Admin principal**: `ubuntu/admin`

### Prerequisites

Before installing the Kerberos server, a properly configured *DNS* server is needed for your domain. Since the Kerberos realm (by convention) matches the domain name, this section uses the `EXAMPLE.COM` domain configured in the primary server section of the *DNS documentation*.

Also, Kerberos is a time sensitive protocol. If the local system time between a client machine and the server differs by more than five minutes (by default), the workstation will not be able to authenticate. To correct the problem all hosts should have their time synchronized using the same Network Time Protocol (NTP) server. Check out the *NTP chapter* for more details.

### Install the Kerberos packages

The first step in creating a Kerberos realm is to install the `krb5-kdc` and `krb5-admin-server` packages. From a terminal enter:

```
sudo apt install krb5-kdc krb5-admin-server
```

You will be asked at the end of the install to supply the *hostname* for the Kerberos and Admin servers for the realm, which may or may not be the same server. Since we are going to create the realm, and thus these servers, type in the full hostname of this server.

> **Note**
>
> By default the realm name will be domain name of the Key Distribution Center (KDC) server.

Next, create the new realm with the `kdb5_newrealm` utility:

```
sudo krb5_newrealm
```

It will ask you for a database master password, which is used to encrypt the local database. Chose a secure password: its strength is not verified for you.

### Configure the Kerberos server

The questions asked during installation are used to configure the `/etc/krb5.conf` and `/etc/krb5kdc/kdc.conf` files. The former is used by the Kerberos 5 libraries, and the latter configures the KDC. If you need to adjust the KDC settings, edit the file and restart the `krb5-kdc` daemon. If you need to reconfigure Kerberos from scratch, perhaps to change the realm name, you can do so by typing:

```
sudo dpkg-reconfigure krb5-kdc
```

> **Note**
>
> The manpage for `krb5.conf` is in the `krb5-doc` package.

Let's create our first principal. Since there is no principal create yet, we need to use `kadmin.local`, which uses a local UNIX socket to talk to the KDC, and requires root privileges:

```
$ sudo kadmin.local
Authenticating as principal root/admin@EXAMPLE.COM with password.
kadmin.local: addprinc ubuntu
WARNING: no policy specified for ubuntu@EXAMPLE.COM; defaulting to no policy
Enter password for principal "ubuntu@EXAMPLE.COM":
Re-enter password for principal "ubuntu@EXAMPLE.COM":
Principal "ubuntu@EXAMPLE.COM" created.
kadmin.local: quit
```

To be able to use `kadmin` remotely, we should create an **admin principal**. Convention suggests it should be an **admin instance**, as that also makes creating a generic Access Control List (*ACL*) easier. Let's create an admin instance for the `ubuntu` principal:

```
$ sudo kadmin.local
Authenticating as principal root/admin@EXAMPLE.COM with password.
kadmin.local: addprinc ubuntu/admin
WARNING: no policy specified for ubuntu/admin@EXAMPLE.COM; defaulting to no policy
Enter password for principal "ubuntu/admin@EXAMPLE.COM":
Re-enter password for principal "ubuntu/admin@EXAMPLE.COM":
Principal "ubuntu/admin@EXAMPLE.COM" created.
kadmin.local: quit
```

Next, the new admin principal needs to have the appropriate ACL permissions. The permissions are configured in the `/etc/krb5kdc/kadm5.acl` file:

```
ubuntu/admin@EXAMPLE.COM        *
```

You can also use a more generic form for this ACL:

```
*/admin@EXAMPLE.COM        *
```

The above will grant all privileges to any admin instance of a principal. See the `kadm5.acl(5)` manpage for details.

Now restart the `krb5-admin-server` for the new ACL to take effect:

```
sudo systemctl restart krb5-admin-server.service
```

The new user principal can be tested using the `kinit` utility:

```
$ kinit ubuntu/admin
Password for ubuntu/admin@EXAMPLE.COM:
```

After entering the password, use the `klist` utility to view information about the Ticket Granting Ticket (TGT):

```
$ klist
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: ubuntu/admin@EXAMPLE.COM

Valid starting     Expires            Service principal
04/03/20 19:16:57  04/04/20 05:16:57  krbtgt/EXAMPLE.COM@EXAMPLE.COM
     renew until 04/04/20 19:16:55
```

Where the cache filename `krb5cc_1000` is composed of the prefix `krb5cc_` and the user id (UID), which in this case is `1000`.

`kinit` will inspect `/etc/krb5.conf` to find out which KDC to contact, and the corresponding address. The KDC can also be found via DNS lookups for special TXT and SRV records. You can add these records to your `example.com` DNS zone:

```
_kerberos._udp.EXAMPLE.COM.      IN SRV 1  0 88  kdc01.example.com.
_kerberos._tcp.EXAMPLE.COM.      IN SRV 1  0 88  kdc01.example.com.
_kerberos._udp.EXAMPLE.COM.      IN SRV 10 0 88  kdc02.example.com.
_kerberos._tcp.EXAMPLE.COM.      IN SRV 10 0 88  kdc02.example.com.
_kerberos-adm._tcp.EXAMPLE.COM.  IN SRV 1  0 749 kdc01.example.com.
_kpasswd._udp.EXAMPLE.COM.       IN SRV 1  0 464 kdc01.example.com.
```

See the *DNS chapter* for detailed instructions on setting up DNS.

A very quick and useful way to troubleshoot what `kinit` is doing is to set the environment variable `KRB5_TRACE` to a file, or `stderr`, and it will show extra information. The output is quite verbose:

```
$ KRB5_TRACE=/dev/stderr kinit ubuntu/admin
[2898] 1585941845.278578: Getting initial credentials for ubuntu/admin@EXAMPLE.COM
[2898] 1585941845.278580: Sending unauthenticated request
[2898] 1585941845.278581: Sending request (189 bytes) to EXAMPLE.COM
[2898] 1585941845.278582: Resolving hostname kdc01.example.com
(...)
```

Your new Kerberos realm is now ready to authenticate clients.

### How to configure Kerberos service principals

The specific steps to enable Kerberos for a service can vary, but in general both of the following are needed:

- A principal for the service – usually `service/host@REALM`
- A keytab accessible to the service wherever it's running – usually in `/etc/krb5.keytab`

For example, let's create a principal for an LDAP service running on the `ldap-server.example.com` host:

```
ubuntu@ldap-server:~$ sudo kadmin -p ubuntu/admin
Authenticating as principal ubuntu/admin with password.
Password for ubuntu/admin@EXAMPLE.COM:
kadmin:  addprinc -randkey ldap/ldap-server.example.com
No policy specified for ldap/ldap-server.example.com@EXAMPLE.COM; defaulting to no
policy
Principal "ldap/ldap-server.example.com@EXAMPLE.COM" created.
```

Let's dig a bit into what is happening here:

- The `kadmin` command is being run on the `ldap-server` machine, not on the Key Distribution Center (KDC). We are using `kadmin` remotely.
- It's being run with `sudo`. The reason for this will become clear later.
- We are logged in on the server as `ubuntu`, but specifying an `ubuntu/admin` principal. Remember the `ubuntu` principal has no special privileges.
- The name of the principal we are creating follows the pattern `service/hostname`.
- In order to select a random secret, we pass the `-randkey` parameter. Otherwise we would be asked to type in a password.

With the principal created, we need to extract the key from the KDC and store it in the `ldap-server` host, so that the `ldap` service can use it to authenticate itself with the KDC. Still in the same `kadmin` session:

```
kadmin:  ktadd ldap/ldap-server.example.com
Entry for principal ldap/ldap-server.example.com with kvno 2, encryption type
aes256-cts-hmac-sha1-96 added to keytab FILE:/etc/krb5.keytab.
Entry for principal ldap/ldap-server.example.com with kvno 2, encryption type
aes128-cts-hmac-sha1-96 added to keytab FILE:/etc/krb5.keytab.
```

This is why we needed to run `kadmin` with `sudo`: so that it can write to `/etc/krb5.keytab`. This is the system keytab file, which is the default file for all keys that might be needed for services on this host, and we can list them with `klist`. Back in the shell:

```
$ sudo klist -k
Keytab name: FILE:/etc/krb5.keytab
KVNO Principal
---- --------------------------------------------------------------------------
   2 ldap/ldap-server.example.com@EXAMPLE.COM
   2 ldap/ldap-server.example.com@EXAMPLE.COM
```

If you don't have the `kadmin` utility on the target host, one alternative is to extract the keys on a different host and into a different file, and then transfer this file *securely* to the target server. For example:

```
kadmin:  ktadd -k /home/ubuntu/ldap.keytab ldap/ldap-server.example.com
Entry for principal ldap/ldap-server.example.com with kvno 3, encryption type
aes256-cts-hmac-sha1-96 added to keytab WRFILE:/home/ubuntu/ldap.keytab.
Entry for principal ldap/ldap-server.example.com with kvno 3, encryption type
aes128-cts-hmac-sha1-96 added to keytab WRFILE:/home/ubuntu/ldap.keytab.
```

**Note**

Notice how the `kvno` changed from `2` to `3` in the example above, when using `ktadd` a second time? This is the key version, and it invalidated the previously extracted key with `kvno 2`. Every time a key is extracted with `ktadd`, its version is bumped and that invalidates the previous ones!

In this case, as long as the target location is writable, you don't even have to run `kadmin` with `sudo`.

Then use `scp` to transfer it to the target host:

```
$ scp /home/ubuntu/ldap.keytab ldap-server.example.com:
```

And over there copy it to `/etc/krb5.keytab`, making sure it's mode `0600` and owned by `root:root`.

### Kerberos encryption types

Encryption is at the heart of Kerberos, and it supports multiple cryptographic algorithms. The default choices are good enough for most deployments, but specific situations might need to tweak these settings.

This document will explain the basic configuration parameters of Kerberos that control the selection of encryption algorithms used in a Kerberos deployment.

### Server-side configuration

There are two main server-side configuration parameters that control the encryption types used on the server for its database and its collection or principals. Both exist in `/etc/krb5kdc/kdc.conf` inside the `[realms]` section and are as follows:

- `master_key_type` Specifies the key type of the master key. This is used to encrypt the database, and the default is `aes256-cts-hmac-sha1-96`.
- `supported_enctypes` Specifies the default key/salt combinations of principals for this realm. The default is `aes256-cts-hmac-sha1-96:normal aes128-cts-hmac-sha1-96:normal`, and the encryption types should be listed in order of preference.

Possible values for the encryption algorithms are listed in the MIT documentation on encryption types, and the salt types can be seen in the MIT keysalt lists.

Here is an example showing the default values (other settings removed for brevity):

```
[realms]
    EXAMPLE.INTERNAL = {
        (...)
        master_key_type = aes256-cts
        supported_enctypes = aes256-cts-hmac-sha1-96:normal aes128-cts-hmac-sha1-
96:normal
        (...)
}
```

The master key is created once per realm, when the realm is bootstrapped. That is usually done with the `krb5_newrealm` tool (see *how to install a Kerberos server* for details). You can check the master key type with either of these commands on the KDC server:

```
$ sudo kadmin.local
kadmin.local:  getprinc K/M
Principal: K/M@EXAMPLE.INTERNAL
(...)
Number of keys: 1
Key: vno 1, aes256-cts-hmac-sha1-96
(...)

$ sudo klist -ke /etc/krb5kdc/stash
Keytab name: FILE:/etc/krb5kdc/stash
KVNO Principal
---- --------------------------------------------------------------------------
   1 K/M@EXAMPLE.INTERNAL (aes256-cts-hmac-sha1-96)
```

When a new Kerberos principal is created through the `kadmind` service (via the `kadmin` or `kadmin.local` utilities), the types of encryption keys it will get are controlled via the `supported_enctypes` configuration parameter.

For example, let's create an `ubuntu` principal, and check the keys that were created for it (output abbreviated):

```
$ sudo kadmin.local
Authenticating as principal root/admin@EXAMPLE.INTERNAL with password.
kadmin.local:  addprinc ubuntu
No policy specified for ubuntu@EXAMPLE.INTERNAL; defaulting to no policy
Enter password for principal "ubuntu@EXAMPLE.INTERNAL":
Re-enter password for principal "ubuntu@EXAMPLE.INTERNAL":
Principal "ubuntu@EXAMPLE.INTERNAL" created.

kadmin.local:  getprinc ubuntu
Principal: ubuntu@EXAMPLE.INTERNAL
(...)
Number of keys: 2
Key: vno 1, aes256-cts-hmac-sha1-96
Key: vno 1, aes128-cts-hmac-sha1-96
(...)
```

Two keys were created for the `ubuntu` principal, following the default setting of `supported_enctypes` in `kdc.conf` for this realm.

**Note**

The server config `supported_enctypes` has the *default* list of key types that are created for a principal. This list applies to the moment when that principal is **created** by `kadmind`. Changing that setting after the fact won't affect the keys that the principal in question has after that event. In particular, principals can be created with specific key types regardless of the `supported_enctypes` setting. See the `-e` parameter for the kadmin add_principal command.

If we had `supported_enctypes` set to `aes256-sha2:normal aes128-sha2:normal camellia256-cts:normal` in `kdc.conf`, then the `ubuntu` principal would get three key types:

```
kadmin.local:  getprinc ubuntu
Principal: ubuntu@EXAMPLE.INTERNAL
(...)
Number of keys: 3
Key: vno 1, aes256-cts-hmac-sha384-192
Key: vno 1, aes128-cts-hmac-sha256-128
Key: vno 1, camellia256-cts-cmac
```

**Note**

Bootstrapping a new Kerberos realm via the `krb5_newrealm` command also creates some system principals required by Kerberos, such as `kadmin/admin`, `kadmin/changepw` and others. They will all also get the same number of keys each: one per encryption type in `supported_enctypes`.

### Client-side configuration

When we say "client-side", we really mean "applications linked with the Kerberos libraries". These live on the server too, so keep that in mind.

The encryption types supported by the Kerberos libraries are defined in the `/etc/krb5.conf` file, inside the `[libdefaults]` section, via the `permitted_enctypes` parameter.

Example:

```
[libdefaults]
(...)
permitted_enctypes = aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96
```

This parameter contains a space-separated list of encryption type names, in order of preference. Default value: `aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96 aes256-cts-hmac-sha384-192 aes128-cts-hmac-sha256-128 des3-cbc-sha1 arcfour-hmac-md5 camellia256-cts-cmac camellia128-cts-cmac`.

Possible values for the encryption algorithms are listed in the MIT documentation (same ones as for the KDC).

**Note**

There are more encryption-related parameters in `krb5.conf`, but most take their defaults from `permitted_enctypes`. See the MIT libdefaults documentation for more information.

### Putting it all together

When a client performs Kerberos authentication and requests a ticket from the KDC, the encryption type used in that ticket is decided by picking the common set of:

- The encryption types supported by the server for that principal
- The encryption types supported by the client

If there is no common algorithm between what the client accepts, and what the server has to offer for that specific principal, then `kinit` will fail.

For example, if the principal on the server has:

```
kadmin.local:  getprinc ubuntu
Principal: ubuntu@EXAMPLE.INTERNAL
(...)
Number of keys: 2
Key: vno 1, aes256-cts-hmac-sha384-192
Key: vno 1, aes128-cts-hmac-sha256-128
```

And the client's `krb5.conf` has:

```
permitted_enctypes = aes256-sha1 aes128-sha1
```

Then `kinit` will fail, because the client only supports `sha1` variants, and the server only has `sha2` to offer for that particular principal the client is requesting:

```
$ kinit ubuntu

kinit: Generic error (see e-text) while getting initial credentials
```

The server log (`journalctl -u krb5-admin-server.service`) will have more details about the error:

```
Apr 19 19:31:49 j-kdc krb5kdc[8597]: AS_REQ (2 etypes {aes256-cts-hmac-sha1-
96(18), aes128-cts-hmac-sha1-96(17)}) fd42:78f4:b1c4:3964:216:3eff:feda:118c: GET_
LOCAL_TGT: ubuntu@EXAMPLE.INTERNAL for krbtgt/EXAMPLE.INTERNAL@EXAMPLE.INTERNAL,
No matching key in entry having a permitted enctype
```

This log says that there was an `AS-REQ` request which accepted two encryption types, but there was no matching key type on the server database for that principal.

### Changing encryption types

Changing encryption types of an existing Kerberos realm is no small task. Just changing the configuration settings won't recreate existing keys, nor add new ones. The modifications have to be done in incremental steps.

MIT Kerberos has a guide on updating encryption types that covers many scenarios, including deployments with multiple replicating servers:

### References

- Encryption types in MIT Kerberos
- `krb5.conf` encryption related configurations options
- Migrating away from older encryption types
- `kdc.conf(5)` manpage
- `krb5.conf(5)` manpage
- Kerberos V5 concepts

### How to set up secondary KDC

Once you have one Key Distribution Center (KDC) on your network, it is good practice to have a secondary KDC in case the primary becomes unavailable.

Also, if you have Kerberos clients that are on different networks (possibly separated by routers using NAT), it is wise to place a secondary KDC in each of those networks.

> **Note**
>
> The native replication mechanism explained here relies on a cron job; it essentially dumps the DB on the primary and loads it back up on the secondary. You may want to take a look at using the `kldap` backend, which can use the OpenLDAP replication mechanism. This is explained further below.

### Install the required packages

First, install the packages, and when asked for the Kerberos and Admin server names enter the name of the Primary KDC:

```
sudo apt install krb5-kdc krb5-admin-server
```

Once you have installed the packages, create the host principals for both KDCs. From a terminal prompt, enter:

```
$ kadmin -q "addprinc -randkey host/kdc01.example.com"
$ kadmin -q "addprinc -randkey host/kdc02.example.com"
```

> **Note**
>
> The `kadmin` command defaults to using a principal like `username/admin@EXAMPLE.COM`, where `username` is your current shell user. If you need to override that, use `-p <principal-you-want>`.

Extract the **key file** for the `kdc02` principal, which is the server we are on:

```
$ sudo kadmin -p ubuntu/admin -q "ktadd host/kdc02.example.com"
```

Next, there needs to be a `kpropd.acl` file on each KDC that lists all KDCs for the realm. For example, on both the **primary and secondary KDC**, create `/etc/krb5kdc/kpropd.acl`:

```
host/kdc01.example.com@EXAMPLE.COM
host/kdc02.example.com@EXAMPLE.COM
```

> **Note**
>
> It's customary to allow both KDCs because one may want to switch their roles if one goes bad. For such an eventuality, both are already listed here.

Create an empty database on the **secondary KDC**:

```
$ sudo kdb5_util create -s
```

Now install `kpropd` daemon, which listens for connections from the `kprop` utility from the **primary KDC**:

```
$ sudo apt install krb5-kpropd
```

The service will be running immediately after installation.

From a terminal on the **primary KDC**, create a dump file of the principal database:

```
$ sudo kdb5_util dump /var/lib/krb5kdc/dump
```

Still on the **Primary KDC**, extract its **key**:

```
$ sudo kadmin.local -q "ktadd host/kdc01.example.com"
```

On the **primary KDC**, run the `kprop` utility to push the database dump made before to the secondary KDC:

```
$ sudo kprop -r EXAMPLE.COM -f /var/lib/krb5kdc/dump kdc02.example.com
Database propagation to kdc02.example.com: SUCCEEDED
```

Note the `SUCCEEDED` message, which signals that the propagation worked. If there is an error message, check `/var/log/syslog` on the secondary KDC for more information.

You may also want to create a cron job to periodically update the database on the **secondary KDC**. For example, the following will push the database every hour:

```
# m h  dom mon dow   command
0 * * * * root /usr/sbin/kdb5_util dump /var/lib/krb5kdc/dump && /usr/sbin/kprop -
r EXAMPLE.COM -f /var/lib/krb5kdc/dump kdc02.example.com
```

Finally, start the `krb5-kdc` daemon on the **secondary KDC**:

```
$ sudo systemctl start krb5-kdc.service
```

**Note**

The secondary KDC does not run an admin server, since it's a read-only copy.

From now on, you can specify both KDC servers in `/etc/krb5.conf` for the `EXAMPLE.COM` realm, in any host participating in this realm (including `kdc01` and `kdc02`), but remember that there can only be one admin server and that's the one running on `kdc01`:

```
[realms]
    EXAMPLE.COM = {
            kdc = kdc01.example.com
            kdc = kdc02.example.com
            admin_server = kdc01.example.com
    }
```

The **secondary KDC** should now be able to issue tickets for the realm. You can test this by stopping the `krb5-kdc` daemon on the primary KDC, then using `kinit` to request a ticket. If all goes well you should receive a ticket from the secondary KDC. Otherwise, check `/var/log/syslog` and `/var/log/auth.log` on the secondary KDC.

### How to set up basic workstation authentication

In this section we'll look at configuring a Linux system as a Kerberos client. This will allow access to any "Kerber-ised" services once a user has successfully logged into the system.

Note that Kerberos alone is not enough for a user to exist in a Linux system. We cannot just point the system at a Kerberos server and expect all the Kerberos principals to be able to *log in* on the Linux system, simply because these users do not *exist* locally.

Kerberos only provides authentication: it doesn't know about user groups, Linux UIDs and *GIDs*, home directories, etc. Normally, another network source is used for this information, such as an LDAP or Windows server, and, in the old days, NIS was used for that as well.

### Set up a Linux system as a Kerberos client

If you have local users matching the principals in a Kerberos realm, and just want to switch the authentication from local to remote using Kerberos, you can follow this section. This is not a very usual scenario, but serves to highlight the separation between user authentication and user information (full name, UID, GID, home directory, groups, etc). If you just want to be able to grab tickets and use them, it's enough to install `krb5-user` and run `kinit`.

We are going to use `sssd` with a trick so that it will fetch the user information from the local system files, instead of a remote source which is the more common case.

### Install the required packages

To install the packages enter the following in a terminal prompt:

```
$ sudo apt install krb5-user sssd-krb5
```

You will be prompted for the addresses of your KDCs and admin servers. If you have followed our *how to install a Kerberos server* and *how to set up a secondary KDC* guides, the KDCs will be (space separated):

```
kdc01.example.com kdc02.example.com`
```

And the admin server will be:

```
kdc01.example.com
```

Remember that `kdc02` is a read-only copy of the primary KDC, so it doesn't run an admin server.

> **Note**
>
> If you have added the appropriate SRV records to *DNS*, none of those prompts will need answering.

### Configure Kerberos

If you missed the questions earlier, you can reconfigure the package to fill them in again:

```
sudo dpkg-reconfigure krb5-config
```

You can test the Kerberos configuration by requesting a ticket using the `kinit` utility. For example:

```
$ kinit ubuntu
Password for ubuntu@EXAMPLE.COM:
```

Note that `kinit` doesn't need the principal to exist as a local user in the system. In fact, you can `kinit` any principal you want. If you don't specify one, then the tool will use the username of whoever is running `kinit`.

### Configure `sssd`

The only remaining configuration now is for `sssd`. Create the file `/etc/sssd/sssd.conf` with the following content:

```
    [sssd]
    config_file_version = 2
    services = pam
    domains = example.com

    [pam]

    [domain/example.com]
    id_provider = proxy
    proxy_lib_name = files
    auth_provider = krb5
    krb5_server = kdc01.example.com,kdc02.example.com
    krb5_kpasswd = kdc01.example.com
    krb5_realm = EXAMPLE.COM
```

The above configuration will use Kerberos for **authentication** (`auth_provider`), but will use the local system users for user and group information (`id_provider`).

Adjust the permissions of the config file and start `sssd`:

```
$ sudo chown root:root /etc/sssd/sssd.conf
$ sudo chmod 0600 /etc/sssd/sssd.conf
$ sudo systemctl start sssd
```

Just by having installed `sssd` and its dependencies, PAM will already have been configured to use `sssd`, with a *fallback* to local user authentication. To try it out, if this is a workstation, simply switch users (in the GUI), or open a login terminal (Ctrl-Alt-number), or spawn a login shell with `sudo login`, and try logging in using the name of a Kerberos principal. Remember that this user must already exist on the local system:

```
$ sudo login
focal-krb5-client login: ubuntu
Password:
Welcome to Ubuntu Focal Fossa (development branch) (GNU/Linux 5.4.0-21-generic
x86_64)

(...)

Last login: Thu Apr  9 21:23:50 UTC 2020 from 10.20.20.1 on pts/0
$ klist
Ticket cache: FILE:/tmp/krb5cc_1000_NlfnSX
Default principal: ubuntu@EXAMPLE.COM

Valid starting     Expires            Service principal
04/09/20 21:36:12  04/10/20 07:36:12  krbtgt/EXAMPLE.COM@EXAMPLE.COM
    renew until 04/10/20 21:36:12
```

You will have a Kerberos ticket already right after login.

### How to set up Kerberos with OpenLDAP backend

Kerberos supports a few different database backends. The default one (which we have used in our other Kerberos guides so far) is called `db2`. The DB types documentation shows all the options, one of which is LDAP.

### Why use LDAP?

There are several reasons why one would want to have the Kerberos principals stored in LDAP as opposed to a local on-disk database. There are also cases when it is not a good idea. Each site has to evaluate the pros and cons. Here are a few:

- Pros:
  - OpenLDAP replication is faster and more robust than the native Kerberos one, based on a cron job
  - If you already have OpenLDAP set up for other things, such as storing users and groups, adding the Kerberos attributes can be beneficial, providing an integrated story

- Cons:
  - Setting up the LDAP backend isn't a trivial task and shouldn't be attempted by administrators without prior knowledge of OpenLDAP
  - As highlighted in the LDAP section of DB types, since `krb5kdc` is single-threaded there may be higher latency in servicing requests when using the OpenLDAP backend

### In this guide

In this section we'll configure a primary and secondary Kerberos server to use OpenLDAP for the principal database. Note that as of version 1.18, the Key Distribution Center (KDC) from MIT Kerberos does not support a primary KDC using a read-only consumer (secondary) LDAP server. What we have to consider here is that a primary KDC is read-write, and it needs a read-write backend. The secondary KDCs can use both a read-write and read-only backend, because they are expected to be read-only. Therefore there are only some possible layouts we can use:

1. **Simple case**:
   - Primary KDC connected to primary OpenLDAP
   - Secondary KDC connected to both primary and secondary OpenLDAP
2. **Extended simple case**:
   - Multiple primary KDCs connected to one primary OpenLDAP
   - Multiple secondary KDCs connected to primary and secondary OpenLDAP
3. **OpenLDAP with multi-master replication**:
   - Multiple primary KDCs connected to all primary OpenLDAP servers

We haven't covered OpenLDAP multi-master replication in this guide, so we will show the **simple case** only. The second scenario is an extension: just add another primary KDC to the mix, talking to the same primary OpenLDAP server.

### Configure OpenLDAP

We are going to install the OpenLDAP server on the same host as the KDC, to simplify the communication between them. In such a setup, we can use the `ldapi:///` transport, which is via a UNIX socket, and we don't need to set up SSL certificates to secure the communication between the Kerberos services and OpenLDAP. Note, however, that SSL is still needed for the OpenLDAP replication. See *LDAP with TLS* for details.

If you want to use an existing OpenLDAP server, that's also possible, but keep in mind that you should then use SSL for the communication between the KDC and this OpenLDAP server.

First, the necessary **schema** needs to be loaded on an OpenLDAP server that has network connectivity to both the **primary** and **secondary** KDCs. The rest of this section assumes that you also have LDAP replication configured between at least two servers. For information on setting up OpenLDAP see *OpenLDAP Server*.

> **Note**
>
> `cn=admin,dc=example,dc=com` is a default admin user that is created during the installation of the `slapd` package (the OpenLDAP server). The domain component will change for your server, so adjust accordingly.

- Install the necessary packages (it's assumed that OpenLDAP is already installed):

```
sudo apt install krb5-kdc-ldap krb5-admin-server
```

- Next, extract the `kerberos.schema.gz` file:

```
sudo cp /usr/share/doc/krb5-kdc-ldap/kerberos.schema.gz /etc/ldap/schema/
sudo gunzip /etc/ldap/schema/kerberos.schema.gz
```

- The **Kerberos schema** needs to be added to the `cn=config` tree. This schema file needs to be converted to LDIF format before it can be added. For that we will use a helper tool, called `schema2ldif`, provided by the package of the same name which is available in the Universe archive:

```
sudo apt install schema2ldif
```

- To import the Kerberos schema, run:

```
$ sudo ldap-schema-manager -i kerberos.schema
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
executing 'ldapadd -Y EXTERNAL -H ldapi:/// -f /etc/ldap/schema/kerberos.ldif
'
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
adding new entry "cn=kerberos,cn=schema,cn=config"
```

- With the new schema loaded, let's index an attribute often used in searches:

```
$ sudo ldapmodify -Q -Y EXTERNAL -H ldapi:/// <<EOF
dn: olcDatabase={1}mdb,cn=config
add: olcDbIndex
olcDbIndex: krbPrincipalName eq,pres,sub
EOF

modifying entry "olcDatabase={1}mdb,cn=config"
```

- Let's create LDAP entries for the Kerberos administrative entities that will contact the OpenLDAP server to perform operations. There are two:
  - `ldap_kdc_dn`: needs to have read rights on the realm container, principal container and realm sub-trees. If `disable_last_success` and `disable_lockout` are not set, however, then `ldap_kdc_dn` needs write access to the Kerberos container just like the admin *DN* below.

  - `ldap_kadmind_dn`: needs to have read and write rights on the realm container, principal container and realm sub-trees

Here is the command to create these entities:

```
$ ldapadd -x -D cn=admin,dc=example,dc=com -W <<EOF
dn: uid=kdc-service,dc=example,dc=com
uid: kdc-service
objectClass: account
objectClass: simpleSecurityObject
userPassword: {CRYPT}x
description: Account used for the Kerberos KDC

dn: uid=kadmin-service,dc=example,dc=com
uid: kadmin-service
objectClass: account
objectClass: simpleSecurityObject
userPassword: {CRYPT}x
description: Account used for the Kerberos Admin server
EOF
Enter LDAP Password:
adding new entry "uid=kdc-service,dc=example,dc=com"

adding new entry "uid=kadmin-service,dc=example,dc=com"
```

Now let's set a password for them. Note that first the tool asks for the password you want for the specified user DN, and then for the password of the `cn=admin` DN:

```
$ ldappasswd -x -D cn=admin,dc=example,dc=com -W -S uid=kdc-service,
dc=example,dc=com
New password:   <-- password you want for uid-kdc-service
Re-enter new password:
Enter LDAP Password:  <-- password for the dn specified with the -D option
```

Repeat for the `uid=kadmin-service` dn. These passwords will be needed later.

You can test these with `ldapwhoami`:

```
$ ldapwhoami -x -D uid=kdc-service,dc=example,dc=com -W
Enter LDAP Password:
dn:uid=kdc-service,dc=example,dc=com
```

- Finally, update the Access Control Lists (*ACL*). These can be tricky, as it highly depends on what you have defined already. By default, the `slapd` package configures your database with the following ACLs:

```
olcAccess: {0}to attrs=userPassword by self write by anonymous auth by * none
olcAccess: {1}to attrs=shadowLastChange by self write by * read
olcAccess: {2}to * by * read
```

We need to insert new rules before the final `to * by * read` one, to control access to the Kerberos related entries and attributes:

```
$ sudo ldapmodify -Q -Y EXTERNAL -H ldapi:/// <<EOF
dn: olcDatabase={1}mdb,cn=config
add: olcAccess
olcAccess: {2}to attrs=krbPrincipalKey
  by anonymous auth
  by dn.exact="uid=kdc-service,dc=example,dc=com" read
  by dn.exact="uid=kadmin-service,dc=example,dc=com" write
  by self write
  by * none
-
add: olcAccess
olcAccess: {3}to dn.subtree="cn=krbContainer,dc=example,dc=com"
  by dn.exact="uid=kdc-service,dc=example,dc=com" read
  by dn.exact="uid=kadmin-service,dc=example,dc=com" write
  by * none
EOF

modifying entry "olcDatabase={1}mdb,cn=config"

This will make the existing {2} rule become {4}. Check with sudo slapcat -b
cn=config (the output below was reformatted a bit for clarity):

olcAccess: {0}to attrs=userPassword
    by self write
    by anonymous auth
    by * none
olcAccess: {1}to attrs=shadowLastChange
    by self write
    by * read
olcAccess: {2}to attrs=krbPrincipalKey by anonymous auth
        by dn.exact="uid=kdc-service,dc=example,dc=com" read
        by dn.exact="uid=kadmin-service,dc=example,dc=com" write
        by self write
        by * none
olcAccess: {3}to dn.subtree="cn=krbContainer,dc=example,dc=com"
        by dn.exact="uid=kdc-service,dc=example,dc=com" read
        by dn.exact="uid=kadmin-service,dc=example,dc=com" write
        by * none
olcAccess: {4}to * by * read
```

Your LDAP directory is now ready to serve as a Kerberos principal database.

### Primary KDC configuration (LDAP)

With OpenLDAP configured it is time to configure the KDC. In this example we are doing it in the same OpenLDAP server to take advantage of local UNIX socket communication.

- Reconfigure the `krb5-config` package if needed to get a good starting point with `/etc/krb5.conf`:

```
sudo dpkg-reconfigure krb5-config
```

- Now edit `/etc/krb5.conf` adding the `database_module` option to the `EXAMPLE.COM` realm section:

```
[realms]
        EXAMPLE.COM = {
                kdc = kdc01.example.com
                kdc = kdc02.example.com
                admin_server = kdc01.example.com
                default_domain = example.com
                database_module = openldap_ldapconf
        }
```

  Then also add these new sections:

```
[dbdefaults]
        ldap_kerberos_container_dn = cn=krbContainer,dc=example,dc=com

[dbmodules]
        openldap_ldapconf = {
                db_library = kldap

                                # if either of these is false, then the ldap_
kdc_dn needs to
                                # have write access
                                disable_last_success = true
                                disable_lockout  = true

                # this object needs to have read rights on
                # the realm container, principal container and realm sub-
trees
                ldap_kdc_dn = "uid=kdc-service,dc=example,dc=com"

                # this object needs to have read and write rights on
                # the realm container, principal container and realm sub-
trees
                ldap_kadmind_dn = "uid=kadmin-service,dc=example,dc=com"

                ldap_service_password_file = /etc/krb5kdc/service.keyfile
                ldap_servers = ldapi:///
                ldap_conns_per_server = 5
        }
```

- Next, use the `kdb5_ldap_util` utility to create the realm:

```
$ sudo kdb5_ldap_util -D cn=admin,dc=example,dc=com create -subtrees
dc=example,dc=com -r EXAMPLE.COM -s -H ldapi:///
Password for "cn=admin,dc=example,dc=com":
Initializing database for realm 'EXAMPLE.COM'
You will be prompted for the database Master Password.
```

(continues on next page)

(continued from previous page)

```
It is important that you NOT FORGET this password.
Enter KDC database master key:
Re-enter KDC database master key to verify:
```

- Create a stash of the password used to bind to the LDAP server. Run it once for each `ldap_kdc_dn` and `ldap_kadmin_dn`:

```
sudo kdb5_ldap_util -D cn=admin,dc=example,dc=com stashsrvpw -f /etc/krb5kdc/
service.keyfile uid=kdc-service,dc=example,dc=com
sudo kdb5_ldap_util -D cn=admin,dc=example,dc=com stashsrvpw -f /etc/krb5kdc/
service.keyfile uid=kadmin-service,dc=example,dc=com
```

> **Note**
>
> The `/etc/krb5kdc/service.keyfile` file now contains clear text versions of the passwords used by the KDC to contact the LDAP server!

- Create a `/etc/krb5kdc/kadm5.acl` file for the admin server, if you haven't already:

```
*/admin@EXAMPLE.COM        *
```

- Start the Kerberos KDC and admin server:

```
sudo systemctl start krb5-kdc.service krb5-admin-server.service
```

You can now add Kerberos principals to the LDAP database, and they will be copied to any other LDAP servers configured for replication. To add a principal using the `kadmin.local` utility enter:

```
$ sudo kadmin.local
Authenticating as principal root/admin@EXAMPLE.COM with password.
kadmin.local:  addprinc ubuntu
WARNING: no policy specified for ubuntu@EXAMPLE.COM; defaulting to no policy
Enter password for principal "ubuntu@EXAMPLE.COM":
Re-enter password for principal "ubuntu@EXAMPLE.COM":
Principal "ubuntu@EXAMPLE.COM" created.
kadmin.local:
```

The above will create an `ubuntu` principal with a DN of `krbPrincipalName=ubuntu@EXAMPLE.COM,cn=EXAMPLE.COM,cn=krbContainer,dc=example,dc=com`.

Let's say, however, that you already have a user in your directory, and it's in `uid=testuser1,ou=People,dc=example,dc=com`. How can you add the Kerberos attributes to it? You use the `-x` parameter to specify the location. For the `ldap_kadmin_dn` to be able to write to it, we first need to update the ACLs:

```
$ sudo ldapmodify -Q -Y EXTERNAL -H ldapi:/// <<EOF
dn: olcDatabase={1}mdb,cn=config
add: olcAccess
olcAccess: {4}to dn.subtree="ou=People,dc=example,dc=com"
```

(continues on next page)

(continued from previous page)

```
    by dn.exact="uid=kdc-service,dc=example,dc=com" read
    by dn.exact="uid=kadmin-service,dc=example,dc=com" write
    by * break
EOF
```

And now we can specify the new location:

```
$ sudo kadmin.local
Authenticating as principal root/admin@EXAMPLE.COM with password.
kadmin.local:  addprinc -x dn=uid=testuser1,ou=People,dc=example,dc=com testuser1
WARNING: no policy specified for testuser1@EXAMPLE.COM; defaulting to no policy
Enter password for principal "testuser1@EXAMPLE.COM":
Re-enter password for principal "testuser1@EXAMPLE.COM":
Principal "testuser1@EXAMPLE.COM" created.
```

Since the specified DN already exists, `kadmin.local` will just add the required Kerberos attributes to this existing entry. If it didn't exist, it would be created from scratch, with only the Kerberos attributes, just like what happened with the `ubuntu` example above, but in the specified location.

**Note**

The `ldap_kadmin_dn` DN (`uid=kadmin-service` in our example) does not have write access to the location specified by the `-x` parameter, you will get an `Insufficient access` error.

Both places are visible for `kinit`, since, when the realm was created with `kdb5_ldap_util`, the default value for the search scope and base were taken: `subtree`, and `dc=example,dc=com`.

### Secondary KDC configuration (LDAP)

The setup of the secondary KDC (and its OpenLDAP replica) is very similar. Once you have the OpenLDAP replication setup, repeat these steps on the secondary:

- Install `krb5-kdc-ldap`, `ldap-utils`. Do **not** install `krb5-admin-server`.
- Load the Kerberos schema using `schema2ldif`.
- Add the index for `krbPrincipalName`.
- Add the ACLs.
- Configure `krb5.conf` in the same way, initially. If you want to, and if you configured SSL properly, you can add `ldaps://kdc01.example.com` to the `ldap_servers` list after `ldapi://`, so that the secondary KDC can have two LDAP backends at its disposal.
- **DO NOT** run `kdb5_ldap_util`. There is no need to create the database since it's being replicated from the primary.
- Copy over the following files from the primary KDC and place them in the same location on the secondary:
  - `/etc/krb5kdc/stash`
  - `/etc/krb5kdc/service.keyfile`

- Start the KDC: `sudo systemctl start krb5-kdc.service`

### Resources

- Configuring Kerberos with OpenLDAP back-end
- MIT Kerberos backend types

### See also

- Explanation: *Introduction to Kerberos*

## Network user authentication with SSSD

These guides will show you how to set up network user authentication with SSSD with…

### How to set up SSSD with Active Directory

This section describes the use of SSSD to authenticate user logins against an Active Directory via using SSSD's "ad" provider. At the end, Active Directory users will be able to log in on the host using their AD credentials. Group membership will also be maintained.

### Group Policies for Ubuntu

SSSD manages user authentication and sets initial security policies.

*ADSys* serves as a Group Policy client for Ubuntu, streamlining the configuration of Ubuntu systems within a Microsoft Active Directory environment. If you are interested in Group Policies support for Ubuntu, detailed information can be found in the ADSys documentation.

### Prerequisites and assumptions

This guide does not explain Active Directory, how it works, how to set one up, or how to maintain it. It assumes that a working Active Directory domain is already configured and you have access to the credentials to join a machine to that domain.

- The domain controller is:
  - Acting as an authoritative *DNS* server for the domain.
  - The primary DNS resolver (check with `resolvectl status`).
- System time is correct and in sync, maintained via a service like `chrony` or `ntp`.
- The domain used in this example is `ad1.example.com` .

### Install necessary software

Install the following packages:

```
sudo apt install sssd-ad sssd-tools realmd adcli
```

### Join the domain

We will use the `realm` command, from the `realmd` package, to join the domain and create the SSSD configuration.

Let's verify the domain is discoverable via DNS:

```
$ sudo realm -v discover ad1.example.com
 * Resolving: _ldap._tcp.ad1.example.com
 * Performing LDAP DSE lookup on: 10.51.0.5
 * Successfully discovered: ad1.example.com
ad1.example.com
  type: kerberos
  realm-name: AD1.EXAMPLE.COM
  domain-name: ad1.example.com
  configured: no
  server-software: active-directory
  client-software: sssd
  required-package: sssd-tools
  required-package: sssd
  required-package: libnss-sss
  required-package: libpam-sss
  required-package: adcli
  required-package: samba-common-bin
```

This performs several checks and determines the best software stack to use with SSSD. SSSD can install the missing packages via `packagekit`, but we already installed them in the previous step.

Now let's join the domain:

```
$ sudo realm join ad1.example.com
Password for Administrator:
```

That was quite uneventful. If you want to see what it was doing, pass the `-v` option:

```
$ sudo realm join -v ad1.example.com
 * Resolving: _ldap._tcp.ad1.example.com
 * Performing LDAP DSE lookup on: 10.51.0.5
 * Successfully discovered: ad1.example.com
Password for Administrator:
 * Unconditionally checking packages
 * Resolving required packages
 * LANG=C /usr/sbin/adcli join --verbose --domain ad1.example.com --domain-realm
AD1.EXAMPLE.COM --domain-controller 10.51.0.5 --login-type user --login-user
Administrator --stdin-password
 * Using domain name: ad1.example.com
 * Calculated computer account name from fqdn: AD-CLIENT
 * Using domain realm: ad1.example.com
 * Sending NetLogon ping to domain controller: 10.51.0.5
 * Received NetLogon info from: SERVER1.ad1.example.com
 * Wrote out krb5.conf snippet to /var/cache/realmd/adcli-krb5-hUfTUg/krb5.d/
adcli-krb5-conf-hv2kzi
```

(continues on next page)

(continued from previous page)

```
 * Authenticated as user: Administrator@AD1.EXAMPLE.COM
 * Looked up short domain name: AD1
 * Looked up domain SID: S-1-5-21-2660147319-831819607-3409034899
 * Using fully qualified name: ad-client.ad1.example.com
 * Using domain name: ad1.example.com
 * Using computer account name: AD-CLIENT
 * Using domain realm: ad1.example.com
 * Calculated computer account name from fqdn: AD-CLIENT
 * Generated 120 character computer password
 * Using keytab: FILE:/etc/krb5.keytab
 * Found computer account for AD-CLIENT$ at: CN=AD-CLIENT,CN=Computers,DC=ad1,
DC=example,DC=com
 * Sending NetLogon ping to domain controller: 10.51.0.5
 * Received NetLogon info from: SERVER1.ad1.example.com
 * Set computer password
 * Retrieved kvno '3' for computer account in directory: CN=AD-CLIENT,
CN=Computers,DC=ad1,DC=example,DC=com
 * Checking RestrictedKrbHost/ad-client.ad1.example.com
 *    Added RestrictedKrbHost/ad-client.ad1.example.com
 * Checking RestrictedKrbHost/AD-CLIENT
 *    Added RestrictedKrbHost/AD-CLIENT
 * Checking host/ad-client.ad1.example.com
 *    Added host/ad-client.ad1.example.com
 * Checking host/AD-CLIENT
 *    Added host/AD-CLIENT
 * Discovered which keytab salt to use
 * Added the entries to the keytab: AD-CLIENT$@AD1.EXAMPLE.COM: FILE:/etc/krb5.
keytab
 * Added the entries to the keytab: host/AD-CLIENT@AD1.EXAMPLE.COM: FILE:/etc/
krb5.keytab
 * Added the entries to the keytab: host/ad-client.ad1.example.com@AD1.EXAMPLE.
COM: FILE:/etc/krb5.keytab
 * Added the entries to the keytab: RestrictedKrbHost/AD-CLIENT@AD1.EXAMPLE.COM:
FILE:/etc/krb5.keytab
 * Added the entries to the keytab: RestrictedKrbHost/ad-client.ad1.example.
com@AD1.EXAMPLE.COM: FILE:/etc/krb5.keytab
 * /usr/sbin/update-rc.d sssd enable
 * /usr/sbin/service sssd restart
 * Successfully enrolled machine in realm
```

By default, `realm` will use the "Administrator" account of the domain to request the join. If you need to use another account, pass it to the tool with the `-U` option.

Another popular way of joining a domain is using a One Time Password (OTP) token. For that, use the `--one-time-password` option.

### SSSD configuration

The `realm` tool already took care of creating an SSSD configuration, adding the PAM and NSS modules, and starting the necessary services.

Let's take a look at `/etc/sssd/sssd.conf`:

```
[sssd]
domains = ad1.example.com
config_file_version = 2
services = nss, pam

[domain/ad1.example.com]
default_shell = /bin/bash
krb5_store_password_if_offline = True
cache_credentials = True
krb5_realm = AD1.EXAMPLE.COM
realmd_tags = manages-system joined-with-adcli
id_provider = ad
fallback_homedir = /home/%u@%d
ad_domain = ad1.example.com
use_fully_qualified_names = True
ldap_id_mapping = True
access_provider = ad
```

**Note**

Something very important to remember is that this file must have permissions `0600` and ownership `root:root`, or else SSSD won't start!

**Note**

[FAILED] Failed to listen on sssd-nss.socket… [FAILED] Failed to listen on sssd-pam.socket… [FAILED] Dependency failed for sssd-pam-priv.socket…

If you get those errors on startup, just remove the line `services = nss, pam`

Let's highlight a few things from this config file:

- `cache_credentials`: This allows logins when the AD server is unreachable
- `fallback_homedir`: The home directory. By default, `/home/<user>@<domain>`. For example, the AD user *john* will have a home directory of */home/john@ad1.example.com*.
- `use_fully_qualified_names`: Users will be of the form `user@domain`, not just `user`. This should only be changed if you are certain no other domains will ever join the AD forest, via one of the several possible trust relationships.

### Automatic home directory creation

What the `realm` tool didn't do for us is setup `pam_mkhomedir`, so that network users can get a home directory when they login. This remaining step can be done by running the following command:

```
sudo pam-auth-update --enable mkhomedir
```

### Testing our setup

You should now be able to fetch information about AD users. In this example, `John Smith` is an AD user:

```
$ getent passwd john@ad1.example.com
john@ad1.example.com:*:1725801106:1725800513:John Smith:/home/john@ad1.example.
com:/bin/bash
```

Let's see his groups:

```
$ groups john@ad1.example.com
john@ad1.example.com : domain users@ad1.example.com engineering@ad1.example.com
```

**Note**

If you just changed the group membership of a user, it may be a while before SSSD notices due to caching.

Finally, how about we try a login:

```
$ sudo login
ad-client login: john@ad1.example.com
Password:
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-24-generic x86_64)
...
Creating directory '/home/john@ad1.example.com'.
john@ad1.example.com@ad-client:~$
```

Notice how the home directory was automatically created.

You can also use SSH, but note that the command will look a bit funny because of the multiple `@` signs:

```
$ ssh john@ad1.example.com@10.51.0.11
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-24-generic x86_64)
(...)
Last login: Thu Apr 16 21:22:55 2020
john@ad1.example.com@ad-client:~$
```

**Note**

In the SSH example, public key authentication was used, so no password was required. Remember that SSH password authentication is disabled by default in `/etc/ssh/sshd_config`.

### Kerberos tickets

If you install `krb5-user`, your AD users will also get a Kerberos ticket upon logging in:

```
john@ad1.example.com@ad-client:~$ klist
Ticket cache: FILE:/tmp/krb5cc_1725801106_9UxVIz
Default principal: john@AD1.EXAMPLE.COM

Valid starting     Expires            Service principal
04/16/20 21:32:12  04/17/20 07:32:12  krbtgt/AD1.EXAMPLE.COM@AD1.EXAMPLE.COM
        renew until 04/17/20 21:32:12
```

**Note**

`realm` also configured `/etc/krb5.conf` for you, so there should be no further configuration prompts when installing `krb5-user`.

Let's test with `smbclient` using Kerberos authentication to list the shares of the domain controller:

```
john@ad1.example.com@ad-client:~$ smbclient -k -L server1.ad1.example.com

        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        C$              Disk      Default share
        IPC$            IPC       Remote IPC
        NETLOGON        Disk      Logon server share
        SYSVOL          Disk      Logon server share
SMB1 disabled -- no workgroup available
```

Notice how we now have a ticket for the `cifs` service, which was used for the share list above:

```
john@ad1.example.com@ad-client:~$ klist
Ticket cache: FILE:/tmp/krb5cc_1725801106_9UxVIz
Default principal: john@AD1.EXAMPLE.COM

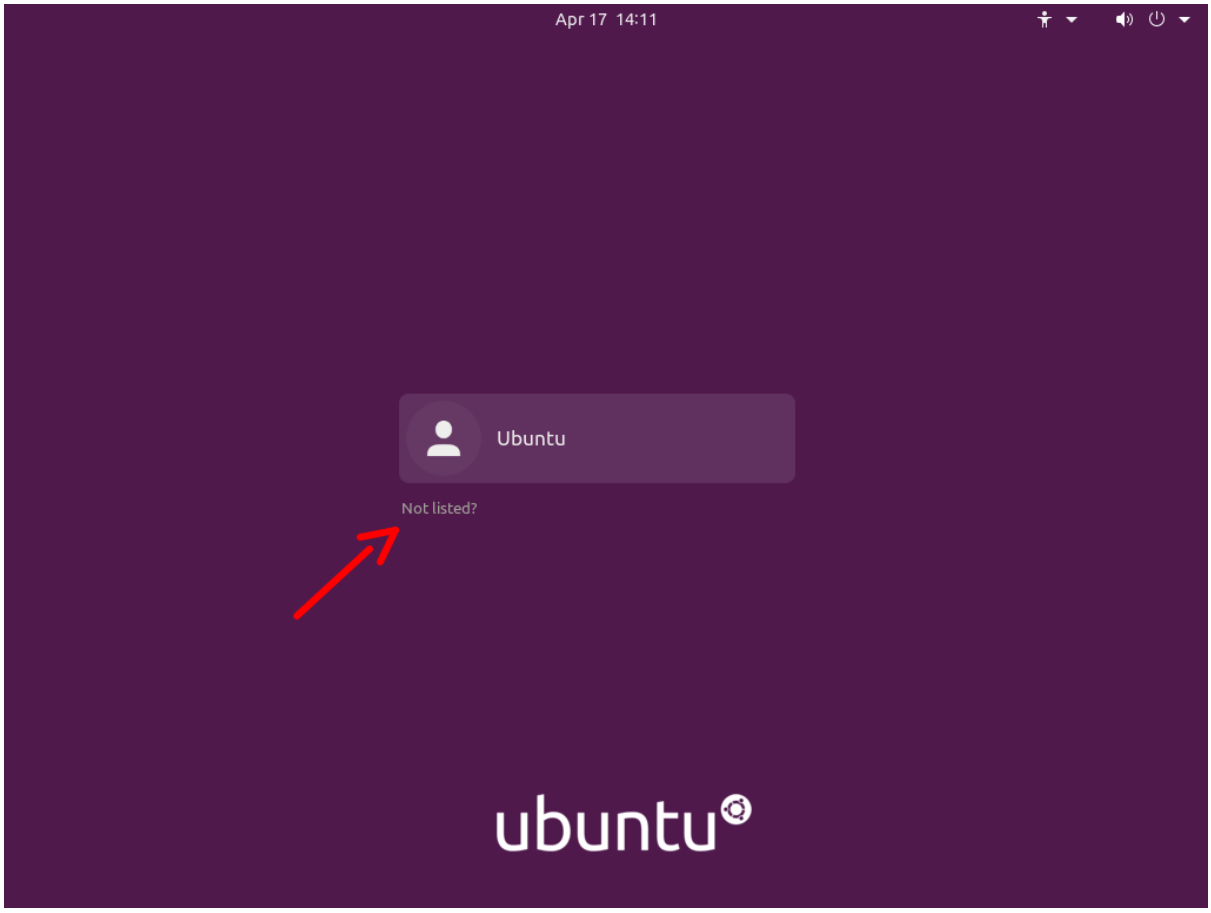
Valid starting     Expires            Service principal
04/16/20 21:32:12  04/17/20 07:32:12  krbtgt/AD1.EXAMPLE.COM@AD1.EXAMPLE.COM
        renew until 04/17/20 21:32:12
04/16/20 21:32:21  04/17/20 07:32:12  cifs/server1.ad1.example.com@AD1.EXAMPLE.COM
```

### Ubuntu Desktop authentication

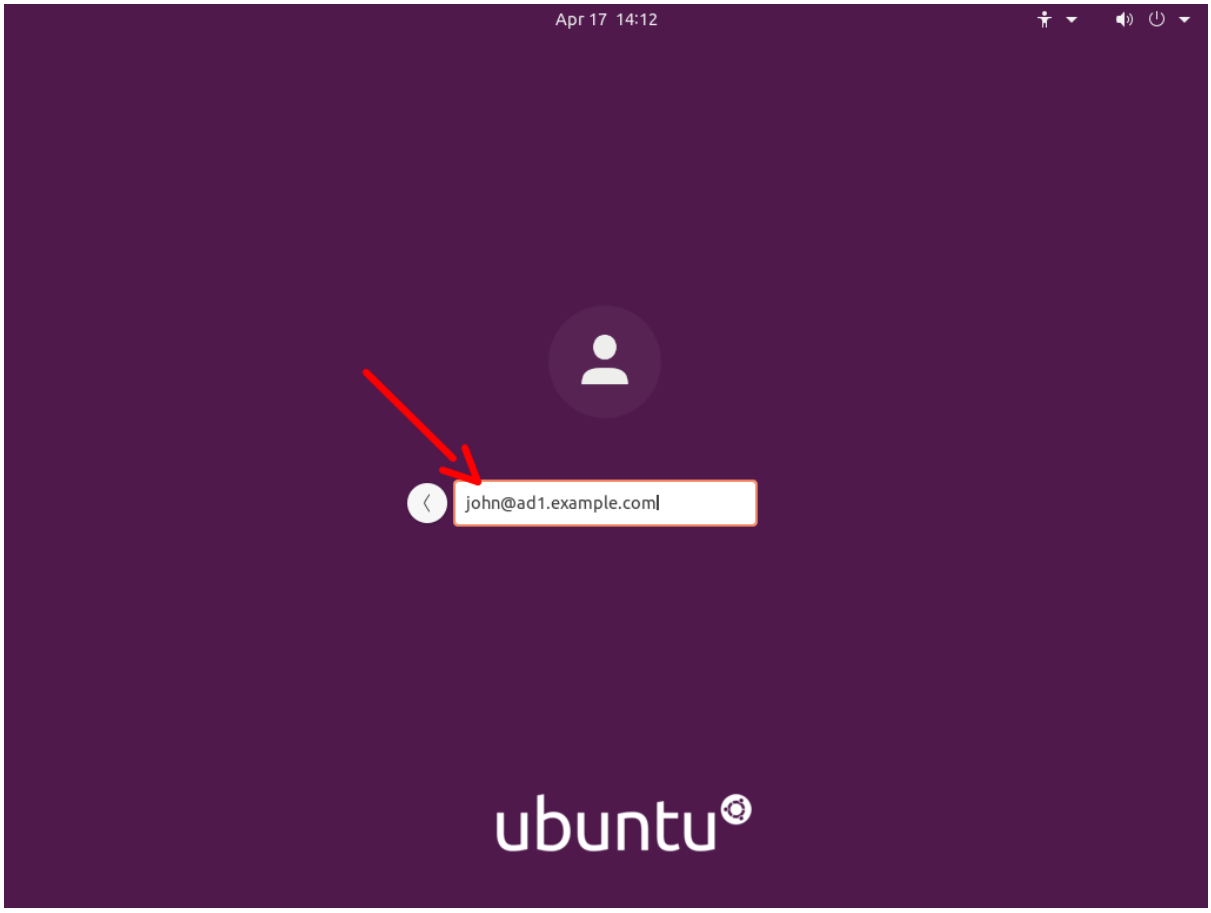
The desktop login only shows local users in the list to pick from, and that's on purpose.

To login with an Active Directory user for the first time, follow these steps:

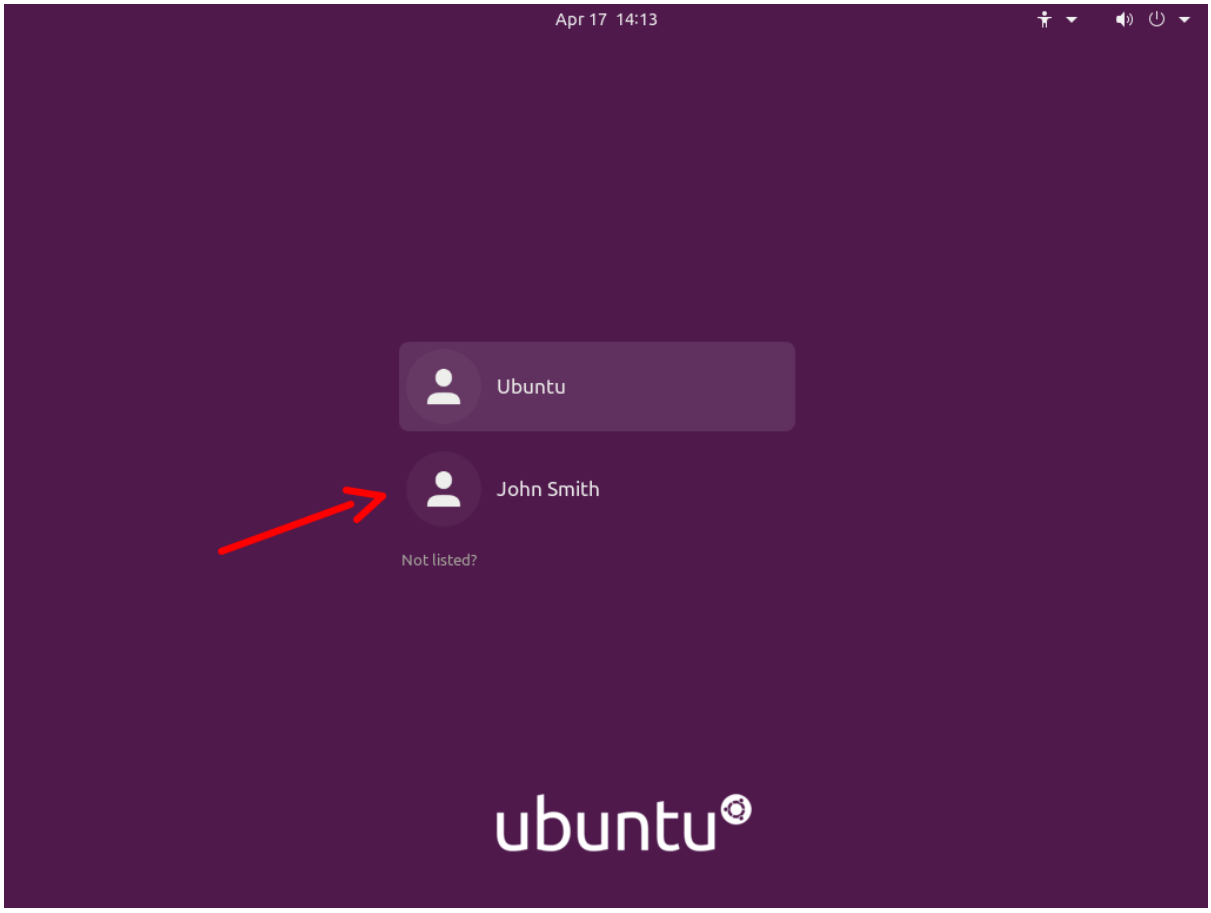
- Click on the "Not listed?" option:




- Type in the login name followed by the password:




- Next time you login, the AD user will be listed as if it was a local user:




### Known issues

When logging in on a system joined with an Active Directory domain, `sssd` (the package responsible for this integration) will try to apply Group Policies by default. There are cases where if a specific policy is missing, the login will be denied.

This is being tracked in bug #1934997. Until the fix becomes available, please see comment #5 in that bug report for existing workarounds.

### Further reading

- GitHub SSSD Project
- Active Directory DNS Zone Entries

### How to set up SSSD with LDAP

SSSD can also use LDAP for authentication, authorisation, and user/group information. In this section we will configure a host to authenticate users from an OpenLDAP directory.

### Prerequisites and assumptions

For this setup, we need:

- An existing OpenLDAP server with SSL enabled and using the RFC2307 schema for users and groups

- A client host where we will install the necessary tools and login as a user from the LDAP server

### Install necessary software

Install the following packages:

```
sudo apt install sssd-ldap ldap-utils
```

### Configure SSSD

Create the `/etc/sssd/sssd.conf` configuration file, with permissions `0600` and ownership `root:root`, and add the following content:

```
[sssd]
config_file_version = 2
domains = example.com

[domain/example.com]
id_provider = ldap
auth_provider = ldap
ldap_uri = ldap://ldap01.example.com
cache_credentials = True
ldap_search_base = dc=example,dc=com
```

Make sure to start the `sssd` service:

```
sudo systemctl start sssd.service
```

> **Note**
>
> `sssd` will use `START_TLS` by default for authentication requests against the LDAP server (the `auth_provider`), but not for the `id_provider`. If you want to also enable `START_TLS` for the `id_provider`, specify `ldap_id_use_start_tls = true`.

### Automatic home directory creation

To enable automatic home directory creation, run the following command:

```
sudo pam-auth-update --enable mkhomedir
```

### Check SSL setup on the client

The client must be able to use `START_TLS` when connecting to the LDAP server, with full certificate checking. This means:

- The client host knows and trusts the CA that signed the LDAP server certificate,
- The server certificate was issued for the correct host (`ldap01.example.com` in this guide),
- The time is correct on all hosts performing the TLS connection, and

- That neither certificate (CA or server's) expired.

If using a custom CA, an easy way to have a host trust it is to place it in `/usr/local/share/ca-certificates/` with a `.crt` extension and run `sudo update-ca-certificates`.

Alternatively, you can edit `/etc/ldap/ldap.conf` and point `TLS_CACERT` to the CA public key file.

**Note**

You may have to restart `sssd` after these changes: `sudo systemctl restart sssd`

Once that is all done, check that you can connect to the LDAP server using verified SSL connections:

```
$ ldapwhoami -x -ZZ -H ldap://ldap01.example.com
anonymous
```

and for `ldaps` (if enabled in `/etc/default/slapd`):

```
$ ldapwhoami -x -H ldaps://ldap01.example.com
```

The `-ZZ` parameter tells the tool to use `START_TLS`, and that it must not fail. If you have LDAP logging enabled on the server, it will show something like this:

```
slapd[779]: conn=1032 op=0 STARTTLS
slapd[779]: conn=1032 op=0 RESULT oid= err=0 text=
slapd[779]: conn=1032 fd=15 TLS established tls_ssf=256 ssf=256
slapd[779]: conn=1032 op=1 BIND dn="" method=128
slapd[779]: conn=1032 op=1 RESULT tag=97 err=0 text=
slapd[779]: conn=1032 op=2 EXT oid=1.3.6.1.4.1.4203.1.11.3
slapd[779]: conn=1032 op=2 WHOAMI
slapd[779]: conn=1032 op=2 RESULT oid= err=0 text=
```

`START_TLS` with `err=0` and `TLS established` is what we want to see there, and, of course, the `WHOAMI` extended operation.

### Final verification

In this example, the LDAP server has the following user and group entry we are going to use for testing:

```
dn: uid=john,ou=People,dc=example,dc=com
uid: john
objectClass: inetOrgPerson
objectClass: posixAccount
cn: John Smith
sn: Smith
givenName: John
mail: john@example.com
userPassword: johnsecret
uidNumber: 10001
```

(continues on next page)

(continued from previous page)

```
gidNumber: 10001
loginShell: /bin/bash
homeDirectory: /home/john

dn: cn=john,ou=Group,dc=example,dc=com
cn: john
objectClass: posixGroup
gidNumber: 10001
memberUid: john

dn: cn=Engineering,ou=Group,dc=example,dc=com
cn: Engineering
objectClass: posixGroup
gidNumber: 10100
memberUid: john
```

The user `john` should be known to the system:

```
ubuntu@ldap-client:~$ getent passwd john
john:*:10001:10001:John Smith:/home/john:/bin/bash

ubuntu@ldap-client:~$ id john
uid=10001(john) gid=10001(john) groups=10001(john),10100(Engineering)
```

And we should be able to authenticate as `john`:

```
ubuntu@ldap-client:~$ sudo login
ldap-client login: john
Password:
Welcome to Ubuntu Focal Fossa (development branch) (GNU/Linux 5.4.0-24-generic
x86_64)
(...)
Creating directory '/home/john'.
john@ldap-client:~$
```

### How to set up SSSD with LDAP and Kerberos

With SSSD we can create a setup that is very similar to Active Directory in terms of the technologies used: using LDAP for users and groups, and Kerberos for authentication.

### Prerequisites and assumptions

For this setup, we will need:

- An existing *OpenLDAP server* using the RFC2307 schema for users and groups. SSL support is recommended, but not strictly necessary because authentication in this setup is being done via Kerberos, and not LDAP.
- A *Kerberos server*. It doesn't have to be *using the OpenLDAP backend*.
- A client host where we will install and configure SSSD.

### Install necessary software

On the client host, install the following packages:

```
sudo apt install sssd-ldap sssd-krb5 ldap-utils krb5-user
```

You may be asked about the default Kerberos realm. For this guide, we are using `EXAMPLE.COM`.

At this point, you should already be able to obtain tickets from your Kerberos server, assuming *DNS* records point at it:

```
$ kinit ubuntu
Password for ubuntu@EXAMPLE.COM:

ubuntu@ldap-krb-client:~$ klist
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: ubuntu@EXAMPLE.COM

Valid starting     Expires            Service principal
04/17/20 19:51:06  04/18/20 05:51:06  krbtgt/EXAMPLE.COM@EXAMPLE.COM
        renew until 04/18/20 19:51:05
```

But we want to be able to login as an LDAP user, authenticated via Kerberos. Let's continue with the configuration.

### Configure SSSD

Create the `/etc/sssd/sssd.conf` configuration file, with permissions `0600` and ownership `root:root`, and add the following content:

```
[sssd]
config_file_version = 2
domains = example.com

[domain/example.com]
id_provider = ldap
ldap_uri = ldap://ldap01.example.com
ldap_search_base = dc=example,dc=com
auth_provider = krb5
krb5_server = kdc01.example.com,kdc02.example.com
krb5_kpasswd = kdc01.example.com
krb5_realm = EXAMPLE.COM
cache_credentials = True
```

This example uses two KDCs, which made it necessary to also specify the `krb5_kpasswd` server because the second KDC is a replica and is not running the admin server.

Start the `sssd` service:

```
sudo systemctl start sssd.service
```

### Automatic home directory creation

To enable automatic home directory creation, run the following command:

```
sudo pam-auth-update --enable mkhomedir
```

### Final verification

In this example, the LDAP server has the following user and group entry we are going to use for testing:

```
dn: uid=john,ou=People,dc=example,dc=com
uid: john
objectClass: inetOrgPerson
objectClass: posixAccount
cn: John Smith
sn: Smith
givenName: John
mail: john@example.com
uidNumber: 10001
gidNumber: 10001
loginShell: /bin/bash
homeDirectory: /home/john

dn: cn=john,ou=Group,dc=example,dc=com
cn: john
objectClass: posixGroup
gidNumber: 10001
memberUid: john

dn: cn=Engineering,ou=Group,dc=example,dc=com
cn: Engineering
objectClass: posixGroup
gidNumber: 10100
memberUid: john
```

Note how the user `john` has no `userPassword` attribute.

The user `john` should be known to the system:

```
ubuntu@ldap-client:~$ getent passwd john
john:*:10001:10001:John Smith:/home/john:/bin/bash

ubuntu@ldap-client:~$ id john
uid=10001(john) gid=10001(john) groups=10001(john),10100(Engineering)
```

Let's try a login as this user:

```
ubuntu@ldap-krb-client:~$ sudo login
ldap-krb-client login: john
Password:
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-24-generic x86_64)
```

(continues on next page)

(continued from previous page)

```
(...)
Creating directory '/home/john'.

john@ldap-krb-client:~$ klist
Ticket cache: FILE:/tmp/krb5cc_10001_BOrxWr
Default principal: john@EXAMPLE.COM

Valid starting     Expires            Service principal
04/17/20 20:29:50  04/18/20 06:29:50  krbtgt/EXAMPLE.COM@EXAMPLE.COM
        renew until 04/18/20 20:29:50
john@ldap-krb-client:~$
```

We logged in using the Kerberos password, and user/group information from the LDAP server.

### SSSD and KDC spoofing

When using SSSD to manage Kerberos logins on a Linux host, there is an attack scenario you should be aware of: KDC spoofing.

The objective of the attacker is to login on a workstation that is using Kerberos authentication. Let's say they know `john` is a valid user on that machine.

The attacker first deploys a rogue Key Distribution Center (KDC) server in the network, and creates the `john` principal there with a password of the attacker's choosing. What they must do now is have their rogue KDC respond to the login request from the workstation, before (or instead of) the real KDC. If the workstation isn't authenticating the KDC, it will accept the reply from the rogue server and let `john` in.

There is a configuration parameter that can be set to protect the workstation from this type of attack. It will have SSSD authenticate the KDC, and block the login if the KDC cannot be verified. This option is called `krb5_validate`, and it's `false` by default.

To enable it, edit `/etc/sssd/sssd.conf` and add this line to the domain section:

```
[sssd]
config_file_version = 2
domains = example.com

[domain/example.com]
id_provider = ldap
...
krb5_validate = True
```

The second step is to create a `host` principal on the KDC for this workstation. This is how the KDC's authenticity is verified. It's like a "machine account", with a shared secret that the attacker cannot control and replicate in the rogue KDC. The `host` principal has the format `host/<fqdn>@REALM`.

After the host principal is created, its keytab needs to be stored on the workstation. This two step process can be easily done on the workstation itself via `kadmin` (not `kadmin.local`) to contact the KDC remotely:

```
$ sudo kadmin -p ubuntu/admin
kadmin:  addprinc -randkey host/ldap-krb-client.example.com@EXAMPLE.COM
WARNING: no policy specified for host/ldap-krb-client.example.com@EXAMPLE.COM;
defaulting to no policy
Principal "host/ldap-krb-client.example.com@EXAMPLE.COM" created.

kadmin:  ktadd -k /etc/krb5.keytab host/ldap-krb-client.example.com
Entry for principal host/ldap-krb-client.example.com with kvno 6, encryption type
aes256-cts-hmac-sha1-96 added to keytab WRFILE:/etc/krb5.keytab.
Entry for principal host/ldap-krb-client.example.com with kvno 6, encryption type
aes128-cts-hmac-sha1-96 added to keytab WRFILE:/etc/krb5.keytab.
```

Then exit the tool and make sure the permissions on the keytab file are tight:

```
sudo chmod 0600 /etc/krb5.keytab
sudo chown root:root /etc/krb5.keytab
```

You can also do it on the KDC itself using `kadmin.local`, but you will have to store the keytab temporarily in another file and securely copy it over to the workstation.

Once these steps are complete, you can restart SSSD on the workstation and perform the login. If the rogue KDC notices the attempt and replies, it will fail the host verification. With debugging we can see this happening on the workstation:

```
==> /var/log/sssd/krb5_child.log <==
(Mon Apr 20 19:43:58 2020) [[sssd[krb5_child[2102]]]] [validate_tgt] (0x0020): TGT
failed verification using key for [host/ldap-krb-client.example.com@EXAMPLE.COM].
(Mon Apr 20 19:43:58 2020) [[sssd[krb5_child[2102]]]] [get_and_save_tgt] (0x0020):
1741: [-1765328377][Server host/ldap-krb-client.example.com@EXAMPLE.COM not found
in Kerberos database]
```

And the login is denied. If the real KDC picks it up, however, the host verification succeeds:

```
==> /var/log/sssd/krb5_child.log <==
(Mon Apr 20 19:46:22 2020) [[sssd[krb5_child[2268]]]] [validate_tgt] (0x0400): TGT
verified using key for [host/ldap-krb-client.example.com@EXAMPLE.COM].
```

And the login is accepted.

If you run into difficulties, refer to our troubleshooting guide.

### Troubleshooting SSSD

Here are some tips to help troubleshoot SSSD.

#### `debug_level`

The debug level of SSSD can be changed on-the-fly via `sssctl`, from the `sssd-tools` package:

```
sudo apt install sssd-tools
sssctl debug-level <new-level>
```

Or add it to the config file and restart SSSD:

```
[sssd]
config_file_version = 2
domains = example.com

[domain/example.com]
debug_level = 6
...
```

Either approach will yield more logs in `/var/log/sssd/*.log` and can help identify what is happening. The `sssctl` approach has the clear advantage of not having to restart the service.

### Caching

Caching is useful to speed things up, but it can get in the way big time when troubleshooting. It's useful to be able to remove the cache while chasing down a problem. This can also be done with the `sssctl` tool from the `sssd-tools` package.

You can either remove the whole cache:

```
# sssctl cache-remove
Creating backup of local data...
SSSD backup of local data already exists, override? (yes/no) [no] yes
Removing cache files...
SSSD= needs to be running. Start SSSD now? (yes/no) [yes] yes
```

Or just one element:

```
sssctl cache-expire -u john
```

Or expire everything:

```
sssctl cache-expire -E
```

## DNS

Kerberos is quite sensitive to *DNS* issues. If you suspect something related to DNS, here are two suggestions:

### FQDN hostname

Make sure `hostname -f` returns a *fully qualified domain name (FQDN)*. Set it in `/etc/hostname` if necessary, and use `sudo hostnamectl set-hostname <fqdn>` to set it at runtime.

### Reverse name lookup

You can try disabling a default reverse name lookup, which the `krb5` libraries do, by editing (or creating) `/etc/krb5.conf` and setting `rdns = false` in the `[libdefaults]` section:

```
[libdefaults]
rdns = false
```

### See also

- Explanation: *Introduction to network user authentication with SSSD*

### Smart card authentication

One of the most popular uses for smart cards is to control access to computer systems. The owner must physically *have* the smart card, and they must know the PIN to unlock it. This provides a higher degree of security than single-factor authentication (such as just using a password). In this page, we describe how to enable smart card authentication on Ubuntu.

> **Note**
>
> This guide is meant for Ubuntu Server 20.04 and newer. If you want to configure a desktop installation refer to the desktop guide.

### Software requirements

The following packages must be installed to obtain a smart card configuration on Ubuntu:

- `pcscd`: contains the drivers needed to communicate with the CCID smart card readers
- `opensc-pkcs11`: (optional, depending on your smartcard hardware) contains the smart card drivers, such as Personal Identify Verification (PIV) or Common Access Card (CAC)
- `sssd`: the authentication daemon that manages smart card access and certificate verification

To install these packages, run the following command in your terminal:

```
sudo apt install opensc-pkcs11 pcscd sssd libpam-sss
```

### Hardware requirements

Any PIV or CAC smart card with the corresponding reader should be sufficient. USB smart cards like Yubikey embed the reader, and work like regular PIV cards.

Each smart card is expected to contain an X.509 certificate and the corresponding private key to be used for authentication.

### Smart card PKCS#11 modules

While `opensc-pkcs11` supports a wide number of smart cards, some of them may require specific PKCS#11 modules, and you must refer to your vendor to install the proper one. From Ubuntu 20.04 onwards, all modules supported by `p11-kit` can be used.

If custom PKCS#11 modules are used, you need to ensure that `p11-kit` is properly configured.

In any case, `p11-kit` can be used to see all the configured modules that can be used for authentication:

```
$ p11-kit list-modules
```

(continues on next page)

(continued from previous page)

```
p11-kit-trust: p11-kit-trust.so
    library-description: PKCS#11 Kit Trust Module
    library-manufacturer: PKCS#11 Kit
    library-version: 0.23
    token: System Trust
        manufacturer: PKCS#11 Kit
        model: p11-kit-trust
        serial-number: 1
        hardware-version: 0.23
        flags:
               write-protected
               token-initialized
opensc-pkcs11: opensc-pkcs11.so
    library-description: OpenSC smartcard framework
    library-manufacturer: OpenSC Project
    library-version: 0.20
    token: MARCO TREVISAN (PIN CNS0)
        manufacturer: IC: STMicroelectronics; mask:...
        model: PKCS#15 emulated
        serial-number: 6090010669298009
        flags:
               login-required
               user-pin-initialized
               token-initialized
               user-pin-locked
```

### X.509 smart card certificates

The authentication is based on X.509 certificate validation and a smart card can provide one or more certificates that can be used for this purpose.

Before continuing, you may need to export or reference the certificate ID that must be used and associated to each user; such operations can be performed in one of the following three ways:

### Using p11tool

This is a more generic implementation that just uses the PKCS#11 protocol so it should work with all modules:

```
sudo apt install gnutls-bin
p11tool --list-tokens
```

Alternatively, URLs can be listed via:

```
p11tool --list-token-urls
```

For example:

```
Token 1:
        URL: pkcs11:model=PKCS%2315%20emulated;manufacturer=IC%3A%20Infineon%3B
%20mask%3A%20IDEMIA%20%28O...;serial=6090033068507002;token=MARCO%20TREVISAN%20
%28PIN%20CNS1%29
        Label: MARCO TREVISAN (PIN CNS1)
        Type: Hardware token
        Flags: Requires login
        Manufacturer: IC: Infineon; mask: IDEMIA (O...
        Model: PKCS#15 emulated
        Serial: 6090033068507002
        Module: opensc-pkcs11.so
```

The command above will show all the available smart cards in the system and their associated PKCS#11 URI. Copy the URI token of the selected card in the following command, which prints all certificates that can be used for authentication and their associated token URIs.

```
p11tool --list-all-certs 'pkcs11:token=[TOKEN-ID]'
```

So in the above example:

```
$ p11tool --list-all-certs 'pkcs11:token=MARCO%20TREVISAN%20%28PIN%20CNS1%29'
Object 0:
        URL: pkcs11:model=PKCS%2315%20emulated;manufacturer=IC%3A%20Infineon%3B
%20mask%3A%20IDEMIA%20%28O...;serial=6090033068507002;token=MARCO%20TREVISAN%20
%28PIN%20CNS1%29;id=%02;object=CNS1;type=cert
        Type: X.509 Certificate (RSA-2048)
        Expires: ven 17 dic 2027, 00:00:00
        Label: CNS1
        ID: 02
```

Now, once the URI of the certificate that will be used for authentication is known, let's extract the **Common Name** from the certificate. In the example we are assuming that our certificate URI is `pkcs11:id=%02;type=cert`.

It can be exported as text Privacy Enhanced Mail (PEM) format using:

```
$ p11tool --export 'pkcs11:id=%02;type=cert'
```

### Using opensc

```
$ sudo apt install opensc
```

Certificates can be via:

```
$ pkcs15-tool --list-certificates
```

And exported using

```
$ pkcs15-tool --read-certificate [CERTIFICATE_ID]
```

So, for example:

```
$ pkcs15-tool --list-certificates
Using reader with a card: Alcor Micro AU9560 00 00
X.509 Certificate [CNS1]
        Object Flags   : [0x00]
        Authority      : no
        Path           : 3f00140090012002
        ID             : 02
        Encoded serial : 02 10 0357B1EC0EB725BA67BD2D838DDF93D5
$ pkcs15-tool --read-certificate 2
Using reader with a card: Alcor Micro AU9560 00 00
-----BEGIN CERTIFICATE-----
MIIHXDCCBUSgAwIBAgIQA1ex7A6.....
```

### Troubleshooting

The card certificate verification can be simulated using openssl:

```
$ sudo apt install openssl

# Save the certificate, using one of the method stated above
$ pkcs15-tool --read-certificate 2 > card-cert.pem
$ p11tool --export 'pkcs11:id=%02;type=cert' > card-cert.pem

# See the certificate contents with
$ openssl x509 -text -noout -in card-cert.pem

# Verify it is valid for the given CA, where 'Ca-Auth-CERT.pem'
# contains all the certificates chain
$ openssl verify -verbose -CAfile CA-Auth-CERT.pem card-cert.pem

# If only the parent CA Certificate is available, can use -partial_chain:
$ openssl verify -verbose -partial_chain -CAfile intermediate_CA_cert.pem
```

### PAM configuration

To enable smart card authentication we should rely on a module that allows PAM supported systems to use X.509 certificates to authenticate logins. The module relies on a PKCS#11 library, such as `opensc-pkcs11` to access the smart card for the credentials it will need.

When a PAM smart card module is enabled, the login process is as follows:

1. Enter login
2. Enter PIN
3. Validate the X.509 certificate
4. Map the certificate to a user
5. Verify the login and match

To enable that process we have to configure the PAM module, add the relevant certificate authorities, add the PAM module to PAM configuration and set the mapping of certificate names to logins.

### Setup guide

This configuration uses SSSD as authentication mechanism, and the example shown here is showing a possible usage for local users, but more complex setups using external remote identity managers such as *FreeIPA*, LDAP, Kerberos or others can be used.

Refer to SSSD documentation to learn more about this.

### Enable SSSD PAM service

Pam service must be enabled in SSSD configuration, it can be done by ensuring that `/etc/sssd/sssd.conf` contains:

```
[sssd]
services = pam

[pam]
pam_cert_auth = True
```

Further `[pam]` configuration options can be changed according to the `sssd.conf(5)` manual page.

### Configure SSSD Certificate Authorities database

The card certificate must be allowed by a Certificate Authority, these should be part of `/etc/sssd/pki/sssd_auth_ca_db.pem` (or any other location configured in `[pam]` config section of `sssd.conf` as `pam_cert_db_path`).

As per SSSD using openssl, we need to add the whole certificates chain to the SSSD CA certificates path (if not changed via `sssd.certificate_verification`), so adding the certificates to the `pam_cert_db_path` is enough:

```
sudo cat Ca-Auth-CERT*.pem >> /etc/sssd/pki/sssd_auth_ca_db.pem
```

Certification Revocation List can be also defined in `sssd.conf`, providing a CRL file path in PEM format

```
[sssd]
crl_file = /etc/sssd/pki/sssd_auth_crl.pem
soft_crl = /etc/sssd/pki/sssd_auth_soft_crl.pem
```

In case that a full certificate authority chain is not available, openssl won't verify the card certificate, and so sssd should be instructed about.

This is not suggested, but it can be done changing `/etc/sssd/sssd.conf` so that it contains:

```
[sssd]
certificate_verification = partial_chain
```

### Troubleshooting

Card certificate verification can be simulated using SSSD tools directly, by using the command SSSD's `p11_child`:

```
# In ubuntu 20.04
$ sudo /usr/libexec/sssd/p11_child --pre -d 10 --debug-fd=2 --nssdb=/etc/sssd/pki/
sssd_auth_ca_db.pem

# In ubuntu 22.04 and later versions
$ sudo /usr/libexec/sssd/p11_child --pre -d 10 --debug-fd=2 --ca_db=/etc/sssd/pki/
sssd_auth_ca_db.pem
```

If certificate verification succeeds, the tool should output the card certificate name, its ID and the certificate itself in base64 format (other than debug data):

```
(Mon Sep 11 16:33:32:129558 2023) [p11_child[1965]] [do_card] (0x4000): Found
certificate has key id [02].
MARCO TREVISAN (PIN CNS1)
/usr/lib/x86_64-linux-gnu/pkcs11/opensc-pkcs11.so
02
CNS1
MIIHXDCCBUSgAwIBAgIQA1ex7....
```

For checking if the smartcard works, without doing any verification check (and so for debugging purposes the option) `--verify=no_ocsp` can also be used, while `--verify=partial_chain` can be used to do partial CA verification.

### Map certificates to user names

The sss PAM module allows certificates to be used for login, though our Linux system needs to know the username associated to a certificate. SSSD provides a variety of cert mappers to do this. Each cert mapper uses specific information from the certificate to map to a user on the system. The different cert mappers may even be stacked. In other words, if the first defined mapper fails to map to a user on the system, the next one will be tried, and so on until a user is found.

For the purposes of this guide, we will use a simple local user mapping as reference.

Mapping for more complex configurations can be done following the official SSSD documentation depending on providers. For up-to-date information on certificate mapping, please also consult the `sss-certmap(5)` manpage.

### Local users mapping

When using only local users, sssd can be easily configured to define an `implicit_domain` that maps all the local users.

Certificate mapping for local users can be easily done using the certificate Subject check, in our example:

```
openssl x509 -noout -subject -in card-cert.pem | sed "s/, /,/g;s/ = /=/g"
subject=C=IT,O=Actalis S.p.A.,OU=REGIONE TOSCANA,SN=TREVISAN,GN=MARCO,CN=TRVMRC[..
.data-removed...]/6090033068507002.UyMnHxfF3gkAeBYHhxa6V1Edazs=
```

So we can use for the user foo:

```
[sssd]
enable_files_domain = True
services = pam

[certmap/implicit_files/foo]
matchrule = <SUBJECT>.*CN=TRVMRC[A-Z0-9]+/6090033068507002\.
UyMnHxfF3gkAeBYHhxa6V1Edazs=.*

[pam]
pam_cert_auth = True
```

### Troubleshooting

User mapping can be tested working in versions newer than Ubuntu 20.04 with:

```
$ sudo dbus-send --system --print-reply \
    --dest=org.freedesktop.sssd.infopipe \
    /org/freedesktop/sssd/infopipe/Users \
    org.freedesktop.sssd.infopipe.Users.ListByCertificate \
    string:"$(cat card-cert.pem)" uint32:10
```

That should return the object path containing the expected user ID:

```
method return time=1605127192.698667 sender=:1.1628 -> destination=:1.1629
serial=6 reply_serial=2
   array [
      object path "/org/freedesktop/sssd/infopipe/Users/implicit_5ffiles/1000"
   ]
```

### Basic SSSD configuration

The SSSD configuration for accessing to the system is out of the scope of this document, however for smart card login it should contain at least such values:

```
[sssd]
# Comma separated list of domains
;domains = your-domain1, your-domain2

# comma-separated list of SSSD services
# pam might be implicitly loaded already, so the line is optional
services = pam

# You can enable debug of the SSSD daemon
# Logs will be in /var/log/sssd/sssd.log
;debug_level = 10

# A mapping between the SC certificate and users
;[certmap/your-domain1/<username>]
;matchrule = <SUBJECT>.*CN=<REGEX MATCHING YOUR CN>.*
```

(continues on next page)

(continued from previous page)

```
[pam]
pam_cert_auth = True

# The Certificate DB to be used:
# - Needs to be an openSSL CA certificates
;pam_cert_db_path = /etc/ssl/certs/ca-certificates.crt

# You can enable debug infos for the PAM module
# Logs will be in /var/log/sssd/sssd_pam.log
# p11 child logs are in /var/log/sssd/p11_child.log
# standard auth logs are in /var/log/auth.log
;pam_verbosity = 10
;debug_level = 10
```

In general what's in the configuration file will affect the way SSSD will call the `p11_child` tool (that is the one in charge for the actual authentication). Check `man sssd.conf` for details.

Remember that this file should be owned by `root` and have permission set to `600`, otherwise won't be loaded and SSSD will not complain gracefully. On errors you can test running SSSD temporary with `sudo sssd -d9 -i`.

Every time the configuration is changed sssd should be restarted (`systemctl restart sssd`).

### Add `pam_sss` to PAM

The next step includes the `pam_sss` module into the PAM stack. There are various ways to do this depending on your local policy. The following example enables smart card support for general authentication.

Edit `/etc/pam.d/common-auth` to include the `pam_sss` module as follows:

### For Ubuntu later than 23.10

```
$ sudo pam-auth-update
```

Then you can interactively enable SSSD profiles for smart-card only or optional smart card access.

You can also set this non-interactively by using:

```
# To use smart-card only authentication
$ sudo pam-auth-update --disable sss-smart-card-optional --enable sss-smart-card-
required

# To use smart-card authentication with fallback
$ sudo pam-auth-update --disable sss-smart-card-required --enable sss-smart-card-
optional
```

### For Ubuntu 23.10 and lower

```
# require SSSD smart card login
auth    [success=done default=die]    pam_sss.so allow_missing_name require_cert_
auth
```

or only try to use it:

```
# try SSSD smart card login
auth    [success=ok default=ignore]    pam_sss.so allow_missing_name try_cert_auth
```

See `man pam.conf`, `man pam_sss` for further details.

---

**Warning:** A global configuration such as this requires a smart card for su and sudo authentication as well! If you want to reduce the scope of this module, move it to the appropriate pam configuration file in `/etc/pam.d` and ensure that's referenced by `pam_p11_allowed_services` in `sssd.conf`.

---

The OS is now ready to do a smart card login for the user foo.

### Troubleshooting

`pamtester` is your friend!

To get better debug logging, also increase the SSSD verbosity by changing `/etc/sssd/sssd.conf` so that it has:

```
[pam]
pam_verbosity = 10
debug_level = 10
```

You can use it to check your configuration without having to login/logout for real, by just using:

```
# Install it!
$ sudo apt install pamtester

# Run the authentication service as standalone
$ pamtester -v login $USER authenticate

# Run the authentication service to get user from cert
$ pamtester -v login "" authenticate

# You can check what happened in the logs, reading:
sudo less /var/log/auth.log
sudo less /var/log/sssd/sssd_pam.log
sudo less /var/log/sssd/p11_child.log
```

### Smart card authentication with SSH

One of the authentication methods supported by the SSH protocol is public key authentication. A public key is copied to the SSH server where it is stored and marked as authorized. The owner of the corresponding private key in the smart card can then SSH login to the server.

We will use `opensc-pkcs11` on the client to access the smart card drivers, and we will copy the public key from the smart card to the SSH server to make the authentication work.

The following instructions apply to Ubuntu 18.04 later.

### Server configuration

The SSH server and client must be configured to permit smart card authentication.

### Configure the SSH server

The SSH server needs to allow public key authentication set in its configuration file and it needs the user's public key.

Ensure the server has the PubkeyAuthentication option set to 'yes' in its `/etc/ssh/sshd_config` file. In a default `/etc/ssh/sshd_config` in Ubuntu, the PubkeyAuthentication option is commented out. However, the default is 'yes'. To ensure the setting, edit the `sshd_config` file and set accordingly.

```
PubkeyAuthentication yes
```

### Restart the SSH server

```
sudo systemctl restart sshd
```

### Set the public key on the server

Extract the user's public key from the smart card on the SSH client. Use sshkeygen to read the public key from the smart card and into a format consumable for SSH.

```
ssh-keygen -D /usr/lib/x86_64-linux-gnu/opensc-pkcs11.so > smartcard.pub
```

Copy this key to the SSH server.

```
ssh-copy-id -f -i smartcard.pub ubuntu@server-2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "smartcard.pub"
ubuntu@server-2's password:
Number of key(s) added: 1
Now try logging into the machine, with: "ssh 'ubuntu@server-2'"
and check to make sure that only the key(s) you wanted were added.
```

### Client configuration

The SSH client needs to identify its PKCS#11 provider. To do that set the PKCS11Provider option in the `~/.ssh/config` file of each user desiring to use SSH smart card login.

```
PKCS11Provider /usr/lib/x86_64-linux-gnu/opensc-pkcs11.so
```

Use this method to enforce SSH smart card login on a per user basis.

After this step you can SSH into the server using the smart card for authentication.

### SSH authentication

See *this page on SSH authentication with smart cards*.

### Cryptography

The Secure Shell (SSH) cryptographic protocol that provides secure channels on an unsecured network. In Ubuntu, OpenSSH is the most commonly used implementation of SSH. It provides a suite of utilities for encrypting data transfers and can also be used for remote login and authentication.

### OpenSSH server

OpenSSH is a powerful collection of tools for remotely controlling networked computers and transferring data between them. Here we'll describe some of the configuration settings possible with the OpenSSH server application and how to change them on your Ubuntu system.

OpenSSH is a freely available version of the Secure Shell (SSH) protocol family of tools. Traditional tools, such as `telnet` or `rcp`, are insecure and transmit the user's password in cleartext when used. OpenSSH provides a server daemon and client tools to facilitate secure, encrypted, remote control and file transfer operations, effectively replacing the legacy tools.

The OpenSSH server component, `sshd`, listens continuously for client connections from any of the client tools. When a connection request occurs, `sshd` sets up the correct connection depending on the type of client tool connecting. For example, if the remote computer is connecting with the SSH client application, the OpenSSH server sets up a remote control session after authentication. If a remote user connects to an OpenSSH server with `scp`, the OpenSSH server daemon initiates a secure copy of files between the server and client after authentication.

OpenSSH can use many authentication methods, including plain password, public key cryptography, and Kerberos tickets.

### Install OpenSSH

To install the OpenSSH client applications on your Ubuntu system, use this command at a terminal prompt:

```
sudo apt install openssh-client
```

To install the OpenSSH server application, and related support files, use this command at a terminal prompt:

```
sudo apt install openssh-server
```

### Configure OpenSSH

To configure the default behavior of the OpenSSH server application, `sshd`, edit the file `/etc/ssh/sshd_config`. For information about the configuration directives used in this file, refer to the online `sshd_config(5)` manpage or run `man sshd_config` at a terminal prompt.

There are many directives in the `sshd` configuration file, which control things like communication settings and authentication modes. The following are examples of configuration directives that can be changed by editing the `/etc/ssh/sshd_config` file.

**Tip**

You can use *etckeeper* to track changes in your `/etc/` with `git`.

Alternatively, before editing the configuration file, you should make a copy of the original `/etc/ssh/sshd_config` file and protect it from writing so you will have the original settings as a reference and to reuse as necessary. You can do this with the following commands:

```
sudo cp /etc/ssh/sshd_config /etc/ssh/sshd_config.original
sudo chmod a-w /etc/ssh/sshd_config.original
```

Since losing an SSH server might mean losing your way to reach a server, check the configuration after changing it and before restarting the server:

```
sudo sshd -t -f /etc/ssh/sshd_config
```

### Example configuration directive

Let's take a look at an example of a configuration directive change. To make your OpenSSH server display the contents of the `/etc/issue.net` file as a pre-login banner, you can add or modify this line in the `/etc/ssh/sshd_config` file:

```
Banner /etc/issue.net
```

After making changes to the `/etc/ssh/sshd_config` file, save the file. Then, restart the `sshd` server application to effect the changes using the following command:

```
sudo systemctl restart ssh.service
```

**Warning**

Many other configuration directives for `sshd` are available to change the server application's behavior to fit your needs. Be advised, however, if your only method of access to a server is SSH, and you make a mistake when configuring `sshd` via the `/etc/ssh/sshd_config` file, you may find you are locked out of the server upon restarting it. Additionally, if an incorrect configuration directive is supplied, the `sshd` server may refuse to start, so be particularly careful when editing this file on a remote server.

### SSH keys

SSH allows authentication between two hosts without the need of a password, using cryptographic keys instead. *SSH-key* authentication uses a **private key** and a **public key**.

We recommend the **ed25519 Elliptic Curve algorithm** (`-t ed25519`) due to shorter keysize and lower computational requirements. Alternatively, you can create a key using the **RSA Algorithm** (`-t rsa -b 4096`, for 4096 bit keysize) instead.

To generate a key pair, run the following command:

```
ssh-keygen -t ed25519
```

During creation, you will be prompted for a key passphrase, which you would enter to use that key (every time, or instead using an `ssh-agent(1)`.

By default, the public key is saved in the file `~/.ssh/id_<algorithm>.pub`, while `~/.ssh/id_<algorithm>` is the private key. To allow login for a user via key, append the content of `id_ed25519.pub` (`id_rsa.pub` for RSA) to `target_machine:~/.ssh/authorized_keys` by running:

```
ssh-copy-id username@target_machine
```

Finally, double check the permissions on the `authorized_keys` file – only the authenticated user must have write permissions. If the permissions are not correct then change them by:

```
chmod go-w .ssh/authorized_keys
```

You should now be able to SSH to the `target_machine` without being prompted for a password.

To troubleshoot this, have a look at the `target_machine`'s live logs of `ssh.service`:

```
sudo journalctl -fu ssh.service
```

Since you can have multiple *SSH-key*s, you can configure which to use for which target machine in `~/.ssh/config` (see `ssh_config(5)`).

### Connection multiplexing

To reduce connection setup times, multiple SSH sessions to the same `target_machine` can reuse the same *TCP* connection.

Configuration to activate multiplexing in `~/.ssh/config`:

```
ControlMaster auto
ControlPath %d/.ssh/ssh_mux_%u@%l_%r@%h:%p
```

Optionally, if you want the session to still be re-usable `1` second after disconnecting, set:

```
ControlPersist 1
```

### Import keys from public keyservers

These days many users have already SSH keys registered with services like Launchpad or GitHub. Those can be imported with:

```
ssh-import-id <username-on-remote-service>
```

The prefix `lp:` is implied and means fetching from Launchpad. The alternative `gh:` will make the tool fetch from GitHub instead.

### Two factor authentication

You can add an extra layer of security to the default key-based authentication using two factor authentication. You can add two factor authentication *using U2F/FIDO hardware authentication devices*. Alternatively, in cases U2F/FIDO hardware authentication devices are unavailable or impractical for your use case you can add it *using HMAC/Time based One Time Passwords (HOTP/TOTP)*.

### Further reading

- Ubuntu Wiki SSH page.
- OpenSSH Website
- Advanced OpenSSH Wiki Page

### Two factor authentication with TOTP/HOTP

For the best two factor authentication (2FA) security, we recommend using hardware authentication devices that support U2F/FIDO. See the guide on *two factor authentication with U2F/FIDO* for details. However, if this is not possible or is impractical to implement in your case, TOTP/*HOTP* based 2FA is an improvement over no two factor at all. Smartphone apps to support this type of 2FA are common, such as Google Authenticator.

### Background

The configuration presented here makes public key authentication the first factor, the TOTP/HOTP code the second factor, and makes password authentication unavailable. Apart from the usual setup steps required for public key authentication, all configuration and setup takes place on the server. No changes are required at the client end; the 2FA prompt appears in place of the password prompt.

The two supported methods are HMAC-based One Time Password (HOTP) and Time-based One Time Password (TOTP). Generally, TOTP is preferable if the 2FA device supports it.

HOTP is based on a sequence predictable only to those who share a secret. The user must take an action to cause the client to generate the next code in the sequence, and this response is sent to the server. The server also generates the next code, and if it matches the one supplied by the user, then the user has proven to the server that they share the secret. A downside of this approach is that if the user generates codes without the server following along, such as in the case of a typo, then the sequence generators can fall “out of sync”. Servers compensate by allowing a gap in the sequence and considering a few subsequent codes to also be valid; if this mechanism is used, then the server “skips ahead” to sync back up. But to remain secure, this can only go so far before the server must refuse. When HOTP

falls out of sync like this, it must be reset using some out-of-band method, such as authenticating using a second backup key in order to reset the secret for the first one.

TOTP avoids this downside of HOTP by using the current timezone-independent date and time to determine the appropriate position in the sequence. However, this results in additional requirements and a different failure mode. Both devices must have the ability to tell the time, which is not practical for a USB 2FA token with no battery, for example. And both the server and client must agree on the correct time. If their clocks are skewed, then they will disagree on their current position in the sequence. Servers compensate for clock skew by allowing a few codes either side to also be valid. But like HOTP, they can only go so far before the server must refuse. One advantage of TOTP over HOTP is that correcting for this condition involves ensuring the clocks are correct at both ends; an out-of-band authentication to reset unfortunate users' secrets is not required. When using a modern smartphone app, for example, the requirement to keep the clock correct isn't usually a problem since this is typically done automatically at both ends by default.

**Note**

It is not recommended to configure U2F/FIDO at the same time as TOTP/HOTP. This combination has not been tested, and using the configuration presented here, TOTP/HOTP would become mandatory for everyone, whether or not they are also using U2F/FIDO.

### Install required software

From a terminal prompt, install the `google-authenticator` PAM module:

```
sudo apt update
sudo apt install libpam-google-authenticator
```

**Note**

The `libpam-google-authenticator` package is in Ubuntu's universe archive component, which receives best-effort community support only.

### Configure users

Since public key authentication with TOTP/HOTP 2FA is about to be configured to be mandatory for all users, each user who wishes to continue using SSH must first set up public key authentication and then configure their 2FA keys by running the user setup tool. If this isn't done first, users will not be able to do it later over SSH, since at that point they won't have public key authentication and/or 2FA configured to authenticate with.

### Configure users' key-based authentication

To set up key-based authentication, see "SSH Keys" above. Once this is done, it can be tested independently of subsequent 2FA configuration. At this stage, user authentication should work with keys only, requiring the supply of the private key passphrase only if it was configured. If configured correctly, the user should not be prompted for their password.

### Configure users' TOTP/HOTP 2FA secrets

Each user needs to run the setup tool to configure 2FA. This will ask some questions, generate a key, and display a QR code for the user to import the secret into their smartphone app, such as the Google Authenticator app on Android. The tool creates the file `~/.google-authenticator`, which contains a shared secret, emergency passcodes and per-user configuration.

As a user who needs 2FA configured, from a terminal prompt run the following command:

```
google-authenticator
```

Follow the prompts, scanning the QR code into your 2FA app as directed.

It's important to plan for the eventuality that the 2FA device gets lost or damaged. Will this lock the user out of their account? In mitigation, it's worthwhile for each user to consider doing one or more of the following:

- Use the 2FA device's backup or cloud sync facility if it has one.
- Write down the backup codes printed by the setup tool.
- Take a photo of the QR code.
- (TOTP only) Scan the QR code on multiple 2FA devices. This only works for TOTP, since multiple HOTP 2FA devices will not be able to stay in sync.
- Ensure that the user has a different authentication path to be able to rerun the setup tool if required.

Of course, any of these backup steps also negate any benefit of 2FA should someone else get access to the backup, so the steps taken to protect any backup should be considered carefully.

### Configure the SSH server

Once all users are configured, configure `sshd` itself by editing `/etc/ssh/sshd_config`. Depending on your installation, some of these settings may be configured already, but not necessarily with the values required for this configuration. Check for and adjust existing occurrences of these configuration directives, or add new ones, as required:

```
KbdInteractiveAuthentication yes
PasswordAuthentication no
AuthenticationMethods publickey,keyboard-interactive
```

> **Note**
>
> On Ubuntu 20.04 "Focal Fossa" and earlier, use `ChallengeResponseAuthentication yes` instead of `KbdInteractiveAUthentication yes`.

Restart the `ssh` service to pick up configuration changes:

```
sudo systemctl try-reload-or-restart ssh
```

Edit `/etc/pam.d/sshd` and replace the line:

```
@include common-auth
```

with:

```
auth required pam_google_authenticator.so
```

Changes to PAM configuration have immediate effect, and no separate reloading command is required.

### Log in using 2FA

Now when you log in using SSH, in addition to the normal public key authentication, you will be prompted for your TOTP or HOTP code:

```
$ ssh jammy.server
Enter passphrase for key 'id_rsa':
(ubuntu@jammy.server) Verification code:
Welcome to Ubuntu Jammy Jellyfish...
(...)
ubuntu@jammy.server:~$
```

### Special cases

On Ubuntu, the following settings are default in `/etc/ssh/sshd_config`, but if you have overridden them, note that they are required for this configuration to work correctly and must be restored as follows:

```
UsePAM yes
PubkeyAuthentication yes
```

Remember to run `sudo systemctl try-reload-or-restart ssh` for any changes made to `sshd` configuration to take effect.

### Further reading

- Wikipedia on TOTP
- Wikipedia on HOTP

### Two factor authentication with U2F/FIDO

OpenSSH 8.2 has added support for U2F/FIDO hardware authentication devices. These devices are used to provide an extra layer of security on top of the existing key-based authentication, as the hardware token needs to be present to finish the authentication.

It's very simple to use and setup. The only extra step is to generate a new keypair that can be used with the hardware device. For that, there are two key types that can be used: `ecdsa-sk` and `ed25519-sk`. The former has broader hardware support, while the latter might need a more recent device.

Once the keypair is generated, it can be used as you would normally use any other type of key in OpenSSH. The only requirement is that in order to use the private key, the U2F device has to be present on the host.

### Example with U2F

For example, plug the U2F device in and generate a keypair to use with it:

```
$ ssh-keygen -t ecdsa-sk
Generating public/private ecdsa-sk key pair.
You may need to touch your authenticator to authorize key generation. <-- touch
device
Enter file in which to save the key (/home/ubuntu/.ssh/id_ecdsa_sk):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_ecdsa_sk
Your public key has been saved in /home/ubuntu/.ssh/id_ecdsa_sk.pub
The key fingerprint is:
SHA256:V9PQ1MqaU8FODXdHqDiH9Mxb8XK3o5aVYDQLVl9IFRo ubuntu@focal
```

Now transfer the public part to the server to `~/.ssh/authorized_keys` and you are ready to go:

```
$ ssh -i .ssh/id_ecdsa_sk ubuntu@focal.server
Confirm user presence for key ECDSA-SK
SHA256:V9PQ1MqaU8FODXdHqDiH9Mxb8XK3o5aVYDQLVl9IFRo <-- touch device
Welcome to Ubuntu Focal Fossa (GNU/Linux 5.4.0-21-generic x86_64)
(...)
ubuntu@focal.server:~$
```

### FIDO2 resident keys

FIDO2 private keys consist of two parts: a **key handle** part, stored in the private key file on disk, and a **per-device key**, which is unique to each FIDO2 token and cannot be exported from the token hardware. These are combined by the hardware at authentication time to derive the real key, which is used to sign authentication challenges.

For tokens that are required to move between computers, it can be cumbersome to have to move the private key file first. To avoid this, tokens implementing the newer FIDO2 standard support **resident keys**, where it is possible to retrieve the key handle part of the key from the hardware.

Using resident keys increases the likelihood of an attacker being able to use a stolen token device. For this reason, tokens normally enforce PIN authentication before allowing the download of keys, and users should set a PIN on their tokens before creating any resident keys. This is done via the hardware token management software.

OpenSSH allows resident keys to be generated using the `ssh-keygen` flag `-O resident` at key generation time:

```
$ ssh-keygen -t ecdsa-sk -O resident -O application=ssh:mykeyname
Generating public/private ecdsa-sk key pair.
You may need to touch your authenticator to authorize key generation.
Enter PIN for authenticator:
Enter file in which to save the key (/home/ubuntu/.ssh/id_ecdsa_sk): mytoken
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

(continues on next page)

(continued from previous page)

```
Your identification has been saved in mytoken
(...)
```

This will produce a public/private key pair as usual, but it will be possible to retrieve the private key part (the key handle) from the token later. This is done by running:

```
$ ssh-keygen -K
Enter PIN for authenticator:
You may need to touch your authenticator to authorize key download.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Saved ECDSA-SK key ssh:mytoken to id_ecdsa_sk_rk_mytoken
```

It will use the part after `ssh:` from the `application` parameter from before as part of the key filenames:

```
$ l id_ecdsa_sk_rk_mytoken*
-rw------- 1 ubuntu ubuntu 598 out  4 18:49 id_ecdsa_sk_rk_mytoken
-rw-r--r-- 1 ubuntu ubuntu 228 out  4 18:49 id_ecdsa_sk_rk_mytoken.pub
```

If you set a passphrase when extracting the keys from the hardware token, and later use these keys, you will be prompted for both the key passphrase *and* the hardware key PIN. You will also have to touch the token:

```
$ ssh -i ./id_ecdsa_sk_rk_mytoken ubuntu@focal.server
Enter passphrase for key './id_ecdsa_sk_rk_mytoken':
Confirm user presence for key ECDSA-SK
SHA256:t+l26IgTXeURY6e36wtrq7wVYJtDVZrO+iuobs1CvVQ
User presence confirmed
(...)
```

It is also possible to download and add resident keys directly to `ssh-agent` by running

```
$ ssh-add -K
```

In this case, no file is written and the public key can be printed by running `ssh-add -L`.

**Note**

If you used the `-O verify-required` option when generating the keys, or if that option is set on the SSH server via the `/etc/ssh/sshd_config` setting `PubkeyAuthOptions verify-required`, then using the agent won't work (in Ubuntu 22.04 LTS).

### Further reading

- OpenSSH 8.2 release notes
- Yubikey documentation for OpenSSH FIDO/FIDO2 usage

### Install a root CA certificate in the trust store

Enterprise environments sometimes have a local Certificate Authority (CA) that issues certificates for use within the organisation. For an Ubuntu server to be functional, and to trust the hosts in this environment, this CA must be installed in Ubuntu's trust store.

### Certificate formats

There are two encoding formats for certificates:

- **Privacy Enhanced Mail (PEM)**: These are human-readable and in Base64-encoded **ASCII** format.
- **Distinguished Encoding Rules** (*DER*): These are encoded in a more compact **binary** format, and not human readable.

To install a certificate in the trust store it must be in PEM format. A PEM certificate starts with the line `----BEGIN CERTIFICATE----`. If you see this, you're ready to install. If not, it is probably a DER certificate and needs to be converted before you can install it in the trust store.

### Install a PEM-format certificate

Assuming your PEM-formatted root CA certificate is in `local-ca.crt`, run the following commands to install it:

- Generate a random certificate

Answer the questions asked after executing the command. When asked to input a `commonName` (`CN`), it should match the hostname of the server.

```
openssl req -x509 -new -nodes -keyout local-ca.key -out local-ca.crt
```

- Install the CA certificate package

```
sudo apt-get install -y ca-certificates
```

- Copy your certificate to the local CA certificates directory

```
sudo cp local-ca.crt /usr/local/share/ca-certificates
```

- Add the certificate to your trust store

```
sudo update-ca-certificates
```

- Verify that your certificate is in pem format

```
$ sudo ls /etc/ssl/certs/ | grep local-ca

local-ca.pem
```

- You can also verify that your certificate is available in the trust store by selecting a few texts from your certificate and comparing them with the certificate at the end of the trust store file to see if it matches yours.

```
$ sudo cat local-ca.crt

...
L4zOd3b41xJtYldofPve
-----END CERTIFICATE-----

$ sudo cat /etc/ssl/certs/ca-certificates.crt | grep L4zOd3b41xJtYldofPve

L4zOd3b41xJtYldofPve
```

**Note**

It is important that the certificate file has the `.crt` extension, otherwise it will not be processed.

After this point, you can use tools like `curl` and `wget` to connect to local sites.

### Uninstall a PEM-format certificate

- Verify that the certificate you'd like to uninstall exists.

```
$ sudo ls /usr/local/share/ca-certificates/local-ca.crt

/usr/local/share/ca-certificates/local-ca.crt
```

- Delete the certificate

```
sudo rm /usr/local/share/ca-certificates/local-ca.crt
```

- The certificate still exists in the trust store.

```
$ sudo cat /etc/ssl/certs/ca-certificates.crt | grep L4zOd3b41xJtYldofPve

L4zOd3b41xJtYldofPve
```

- Update the trust store

```
$ sudo update-ca-certificates --fresh
```

- Verify that the certificate has been uninstalled

```
$ sudo cat /etc/ssl/certs/ca-certificates.crt | grep L4zOd3b41xJtYldofPve
```

### Convert from DER to PEM format

You can convert a DER-formatted certificate called `local-ca.der` to PEM form like this:

```
sudo openssl x509 -inform der -outform pem -in local-ca.der -out local-ca.crt`
```

### The CA trust store location

The CA trust store (as generated by `update-ca-certificates`) is available at the following locations:

- As a single file (PEM bundle) in `/etc/ssl/certs/ca-certificates.crt`
- As an OpenSSL-compatible certificate directory in `/etc/ssl/certs`

### Virtual Private Network (VPN)

VPNs are commonly used to provide encrypted, secure access to a network. Two of the most popular choices in Ubuntu are OpenVPN and WireGuard VPN.

- *OpenVPN* is a well-established option that supports many platforms besides Linux
- *WireGuard VPN* is a modern and performant option that removes a lot of the complexity from configuring a VPN

### How to install and use OpenVPN

OpenVPN is a flexible, reliable and secure Virtual Private Networking (VPN) solution. It belongs to the family of SSL/TLS VPN stacks (different from IPSec VPNs). This chapter will show how to install and configure OpenVPN to create a VPN.

### Install the server

To install OpenVPN, run the following command in your terminal:

```
sudo apt install openvpn easy-rsa
```

### Set up the Public Key Infrastructure (PKI)

If you want more than just pre-shared keys, OpenVPN makes it easy to set up a Public Key Infrastructure (PKI) to use SSL/TLS certificates for authentication and key exchange between the VPN server and clients.

OpenVPN can be used in a routed or bridged VPN mode and can be configured to use either UDP or TCP. The port number can be configured as well, but port 1194 is the official one; this single port is used for all communication. *VPN client implementations* are available for almost anything including all Linux distributions, macOS, Windows and OpenWRT-based WLAN routers.

The first step in building an OpenVPN configuration is to establish a PKI, which consists of:

- A separate certificate (also known as a public key) and private key for the server and each client
- A primary Certificate Authority (CA) certificate and key, used to sign the server and client certificates

OpenVPN supports bi-directional authentication based on certificates, meaning that the client must authenticate the server certificate and the server must authenticate the client certificate before mutual trust is established.

Both the server and the client will authenticate each other by first verifying that the presented certificate was signed by the primary Certificate Authority (CA), and then by testing information in the now-authenticated certificate header, such as the certificate common name or certificate type (client or server).

### Set up the Certificate Authority

To set up your own CA, and generate certificates and keys for an OpenVPN server with multiple clients, first copy the `easy-rsa` directory to `/etc/openvpn`. This will ensure that any changes to the scripts will not be lost when the package is updated. From a terminal, run:

```
sudo make-cadir /etc/openvpn/easy-rsa
```

> **Note**
>
> You can alternatively edit `/etc/openvpn/easy-rsa/vars` directly, adjusting it to your needs.

As a `root` user, change to the newly created directory `/etc/openvpn/easy-rsa` and run:

```
./easyrsa init-pki
./easyrsa build-ca
```

The PEM passphrase set when creating the CA will be asked for every time you need to encrypt the output of a command (such as a private key). The encryption here is important, to avoid printing any private key in plain text.

### Create server keys and certificates

Next, we will generate a key pair for the server:

```
./easyrsa gen-req myservername nopass
```

Diffie Hellman parameters must be generated for the OpenVPN server. The following command will place them in `pki/dh.pem`:

```
./easyrsa gen-dh
```

And finally, create a certificate for the server:

```
./easyrsa sign-req server myservername
```

All certificates and keys have been generated in subdirectories. Common practice is to copy them to `/etc/openvpn/`:

```
cp pki/dh.pem pki/ca.crt pki/issued/myservername.crt pki/private/myservername.key
/etc/openvpn/
```

### Create client certificates

The VPN client will also need a certificate to authenticate itself to the server. Usually you create a different certificate for each client.

This can be done either on the server (as with the keys and certificates above) and then securely distributed to the client, or the client can generate and submit a request that is sent and signed by the server.

To create the certificate, enter the following in a terminal as a root user:

```
./easyrsa gen-req myclient1 nopass
./easyrsa sign-req client myclient1
```

If the first command above was done on a remote system, then copy the `.req` file to the CA server. From there, you can import it via `easyrsa import-req /incoming/myclient1.req myclient1`. Then you can go on with the second `sign-eq` command.

After this is done, in both cases you will need to copy the following files to the client using a secure method:

- `pki/ca.crt`
- `pki/issued/myclient1.crt`

Since the client certificates and keys are only required on the client machine, you can remove them from the server.

### Simple server configuration

Included with your OpenVPN installation are these (and many more) sample configuration files:

```
/usr/share/doc/openvpn/examples/sample-config-files/client.conf
/usr/share/doc/openvpn/examples/sample-config-files/server.conf
```

If these files under `/usr/share/doc/*` are not available (for example, the system was provisioned as a minimal install), you can fetch them directly from the Ubuntu package repositories:

- client.conf
- server.conf

Start by copying the example server configuration to `/etc/openvpn/server.conf`:

```
sudo cp /usr/share/doc/openvpn/examples/sample-config-files/server.conf /etc/
openvpn/myserver.conf
```

In Ubuntu 20.04 or older, the file is compressed, so do this instead:

```
sudo cp /usr/share/doc/openvpn/examples/sample-config-files/server.conf.gz /etc/
openvpn/myserver.conf.gz
sudo gzip -d /etc/openvpn/myserver.conf.gz
```

Edit `/etc/openvpn/myserver.conf` to make sure the following lines are pointing to the certificates and keys you created in the section above.

```
ca ca.crt
cert myservername.crt
key myservername.key
dh dh.pem
tls-auth ta.key 0
```

Complete this set with a TLS Authentication (TA) key in `/etc/openvpn` for `tls-auth` like this:

```
sudo openvpn --genkey secret ta.key
```

Edit `/etc/sysctl.conf` and uncomment the following line to enable IP forwarding:

```
#net.ipv4.ip_forward=1
```

Then reload `sysctl`:

```
sudo sysctl -p /etc/sysctl.conf
```

This is the minimum you need to configure to get a working OpenVPN server. You can use all the default settings in the sample `server.conf` file. Now you can start the server.

```
$ sudo systemctl start openvpn@myserver
```

**Note**

Be aware that the `systemctl start openvpn` is **not** starting the `openvpn` you just defined. OpenVPN uses templated `systemd` jobs, `openvpn@CONFIGFILENAME`. So if, for example, your configuration file is `myserver.conf` your service is called `openvpn@myserver`. You can run all kinds of service and `systemctl` commands like `start/stop/enable/disable/preset` against a templated service like `openvpn@server`.

You will find logging and error messages in the journal. For example, if you started a templated service `openvpn@server` you can filter for this particular message source with:

```
sudo journalctl -u openvpn@myserver -xe
```

The same templated approach works for all of `systemctl`:

```
$ sudo systemctl status openvpn@myserver
openvpn@myserver.service - OpenVPN connection to myserver
   Loaded: loaded (/lib/systemd/system/openvpn@.service; disabled; vendor preset:
enabled)
   Active: active (running) since Thu 2019-10-24 10:59:25 UTC; 10s ago
     Docs: man:openvpn(8)
           https://community.openvpn.net/openvpn/wiki/Openvpn24ManPage
           https://community.openvpn.net/openvpn/wiki/HOWTO
 Main PID: 4138 (openvpn)
   Status: "Initialization Sequence Completed"
    Tasks: 1 (limit: 533)
   Memory: 1.0M
```

(continues on next page)

(continued from previous page)

```
   CGroup: /system.slice/system-openvpn.slice/openvpn@myserver.service
           └─4138 /usr/sbin/openvpn --daemon ovpn-myserver --status /run/openvpn/
myserver.status 10 --cd /etc/openvpn --script-security 2 --config /etc/openvpn/
myserver.conf --writepid /run/

Oct 24 10:59:26 eoan-vpn-server ovpn-myserver[4138]: /sbin/ip addr add dev tun0
local 10.8.0.1 peer 10.8.0.2
Oct 24 10:59:26 eoan-vpn-server ovpn-myserver[4138]: /sbin/ip route add 10.8.0.0/
24 via 10.8.0.2
Oct 24 10:59:26 eoan-vpn-server ovpn-myserver[4138]: Could not determine IPv4/IPv6
protocol. Using AF_INET
Oct 24 10:59:26 eoan-vpn-server ovpn-myserver[4138]: Socket Buffers: R=[212992->
212992] S=[212992->212992]
Oct 24 10:59:26 eoan-vpn-server ovpn-myserver[4138]: UDPv4 link local (bound):
[AF_INET][undef]:1194
Oct 24 10:59:26 eoan-vpn-server ovpn-myserver[4138]: UDPv4 link remote: [AF_
UNSPEC]
Oct 24 10:59:26 eoan-vpn-server ovpn-myserver[4138]: MULTI: multi_init called,
r=256 v=256
Oct 24 10:59:26 eoan-vpn-server ovpn-myserver[4138]: IFCONFIG POOL: base=10.8.0.4
size=62, ipv6=0
Oct 24 10:59:26 eoan-vpn-server ovpn-myserver[4138]: IFCONFIG POOL LIST
Oct 24 10:59:26 eoan-vpn-server ovpn-myserver[4138]: Initialization Sequence
Completed
```

You can enable/disable various OpenVPN services on one system, but you could also let Ubuntu do it for you. There is a config for `AUTOSTART` in `/etc/default/openvpn`. Allowed values are "all", "none" or a space-separated list of names of the VPNs. If empty, "all" is assumed. The VPN name refers to the VPN configuration file name, i.e., `home` would be `/etc/openvpn/home.conf`.

If you're running `systemd`, changing this variable requires running `systemctl daemon-reload` followed by a restart of the `openvpn` service (if you removed entries you may have to stop those manually).

After `systemctl daemon-reload`, a restart of the "generic" OpenVPN will restart all dependent services that the generator in `/lib/systemd/system-generators/openvpn-generator` created for your `conf` files when you called `daemon-reload`.

Now, check if OpenVPN created a `tun0` interface:

```
root@server:/etc/openvpn# ip addr show dev tun0
5: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
UNKNOWN group default qlen 100
    link/none
    inet 10.8.0.1 peer 10.8.0.2/32 scope global tun0
       valid_lft forever preferred_lft forever
    inet6 fe80::b5ac:7829:f31e:32c5/64 scope link stable-privacy
       valid_lft forever preferred_lft forever
```

### Simple client configuration

There are various different OpenVPN client implementations – both with and without *GUIs*. You can read more about clients in *our page on OpenVPN Clients*. For now, we use the command-line/service-based OpenVPN client for Ubuntu, which is part of the same package as the server. So you must install the `openvpn` package again on the client machine:

```
sudo apt install openvpn
```

This time, copy the `client.conf` sample config file to `/etc/openvpn/`:

```
sudo cp /usr/share/doc/openvpn/examples/sample-config-files/client.conf /etc/
openvpn/
```

Copy the following client keys and certificate files you created in the section above to e.g. `/etc/openvpn/` and edit `/etc/openvpn/client.conf` to make sure the following lines are pointing to those files. If you have the files in `/etc/openvpn/` you can omit the path:

```
ca ca.crt
cert myclient1.crt
key myclient1.key
tls-auth ta.key 1
```

And you have to specify the OpenVPN server name or address. Make sure the keyword `client` is in the config file, since that's what enables client mode.

```
client
remote vpnserver.example.com 1194
```

Now start the OpenVPN client with the same templated mechanism:

```
$ sudo systemctl start openvpn@client
```

You can check the status as you did on the server:

```
$ sudo systemctl status openvpn@client
openvpn@client.service - OpenVPN connection to client
   Loaded: loaded (/lib/systemd/system/openvpn@.service; disabled; vendor preset:
enabled)
   Active: active (running) since Thu 2019-10-24 11:42:35 UTC; 6s ago
     Docs: man:openvpn(8)
           https://community.openvpn.net/openvpn/wiki/Openvpn24ManPage
           https://community.openvpn.net/openvpn/wiki/HOWTO
 Main PID: 3616 (openvpn)
   Status: "Initialization Sequence Completed"
    Tasks: 1 (limit: 533)
   Memory: 1.3M
   CGroup: /system.slice/system-openvpn.slice/openvpn@client.service
           └─3616 /usr/sbin/openvpn --daemon ovpn-client --status /run/openvpn/
client.status 10 --cd /etc/openvpn --script-security 2 --config /etc/openvpn/
client.conf --writepid /run/openvp
```

(continues on next page)

(continued from previous page)

```
Oct 24 11:42:36 eoan-vpn-client ovpn-client[3616]: Outgoing Data Channel: Cipher
'AES-256-GCM' initialized with 256 bit key
Oct 24 11:42:36 eoan-vpn-client ovpn-client[3616]: Incoming Data Channel: Cipher
'AES-256-GCM' initialized with 256 bit key
Oct 24 11:42:36 eoan-vpn-client ovpn-client[3616]: ROUTE_GATEWAY 192.168.122.1/
255.255.255.0 IFACE=ens3 HWADDR=52:54:00:3c:5a:88
Oct 24 11:42:36 eoan-vpn-client ovpn-client[3616]: TUN/TAP device tun0 opened
Oct 24 11:42:36 eoan-vpn-client ovpn-client[3616]: TUN/TAP TX queue length set to
100
Oct 24 11:42:36 eoan-vpn-client ovpn-client[3616]: /sbin/ip link set dev tun0 up
mtu 1500
Oct 24 11:42:36 eoan-vpn-client ovpn-client[3616]: /sbin/ip addr add dev tun0
local 10.8.0.6 peer 10.8.0.5
Oct 24 11:42:36 eoan-vpn-client ovpn-client[3616]: /sbin/ip route add 10.8.0.1/32
via 10.8.0.5
Oct 24 11:42:36 eoan-vpn-client ovpn-client[3616]: WARNING: this configuration may
cache passwords in memory -- use the auth-nocache option to prevent this
Oct 24 11:42:36 eoan-vpn-client ovpn-client[3616]: Initialization Sequence
Completed
```

On the server log, an incoming connection looks like the following (you can see client name and source address as well as success/failure messages):

```
ovpn-myserver[4818]: 192.168.122.114:55738 TLS: Initial packet from [AF_INET]192.
168.122.114:55738, sid=5e943ab8 40ab9fed
ovpn-myserver[4818]: 192.168.122.114:55738 VERIFY OK: depth=1, CN=Easy-RSA CA
ovpn-myserver[4818]: 192.168.122.114:55738 VERIFY OK: depth=0, CN=myclient1
ovpn-myserver[4818]: 192.168.122.114:55738 peer info: IV_VER=2.4.7
ovpn-myserver[4818]: 192.168.122.114:55738 peer info: IV_PLAT=linux
ovpn-myserver[4818]: 192.168.122.114:55738 peer info: IV_PROTO=2
ovpn-myserver[4818]: 192.168.122.114:55738 peer info: IV_NCP=2
ovpn-myserver[4818]: 192.168.122.114:55738 peer info: IV_LZ4=1
ovpn-myserver[4818]: 192.168.122.114:55738 peer info: IV_LZ4v2=1
ovpn-myserver[4818]: 192.168.122.114:55738 peer info: IV_LZO=1
ovpn-myserver[4818]: 192.168.122.114:55738 peer info: IV_COMP_STUB=1
ovpn-myserver[4818]: 192.168.122.114:55738 peer info: IV_COMP_STUBv2=1
ovpn-myserver[4818]: 192.168.122.114:55738 peer info: IV_TCPNL=1
ovpn-myserver[4818]: 192.168.122.114:55738 Control Channel: TLSv1.3, cipher TLSv1.
3 TLS_AES_256_GCM_SHA384, 2048 bit RSA
ovpn-myserver[4818]: 192.168.122.114:55738 [myclient1] Peer Connection Initiated
with [AF_INET]192.168.122.114:55738
ovpn-myserver[4818]: myclient1/192.168.122.114:55738 MULTI_sva: pool returned
IPv4=10.8.0.6, IPv6=(Not enabled)
ovpn-myserver[4818]: myclient1/192.168.122.114:55738 MULTI: Learn: 10.8.0.6 ->
myclient1/192.168.122.114:55738
ovpn-myserver[4818]: myclient1/192.168.122.114:55738 MULTI: primary virtual IP for
myclient1/192.168.122.114:55738: 10.8.0.6
ovpn-myserver[4818]: myclient1/192.168.122.114:55738 PUSH: Received control
message: 'PUSH_REQUEST'
```

(continues on next page)

(continued from previous page)

```
ovpn-myserver[4818]: myclient1/192.168.122.114:55738 SENT CONTROL [myclient1]:
'PUSH_REPLY,route 10.8.0.1,topology net30,ping 10,ping-restart 120,ifconfig 10.8.
0.6 10.8.0.5,peer-id 0,cipher AES-256-GCM' (status=1)
ovpn-myserver[4818]: myclient1/192.168.122.114:55738 Data Channel: using
negotiated cipher 'AES-256-GCM'
ovpn-myserver[4818]: myclient1/192.168.122.114:55738 Outgoing Data Channel: Cipher
'AES-256-GCM' initialized with 256 bit key
ovpn-myserver[4818]: myclient1/192.168.122.114:55738 Incoming Data Channel: Cipher
'AES-256-GCM' initialized with 256 bit key
```

And you can check on the client if it created a `tun0` interface:

```
$ ip addr show dev tun0
4: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
UNKNOWN group default qlen 100
    link/none
    inet 10.8.0.6 peer 10.8.0.5/32 scope global tun0
       valid_lft forever preferred_lft forever
    inet6 fe80::5a94:ae12:8901:5a75/64 scope link stable-privacy
       valid_lft forever preferred_lft forever
```

Check if you can ping the OpenVPN server:

```
root@client:/etc/openvpn# ping 10.8.0.1
PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data.
64 bytes from 10.8.0.1: icmp_req=1 ttl=64 time=0.920 ms
```

**Note**

The OpenVPN server always uses the first usable IP address in the client network and only that IP is pingable. E.g., if you configured a `/24` for the client network mask, the `.1` address will be used. The P-t-P address you see in the `ip addr` output above does not usually answer ping requests.

Check out your routes:

```
$ ip route
default via 192.168.122.1 dev ens3 proto dhcp src 192.168.122.114 metric 100
10.8.0.1 via 10.8.0.5 dev tun0
10.8.0.5 dev tun0 proto kernel scope link src 10.8.0.6
192.168.122.0/24 dev ens3 proto kernel scope link src 192.168.122.114
192.168.122.1 dev ens3 proto dhcp scope link src 192.168.122.114 metric 100
```

### First troubleshooting

If the above didn't work for you, check the following:

- Check your `journal -xe`.
- Check that you have specified the key filenames correctly in the client and server `conf`

files.

- Can the client connect to the server machine? Maybe a firewall is blocking access? Check the journal on the server.
- Client and server must use same protocol and port, e.g. UDP port 1194, see `port` and `proto` config options.
- Client and server must use the same compression configuration, see `comp-lzo` config option.
- Client and server must use same config regarding bridged vs. routed mode, see `server vs server-bridge` config option
- Client must use the config `tls-auth` with index `1` (example client config: `tls-auth ta.key 1`), but server must use `tls-auth` with index `0` (example server config: `tls-auth ta.key 0`).

### Advanced configuration

#### Advanced routed VPN configuration on server

The above is a very simple working VPN. The client can access services on the VPN server machine through an encrypted tunnel. If you want to reach more servers or anything in other networks, push some routes to the clients. E.g. if your company's network can be summarised to the network `192.168.0.0/16`, you could push this route to the clients. But you will also have to change the routing for the way back – your servers need to know a route to the VPN client-network.

The example config files that we have been using in this guide are full of these advanced options in the form of a comment and a disabled configuration line as an example.

> **Note**
>
> Read the OpenVPN hardening security guide for further security advice.

#### Advanced bridged VPN configuration on server

OpenVPN can be set up for either a routed or a bridged VPN mode. Sometimes this is also referred to as OSI layer-2 versus layer-3 VPN. In a bridged VPN all layer-2 frames – e.g. all Ethernet frames – are sent to the VPN partners and in a routed VPN only layer-3 packets are sent to VPN partners. In bridged mode, all traffic including traffic which was traditionally LAN-local (like local network broadcasts, *DHCP* requests, ARP requests etc) are sent to VPN partners, whereas in routed mode this would be filtered.

#### Prepare interface config for bridging on server

First, use Netplan to configure a bridge device using the desired Ethernet device:

```
$ cat /etc/netplan/01-netcfg.yaml
network:
    version: 2
    renderer: networkd
```

(continues on next page)

(continued from previous page)

```
    ethernets:
        enp0s31f6:
            dhcp4: no
    bridges:
        br0:
            interfaces: [enp0s31f6]
            dhcp4: no
            addresses: [10.0.1.100/24]
            gateway4: 10.0.1.1
            nameservers:
                addresses: [10.0.1.1]
```

Static IP addressing is highly suggested. DHCP addressing can also work, but you will still have to encode a static address in the OpenVPN configuration file.

The next step on the server is to configure the Ethernet device for promiscuous mode on boot. To do this, ensure the `networkd-dispatcher` package is installed and create the following configuration script:

```
sudo apt update
sudo apt install networkd-dispatcher
sudo touch /usr/lib/networkd-dispatcher/dormant.d/promisc_bridge
sudo chmod +x /usr/lib/networkd-dispatcher/dormant.d/promisc_bridge
```

Then add the following contents:

```
#!/bin/sh
set -e
if [ "$IFACE" = br0 ]; then
    # no networkd-dispatcher event for 'carrier' on the physical interface
    ip link set enp0s31f6 up promisc on
fi
```

### Prepare server config for bridging

Edit `/etc/openvpn/server.conf` to use `tap` rather than `tun` and set the server to use the `server-bridge` directive:

```
;dev tun
dev tap
;server 10.8.0.0 255.255.255.0
server-bridge 10.0.0.4 255.255.255.0 10.0.0.128 10.0.0.254
```

After configuring the server, restart OpenVPN by entering:

```
sudo systemctl restart openvpn@myserver
```

### Prepare client config for bridging

The only difference on the client side for bridged mode to what was outlined above is that you need to edit `/etc/openvpn/client.conf` and set `tap` mode:

```
dev tap
;dev tun
```

Finally, restart OpenVPN:

```
sudo systemctl restart openvpn@client
```

You should now be able to connect to the fully remote LAN through the VPN.

### Using OpenSSL providers

OpenVPN uses the OpenSSL 3 Default Provider on Ubuntu. However, you can include additional providers dynamically depending on your use case.

### Legacy provider

In Ubuntu 24.04 LTS and later, you can still use legacy algorithms by explicitly adding them through the `legacy` provider alongside `default`. To do this, add the following line to your configuration.

```
providers legacy default
```

You can also run openvpn with the `--providers` argument.

```
openvpn --providers legacy default ...
```

**Note**

On Ubuntu 22.04 LTS, legacy algorithms are included in OpenVPN by default.

### TPM 2.0

OpenVPN also works with Trusted Platform Module (TPM) 2.0 encryption using the `tpm2` provider. Set it up by installing the `tpm2-openssl` package and including both `legacy` and `tpm2` in your provider configuration.

```
providers legacy default tpm2
```

Alternatively, run with:

```
openvpn --providers legacy default tpm2 ...
```

**Note**

The provider order matters here as tpm2 currently requires the legacy provider to load first.

### Further reading

- EasyRSA
- OpenVPN quick start guide
- Snap'ed version of OpenVPN easy-openvpn
- Debian's OpenVPN Guide

## WireGuard VPN

WireGuard is a straightforward, modern, cross-platform VPN implementation. It is usually used to either connect a single system to a remote site ("peer-to-site") or to connect two distinct networks over the internet ("site-to-site").

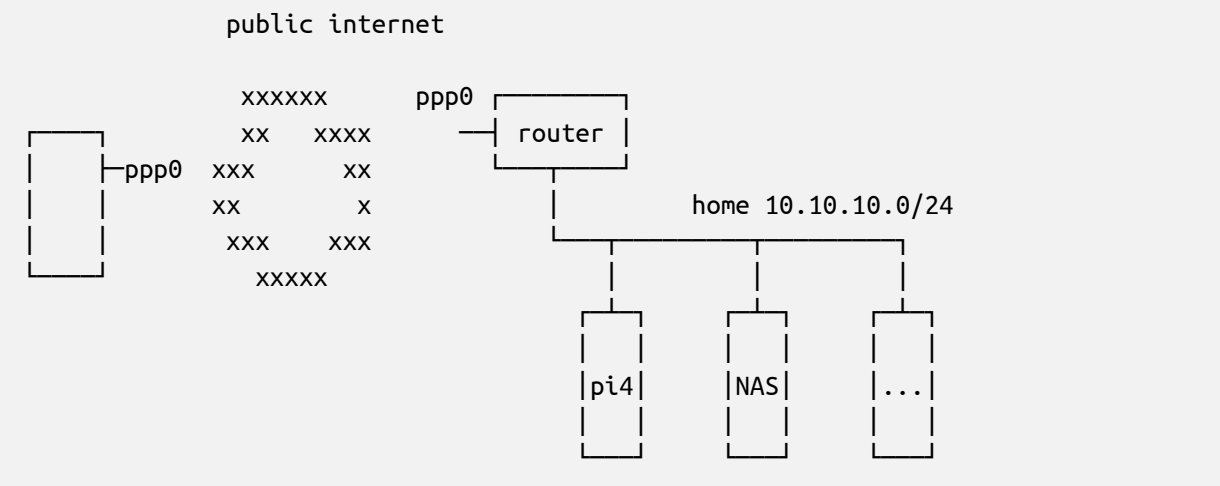
### WireGuard VPN peer-to-site

To help understand the WireGuard concepts, we will show some practical setups that hopefully match many scenarios out there.

This is probably the most common setup for a VPN: connecting a single system to a remote site, and getting access to the remote network "as if you were there".

Where to place the remote WireGuard endpoint in the network will vary a lot depending on the topology. It can be in a firewall box, the router itself, or some random system in the middle of the network.

Here we will cover a simpler case more resembling what a home network could be like:



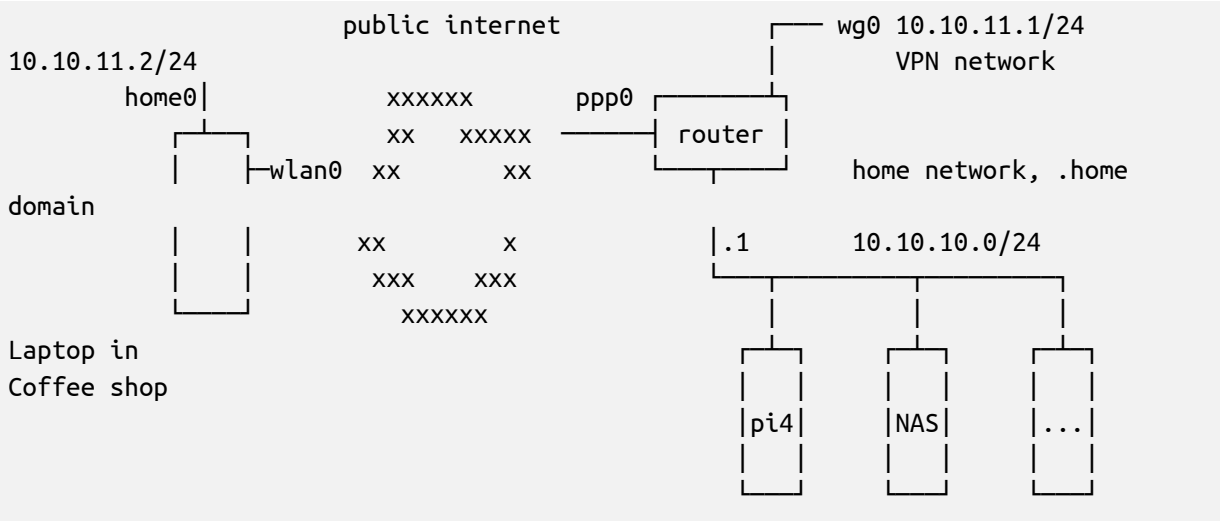

This diagram represents a typical simple home network setup. You have a router/modem, usually provided by the ISP (Internet Service Provider), and some internal devices like a Raspberry PI perhaps, a NAS (Network Attached Storage), and some other device.

There are basically two approaches that can be taken here: install WireGuard *on the router*, or on *another system in the home network*.

Note that in this scenario the "fixed" side, the home network, normally won't have a WireGuard `Endpoint` configured, as the peer is typically "on the road" and will have a dynamic IP address.

### WireGuard VPN peer-to-site (on router)

In this diagram, we are depicting a home network with some devices and a router where we can install WireGuard.




Of course, this setup is only possible if you can install software on the router. Most of the time, when it's provided by your ISP, you can't. But some ISPs allow their device to be put into a bridge mode, in which case you can use your own device (a computer, a Raspberry PI, or something else) as the routing device.

Since the router is the default gateway of the network already, this means you can create a whole new network for your VPN users. You also won't have to create any (D)NAT rules since the router is directly reachable from the Internet.

Let's define some addresses, networks, and terms used in this guide:

- **laptop in coffee shop**: just your normal user at a coffee shop, using the provided Wi-Fi access to connect to their home network. This will be one of our **peers** in the VPN setup.
- `home0`: this will be the WireGuard interface on the laptop. It's called `home0` to convey that it is used to connect to the **home** network.
- **router**: the existing router at the home network. It has a public interface `ppp0` that has a routable but dynamic IPv4 address (not CGNAT), and an internal interface at `10.10.10.1/24` which is the default gateway for the home network.
- **home network**: the existing home network (`10.10.10.0/24` in this example), with existing devices that the user wishes to access remotely over the WireGuard VPN.
- `10.10.11.0/24`: the WireGuard VPN network. This is a whole new network that was created just for the VPN users.
- `wg0` on the **router**: this is the WireGuard interface that we will bring up on the router, at the `10.10.11.1/24` address. It is the gateway for the `10.10.11.0/24` VPN network.

With this topology, if, say, the NAS wants to send traffic to `10.10.11.2/24`, it will send it to the default gateway (since the NAS has no specific route to `10.10.11.0/24`), and the gateway will know how to send it to `10.10.11.2/24` because it has the `wg0` interface on that network.

### Configuration

First, we need to create keys for the peers of this setup. We need one pair of keys for the laptop, and another for the home router:

```
$ umask 077
$ wg genkey > laptop-private.key
$ wg pubkey < laptop-private.key > laptop-public.key
$ wg genkey > router-private.key
$ wg pubkey < router-private.key > router-public.key
```

Let's create the router `wg0` interface configuration file. The file will be `/etc/wireguard/wg0.conf` and have these contents:

```
[Interface]
PrivateKey = <contents-of-router-private.key>
ListenPort = 51000
Address = 10.10.11.1/24

[Peer]
PublicKey = <contents-of-laptop-public.key>
AllowedIPs = 10.10.11.2
```

There is no `Endpoint` configured for the laptop peer, because we don't know what IP address it will have beforehand, nor will that IP address be always the same. This laptop could be connecting from a coffee shop's free Wi-Fi, an airport lounge, or a friend's house.

Not having an endpoint here also means that the home network side will never be able to *initiate* the VPN connection. It will sit and wait, and can only *respond* to VPN handshake requests, at which time it will learn the endpoint from the peer and use that until it changes (i.e. when the peer reconnects from a different site) or it times out.

**Important**

This configuration file contains a secret: **PrivateKey**. Make sure to adjust its permissions accordingly, as follows:

```
sudo chmod 0600 /etc/wireguard/wg0.conf
sudo chown root: /etc/wireguard/wg0.conf
```

When activated, this will bring up a `wg0` interface with the address `10.10.11.1/24`, listening on port `51000/udp`, and add a route for the `10.10.11.0/24` network using that interface.

The `[Peer]` section is identifying a peer via its public key, and listing who can connect from that peer. This `AllowedIPs` setting has two meanings:

- When sending packets, the `AllowedIPs` list serves as a routing table, indicating that this peer's public key should be used to encrypt the traffic.
- When receiving packets, `AllowedIPs` behaves like an access control list. After decryption, the traffic is only allowed if it matches the list.

Finally, the `ListenPort` parameter specifies the **UDP** port on which WireGuard will listen for traffic. This port will have to be allowed in the firewall rules of the router. There is neither a

default nor a standard port for WireGuard, so you can pick any value you prefer.

Now let's create a similar configuration on the other peer, the laptop. Here the interface is called `home0`, so the configuration file is `/etc/wireguard/home0.conf`:

```
[Interface]
PrivateKey = <contents-of-laptop-private.key>
ListenPort = 51000
Address = 10.10.11.2/24

[Peer]
PublicKey = <contents-of-router-public.key>
Endpoint = <home-ppp0-IP-or-hostname>:51000
AllowedIPs = 10.10.11.0/24,10.10.10.0/24
```

**Important**

As before, this configuration file contains a secret: **PrivateKey**. You need to adjust its permissions accordingly, as follows:

```
sudo chmod 0600 /etc/wireguard/home0.conf
sudo chown root: /etc/wireguard/home0.conf
```

We have given this laptop the `10.10.11.2/24` address. It could have been any valid address in the `10.10.11.0/24` network, as long as it doesn't collide with an existing one, and is allowed in the router's peer's `AllowedIPs` list.

**Note**

You may have noticed by now that address allocation is manual, and not via something like *DHCP*. Keep tabs on it!

In the `[Peer]` stanza for the laptop we have:

- The usual `PublicKey` item, which identifies the peer. Traffic to this peer will be encrypted using this public key.
- `Endpoint`: this tells WireGuard where to actually send the encrypted traffic to. Since in our scenario the laptop will be initiating connections, it has to know the public IP address of the home router. If your ISP gave you a fixed IP address, great! You have nothing else to do. If, however, you have a dynamic IP address (one that changes every time you establish a new connection), then you will have to set up some sort of dynamic *DNS* service. There are many such services available for free on the Internet, but setting one up is out of scope for this guide.
- In `AllowedIPs` we list our destinations. The VPN network `10.10.11.0/24` is listed so that we can ping `wg0` on the home router as well as other devices on the same VPN, and the actual home network, which is `10.10.10.0/24`.

If we had used `0.0.0.0/0` alone in `AllowedIPs`, then the VPN would become our default gateway, and all traffic would be sent to this peer. See *Default Gateway* for details on that type of setup.

### Testing

With these configuration files in place, it's time to bring the WireGuard interfaces up.

On the home router, run:

```
$ sudo wg-quick up wg0
[#] ip link add wg0 type wireguard
[#] wg setconf wg0 /dev/fd/63
[#] ip -4 address add 10.10.11.1/24 dev wg0
[#] ip link set mtu 1378 up dev wg0
```

Verify you have a `wg0` interface up with an address of `10.10.11.1/24`:

```
$ ip a show dev wg0
9: wg0: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1378 qdisc noqueue state UNKNOWN group
default qlen 1000
    link/none
    inet 10.10.11.1/24 scope global wg0
       valid_lft forever preferred_lft forever
```

And a route to the `10.10.1.0/24` network via the `wg0` interface:

```
$ ip route | grep wg0
10.10.11.0/24 dev wg0 proto kernel scope link src 10.10.11.1
```

And `wg show` should show some status information, but no connected peer yet:

```
$ sudo wg show
interface: wg0
  public key: <router public key>
  private key: (hidden)
  listening port: 51000

peer: <laptop public key>
  allowed ips: 10.10.11.2/32
```

In particular, verify that the listed public keys match what you created (and expected!).

Before we start the interface on the other peer, it helps to leave the above `show` command running continuously, so we can see when there are changes:

```
$ sudo watch wg show
```

Now start the interface on the laptop:

```
$ sudo wg-quick up home0
[#] ip link add home0 type wireguard
[#] wg setconf home0 /dev/fd/63
[#] ip -4 address add 10.10.11.2/24 dev home0
[#] ip link set mtu 1420 up dev home0
[#] ip -4 route add 10.10.10.0/24 dev home0
```

Similarly, verify the interface's IP and added routes:

```
$ ip a show dev home0
24: home0: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1420 qdisc noqueue state UNKNOWN
group default qlen 1000
    link/none
    inet 10.10.11.2/24 scope global home0
       valid_lft forever preferred_lft forever

$ ip route | grep home0
10.10.10.0/24 dev home0 scope link
10.10.11.0/24 dev home0 proto kernel scope link src 10.10.11.2
```

Up to this point, the `wg show` output on the home router probably didn't change. That's because we haven't sent any traffic to the home network, which didn't trigger the VPN yet. By default, WireGuard is very "quiet" on the network.

If we trigger some traffic, however, the VPN will "wake up". Let's ping the internal address of the home router a few times:

```
$ ping -c 3 10.10.10.1
PING 10.10.10.1 (10.10.10.1) 56(84) bytes of data.
64 bytes from 10.10.10.1: icmp_seq=1 ttl=64 time=603 ms
64 bytes from 10.10.10.1: icmp_seq=2 ttl=64 time=300 ms
64 bytes from 10.10.10.1: icmp_seq=3 ttl=64 time=304 ms
```

Note how the first ping was slower. That's because the VPN was "waking up" and being established. Afterwards, with the tunnel already established, the latency reduced.

At the same time, the `wg show` output on the home router will have changed to something like this:

```
$ sudo wg show
interface: wg0
  public key: <router public key>
  private key: (hidden)
  listening port: 51000

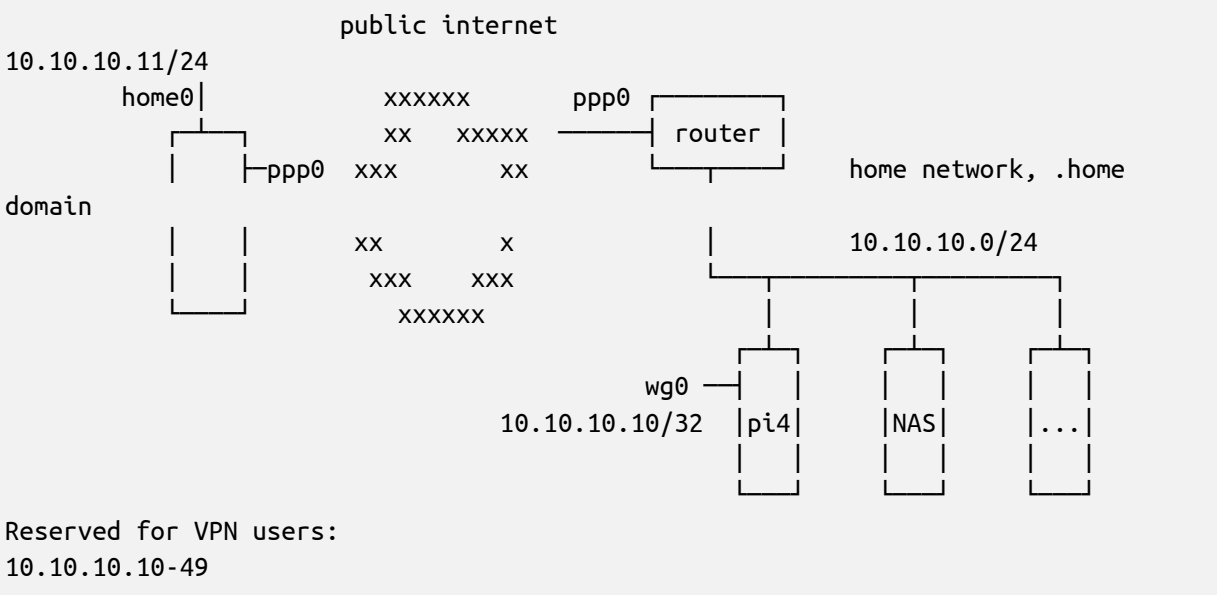
peer: <laptop public key>
  endpoint: <laptop public IP>:51000
  allowed ips: 10.10.11.2/32
  latest handshake: 1 minute, 8 seconds ago
  transfer: 564 B received, 476 B sent
```

### WireGuard on an internal system (peer-to-site)

Sometimes it's not possible to install WireGuard *on the home router itself*. Perhaps it's a closed system to which you do not have access, or there is no easy build for that architecture, or any of the other possible reasons.

However, you do have a spare system inside your network that you could use. Here we are going to show one way to make this work. There are others, but we believe this to be the least "involved" as it only requires a couple of (very common) changes in the router itself: NAT port forwarding, and *DHCP* range editing.

To recap, our home network has the `10.10.10.0/24` address, and we want to connect to it from a remote location and be "inserted" into that network as if we were there:




### Router changes

Since, in this scenario, we don't have a new network dedicated to our VPN users, we need to "carve out" a section of the home network and reserve it for the VPN.

The easiest way to reserve IPs for the VPN is to change the router configuration (assuming it's responsible for DHCP in this network) and tell its DHCP server to only hand out addresses from a specific range, leaving a "hole" for our VPN users.

For example, in the case of the `10.10.10.0/24` network, the DHCP server on the router might already be configured to hand out IP addresses from `10.10.10.2` to `10.10.10.254`. We can carve out a "hole" for our VPN users by reducing the DHCP range, as in this table:

| | |
|---|---|
| Network | `10.10.10.0/24` |
| Usable addresses | `10.10.10.2` – `10.10.10.254` (`.1` is the router) |
| DHCP range | `10.10.10.50` – `10.10.10.254` |
| VPN range | `10.10.10.10` – `10.10.10.49` |

Or via any other layout that is better suited for your case. In this way, the router will never hand out a DHCP address that conflicts with one that we selected for a VPN user.

The second change we need to do in the router is to **port forward** the WireGuard traffic to the internal system that will be the endpoint. In the diagram above, we selected the `10.10.10.10` system to be the internal WireGuard endpoint, and we will run it on the `51000/udp` port. Therefore, you need to configure the router to forward all `51000/udp` traffic to `10.10.10.10` on the same `51000/udp` port.

Finally, we also need to allow hosts on the internet to send traffic to the router on the `51000/udp` port we selected for WireGuard. This is done in the firewall rules of the device. Sometimes, performing the port forwarding as described earlier also configures the firewall to allow that traffic, but it's better to check.

Now we are ready to configure the internal endpoint.

### Configure the internal WireGuard endpoint

Install the `wireguard` package:

```
$ sudo apt install wireguard
```

Generate the keys for this host:

```
$ umask 077
$ wg genkey > internal-private.key
$ wg pubkey < internal-private.key > internal-public.key
```

And create the `/etc/wireguard/wg0.conf` file with these contents:

```
[Interface]
Address = 10.10.10.10/32
ListenPort = 51000
PrivateKey = <contents of internal-private.key>

[Peer]
# laptop
PublicKey = <contents of laptop-public.key>
AllowedIPs = 10.10.10.11/32 # any available IP in the VPN range
```

**Note**

Just like in the *peer-to-site* scenario with WireGuard on the router, there is no `Endpoint` configuration here for the laptop peer, because we don't know where it will be connecting from beforehand.

The final step is to configure this internal system as a router for the VPN users. For that, we need to enable a couple of settings:

- `ip_forward`: to enable forwarding (aka, routing) of traffic between interfaces.
- `proxy_arp`: to reply to Address Resolution Protocol (ARP) requests on behalf of the VPN systems, as if they were locally present on the network segment.

To do that, and make it persist across reboots, create the file `/etc/sysctl.d/70-wireguard-routing.conf` file with this content:

```
net.ipv4.ip_forward = 1
net.ipv4.conf.all.proxy_arp = 1
```

Then run this command to apply those settings:

```
$ sudo sysctl -p /etc/sysctl.d/70-wireguard-routing.conf -w
```

Now the WireGuard interface can be brought up:

```
$ sudo wg-quick up wg0
```

### Configuring the peer

The peer configuration will be very similar to what was done before. What changes will be the address, since now it won't be on an exclusive network for the VPN, but will have an address carved out of the home network block.

Let's call this new configuration file `/etc/wireguard/home0.conf`:

```
[Interface]
ListenPort = 51000
Address = 10.10.10.11/24
PrivateKey = <contents of the private key for this system>

[Peer]
PublicKey = <contents of internal-public.key>
Endpoint = <home-ppp0-IP-or-hostname>:51000
AllowedIPs = 10.10.10.0/24
```

And bring up this WireGuard interface:

```
$ sudo wg-quick up home0
```

> **Note**
>
> There is no need to add an index number to the end of the interface name. That is a convention, but not strictly a requirement.

### Testing

With the WireGuard interfaces up on both peers, traffic should flow seamlessly in the `10.10.10.0/24` network between remote and local systems.

More specifically, it's best to test the non-trivial cases, that is, traffic between the remote peer and a host other than the one with the WireGuard interface on the home network.

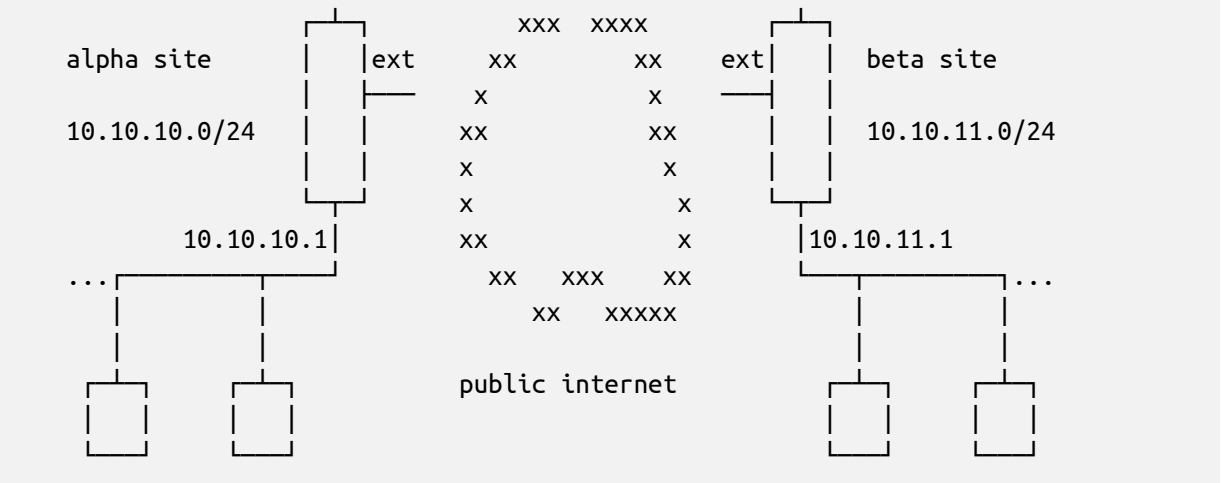
### WireGuard VPN site-to-site

Another usual VPN configuration where one could deploy WireGuard is to connect two distinct networks over the internet. Here is a simplified diagram:

```
                    ┌──────── WireGuard tunnel ────────┐
                    │          10.10.9.0/31            │
                    │                                  │
       10.10.9.0 wgA|                xx                |wgB 10.10.9.1
```

(continues on next page)

(continued from previous page)




The goal here is to seamlessly integrate network **alpha** with network **beta**, so that systems on the alpha site can transparently access systems on the beta site, and vice-versa.

Such a setup has a few particular details:

- Both peers are likely to be always up and running.
- We can't assume one side will always be the initiator, like the laptop in a coffee shop scenario.
- Because of the above, both peers should have a static endpoint, like a fixed IP address, or valid domain name.
- Since we are not assigning VPN IPs to all systems on each side, the VPN network here will be very small (a `/31`, which allows for two IPs) and only used for routing. The only systems with an IP in the VPN network are the gateways themselves.
- There will be no NAT applied to traffic going over the WireGuard network. Therefore, the networks of both sites **must** be different and not overlap.

This is what an MTR (My Traceroute) report from a system in the beta network to an alpha system will look like:

```
ubuntu@b1:~$ mtr -n -r 10.10.10.230
Start: 2022-09-02T18:56:51+0000
HOST: b1                  Loss%   Snt   Last   Avg  Best  Wrst StDev
  1.|-- 10.10.11.1         0.0%    10    0.1   0.1   0.1   0.2   0.0
  2.|-- 10.10.9.0          0.0%    10  299.6 299.3 298.3 300.0   0.6
  3.|-- 10.10.10.230       0.0%    10  299.1 299.1 298.0 300.2   0.6
```

**Note**

Technically, a `/31` Classless Inter-Domain Routing (CIDR) network has no usable IP addresses, since the first one is the network address, and the second (and last) one is the broadcast address. RFC 3021 allows for it, but if you encounter routing or other networking issues, switch to a `/30` CIDR and its two valid host IPs.

### Configure WireGuard

On the system that is the gateway for each site (that has internet connectivity), we start by installing WireGuard and generating the keys. For the **alpha** site:

```
$ sudo apt install wireguard
$ wg genkey | sudo tee /etc/wireguard/wgA.key
$ sudo cat /etc/wireguard/wgA.key | wg pubkey | sudo tee /etc/wireguard/wgA.pub
```

And the configuration on alpha will be:

```
[Interface]
PostUp = wg set %i private-key /etc/wireguard/%i.key
Address = 10.10.9.0/31
ListenPort = 51000

[Peer]
# beta site
PublicKey = <contents of /etc/wireguard/wgB.pub>
AllowedIPs = 10.10.11.0/24,10.10.9.0/31
Endpoint = <beta-gw-ip>:51000
```

On the gateway for the **beta** site we take similar steps:

```
$ sudo apt install wireguard
$ wg genkey | sudo tee /etc/wireguard/wgB.key
$ sudo cat /etc/wireguard/wgB.key | wg pubkey | sudo tee /etc/wireguard/wgB.pub
```

And create the corresponding configuration file for beta:

```
[Interface]
Address = 10.10.9.1/31
PostUp = wg set %i private-key /etc/wireguard/%i.key
ListenPort = 51000

[Peer]
# alpha site
PublicKey = <contents of /etc/wireguard/wgA.pub>
AllowedIPs = 10.10.10.0/24,10.10.9.0/31
Endpoint = <alpha-gw-ip>:51000
```

**Important**

WireGuard is being set up on the gateways for these two networks. As such, there are no changes needed on individual hosts of each network, but keep in mind that the WireGuard tunneling and encryption is only happening between the *alpha* and *beta* gateways, and **NOT** between the hosts of each network.

### Bring the interfaces up

Since this VPN is permanent between static sites, it's best to use the systemd unit file for `wg-quick` to bring the interfaces up and control them in general. In particular, we want them to be brought up automatically on reboot events.

On **alpha**:

```
$ sudo systemctl enable --now wg-quick@wgA
```

And similarly on **beta**:

```
$ sudo systemctl enable --now wg-quick@wgB
```

This both enables the interface on reboot, and starts it right away.

### Firewall and routing

Both gateways probably already have some routing and firewall rules. These might need changes depending on how they are set up.

The individual hosts on each network won't need any changes regarding the remote alpha or beta networks, because they will just send that traffic to the default gateway (as any other non-local traffic), which knows how to route it because of the routes that `wg-quick` added.

In the configuration we did so far, there have been no restrictions in place, so traffic between both sites flows without impediments.

In general, what needs to be done or checked is:

- Make sure both gateways can contact each other on the specified endpoint addresses and UDP port. In the case of this example, that is port `51000`. For extra security, create a firewall rule that only allows each peer to contact this port, instead of the Internet at large.
- Do NOT masquerade or NAT the traffic coming from the internal network and going out via the WireGuard interface towards the other site. This is purely routed traffic.
- There shouldn't be any routing changes needed on the gateways, since `wg-quick` takes care of adding the route for the remote site, but do check the routing table to see if it makes sense (`ip route` and `ip route | grep wg` are a good start).
- You may have to create new firewall rules if you need to restrict traffic between the alpha and beta networks.

  For example, if you want to prevent SSH between the sites, you could add a firewall rule like this one to **alpha**:

  ```
  $ sudo iptables -A FORWARD -i wgA -p tcp --dport 22 -j REJECT
  ```

  And similarly on **beta**:

  ```
  $ sudo iptables -A FORWARD -i wgB -p tcp --dport 22 -j REJECT
  ```

  You can add these as `PostUp` actions in the WireGuard interface config. Just don't forget to remove them in the corresponding `PreDown` hook, or you will end up with multiple rules.

WireGuard can also be set up to route all traffic through the VPN.

### Using the VPN as the default gateway

WireGuard can be set up to route all traffic through the VPN, and not just specific remote networks. There could be many reasons to do this, but mostly they are related to privacy.

Here we will assume a scenario where the local network is considered to be "untrusted", and we want to leak as little information as possible about our behaviour on the Internet. This could apply to the case of an airport, or a coffee shop, a conference, a hotel, or any other public network.

```
                      public untrusted          ┌── wg0 10.90.90.2/24
10.90.90.1/24         network/internet          │   VPN network
        wg0│            xxxxxx              ┌──────┴──┐
          ┌┴──┐          xx   xxxxx  ──────┤ VPN gw │
          │   ├─wlan0  xx       xx   eth0 └─────────┘
          │   │       xx        x
          │   │        xxx    xxx
          └───┘          xxxxxx
          Laptop
```

For the best results, we need a system we can reach on the internet and that we control. Most commonly this can be a simple small VM in a public cloud, but a home network also works. Here we will assume it's a brand new system that will be configured from scratch for this very specific purpose.

### Install and configure WireGuard

Let's start the configuration by installing WireGuard and generating the keys. On the client, run the following commands:

```
sudo apt install wireguard
umask 077
wg genkey > wg0.key
wg pubkey < wg0.key > wg0.pub
sudo mv wg0.key wg0.pub /etc/wireguard
```

And on the gateway server:

```
sudo apt install wireguard
umask 077
wg genkey > gateway0.key
wg pubkey < gateway0.key > gateway0.pub
sudo mv gateway0.key gateway0.pub /etc/wireguard
```

On the client, we will create `/etc/wireguard/wg0.conf`:

```
[Interface]
PostUp = wg set %i private-key /etc/wireguard/wg0.key
ListenPort = 51000
Address = 10.90.90.1/24

[Peer]
```

(continues on next page)

(continued from previous page)

```
PublicKey = <contents of gateway0.pub>
Endpoint = <public IP of gateway server>
AllowedIPs = 0.0.0.0/0
```

Key points here:

- We selected the `10.90.90.1/24` IP address for the WireGuard interface. This can be any private IP address, as long as it doesn't conflict with the network you are on, so double check that. If it needs to be changed, don't forget to also change the IP for the WireGuard interface on the gateway server.
- The `AllowedIPs` value is `0.0.0.0/0`, which means "all IPv4 addresses".
- We are using `PostUp` to load the private key instead of specifying it directly in the configuration file, so we don't have to set the permissions on the config file to `0600`.

The counterpart configuration on the gateway server is `/etc/wireguard/gateway0.conf` with these contents:

```
[Interface]
PostUp = wg set %i private-key /etc/wireguard/%i.key
Address = 10.90.90.2/24
ListenPort = 51000

[Peer]
PublicKey = <contents of wg0.pub>
AllowedIPs = 10.90.90.1/32
```

Since we don't know where this remote peer will be connecting from, there is no `Endpoint` setting for it, and the expectation is that the peer will be the one initiating the VPN.

This finishes the WireGuard configuration on both ends, but there is one extra step we need to take on the gateway server.

### Routing and masquerading

The WireGuard configuration that we did so far is enough to send the traffic from the client (in the untrusted network) to the gateway server. But what about from there onward? There are two extra configuration changes we need to make on the gateway server:

- Masquerade (or apply source NAT rules) the traffic from `10.90.90.1/24`.
- Enable IPv4 forwarding so our gateway server acts as a router.

To enable routing, create `/etc/sysctl.d/70-wireguard-routing.conf` with this content:

```
net.ipv4.ip_forward = 1
```

And run:

```
sudo sysctl -p /etc/sysctl.d/70-wireguard-routing.conf -w
```

To masquerade the traffic from the VPN, one simple rule is needed:

```
sudo iptables -t nat -A POSTROUTING -s 10.90.90.0/24 -o eth0 -j MASQUERADE
```

Replace `eth0` with the name of the network interface on the gateway server, if it's different.

To have this rule persist across reboots, you can add it to `/etc/rc.local` (create the file if it doesn't exist and make it executable):

```
#!/bin/sh
iptables -t nat -A POSTROUTING -s 10.90.90.0/24 -o eth0 -j MASQUERADE
```

This completes the gateway server configuration.

### Testing

Let's bring up the WireGuard interfaces on both peers. On the gateway server:

```
$ sudo wg-quick up gateway0
[#] ip link add gateway0 type wireguard
[#] wg setconf gateway0 /dev/fd/63
[#] ip -4 address add 10.90.90.2/24 dev gateway0
[#] ip link set mtu 1378 up dev gateway0
[#] wg set gateway0 private-key /etc/wireguard/gateway0.key
```

And on the client:

```
$ sudo wg-quick up wg0
[#] ip link add wg0 type wireguard
[#] wg setconf wg0 /dev/fd/63
[#] ip -4 address add 10.90.90.1/24 dev wg0
[#] ip link set mtu 1420 up dev wg0
[#] wg set wg0 fwmark 51820
[#] ip -4 route add 0.0.0.0/0 dev wg0 table 51820
[#] ip -4 rule add not fwmark 51820 table 51820
[#] ip -4 rule add table main suppress_prefixlength 0
[#] sysctl -q net.ipv4.conf.all.src_valid_mark=1
[#] nft -f /dev/fd/63
[#] wg set wg0 private-key /etc/wireguard/wg0.key
```

From the client you should now be able to verify that your traffic reaching out to the internet is going through the gateway server via the WireGuard VPN. For example:

```
$ mtr -r 1.1.1.1
Start: 2022-09-01T12:42:59+0000
HOST: laptop.lan                  Loss%   Snt   Last   Avg  Best  Wrst StDev
  1.|-- 10.90.90.2                  0.0%    10  184.9 185.5 184.9 186.9   0.6
  2.|-- 10.48.128.1                 0.0%    10  185.6 185.8 185.2 188.3   0.9
  (...)
  7.|-- one.one.one.one             0.0%    10  186.2 186.3 185.9 186.6   0.2
```

Above, hop 1 is the `gateway0` interface on the gateway server, then `10.48.128.1` is the default gateway for that server, then come some in-between hops, and the final hit is the target.

If you only look at the output of `ip route`, however, it's not immediately obvious that the WireGuard VPN is the default gateway:

```
$ ip route
default via 192.168.122.1 dev enp1s0 proto dhcp src 192.168.122.160 metric 100
10.90.90.0/24 dev wg0 proto kernel scope link src 10.90.90.1
192.168.122.0/24 dev enp1s0 proto kernel scope link src 192.168.122.160 metric 100

192.168.122.1 dev enp1s0 proto dhcp scope link src 192.168.122.160 metric 100
```

That's because WireGuard is using `fwmarks` and policy routing. WireGuard cannot simply set the `wg0` interface as the default gateway: that traffic needs to reach the specified endpoint on port `51000/UDP` outside of the VPN tunnel.

If you want to dive deeper into how this works, check `ip rule list`, `ip route list table 51820`, and consult the documentation on "Linux Policy Routing".

### DNS leaks

The traffic is now being routed through the VPN to the gateway server that you control, and from there onwards, to the Internet at large. The local network you are in cannot see the contents of that traffic, because it's encrypted. But you are still leaking information about the sites you access via *DNS*.

When the laptop got its IP address in the local (untrusted) network it is in, it likely also got a pair of IPs for DNS servers to use. These might be servers from that local network, or other DNS servers from the internet like `1.1.1.1` or `8.8.8.8`. When you access an internet site, a DNS query will be sent to those servers to discover their IP addresses. Sure, that traffic goes over the VPN, but at some point it exits the VPN, and then reaches those servers, which will then know what you are trying to access.

There are DNS leak detectors out there, and if you want a quick check you can try out https://dnsleaktest.com. It will tell you which DNS servers your connection is using, and it's up to you if you trust them or not. You might be surprised that even if you are in a conference network, for example, using a default gateway VPN like the one described here, you are still using the DNS servers from the conference infrastructure. In a way, the DNS traffic is leaving your machine encrypted, and then coming back in clear text to the local DNS server.

There are two things you can do about this: select a specific DNS server to use for your VPN connection, or install your own DNS server.

### Selecting a DNS server

If you can use a DNS server that you trust, or don't mind using, this is probably the easiest solution. Many people would start with the DNS server assigned to the gateway server used for the VPN. This address can be checked by running the following command in a shell on the gateway server:

```
$ resolvectl status
Global
       Protocols: -LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported
resolv.conf mode: stub
```

(continues on next page)

(continued from previous page)

```
Link 2 (ens2)
    Current Scopes: DNS
         Protocols: +DefaultRoute +LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported
Current DNS Server: 10.48.0.5
       DNS Servers: 10.48.0.5
        DNS Domain: openstacklocal

Link 5 (gateway0)
Current Scopes: none
     Protocols: -DefaultRoute +LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported
```

Look for `Current DNS Server`. In the example above, it's `10.48.0.5`.

Let's change the WireGuard `wg0` interface config to use that DNS server. Edit `/etc/wireguard/wg0.conf` and add a second `PostUp` line with the `resolvectl` command like below:

```
[Interface]
PostUp = wg set %i private-key /etc/wireguard/wg0.key
PostUp = resolvectl dns %i 10.48.0.5; resolvectl domain %i \~.
ListenPort = 51000
Address = 10.90.90.1/24

[Peer]
PublicKey = <contents of gateway0.pub>
Endpoint = <public IP of gateway server>
AllowedIPs = 0.0.0.0/0
```

You can run that `resolvectl` command by hand if you want to avoid having to restart the WireGuard VPN:

```
sudo resolvectl dns wg0 10.48.0.5; sudo resolvectl domain wg0 \~.
```

Or just restart the WireGuard interface:

```
sudo wg-quick down wg0; sudo wg-quick up wg0
```

And if you check again for DNS leaks, this time you will see that you are only using the DNS server you specified.

### Installing your own DNS server

If you don't want to use even the DNS server from the hosting provider where you have your gateway server, another alternative is to install your own DNS server.

There are multiple choices out there for this: `bind9` and `unbound` are quite popular, and it is easy to find quick tutorials and instructions on how to do it.

Here we will proceed with `bind9`, which is in the Ubuntu *main* repository.

On the gateway server, install the `bind9` package:

```
sudo apt install bind9
```

And that's it for the server part.

On the client, add a `PostUp` line specifying this IP (or change the line we added in the previous section):

```
[Interface]
PostUp = wg set %i private-key /etc/wireguard/wg0.key
PostUp = resolvectl dns %i 10.90.90.2; resolvectl domain %i \~.
ListenPort = 51000
Address = 10.90.90.1/24

[Peer]
PublicKey = <contents of gateway0.pub>
Endpoint = <public IP of gateway server>
AllowedIPs = 0.0.0.0/0
```

And restart the WireGuard interface. Now your VPN client will be using the gateway server as the DNS server.

There are also some common tasks and tips that will help you adjust your configuration to your needs.

### Common tasks in WireGuard VPN

Here are some common tasks and other useful tips that can help you in your WireGuard deployment.

### Controlling the WireGuard interface with systemd

The `wg-quick` tool is a simple way to bring the WireGuard interface up and down. That control is also exposed via a systemd service, which means the standard `systemctl` tool can be used.

Probably the greatest benefit of this is that it gives you the ability to configure the interface to be brought up automatically on system boot. For example, to configure the `wg0` interface to be brought up at boot, run the following command:

```
$ sudo systemctl enable wg-quick@wg0
```

The name of the systemd service follows the WireGuard interface name, and multiple such services can be enabled/started at the same time. You can also use the `systemctl status`, `start`, `stop`, `reload` and `restart` commands to control the WireGuard interface and query its status:

```
$ sudo systemctl reload wg-quick@wg0
```

The `reload` action does exactly what we expect: it reloads the configuration of the interface without disrupting existing WireGuard tunnels. To add or remove peers, `reload` is sufficient, but if `wg-quick` options, such as `PostUp`, `Address`, or similar are changed, then a `restart` is needed.

### DNS resolving

Let's say when you are inside the home network (literally – at home), you can connect to your other systems via *DNS* names, because your router at `10.10.10.1` can act as an internal DNS server. It would be nice to have this capability also when connected via the WireGuard VPN.

To do that, we can add a `PostUp` command to the WireGuard configuration to run a command for us right after the VPN is established. This command can be anything you would run in a shell (as root). We can use that to adjust the DNS resolver configuration of the laptop that is remotely connected to the home network.

For example, if we have a WireGuard setup as follows:

- `home0` WireGuard interface.
- `.home` DNS domain for the remote network.
- `10.10.10.1/24` is the DNS server for the `.home` domain, reachable after the VPN is established.

We can add this `PostUp` command to the `home0.conf` configuration file to have our systemd-based resolver use `10.10.10.1` as the DNS server for any queries for the `.home` domain:

```
[Interface]
...
PostUp = resolvectl dns %i 10.10.10.1; resolvectl domain %i \~home
```

For `PostUp` (and `PostDown` – see the `wg-quick(8)` manpage for details), the `%i` text is replaced with the WireGuard interface name. In this case, that would be `home0`.

These two `resolvectl` commands tell the local *systemd-resolved* resolver to:

- associate the DNS server at `10.10.10.1` to the `home0` interface, and
- associate the `home` domain to the `home0` interface.

When you bring the `home0` WireGuard interface up again, it will run the `resolvectl` commands:

```
$ sudo wg-quick up home0
[#] ip link add home0 type wireguard
[#] wg setconf home0 /dev/fd/63
[#] ip -4 address add 10.10.11.2/24 dev home0
[#] ip link set mtu 1420 up dev home0
[#] ip -4 route add 10.10.10.0/24 dev home0
[#] resolvectl dns home0 10.10.10.1; resolvectl domain home0 \~home
```

You can verify that it worked by pinging some *hostname* in your home network, or checking the DNS resolution status for the `home0` interface:

```
$ resolvectl status home0
Link 26 (home0)
    Current Scopes: DNS
         Protocols: -DefaultRoute +LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported
Current DNS Server: 10.10.10.1
       DNS Servers: 10.10.10.1
        DNS Domain: ~home
```

If you are using `systemctl` to control the WireGuard interface, this is the type of change (adding or changing `PostUp`) where the `reload` action won't be enough, and you actually need to issue a `restart`.

**Note**

The `wg-quick(8)` manpage documents the DNS setting of the WireGuard interface which has the same purpose, but only works if you have `resolveconf` installed. Ubuntu systems by default don't, and rely on `systemd-resolved` instead.

### Adding another peer

To add another peer to an existing WireGuard setup, we have to:

1. Generate a new keypair for the new peer
2. Create a new `[Peer]` section on the "other side" of the WireGuard setup
3. Pick a new IP for the new peer

Let's call the new system `ontheroad`, and generate the keys for it:

```
$ umask 077
$ wg genkey > ontheroad-private.key
$ wg pubkey < ontheroad-private.key > ontheroad-public.key
$ ls -la ontheroad.*
-rw------- 1 ubuntu ubuntu 45 Aug 22 20:12 ontheroad-private.key
-rw------- 1 ubuntu ubuntu 45 Aug 22 20:13 ontheroad-public.key
```

As for its IP address, let's pick `10.10.11.3/24` for it, which is the next one in the sequence from one of the previous examples in our WireGuard guide:

```
[Interface]
PrivateKey = <contents-of-ontheroad-private.key>
ListenPort = 51000
Address = 10.10.11.3/24

[Peer]
PublicKey = <contents-of-router-public.key>
Endpoint = <home-ppp0-IP-or-hostname>:51000
AllowedIPs = 10.10.11.0/24,10.10.10.0/24
```

The only difference between this config and one for an existing system in this same WireGuard setup will be `PrivateKey` and `Address`.

On the "other side", we add the new `[Peer]` section to the existing config:

```
[Interface]
PrivateKey = <contents-of-router-private.key>
ListenPort = 51000
Address = 10.10.11.1/24

[Peer]
```

(continues on next page)

(continued from previous page)

```
# laptop
PublicKey = <contents-of-laptop-public.key>
AllowedIPs = 10.10.11.2

[Peer]
# ontheroad
PublicKey = <contents-of-ontheroad-public.key>
AllowedIPs = 10.10.11.3
```

To update the interface with the new peer without disrupting existing connections, we use the `reload` action of the systemd unit:

```
$ systemctl reload wg-quick@wg0
```

**Note**

For this case of a "server" or "VPN gateway", where we are just adding another peer to an existing config, the `systemctl reload` action will work well enough to insert the new peer into the WireGuard configuration. However, it won't create new routes, or do any of the other steps that `wg-quick` does. Depending on your setup, you might need a full restart so that `wg-quick` can fully do its job.

### Adding a smartphone peer

WireGuard can be installed on many different platforms, and smartphones are included. The upstream installation page has links for Android and for iOS apps.

Such a mobile client can be configured more easily with the use of QR codes.

We start by creating the new peer's config normally, as if it were any other system (generate keys, pick an IP address, etc). Then, to convert that configuration file to a QR code, install the `qrencode` package:

```
$ sudo apt install qrencode
```

Next, run the following command (assuming the config was written to `phone.conf`):

```
$ cat phone.conf | qrencode -t ansiutf8
```

That will generate a QR code in the terminal, ready for scanning with the smartphone app. Note that there is no need for a graphical environment, and this command can be run remotely over SSH for example.

Note that you need to put the private key contents directly into that configuration file, and not use `PostUp` to load it from a separate file.

**Important**

Treat this QR code as a secret, as it contains the private key for the WireGuard interface!

### Security tips for WireGuard VPN

Here are some security tips for your WireGuard deployment.

### Traffic goes both ways

Remember that the VPN traffic goes both ways. Once you are connected to the remote network, it means any device on that network can connect back to you! That is, unless you create specific firewall rules for this VPN network.

Since WireGuard is "just" an interface, you can create normal firewall rules for its traffic, and control the access to the network resources as usual. This is done more easily if you have a dedicated network for the VPN clients.

### Using PreSharedKey

You can add another layer of cryptographic protection to your VPN with the `PreSharedKey` option. Its use is optional, and adds a layer of symmetric-key cryptography to the traffic between specific peers.

Such a key can be generated with the `genpsk` command:

```
$ wg genpsk
vxlX6eMMin8uhxbKEhe/iOxi8ru+q1qWzCdjESXoFZY=
```

And then used in a `[Peer]` section, like this:

```
[Peer]
PublicKey = ....
Endpoint = ....
AllowedIPs = ....
PresharedKey = vxlX6eMMin8uhxbKEhe/iOxi8ru+q1qWzCdjESXoFZY=
```

> **Note**
>
> Both sides need to have the same `PresharedKey` in their respective `[Peer]` sections.

### Preventing accidental leakage of private keys

When troubleshooting WireGuard, it's common to post the contents of the interface configuration file somewhere for others to help, like in a mailing list, or internet forum. Since the private key is listed in that file, one has to remember to strip or obfuscate it before sharing, or else the secret is leaked.

To avoid such mistakes, we can remove the private key from the configuration file and leave it in its own file. This can be done via a `PostUp`` hook. For example, let's update the home0.conf` file to use such a hook:

```
[Interface]
ListenPort = 51000
Address = 10.10.11.3/24
PostUp = wg set %i private-key /etc/wireguard/%i.key
```

(continues on next page)

(continued from previous page)

```
[Peer]
PublicKey = <contents-of-router-public.key>
Endpoint = 10.48.132.39:51000
AllowedIPs = 10.10.11.0/24,10.10.10.0/24
```

The `%i` macro is replaced by the WireGuard interface name (`home0` in this case). When the interface comes up, the `PostUp` shell commands will be executed with that substitution in place, and the private key for this interface will be set with the contents of the `/etc/wireguard/home0.key` file.

There are some other advantages to this method, and perhaps one disadvantage.

Pros:

- The configuration file can now safely be stored in version control, like a git repository, without fear of leaking the private key (unless you also use the `PreSharedKey` option, which is also a secret).
- Since the key is now stored in a file, you can give that file a meaningful name, which helps to avoid mix-ups with keys and peers when setting up WireGuard.

Cons:

- You cannot directly use the `qrcode` tool to convert this image to a QR code and use it to configure the mobile version of WireGuard, because that tool won't go after the private key in that separate file.

### Troubleshooting WireGuard VPN

The following general checklist should help as a first set of steps to try when you run into problems with WireGuard.

- **Verify public and private keys**: When dealing with multiple peers, it's easy to mix these up, especially because the contents of these keys are just random data. There is nothing identifying them, and public and private keys are basically the same (format-wise).
- **Verify** `AllowedIPs` **list on all peers**.
- **Check with** `ip route` **and** `ip addr show dev <wg-interface>` if the routes and IPs are set as you expect.
- Double check that you have `/proc/sys/net/ipv4/ip_forward` **set to** `1` where needed.
- When injecting the VPN users into an existing network, without routing, make sure `/proc/sys/net/ipv4/conf/all/proxy_arp` **is set to** `1`.
- Make sure the above `/proc` entries are in `/etc/sysctl.conf` **or a file in** `/etc/sysctl.d` so that they persist reboots.

### The watch wg command

It can be helpful to leave a terminal open with the `watch wg` command. Here is a sample output showing a system with two peers configured, where only one has established the VPN so far:

```
Every 2.0s: wg                  j-wg: Fri Aug 26 17:44:37 2022

interface: wg0
  public key: +T3T3HTMeyrEDvim8FBxbYjbz+/POeOtG3Rlvl9kJmM=
  private key: (hidden)
  listening port: 51000

peer: 2cJdFcNzXv4YUGyDTahtOfrbsrFsCByatPnNzKTs0Qo=
  endpoint: 10.172.196.106:51000
  allowed ips: 10.10.11.2/32
  latest handshake: 3 hours, 27 minutes, 35 seconds ago
  transfer: 3.06 KiB received, 2.80 KiB sent

peer: ZliZ1hlarZqvfxPMyME2ECtXDk611NB7uzLAD4McpgI=
  allowed ips: 10.10.11.3/32
```

### Kernel debug messages

WireGuard is also silent when it comes to logging. Being (essentially) a kernel module, we need to explicitly enable verbose logging of its module. This is done with the following command:

```
$ echo "module wireguard +p" | sudo tee /sys/kernel/debug/dynamic_debug/control
```

This will write WireGuard logging messages to the kernel log, which can be watched live with:

```
$ sudo dmesg -wT
```

To disable logging, run this:

```
$ echo "module wireguard -p" | sudo tee /sys/kernel/debug/dynamic_debug/control
```

### Destination address required

If you ping an IP and get back an error like this:

```
$ ping 10.10.11.2
PING 10.10.11.2 (10.10.11.2) 56(84) bytes of data.
From 10.10.11.1 icmp_seq=1 Destination Host Unreachable
ping: sendmsg: Destination address required
```

This is happening because the WireGuard interface selected for this destination doesn't know the endpoint for it. In other words, it doesn't know where to send the encrypted traffic.

One common scenario for this is on a peer where there is no `Endpoint` configuration, which is perfectly valid, and the host is trying to send traffic to that peer. Let's take the coffee shop scenario we described earlier as an example.

The laptop is connected to the VPN and exchanging traffic as usual. Then it stops for a bit (the person went to get one more cup). Traffic ceases (WireGuard is silent, remember). If the WireGuard on the home router is now restarted, when it comes back up, it won't know how to reach the laptop, because it was never contacted by it before. This means that at this time,

if the home router tries to send traffic to the laptop in the coffee shop, it will get the above error.

Now the laptop user comes back, and generates some traffic to the home network (remember: the laptop has the home network's `Endpoint` value). The VPN "wakes up", data is exchanged, handshakes are completed, and now the home router knows the `Endpoint` associated with the laptop, and can again initiate new traffic to it without issues.

Another possibility is that one of the peers is behind a NAT, and there wasn't enough traffic for the stateful firewall to consider the "connection" alive, and it dropped the NAT mapping it had. In this case, the peer might benefit from the `PersistentKeepalive` configuration, which makes WireGuard send a *keepalive* probe every so many seconds.

#### Required key not available

This error:

```
$ ping 10.10.11.1
PING 10.10.11.1 (10.10.11.1) 56(84) bytes of data.
From 10.10.11.2 icmp_seq=1 Destination Host Unreachable
ping: sendmsg: Required key not available
```

Can happen when you have a route directing traffic to the WireGuard interface, but that interface does not have the target address listed in its `AllowedIPs` configuration.

If you have enabled kernel debugging for WireGuard, you will also see a message like this one in the *dmesg* output:

```
wireguard: home0: No peer has allowed IPs matching 10.10.11.1
```

#### See also

- Explanation: *Introduction to WireGuard VPN*

#### See also

- Explanation: *Introduction to security*
- Explanation: *Security topics*

*System security* is a crucial topic for any Ubuntu user. In addition to general security topics such as setting up a firewall, AppArmor profiles and user/group management, you will also find how-to guides on:

- **Authentication** with Kerberos, network user authentication with SSSD and physical authentication with smart cards
- **Cryptography** with OpenSSH
- **Virtual Private Networks** OpenVPN and WireGuard VPN

# 2.3. Networking

## 2.3.1. Networking

This section contains how-to guides on most aspects of networking in Ubuntu. If you would like a broader overview of these topics before getting started, refer to our *introduction to*

*networking*.

### Configuration

Network configuration in Ubuntu is handled through Netplan. See our general walkthrough on *Configuring networks*, or refer to the Netplan documentation for more specific instructions.

### Network tools

The File Transfer Protocol (FTP) can be set up to provide files for download.

### Set up an FTP server

File Transfer Protocol (FTP) is a TCP protocol for downloading files between computers. In the past, it has also been used for uploading but, as that method does not use encryption, user credentials as well as data transferred in the clear and are easily intercepted. So if you are here looking for a way to upload and download files securely, see the *OpenSSH documentation* instead.

FTP works on a client/server model. The server component is called an *FTP daemon*. It continuously listens for FTP requests from remote clients. When a request is received, it manages the login and sets up the connection. For the duration of the session it executes any of commands sent by the FTP client.

Access to an FTP server can be managed in two ways:

- Anonymous
- Authenticated

In the Anonymous mode, remote clients can access the FTP server by using the default user account called “anonymous” or “ftp” and sending an email address as the password. In the Authenticated mode a user must have an account and a password. This latter choice is very insecure and should not be used except in special circumstances. If you are looking to transfer files securely see SFTP in the section on OpenSSH-Server. User access to the FTP server directories and files is dependent on the permissions defined for the account used at login. As a general rule, the FTP daemon will hide the root directory of the FTP server and change it to the FTP Home directory. This hides the rest of the file system from remote sessions.

### vsftpd - FTP Server Installation

vsftpd is an FTP daemon available in Ubuntu. It is easy to install, set up, and maintain. To install vsftpd you can run the following command:

```
sudo apt install vsftpd
```

### Anonymous FTP Configuration

By default vsftpd is *not* configured to allow anonymous download. If you wish to enable anonymous download edit `/etc/vsftpd.conf` by changing:

```
anonymous_enable=YES
```

During installation a *ftp* user is created with a home directory of `/srv/ftp`. This is the default FTP directory.

If you wish to change this location, to `/srv/files/ftp` for example, simply create a directory in another location and change the *ftp* user's home directory:

```
sudo mkdir -p /srv/files/ftp
sudo usermod -d /srv/files/ftp ftp
```

After making the change restart vsftpd:

```
sudo systemctl restart vsftpd.service
```

Finally, copy any files and directories you would like to make available through anonymous FTP to `/srv/files/ftp`, or `/srv/ftp` if you wish to use the default.

### User Authenticated FTP Configuration

By default vsftpd is configured to authenticate system users and allow them to download files. If you want users to be able to upload files, edit `/etc/vsftpd.conf`:

```
write_enable=YES
```

Now restart vsftpd:

```
sudo systemctl restart vsftpd.service
```

Now when system users login to FTP they will start in their *home* directories where they can download, upload, create directories, etc.

Similarly, by default, anonymous users are not allowed to upload files to FTP server. To change this setting, you should uncomment the following line, and restart vsftpd:

```
anon_upload_enable=YES
```

**Warning**

Enabling anonymous FTP upload can be an extreme security risk. It is best to not enable anonymous upload on servers accessed directly from the Internet.

The configuration file consists of many configuration parameters. The information about each parameter is available in the configuration file. Alternatively, you can refer to the man page, `man 5 vsftpd.conf` for details of each parameter.

### Securing FTP

There are options in `/etc/vsftpd.conf` to help make vsftpd more secure. For example users can be limited to their home directories by uncommenting:

```
chroot_local_user=YES
```

You can also limit a specific list of users to just their home directories:

```
chroot_list_enable=YES
chroot_list_file=/etc/vsftpd.chroot_list
```

After uncommenting the above options, create a `/etc/vsftpd.chroot_list` containing a list of users one per line. Then restart vsftpd:

```
sudo systemctl restart vsftpd.service
```

Also, the `/etc/ftpusers` file is a list of users that are *disallowed* FTP access. The default list includes root, daemon, nobody, etc. To disable FTP access for additional users simply add them to the list.

FTP can also be encrypted using *FTPS*. Different from *SFTP*, *FTPS* is FTP over Secure Socket Layer (SSL). *SFTP* is a FTP like session over an encrypted *SSH* connection. A major difference is that users of SFTP need to have a *shell* account on the system, instead of a *nologin* shell. Providing all users with a shell may not be ideal for some environments, such as a shared web host. However, it is possible to restrict such accounts to only SFTP and disable shell interaction.

To configure *FTPS*, edit `/etc/vsftpd.conf` and at the bottom add:

```
ssl_enable=YES
```

Also, notice the certificate and key related options:

```
rsa_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
rsa_private_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
```

By default these options are set to the certificate and key provided by the ssl-cert package. In a production environment these should be replaced with a certificate and key generated for the specific host. For more information on certificates see *Security - Certificates*.

Now restart vsftpd, and non-anonymous users will be forced to use *FTPS*:

```
sudo systemctl restart vsftpd.service
```

To allow users with a shell of `/usr/sbin/nologin` access to FTP, but have no shell access, edit `/etc/shells` adding the *nologin* shell:

```
# /etc/shells: valid login shells
/bin/csh
/bin/sh
/usr/bin/es
/usr/bin/ksh
/bin/ksh
/usr/bin/rc
/usr/bin/tcsh
/bin/tcsh
/usr/bin/esh
/bin/dash
/bin/bash
/bin/rbash
```

(continues on next page)

(continued from previous page)

```
/usr/bin/screen
/usr/sbin/nologin
```

This is necessary because, by default vsftpd uses PAM for authentication, and the `/etc/pam.d/vsftpd` configuration file contains:

```
auth    required        pam_shells.so
```

The *shells* PAM module restricts access to shells listed in the `/etc/shells` file.

Most popular FTP clients can be configured to connect using FTPS. The lftp command line FTP client has the ability to use FTPS as well.

### References

- See the vsftpd website for more information.

The Domain Name Service (DNS) maps IP addresses to fully qualified domain names (FQDN). The DNS Security Extensions (DNSSEC) allow DNS data to be verified.

### Domain Name Service (DNS)

Domain Name Service (DNS) is an Internet service that maps IP addresses and *fully qualified domain names (FQDN)* to one another. In this way, DNS alleviates the need to remember IP addresses. Computers that run DNS are called **name servers**. Ubuntu ships with the Berkley Internet Naming Daemon (BIND), the most common program used for maintaining a name server on Linux.

### Install DNS

At a terminal prompt, run the following command to install the `bind9` package:

```
sudo apt install bind9
```

A useful package for testing and troubleshooting DNS issues is the `dnsutils` package. Very often these tools will be installed already, but to check and/or install `dnsutils` enter the following:

```
sudo apt install dnsutils
```

### DNS configuration overview

There are many ways to configure BIND9. Some of the most common configurations include a caching nameserver, primary server, and secondary server.

- When configured as a **caching nameserver**, BIND9 will find the answer to name queries and remember the answer when the domain is queried again.
- As a **primary server**, BIND9 reads the data for a zone from a file on its host, and is authoritative for that zone.
- As a **secondary server**, BIND9 gets the zone data from another nameserver that is authoritative for the zone.

### Configuration files

The DNS configuration files are stored in the `/etc/bind` directory. The primary configuration file is `/etc/bind/named.conf`, which in the layout provided by the package just includes these files:

- `/etc/bind/named.conf.options`: Global DNS options
- `/etc/bind/named.conf.local`: For your zones
- `/etc/bind/named.conf.default-zones`: Default zones such as localhost, its reverse, and the root hints

The root nameservers used to be described in the file `/etc/bind/db.root`. This is now provided instead by the `/usr/share/dns/root.hints` file shipped with the `dns-root-data` package, and is referenced in the `named.conf.default-zones` configuration file above.

It is possible to configure the same server to be a caching name server, primary, and secondary: it all depends on the zones it is serving. A server can be the Start of Authority (SOA) for one zone, while providing secondary service for another zone. All the while providing caching services for hosts on the local LAN.

### Set up a caching nameserver

The default configuration acts as a caching server. Simply uncomment and edit `/etc/bind/named.conf.options` to set the IP addresses of your ISP's DNS servers:

```
forwarders {
    1.2.3.4;
    5.6.7.8;
};
```

**Note**

Replace `1.2.3.4` and `5.6.7.8` with the IP addresses of actual nameservers.

To enable the new configuration, restart the DNS server. From a terminal prompt, run:

```
sudo systemctl restart bind9.service
```

*See the dig section* for information on testing a caching DNS server.

### Set up a primary server

In this section BIND9 will be configured as the primary server for the domain `example.com`. You can replace `example.com` with your FQDN (Fully Qualified Domain Name).

### Forward zone file

To add a DNS zone to BIND9, turning BIND9 into a primary server, first edit `/etc/bind/named.conf.local`:

```
zone "example.com" {
    type master;
    file "/etc/bind/db.example.com";
};
```

**Note**

If BIND will be receiving automatic updates to the file as with *DDNS*, then use `/var/lib/bind/db.example.com` rather than `/etc/bind/db.example.com` both here and in the copy command below.

Now use an existing zone file as a template to create the `/etc/bind/db.example.com` file:

```
sudo cp /etc/bind/db.local /etc/bind/db.example.com
```

Edit the new zone file `/etc/bind/db.example.com` and change `localhost.` to the FQDN of your server, including the additional `.` at the end. Change `127.0.0.1` to the nameserver's IP address and `root.localhost` to a valid email address, but with a `.` instead of the usual `@` symbol, again including the `.` at the end. Change the comment to indicate the domain that this file is for.

Create an **A record** for the base domain, `example.com`. Also, create an **A record** for `ns.example.com`, the name server in this example:

```
;
; BIND data file for example.com
;
$TTL    604800
@       IN      SOA     example.com. root.example.com. (
                              2         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL

@       IN      NS      ns.example.com.
@       IN      A       192.168.1.10
@       IN      AAAA    ::1
ns      IN      A       192.168.1.10
```

You must increment the `Serial Number` every time you make changes to the zone file. If you make multiple changes before restarting BIND9, only increment `Serial` once.

Now, you can add DNS records to the bottom of the zone file. See *Common Record Types* for details.

**Note**

Many admins like to use the "last edited" date as the Serial of a zone, such as **2020012100** which is **yyyymmddss** (where **ss** is the Serial Number)

Once you have made changes to the zone file, BIND9 needs to be restarted for the changes to take effect:

```
sudo systemctl restart bind9.service
```

### Reverse zone file

Now that the zone is set up and resolving names to IP Addresses, a **reverse zone** needs to be added to allow DNS to resolve an address to a name.

Edit `/etc/bind/named.conf.local` and add the following:

```
zone "1.168.192.in-addr.arpa" {
    type master;
    file "/etc/bind/db.192";
};
```

> **Note**
>
> Replace `1.168.192` with the first three octets of whatever network you are using. Also, name the zone file `/etc/bind/db.192` appropriately. It should match the first octet of your network.

Now create the `/etc/bind/db.192` file:

```
sudo cp /etc/bind/db.127 /etc/bind/db.192
```

Next edit `/etc/bind/db.192`, changing the same options as `/etc/bind/db.example.com`:

```
;
; BIND reverse data file for local 192.168.1.XXX net
;
$TTL    604800
@       IN      SOA     ns.example.com. root.example.com. (
                              2         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      ns.
10      IN      PTR     ns.example.com.
```

The `Serial Number` in the reverse zone needs to be incremented on each change as well. For each **A record** you configure in `/etc/bind/db.example.com` that is for a different address, you will need to create a **PTR record** in `/etc/bind/db.192`.

After creating the reverse zone file restart BIND9:

```
sudo systemctl restart bind9.service
```

### Set up a secondary server

Once a primary server has been configured, a **secondary server** is highly recommended. This will maintain the availability of the domain if the primary becomes unavailable.

First, on the primary server, the zone transfer needs to be allowed. Add the `allow-transfer` option to the example **Forward** and **Reverse** zone definitions in `/etc/bind/named.conf.local`:

```
zone "example.com" {
    type master;
    file "/etc/bind/db.example.com";
    allow-transfer { 192.168.1.11; };
};

zone "1.168.192.in-addr.arpa" {
    type master;
    file "/etc/bind/db.192";
    allow-transfer { 192.168.1.11; };
};
```

**Note**

Replace `192.168.1.11` with the IP address of your secondary nameserver.

Restart BIND9 on the primary server:

```
sudo systemctl restart bind9.service
```

Next, on the secondary server, install the `bind9` package the same way as on the primary. Then edit the `/etc/bind/named.conf.local` and add the following declarations for the Forward and Reverse zones:

```
zone "example.com" {
    type secondary;
    file "db.example.com";
    masters { 192.168.1.10; };
};

zone "1.168.192.in-addr.arpa" {
    type secondary;
    file "db.192";
    masters { 192.168.1.10; };
};
```

Once again, replace `192.168.1.10` with the IP address of your primary nameserver, then restart BIND9 on the secondary server:

```
sudo systemctl restart bind9.service
```

In `/var/log/syslog` you should see something similar to the following (some lines have been split to fit the format of this document):

```
client 192.168.1.10#39448: received notify for zone '1.168.192.in-addr.arpa'
zone 1.168.192.in-addr.arpa/IN: Transfer started.
transfer of '100.18.172.in-addr.arpa/IN' from 192.168.1.10#53:
 connected using 192.168.1.11#37531
zone 1.168.192.in-addr.arpa/IN: transferred serial 5
transfer of '100.18.172.in-addr.arpa/IN' from 192.168.1.10#53:
 Transfer completed: 1 messages,
6 records, 212 bytes, 0.002 secs (106000 bytes/sec)
zone 1.168.192.in-addr.arpa/IN: sending notifies (serial 5)

client 192.168.1.10#20329: received notify for zone 'example.com'
zone example.com/IN: Transfer started.
transfer of 'example.com/IN' from 192.168.1.10#53: connected using 192.168.1.11
#38577
zone example.com/IN: transferred serial 5
transfer of 'example.com/IN' from 192.168.1.10#53: Transfer completed: 1 messages,
8 records, 225 bytes, 0.002 secs (112500 bytes/sec)
```

**Note**

A zone is only transferred if the `Serial Number` on the primary is larger than the one on the secondary. If you want to have your primary DNS notify other secondary DNS servers of zone changes, you can add `also-notify { ipaddress; };` to `/etc/bind/named.conf.local` as shown in the example below:

```
zone "example.com" {
    type master;
    file "/etc/bind/db.example.com";
    allow-transfer { 192.168.1.11; };
    also-notify { 192.168.1.11; };
};

zone "1.168.192.in-addr.arpa" {
    type master;
    file "/etc/bind/db.192";
    allow-transfer { 192.168.1.11; };
    also-notify { 192.168.1.11; };
};
```

**Note**

The default directory for non-authoritative zone files is `/var/cache/bind/`. This directory is also configured in AppArmor to allow the named daemon to write to it. See this page for *more information on AppArmor*.

## Testing your setup

### resolv.conf

The first step in testing BIND9 is to add the nameserver's IP address to a **hosts resolver**. The Primary nameserver should be configured as well as another host to double check things. Refer to *DNS client configuration* for details on adding nameserver addresses to your network clients. In the end your `nameserver` line in `/etc/resolv.conf` should be pointing at `127.0.0.53` and you should have a `search` parameter for your domain. Something like this:

```
nameserver  127.0.0.53
search example.com
```

To check which DNS server your local resolver is using, run:

```
resolvectl status
```

> **Note**
>
> You should also add the IP address of the secondary nameserver to your client configuration in case the primary becomes unavailable.

### dig

If you installed the `dnsutils` package you can test your setup using the DNS lookup utility `dig`:

After installing BIND9 use `dig` against the loopback interface to make sure it is listening on port 53. From a terminal prompt:

```
dig -x 127.0.0.1
```

You should see lines similar to the following in the command output:

```
;; Query time: 1 msec
;; SERVER: 192.168.1.10#53(192.168.1.10)
```

If you have configured BIND9 as a caching nameserver, "dig" an outside domain to check the query time:

```
dig ubuntu.com
```

Note the query time toward the end of the command output:

```
;; Query time: 49 msec
```

After a second `dig` there should be improvement:

```
;; Query time: 1 msec
```

### ping

Now let's demonstrate how applications make use of DNS to resolve a host name, by using the `ping` utility to send an ICMP echo request:

```
ping example.com
```

This tests if the nameserver can resolve the name `ns.example.com` to an IP address. The command output should resemble:

```
PING ns.example.com (192.168.1.10) 56(84) bytes of data.
64 bytes from 192.168.1.10: icmp_seq=1 ttl=64 time=0.800 ms
64 bytes from 192.168.1.10: icmp_seq=2 ttl=64 time=0.813 ms
```

### named-checkzone

A great way to test your zone files is by using the `named-checkzone` utility installed with the `bind9` package. This utility allows you to make sure the configuration is correct before restarting BIND9 and making the changes live.

To test our example forward zone file, enter the following from a command prompt:

```
named-checkzone example.com /etc/bind/db.example.com
```

If everything is configured correctly you should see output similar to:

```
zone example.com/IN: loaded serial 6
OK
```

Similarly, to test the reverse zone file enter the following:

```
named-checkzone 1.168.192.in-addr.arpa /etc/bind/db.192
```

The output should be similar to:

```
zone 1.168.192.in-addr.arpa/IN: loaded serial 3
OK
```

> **Note**
>
> The Serial Number of your zone file will probably be different.

### Quick temporary query logging

With the `rndc` tool, you can quickly turn query logging on and off, without restarting the service or changing the configuration file.

To turn query logging on, run:

```
sudo rndc querylog on
```

Likewise, to turn it off, run:

```
sudo rndc querylog off
```

The logs will be sent to `syslog` and will show up in `/var/log/syslog` by default:

```
Jan 20 19:40:50 new-n1 named[816]: received control channel command 'querylog on'
Jan 20 19:40:50 new-n1 named[816]: query logging is now on
Jan 20 19:40:57 new-n1 named[816]: client @0x7f48ec101480 192.168.1.10#36139
(ubuntu.com): query: ubuntu.com IN A +E(0)K (192.168.1.10)
```

> **Note**
>
> The amount of logs generated by enabling `querylog` could be huge!

### Logging

BIND9 has a wide variety of logging configuration options available, but the two main ones are **channel** and **category**, which configure where logs go, and what information gets logged, respectively.

If no logging options are configured the default configuration is:

```
logging {
     category default { default_syslog; default_debug; };
     category unmatched { null; };
};
```

Let's instead configure BIND9 to send **debug** messages related to DNS queries to a separate file.

We need to configure a **channel** to specify which file to send the messages to, and a **category**. In this example, the category will log all queries. Edit `/etc/bind/named.conf.local` and add the following:

```
logging {
    channel query.log {
        file "/var/log/named/query.log";
        severity debug 3;
    };
    category queries { query.log; };
};
```

> **Note**
>
> The `debug` option can be set from 1 to 3. If a level isn't specified, level 1 is the default.

Since the **named daemon** runs as the `bind` user, the `/var/log/named` directory must be created and the ownership changed:

```
sudo mkdir /var/log/named
sudo chown bind:bind /var/log/named
```

Now restart BIND9 for the changes to take effect:

```
sudo systemctl restart bind9.service
```

You should see the file `/var/log/named/query.log` fill with query information. This is a simple example of the BIND9 logging options. For coverage of advanced options see the "Further Reading" section at the bottom of this page.

### Common record types

This section covers some of the most common DNS record types.

- `A` **record** This record maps an IP address to a *hostname*.

```
www      IN    A      192.168.1.12
```

- `CNAME` **record** Used to create an alias to an existing A record. You cannot create a `CNAME` record pointing to another `CNAME` record.

```
web      IN    CNAME  www
```

- `MX` **record** Used to define where emails should be sent to. Must point to an `A` record, not a `CNAME`.

```
@        IN    MX  1   mail.example.com.
mail     IN    A       192.168.1.13
```

- `NS` **record** Used to define which servers serve copies of a zone. It must point to an `A` record, not a `CNAME`. This is where primary and secondary servers are defined.

```
@        IN    NS     ns.example.com.
@        IN    NS     ns2.example.com.
ns       IN    A      192.168.1.10
ns2      IN    A      192.168.1.11
```

### Further reading

- *Installing DNS Security Extensions (DNSSEC)*
- Upstream BIND9 Documentation
- DNS and BIND is a popular book now in its fifth edition. There is now also a DNS and BIND on IPv6 book.
- A great place to ask for BIND9 assistance, and get involved with the Ubuntu Server community, is the `#ubuntu-server` IRC channel on Libera Chat.

### Installing DNS Security Extensions (DNSSEC)

DNSSEC is a set of security extensions to *DNS* which allow DNS data to be verified for authenticity and integrity.

This guide will show you how to enable DNSSEC for an existing zone in your BIND9 DNS server deployment, by serving additional records like `DS`, `DNSKEY` or `RRSIG` for signed zones.

### Starting point

The starting point for this how-to is an existing BIND9 DNS server deployed with an authoritative zone. For details on how to deploy BIND9 in this fashion, please see the *DNS How-To*. One key difference from that guide, however, is that we need the zone file to be in a directory where the server can write to, like `/var/lib/bind`, instead of `/etc/bind`.

Nevertheless, here is a quick set of steps to reach that state for an example domain called `example.internal`.

First, install the `bind9` package:

```
sudo apt install bind9 -y
```

Edit `/etc/bind/named.conf.local` and add this *zone* block:

```
zone "example.internal" {
    type master;
    file "/var/lib/bind/db.example.internal";
};
```

Create the file `/var/lib/bind/db.example.internal` with these contents:

```
$TTL 86400      ; 1 day
example.internal.       IN SOA  example.internal. root.example.internal. (
                                1          ; serial
                                43200      ; refresh (12 hours)
                                900        ; retry (15 minutes)
                                1814400    ; expire (3 weeks)
                                7200       ; minimum (2 hours)
                                )
example.internal.       IN  NS      ns.example.internal.
ns                      IN  A       192.168.1.10
noble                   IN  A       192.168.1.11
```

Restart the service:

```
sudo systemctl restart named
```

Check if the service can resolve the name `noble.example.internal`:

```
$ dig @127.0.0.1 +short noble.example.internal
192.168.1.11
```

### Enabling DNSSEC

Enabling DNSSEC for a zone involves multiple steps. Thankfully for us, the BIND9 DNS server takes care of all of them by default, automatically, leaving very little for us to do. Converting a zone in this way means at least generating new keys, and signing all the Resource Records of the zone. But that is only “day 1”: such a zone must be maintained properly. Keys must be rotated, expiration dates must be set, etc. The current versions of BIND9 take care of that with a DNSSEC policy, and of course there is a default one that can be used from the start.

To migrate our *example.internal* zone to DNSSEC, we just need to add two lines to its definition in `/etc/bind/named.conf.local`, so that it looks like this:

```
zone "example.internal" {
    type master;
    file "/var/lib/bind/db.example.internal";
    dnssec-policy default;
    inline-signing yes;
};
```

What was added:

- `dnssec-policy default`: Use the default DNSSEC policy. A DNSSEC policy includes settings for key rotation, default TTL, and many others.
- `inline-signing yes`: keep a separate file for the signed zone.

After this change, there is no need to restart the service, but it needs to be told to reload its configuration. This can be done with the `rndc` tool:

```
sudo rndc reconfig
```

The server will immediately notice the new configuration and start the process to sign the *example.internal* zone. The journal logs will show the progress, and can be inspected with the command:

```
sudo journalctl -u named.service -f
```

The logs will show something similar to this:

```
named[3246]: zone example.internal/IN (unsigned): loaded serial 1
named[3246]: zone example.internal/IN (signed): loaded serial 1
named[3246]: zone example.internal/IN (signed): receive_secure_serial: unchanged
named[3246]: zone example.internal/IN (signed): sending notifies (serial 1)
named[3246]: zone example.internal/IN (signed): reconfiguring zone keys
named[3246]: keymgr: DNSKEY example.internal/ECDSAP256SHA256/44911 (CSK) created
for policy default
named[3246]: Fetching example.internal/ECDSAP256SHA256/44911 (CSK) from key
repository.
named[3246]: DNSKEY example.internal/ECDSAP256SHA256/44911 (CSK) is now published
named[3246]: DNSKEY example.internal/ECDSAP256SHA256/44911 (CSK) is now active
named[3246]: zone example.internal/IN (signed): next key event: 23-Oct-2024
22:47:12.544
named[3246]: any newly configured zones are now loaded
named[3246]: running
named[3246]: resolver priming query complete: success
named[3246]: managed-keys-zone: Key 20326 for zone . is now trusted (acceptance
timer complete)
named[3246]: zone example.internal/IN (signed): sending notifies (serial 3)
```

Depending on the zone size, signing all records can take longer.

A few interesting events can be seen in the logs above:

- Keys were generated for the *example.internal* zone.

- The *example.internal* zone became signed.
- A *key event* was scheduled. This is BIND9 also taking care of the maintenance tasks of the signed zone and its keys.
- Since the zone changed, its serial number was incremented (started as 1, now it's 3).

The DNSSEC keys are kept in `/var/cache/bind`:

```
-rw-r--r-- 1 bind bind  413 Oct 23 20:50 Kexample.internal.+013+48112.key
-rw------- 1 bind bind  215 Oct 23 20:50 Kexample.internal.+013+48112.private
-rw-r--r-- 1 bind bind  647 Oct 23 20:50 Kexample.internal.+013+48112.state
```

This is the bulk of the work. This zone is now signed, and maintained automatically by BIND9 using the *default* DNSSEC policy.

### Verification

The zone that was just signed is almost ready to serve DNSSEC. Let's perform some verification steps.

As it is now, this zone *example.internal* is "disconnected" from the parent zone. Its name was made up for this guide, but even if it represented a real domain, it would still be missing the connection to the parent zone. Remember that DNSSEC relies on the chain of trust, and the parent of our zone needs to be able to vouch for it.

Before taking that step, however, it's important to verify if everything else is working. In particular, we would want to perform some DNSSEC queries, and perform validation. A good tool to perform this validation is `delv`.

`delv` is similar to `dig`, but it will perform validation on the results using the same internal resolver and validator logic as the BIND9 server itself.

Since our zone is disconnected, we need to tell `delv` to use the public key created for the zone as a trusted anchor, and to not try to reach out to the root servers of the internet.

First, copy the public zone key somewhere else so it can be edited. For example:

```
cp /var/cache/bind/Kexample.internal.+013+48112.key /tmp/example.key
```

That file will have some comments at the top, and then have a line that starts with the zone name, like this (the full key was truncated below for brevity):

```
example.internal. 3600 IN DNSKEY 257 3 13 jkmS5hfyY3nSww....
```

We need to make some changes here:

- Remove the comment lines from the top.
- Edit that line and replace the `3600 IN DNSKEY` text with `static-key`.
- The key material after the `13` number must be enclosed in double quotes (").
- The line needs to end with a `;`.
- And finally the line needs to be inside a `trust-anchors` block.

In the end, the `/tmp/example.key` file should look like this:

```
trust-anchors {
    example.internal. static-key 257 3 13 "jkmS5hfyY3nSww....";
};
```

Now `delv` can be used to query the *example.internal* zone and perform DNSSEC validation:

```
$ delv @127.0.0.1 -a /tmp/example.key +root=example.internal noble.example.
internal +multiline
; fully validated
noble.example.internal. 86400 IN A 192.168.1.11
noble.example.internal. 86400 IN RRSIG A 13 3 86400 (
                                20241106131533 20241023195023 48112 example.
internal.
                                5fL4apIwCD9kt4XbzzlLxMXY3mj8Li1WZu3qzlcBpERp
                                lXPgLODbRrWyp7L81xEFnfhecKtEYv+6Y0Xa5iVRug== )
```

This output shows important DNSSEC attributes:

- `; fully validated`: The DNSSEC validation was completed successfully, and the presented data is authenticated.
- `RRSIG`: This is the signature Resource Record that accompanies the `A` record from the result.

### Connecting the dots: the parent zone

In order to complete the chain of trust, the parent zone needs to be able to vouch for the child zone we just signed. How this is exactly done varies, depending on who is the administrator of the parent zone. It could be as simple as just pasting the DS or DNSKEY records in a form.

Typically what is needed is either a DS record, or a DNSKEY record. Here is how to produce them, ready to be sent to the parent.

### DS record format

A DS record can be produced from the zone public key via the `dnssec-dsfromkey` tool. For example:

```
$ dnssec-dsfromkey /var/cache/bind/Kexample.internal.+013+48112.key
example.internal. IN DS 48112 13 2
1212DE7DA534556F1E11898F2C7A66736D5107962A19A0BFE1C2A67D6841962A
```

### DNSKEY record format

The DNSKEY format doesn't need tooling, and can be found directly in the zone's public key file, after the lines that start with the comment delimiter `;`:

```
$ cat /var/cache/bind/Kexample.internal.+013+48112.key
; This is a key-signing key, keyid 48112, for example.internal.
; Created: 20241023205023 (Wed Oct 23 20:50:23 2024)
; Publish: 20241023205023 (Wed Oct 23 20:50:23 2024)
; Activate: 20241023205023 (Wed Oct 23 20:50:23 2024)
```

(continues on next page)

(continued from previous page)

```
; SyncPublish: 20241024215523 (Thu Oct 24 21:55:23 2024)
example.internal. 3600 IN DNSKEY 257 3 13 jkmS5hfyY3nSww4tD9Fy5d+GGc3A/
zR1CFUxmN8T2TKTkgGWp8dusllM 7TrIZTEg6wZxmMs754/ftoTA6jmM1g==
```

Taking that line, the actual record to send to the parent zone just needs a little bit of tweaking so it looks like this:

```
example.internal. 3600 IN DNSKEY 257 3 13 (jkmS5hfyY3nSww4tD9Fy5d+GGc3A/
zR1CFUxmN8T2TKTkgGWp8dusllM 7TrIZTEg6wZxmMs754/ftoTA6jmM1g==);
```

### Further reading

- The DNSSEC Guide from bind9
- Easy-Start Guide for Signing Authoritative Zones
- Creating a Custom DNSSEC Policy
- Detailed DNSSEC chapter from the bind9 documentation
- `delv(1)` manual page
- Working with the parent zone

### Basic DNSSEC troubleshooting

Some of the troubleshooting tips that will be shown here are focused on the BIND9 *DNS* server and its tools, but the general principle applies to *DNSSEC* in all implementations.

### Handy “bad” and “good” DNSSEC domains

It helps to have some good known domains with broken and working DNSSEC available for testing, so we can be sure our tooling is catching those, and not just failing everywhere. There is no guarantee that these domains will be up forever, and certainly there are more out there, but this list is a good first choice:

- These should fail DNSSEC validation:
  - *dnssec-failed.org*
  - *sigfail.ippacket.stream*
- These should pass DNSSEC validation:
  - *isc.org*
  - *sigok.ippacket.stream*

### Logs

By default, the BIND9 server will log certain DNSSEC failures, and the journal log should be the first place to check.

For example, if we ask a BIND9 Validating Resolver for the IP address of the `www.dnssec-failed.org` name, we get a failure:

```
$ dig @127.0.0.1 -t A www.dnssec-failed.org
; <<>> DiG 9.18.28-0ubuntu0.24.04.1-Ubuntu <<>> @127.0.0.1 -t A www.dnssec-failed.
org
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 26260
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 6339d7228b8587f401000000671bc2eb2fe25bdf099ef1af (good)
;; QUESTION SECTION:
;www.dnssec-failed.org.         IN      A

;; Query time: 460 msec
;; SERVER: 127.0.0.1#53(127.0.0.1) (UDP)
;; WHEN: Fri Oct 25 16:10:19 UTC 2024
;; MSG SIZE  rcvd: 78
```

That's a very generic failure: it just says `SERVFAIL`, and gives us no IP: `IN A` is empty. The BIND9 logs, however, tell a more detailed story:

```
$ journalctl -u named.service -f
(...)
named[286]: validating dnssec-failed.org/DNSKEY: no valid signature found (DS)
named[286]: no valid RRSIG resolving 'dnssec-failed.org/DNSKEY/IN': 68.87.85.132
#53
named[286]: validating dnssec-failed.org/DNSKEY: no valid signature found (DS)
named[286]: no valid RRSIG resolving 'dnssec-failed.org/DNSKEY/IN': 68.87.68.244
#53
named[286]: validating dnssec-failed.org/DNSKEY: no valid signature found (DS)
named[286]: no valid RRSIG resolving 'dnssec-failed.org/DNSKEY/IN': 68.87.76.228
#53
named[286]: validating dnssec-failed.org/DNSKEY: no valid signature found (DS)
named[286]: no valid RRSIG resolving 'dnssec-failed.org/DNSKEY/IN': 68.87.72.244
#53
named[286]: validating dnssec-failed.org/DNSKEY: no valid signature found (DS)
named[286]: no valid RRSIG resolving 'dnssec-failed.org/DNSKEY/IN': 69.252.250.103
#53
named[286]: broken trust chain resolving 'www.dnssec-failed.org/A/IN': 68.87.72.
244#53
```

### Client-side tooling: dig

One of the more versatile DNS troubleshooting tools is `dig`, generally used for interrogating DNS name servers to lookup and display domain information, but its broad functionality makes it a flexible aid for DNS troubleshooting. It provides direct control over setting most of the DNS flags in queries, and displays detailed responses for inspection.

For DNSSEC troubleshooting purposes, we are interested in the following features:

- `+dnssec`: Set the "DNSSEC OK" bit in the queries, which tells the resolver to include in its responses the DNSSEC RRSIG records. This is also shown as a `do` flag in queries.
- `+cd`: This means *check disabled* and tells the resolver we can accept unauthenticated data in the DNS responses.
- `ad`: When included in a response, this flag means *authenticated data*, and tells us that the resolver who provided this answer has performed DNSSEC validation.
- `@<IP>`: This parameter lets us direct the query to a specific DNS server running at the provided IP address.

For example, let's query the local systemd-resolved DNS stub resolver (running at **127.0.0.53**) for the *isc.org* type *A* record, and request DNSSEC data:

```
$ dig @127.0.0.53 -t A +dnssec +multiline isc.org

; <<>> DiG 9.18.30-0ubuntu0.24.04.2-Ubuntu <<>> @127.0.0.53 -t A +dnssec
+multiline isc.org
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 13164
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 65494
;; QUESTION SECTION:
;isc.org.                IN A

;; ANSWER SECTION:
isc.org.                300 IN A 151.101.130.217
isc.org.                300 IN A 151.101.66.217
isc.org.                300 IN A 151.101.2.217
isc.org.                300 IN A 151.101.194.217
isc.org.                300 IN RRSIG A 13 2 300 (
                               20250830114017 20250816111944 27566 isc.org.
                               u6hKKZGX3DUD6JJAjHsGog+nfR9bz5qp2g1h3qibZI+A
                               qWH05fWJJjoSMzcnOgzIO1299zPIZd0xdMAh1wbkjw== )

;; Query time: 83 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Mon Aug 25 13:28:33 CEST 2025
;; MSG SIZE  rcvd: 203
```

Let's unpack this answer for the important troubleshooting parts:

- The answer has the `ad` flag set, meaning this data was authenticated. In other words, DNSSEC validation was successful.
- The status of the query is `NOERROR`, and we have 5 records in the answer section.
- An RRSIG record for the "A" Resource Record was returned as requested by the `+dnssec` command-line parameter. This is also confirmed by the presence of the "do" flag in the "OPT PSEUDOSECTION".

If we repeat this query with a domain that we know fails DNSSEC validation, we get the following reply:

```
$ dig @127.0.0.53 -t A +dnssec +multiline dnssec-failed.org

; <<>> DiG 9.18.30-0ubuntu0.24.04.2-Ubuntu <<>> @127.0.0.53 -t A +dnssec
+multiline dnssec-failed.org
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 8314
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 65494
;; QUESTION SECTION:
;dnssec-failed.org.        IN A

;; ANSWER SECTION:
dnssec-failed.org.        300 IN A 96.99.227.255

;; ADDITIONAL SECTION:
dnssec-failed.org.        300 IN RRSIG A 5 2 300 (
                              20250904145125 20250818144625 44973 dnssec-failed.
org.
                              UF75l9JkH/AZ9ApNF86stA81B+L36j0F/L8ENvMknsfK
                              Fwll6cLEJWBalKeyhK7p3U/Lqet9L3Oti8H7RudmgZ4v
                              kdYMDrb9mqnXscY1R/kmSrtu3gOO1ob+khKinyAVwKLb
                              R0CVZnJLSb7c++BI9fRJjE2caZ+2RkHwkLATR28= )

;; Query time: 138 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Mon Aug 25 13:37:11 CEST 2025
;; MSG SIZE  rcvd: 239
```

This time:

- There is no `ad` flag set in the answer.
- The status of the query is a generic `SERVFAIL`, and zero answers were provided.

We can tell the local stub resolver (systemd-resolved running at `@127.0.0.53` address) that we don't want it to perform DNSSEC validation. We do that by setting the `+cd` (check disabled) flag. Then things change in our answer:

```
$ dig @127.0.0.53 -t A +dnssec +cd +multiline dnssec-failed.org

; <<>> DiG 9.18.30-0ubuntu0.24.04.2-Ubuntu <<>> @127.0.0.53 -t A +dnssec +cd
+multiline dnssec-failed.org
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 55352
```

(continues on next page)

(continued from previous page)

```
;; flags: qr rd ra cd; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 65494
; EDE: 9 (DNSKEY Missing)
;; QUESTION SECTION:
;dnssec-failed.org.        IN A

;; ANSWER SECTION:
dnssec-failed.org.        181 IN A 96.99.227.255
dnssec-failed.org.        181 IN RRSIG A 5 2 300 (
                              20250904145125 20250818144625 44973 dnssec-failed.
org.
                              UF75l9JkH/AZ9ApNF86stA81B+L36j0F/L8ENvMknsfK
                              Fwll6cLEJWBalKeyhK7p3U/Lqet9L3Oti8H7RudmgZ4v
                              kdYMDrb9mqnXscY1R/kmSrtu3gOO1ob+khKinyAVwKLb
                              R0CVZnJLSb7c++BI9fRJjE2caZ+2RkHwkLATR28= )

;; Query time: 3 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Mon Aug 25 13:39:09 CEST 2025
;; MSG SIZE  rcvd: 245
```

Looks like we have some sort of answer, but:

- There is no `ad` flag in the answer, so this data was not authenticated.
- The status is `NOERROR`, and we got two answers.
- Note there is a `cd` flag in the answer, meaning "check disabled". No attempt at validating the answer was done by the resolver, as requested.

In none of these cases, though, did `dig` perform DNSSEC validation: it just presented the results provided by the resolver, which in some cases was no validation at all (via the `+cd` flag). To perform the validation ourselves, we have to use a different tool.

### Digging a bit deeper: `delv`

The `delv` tool is very similar to `dig`, and can perform the same DNS queries, but with a crucial difference: it can also validate DNSSEC. This is very useful in troubleshooting, because rather than returning a generic `SERVFAIL` error when something goes wrong with DNSSEC validation, it can tell us what was wrong in more detail.

But to bring the responsibility of doing DNSSEC validation to the tool itself, we use the `+cd` flag in our queries, to tell the resolver to not attempt that validation. Otherwise we will just get back a generic `SERVFAIL` error:

```
$ delv @127.0.0.53 -t A +dnssec +multiline dnssec-failed.org
;; resolution failed: SERVFAIL
```

With the `+cd` flag present, however, `delv` itself will do the validation. It will fail again, but now with a DNSSEC-specific error:

```
$ delv @127.0.0.53 -t A +dnssec +cd +multiline dnssec-failed.org
;; validating dnssec-failed.org/DNSKEY: no valid signature found (DS)
;; no valid RRSIG resolving 'dnssec-failed.org/DNSKEY/IN': 127.0.0.1#53
;; broken trust chain resolving 'dnssec-failed.org/A/IN': 127.0.0.1#53
;; resolution failed: broken trust chain
```

If needed, `delv` can be told to not perform DNSSEC validation at all, by passing the `-i` flag. Together with the `+cd` flag, which instructs the Validating Resolver to not perform validation either, we get this result:

```
$ delv @127.0.0.53 -i -t A +dnssec +cd +multiline dnssec-failed.org
; answer not validated
dnssec-failed.org.      100 IN A 96.99.227.255
```

For a good DNSSEC domain, `delv` will return a validated answer:

```
$ delv @127.0.0.53 -t A +multiline +cd isc.org
; fully validated
isc.org.                299 IN A 151.101.2.217
isc.org.                299 IN A 151.101.66.217
isc.org.                299 IN A 151.101.130.217
isc.org.                299 IN A 151.101.194.217
isc.org.                299 IN RRSIG A 13 2 300 (
                               20250908112301 20250825104017 27566 isc.org.
                               9oclTno0Ub2NmUEXdyLv0zqwPBbbUVmT3RX4aP4BQQ+h
                               4g839JXuCKHufXSPkWh/GJe/MveP83dDvJkrMmEIzg== )
```

Given that above we used the `+cd` flag, this means that the validation was done by `delv` itself. We will get the same result without that flag if the resolver also succeeds in the DNSSEC validation, and provides an answer.

### Client-side tooling: resolvectl

The local stub resolver *systemd-resolved* can perform DNSSEC validation locally using its high-level `resolvectl` tool. The local cache and state about DNS servers should be reset in systemd-resolved, to get reliable results.

Flush systemd-resolved caches & state and confirm DNSSEC is enabled:

```
$ sudo resolvectl flush-caches
$ sudo resolvectl reset-server-features
$ sudo resolvectl dnssec eth0 yes
$ resolvectl dnssec
Global: no
Link 44 (eth0): yes
```

Query a DNSSEC enabled domain and confirm the data was fetched from the network and authenticated, as displayed by the `Data is authenticated` and `Data from` fields:

```
$ resolvectl query --type=MX isc.org
isc.org IN MX 5 mx.pao1.isc.org                                -- link: eth0
isc.org IN MX 10 mx.ams1.isc.org                               -- link: eth0
```

(continues on next page)

(continued from previous page)

```
-- Information acquired via protocol DNS in 3.0ms.
-- Data is authenticated: yes; Data was acquired via local or encrypted transport:
no
-- Data from: network
```

### Incorrect time

As with everything related to cryptography, having an accurate measurement of time is of crucial importance. In a nutshell, digital signatures and keys have expiration dates.

An RRSIG record (a digital signature of a Resource Record) has a validity. For example, this record:

```
noble.example.internal. 86400 IN RRSIG A 13 3 86400 (
                                20241106131533 20241023195023 48112 example.
internal.
                                5fL4apIwCD9kt4XbzzlLxMXY3mj8Li1WZu3qzlcBpERp
                                lXPgLODbRrWyp7L81xEFnfhecKtEYv+6Y0Xa5iVRug== )
```

Has this validity range:

- valid until: 20241106131533 (2024-11-06 13:15:33 UTC)
- valid since: 20241023195023 (2024-10-23 19:50:23 UTC)

If the DNSSEC validator has an incorrect clock, outside of the validity range, the DNSSEC validation will fail. For example, with the clock incorrectly set to before the beginning of the validity period, `delv` will complain like this:

```
$ date
Tue Oct 10 10:10:19 UTC 2000

$ delv @10.10.17.229 -a example.internal.key +root=example.internal +multiline
noble.example.internal
;; validating example.internal/DNSKEY: verify failed due to bad signature
(keyid=48112): RRSIG validity period has not begun
;; validating example.internal/DNSKEY: no valid signature found (DS)
;; no valid RRSIG resolving 'example.internal/DNSKEY/IN': 10.10.17.229#53
;; broken trust chain resolving 'noble.example.internal/A/IN': 10.10.17.229#53
;; resolution failed: broken trust chain
```

Any other Validating Resolver will fail in a similar way, and should indicate this error in its logs.

BIND9 will complain loudly if it's running on a system with an incorrect clock, as the root zones will fail validation:

```
named[3593]: managed-keys-zone: DNSKEY set for zone '.' could not be verified with
current keys
named[3593]:   validating ./DNSKEY: verify failed due to bad signature
(keyid=20326): RRSIG validity period has not begun
```

(continues on next page)

(continued from previous page)

```
named[3593]:   validating ./DNSKEY: no valid signature found (DS)
named[3593]: broken trust chain resolving './NS/IN': 199.7.83.42#53
named[3593]: resolver priming query complete: broken trust chain
```

### Third-party Web-based diagnostics

There are some public third-party web-based tools that will check the status of DNSSEC of a public domain. Here are some:

- https://dnsviz.net: Returns a graphical diagram showing the chain of trust and where it breaks down, if that's the case.
- https://dnssec-debugger.verisignlabs.com: A DNSSEC debugger which also shows the chain of trust and where it breaks down, in a table format.

### Further reading

- bind9's guide to DNSSEC troubleshooting
- How to validate DNSSEC using the command line
- `delv(1)` manpage
- `dig(1)` manpage

Open vSwitch (OVS) with the Data Plane Development Kit (DPDK) provides virtual switching for network automation in virtualized environments.

### How to use Open vSwitch with DPDK

Since DPDK is *just* a library, it doesn't do a lot on its own so it depends on emerging projects making use of it. One consumer of the library that is already part of Ubuntu is Open vSwitch with DPDK (OvS-DPDK) support in the package `openvswitch-switch-dpdk`.

Here is a brief example of how to install and configure a basic Open vSwitch using DPDK for later use via `libvirt/qemu-kvm`.

```
sudo apt-get install openvswitch-switch-dpdk
sudo update-alternatives --set ovs-vswitchd /usr/lib/openvswitch-switch-dpdk/ovs-
vswitchd-dpdk
ovs-vsctl set Open_vSwitch . "other_config:dpdk-init=true"
# run on core 0 only
ovs-vsctl set Open_vSwitch . "other_config:dpdk-lcore-mask=0x1"
# Allocate 2G huge pages (not Numa node aware)
ovs-vsctl set Open_vSwitch . "other_config:dpdk-alloc-mem=2048"
# limit to one whitelisted device
ovs-vsctl set Open_vSwitch . "other_config:dpdk-extra=--pci-whitelist=0000:04:00.0
"
sudo service openvswitch-switch restart
```

**Note**

You need to assign devices to DPDK-compatible drivers before restarting – see the DPDK section on unassigning the default kernel drivers.

Please note that the section _dpdk-alloc-mem=2048_ in the above example is the most basic non-uniform memory access (NUMA) setup for a single socket system. If you have multiple sockets you may want to define how the memory should be split among them. More details about these options are outlined in Open vSwitch setup.

### Attach DPDK ports to Open vSwitch

The Open vSwitch you started above supports all the same port types as Open vSwitch usually does, *plus* DPDK port types. The following example shows how to create a bridge and – instead of a normal external port – add an external DPDK port to it. When doing so you can specify the associated device.

```
ovs-vsctl add-br ovsdpdkbr0 -- set bridge ovsdpdkbr0 datapath_type=netdev
ovs-vsctl add-port ovsdpdkbr0 dpdk0 -- set Interface dpdk0 type=dpdk
"options:dpdk-devargs=${OVSDEV_PCIID}"
```

You can tune this further by setting options:

```
ovs-vsctl set Interface dpdk0 "options:n_rxq=2"
```

### Open vSwitch DPDK to KVM guests

If you are not building some sort of software-defined networking (SDN) switch or NFV on top of DPDK, it is very likely that you want to forward traffic to KVM guests. The good news is; with the new `qemu/libvirt/dpdk/openvswitch` versions in Ubuntu this is no longer about manually appending a command line string. This section demonstrates a basic setup to connect a KVM guest to an Open vSwitch DPDK instance.

The recommended way to get to a KVM guest is using `vhost_user_client`. This will cause OvS-DPDK to connect to a socket created by QEMU. In this way, we can avoid old issues like "guest failures on OvS restart". Here is an example of how to add such a port to the bridge you created above.

```
ovs-vsctl add-port ovsdpdkbr0 vhost-user-1 -- set Interface vhost-user-1
type=dpdkvhostuserclient "options:vhost-server-path=/var/run/vhostuserclient/
vhost-user-client-1"
```

This will connect to the specified path that has to be created by a guest listening for it.

To let `libvirt/kvm` consume this socket and create a guest VirtIO network device for it, add the following snippet to your guest definition as the network definition.

```
<interface type='vhostuser'>
<source type='unix'
path='/var/run/vhostuserclient/vhost-user-client-1'
mode='server'/>
<model type='virtio'/>
</interface>
```

### Tuning Open vSwitch-DPDK

DPDK has plenty of options – in combination with Open vSwitch-DPDK the two most commonly used are:

```
ovs-vsctl set Open_vSwitch . other_config:n-dpdk-rxqs=2
ovs-vsctl set Open_vSwitch . other_config:pmd-cpu-mask=0x6
```

The first line selects how many Rx Queues are to be used for each DPDK interface, while the second controls how many poll mode driver (PMD) threads to run (and where to run them). The example above will use two Rx Queues, and run PMD threads on CPU 1 and 2.

> **See also**
>
> Check the links to *EAL* Command-line Options and "Open vSwitch DPDK installation" at the end of this document for more information.

As usual with tunings, you need to know your system and workload really well - so please verify any tunings with workloads matching your real use case.

### Support and troubleshooting

DPDK is a fast-evolving project. In any search for support and/or further guides, we highly recommended first checking to see if they apply to the current version.

You can check if your issues is known on:

- DPDK Mailing Lists
- For OpenVswitch-DPDK OpenStack Mailing Lists
- Known issues in DPDK Launchpad Area
- Join the IRC channels #DPDK or #openvswitch on *freenode*.

Issues are often due to missing small details in the general setup. Later on, these missing details cause problems which can be hard to track down to their root cause.

A common case seems to be the "could not open network device dpdk0 (No such device)" issue. This occurs rather late when setting up a port in Open vSwitch with DPDK, but the root cause (most of the time) is very early in the setup and initialisation. Here is an example of how proper initialiasation of a device looks - this can be found in the `syslog`/`journal` when starting Open vSwitch with DPDK enabled.

```
ovs-ctl[3560]: EAL: PCI device 0000:04:00.1 on NUMA socket 0
ovs-ctl[3560]: EAL:   probe driver: 8086:1528 rte_ixgbe_pmd
ovs-ctl[3560]: EAL:   PCI memory mapped at 0x7f2140000000
ovs-ctl[3560]: EAL:   PCI memory mapped at 0x7f2140200000
```

If this is missing, either by ignored cards, failed initialisation or other reasons, later on there will be no DPDK device to refer to. Unfortunately, the logging is spread across `syslog`/`journal` and the `openvswitch` log. To enable some cross-checking, here is an example of what can be found in these logs, relative to the entered command.

```
#Note: This log was taken with dpdk 2.2 and openvswitch 2.5 but still looks quite
similar (a bit extended) these days
Captions:
CMD: that you enter
SYSLOG: (Inlcuding EAL and OVS Messages)
OVS-LOG: (Openvswitch messages)

#PREPARATION
Bind an interface to DPDK UIO drivers, make Hugepages available, enable DPDK on
OVS

CMD: sudo service openvswitch-switch restart

SYSLOG:
2016-01-22T08:58:31.372Z|00003|daemon_unix(monitor)|INFO|pid 3329 died, killed
(Terminated), exiting
2016-01-22T08:58:33.377Z|00002|vlog|INFO|opened log file /var/log/openvswitch/ovs-
vswitchd.log
2016-01-22T08:58:33.381Z|00003|ovs_numa|INFO|Discovered 12 CPU cores on NUMA node
0
2016-01-22T08:58:33.381Z|00004|ovs_numa|INFO|Discovered 1 NUMA nodes and 12 CPU
cores
2016-01-22T08:58:33.381Z|00005|reconnect|INFO|unix:/var/run/openvswitch/db.sock:
connecting...
2016-01-22T08:58:33.383Z|00006|reconnect|INFO|unix:/var/run/openvswitch/db.sock:
connected
2016-01-22T08:58:33.386Z|00007|bridge|INFO|ovs-vswitchd (Open vSwitch) 2.5.0

OVS-LOG:
systemd[1]: Stopping Open vSwitch...
systemd[1]: Stopped Open vSwitch.
systemd[1]: Stopping Open vSwitch Internal Unit...
ovs-ctl[3541]: * Killing ovs-vswitchd (3329)
ovs-ctl[3541]: * Killing ovsdb-server (3318)
systemd[1]: Stopped Open vSwitch Internal Unit.
systemd[1]: Starting Open vSwitch Internal Unit...
ovs-ctl[3560]: * Starting ovsdb-server
ovs-vsctl: ovs|00001|vsctl|INFO|Called as ovs-vsctl --no-wait -- init -- set Open_
vSwitch . db-version=7.12.1
ovs-vsctl: ovs|00001|vsctl|INFO|Called as ovs-vsctl --no-wait set Open_vSwitch .
ovs-version=2.5.0 "external-ids:system-id=\"e7c5ba80-bb14-45c1-b8eb-628f3ad03903\"
" "system-type=\"Ubuntu\"" "system-version=\"16.04-xenial\""
ovs-ctl[3560]: * Configuring Open vSwitch system IDs
ovs-ctl[3560]: 2016-01-22T08:58:31Z|00001|dpdk|INFO|No -vhost_sock_dir provided -
defaulting to /var/run/openvswitch
ovs-vswitchd: ovs|00001|dpdk|INFO|No -vhost_sock_dir provided - defaulting to /
var/run/openvswitch
ovs-ctl[3560]: EAL: Detected lcore 0 as core 0 on socket 0
ovs-ctl[3560]: EAL: Detected lcore 1 as core 1 on socket 0
ovs-ctl[3560]: EAL: Detected lcore 2 as core 2 on socket 0
```

(continues on next page)

(continued from previous page)

```
ovs-ctl[3560]: EAL: Detected lcore 3 as core 3 on socket 0
ovs-ctl[3560]: EAL: Detected lcore 4 as core 4 on socket 0
ovs-ctl[3560]: EAL: Detected lcore 5 as core 5 on socket 0
ovs-ctl[3560]: EAL: Detected lcore 6 as core 0 on socket 0
ovs-ctl[3560]: EAL: Detected lcore 7 as core 1 on socket 0
ovs-ctl[3560]: EAL: Detected lcore 8 as core 2 on socket 0
ovs-ctl[3560]: EAL: Detected lcore 9 as core 3 on socket 0
ovs-ctl[3560]: EAL: Detected lcore 10 as core 4 on socket 0
ovs-ctl[3560]: EAL: Detected lcore 11 as core 5 on socket 0
ovs-ctl[3560]: EAL: Support maximum 128 logical core(s) by configuration.
ovs-ctl[3560]: EAL: Detected 12 lcore(s)
ovs-ctl[3560]: EAL: VFIO modules not all loaded, skip VFIO support...
ovs-ctl[3560]: EAL: Setting up physically contiguous memory...
ovs-ctl[3560]: EAL: Ask a virtual area of 0x100000000 bytes
ovs-ctl[3560]: EAL: Virtual area found at 0x7f2040000000 (size = 0x100000000)
ovs-ctl[3560]: EAL: Requesting 4 pages of size 1024MB from socket 0
ovs-ctl[3560]: EAL: TSC frequency is ~2397202 KHz
ovs-vswitchd[3592]: EAL: TSC frequency is ~2397202 KHz
ovs-vswitchd[3592]: EAL: Master lcore 0 is ready (tid=fc6cbb00;cpuset=[0])
ovs-vswitchd[3592]: EAL: PCI device 0000:04:00.0 on NUMA socket 0
ovs-vswitchd[3592]: EAL:   probe driver: 8086:1528 rte_ixgbe_pmd
ovs-vswitchd[3592]: EAL:   Not managed by a supported kernel driver, skipped
ovs-vswitchd[3592]: EAL: PCI device 0000:04:00.1 on NUMA socket 0
ovs-vswitchd[3592]: EAL:   probe driver: 8086:1528 rte_ixgbe_pmd
ovs-vswitchd[3592]: EAL:   PCI memory mapped at 0x7f2140000000
ovs-vswitchd[3592]: EAL:   PCI memory mapped at 0x7f2140200000
ovs-ctl[3560]: EAL: Master lcore 0 is ready (tid=fc6cbb00;cpuset=[0])
ovs-ctl[3560]: EAL: PCI device 0000:04:00.0 on NUMA socket 0
ovs-ctl[3560]: EAL:   probe driver: 8086:1528 rte_ixgbe_pmd
ovs-ctl[3560]: EAL:   Not managed by a supported kernel driver, skipped
ovs-ctl[3560]: EAL: PCI device 0000:04:00.1 on NUMA socket 0
ovs-ctl[3560]: EAL:   probe driver: 8086:1528 rte_ixgbe_pmd
ovs-ctl[3560]: EAL:   PCI memory mapped at 0x7f2140000000
ovs-ctl[3560]: EAL:   PCI memory mapped at 0x7f2140200000
ovs-vswitchd[3592]: PMD: eth_ixgbe_dev_init(): MAC: 4, PHY: 3
ovs-vswitchd[3592]: PMD: eth_ixgbe_dev_init(): port 0 vendorID=0x8086
deviceID=0x1528
ovs-ctl[3560]: PMD: eth_ixgbe_dev_init(): MAC: 4, PHY: 3
ovs-ctl[3560]: PMD: eth_ixgbe_dev_init(): port 0 vendorID=0x8086 deviceID=0x1528
ovs-ctl[3560]: Zone 0: name:<RG_MP_log_history>, phys:0x83fffdec0, len:0x2080,
virt:0x7f213fffdec0, socket_id:0, flags:0
ovs-ctl[3560]: Zone 1: name:<MP_log_history>, phys:0x83fd73d40, len:0x28a0c0,
virt:0x7f213fd73d40, socket_id:0, flags:0
ovs-ctl[3560]: Zone 2: name:<rte_eth_dev_data>, phys:0x83fd43380, len:0x2f700,
virt:0x7f213fd43380, socket_id:0, flags:0
ovs-ctl[3560]: * Starting ovs-vswitchd
ovs-ctl[3560]: * Enabling remote OVSDB managers
systemd[1]: Started Open vSwitch Internal Unit.
systemd[1]: Starting Open vSwitch...
```

(continues on next page)

(continued from previous page)

```
systemd[1]: Started Open vSwitch.


CMD: sudo ovs-vsctl add-br ovsdpdkbr0 -- set bridge ovsdpdkbr0 datapath_
type=netdev

SYSLOG:
2016-01-22T08:58:56.344Z|00008|memory|INFO|37256 kB peak resident set size after
24.5 seconds
2016-01-22T08:58:56.346Z|00009|ofproto_dpif|INFO|netdev@ovs-netdev: Datapath
supports recirculation
2016-01-22T08:58:56.346Z|00010|ofproto_dpif|INFO|netdev@ovs-netdev: MPLS label
stack length probed as 3
2016-01-22T08:58:56.346Z|00011|ofproto_dpif|INFO|netdev@ovs-netdev: Datapath
supports unique flow ids
2016-01-22T08:58:56.346Z|00012|ofproto_dpif|INFO|netdev@ovs-netdev: Datapath does
not support ct_state
2016-01-22T08:58:56.346Z|00013|ofproto_dpif|INFO|netdev@ovs-netdev: Datapath does
not support ct_zone
2016-01-22T08:58:56.346Z|00014|ofproto_dpif|INFO|netdev@ovs-netdev: Datapath does
not support ct_mark
2016-01-22T08:58:56.346Z|00015|ofproto_dpif|INFO|netdev@ovs-netdev: Datapath does
not support ct_label
2016-01-22T08:58:56.360Z|00016|bridge|INFO|bridge ovsdpdkbr0: added interface
ovsdpdkbr0 on port 65534
2016-01-22T08:58:56.361Z|00017|bridge|INFO|bridge ovsdpdkbr0: using datapath ID
00005a4a1ed0a14d
2016-01-22T08:58:56.361Z|00018|connmgr|INFO|ovsdpdkbr0: added service controller
"punix:/var/run/openvswitch/ovsdpdkbr0.mgmt"

OVS-LOG:
ovs-vsctl: ovs|00001|vsctl|INFO|Called as ovs-vsctl add-br ovsdpdkbr0 -- set
bridge ovsdpdkbr0 datapath_type=netdev
systemd-udevd[3607]: Could not generate persistent MAC address for ovs-netdev: No
such file or directory
kernel: [50165.886554] device ovs-netdev entered promiscuous mode
kernel: [50165.901261] device ovsdpdkbr0 entered promiscuous mode


CMD: sudo ovs-vsctl add-port ovsdpdkbr0 dpdk0 -- set Interface dpdk0 type=dpdk

SYSLOG:
2016-01-22T08:59:06.369Z|00019|memory|INFO|peak resident set size grew 155% in
last 10.0 seconds, from 37256 kB to 95008 kB
2016-01-22T08:59:06.369Z|00020|memory|INFO|handlers:4 ports:1 revalidators:2
rules:5
2016-01-22T08:59:30.989Z|00021|dpdk|INFO|Port 0: 8c:dc:d4:b3:6d:e9
2016-01-22T08:59:31.520Z|00022|dpdk|INFO|Port 0: 8c:dc:d4:b3:6d:e9
2016-01-22T08:59:31.521Z|00023|dpif_netdev|INFO|Created 1 pmd threads on numa node
```

(continues on next page)

(continued from previous page)

```
0
2016-01-22T08:59:31.522Z|00001|dpif_netdev(pmd16)|INFO|Core 0 processing port
'dpdk0'
2016-01-22T08:59:31.522Z|00024|bridge|INFO|bridge ovsdpdkbr0: added interface
dpdk0 on port 1
2016-01-22T08:59:31.522Z|00025|bridge|INFO|bridge ovsdpdkbr0: using datapath ID
00008cdcd4b36de9
2016-01-22T08:59:31.523Z|00002|dpif_netdev(pmd16)|INFO|Core 0 processing port
'dpdk0'

OVS-LOG:
ovs-vsctl: ovs|00001|vsctl|INFO|Called as ovs-vsctl add-port ovsdpdkbr0 dpdk0 --
set Interface dpdk0 type=dpdk
ovs-vswitchd[3595]: PMD: ixgbe_dev_tx_queue_setup(): sw_ring=0x7f211a79ebc0 hw_
ring=0x7f211a7a6c00 dma_addr=0x81a7a6c00
ovs-vswitchd[3595]: PMD: ixgbe_set_tx_function(): Using simple tx code path
ovs-vswitchd[3595]: PMD: ixgbe_set_tx_function(): Vector tx enabled.
ovs-vswitchd[3595]: PMD: ixgbe_dev_rx_queue_setup(): sw_ring=0x7f211a78a6c0 sw_sc_
ring=0x7f211a786580 hw_ring=0x7f211a78e800 dma_addr=0x81a78e800
ovs-vswitchd[3595]: PMD: ixgbe_set_rx_function(): Vector rx enabled, please make
sure RX burst size no less than 4 (port=0).
ovs-vswitchd[3595]: PMD: ixgbe_dev_tx_queue_setup(): sw_ring=0x7f211a79ebc0 hw_
ring=0x7f211a7a6c00 dma_addr=0x81a7a6c00
...


CMD: sudo ovs-vsctl add-port ovsdpdkbr0 vhost-user-1 -- set Interface vhost-user-1
type=dpdkvhostuser

OVS-LOG:
2016-01-22T09:00:35.145Z|00026|dpdk|INFO|Socket /var/run/openvswitch/vhost-user-1
created for vhost-user port vhost-user-1
2016-01-22T09:00:35.145Z|00003|dpif_netdev(pmd16)|INFO|Core 0 processing port
'dpdk0'
2016-01-22T09:00:35.145Z|00004|dpif_netdev(pmd16)|INFO|Core 0 processing port
'vhost-user-1'
2016-01-22T09:00:35.145Z|00027|bridge|INFO|bridge ovsdpdkbr0: added interface
vhost-user-1 on port 2

SYSLOG:
ovs-vsctl: ovs|00001|vsctl|INFO|Called as ovs-vsctl add-port ovsdpdkbr0 vhost-
user-1 -- set Interface vhost-user-1 type=dpdkvhostuser
ovs-vswitchd[3595]: VHOST_CONFIG: socket created, fd:46
ovs-vswitchd[3595]: VHOST_CONFIG: bind to /var/run/openvswitch/vhost-user-1

Eventually we can see the poll thread in top
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 3595 root      10 -10 4975344 103936   9916 S 100.0  0.3  33:13.56 ovs-vswitchd
```

### Resources

- DPDK documentation
- Release Notes matching the version packages in Ubuntu 16.04
- Linux DPDK user getting started
- EAL command-line options
- DPDK API documentation
- Open Vswitch DPDK installation
- Wikipedia's definition of DPDK

## DHCP

Set up Dynamic Host Configuration Protocol (DHCP) for automatic IP address assignment for devices on your network. There are two DHCP servers available in Ubuntu: `isc-kea` is the most modern, and is available from 23.04 onwards.

### How to install and configure isc-kea

In this guide we show how to install and configure `isc-kea` in Ubuntu 23.04 or greater. Kea is the *DHCP* server developed by ISC to replace `isc-dhcp`. It is newer and designed for more modern network environments.

For `isc-dhcp-server` instructions, *refer to this guide instead*.

### Install isc-kea

At a terminal prompt, enter the following command to install `isc-kea`:

```
sudo apt install kea
```

This will also install a few binary packages, including

- `kea-dhcp4-server`: The IPv4 DHCP server (the one we will configure in this guide).
- `kea-dhcp6-server`: The IPv6 DHCP server.
- `kea-ctrl-agent`: A REST API service for Kea.
- `kea-dhcp-ddns-server`: A Dynamic *DNS* service to update DNS based on DHCP lease events.

Since the `kea-ctrl-agent` service has some administrative rights to the Kea services, we need to ensure regular users are not allowed to use the API without permissions. Ubuntu does it by requiring user authentication to access the `kea-ctrl-agent` API service (LP: #2007312 has more details on this).

Therefore, the installation process described above will get a *debconf* "high" priority prompt with 3 options:

- no action (default);
- configure with a random password; or
- configure with a given password.

If there is no password, the `kea-ctrl-agent` will **not** start.

The password is expected to be in `/etc/kea/kea-api-password`, with ownership `root:_kea` and permissions `0640`. To change it, run `dpkg-reconfigure kea-ctrl-agent` (which will present the same 3 options from above again), or just edit the file manually.

### Configure kea-dhcp4

The `kea-dhcp4` service can be configured by editing `/etc/kea/kea-dhcp4.conf`.

Most commonly, what you want to do is let Kea assign an IP address from a pre-configured IP address pool. This can be done with settings as follows:

```
{
    "Dhcp4": {
        "interfaces-config": {
            "interfaces": ["eth4"]
        },
        "control-socket": {
            "socket-type": "unix",
            "socket-name": "/run/kea/kea4-ctrl-socket"
        },
        "lease-database": {
            "type": "memfile",
            "lfc-interval": 3600
        },
        "valid-lifetime": 600,
        "max-valid-lifetime": 7200,
        "subnet4": [{
            "id": 1,
            "subnet": "192.168.1.0/24",
            "pools": [{
                "pool": "192.168.1.150 - 192.168.1.200"
            }],
            "option-data": [{
                    "name": "routers",
                    "data": "192.168.1.254"
                },
                {
                    "name": "domain-name-servers",
                    "data": "192.168.1.1, 192.168.1.2"
                },
                {
                    "name": "domain-name",
                    "data": "mydomain.example"
                }
            ]
        }]
    }
}
```

This will result in the DHCP server listening on interface “eth4”, giving clients an IP address from the range `192.168.1.150 - 192.168.1.200`. It will lease an IP address for 600 seconds

if the client doesn't ask for a specific time frame. Otherwise the maximum (allowed) lease will be 7200 seconds. The server will also "advise" the client to use `192.168.1.254` as the default-gateway and `192.168.1.1` and `192.168.1.2` as its DNS servers.

After changing the config file you can reload the server configuration through `kea-shell` with the following command (considering you have the `kea-ctrl-agent` running as described above):

```
kea-shell --host 127.0.0.1 --port 8000 --auth-user kea-api --auth-password $(cat /
etc/kea/kea-api-password) --service dhcp4 config-reload
```

Then, press ctrl-d. The server should respond with:

```
[ { "result": 0, "text": "Configuration successful." } ]
```

meaning your configuration was received by the server.

The `kea-dhcp4-server` service logs should contain an entry similar to:

```
DHCP4_DYNAMIC_RECONFIGURATION_SUCCESS dynamic server reconfiguration succeeded
with file: /etc/kea/kea-dhcp4.conf
```

signaling that the server was successfully reconfigured.

You can read `kea-dhcp4-server` service logs with `journalctl`:

```
journalctl -u kea-dhcp4-server
```

Alternatively, instead of reloading the DHCP4 server configuration through `kea-shell`, you can restart the `kea-dhcp4-service` with:

```
systemctl restart kea-dhcp4-server
```

### Further reading

- ISC Kea Documentation

### How to install and configure isc-dhcp-server

> **Note**
>
> Although Ubuntu still supports `isc-dhcp-server`, this software is no longer supported by its vendor. It has been replaced by Kea.

In this guide we show how to install and configure `isc-dhcp-server`, which installs the dynamic host configuration protocol daemon, *DHCPD*. For `isc-kea` instructions, *refer to this guide instead*.

### Install isc-dhcp-server

At a terminal prompt, enter the following command to install `isc-dhcp-server`:

```
sudo apt install isc-dhcp-server
```

> **Note**
>
> You can find diagnostic messages from `dhcpd` in `syslog`.

### Configure isc-dhcp-server

You will probably need to change the default configuration by editing `/etc/dhcp/dhcpd.conf` to suit your needs and particular configuration.

Most commonly, what you want to do is assign an IP address randomly. This can be done with `/etc/dhcp/dhcpd.conf` settings as follows:

```
# minimal sample /etc/dhcp/dhcpd.conf
default-lease-time 600;
max-lease-time 7200;

subnet 192.168.1.0 netmask 255.255.255.0 {
 range 192.168.1.150 192.168.1.200;
 option routers 192.168.1.254;
 option domain-name-servers 192.168.1.1, 192.168.1.2;
 option domain-name "mydomain.example";
}
```

This will result in the DHCP server giving clients an IP address from the range `192.168.1.150 - 192.168.1.200`. It will lease an IP address for 600 seconds if the client doesn't ask for a specific time frame. Otherwise the maximum (allowed) lease will be 7200 seconds. The server will also "advise" the client to use `192.168.1.254` as the default-gateway and `192.168.1.1` and `192.168.1.2` as its *DNS* servers.

You also may need to edit `/etc/default/isc-dhcp-server` to specify the interfaces `dhcpd` should listen to.

In the example below, `eth4` is used, but you should replace this with the appropriate interface for your system. The name of the network interface can vary depending on your setup. For instance, it could be `eth0`, `ens33`, or any other name depending on the device you're using.

```
INTERFACESv4="eth4"
```

After changing the config files you need to restart the `dhcpd` service:

```
sudo systemctl restart isc-dhcp-server.service
```

### Further reading

- The isc-dhcp-server Ubuntu Wiki page has more information.
- For more `/etc/dhcp/dhcpd.conf` options see the `dhcpd.conf(5)` manual page
- ISC dhcp-server

### Time synchronization

The Network Time Protocol (NTP) synchronizes time over a network. Ubuntu uses `chrony` by default to handle this. However, users can install and use `timedatectl`/`timesyncd` instead if preferred.

### Synchronize time using Chrony

Ubuntu uses `chrony` for synchronizing time, which is installed by default as of Ubuntu 25.10. You can optionally use `timedatectl`/`timesyncd` to *synchronize time using systemd*.

### Check status of `chrony` client

The current status of time synchronization can be checked with the `timedatectl status` command, which is available via `systemd` and will produce output like this:

```
               Local time: Mo 2025-06-16 15:21:46 CEST
           Universal time: Mo 2025-06-16 13:21:46 UTC
                 RTC time: Mo 2025-06-16 13:21:46
                Time zone: Europe/Berlin (CEST, +0200)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

For more details on time accuracy, Chrony can be queried directly, using the `chronyc -N tracking` command, producing output like this:

```
Reference ID    : B97DBE7B (2.ntp.ubuntu.com)
Stratum         : 3
Ref time (UTC)  : Mon Jun 16 13:06:04 2025
System time     : 0.000000004 seconds slow of NTP time
Last offset     : +0.001758954 seconds
RMS offset      : 0.017604901 seconds
Frequency       : 3.889 ppm slow
Residual freq   : +0.202 ppm
Skew            : 1.458 ppm
Root delay      : 0.022837413 seconds
Root dispersion : 0.003050051 seconds
Update interval : 1032.5 seconds
Leap status     : Normal
```

### Network Time Security (NTS)

Chrony supports "Network Time Security" (NTS) and enables it by default, using the Ubuntu NTS pools. This is done by specifying a `server` or `pool` as usual. Afterwards, options can be listed and it is there that `nts` can be added. For example:

```
server <server-fqdn-or-IP> iburst nts
# or as concrete example
pool 1.ntp.ubuntu.com iburst maxsources 1 nts prefer
```

For **validation of NTS enablement**, one can list the time sources in use by executing the `chronyc -N sources` command, to find the timeserver in use, as indicated by the `^*` symbol in the first column. Then check the `authdata` of that connection using `sudo chronyc -N authdata`. If the client was able to successfully establish a NTS connection, it will show the `Mode: NTS` field and non-zero values for `KeyID`, `Type` and `KLen`:

```
$ sudo chronyc -N authdata
Name/IP address             Mode KeyID Type KLen Last Atmp  NAK Cook CLen
==============================================================================
1.ntp.ubuntu.com             NTS     6   30  128  14d    0    0    8   64
2.ntp.ubuntu.com             NTS     6   30  128  14d    0    0    8   64
3.ntp.ubuntu.com             NTS     1   30  128  27d    0    0    8   64
4.ntp.ubuntu.com             NTS     2   30  128  20d    0    0    5   64
ntp-bootstrap.ubuntu.com     NTS     3   30  128   7d    0    0    8   64
```

### NTS related constraints

**Key Exchange port:** NTS/KE uses a separate port (4460/tcp)** to negotiate security parameters, which are then used via the normal NTP port (123/udp). This is a new deployment, running on different IP addresses than the traditional Ubuntu NTP pool.

> **Warning**
>
> If the network does not allow access to the Ubuntu NTS servers and required ports, with the default configuration is in place, `chrony` will not be able to adjust the system's clock. To revert to NTP, edit the configuration file in `/etc/chrony/sources.d/ubuntu-ntp-pools.sources` and revert to using the listed NTP servers in favor of the NTS ones.

**Bad Clocks and secure time syncing:** NTS is based on TLS, and TLS needs a reasonably correct clock. Due to that, an NTS-based sync might fail if the clock is too far off. On hardware affected by this problem, one can consider using the `nocerttimecheck` option, which allows to set the number of times that the time can be synced without checking validation and expiration.

> **Note**
>
> A new CA is installed in `/etc/chrony/nts-bootstrap-ubuntu.crt` that is used specifically for the Ubuntu NTS bootstrap server, needed for when the clock is too far off. This is added to certificate set ID "1", and defined via `/etc/chrony/conf.d/ubuntu-nts.conf`.

### Configure Chrony

An admin can control the timezone and how the system clock should relate to the `hwclock` using the common `timedatectl [set-timezone/set-local-rtc]` commands, provided by `systemd`. For more specific actions, like adding of time-sources, the `chronyc` command can be used. See `man chronyc` for more details.

One can edit configuration in `/etc/chrony/sources.d/` to add/remove server lines. By default these servers are configured:

```
# Use NTS by default
# NTS uses an additional port to negotiate security: 4460/tcp
# The normal NTP port remains in use: 123/udp
pool 1.ntp.ubuntu.com iburst maxsources 1 nts prefer
pool 2.ntp.ubuntu.com iburst maxsources 1 nts prefer
pool 3.ntp.ubuntu.com iburst maxsources 1 nts prefer
pool 4.ntp.ubuntu.com iburst maxsources 1 nts prefer
# The bootstrap server is needed by systems without a hardware clock, or a very
# large initial clock offset. The specified certificate set is defined in
# /etc/chrony/conf.d/ubuntu-nts.conf.
pool ntp-bootstrap.ubuntu.com iburst maxsources 1 nts certset 1
```

After adding or removing sources, they can be reloaded using `sudo chrony reload sources`.

Of the pool, `2.ubuntu.pool.ntp.org` and `ntp.ubuntu.com` also support IPv6, if needed. If you need to force IPv6, there is also `ipv6.ntp.ubuntu.com` which is not configured by default.

### Time sources provided by DHCP (option 42)

Chrony consumes time sources provided by DHCP (option 42). Those could be traditional, non-authenticated NTP sources. Should one want to avoid this behavior, overruling the choices made by a local DHCP administrator, it can be disabled in `/etc/chrony/chrony.conf`. To do that one would comment out the corresponding setting:

```
# Use time sources from DHCP.
# sourcedir /run/chrony-dhcp
```

### Chrony time-daemon

`chronyd` itself is a normal service, so you can check its status in more detail using:

```
systemctl status chrony.service
```

The output produced will look something like this:

```
● chrony.service - chrony, an NTP client/server
     Loaded: loaded (/usr/lib/systemd/system/chrony.service; enabled; preset:
enabled)
     Active: active (running) since Mon 2025-06-02 11:27:09 CEST; 2 weeks 0 days
ago
       Docs: man:chronyd(8)
             man:chronyc(1)
```

(continues on next page)

(continued from previous page)

```
             man:chrony.conf(5)
   Main PID: 36027 (chronyd)
      Tasks: 2 (limit: 28388)
     Memory: 5.8M (peak: 6.8M swap: 604.0K swap peak: 4.5M)
        CPU: 5.038s
     CGroup: /system.slice/chrony.service
             ├─36027 /usr/sbin/chronyd -F 1
             └─36028 /usr/sbin/chronyd -F 1

    Jun 02 11:27:09 questing chronyd[36027]: Using right/UTC timezone to obtain
leap second data
    Jun 02 11:27:09 questing chronyd[36027]: Loaded seccomp filter (level 1)
    Jun 02 11:27:09 questing chronyd[36027]: Added pool 1.ntp.ubuntu.com
    Jun 02 11:27:09 questing chronyd[36027]: Added pool 2.ntp.ubuntu.com
    Jun 02 11:27:09 questing chronyd[36027]: Added pool 3.ntp.ubuntu.com
    Jun 02 11:27:09 questing chronyd[36027]: Added pool 4.ntp.ubuntu.com
    Jun 02 11:27:09 questing chronyd[36027]: Added pool ntp-bootstrap.ubuntu.com
    Jun 02 11:27:09 questing systemd[1]: Started chrony.service - chrony, an NTP
client/server.
```

Default configuration such as `sourcedir`, `ntsdumpdir` or `rtcsync` is provided in `/etc/chrony/chrony.conf` and additional config files can be stored in `/etc/chrony/conf.d/`. The NTS servers from which to fetch time for `chrony` are in defined in `/etc/chrony/sources.d/ubuntu-ntp-pools.sources`. There are more advanced options documented in the `chrony.conf(5)` manpage. Common use cases are specifying an explicit trusted certificate. After changing any part of the config file you need to restart `chrony`, as follows:

```
sudo systemctl restart chrony.service
```

### Next steps

If you would now like to also serve the Network Time Protocol via `chrony`, this guide will walk you through *how to configure your Chrony setup*.

### References

- `chronyc(1)` manual page
- `chronyd(8)` manual page
- `chrony.conf(5)` manual page

### Synchronise time using timedatectl and timesyncd

Ubuntu can use `timedatectl` and `timesyncd` for synchronising time, which can be installed as follows. `systemd-timesyncd` used to be part of the default installation, but was replaced by `chrony` since Ubuntu 25.10. You can optionally use `chrony` as a *Network Time Security (NTS) client* or to *serve the Network Time Protocol*.

```
sudo apt-mark auto chrony && apt install systemd-timesyncd
```

In this guide, we will show you how to configure these services.

> **Note**
>
> If `chrony` is installed, `timedatectl` steps back to let `chrony` handle timekeeping. This ensures that no two time-syncing services will be in conflict.

### Check status of `timedatectl`

The current status of time and time configuration via `timedatectl` and `timesyncd` can be checked with the `timedatectl status` command, which will produce output like this:

```
               Local time: Wed 2023-06-14 12:05:11 BST
           Universal time: Wed 2023-06-14 11:05:11 UTC
                 RTC time: Wed 2023-06-14 11:05:11
                Time zone: Europe/Isle_of_Man (BST, +0100)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

If `chrony` is running, it will automatically switch to:

```
[...]
 systemd-timesyncd.service active: no
```

### Configure `timedatectl`

By using `timedatectl`, an admin can control the timezone, how the system clock should relate to the `hwclock` and whether permanent synchronisation should be enabled. See `man timedatectl` for more details.

### Check status of `timesyncd`

`timesyncd` itself is a normal service, so you can check its status in more detail using:

```
systemctl status systemd-timesyncd
```

The output produced will look something like this:

```
      systemd-timesyncd.service - Network Time Synchronization
       Loaded: loaded (/lib/systemd/system/systemd-timesyncd.service; enabled;
vendor preset: enabled)
       Active: active (running) since Fri 2018-02-23 08:55:46 UTC; 10s ago
         Docs: man:systemd-timesyncd.service(8)
     Main PID: 3744 (systemd-timesyn)
       Status: "Synchronized to time server 91.189.89.198:123 (ntp.ubuntu.com)."
        Tasks: 2 (limit: 4915)
       CGroup: /system.slice/systemd-timesyncd.service
               |-3744 /lib/systemd/systemd-timesyncd
```

(continues on next page)

(continued from previous page)

```
    Feb 23 08:55:46 bionic-test systemd[1]: Starting Network Time Synchronization.
..
    Feb 23 08:55:46 bionic-test systemd[1]: Started Network Time Synchronization.
    Feb 23 08:55:46 bionic-test systemd-timesyncd[3744]: Synchronized to time
server 91.189.89.198:123 (ntp.ubuntu.com).
```

### Configure `timesyncd`

The server from which to fetch time for `timedatectl` and `timesyncd` can be specified in `/etc/systemd/timesyncd.conf`. Additional config files can be stored in `/etc/systemd/timesyncd.conf.d/`. The entries for `NTP=` and `FallbackNTP=` are space-separated lists. See `man timesyncd.conf` for more details.

### Next steps

If you would now like to serve the Network Time Protocol via `chrony`, this guide will walk you through *how to install and configure your setup*.

### References

- Freedesktop.org info on timedatectl
- Freedesktop.org info on systemd-timesyncd service
- See the Ubuntu Time wiki page for more information.

## How to serve the Network Time Protocol with Chrony

As of Ubuntu 25.10 `chrony` is installed by default, *functioning as a client* and keeping your time in sync. However, if you also want to serve NTP information then you need an NTP server.

Between `chrony`, the now-deprecated `ntpd`, and `open-ntp`, there are plenty of options. The solution we recommend is `chrony`.

The NTP daemon `chronyd` calculates the drift and offset of your system clock and continuously adjusts it, so there are no large corrections that could lead to inconsistent logs, for instance. The cost is a little processing power and memory, but for a modern server this is usually negligible.

### Install `chronyd`

To install `chrony`, run the following command from a terminal prompt:

```
sudo apt install chrony
```

This will provide two binaries:

- `chronyd` - the actual daemon to sync and serve via the Network Time Protocol
- `chronyc` - command-line interface for the `chrony` daemon

### Enable serving the Network Time Protocol

You can install `chrony` (above) and configure special Hardware (below) for a local synchronisation and as-installed that is the default to stay on the secure and conservative side. But if you want to *serve* NTP you need adapt your configuration.

To enable serving NTP you'll need to at least set the `allow` rule. This controls which clients/networks you want `chrony` to serve NTP to.

An example would be:

```
allow 1.2.3.4
```

See the section "NTP server" in the `chrony.conf(5)` manual page for more details on how you can control and restrict access to your NTP server.

### View `chrony` status

You can use `chronyc` to see query the status of the `chrony` daemon. For example, to get an overview of the currently available and selected time sources, run `chronyc sources`, which provides output like this:

```
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^+ gamma.rueckgr.at              2   8   377   135  -1048us[-1048us] +/-   29ms
^- 2b.ncomputers.org             2   8   377   204  -1141us[-1124us] +/-   50ms
^+ www.kashra.com                2   8   377   139  +3483us[+3483us] +/-   18ms
^+ stratum2-4.NTP.TechFak.U>     2   8   377   143  -2090us[-2073us] +/-   19ms
^- zepto.mcl.gg                  2   7   377     9   -774us[ -774us] +/-   29ms
^- mirrorhost.pw                 2   7   377    78   -660us[ -660us] +/-   53ms
^- atto.mcl.gg                   2   7   377     8   -823us[ -823us] +/-   50ms
^- static.140.107.46.78.cli>     2   8   377     9  -1503us[-1503us] +/-   45ms
^- 4.53.160.75                   2   8   377   137    -11ms[  -11ms] +/-  117ms
^- 37.44.185.42                  3   7   377    10  -3274us[-3274us] +/-   70ms
^- bagnikita.com                 2   7   377    74  +3131us[+3131us] +/-   71ms
^- europa.ellipse.net            2   8   377   204   -790us[ -773us] +/-   97ms
^- tethys.hot-chilli.net         2   8   377   141   -797us[ -797us] +/-   59ms
^- 66-232-97-8.static.hvvc.>     2   7   377   206  +1669us[+1686us] +/-  133ms
^+ 85.199.214.102                1   8   377   205   +175us[ +192us] +/-   12ms
^* 46-243-26-34.tangos.nl        1   8   377   141   -123us[ -106us] +/-   10ms
^- pugot.canonical.com           2   8   377    21    -95us[  -95us] +/-   57ms
^- alphyn.canonical.com          2   6   377    23  -1569us[-1569us] +/-   79ms
^- golem.canonical.com           2   7   377    92  -1018us[-1018us] +/-   31ms
^- chilipepper.canonical.com     2   8   377    21  -1106us[-1106us] +/-   27ms
```

You can also make use of the `chronyc sourcestats` command, which produces output like this:

```
210 Number of sources = 20
Name/IP Address            NP  NR  Span  Frequency  Freq Skew  Offset  Std Dev
==============================================================================
gamma.rueckgr.at           25  15   32m     -0.007      0.142   -878us   106us
2b.ncomputers.org          26  16   35m     -0.132      0.283  -1169us   256us
```

(continues on next page)

(continued from previous page)

```
www.kashra.com              25  15   32m     -0.092      0.259  +3426us   195us
stratum2-4.NTP.TechFak.U>   25  14   32m     -0.018      0.130  -2056us    96us
zepto.mcl.gg                13  11   21m     +0.148      0.196   -683us    66us
mirrorhost.pw                6   5   645     +0.117      0.445   -591us    19us
atto.mcl.gg                 21  13   25m     -0.069      0.199   -904us   103us
static.140.107.46.78.cli>   25  18   34m     -0.005      0.094  -1526us    78us
4.53.160.75                 25  10   32m     +0.412      0.110    -11ms    84us
37.44.185.42                24  12   30m     -0.983      0.173  -3718us   122us
bagnikita.com               17   7   31m     -0.132      0.217  +3527us   139us
europa.ellipse.net          26  15   35m     +0.038      0.553   -473us   424us
tethys.hot-chilli.net       25  11   32m     -0.094      0.110   -864us    88us
66-232-97-8.static.hvvc.>   20  11   35m     -0.116      0.165  +1561us   109us
85.199.214.102              26  11   35m     -0.054      0.390   +129us   343us
46-243-26-34.tangos.nl      25  16   32m     +0.129      0.297   -307us   198us
pugot.canonical.com         25  14   34m     -0.271      0.176   -143us   135us
alphyn.canonical.com        17  11  1100     -0.087      0.360  -1749us   114us
golem.canonical.com         23  12   30m     +0.057      0.370   -988us   229us
chilipepper.canonical.com   25  18   34m     -0.084      0.224  -1116us   169us
```

Certain `chronyc` commands are privileged and cannot be run via the network without explicitly allowing them. See the **Command and monitoring access** section in `man chrony.conf` for more details. A local admin can use `sudo` since this will grant access to the local admin socket `/var/run/chrony/chronyd.sock`.

### Pulse-Per-Second (PPS) support

`Chrony` supports various PPS types natively. It can use kernel PPS API as well as Precision Time Protocol (PTP) hardware clocks. Most general GPS receivers can be leveraged via *GPSD*. The latter (and potentially more) can be accessed via **SHM** or via a **socket** (recommended). All of the above can be used to augment `chrony` with additional high quality time sources for better accuracy, *jitter*, drift, and longer- or shorter-term accuracy. Usually, each kind of clock type is good at one of those, but non-perfect at the others. For more details on configuration see some of the external PPS/GPSD resources listed below.

> **Note**
>
> As of the release of 20.04, there was a bug which - until fixed - you might want to add this content to your `/etc/apparmor.d/local/usr.sbin.gpsd`.

### Example configuration for GPSD to feed `chrony`

For the installation and setup you will first need to run the following command in your terminal window:

```
sudo apt install gpsd chrony
```

However, since you will want to test/debug your setup (especially the GPS reception), you should also install:

```
sudo apt install pps-tools gpsd-clients
```

GPS devices usually communicate via serial interfaces. The most common type these days are USB GPS devices, which have a serial converter behind USB. If you want to use one of these devices for PPS then please be aware that the majority do not signal PPS via USB. Check the GPSD hardware list for details. The examples below were run with a Navisys GR701-W.

When plugging in such a device (or at boot time) *dmesg* should report a serial connection of some sort, as in this example:

```
[   52.442199] usb 1-1.1: new full-speed USB device number 3 using xhci_hcd
[   52.546639] usb 1-1.1: New USB device found, idVendor=067b, idProduct=2303,
bcdDevice= 4.00
[   52.546654] usb 1-1.1: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[   52.546665] usb 1-1.1: Product: USB-Serial Controller D
[   52.546675] usb 1-1.1: Manufacturer: Prolific Technology Inc.
[   52.602103] usbcore: registered new interface driver usbserial_generic
[   52.602244] usbserial: USB Serial support registered for generic
[   52.609471] usbcore: registered new interface driver pl2303
[   52.609503] usbserial: USB Serial support registered for pl2303
[   52.609564] pl2303 1-1.1:1.0: pl2303 converter detected
[   52.618366] usb 1-1.1: pl2303 converter now attached to ttyUSB0
```

We see in this example that the device appeared as `ttyUSB0`. So that `chrony` later accepts being fed time information by this device, we have to set it up in `/etc/chrony/chrony.conf` (please replace `USB0` with whatever applies to your setup):

```
refclock SHM 0 refid GPS precision 1e-1 offset 0.9999 delay 0.2
refclock SOCK /var/run/chrony.ttyUSB0.sock refid PPS
```

Next, we need to restart `chrony` to make the socket available and have it waiting.

```
sudo systemctl restart chrony
```

We then need to tell `gpsd` which device to manage. Therefore, in `/etc/default/gpsd` we set:

```
DEVICES="/dev/ttyUSB0"
```

It should be noted that since the *default* use-case of `gpsd` is, well, for *gps position tracking*, it will normally not consume any CPU since it is just waiting on a **socket** for clients. Furthermore, the client will tell `gpsd` what it requests, and `gpsd` will only provide what is asked for.

For the use case of `gpsd` as a PPS-providing-daemon, you want to set the option to:

- Immediately start (even without a client connected). This can be set in `GPSD_OPTIONS` of `/etc/default/gpsd`:
  - `GPSD_OPTIONS="-n"`
- Enable the service itself and not wait for a client to reach the socket in the future:
  - `sudo systemctl enable /lib/systemd/system/gpsd.service`

Restarting `gpsd` will now initialize the PPS from GPS and in `dmesg` you will see:

```
pps_ldisc: PPS line discipline registered
pps pps0: new PPS source usbserial0
pps pps0: source "/dev/ttyUSB0" added
```

If you have multiple PPS sources, the tool `ppsfind` may be useful to help identify which PPS belongs to which GPS. In our example, the command `sudo ppsfind /dev/ttyUSB0` would return the following:

```
pps0: name=usbserial0 path=/dev/ttyUSB0
```

Now we have completed the basic setup. To proceed, we now need our GPS to get a lock. Tools like `cgps` or `gpsmon` need to report a 3D "fix" in order to provide accurate data. Let's run the command `cgps`, which in our case returns:

```
...
| Status:         3D FIX (7 secs) ...
```

You would then want to use `ppstest` in order to check that you are really receiving PPS data. So, let us run the command `sudo ppstest /dev/pps0`, which will produce an output like this:

```
trying PPS source "/dev/pps0"
found PPS source "/dev/pps0"
ok, found 1 source(s), now start fetching data...
source 0 - assert 1588140739.099526246, sequence: 69 - clear  1588140739.
999663721, sequence: 70
source 0 - assert 1588140740.099661485, sequence: 70 - clear  1588140739.
999663721, sequence: 70
source 0 - assert 1588140740.099661485, sequence: 70 - clear  1588140740.
999786664, sequence: 71
source 0 - assert 1588140741.099792447, sequence: 71 - clear  1588140740.
999786664, sequence: 71
```

Ok, `gpsd` is now running, the GPS reception has found a fix, and it has fed this into `chrony`. Let's check on that from the point of view of `chrony`.

Initially, before `gpsd` has started or before it has a lock, these sources will be new and "untrusted" - they will be marked with a "?" as shown in the example below. If your devices remain in the "?" state (even after some time) then `gpsd` is not feeding any data to `chrony` and you will need to debug why.

```
chronyc> sources
210 Number of sources = 10
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
#? GPS                           0   4     0     -     +0ns[   +0ns] +/-    0ns
#? PPS                           0   4     0     -     +0ns[   +0ns] +/-    0ns
```

Over time, `chrony` will classify all of the unknown sources as "good" or "bad". In the example below, the raw GPS had too much deviation (+- 200ms) but the PPS is good (+- 63us).

```
chronyc> sources
210 Number of sources = 10
```

(continues on next page)

(continued from previous page)

```
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
#x GPS                           0   4   177    24 -876ms[ -876ms] +/- 200ms
#- PPS                           0   4   177    21 +916us[ +916us] +/- 63us
^- chilipepper.canonical.com     2   6    37    53  +33us[ +33us]  +/- 33ms
```

Finally, after a while it used the hardware PPS input (as it was better):

```
chronyc> sources
210 Number of sources = 10
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
#x GPS                           0   4   377    20   -884ms[ -884ms] +/-  200ms
#* PPS                           0   4   377    18  +6677ns[  +52us] +/-   58us
^- alphyn.canonical.com          2   6   377    20  -1303us[-1258us] +/-  114ms
```

The PPS might also be OK – but used in a combined way with the selected server, for example. See `man chronyc` for more details about how these combinations can look:

```
chronyc> sources
210 Number of sources = 11
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
#? GPS                           0   4     0     -     +0ns[   +0ns] +/-    0ns
#+ PPS                           0   4   377    22   +154us[ +154us] +/- 8561us
^* chilipepper.canonical.com     2   6   377    50   -353us[ -300us] +/-   44ms
```

If you're wondering if your SHM-based GPS data is any good, you can check on that as well. `chrony` will not only tell you if the data is classified as good or bad – using `sourcestats` you can also check the details:

```
chronyc> sourcestats
210 Number of sources = 10
Name/IP Address            NP  NR  Span  Frequency  Freq Skew  Offset  Std Dev
===============================================================================
GPS                        20   9   302     +1.993     11.501   -868ms  1208us
PPS                         6   3    78     +0.324      5.009  +3365ns    41us
golem.canonical.com        15  10   783     +0.859      0.509   -750us   108us
```

You can also track the raw data that `gpsd` or other `ntpd`-compliant reference clocks are sending via shared memory by using `ntpshmmon`. Let us run the command `sudo ntpshmmon -o`, which should provide the following output:

```
ntpshmmon: version 3.20
#      Name          Offset       Clock                Real                 L Prc
sample NTP1          0.000223854  1588265805.000223854  1588265805.000000000 0 -10
sample NTP0          0.125691783  1588265805.125999851  1588265805.000308068 0 -20
sample NTP1          0.000349341  1588265806.000349341  1588265806.000000000 0 -10
sample NTP0          0.130326636  1588265806.130634945  1588265806.000308309 0 -20
sample NTP1          0.000485216  1588265807.000485216  1588265807.000000000 0 -10
```

### NTS Support

In Chrony 4.0 (which first appeared in Ubuntu 21.04 Hirsute) support for Network Time Security "NTS" was added.

### NTS server

To set up your server with NTS you'll need certificates so that the server can authenticate itself and, based on that, allow the encryption and verification of NTP traffic.

In addition to the `allow` statement that any `chrony` (while working as an NTP server) needs there are two mandatory config entries that will be needed. Example certificates for those entries would look like:

```
ntsservercert /etc/chrony/fullchain.pem
ntsserverkey /etc/chrony/privkey.pem
```

It is important to note that for isolation reasons `chrony`, by default, runs as user and group `_chrony`. Therefore you need to grant access to the certificates for that user, by running the following command:.

```
sudo chown _chrony:_chrony /etc/chrony/*.pem
```

Then restart `chrony` with `systemctl restart chrony` and it will be ready to provide NTS-based time services.

A running `chrony` server measures various statistics. One of them counts the number of NTS connections that were established (or dropped) – we can check this by running `sudo chronyc -N serverstats`, which shows us the statistics:

```
NTP packets received       : 213
NTP packets dropped        : 0
Command packets received   : 117
Command packets dropped    : 0
Client log records dropped : 0
NTS-KE connections accepted: 2
NTS-KE connections dropped : 0
Authenticated NTP packets  : 197
```

There is also a per-client statistic which can be enabled by the `-p` option of the `clients` command.

```
sudo chronyc -N clients -k
```

This provides output in the following form:

```
    Hostname                      NTP   Drop Int IntL Last  NTS-KE   Drop Int
Last

===============================================================================
    10.172.196.173                197      0  10   -   595       2      0   5
48h
    ...
```

For more complex scenarios there are many more advanced options for configuring NTS. These are documented in the `chrony.conf(5)` manual page.

**Note**

*About certificate placement*

Chrony, by default, is isolated via AppArmor and uses a number of `protect*` features of `systemd`. Due to that, there are not many paths `chrony` can access for the certificates. But `/etc/chrony/*` is allowed as read-only and that is enough. Check `/etc/apparmor.d/usr.sbin.chronyd` if you want other paths or allow custom paths in `/etc/apparmor.d/local/usr.sbin.chronyd`.

### References

- Chrony FAQ
- ntp.org: home of the Network Time Protocol project
- pool.ntp.org: project of virtual cluster of timeservers
- Freedesktop.org info on timedatectl
- Freedesktop.org info on systemd-timesyncd service
- Feeding chrony from GPSD
- See the Ubuntu Time wiki page for more information.

### Network shares

Sharing files and resources across a network is a common requirement - this is where the Network File System (NFS) comes in.

### Network File System (NFS)

NFS allows a system to share directories and files with others over a network. By using NFS, users and programs can access files on remote systems almost as if they were local files.

Some of the most notable benefits that NFS can provide are:

- Local workstations use less disk space because commonly used data can be stored on a single machine and still remain accessible to others over the network.
- There is no need for users to have separate home directories on every network machine. Home directories could be set up on the NFS server and made available throughout the network.
- Storage devices such as floppy disks, CDROM drives, and USB Thumb drives can be used by other machines on the network. This may reduce the number of removable media drives throughout the network.

**Warning**

If you use **NFS and authd** at the same time, you must add a Kerberos configuration on both the client and the server. Otherwise, you will encounter permission issues due to mismatched user and group identifiers.

For details, see Using authd with NFS.

### Installation

At a terminal prompt enter the following command to install the NFS Server:

```
sudo apt install nfs-kernel-server
```

To start the NFS server, you can run the following command at a terminal prompt:

```
sudo systemctl start nfs-kernel-server.service
```

### Configuration

You can configure the directories to be exported by adding them to the `/etc/exports` file. For example:

```
/srv     *(ro,sync,subtree_check)
/home    *.hostname.com(rw,sync,no_subtree_check)
/scratch *(rw,async,no_subtree_check,no_root_squash)
```

Make sure any custom mount points you're adding have been created (/srv and /home will already exist):

```
sudo mkdir /scratch
```

Apply the new config via:

```
sudo exportfs -a
```

You can replace * with one of the *hostname* formats. Make the hostname declaration as specific as possible so unwanted systems cannot access the NFS mount. Be aware that `*.hostname.com` will match `foo.hostname.com` but not `foo.bar.my-domain.com`.

The *sync*/*async* options control whether changes are guaranteed to be committed to stable storage before replying to requests. *async* thus gives a performance benefit but risks data loss or corruption. Even though *sync* is the default, it's worth setting since exportfs will issue a warning if it's left unspecified.

*subtree_check* and *no_subtree_check* enables or disables a security verification that subdirectories a client attempts to mount for an exported *filesystem* are ones they're permitted to do so. This verification step has some performance implications for some use cases, such as home directories with frequent file renames. Read-only filesystems are more suitable to enable *subtree_check* on. Like with sync, exportfs will warn if it's left unspecified.

There are a number of optional settings for NFS mounts for tuning performance, tightening security, or providing conveniences. These settings each have their own trade-offs so it is important to use them with care, only as needed for the particular use case. *no_root_squash*, for example, adds a convenience to allow root-owned files to be modified by any client system's

root user; in a multi-user environment where executables are allowed on a shared mount point, this could lead to security problems.

### NFS Client Configuration

To enable NFS support on a client system, enter the following command at the terminal prompt:

```
sudo apt install nfs-common
```

Use the mount command to mount a shared NFS directory from another machine, by typing a command line similar to the following at a terminal prompt:

```
sudo mkdir /opt/example
sudo mount example.hostname.com:/srv /opt/example
```

> **Warning**
>
> The mount point directory `/opt/example` must exist. There should be no files or subdirectories in the `/opt/example` directory, else they will become inaccessible until the nfs filesystem is unmounted.

An alternate way to mount an NFS share from another machine is to add a line to the `/etc/fstab` file. The line must state the hostname of the NFS server, the directory on the server being exported, and the directory on the local machine where the NFS share is to be mounted.

The general syntax for the line in `/etc/fstab` file is as follows:

```
example.hostname.com:/srv /opt/example nfs rsize=8192,wsize=8192,timeo=14,intr
```

### Advanced Configuration

NFS is comprised of several services, both on the server and the client. Each one of these services can have its own default configuration, and depending on the Ubuntu Server release you have installed, this configuration is done in different files, and with a different syntax.

### Ubuntu Server 22.04 LTS (“jammy”)

All NFS related services read a single configuration file: `/etc/nfs.conf`. This is a INI-style config file, see the `nfs.conf(5)` manual page for details. Furthermore, there is a `/etc/nfs.conf.d` directory which can hold `*.conf` snippets that can override settings from previous snippets or from the `nfs.conf` main config file itself.

There is a new command-line tool called `nfsconf(8)` which can be used to query or even set configuration parameters in `nfs.conf`. In particular, it has a `--dump` parameter which will show the effective configuration including all changes done by `/etc/nfs.conf.d/*.conf` snippets.

### For Ubuntu Server 20.04 LTS ("focal") and earlier

Earlier Ubuntu releases use the traditional configuration mechanism for the NFS services via `/etc/defaults/` configuration files. These are `/etc/default/nfs-common` and `/etc/default/nfs/kernel-server`, and are used basically to adjust the command-line options given to each daemon.

Each file has a small explanation about the available settings.

> **Warning**
>
> The `NEED_*` parameters have no effect on systemd-based installations, like Ubuntu 20.04 LTS ("focal") and Ubuntu 18.04 LTS ("bionic"). In those systems, to control whether a service should be running or not, use `systemctl enable` or `systemctl disable`, respectively.

### Upgrading to Ubuntu 22.04 LTS ("jammy")

The main change to the NFS packages in Ubuntu 22.04 LTS ("jammy") is the configuration file. Instead of multiple files sourced by startup scripts from `/etc/default/nfs-*`, now there is one main configuration file in `/etc/nfs.conf`, with an INI-style syntax. When upgrading to Ubuntu 22.04 LTS ("jammy") from a release that still uses the `/etc/defaults/nfs-*` configuration files, the following will happen:

- a default `/etc/nfs.conf` configuration file will be installed
- if the `/etc/default/nfs-*` files have been modified, a conversion script will be run and it will create `/etc/nfs.conf.d/local.conf` with the local modifications.

If this conversion script fails, then the package installation will fail. This can happen if the `/etc/default/nfs-*` files have an option that the conversion script wasn't prepared to handle, or a syntax error for example. In such cases, please file a bug using this link: https://bugs.launchpad.net/ubuntu/+source/nfs-utils/+filebug

You can run the conversion tool manually to gather more information about the error: it's in `/usr/share/nfs-common/nfsconvert.py` and must be run as `root`.

If all goes well, as it should in most cases, the system will have `/etc/nfs.conf` with the defaults, and `/etc/nfs.conf.d/local.conf` with the changes. You can merge these two together manually, and then delete `local.conf`, or leave it as is. Just keep in mind that `/etc/nfs.conf` is not the whole story: always inspect `/etc/nfs.conf.d` as well, as it may contain files overriding the defaults.

You can always run `nfsconf --dump` to check the final settings, as it merges together all configuration files and shows the resulting non-default settings.

### Restarting NFS services

Since NFS is comprised of several individual services, it can be difficult to determine what to restart after a certain configuration change.

The tables below summarize all available services, which "meta" service they are linked to, and which configuration file each service uses.

| Service | nfs-utils.service | nfs-server.service | config file (22.04) | config file (< 22.04) /etc/default/nfs-* |
|---|---|---|---|---|
| nfs-blkmap | PartOf | | nfs.conf | |
| nfs-mountd | | BindsTo | nfs.conf | nfs-kernel-server |
| nfsdcld | | | | |
| nfs-idmapd | | BindsTo | nfs.conf, idmapd.conf | idmapd.conf |
| rpc-gssd | PartOf | | nfs.conf | |
| rpc-statd | PartOf | | nfs.conf | nfs-common |
| rpc-svcgssd | PartOf | BindsTo | nfs.conf | nfs-kernel-server |

For example, `systemctl restart nfs-server.service` will restart `nfs-mountd`, `nfs-idmapd` and `rpc-svcgssd` (if running). On the other hand, restarting `nfs-utils.service` will restart `nfs-blkmap`, `rpc-gssd`, `rpc-statd` and `rpc-svcgssd`.

Of course, each service can still be individually restarted with the usual `systemctl restart <service>`.

The `nfs.systemd(7)` manpage has more details on the several systemd units available with the NFS packages.

### NFS with Kerberos

Kerberos with NFS adds an extra layer of security on top of NFS. It can be just a stronger authentication mechanism, or it can also be used to sign and encrypt the NFS traffic.

This section will assume you already have setup a Kerberos server, with a running KDC and admin services. Setting that up is explained elsewhere in the Ubuntu Server Guide.

### NFS server (using kerberos)

The NFS server will have the usual `nfs-kernel-server` package and its dependencies, but we will also have to install kerberos packages. The kerberos packages are not strictly necessary, as the necessary keys can be copied over from the KDC, but it makes things much easier.

For this example, we will use:

- `.vms` *DNS* domain
- `VMS` Kerberos realm
- `j-nfs-server.vms` for the NFS server
- `j-nfs-client.vms` for the NFS client
- `ubuntu/admin` principal has admin privileges on the KDC

Adjust these names according to your setup.

First, install the `krb5-user` package:

```
sudo apt install krb5-user
```

Then, with an admin principal, let's create a key for the NFS server:

```
$ sudo kadmin -p ubuntu/admin -q "addprinc -randkey nfs/j-nfs-server.vms"
```

And extract the key into the local keytab:

```
$ sudo kadmin -p ubuntu/admin -q "ktadd nfs/j-nfs-server.vms"
Authenticating as principal ubuntu/admin with password.
Password for ubuntu/admin@VMS:
Entry for principal nfs/j-nfs-server.vms with kvno 2, encryption type aes256-cts-
hmac-sha1-96 added to keytab FILE:/etc/krb5.keytab.
Entry for principal nfs/j-nfs-server.vms with kvno 2, encryption type aes128-cts-
hmac-sha1-96 added to keytab FILE:/etc/krb5.keytab.
```

Confirm the key is available:

```
$ sudo klist -k
Keytab name: FILE:/etc/krb5.keytab
KVNO Principal
---- --------------------------------------------------------------------------
   2 nfs/j-nfs-server.vms@VMS
   2 nfs/j-nfs-server.vms@VMS
```

Now install the NFS server:

```
$ sudo apt install nfs-kernel-server
```

This will already automatically start the kerberos-related nfs services, because of the presence of `/etc/krb5.keytab`.

Now populate `/etc/exports`, restricting the exports to krb5 authentication. For example, exporting `/storage` using `krb5p`:

```
/storage *(rw,sync,no_subtree_check,sec=krb5p)
```

Refresh the exports:

```
$ sudo exportfs -rav
exporting *:/storage
```

The security options are explained in the `exports(5)` manpage, but generally they are:

- `krb5`: use kerberos for authentication only (non-auth traffic is in clear text)
- `krb5i`: use kerberos for authentication and integrity checks (non-auth traffic is in clear text)
- `krb5p`: use kerberos for authentication, integrity and privacy protection (non-auth traffic is encrypted)

### NFS client (using kerberos)

The NFS client has a similar set of steps. First we will prepare the client's keytab, so that when we install the NFS client package it will start the extra kerberos services automatically just by detecting the presence of the keytab:

```
sudo apt install krb5-user
```

To allow the `root` user to mount NFS shares via kerberos without a password, we have to create a host key for the NFS client:

```
sudo kadmin -p ubuntu/admin -q "addprinc -randkey host/j-nfs-client.vms"
```

And extract it:

```
$ sudo kadmin -p ubuntu/admin -q "ktadd host/j-nfs-client.vms"
```

Now install the NFS client package:

```
$ sudo apt install nfs-common
```

And you should be able to do your first NFS kerberos mount:

```
$ sudo mount j-nfs-server:/storage /mnt
```

If you are using a machine credential, then the above mount will work without having a kerberos ticket, i.e., `klist` will show no tickets:

```
# mount j-nfs-server:/storage /mnt
# ls -l /mnt/*
-rw-r--r-- 1 root root 0 Apr  5 14:50 /mnt/hello-from-nfs-server.txt
# klist
klist: No credentials cache found (filename: /tmp/krb5cc_0)
```

Notice the above was done with `root`. Let's try accessing that existing mount with the `ubuntu` user, without acquiring a kerberos ticket:

```
# sudo -u ubuntu -i
$ ls -l /mnt/*
ls: cannot access '/mnt/*': Permission denied
```

The `ubuntu` user will only be able to access that mount if they have a kerberos ticket:

```
$ kinit
Password for ubuntu@VMS:
$ ls -l /mnt/*
-rw-r--r-- 1 root root 0 Apr  5 14:50 /mnt/hello-from-nfs-server.txt
```

And now we have not only the TGT, but also a ticket for the NFS service:

```
$ klist
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: ubuntu@VMS
```

(continues on next page)

(continued from previous page)

```
Valid starting     Expires            Service principal
04/05/22 17:48:50  04/06/22 03:48:50  krbtgt/VMS@VMS
        renew until 04/06/22 17:48:48
04/05/22 17:48:52  04/06/22 03:48:50  nfs/j-nfs-server.vms@
        renew until 04/06/22 17:48:48
        Ticket server: nfs/j-nfs-server.vms@VMS
```

One drawback of using a machine credential for mounts done by the `root` user is that you need a persistent secret (the `/etc/krb5.keytab` file) in the filesystem. Some sites may not allow such a persistent secret to be stored in the filesystem. An alternative is to use `rpc.gssds -n` option. From `rpc.gssd(8)`:

- `-n`: when specified, UID 0 is forced to obtain user credentials which are used instead of the local system's machine credentials.

When this option is enabled and `rpc.gssd` restarted, then even the `root` user will need to obtain a kerberos ticket to perform an NFS kerberos mount.

> Warning
>
> Note that this prevents automatic NFS mounts via `/etc/fstab`, unless a kerberos ticket is obtained before.

In Ubuntu 22.04 LTS ("jammy"), this option is controlled in `/etc/nfs.conf` in the `[gssd]` section:

```
[gssd]
use-machine-creds=0
```

In older Ubuntu releases, the command line options for the `rpc.gssd` daemon are not exposed in `/etc/default/nfs-common`, therefore a systemd override file needs to be created. You can either run:

```
$ sudo systemctl edit rpc-gssd.service
```

And paste the following into the editor that will open:

```
[Service]
ExecStart=
ExecStart=/usr/sbin/rpc.gssd $GSSDARGS -n
```

Or manually create the file `/etc/systemd/system/rpc-gssd.service.d/override.conf` and any needed directories up to it, with the contents above.

After you restart the service with `systemctl restart rpc-gssd.service`, the `root` user won't be able to mount the NFS kerberos share without obtaining a ticket first.

### References

- Linux NFS wiki
- Linux NFS faq
- Ubuntu Wiki NFS Howto

- Ubuntu Wiki NFSv4 Howto

If you need to share network resources between Linux and Windows systems, see our sections on Samba and Active Directory.

### Samba

A Samba server can be deployed as a full Active Directory Domain Controller (Samba AD/DC), providing authentication to domain users – whether Linux or Windows.

### Provisioning a Samba Active Directory Domain Controller

A Samba Active Directory Domain Controller (also known as just Samba AD/DC) is a server running Samba services that can provide authentication to domain users and computers, linux or Windows. It should be dedicated to authentication and authorization services, and not provide file or print services: that should be the role of member servers joined to the domain.

**See also**

For more information on why the Samba AD/DC server should not be used to provide file and print services, please refer to this list of reasons and caveats in the Samba Wiki.

This guide will show how to bootstrap a Samba AD/DC server and verify it's functioning properly.

### Installation

This command will install the packages necessary for bootstrapping and testing the Samba AD/DC services:

```
sudo apt install samba-ad-dc krb5-user bind9-dnsutils
```

Note that the installation of `krb5-user` might prompt some questions. It's fine to answer with just the default values (just hit ENTER) at this stage, including when it asks about what the Kerberos realm and servers are.

Next, we should make sure that the normal Samba services `smbd`, `nmbd`, and `windind`, are disabled:

```
sudo systemctl disable --now smbd nmbd winbind
sudo systemctl mask smbd nmbd winbind
```

And enable the Samba AD/DC service, but without starting it yet:

```
sudo systemctl unmask samba-ad-dc
sudo systemctl enable samba-ad-dc
```

**Note**

A Samba AD/DC deployment represents a collection of services connected to each other, and needs its own specific systemd service unit.

The Samba AD/DC provisioning tool will want to create a new Samba configuration file, dedicated to the AD/DC role, but it will refrain from replacing an existing one. We have to therefore move it away before continuing:

```
sudo mv /etc/samba/smb.conf /etc/samba/smb.conf.orig
```

### Provisioning

With the packages installed, the Samba AD/DC service can be provisioned. For this how-to, we will use the following values:

- Domain: `EXAMPLE`
- Realm: `EXAMPLE.INTERNAL`
- Administrator password: `Passw0rd` (pick your own)

To perform the provisioning, run this command:

```
sudo samba-tool domain provision \
    --domain EXAMPLE \
    --realm=EXAMPLE.INTERNAL \
    --adminpass=Passw0rd \
    --server-role=dc \
    --use-rfc2307 \
    --dns-backend=SAMBA_INTERNAL
```

If you omit the `--adminpass` option, a random password will be chosen and be included in the provisioning output. Be sure to save it!

> **Warning**
>
> Providing passwords in the command line is generally unsafe. Other users on the system who can see the process listing can spot the password, and it will also be saved in the shell history, unless the command starts with a blank space.

The command will take a few seconds to run, and will output a lot of information. In the end, it should be like this (long lines truncated for better readability):

```
(...)
INFO ... #498: Server Role:     active directory domain controller
INFO ... #499: Hostname:        ad
INFO ... #500: NetBIOS Domain:  EXAMPLE
INFO ... #501: DNS Domain:      example.internal
INFO ... #502: DOMAIN SID:      S-1-5-21-2373640847-2123283686-338028823
```

If you didn't use the `--adminpass` option, the administrator password will be part of the output above in a line like this:

```
INFO ... #497: Admin password:  sbruR-Py>];k=KDn1H58PB#
```

### Post-installation steps

The AD/DC services are not running yet. Some post-installation steps are necessary before the services can be started.

First, adjust `dns forwarder` in `/etc/samba/smb.conf` to point at your *DNS* server. It will be used for all queries that are not local to the Active Directory domain we just deployed (`EXAMPLE.INTERNAL`). The provisioning script simply copied the server IP from `/etc/resolv.conf` to this parameter, but if we leave it like that, it will point back to itself:

```
[global]
    dns forwarder = 127.0.0.53
```

If unsure, it's best to use the current DNS server this system is already using. That can be seen with the `resolvectl status` command. Look for the `Current DNS Server` line and note the IP address:

```
Global
         Protocols: -LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported
  resolv.conf mode: stub

Link 2 (enp5s0)
    Current Scopes: DNS
         Protocols: +DefaultRoute -LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported
Current DNS Server: 10.10.17.1
       DNS Servers: 10.10.17.1
        DNS Domain: lxd
```

In the above example, the DNS server is `10.10.17.1`, so that should be used in `/etc/samba/smb.conf`'s `dns forwarder`:

```
[global]
    dns forwarder = 10.10.17.1
```

Next, we need to be sure this system will be using the Samba DNS server for its queries, and for that we need to adjust `/etc/resolv.conf`. That file will be a symlink, so instead of just rewriting its contents, first we have to remove it:

```
sudo unlink /etc/resolv.conf
```

Note: this will make sudo issue complaints about DNS from this point on, until the Samba DNS service is up and running.

And now recreate the file `/etc/resolv.conf` with this content:

```
nameserver 127.0.0.1
search example.internal
```

Stop and disable `systemd-resolved` as the resolver will now be using the Samba DNS server directly:

```
sudo systemctl disable --now systemd-resolved
```

Finally, we need to update `/etc/krb5.conf` with the content generated by the Samba provisioning script. Since this system is dedicated to being a Samba AD/DC server, we can just copy the generated file over:

```
sudo cp -f /var/lib/samba/private/krb5.conf /etc/krb5.conf
```

If there are other Kerberos realms involved, you should manually merge the two files.

We are now ready to start the Samba AD/DC services:

```
sudo systemctl start samba-ad-dc
```

And this concludes the installation. The next section will show how to perform some basic checks.

### Verification

Here are some verification steps that can be run to check that the provisioning was done correctly and the service is ready.

### Kerberos authentication

A Kerberos ticket for the *Administrator* principal can be obtained with the `kinit` command. Note you don't have to be *root* to run this command. It's perfectly fine to get a ticket for a different principal than your own user, even a privileged one:

```
kinit Administrator
```

The command will ask for a password. The password is the one supplied to the `samba-tool` command earlier, when the domain was provisioned, or the randomly chosen one if the `--adminpass` option was not used.

```
Password for Administrator@EXAMPLE.INTERNAL:
```

**See also**

If you are not familiar with Kerberos, please see our *Introduction to Kerberos*.

To verify the ticket was obtained, use `klist`, which should have output similar to the following:

```
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: Administrator@EXAMPLE.INTERNAL

Valid starting     Expires            Service principal
07/24/24 13:52:33  07/24/24 23:52:33  krbtgt/EXAMPLE.INTERNAL@EXAMPLE.INTERNAL
        renew until 07/25/24 13:52:02
```

### DNS tests

Using the Kerberos ticket from the step above, we can check the DNS server that Samba is running.

If everything is correct, the `hostname` command should be able to return both the short *hostname*, and the fully qualified hostname.

For the short hostname, use the command:

```
hostname
```

For the fully qualified hostname, run this instead:

```
hostname -f
```

For example, `hostname -f` would return something like `ad.example.internal`, while `hostname` returns only `ad`.

### Server Information

With that information at hand and verified, we can perform further checks. Let's get information about the DNS service provided by this domain controller:

```
samba-tool dns serverinfo $(hostname -f)
```

This command will produce a long output, truncated below:

```
dwVersion                   : 0xece0205
fBootMethod                 : DNS_BOOT_METHOD_DIRECTORY
fAdminConfigured            : FALSE
fAllowUpdate                : TRUE
fDsAvailable                : TRUE
pszServerName               : AD.example.internal
pszDsContainer              : CN=MicrosoftDNS,DC=DomainDnsZones,DC=example,
DC=internal
(...)
```

Even though it doesn't look like it, the `samba-tool dns serverinfo` command used Kerberos authentication. The `klist` command output now shows another ticket that was obtained automatically:

```
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: Administrator@EXAMPLE.INTERNAL

Valid starting     Expires            Service principal
07/24/24 14:29:55  07/25/24 00:29:55  krbtgt/EXAMPLE.INTERNAL@EXAMPLE.INTERNAL
        renew until 07/25/24 14:29:53
07/24/24 14:29:59  07/25/24 00:29:55  host/ad.example.internal@EXAMPLE.INTERNAL
        renew until 07/25/24 14:29:53
```

A ticket for the `host/ad.example.internal@EXAMPLE.INTERNAL` principal is now also part of the ticket cache.

### DNS records

The DNS server backing the Samba Active Directory deployment also needs to have some specific DNS records in its zones, which are needed for service discoverability. Let's check if they were added correctly with this simple script:

```
for srv in _ldap._tcp _kerberos._tcp _kerberos._udp _kpasswd._udp; do
    echo -n "${srv}.example.internal: "
    dig @localhost +short -t SRV ${srv}.example.internal
done
```

The output should have no errors, and show the DNS records for each query:

```
_ldap._tcp.example.internal: 0 100 389 ad.example.internal.
_kerberos._tcp.example.internal: 0 100 88 ad.example.internal.
_kerberos._udp.example.internal: 0 100 88 ad.example.internal.
_kpasswd._udp.example.internal: 0 100 464 ad.example.internal.
```

And, of course, our own hostname must be in DNS as well:

```
dig @localhost +short -t A $(hostname -f)
```

The above command should return your own IP address.

### User creation test

Users (and groups, and other entities as well) can be created with the `samba-tool` command. It can be used remotely, to manage users on a remote Samba AD server, or locally on the same server it is managing.

When run **locally as root**, no further authentication is required:

```
samba-tool user add noble
```

The command above will prompt for the desired password for the `noble` user, and if it satisfies the password complexity criteria, the user will be created. As with the *Administrator* Samba user, `kinit noble` can be used to test that the `noble` user can be authenticated.

> **Note**
>
> `samba-tool` creates **Samba** users, not local Linux users! These Samba users only exist for domain joined computers and other SMB connections/shares.

The default password policy is quite severe, requiring complex passwords. To display the current policy, run:

```
sudo samba-tool domain passwordsettings show
```

Which will show the default password policy for the domain:

```
Password information for domain 'DC=example,DC=internal'
```

(continues on next page)

(continued from previous page)

```
Password complexity: on
Store plaintext passwords: off
Password history length: 24
Minimum password length: 7
Minimum password age (days): 1
Maximum password age (days): 42
Account lockout duration (mins): 30
Account lockout threshold (attempts): 0
Reset account lockout after (mins): 30
```

Each one of these can be changed, including the password complexity. For example, to set the minimum password length to 12:

```
sudo samba-tool domain passwordsettings set --min-pwd-length=12
```

To see all the options, run:

```
samba-tool domain passwordsettings set --help
```

### Next steps

This Samba AD/DC server can be treated as an Active Directory server for Window and Linux systems. Typically next steps would be to create users and groups, and join member servers and workstations to this domain. Workstation users would login using the domain credentials, and member servers are used to provide file and print services.

### References

- Active Directory Domain Services Overview
- `samba-tool(8)` manual page
- Active Directory integration:
  - *Choosing an integration method*
  - *Joining a Member Server*
  - *Joining Workstations (without Samba services)*

### Member server in an Active Directory domain

A Samba server needs to join the Active Directory (AD) domain before it can serve files and printers to Active Directory users. This is different from Network User Authentication with SSSD, where we integrate the AD users and groups into the local Ubuntu system as if they were local.

For Samba to authenticate these users via Server Message Block (SMB) authentication protocols, we need both for the remote users to be “seen”, and for Samba itself to be aware of the domain. In this scenario, Samba is called a Member Server or Domain Member.

**See also**

Samba itself has the necessary tooling to join an Active Directory domain. It requires a sequence of manual steps and configuration file editing, which is thoroughly documented on the Samba wiki. It's useful to read that documentation to get an idea of the steps necessary, and the decisions you will need to make.

### Use `realmd` to join the Active Directory domain

For this guide, though, we are going to use the `realmd` package and instruct it to use the Samba tooling for joining the AD domain. This package will make certain decisions for us which will work for most cases, but more complex setups involving multiple or very large domains might require additional tweaking.

### Install `realmd`

First, let's install the necessary packages:

```
sudo apt install realmd samba
```

In order to have the joined machine registered in the AD *DNS*, it needs to have an *FQDN* set. You might have that already, if running the `hostname -f` command returns a full *hostname* with domain. If it doesn't, then set the hostname as follows:

```
sudo hostnamectl hostname <yourfqdn>
```

For this guide, we will be using `j1.internal.example.fake`, and the AD domain will be `internal.example.fake`.

### Verify the AD server

Next, we need to verify that the AD server is both reachable and known by running the following command:

```
sudo realm discover internal.example.fake
```

This should provide an output like this, given our setup:

```
internal.example.fake
  type: kerberos
  realm-name: INTERNAL.EXAMPLE.FAKE
  domain-name: internal.example.fake
  configured: no
  server-software: active-directory
  client-software: sssd
  required-package: sssd-tools
  required-package: sssd
  required-package: libnss-sss
  required-package: libpam-sss
  required-package: adcli
  required-package: samba-common-bin
```

`realm` is suggesting a set of packages for the discovered domain, but we will override that and select the Samba tooling for this join, because we want Samba to become a Member Server.

### Join the AD domain

Let's join the domain in verbose mode so we can see all the steps:

```
sudo realm join -v --membership-software=samba --client-software=winbind
internal.example.fake
```

This should produce the following output for us:

```
 * Resolving: _ldap._tcp.internal.example.fake
 * Performing LDAP DSE lookup on: 10.0.16.5
 * Successfully discovered: internal.example.fake
Password for Administrator:
 * Unconditionally checking packages
 * Resolving required packages
 * Installing necessary packages: libnss-winbind samba-common-bin libpam-winbind
winbind
 * LANG=C LOGNAME=root /usr/bin/net --configfile /var/cache/realmd/realmd-smb-
conf.A53NO1 -U Administrator --use-kerberos=required ads join internal.example.
fake
Password for [INTEXAMPLE\Administrator]:
Using short domain name -- INTEXAMPLE
Joined 'J1' to dns domain 'internal.example.fake'
 * LANG=C LOGNAME=root /usr/bin/net --configfile /var/cache/realmd/realmd-smb-
conf.A53NO1 -U Administrator ads keytab create
Password for [INTEXAMPLE\Administrator]:
 * /usr/sbin/update-rc.d winbind enable
 * /usr/sbin/service winbind restart
 * Successfully enrolled machine in realm
```

**Note**

This command also installed the `libpam-winbind` package, **which allows AD users to authenticate to other services on this system via PAM, like SSH or console logins**. For example, if your SSH server allows password authentication (`PasswordAuthentication yes` in `/etc/ssh/sshd_config`), then the domain users will be allowed to login remotely on this system via SSH. If you don't expect or need AD users to log into this system (unless it's via Samba or Windows), then it's safe and probably best to remove the `libpam-winbind` package.

Until bug #1980246 is fixed, one extra step is needed:

- Configure `/etc/nsswitch.conf` by adding the word `winbind` to the `passwd` and `group` lines as shown below:

```
passwd:         files systemd winbind
group:          files systemd winbind
```

Now you will be able to query users from the AD domain. Winbind adds the short domain name as a prefix to domain users and groups:

```
$ getent passwd INTEXAMPLE\\Administrator
INTEXAMPLE\administrator:*:2000500:2000513::/home/administrator@INTEXAMPLE:/
bin/bash
```

You can find out the short domain name in the `realm` output shown earlier, or inspect the `workgroup` parameter of `/etc/samba/smb.conf`.

### Common installation options

When domain users and groups are brought to the Linux world, a bit of translation needs to happen, and sometimes new values need to be created. For example, there is no concept of a "login shell" for AD users, but it exists in Linux.

The following are some common `/etc/samba/smb.conf` options you are likely to want to tweak in your installation. The `smb.conf(5)` manual page explains the `%` variable substitutions and other details:

- **home directory** `template homedir = /home/%U@%D` (Another popular choice is `/home/%D/%U`)
- **login shell** `template shell = /bin/bash`
- `winbind separator = \` This is the `\` character between the short domain name and the user or group name that we saw in the `getent passwd` output above.
- `winbind use default domain` If this is set to `yes`, then the domain name will not be part of the users and groups. Setting this to `yes` makes the system more friendly towards Linux users, as they won't have to remember to include the domain name every time a user or group is referenced. However, if multiple domains are involved, such as in an AD forest or other form of domain trust relationship, then leave this setting at `no` (default).

To have the home directory created automatically the first time a user logs in to the system, and if you haven't removed `libpam-winbind`, then enable the `pam_mkhomedir` module via this command:

```
sudo pam-auth-update --enable mkhomedir
```

Note that this won't apply to logins via Samba: this only creates the home directory for system logins like those via `ssh` or the console.

### Export shares

Shares can be exported as usual. Since this is now a Member Server, there is no need to deal with user and group management. All of this is integrated with the Active Directory server we joined.

For example, let's create a simple `[storage]` share. Add this to the `/etc/samba/smb.conf` file:

```
[storage]
    path = /storage
    comment = Storage share
```

(continues on next page)

(continued from previous page)

```
    writable = yes
    guest ok = no
```

Then create the `/storage` directory. Let's also make it `1777` so all users can use it, and then ask samba to reload its configuration:

```
sudo mkdir -m 1777 /storage
sudo smbcontrol smbd reload-config
```

With this, users from the AD domain will be able to access this share. For example, if there is a user `ubuntu` the following command would access the share from another system, using the domain credentials:

```
$ smbclient //j1.internal.example.fake/storage -U INTEXAMPLE\\ubuntu
Enter INTEXAMPLE\ubuntu's password:
Try "help" to get a list of possible commands.
smb: \>
```

And `smbstatus` on the member server will show the connected user:

```
$ sudo smbstatus

Samba version 4.15.5-Ubuntu
PID     Username     Group        Machine
Protocol Version  Encryption           Signing
----------------------------------------------------------------------------------
-------------------------------------------------
3631    INTEXAMPLE\ubuntu INTEXAMPLE\domain users 10.0.16.1 (ipv4:10.0.16.1:39534)
         SMB3_11           -                   partial(AES-128-CMAC)

Service      pid     Machine       Connected at                     Encryption
Signing
----------------------------------------------------------------------------------
-----------
storage      3631    10.0.16.1     Wed Jun 29 17:42:54 2022 UTC     -            -

No locked files
```

You can also restrict access to the share as usual. Just keep in mind the syntax for the domain users. For example, to restrict access to the `[storage]` share we just created to *only* members of the `LTS Releases` domain group, add the `valid users` parameter like below:

```
[storage]
    path = /storage
    comment = Storage share
    writable = yes
    guest ok = no
    valid users = "@INTEXAMPLE\ LTS Releases"
```

### Choose an `idmap` backend

`realm` made some choices for us when we joined the domain. A very important one is the `idmap` backend, and it might need changing for more complex setups.

User and group identifiers on the AD side are not directly usable as identifiers on the Linux site. A *mapping* needs to be performed.

Winbind supports several `idmap` backends, and each one has its own manual page. The three main ones are:

- `idmap_ad(8)`
- `idmap_autorid(8)`
- `idmap_rid(8)`

Choosing the correct backend for each deployment type needs careful planing. Upstream has some guidelines at Choosing an `idmap` backend, and each man page has more details and recommendations.

The `realm` tool selects (by default) the `rid` backend. This backend uses an algorithm to calculate the Unix user and group IDs from the respective RID value on the AD side. You might need to review the `idmap config` settings in `/etc/samba/smb.conf` and make sure they can accommodate the number of users and groups that exist in the domain, and that the range does not overlap with users from other sources.

For example, these settings:

```
idmap config * : range = 10000-999999
idmap config intexample : backend = rid
idmap config intexample : range = 2000000-2999999
idmap config * : backend = tdb
```

Will reserve the `2,000,000` through `2,999,999` range for user and group ID allocations on the Linux side for the `intexample` domain. The default backend (`*`, which acts as a "globbing" catch-all rule) is used for the `BUILTIN` user and groups, and other domains (if they exist). It's important that these ranges do not overlap.

The `Administrator` user we inspected before with `getent passwd` can give us a glimpse of how these ranges are used (output format changed for clarity):

```
$ id INTEXAMPLE\\Administrator
uid=2000500(INTEXAMPLE\administrator)
gid=2000513(INTEXAMPLE\domain users)
groups=2000513(INTEXAMPLE\domain users),
       2000500(INTEXAMPLE\administrator),
       2000572(INTEXAMPLE\denied rodc password replication group),
       2000519(INTEXAMPLE\enterprise admins),
       2000518(INTEXAMPLE\schema admins),
       2000520(INTEXAMPLE\group policy creator owners),
       2000512(INTEXAMPLE\domain admins),
       10001(BUILTIN\users),
       10000(BUILTIN\administrators)
```

### Further reading

- The Samba Wiki

### Set up sharing services

Samba can be configured as a file server or print server, to share files and printers with Windows clients.

### Set up Samba as a file server

One of the most common ways to network Ubuntu and Windows computers is to configure Samba as a *file server*. It can be set up to share files with Windows clients, as we'll see in this section.

The server will be configured to share files with any client on the network without prompting for a password. If your environment requires stricter Access Controls see *Share Access Control*.

> **Warning**
>
> If you use **Samba and authd** at the same time, you must specify user and group mapping. Otherwise, you will encounter permission issues due to mismatched user and group identifiers.
>
> If you *are* using Samba with authd, you should follow the instructions in the steps for the server guide in the authd documentation instead.

### Install Samba

The first step is to install the `samba` package. From a terminal prompt enter:

```
sudo apt install samba
```

That's all there is to it; you are now ready to configure Samba to share files.

### Configure Samba as a file server

The main Samba configuration file is located in `/etc/samba/smb.conf`. The default configuration file contains a significant number of comments, which document various configuration directives.

> **Note**
>
> Not all available options are included in the default configuration file. See the `smb.conf` man page or the Samba HOWTO Collection for more details.

First, edit the `workgroup` parameter in the *[global]* section of `/etc/samba/smb.conf` and change it to better match your environment:

```
workgroup = EXAMPLE
```

Create a new section at the bottom of the file, or uncomment one of the examples, for the directory you want to share:

```
[share]
    comment = Ubuntu File Server Share
    path = /srv/samba/share
    browsable = yes
    guest ok = yes
    read only = no
    create mask = 0755
```

- `comment` A short description of the share. Adjust to fit your needs.
- `path` The path to the directory you want to share.

> **Note**
>
> This example uses `/srv/samba/sharename` because, according to the *Filesystem Hierarchy Standard (FHS)*, `/srv` is where site-specific data should be served. Technically, Samba shares can be placed anywhere on the *filesystem* as long as the permissions are correct, but adhering to standards is recommended.

- `browsable` Enables Windows clients to browse the shared directory using Windows Explorer.
- `guest ok` Allows clients to connect to the share without supplying a password.
- `read only` determines if the share is read only or if write privileges are granted. Write privileges are allowed only when the value is *no*, as is seen in this example. If the value is *yes*, then access to the share is read only.
- `create mask` Determines the permissions that new files will have when created.

### Create the directory

Now that Samba is configured, the directory needs to be created and the permissions changed. From a terminal, run the following commands:

```
sudo mkdir -p /srv/samba/share
sudo chown nobody:nogroup /srv/samba/share/
```

The `-p` switch tells `mkdir` to create the entire directory tree if it doesn't already exist.

### Enable the new configuration

Finally, restart the Samba services to enable the new configuration by running the following command:

```
sudo systemctl restart smbd.service nmbd.service
```

**Warning**

Once again, the above configuration gives full access to any client on the local network. For a more secure configuration see *Share Access Control*.

From a Windows client you should now be able to browse to the Ubuntu file server and see the shared directory. If your client doesn't show your share automatically, try to access your server by its IP address, e.g. `\\192.168.1.1`, in a Windows Explorer window. To check that everything is working try creating a directory from Windows.

To create additional shares simply create new `[sharename]` sections in `/etc/samba/smb.conf`, and restart Samba. Just make sure that the directory you want to share actually exists and the permissions are correct.

The file share named `[share]` and the path `/srv/samba/share` used in this example can be adjusted to fit your environment. It is a good idea to name a share after a directory on the file system. Another example would be a share name of `[qa]` with a path of `/srv/samba/qa`.

### Further reading

- For in-depth Samba configurations see the Samba how-to collection
- The guide is also available in printed format.
- O'Reilly's Using Samba is another good reference.
- The Ubuntu Wiki Samba page.

### Set up Samba as a print server

Another common way to network Ubuntu and Windows computers is to configure Samba as a *print server*. This will allow it to share printers installed on an Ubuntu server, whether locally or over the network.

Just as we did in using Samba as a file server, this section will configure Samba to allow any client on the local network to use the installed printers without prompting for a username and password.

If your environment requires stricter Access Controls see Share Access Control.

### Install and configure CUPS

Before installing and configuring Samba as a print server, it is best to already have a working CUPS installation. See our guide on CUPS for details.

### Install Samba

To install the `samba` package, run the following command in your terminal:

```
sudo apt install samba
```

### Configure Samba

After installing `samba`, edit `/etc/samba/smb.conf`. Change the *workgroup* attribute to what is appropriate for your network:

```
workgroup = EXAMPLE
```

In the *[printers]* section, change the *guest ok* option to 'yes':

```
browsable = yes
guest ok = yes
```

After editing `smb.conf`, restart Samba:

```
sudo systemctl restart smbd.service nmbd.service
```

The default Samba configuration will automatically share any printers installed. Now all you need to do is install the printer locally on your Windows clients.

### Further reading

- For in-depth Samba configurations see the Samba HOWTO Collection.
- The guide is also available in printed format.
- O'Reilly's Using Samba is another good reference.
- Also, see the CUPS Website for more information on configuring CUPS.
- The Ubuntu Wiki Samba page.

### Access controls

### Share access controls

There are several options available to control access for each individual shared directory. Using the *[share]* example, this section will cover some common options.

Since most of these options will deal with user authentication, we first need to briefly address how that is done in Samba.

### Authenticating Samba users

Samba cannot authenticate existing Linux users using the its native protocols. This is just not compatible with the way Linux passwords are stored in the system (in `/etc/shadow`, for example). All local Linux users that the system may have are not automatically available as Samba users. To have a local Linux user available as a Samba user, they need to be created in the Samba credentials database. That means we will have two user databases: the Linux one, and the Samba one.

**See also**

How to create and manage Linux users is covered in *Users and groups management*.

To add an existing Linux user to the Samba user database, the command `smbpasswd` is used. For example, here we are adding an existing Linux user called `melissa` to the Samba user database:

```
sudo smbpasswd -a melissa
```

The command will prompt for a password twice, for confirmation, and create the Samba user.

> **Note**
>
> As this is a separate user database, the password selected for the Samba user does not need to be the same as the Linux password for that user. In fact, most Samba servers setup this way will have the Linux users setup without a valid password: these users only exist so that the corresponding Samba users can be created.

If this user does not exist in Linux, the `smbpasswd` command will fail. Samba users must first exist as Linux users.

To later change the password of an existing Samba user, the same command is used. For example:

```
sudo smbpasswd melissa
```

Samba also has the concept of Samba groups, but since there is no authentication going on, there is no need to create a separate group database just for Samba groups. We can use the normal Linux group, as long as the group members (the users) exist both in the Linux and Samba user database.

### Groups

*Groups* define a collection of users who have a common level of access to particular network resources. This provides granularity in controlling access to such resources. For example, let's consider a group called "qa" is defined to contain the users *Freda*, *Danika*, and *Rob*, and then a group called "support" is created containing the users *Danika*, *Jeremy*, and *Vincent*. Any network resources configured to allow access by the "qa" group will be available to Freda, Danika, and Rob, but not Jeremy or Vincent. Danika can access resources available to both groups since she belongs to both the "qa" and "support" groups. All other users only have access to resources explicitly allowed to the group they are part of.

When mentioning groups in the Samba configuration file, `/etc/samba/smb.conf`, the recognized syntax is to preface the group name with an "@" symbol. For example, if you wished to use a group named *sysadmin* in a certain section of the `/etc/samba/smb.conf`, you would do so by entering the group name as `@sysadmin`. If a group name has a space in it, use double quotes, like `"@LTS Releases"`.

### Read and write permissions

Read and write permissions define the explicit rights a computer or user has to a particular share. Such permissions may be defined by editing the `/etc/samba/smb.conf` file and specifying the explicit permissions inside a share.

For example, if you have defined a Samba share called *share* and wish to give read-only permissions to the group of users known as "qa", but wanted to allow writing to the share by the

group called “sysadmin” *and* the user named “vincent”, then you could edit the `/etc/samba/smb.conf` file and add the following entries under the *[share]* entry:

```
read list = @qa
write list = @sysadmin, vincent
```

Another possible Samba permission is to declare *administrative* permissions to a particular shared resource. Users having administrative permissions may read, write, or modify any information contained in the resource the user has been given explicit administrative permissions to.

For example, if you wanted to give the user *Melissa* administrative permissions to the *share* example, you would edit the `/etc/samba/smb.conf` file, and add the following line under the *[share]* entry:

```
admin users = melissa
```

**Note**

Remember that the users listed in `smb.conf` for these access controls need to exist both as Linux users, and Samba users.

After editing `/etc/samba/smb.conf`, reload Samba for the changes to take effect by running the following command:

```
sudo smbcontrol smbd reload-config
```

### Filesystem permissions

Now that Samba has been configured to limit which groups have access to the shared directory, the *filesystem* permissions need to be checked.

Traditional Linux file permissions do not map well to Windows NT Access Control Lists (*ACL*s). Fortunately POSIX ACLs are available on Ubuntu servers, which provides more fine-grained control. For example, to enable ACLs on `/srv` in an EXT3 filesystem, edit `/etc/fstab` and add the *acl* option:

```
UUID=66bcdd2e-8861-4fb0-b7e4-e61c569fe17d /srv  ext3    noatime,relatime,acl 0
  1
```

Then remount the partition:

```
sudo mount -v -o remount /srv
```

**Note**

This example assumes `/srv` is on a separate partition. If `/srv`, or wherever you have configured your share path, is part of the `/` partition then a reboot may be required.

To match the Samba configuration above, the “sysadmin” group will be given read, write, and execute permissions to `/srv/samba/share`, the “qa” group will be given read and execute

permissions, and the files will be owned by the username “Melissa”. Enter the following in a terminal:

```
sudo chown -R melissa /srv/samba/share/
sudo chgrp -R sysadmin /srv/samba/share/
sudo setfacl -R -m g:qa:rx /srv/samba/share/
```

**Note**

The `setfacl` command above gives *execute* permissions to all files in the `/srv/samba/share` directory, which you may or may not want.

Now from a Windows client you should notice the new file permissions are implemented. See the `acl(5)` and `setfacl(1)` manual pages for more information on POSIX ACLs.

### Further reading

- For in-depth Samba configurations see the Samba HOWTO Collection.
- The guide is also available in printed format.
- O’Reilly’s Using Samba is also a good reference.
- Chapter 18 of the Samba HOWTO Collection is devoted to security.
- For more information on Samba and ACLs see the Samba ACLs page.
- The Ubuntu Wiki Samba page.

### Create a Samba AppArmor profile

Ubuntu comes with the AppArmor security module, which provides mandatory access controls. The default AppArmor profile for Samba may need to be adapted to your configuration. More details on using AppArmor can be found in this guide.

There are default AppArmor profiles for `/usr/sbin/smbd` and `/usr/sbin/nmbd`, the Samba daemon binaries, as part of the `apparmor-profiles` package.

#### Install `apparmor-profiles`

To install the package, enter the following command from a terminal prompt:

```
sudo apt install apparmor-profiles apparmor-utils
```

**Note**

This package contains profiles for several other binaries.

### AppArmor profile modes

By default, the profiles for `smbd` and `nmbd` are set to 'complain' mode. In this mode, Samba can work without modifying the profile, and only logs errors or violations. There is no need to add exceptions for the shares, as the `smbd` service unit takes care of doing that automatically via a helper script.

This is what an `ALLOWED` message looks like. It means that, were the profile not in `complain` mode, this action would have been denied instead (formatted into multiple lines here for better visibility):

```
Jun 30 14:41:09 ubuntu kernel: [  621.478989] audit:
type=1400 audit(1656600069.123:418):
apparmor="ALLOWED" operation="exec" profile="smbd"
name="/usr/lib/x86_64-linux-gnu/samba/samba-bgqd" pid=4122 comm="smbd"
requested_mask="x" denied_mask="x" fsuid=0 ouid=0
target="smbd//null-/usr/lib/x86_64-linux-gnu/samba/samba-bgqd"
```

The alternative to 'complain' mode is 'enforce' mode, where any operations that violate policy are blocked. To place the profile into `enforce` mode and reload it, run:

```
sudo aa-enforce /usr/sbin/smbd
sudo apparmor_parser -r -W -T /etc/apparmor.d/usr.sbin.smbd
```

It's advisable to monitor `/var/log/syslog` for `audit` entries that contain AppArmor `DENIED` messages, or `/var/log/audit/audit.log` if you are running the `auditd` daemon. Actions blocked by AppArmor may surface as odd or unrelated errors in the application.

### Further reading

- For more information on how to use AppArmor, including details of the profile modes, the Debian AppArmor guide may be helpful.

### How to mount CIFS shares permanently

Common Internet File System (CIFS) shares are a file-sharing protocol used (mainly) in Windows for accessing files and resources (such as printers) over a network.

Permanently mounting CIFS shares involves configuring your system to automatically connect to these shared resources when the system boots, which is useful when network users need consistent and regular access to them.

In this guide, we will show you how to permanently mount and access CIFS shares. The shares can be hosted on a Windows computer/server, or on a Linux/UNIX server running Samba. If you want to know how to host shares, see *Introduction to Samba*.

### Prerequisites

In order to use this guide, you will need to ensure that your network connections have been configured properly. Throughout this guide, we will use the following naming conventions:

- The local (Ubuntu) username is `ubuntuusername`
- The share username on the Windows computer is `msusername`

- The share password on the Windows computer is `mspassword`
- The Windows computer's name is `servername` (this can be either an IP address or an assigned name)
- The name of the share is `sharename`
- The shares are to be mounted in `/media/windowsshare`

### Install CIFS

To install CIFS, run the following command:

```
sudo apt-get install cifs-utils
```

### Mount unprotected (guest) network folders

First, let's create the mount directory. You will need a separate directory for each mount:

```
sudo mkdir /media/windowsshare
```

Then edit your `/etc/fstab` file (with root privileges) to add this line:

```
//servername/sharename /media/windowsshare cifs guest,uid=1000 0 0
```

Where:

- `servername` is the server *hostname* or IP address,
- `guest` indicates you don't need a password to access the share,
- `uid=1000` makes the Linux user (specified by the ID) the owner of the mounted share, allowing them to rename files, and
- If there is any space in the server path, you need to replace it by `\040`, for example: `//servername/My\040Documents`

After you add the entry to `/etc/fstab`, type:

```
sudo mount /media/windowsshare
```

### Mount password-protected network folders

To auto-mount a password-protected share, you can edit `/etc/fstab` (with root privileges), and add this line:

```
//servername/sharename /media/windowsshare cifs username=msusername,
password=mspassword 0 0
```

This is not a good idea however: `/etc/fstab` is readable by everyone – and so is your Windows password within it. The way around this is to use a credentials file. This is a file that contains just the username and password.

### Create a credentials file

Using a text editor, create a file for your remote server's logon credential:

```
gedit ~/.smbcredentials
```

Enter your Windows username and password in the file:

```
username=msusername

password=mspassword
```

Save the file and exit the editor.

Change the permissions of the file to prevent unwanted access to your credentials:

```
chmod 600 ~/.smbcredentials
```

Then edit your `/etc/fstab` file (with root privileges) to add this line (replacing the insecure line in the example above, if you added it):

```
//servername/sharename /media/windowsshare cifs credentials=/home/ubuntuusername/.
smbcredentials 0 0
```

Save the file and exit the editor.

Finally, test mounting the share by running:

```
sudo mount /media/windowsshare
```

If there are no errors, you should test how it works after a reboot. Your remote share should mount automatically. However, if the remote server goes offline, the boot process could present errors because it won't be possible to mount the share.

### Changing the share ownership

If you need to change the owner of a share, you'll need to add a **UID** (short for 'User ID') or **GID** (short for 'Group ID') parameter to the share's mount options:

```
//servername/sharename /media/windowsshare cifs uid=ubuntuusername,credentials=/
home/ubuntuusername/.smbcredentials 0 0
```

### Mount network folders with authd

If you use Samba and authd at the same time, you must specify user and group mapping. Otherwise, you will encounter permission issues due to mismatched user and group identifiers.

If you *are* using Samba with authd, follow the instructions in the steps for the client guide in the authd documentation.

### Mount password-protected shares using `libpam-mount`

In addition to the initial assumptions, we're assuming here that your username and password are the same on both the Ubuntu machine and the network drive.

#### Install `libpam-mount`

```
sudo apt-get install libpam-mount
```

Edit `/etc/security/pam_mount.conf.xml` using your preferred text editor.

```
sudo gedit /etc/security/pam_mount.conf.xml
```

First, we're moving the user-specific config parts to a file which users can actually edit themselves.

Remove the commenting tags (`<!--` and `-->`) surrounding the section called `<luserconf name=".pam_mount.conf.xml" />`. We also need to enable some extra mount options to be used. For that, edit the "`<mntoptions allow=...`" tag and add `uid,gid,dir_mode,credentials` to it.

Save the file when done. With this in place, users can create their own `~/.pam_mount.conf.xml`.

```
gedit ~/.pam_mount.conf.xml
```

Add the following:

```
<?xml version="1.0" encoding="utf-8" ?>

<pam_mount>

<volume options="uid=%(USER),gid=100,dir_mode=0700,credentials=/home/
ubuntuusername/.smbcredentials,nosuid,nodev" user="*" mountpoint="/media/
windowsshare" path="sharename" server="servername" fstype="cifs" />

</pam_mount>
```

### Troubleshooting

#### Login errors

If you get the error "mount error(13) permission denied", then the server denied your access. Here are the first things to check:

- Are you using a valid username and password? Does that account really have access to this folder?
- Do you have blank space in your credentials file? It should be `password=mspassword`, not `password = mspassword`.
- Do you need a domain? For example, if you are told that your username is `SALES\sally`, then actually your username is `sally` and your domain is `SALES`. You can add a `domain=SALES` line to the `~/.credentials` file.

- The security and version settings are interrelated. SMB1 is insecure and no longer supported. At first, try to not specify either security or version: do not specify `sec=` or `vers=`. If you still have authentication errors then you may need to specify either `sec=` or `vers=` or both. You can try the options listed at the `mount.cifs(8)` manual page.

### Mount after login instead of boot

If for some reason, such as networking problems, the automatic mounting during boot doesn't work, you can add the `noauto` parameter to your CIFS `fstab` entry and then have the share mounted at login.

In `/etc/fstab`:

```
//servername/sharename /media/windowsshare cifs noauto,credentials=/home/
ubuntuusername/.smbpasswd 0 0
```

You can now manually mount the share after you log in. If you would like the share to be automatically mounted after each login, please see the section above about `libpam-mount`.

### Legacy options

These options are now deprecated, but still available.

### NT4 domain controller (legacy)

**Note**

This section is flagged as *legacy* because nowadays, Samba can be deployed in full Active Directory domain controller mode, and the old-style NT4 Primary Domain Controller is deprecated.

A Samba server can be configured to appear as a Windows NT4-style domain controller. A major advantage of this configuration is the ability to centralise user and machine credentials. Samba can also use multiple backends to store the user information.

### Primary domain controller

In this section, we'll install and configure Samba as a Primary Domain Controller (PDC) using the default `smbpasswd` backend.

### Install Samba

First, we'll install Samba, and `libpam-winbind` (to sync the user accounts), by entering the following in a terminal prompt:

```
sudo apt install samba libpam-winbind
```

### Configure Samba

Next, we'll configure Samba by editing `/etc/samba/smb.conf`. The *security* mode should be set to *user*, and the *workgroup* should relate to your organization:

```
workgroup = EXAMPLE
...
security = user
```

In the commented "Domains" section, add or uncomment the following (the last line has been split to fit the format of this document):

```
domain logons = yes
logon path = \\%N\%U\profile
logon drive = H:
logon home = \\%N\%U
logon script = logon.cmd
add machine script = sudo /usr/sbin/useradd -N -g machines -c Machine -d
      /var/lib/samba -s /bin/false %u
```

**Note**

If you wish to not use *Roaming Profiles* leave the `logon home` and `logon path` options commented out.

- `domain logons` Provides the `netlogon` service, causing Samba to act as a domain controller.
- `logon path` Places the user's Windows profile into their home directory. It is also possible to configure a *[profiles]* share placing all profiles under a single directory.
- `logon drive` Specifies the home directory local path.
- `logon home` Specifies the home directory location.
- `logon script` Determines the script to be run locally once a user has logged in. The script needs to be placed in the *[netlogon]* share.
- `add machine script` A script that will automatically create the *Machine Trust Account* needed for a workstation to join the domain.

In this example the *machines* group will need to be created using the `addgroup` utility (see *Security - Users: Adding and Deleting Users* for details).

### Mapping shares

Uncomment the *[homes]* share to allow the `logon home` to be mapped:

```
[homes]
   comment = Home Directories
   browseable = no
   read only = no
   create mask = 0700
```

(continues on next page)

(continued from previous page)

```
   directory mask = 0700
   valid users = %S
```

When configured as a domain controller, a *[netlogon]* share needs to be configured. To enable the share, uncomment:

```
[netlogon]
   comment = Network Logon Service
   path = /srv/samba/netlogon
   guest ok = yes
   read only = yes
   share modes = no
```

**Note**

The original `netlogon` share path is `/home/samba/netlogon`, but according to the *Filesystem Hierarchy Standard (FHS)*, /srv is the correct location for site-specific data provided by the system.

Now create the `netlogon` directory, and an empty (for now) `logon.cmd` script file:

```
sudo mkdir -p /srv/samba/netlogon
sudo touch /srv/samba/netlogon/logon.cmd
```

You can enter any normal Windows logon script commands in `logon.cmd` to customise the client's environment.

Restart Samba to enable the new domain controller, using the following command:

```
sudo systemctl restart smbd.service nmbd.service
```

### Final setup tasks

Lastly, there are a few additional commands needed to set up the appropriate rights.

Since *root* is disabled by default, a system group needs to be mapped to the Windows *Domain Admins* group in order to join a workstation to the domain. Using the `net` utility, from a terminal enter:

```
sudo net groupmap add ntgroup="Domain Admins" unixgroup=sysadmin rid=512 type=d
```

You should change *sysadmin* to whichever group you prefer. Also, the user joining the domain needs to be a member of the *sysadmin* group, as well as a member of the system *admin* group. The *admin* group allows `sudo` use.

If the user does not have Samba credentials yet, you can add them with the `smbpasswd` utility. Change the *sysadmin* username appropriately:

```
sudo smbpasswd -a sysadmin
```

Also, rights need to be explicitly provided to the *Domain Admins* group to allow the *add machine script* (and other admin functions) to work. This is achieved by executing:

```
net rpc rights grant -U sysadmin "EXAMPLE\Domain Admins" SeMachineAccountPrivilege
\
SePrintOperatorPrivilege SeAddUsersPrivilege SeDiskOperatorPrivilege \
SeRemoteShutdownPrivilege
```

You should now be able to join Windows clients to the Domain in the same manner as joining them to an NT4 domain running on a Windows server.

### Backup domain controller

With a Primary Domain Controller (PDC) on the network it is best to have a Backup Domain Controller (BDC) as well. This will allow clients to authenticate in case the PDC becomes unavailable.

When configuring Samba as a BDC you need a way to sync account information with the PDC. There are multiple ways of accomplishing this; secure copy protocol (SCP), `rsync`, or by using LDAP as the `passdb` backend.

Using LDAP is the most robust way to sync account information, because both domain controllers can use the same information in real time. However, setting up an LDAP server may be overly complicated for a small number of user and computer accounts. See *Samba - OpenLDAP Backend* for details.

First, install `samba` and `libpam-winbind`. From a terminal enter:

```
sudo apt install samba libpam-winbind
```

Now, edit `/etc/samba/smb.conf` and uncomment the following in the *[global]*:

```
workgroup = EXAMPLE
...
security = user
```

In the commented *Domains* uncomment or add:

```
domain logons = yes
domain master = no
```

Make sure a user has rights to read the files in `/var/lib/samba`. For example, to allow users in the *admin* group to SCP the files, enter:

```
sudo chgrp -R admin /var/lib/samba
```

Next, sync the user accounts, using SCP to copy the `/var/lib/samba` directory from the PDC:

```
sudo scp -r username@pdc:/var/lib/samba /var/lib
```

You can replace *username* with a valid username and *pdc* with the *hostname* or IP address of your actual PDC.

Finally, restart samba:

```
sudo systemctl restart smbd.service nmbd.service
```

You can test that your Backup Domain Controller is working by first stopping the Samba daemon on the PDC – then try to log in to a Windows client joined to the domain.

Another thing to keep in mind is if you have configured the `logon home` option as a directory on the PDC, and the PDC becomes unavailable, access to the user's *Home* drive will also be unavailable. For this reason it is best to configure the `logon home` to reside on a separate file server from the PDC and BDC.

### Further reading

- For in depth Samba configurations see the Samba HOWTO Collection.
- The guide is also available in printed format.
- O'Reilly's Using Samba is also a good reference.
- Chapter 4 of the Samba HOWTO Collection explains setting up a Primary Domain Controller.
- Chapter 5 of the Samba HOWTO Collection explains setting up a Backup Domain Controller.
- The Ubuntu Wiki Samba page.

## OpenLDAP backend (legacy)

> **Note**
>
> This section is flagged as *legacy* because nowadays, Samba 4 is best integrated with its own LDAP server in Active Directory mode. Integrating Samba with LDAP as described here covers the NT4 mode, which has been deprecated for many years.

This section covers the integration of Samba with LDAP. The Samba server's role will be that of a "standalone" server and the LDAP directory will provide the authentication layer in addition to containing the user, group, and machine account information that Samba requires in order to function (in any of its 3 possible roles). The pre-requisite is an OpenLDAP server configured with a directory that can accept authentication requests. See *Install LDAP* and *LDAP with Transport Layer Security* for details on fulfilling this requirement. Once those steps are completed, you will need to decide what specifically you want Samba to do for you and then configure it accordingly.

This guide will assume that the LDAP and Samba services are running on the same server and therefore use SASL EXTERNAL authentication whenever changing something under *cn=config*. If that is not your scenario, you will have to run those LDAP commands on the LDAP server.

### Install the software

There are two packages needed when integrating Samba with LDAP: `samba` and `smbldap-tools`.

Strictly speaking, the `smbldap-tools` package isn't needed, but unless you have some other way to manage the various Samba entities (users, groups, computers) in an LDAP context then you should install it.

Install these packages now:

```
sudo apt install samba smbldap-tools
```

### Configure LDAP

We will now configure the LDAP server so that it can accommodate Samba data. We will perform three tasks in this section:

- Import a schema
- Index some entries
- Add objects

### Samba schema

In order for OpenLDAP to be used as a backend for Samba, the DIT will need to use attributes that can properly describe Samba data. Such attributes can be obtained by introducing a Samba LDAP schema. Let's do this now.

The schema is found in the now-installed samba package and is already in the LDIF format. We can import it with one simple command:

```
sudo ldapadd -Q -Y EXTERNAL -H ldapi:/// -f /usr/share/doc/samba/examples/LDAP/
samba.ldif
```

To query and view this new schema:

```
sudo ldapsearch -Q -LLL -Y EXTERNAL -H ldapi:/// -b cn=schema,cn=config
'cn=*samba*'
```

### Samba indices

Now that `slapd` knows about the Samba attributes, we can set up some indices based on them. Indexing entries is a way to improve performance when a client performs a filtered search on the DIT.

Create the file `samba_indices.ldif` with the following contents:

```
dn: olcDatabase={1}mdb,cn=config
changetype: modify
replace: olcDbIndex
olcDbIndex: objectClass eq
olcDbIndex: uidNumber,gidNumber eq
olcDbIndex: loginShell eq
olcDbIndex: uid,cn eq,sub
olcDbIndex: memberUid eq,sub
olcDbIndex: member,uniqueMember eq
olcDbIndex: sambaSID eq
olcDbIndex: sambaPrimaryGroupSID eq
olcDbIndex: sambaGroupType eq
olcDbIndex: sambaSIDList eq
```

(continues on next page)

(continued from previous page)

```
olcDbIndex: sambaDomainName eq
olcDbIndex: default sub,eq
```

Using the `ldapmodify` utility load the new indices:

```
sudo ldapmodify -Q -Y EXTERNAL -H ldapi:/// -f samba_indices.ldif
```

If all went well you should see the new indices when using `ldapsearch`:

```
sudo ldapsearch -Q -LLL -Y EXTERNAL -H \
ldapi:/// -b cn=config olcDatabase={1}mdb olcDbIndex
```

### Adding Samba LDAP objects

Next, configure the `smbldap-tools` package to match your environment. The package comes with a configuration helper script called `smbldap-config`. Before running it, though, you should decide on two important configuration settings in `/etc/samba/smb.conf`:

- **netbios name** How this server will be known. The default value is derived from the server's *hostname*, but truncated at 15 characters.
- **workgroup** The workgroup name for this server, or, if you later decide to make it a domain controller, this will be the domain.

It's important to make these choices now because `smbldap-config` will use them to generate the config that will be later stored in the LDAP directory. If you run `smbldap-config` now and later change these values in `/etc/samba/smb.conf` there will be an inconsistency.

Once you are happy with `netbios name` and `workgroup`, proceed to generate the `smbldap-tools` configuration by running the configuration script which will ask you some questions:

```
sudo smbldap-config
```

Some of the more important ones:

- **workgroup name** Has to match what you will configure in `/etc/samba/smb.conf` later on.
- **ldap suffix** Has to match the LDAP suffix you chose when you configured the LDAP server.
- **other ldap suffixes** They are all relative to `ldap suffix` above. For example, for `ldap user suffix` you should use `ou=People`, and for computer/machines, use `ou=Computers`.
- **ldap master bind dn** and **bind password** Use the Root *DN* credentials.

The `smbldap-populate` script will then add the LDAP objects required for Samba. It will ask you for a password for the "domain root" user, which is also the "root" user stored in LDAP:

```
sudo smbldap-populate -g 10000 -u 10000 -r 10000
```

The `-g`, `-u` and `-r` parameters tell `smbldap-tools` where to start the numeric `uid` and *gid* allocation for the LDAP users. You should pick a range start that does not overlap with your local `/etc/passwd` users.

You can create a LDIF file containing the new Samba objects by executing `sudo smbldap-populate -e samba.ldif`. This allows you to look over the changes making sure everything is correct. If it is, rerun the script without the `'-e'` switch. Alternatively, you can take the LDIF file and import its data as per usual.

Your LDAP directory now has the necessary information to authenticate Samba users.

### Samba configuration

To configure Samba to use LDAP, edit its configuration file `/etc/samba/smb.conf` commenting out the default `passdb backend` parameter and adding some LDAP-related ones. Make sure to use the same values you used when running `smbldap-populate`:

```
#  passdb backend = tdbsam
workgroup = EXAMPLE

# LDAP Settings
passdb backend = ldapsam:ldap://ldap01.example.com
ldap suffix = dc=example,dc=com
ldap user suffix = ou=People
ldap group suffix = ou=Groups
ldap machine suffix = ou=Computers
ldap idmap suffix = ou=Idmap
ldap admin dn = cn=admin,dc=example,dc=com
ldap ssl = start tls
ldap passwd sync = yes
```

Change the values to match your environment.

> **Note**
>
> The `smb.conf` as shipped by the package is quite long and has many configuration examples. An easy way to visualise it without any comments is to run `testparm -s`.

Now inform Samba about the Root DN user's password (the one set during the installation of the `slapd` package):

```
sudo smbpasswd -W
```

As a final step to have your LDAP users be able to connect to Samba and authenticate, we need these users to also show up in the system as "Unix" users. Use SSSD for that as detailed in *Network User Authentication with SSSD*.

Install `sssd-ldap`:

```
sudo apt install sssd-ldap
```

Configure `/etc/sssd/sssd.conf`:

```
[sssd]
config_file_version = 2
domains = example.com
```

(continues on next page)

(continued from previous page)

```
[domain/example.com]
id_provider = ldap
auth_provider = ldap
ldap_uri = ldap://ldap01.example.com
cache_credentials = True
ldap_search_base = dc=example,dc=com
```

Adjust permissions and start the service:

```
sudo chmod 0600 /etc/sssd/sssd.conf
sudo chown root:root /etc/sssd/sssd.conf
sudo systemctl start sssd
```

Restart the Samba services:

```
sudo systemctl restart smbd.service nmbd.service
```

To quickly test the setup, see if `getent` can list the Samba groups:

```
$ getent group Replicators
Replicators:*:552:
```

**Note**

The names are case sensitive!

If you have existing LDAP users that you want to include in your new LDAP-backed Samba they will, of course, also need to be given some of the extra Samba specific attributes. The `smbpasswd` utility can do this for you:

```
sudo smbpasswd -a username
```

You will be prompted to enter a password. It will be considered as the new password for that user. Making it the same as before is reasonable. Note that this command cannot be used to create a new user from scratch in LDAP (unless you are using `ldapsam:trusted` and `ldapsam:editposix`, which are not covered in this guide).

To manage user, group, and machine accounts use the utilities provided by the `smbldap-tools` package. Here are some examples:

- To add a new user with a home directory:

  ```
  sudo smbldap-useradd -a -P -m username
  ```

  The `-a` option adds the Samba attributes, and the `-P` option calls the `smbldap-passwd` utility after the user is created allowing you to enter a password for the user. Finally, `-m` creates a local home directory. Test with the `getent` command:

  ```
  getent passwd username
  ```

- To remove a user:

```
sudo smbldap-userdel username
```

In the above command, use the `-r` option to remove the user's home directory.

- To add a group:

```
sudo smbldap-groupadd -a groupname
```

As for *smbldap-useradd*, the *-a* adds the Samba attributes.

- To make an existing user a member of a group:

```
sudo smbldap-groupmod -m username groupname
```

The `-m` option can add more than one user at a time by listing them in comma-separated format.

- To remove a user from a group:

```
sudo smbldap-groupmod -x username groupname
```

- To add a Samba machine account:

```
sudo smbldap-useradd -t 0 -w username
```

Replace `username` with the name of the workstation. The `-t 0` option creates the machine account without a delay, while the `-w` option specifies the user as a machine account.

### Resources

- Upstream documentation collection
- Upstream samba wiki

### See also

- Explanation: *Introduction to Samba*

### Active Directory integration

If you have a Microsoft Active Directory domain set up, you can join your Ubuntu Server to it. There are a number of choices you need to make about your setup before you can begin, so you may want to refer to our *AD integration explanations* first.

### Join a domain with winbind: preparation

Choosing the identity mapping backend, and planning its ranges, is the first and most important aspect of joining a domain. To actually perform the join, however, a few more configuration steps are necessary. These steps are common to both backend types, the only difference being the actual idmap configuration.

To continue, this is the minimum set of packages that are needed:

```
sudo apt install winbind libnss-winbind libpam-winbind
```

Next, it will make everything much easier if the *DNS* resolver is pointed at the Active Directory DNS server. If that is already the case as provided by the *DHCP* server, this part can be skipped.

For example, for a default netplan configuration file which looks like this:

```
    network:
    version: 2
    ethernets:
        eth0:
            dhcp4: true
```

You can add a `nameservers` block which will override the DNS options sent by the DHCP server. For example, if the DNS server is at `10.10.4.5` and the domain search value is `example.internal`, this would be the new configuration:

```
    network:
    version: 2
    ethernets:
        eth0:
            dhcp4: true
            nameservers:
                addresses: [10.10.4.5]
                search: [example.internal]
```

To make the changes effective, first make sure there are no syntax errors:

```
sudo netplan generate
```

If there are no complaints, the changes can be applied:

```
sudo netplan apply
```

**Note**

Be careful whenever changing network parameters over an ssh connection. If there are any mistakes, you might lose remote access!

To check if the resolver was updated, run `resolvectl status`:

```
Global
         Protocols: -LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported
  resolv.conf mode: stub

Link 281 (eth0)
    Current Scopes: DNS
         Protocols: +DefaultRoute -LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported
       DNS Servers: 10.10.4.5 10.10.4.1
        DNS Domain: example.internal
```

Now we need to configure the system to also use the winbind NSS module to look for users and groups. In Ubuntu 24.04 LTS and later, this is done automatically, but for older LTS releases, edit the file `/etc/nsswitch.conf` and add `winbind` to the end of the `passwd:` and `group:` lines:

```
# /etc/nsswitch.conf
#
# Example configuration of GNU Name Service Switch functionality.
# If you have the `glibc-doc-reference' and `info' packages installed, try:
# `info libc "Name Service Switch"' for information about this file.

passwd:         files systemd winbind
group:          files systemd winbind
(...)
```

Finally, let's enable automatic home directory creation for users as they login. Run the command:

```
sudo pam-auth-update --enable mkhomedir
```

Now we are set to perform the final winbind configuration depending on the identity mapping backend that was chosen, and actually join the domain.

### Join a simple domain with the rid backend

Let's expand on the configuration we had for the *rid* backend and complete the `/etc/samba/smb.conf` configuration file with the remaining details. We are joining a single domain called `EXAMPLE.INTERNAL`. The new configuration options were added at the end of the `[global]` section:

```
[global]
    security = ads
    realm = EXAMPLE.INTERNAL
    workgroup = EXAMPLE

    idmap config * : backend       = tdb
    idmap config * : range         = 100000 - 199999
    idmap config EXAMPLE : backend = rid
    idmap config EXAMPLE : range   = 1000000 - 1999999

    # allow logins when the DC is unreachable
    winbind offline logon = yes
    # this *can* be yes if there is absolute certainty that there is only a
    # single domain involved
    winbind use default domain = no
    # setting these enumeration options to yes has a high performance impact
    # and can cause instabilities
    winbind enum groups = no
    winbind enum users = no
    winbind refresh tickets = yes
    # if domain users should be allowed to login, they will need a login shell
```

(continues on next page)

(continued from previous page)

```
    template shell = /bin/bash
    # the home directory template for domain users
    template homedir = /home/%D/%U
    kerberos method = secrets and keytab
```

Right after saving `/etc/samba/smb.conf`, it's always good practice to run the `testparm` utility. It will perform a quick syntax check on the configuration file and alert you of any issues. Here is the output we get with the above configuration settings:

```
Load smb config files from /etc/samba/smb.conf
Loaded services file OK.
Weak crypto is allowed by GnuTLS (e.g. NTLM as a compatibility fallback)

Server role: ROLE_DOMAIN_MEMBER

Press enter to see a dump of your service definitions
(...)
```

During the domain join process, the tooling will attempt to update the *DNS* server with the *hostname* of this system. Since its IP is likely not yet registered in DNS, that's kind of a chicken and egg problem. It helps to, beforehand, set the hostname manually to the *FQDN*. For this example, we will use a host named `n1` in the `example.internal` domain:

```
sudo hostnamectl hostname n1.example.internal
```

So that the output of `hostname -f` (and also just `hostname`) is `n1.example.internal`.

With the config file in place and checked, and all the other changes we made in the previous section, the domain join can be performed:

```
$ sudo net ads join -U Administrator
Password for [EXAMPLE\Administrator]:
Using short domain name -- EXAMPLE
Joined 'N1' to dns domain 'example.internal'
```

You can now revert the `hostnamectl` change from before, and set the hostname back to the short version, i.e., `n1` in this example:

```
sudo hostnamectl hostname n1
```

As the last step of the process, the `winbind` service must be restarted:

```
sudo systemctl restart winbind.service
```

### Verifying the join

The quickest way to test the integrity of the domain join is via the `wbinfo` command:

```
$ sudo wbinfo -t
checking the trust secret for domain EXAMPLE via RPC calls succeeded
```

The next verification step should be to actually try to resolve an existing username from the domain. In the `EXAMPLE.INTERNAL` domain, for example, we have some test users we can check:

```
$ id jammy@example.internal
uid=1001103(EXAMPLE\jammy) gid=1000513(EXAMPLE\domain users)
groups=1000513(EXAMPLE\domain users),1001103(EXAMPLE\jammy)
```

Another valid syntax for domain users is prefixing the name with the domain, like this:

```
$ id EXAMPLE\\jammy
uid=1001103(EXAMPLE\jammy) gid=1000513(EXAMPLE\domain users)
groups=1000513(EXAMPLE\domain users),1001103(EXAMPLE\jammy)
```

And finally, attempt a console login:

```
n1 login: jammy@example.internal
Password:
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.5.0-26-generic x86_64)
(...)
Creating directory '/home/EXAMPLE/jammy'.
EXAMPLE\jammy@n1:~$
```

The output above also shows the automatic on-demand home directory creation, according to the template defined in `/etc/samba/smb.conf`.

**Note**

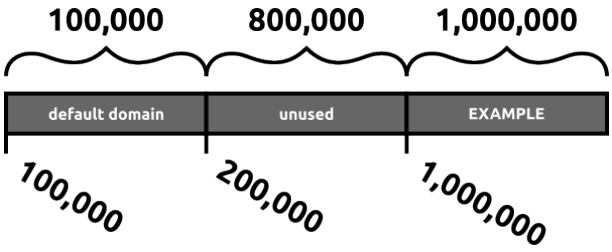
The actual login name used can have multiple formats: `DOMAIN\user` at the terminal login prompt, `DOMAIN\\user` when referred to in shell scripts (note the escaping of the '\' character), and `user@domain` is also accepted.

### Join a forest with the rid backend

It's also possible to join an Active Directory forest using the *rid* identity mapping backend. To better understand what is involved, and why it is tricky, let's reuse the example where we joined a single domain with this backend:

```
[global]
    security = ads
    realm = EXAMPLE.INTERNAL
    workgroup = EXAMPLE

    idmap config * : backend       = tdb
    # 100,000 - 199,999
    idmap config * : range         = 100000 - 199999
    idmap config EXAMPLE : backend = rid
    # 1,000,000 - 1,999,999
    idmap config EXAMPLE : range   = 1000000 - 1999999
```




With this configuration, we are expected to join the *EXAMPLE.INTERNAL* domain, and have given it a range of 1 million IDs starting with the ID `1000000` (1,000,000). There is also the mandatory reserved range for the default domain, represented by the identity mapping configuration for "*", which has a smaller range of 100,000 IDs, starting at `100000` (100,000).

The `testparm` utility is happy with this configuration, and there is no overlap of ID ranges:

```
$ testparm
Load smb config files from /etc/samba/smb.conf
Loaded services file OK.
Weak crypto is allowed by GnuTLS (e.g. NTLM as a compatibility fallback)

Server role: ROLE_DOMAIN_MEMBER
```

We next adjust the *hostname* and perform the join:

```
$ sudo hostnamectl hostname n3.example.internal

$ hostname
n3.example.internal

$ hostname -f
n3.example.internal

$ sudo net ads join -U Administrator
Password for [EXAMPLE\Administrator]:
Using short domain name -- EXAMPLE
Joined 'N3' to dns domain 'example.internal'

$ sudo hostnamectl hostname n3
$ sudo systemctl restart winbind.service
```

A quick check shows that the users from *EXAMPLE.INTERNAL* are recognized:

```
$ id jammy@example.internal
uid=1001103(EXAMPLE\jammy) gid=1000513(EXAMPLE\domain users)
groups=1000513(EXAMPLE\domain users),1001103(EXAMPLE\jammy)
```

But what happens if this single domain establishes a trust relationship with another domain, and we don't modify the `/etc/samba/smb.conf` file to cope with that? Where will the users from the new trusted domain get their IDs from? Since there is no specific idmap configuration for the new trusted domain, its users will get IDs from the default domain:

```
$ id noble@mydomain.internal
uid=100000(MYDOMAIN\noble) gid=100000(MYDOMAIN\domain users)
```

(continues on next page)

(continued from previous page)

```
groups=100000(MYDOMAIN\domain users)
```

Oops. That is from the much smaller range 100,000 - 199,999, reserved for the catch-all default domain. Furthermore, if yet another trust relationship is established, those users will also get their IDs from this range, mixing multiple domains up in the same ID range, in whatever order they are being looked up.

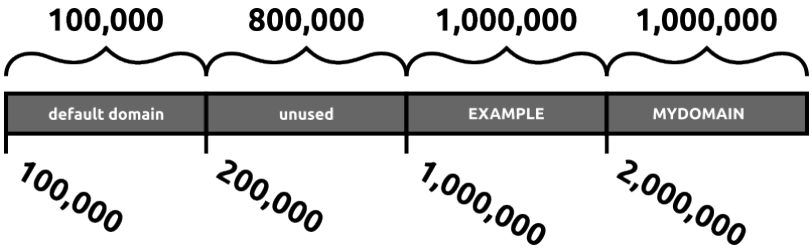
If above we had looked up another user instead of *noble@mydomain.internal*, that other user would have been given the ID 100000. There is no deterministic formula for the default domain ID allocation, like there is for the *rid* backend. In the default domain, IDs are allocated on a first come, first serve basis.

To address this, we can add another *idmap config* configuration for the *rid* backend, giving the new domain a separate range:

```
[global]
    security = ads
    realm = EXAMPLE.INTERNAL
    workgroup = EXAMPLE

    idmap config * : backend       = tdb
    # 100,000 - 199,999
    idmap config * : range         = 100000 - 199999
    idmap config EXAMPLE : backend = rid
    # 1,000,000 - 1,999,999
    idmap config EXAMPLE : range   = 1000000 - 1999999

    # MYDOMAIN.INTERNAL idmap configuration
    idmap config MYDOMAIN : backend = rid
    # 2,000,000 - 2,999,999
    idmap config MYDOMAIN : range   = 2000000 - 2999999
```




With this configuration, nothing changed for the *EXAMPLE.INTERNAL* users, as expected:

```
$ id jammy@example.internal
uid=1001103(EXAMPLE\jammy) gid=1000513(EXAMPLE\domain users)
groups=1000513(EXAMPLE\domain users),1001103(EXAMPLE\jammy)
```

But the users from the trusted domain *MYDOMAIN.INTERNAL* will get their IDs allocated from the 2,000,000 - 2,999,999 range, instead of the default one:

```
$ id noble@mydomain.internal
uid=2001104(MYDOMAIN\noble) gid=2000513(MYDOMAIN\domain users)
groups=2000513(MYDOMAIN\domain users),2001104(MYDOMAIN\noble)
```

And this allocation, which is using the *rid* backend, is deterministic.

Problem solved! Well, until another trust relationship is established, then we have to allocate another range for it.

So is it possible to use the *rid* identity mapping backend with an Active Directory forest, with multiple domains? Yes, but following these steps **BEFORE** establishing any new trust relationship:

- plan an ID range for the new domain
- update **ALL** systems that are using the *rid* idmap backend with the new range configuration for the new domain
- restart winbind on **ALL** such systems
- then the new trust relationship can be established

If a system is missed, and doesn't have an idmap configuration entry for the new domain, the moment a user from that new domain is looked up, it will be assigned an ID from the default domain, which will be non-deterministic and different from the ID assigned to the same user in another system which had the new idmap configuration entry. Quite a mess. Unless the different ID is not important, in which case it's much simpler to just use the *autorid* identity mapping backend.

### Join a forest with the autorid backend

Joining a more complex Active Directory forest with the autorid backend is very similar to the rid backend. The only difference is in the idmap configuration in `/etc/samba/smb.conf`:

```
[global]
    security = ads
    realm = EXAMPLE.INTERNAL
    workgroup = EXAMPLE

    idmap config * : backend = autorid
    # 1,000,000 - 19,999,999
    idmap config * : range   = 1000000 - 19999999
    # 1,000,000
    idmap config * : rangesize = 1000000

    # allow logins when the DC is unreachable
    winbind offline logon = yes
    # this *can* be yes if there is absolute certainty that there is only a
    # single domain involved
    winbind use default domain = no
    # setting these enumeration options to yes has a high performance impact
    # and can cause instabilities
    winbind enum groups = no
    winbind enum users = no
    winbind refresh tickets = yes
    # if domain users should be allowed to login, they will need a login shell
    template shell = /bin/bash
    # the home directory template for domain users
```

(continues on next page)

(continued from previous page)

```
    template homedir = /home/%D/%U
    kerberos method = secrets and keytab
```

Note that there is no specific domain mentioned in the idmap configuration. That's because the autorid backend does the allocations on demand, according to the defined slots. The configuration above defines the following:

- 1 million IDs per slot
- 19 slots (or domains)
- full ID range, covering all slots, is from 1,000,000 to 19,999,999

That being said, the machine still needs to be joined to a specific domain of that forest, and in this example that will be *EXAMPLE.INTERNAL*.

Running the recommended `testparm` command gives us confidence that the configuration is at least free from syntax and other logical errors:

```
$ testparm
Load smb config files from /etc/samba/smb.conf
Loaded services file OK.
Weak crypto is allowed by GnuTLS (e.g. NTLM as a compatibility fallback)

Server role: ROLE_DOMAIN_MEMBER

Press enter to see a dump of your service definitions
```

Like with the *rid* idmap backend, if this system is not yet in the AD *DNS* server, it's best to change its *hostname* (including the short hostname) to be the fully qualified domain name (FQDN), as that will allow the joining procedure to also update the DNS records, if so allowed by the AD server (normally it is).

For this example, the system's hostname is `n2` in the `example.internal` domain, so the FQDN is `n2.example.internal`:

```
sudo hostnamectl hostname n2.example.internal
```

Now the domain join can be performed:

```
$ sudo net ads join -U Administrator
Password for [EXAMPLE\Administrator]:
Using short domain name -- EXAMPLE
Joined 'N2' to dns domain 'example.internal'
```

And we can revert the hostname change:

```
sudo hostnamectl hostname n2
```

If the DNS server was updated correctly (and there were no errors about that in the join output above), then the hostname should now be correctly set, even though we have just the short name in `/etc/hostname`:

```
$ hostname
n2

$ hostname -f
n2.example.internal
```

The last step is to restart the *winbind* service:

```
sudo systemctl restart winbind.service
```

### Verifying the join

The quickest way to test the integrity of the domain join is via the *wbinfo* command:

```
$ sudo wbinfo -t
checking the trust secret for domain EXAMPLE via RPC calls succeeded
```

The next verification step should be to actually try to resolve an existing username from the domain. In the *EXAMPLE.INTERNAL* domain, for example, we have some test users we can check:

```
$ id jammy@example.internal
uid=2001103(EXAMPLE\jammy) gid=2000513(EXAMPLE\domain users)
groups=2000513(EXAMPLE\domain users),2001103(EXAMPLE\jammy)
```

If you compare this with the *rid* domain join, note how the ID that the *jammy* user got is different. That's why it's important to correctly chose an idmap backend, and correctly assess if deterministic IDs are important for your use case or not.

Another valid syntax for domain users is prefixing the name with the domain, like this:

```
$ id EXAMPLE\\jammy
uid=2001103(EXAMPLE\jammy) gid=2000513(EXAMPLE\domain users)
groups=2000513(EXAMPLE\domain users),2001103(EXAMPLE\jammy)
```

And here we try a console login:

```
n2 login: jammy@example.internal
Password:
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.5.0-26-generic x86_64)
(...)
Creating directory '/home/EXAMPLE/jammy'.
EXAMPLE\jammy@n1:~$
```

The output above also shows the automatic on-demand home directory creation, according to the template defined in `/etc/samba/smb.conf`.

Since we joined a forest, we should also be able to verify users from other domains in that forest. For example, in this example, the domain *MYDOMAIN.INTERNAL* is also part of the forest, and we can verify its users:

```
$ id noble@mydomain.internal
uid=3001104(MYDOMAIN\noble) gid=3000513(MYDOMAIN\domain users)
groups=3000513(MYDOMAIN\domain users),3001104(MYDOMAIN\noble)

$ id MYDOMAIN\\noble
uid=3001104(MYDOMAIN\noble) gid=3000513(MYDOMAIN\domain users)
groups=3000513(MYDOMAIN\domain users),3001104(MYDOMAIN\noble)
```

A console login also works:

```
n2 login: noble@mydomain.internal
Password:
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-31-generic x86_64)
(...)
Creating directory '/home/MYDOMAIN/noble'.
MYDOMAIN\noble@n2:~$
```

Notice how the domain name being part of the home directory path is useful: it separates the users from different domains, avoiding collisions for the same username.

**Note**

The actual login name used can have multiple formats: `DOMAIN\user` at the terminal login prompt, `DOMAIN\\user` when refered to in shell scripts (note the escaping of the '\' character), and `user@domain` is also accepted.

### See also

- Explanation: *Introduction to Active Directory integration*
- *Samba*
- *Active Directory integration*

## Printing

The Common UNIX Printing System (CUPS) is the most common way to manage print services in Ubuntu.

### Install and configure a CUPS print server

The Common UNIX Printing System, or CUPS, is the most widely-used way to manage printing and print services in Ubuntu. This freely-available printing system has become the standard for printing in most Linux distributions, and uses the standard Internet Printing Protocol (IPP) to handle network printing.

CUPS manages print jobs and queues, and provides support for a wide range of printers, from dot-matrix to laser, and many in between. CUPS also supports PostScript Printer Description (PPD) and auto-detection of network printers, and features a simple web-based configuration and administration tool.

### Install CUPS

A complete CUPS install has many package dependencies, but they can all be specified on the same command line. To perform a basic installation of CUPS, enter the following command in your terminal:

```
sudo apt install cups
```

Once the download and installation have finished, the CUPS server will be started automatically.

### Configure the CUPS server

The CUPS server's behavior is configured through directives found in the `/etc/cups/cupsd.conf` configuration file. This CUPS configuration file follows the same syntax as the main configuration file for the Apache HTTP server. Some examples of commonly-configured settings will be presented here.

### Make a copy of the configuration file

We recommend that you make a copy of the original CUPS configuration file and protect it from writing, before you start configuring CUPS. You will then have the original settings as a reference, which you can reuse or restore as necessary.

```
sudo cp /etc/cups/cupsd.conf /etc/cups/cupsd.conf.original
sudo chmod a-w /etc/cups/cupsd.conf.original
```

### Configure Server administrator

To configure the email address of the designated CUPS server administrator, edit the `/etc/cups/cupsd.conf` configuration file with your preferred text editor, and add or modify the **ServerAdmin** line accordingly. For example, if you are the administrator for the CUPS server, and your e-mail address is `bjoy@somebigco.com`, then you would modify the ServerAdmin line to appear as follows:

```
ServerAdmin bjoy@somebigco.com
```

### Configure Listen

By default on Ubuntu, CUPS listens only on the loopback interface at IP address `127.0.0.1`.

To instruct CUPS to listen on an actual network adapter's IP address, you must specify either a *hostname*, the IP address, or (optionally) an IP address/port pairing via the addition of a **Listen** directive.

For example, if your CUPS server resides on a local network at the IP address `192.168.10.250` and you'd like to make it accessible to the other systems on this subnetwork, you would edit the `/etc/cups/cupsd.conf` and add a Listen directive, as follows:

```
Listen 127.0.0.1:631          # existing loopback Listen
Listen /var/run/cups/cups.sock # existing socket Listen
Listen 192.168.10.250:631     # Listen on the LAN interface, Port 631 (IPP)
```

In the example above, you can comment out or remove the reference to the Loopback address (`127.0.0.1`) if you do not want the CUPS daemon (`cupsd`) to listen on that interface, but would rather have it only listen on the Ethernet interfaces of the Local Area Network (LAN). To enable listening for all network interfaces for which a certain hostname is bound, including the Loopback, you could create a Listen entry for the hostname `socrates` like this:

```
Listen socrates:631  # Listen on all interfaces for the hostname 'socrates'
```

or by omitting the Listen directive and using **Port** instead, as in:

```
Port 631  # Listen on port 631 on all interfaces
```

For more examples of configuration directives in the CUPS server configuration file, view the associated system manual page by entering the following command:

```
man cupsd.conf
```

### Post-configuration restart

Whenever you make changes to the `/etc/cups/cupsd.conf` configuration file, you'll need to restart the CUPS server by typing the following command at a terminal prompt:

```
sudo systemctl restart cups.service
```

### Web Interface

CUPS can be configured and monitored using a web interface, which by default is available at `http://localhost:631/admin`. The web interface can be used to perform all printer management tasks.

To perform administrative tasks via the web interface, you must either have the root account enabled on your server, or authenticate as a user in the `lpadmin` group. For security reasons, CUPS won't authenticate a user that doesn't have a password.

To add a user to the `lpadmin` group, run at the terminal prompt:

```
sudo usermod -aG lpadmin username
```

Further documentation is available in the "Documentation/Help" tab of the web interface.

### Error logs

For troubleshooting purposes, you can access CUPS server errors via the error log file at: `/var/log/cups/error_log`. If the error log does not show enough information to troubleshoot any problems you encounter, the verbosity of the CUPS log can be increased by changing the **LogLevel** directive in the configuration file (discussed above) from the default of "info" to "debug" or even "debug2", which logs everything.

If you make this change, remember to change it back once you've solved your problem, to prevent the log file from becoming overly large.

### References

- CUPS Website
- Debian Open-iSCSI page

### See also

- Explanation: *Networking section*

*Our networking section* is where you will find how-to guides on a broad range of networking topics, such as:

- **Network tooling and configuration** including time synchronisation, DHCP for IP address assignment, Domain Name Service (DNS) (and more!)
- **Network shares** for sharing resources (files, services, directories) across networks, including integration with Windows

# 2.4. Managing software

## 2.4.1. Managing software

### Install and manage packages

The recommended way to install Debian packages ("deb" files) is using the Advanced Packaging Tool (APT), which can be used on the command line using the `apt` utility.

The commands contained within `apt` provide the means to install new software packages, upgrade existing software packages, update the package list index, and even upgrade the entire Ubuntu system.

### Update the package index

The APT package index is a database of available packages from the repositories defined in the `/etc/apt/sources.list` file and in the `/etc/apt/sources.list.d` directory. To update the local package index with the latest changes made in the repositories, and thereby access the most up-to-date version of the package you're interested in, type the following:

```
sudo apt update
```

### Install a package

As an example, to install the `nmap` network scanner, run the following command:

```
sudo apt install nmap
```

> **Tip**
>
> You can install or remove multiple packages at once by separating them with spaces.

### Remove a package

To remove the package installed in the previous example, run the following:

```
sudo apt remove nmap
```

Adding the `--purge` option to `apt remove` will remove the package configuration files as well. This may or may not be what you want, so use it with caution.

> **Note**
>
> While `apt` is a command-line tool, it is intended to be used interactively, and not to be called from non-interactive scripts. The `apt-get` command should be used in scripts (perhaps with the `--quiet` flag). For basic commands the syntax of the two tools is identical.

### Upgrading packages

Installed packages on your computer may periodically have upgrades available from the package repositories (e.g., security updates). To upgrade your system, first update your package index and then perform the upgrade – as follows:

```
sudo apt update
sudo apt upgrade
```

For details on how to upgrade to a new Ubuntu release, see our *guide on upgrading releases*. For further information about using APT, read the comprehensive APT User's Guide, or type `apt help`.

### Aptitude

Launching Aptitude with no command-line options will give you a menu-driven, text-based *frontend* to the APT system. Many of the common package management functions, such as installation, removal, and upgrade, can be performed in Aptitude with single-key commands, which are typically lowercase letters.

Aptitude is best suited for use in a non-graphical terminal environment to ensure the command keys work properly. You can start the menu-driven interface of Aptitude as a regular user by typing the following command at a terminal prompt:

```
sudo aptitude
```

When Aptitude starts, you will see a menu bar at the top of the screen and two panes below the menu bar. The top pane contains package categories, such as "New Packages" and "Not Installed Packages". The bottom pane contains information related to the packages and package categories.

Using Aptitude for package management is relatively straightforward thanks to its user interface. The following are examples of common package management functions as performed in Aptitude:

### Installing packages

To install a package, locate it in the “Not Installed Packages” category by using the keyboard arrow keys and the Enter key.

Highlight the desired package, then press the + key. The package entry should turn **green**, which indicates it has been marked for installation. Now press g to be presented with a summary of package actions. Press g again, and the package will be downloaded and installed. When finished, press Enter to return to the menu.

### Remove Packages

To remove a package, locate it in the “Installed Packages” category by using the keyboard arrow keys and the Enter key.

Highlight the package you want to remove, then press the - key. The package entry should turn **pink**, indicating it has been marked for removal. Now press g to be presented with a summary of package actions. Press g again, and the package will be removed. When finished, press Enter to return to the menu.

### Updating the package index

To update the package index, press the u key.

### Upgrade packages

To upgrade packages, first update the package index as detailed above, and then press the U key to mark all packages with available updates. Now press g, which will present you with a summary of package actions. Press g again to begin the download and installation. When finished, press Enter to return to the menu.

The first column of information displayed in the package list (in the top pane) lists the current state of the package (when viewing packages). It uses the following key to describe the package state:

- **i** : Installed package
- **c** : Package not installed, but package configuration remains on the system
- **p** : Purged from system
- **v** : Virtual package
- **B** : Broken package
- **u** : Unpacked files, but package not yet configured
- **C** : Half-configured - configuration failed and requires fix
- **H** : Half-installed - removal failed and requires a fix

To exit Aptitude, simply press the q key and confirm you want to exit. Many other functions are available from the Aptitude menu by pressing the F10 key.

### Command-line Aptitude

You can also use Aptitude as a command-line tool, similar to `apt`. To install the `nmap` package with all necessary dependencies (as in the `apt` example), you would use the following command:

```
sudo aptitude install nmap
```

To remove the same package, you would use the command:

```
sudo aptitude remove nmap
```

Consult the `Aptitude manpages` for full details of Aptitude's command-line options.

### dpkg

`dpkg` is a package manager for Debian-based systems. It can install, remove, and build packages, but unlike other package management systems, it cannot automatically download and install packages – or their dependencies.

APT and Aptitude are newer, and layer additional features on top of `dpkg`. This section covers using `dpkg` to manage locally installed packages.

### List packages

To list *all* packages in the system's package database (both installed and uninstalled) run the following command from a terminal prompt:

```
dpkg -l
```

Depending on the number of packages on your system, this can generate a large amount of output. Pipe the output through `grep` to see if a specific package is installed:

```
dpkg -l | grep apache2
```

Replace `apache2` with any package name, part of a package name, or a regular expression.

### List files

To list the files installed by a package, in this case the `ufw` package, enter:

```
dpkg -L ufw
```

If you are unsure which package installed a file, `dpkg -S` may be able to tell you. For example:

```
dpkg -S /etc/host.conf
base-files: /etc/host.conf
```

The output shows that the `/etc/host.conf` belongs to the base-files package.

**Note**

Many files are automatically generated during the package install process, and even though they are on the *filesystem*, `dpkg -S` may not know which package they belong to.

### Installing a deb file

You can install a local `.deb` file by entering:

```
sudo dpkg -i zip_3.0-4_amd64.deb
```

Change `zip_3.0-4_amd64.deb` to the actual file name of the local `.deb` file you wish to install.

### Uninstalling packages

You can uninstall a package by running:

```
sudo dpkg -r zip
```

**Caution**

Uninstalling packages using `dpkg`, is **NOT** recommended in most cases. It is better to use a package manager that handles dependencies to ensure that the system is left in a consistent state. For example, using `dpkg -r zip` will remove the `zip` package, but any packages that depend on it will still be installed and may no longer function correctly as a result.

For more `dpkg` options see the `dpkg(1)` manpage: `man dpkg`.

### APT configuration

Configuration of the APT system repositories is stored in the `/etc/apt/sources.list` file and the `/etc/apt/sources.list.d` directory. An example of this file is referenced here, along with information on adding or removing repository references from the file.

You can edit the file to enable and disable repositories. For example, to disable the requirement to insert the Ubuntu CD-ROM whenever package operations occur, simply comment out the appropriate line for the CD-ROM, which appears at the top of the file:

```
# no more prompting for CD-ROM please
# deb cdrom:[DISTRO-APT-CD-NAME - Release i386 (20111013.1)]/ DISTRO-SHORT-
CODENAME main restricted
```

### Automatic updates

It's possible to setup an Ubuntu system with Automatic Updates, such that certain types of upgrades are applied automatically. In fact, the default for Ubuntu Server is to automatically apply security updates. Please see the *Automatic updates* section for details.

### Extra repositories

In addition to the officially-supported package repositories available for Ubuntu, there are also community-maintained repositories which add thousands more packages for potential installation. Two of the most popular are the *Universe* and *Multiverse* repositories. These repositories are not officially supported by Ubuntu, but because they are maintained by the community they generally provide packages which are safe for use with your Ubuntu computer.

For more information, see our guide on *using third-party repositories*.

**Warning**

Be advised that packages in Universe and Multiverse are not officially supported and do not receive security patches, except through Ubuntu Pro's Expanded Security Maintenance. A subscription to Ubuntu Pro is free for personal use on up to five machines.

> Packages in the *multiverse* repository often have licensing issues that prevent them from being distributed with a free operating system, and they may be illegal in your locality.

Many other package sources are available – sometimes even offering only one package, as in the case of packages provided by the developer of a single application. You should always be cautious when using non-standard package sources/repos, however. Research the packages and their origins carefully before performing any installation, as some packages could render your system unstable or non-functional in some respects.

By default, the *universe* and *multiverse* repositories are enabled. If you would like to disable them, edit `/etc/apt/sources.list` and comment out the following lines:

```
deb http://archive.ubuntu.com/ubuntu DISTRO-SHORT-CODENAME universe multiverse
deb-src http://archive.ubuntu.com/ubuntu DISTRO-SHORT-CODENAME universe multiverse

deb http://us.archive.ubuntu.com/ubuntu/ DISTRO-SHORT-CODENAME universe
deb-src http://us.archive.ubuntu.com/ubuntu/ DISTRO-SHORT-CODENAME universe
deb http://us.archive.ubuntu.com/ubuntu/ DISTRO-SHORT-CODENAME-updates universe
deb-src http://us.archive.ubuntu.com/ubuntu/ DISTRO-SHORT-CODENAME-updates
universe

deb http://us.archive.ubuntu.com/ubuntu/ DISTRO-SHORT-CODENAME multiverse
deb-src http://us.archive.ubuntu.com/ubuntu/ DISTRO-SHORT-CODENAME multiverse
deb http://us.archive.ubuntu.com/ubuntu/ DISTRO-SHORT-CODENAME-updates multiverse
deb-src http://us.archive.ubuntu.com/ubuntu/ DISTRO-SHORT-CODENAME-updates
multiverse

deb http://security.ubuntu.com/ubuntu DISTRO-SHORT-CODENAME-security universe
deb-src http://security.ubuntu.com/ubuntu DISTRO-SHORT-CODENAME-security universe
deb http://security.ubuntu.com/ubuntu DISTRO-SHORT-CODENAME-security multiverse
deb-src http://security.ubuntu.com/ubuntu DISTRO-SHORT-CODENAME-security
multiverse
```

### logging

Actions of the `apt` command, such as installation and removal of packages, are logged in the `/var/log/dpkg.log` log file.

### Further reading

Most of the material covered in this chapter is available in the respective man pages, many of which are available online.

- The Installing Software Ubuntu wiki page has more information.
- The APT User's Guide contains useful information regarding APT usage.
- For more information about systemd timer units (and systemd in general), visit the `systemd(1)` manual page and `systemd.timer(5)` manual page
- See the Aptitude user's manual for more Aptitude options.
- The Adding Repositories HOWTO (Ubuntu Wiki) page contains more details on adding repositories.
- *Package management* shows you how to manage the software on your system using APT, dpkg, and other package managers

## Updates

### Automatic updates

Ubuntu will apply security updates automatically, without user interaction. This is done via the `unattended-upgrades` package, which is installed by default.

But as the name suggests, it can apply other types of updates, and with interesting options alongside. For example:

- It can reboot the system, if an update warrants it.
- It can apply other types of updates, not just security.
- It can block certain updates from ever being applied automatically.

And more. Let's explore some of these options.

> **Important**
>
> Just adding another package repository to an Ubuntu system WILL NOT make `unattended-upgrades` consider it for updates! This is explained in *where to pick updates from* later in this document.

### Configuration layout

If for some reason the package is not present, it can be installed with the following command:

```
sudo apt install unattended-upgrades
```

Important files and directories:

- `/etc/apt/apt.conf.d/50unattended-upgrades`: This file contains the options that control the behavior of the tool, such as if a reboot should be scheduled or not, or which packages are blocked from being upgraded.
- `/etc/apt/apt.conf.d/20auto-upgrades`: This file is used to control whether the unattended upgrade should be enabled or not, and how often it should run.
- `/var/log/unattended-upgrades`: This directory contains detailed logs of each unattended upgrade run.

Right after installation, automatic installation of security updates will be enabled, including Expanded Security Maintenance (ESM) if that is available on the system. By default, `unattended-upgrades` runs once per day.

### Enabling and disabling unattended upgrades

Unattended upgrades performs the equivalent of `apt update` and `apt upgrade` (see *Upgrading packages* for details on these commands. First, it refreshes the package lists, to become aware of the new state of the package repositories. Then it checks which upgrades are available and applies them.

These two steps are controlled via the `Update-Package-Lists` and `Unattended-Upgrade` options in `/etc/apt/apt.conf.d/20auto-upgrades`:

```
APT::Periodic::Update-Package-Lists "1";
APT::Periodic::Unattended-Upgrade "1";
```

Each option accepts a time-based value, representing the number of days. A value of `0` disables the action. The default value, `1`, executes the action daily. A value of `2` executes it every two days, and so forth.

Therefore, to disable unattended upgrades, set these options to zero:

```
APT::Periodic::Update-Package-Lists "0";
APT::Periodic::Unattended-Upgrade "0";
```

Systemd timer units, `apt-daily.timer` and `apt-daily-upgrade.timer`, trigger these actions at a scheduled time with a random delay. These timers activate services that execute the `/usr/lib/apt/apt.systemd.daily` script.

However, it may happen that if the machine is off at the time the timer unit elapses, the timer may be triggered immediately at the next startup (still subject to the `RandomizedDelaySec` value). As a result, `unattended-upgrades` may often run on system startup and thereby cause immediate activity and prevent other package operations from taking place at that time. For example, if another package has to be installed, it would have to wait until the upgrades are completed.

In many cases this is beneficial, but in some cases it might be counter-productive; examples are administrators with many shut-down machines or VM images that are only started for some quick action, which is delayed or even blocked by the unattended upgrades. To change this behaviour, we can change/override the configuration of both APT's timer units `apt-daily-upgrade.timer` and `apt-daily.timer`. To do so, use `systemctl edit <timer_unit>` and override the *Persistent* attribute setting it to *false*:

```
[Timer]
Persistent=false
```

With this change, the timer will trigger the service only on the next scheduled time. In other words, it won't catch up to the run it missed while the system was off. See the explanation for the *Persistent* option in `systemd.timer(5)` manpage for more details.

### Where to pick updates from

In `/etc/apt/apt.conf.d/50unattended-upgrades`, the `Allowed-Origins` section specifies which repositories will be used to gather updates from. See the Ubuntu Packaging Guide for additional information about each official repository that Ubuntu uses.

This is the default:

```
Unattended-Upgrade::Allowed-Origins {
    "${distro_id}:${distro_codename}";
    "${distro_id}:${distro_codename}-security";
    // Extended Security Maintenance; doesn't necessarily exist for
    // every release and this system may not have it installed, but if
    // available, the policy for updates is such that unattended-upgrades
    // should also install from here by default.
    "${distro_id}ESMApps:${distro_codename}-apps-security";
    "${distro_id}ESM:${distro_codename}-infra-security";
//  "${distro_id}:${distro_codename}-updates";
//  "${distro_id}:${distro_codename}-proposed";
//  "${distro_id}:${distro_codename}-backports";
};
```

**Note**

The double "//" indicates a comment, so whatever follows "//" will not be evaluated.

If you want to also allow non-security updates to be applied automatically, then uncomment the line about `-updates`, like so:

```
Unattended-Upgrade::Allowed-Origins {
    "${distro_id}:${distro_codename}";
    "${distro_id}:${distro_codename}-security";
    // Extended Security Maintenance; doesn't necessarily exist for
    // every release and this system may not have it installed, but if
    // available, the policy for updates is such that unattended-upgrades
    // should also install from here by default.
    "${distro_id}ESMApps:${distro_codename}-apps-security";
    "${distro_id}ESM:${distro_codename}-infra-security";
    "${distro_id}:${distro_codename}-updates";
//  "${distro_id}:${distro_codename}-proposed";
//  "${distro_id}:${distro_codename}-backports";
};
```

The `Origin` field is a standard field used in package repositories. By default,

`unattended-upgrades` will ship with only official Ubuntu repositories configured, which is the configuration shown above. To have the system apply upgrades automatically from other repositories, its *Origin* needs to be added to this configuration option.

### Automatic upgrades from a PPA

A very popular package repository type is a Launchpad PPA. PPAs are normally referred to using the format `ppa:\<user\>/\<name\>`. For example, the PPA at https://launchpad.net/~canonical-server/+archive/ubuntu/server-backports is also referred to as `ppa:canonical-server/server-backports`.

To use a PPA in the *Allowed-Origins* configuration, we need its *Origin* field. For PPAs, it is in the format *LP-PPA-<user>-<name>*. Adding it to the `Allowed-Origins` configuration would result in the following (continuing from the example above):

```
Unattended-Upgrade::Allowed-Origins {
    "${distro_id}:${distro_codename}";
    "${distro_id}:${distro_codename}-security";
    // Extended Security Maintenance; doesn't necessarily exist for
    // every release and this system may not have it installed, but if
    // available, the policy for updates is such that unattended-upgrades
    // should also install from here by default.
    "${distro_id}ESMApps:${distro_codename}-apps-security";
    "${distro_id}ESM:${distro_codename}-infra-security";
    "${distro_id}:${distro_codename}-updates";
//  "${distro_id}:${distro_codename}-proposed";
//  "${distro_id}:${distro_codename}-backports";
    "LP-PPA-canonical-server-server-backports:${distro_codename}";
};
```

Due to the hyphens acting as both separators and part of the name, the complete configuration can become visually confusing, making it difficult to immediately distinguish between the username and PPA name. But that's ok, because it's the whole text that matters.

Now when the tool runs, that PPA will be considered for upgrades and is listed in *Allowed origins*:

```
2025-03-13 22:44:29,802 INFO Starting unattended upgrades script
2025-03-13 22:44:29,803 INFO Allowed origins are: o=Ubuntu,a=noble, o=Ubuntu,
a=noble-security, o=UbuntuESMApps,a=noble-apps-security, o=UbuntuESM,a=noble-
infra-security, o=LP-PPA-canonical-server-server-backports,a=noble
2025-03-13 22:44:29,803 INFO Initial blacklist:
2025-03-13 22:44:29,803 INFO Initial whitelist (not strict):
2025-03-13 22:44:33,029 INFO Option --dry-run given, *not* performing real actions
2025-03-13 22:44:33,029 INFO Packages that will be upgraded: ibverbs-providers
libibverbs1 rdma-core
2025-03-13 22:44:33,029 INFO Writing dpkg log to /var/log/unattended-upgrades/
unattended-upgrades-dpkg.log
2025-03-13 22:44:34,421 INFO All upgrades installed
2025-03-13 22:44:34,855 INFO The list of kept packages can't be calculated in dry-
run mode.
```

The correct *Origin* value to use is available in the repository's `InRelease` (or, for older formats,

the `Release` file), which can be found at the URL of the repository, or locally on the system after an `apt update` command was run. Locally these files are in the `/var/lib/apt/lists/` directory. For example, for the PPA case, we have:

```
/var/lib/apt/lists/ppa.launchpadcontent.net_canonical-server_server-backports_
ubuntu_dists_noble_InRelease
```

Which has contents:

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA512

Origin: LP-PPA-canonical-server-server-backports
Label: Server Team Backports
Suite: noble
Version: 24.04
Codename: noble
Date: Tue, 03 Dec 2024  6:00:43 UTC
Architectures: amd64 arm64 armhf i386 ppc64el riscv64 s390x
Components: main
Description: Ubuntu Noble 24.04
(...)
```

And there we can see the *Origin*.

### How to block certain packages

Specific packages can also be excluded from an update. This is controlled via the `Unattended-Upgrade::Package-Blacklist` configuration option in `/etc/apt/apt.conf.d/50unattended-upgrades`, which contains a list of Python Regular Expressions. Each line of this list is checked against the available package updates, and if there is a match, that package is not upgraded.

> **Note**
>
> Keep in mind that blocking a package might prevent other updates from being installed if they depend on the blocked package!

For example, this will block all packages that start with `linux-` from being automatically upgraded:

```
Unattended-Upgrade::Package-Blacklist {
    "linux-";
}
```

A more specific configuration like the one below will block only the `libc6` and `libc-bin` packages from being automatically upgraded:

```
Unattended-Upgrade::Package-Blacklist {
    "libc6$";
```

(continues on next page)

(continued from previous page)

```
    "libc-bin$";
}
```

Here, the use of the $ character marks the end of the package name (in regular expression terms, it's the end of the line, i.e., the end of the match).

**Note**

The regular expressions used here behave as if the "^" character is present at the start, i.e., the `libc6$` expression will match `libc6`, but will NOT match `glibc6` for example.

Of course, this being a regular expression means we could also write the above like this:

```
Unattended-Upgrade::Package-Blacklist {
    "libc(6|-bin)$";
}
```

Just be careful to not overuse the power of regular expressions: readability is key.

### Notifications

Besides logging, `unattended-upgrades` can also send out reports via email. There are two options that control this behavior in `/etc/apt/apt.conf.d/50unattended-upgrades`:

- `Unattended-Upgrade::Mail "user@example.com";`: If set to an email address, this option will trigger an email to this address containing an activity report. When this value is empty, or not set, (which is the default), no report is sent.
- `Unattended-Upgrade::MailReport "on-change";`: This option controls when a report is sent:
  - `always`: Always send a report, regardless if upgrades were applied or not.
  - `only-on-error`: Only send a report if there was an error.
  - `on-change`: Only send a report if upgrades were applied. This is the default value.

**Note**

Sending out emails like this requires the separate configuration of a package like `ssmtp(8)` or another minimalistic mail client that is capable of sending messages to a mail server.

### Notification examples

Here are some email examples (lines wrapped for better legibility).

### No changes applied, no errors

This would only be sent if `Unattended-Upgrade::MailReport` is set to `always`:

```
Subject: unattended-upgrades result for <hostname>: SUCCESS

Unattended upgrade result: No packages found that can be upgraded
 unattended and no pending auto-removals

Unattended-upgrades log:
Starting unattended upgrades script
Allowed origins are: o=Ubuntu,a=noble, o=Ubuntu,a=noble-security,
 o=UbuntuESMApps,a=noble-apps-security,
 o=UbuntuESM,a=noble-infra-security, o=Ubuntu,a=noble,
 o=Ubuntu,a=noble-security, o=UbuntuESMApps,a=noble-apps-security,
 o=UbuntuESM,a=noble-infra-security
Initial blacklist:
Initial whitelist (not strict):
No packages found that can be upgraded unattended and no pending auto-removals
```

### Upgrades applied, no errors

This is the default email report, when `Unattended-Upgrade::MailReport` is set to `on-change` :

```
Subject: unattended-upgrades result for nuc1: SUCCESS

Unattended upgrade result: All upgrades installed

Packages that were upgraded:
 linux-firmware

Package installation log:
Log started: 2025-03-13  06:19:10
Preparing to unpack
 .../linux-firmware_20240318.git3b128b60-0ubuntu2.10_amd64.deb ...
Unpacking linux-firmware (20240318.git3b128b60-0ubuntu2.10) over
 (20240318.git3b128b60-0ubuntu2.9) ...
Setting up linux-firmware (20240318.git3b128b60-0ubuntu2.10) ...
Processing triggers for initramfs-tools (0.142ubuntu25.5) ...
update-initramfs: Generating /boot/initrd.img-6.8.0-55-generic

Running kernel seems to be up-to-date.

The processor microcode seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
Log ended: 2025-03-13  06:19:26
```

(continues on next page)

(continued from previous page)

```
Unattended-upgrades log:
Starting unattended upgrades script
Allowed origins are: o=Ubuntu,a=noble, o=Ubuntu,a=noble-security,
 o=UbuntuESMApps,a=noble-apps-security,
 o=UbuntuESM,a=noble-infra-security, o=Ubuntu,a=noble,
 o=Ubuntu,a=noble-security, o=UbuntuESMApps,a=noble-apps-security,
 o=UbuntuESM,a=noble-infra-security
Initial blacklist:
Initial whitelist (not strict):
Packages that will be upgraded: linux-firmware
Writing dpkg log to /var/log/unattended-upgrades/unattended-upgrades-dpkg.log
All upgrades installed
```

### Reboots

Sometimes a system needs to be rebooted to fully apply an update. Such updates can use a mechanism in Ubuntu to let the system know that a reboot is recommended. `unattended-upgrades` can benefit from this mechanism and optionally reboot the system automatically when needed.

Reboots can be very disruptive, especially if the system fails to come back. There are some configuration options where this behavior can be adjusted:

- `Unattended-Upgrade::Automatic-Reboot "false";`: If this option is set to `true`, the system will be rebooted **without confirmation** at the end of an upgrade run if a reboot was requested. The default value is `false`.
- `Unattended-Upgrade::Automatic-Reboot-WithUsers "true";`: Automatically reboot even if there are users currently logged in when `Unattended-Upgrade::Automatic-Reboot` (the option above) is set to `true`. The default value is `true`.
- `Unattended-Upgrade::Automatic-Reboot-Time "now";`: If automatic reboot is enabled and needed, reboot at the specific time instead of immediately. The time value is passed as-is to the `shutdown(8)` command. It can be the text "now" (which is the default), or in the format "hh:mm" (hours:minutes), or an offset in minutes specified like "+m". Note that if using "hh:mm", it will be in the local system's timezone.

> **Note**
>
> For more information about this time specification for the reboot, and other options like cancelling a scheduled reboot, see the `shutdown(8)` manpage.

Below are the logs of an `unattended-upgrades` run that started at 20:43. The tool installed the available upgrades and detected that a reboot was requested, which was scheduled using the configured `Automatic-Reboot-Time` (20:45 in this example):

```
2025-03-13 20:43:25,923 INFO Starting unattended upgrades script
2025-03-13 20:43:25,924 INFO Allowed origins are: o=Ubuntu,a=noble, o=Ubuntu,
a=noble-security, o=UbuntuESMApps,a=noble-apps-security, o=UbuntuESM,a=noble-
infra-security
2025-03-13 20:43:25,924 INFO Initial blacklist:
2025-03-13 20:43:25,924 INFO Initial whitelist (not strict):
2025-03-13 20:43:29,082 INFO Packages that will be upgraded: libc6 python3-jinja2
2025-03-13 20:43:29,082 INFO Writing dpkg log to /var/log/unattended-upgrades/
unattended-upgrades-dpkg.log
2025-03-13 20:43:39,532 INFO All upgrades installed
2025-03-13 20:43:40,201 WARNING Found /var/run/reboot-required, rebooting
2025-03-13 20:43:40,207 WARNING Shutdown msg: b"Reboot scheduled for Thu 2025-03-
13 20:45:00 UTC, use 'shutdown -c' to cancel."
```

### When to consider disabling automatic updates

While automatic security updates are enabled in Ubuntu by default, in some situations it might make sense to disable this feature, or carefully limit its reach.

Here are some considerations.

### Systems which just get recreated

Some systems are designed to be redeployed from a new base image rather than receive updates. This is common in cloud and container-based applications, where outdated instances are destroyed and replaced with newer ones. These systems are typically very lean, focused solely on running specific applications, and so may lack self-update tools.

Keep in mind that the security exposure is still there: it's only the update mechanism that is different, and comes in the form of a new deployment. The update still has to happen somewhere, it's just not at runtime. Until that new deployment is done, outdated software might still be running.

### Manual steps required

While Ubuntu updates rarely require manual steps to complete an upgrade (at most a reboot can be required), it could be plausible that other applications require some manual steps after or before an update is applied. If that is the case, and if such steps cannot be safely automated, then maybe `unattended-upgrades` should be disabled on such systems.

Do consider block-listing such packages instead, if they are known to trigger such manual steps. In that case, the system can still benefit from all the other upgrades that might become available.

### Too much of a risk

Even with all the care in the world, applying updates to a running system comes with risk. Ubuntu believes that risk to be less than the risk of NOT applying a security update, which is why `unattended-upgrades` will apply security updates by default. But for some specific systems, the risk vs benefit equation might favor staying put and not applying an update unless specifically requested.

Always keep in mind, however, that specific packages can be blocked from receiving updates. For example, if a particular system runs a critical application that could break if certain libraries on the system are updated, then perhaps an acceptable compromise is to block these library packages from receiving upgrades, instead of disabling the whole feature.

As a middle-ground solution, you can configure `unattended-upgrades` to postpone impending updates to a later time. Read how to configure this feature in the *Postponable updates* section.

### Fleet management

The `unattended-upgrades` feature is helpful, does its job, and even sends out reports. But it's not intended to be a replacement for fleet management software. If a large number of Ubuntu systems needs to be kept updated, other solutions are better suited for the job. Such large deployments usually come with much stricter and wider requirements, like:

- Compliance reports: How many systems are up-do-date, how many are still behind, for how long has a system been exposed to a known vulnerability, etc.
- Maintenance windows: Different systems might require different maintenance windows. Some can be updated anytime, others only on weekends, etc.
- Canary rollouts: The ability to rollout updates to an initial group of systems, and over time increase the number of systems that will receive the update.

An example of such a Fleet Management software for Ubuntu systems is Landscape.

### Postponable updates

By default, system updates are applied automatically in the background without any user interaction.

Starting with Ubuntu 25.04, a system administrator can allow users to postpone these automatic updates for a limited number of days by setting the `Unattended-Update::Postpone-For-Days` option.

When this option is set, `unattended-upgrade` will run according to the cadence set by the administrator and check for updates. If there are updates available it will notify active users and prompt them to choose if they want to upgrade immediately, or postpone them. For example, if `Unattended-Upgrade::Postpone-For-Days "3"` is set, then the user can postpone upgrades for up to three days. After that, the next time `unattended-updates` runs the user will not be prompted and the upgrades will be applied to the system.

To enable the feature, edit the `/etc/apt/apt.conf.d/50unattended-upgrades` file and set the number of days that a user is allowed to postpone the automatic updates for. To postpone for up to **3** days:

```
Unattended-Upgrade::Postpone-For-Days "3";
```

To disable the feature, set the number of days to **0**.

### Prompt duration

The `Unattended-Upgrade::Postpone-Wait-Time` configuration option controls the amount of time (in seconds) that a user has available to send a postpone request after being prompted. If no postpone request is received within the specified time, the updates will start being applied as normal.

### Who can postpone

The system administrator can restrict access to the postpone request by defining Polkit rules for the `com.ubuntu.UnattendedUpgrade.Pending.Postpone` action. By default, access is granted to users of an active session. See the polkit documentation for how to set up authorization rules.

### Notifications in different environments

The prompting functionality is implemented graphically on Ubuntu Desktop by the `update-notifier` program. The user is shown a notification with the option to postpone the updates. Then, while updates are being applied an icon is visible in the system tray area informing the user so they know when it is safe to resume critical activities that may be affected by the updates.

On other environments, such as Ubuntu Server, you can implement your own prompting client by listening for the `AboutToStart` signal on the system bus and send a call to the `Postpone()` method. Read the `/usr/share/dbus-1/interfaces/com.ubuntu.UnattendedUpgrade.Pending.xml` interface specification for more details.

### Testing and troubleshooting

It's possible to test some configuration changes to `unattended-upgrade` without having to wait for the next time it would run. The `unattended-upgrade` tool has a `manual page` that explains all its command-line options. Here are the most useful ones for testing and troubleshooting:

- `-v`: Show a more verbose output.
- `--dry-run`: Just simulate what would happen, without actually making any changes.

For example, let's say we want to check if the PPA origin was included correctly in the *Allowed-Origins* configuration, and if an update that we know is available would be considered.

After we add `"LP-PPA-canonical-server-server-backports:${distro_codename}";` to `Allowed-Origins` in `/etc/apt/apt.conf.d/50unattended-upgrades`, we can run the tool in *verbose* and *dry-run* modes to check what would happen:

```
sudo unattended-upgrade -v --dry-run
```

Which produces the following output, in this example scenario:

```
Starting unattended upgrades script
Allowed origins are: o=Ubuntu,a=noble, o=Ubuntu,a=noble-security, o=UbuntuESMApps,
a=noble-apps-security, o=UbuntuESM,a=noble-infra-security, o=LP-PPA-canonical-
server-server-backports,a=noble
Initial blacklist:
```

(continues on next page)

(continued from previous page)

```
Initial whitelist (not strict):
Option --dry-run given, *not* performing real actions
Packages that will be upgraded: rdma-core
Writing dpkg log to /var/log/unattended-upgrades/unattended-upgrades-dpkg.log
/usr/bin/unattended-upgrade:567: DeprecationWarning: This process (pid=1213) is
multi-threaded, use of fork() may lead to deadlocks in the child.
  pid = os.fork()
/usr/bin/dpkg --status-fd 10 --no-triggers --unpack --auto-deconfigure /var/cache/
apt/archives/rdma-core_52.0-2ubuntu1~backport24.04.202410192216~ubuntu24.04.1_
amd64.deb
/usr/bin/dpkg --status-fd 10 --configure --pending
All upgrades installed
The list of kept packages can't be calculated in dry-run mode.
```

Of note, we see:

- `Allowed origins` include `o=LP-PPA-canonical-server-server-backports,a=noble`, which is the PPA we included.
- The `rdma-core` package would be updated.

Let's check this `rdma-core` package with the command `apt-cache policy rdma-core`:

```
rdma-core:
  Installed: 50.0-2build2
  Candidate: 52.0-2ubuntu1~backport24.04.202410192216~ubuntu24.04.1
  Version table:
     52.0-2ubuntu1~backport24.04.202410192216~ubuntu24.04.1 500
        500 https://ppa.launchpadcontent.net/canonical-server/server-backports/
ubuntu noble/main amd64 Packages
 *** 50.0-2build2 500
        500 http://br.archive.ubuntu.com/ubuntu noble/main amd64 Packages
        100 /var/lib/dpkg/status
```

And indeed, there is an update available from that PPA, and the next time `unattended-upgrade` runs on its own, it will apply that update. In fact, if the `--dry-run` option is removed from the command-line we just ran, the update will be installed.

### How to upgrade your Ubuntu release

In this page we show how to upgrade an Ubuntu Server or Ubuntu cloud image to the next major release. This is different from your regular software updates.

We recommend running a Long Term Support (LTS) release as it provides 5 years of standard support and security updates, whereas interim releases are only supported for nine months.

After the initial standard support period ends for an LTS release, an extended maintenance period is available via an Ubuntu Pro subscription, which provides coverage for an additional five years and is available for free on up to five machines. Find out more about the release lifecycle and support period for your release.

### Understanding Upgrade Paths

You can only upgrade from one LTS release directly to the **next sequential LTS release**. For example, if you are on Ubuntu 16.04 LTS, you can upgrade to Ubuntu 18.04 LTS. However, you cannot skip releases (e.g. jump from 16.04 LTS to 20.04 LTS).If you need to reach a later LTS, you will have to upgrade in stages: first to Ubuntu 18.04 LTS, then to Ubuntu 20.04 LTS, and so on.

### Pre-upgrade checklist

Before starting a major release upgrade, it's important to prepare your system to ensure a smooth transition. This step is essential, so we need to review the following items:

- **Review Releases notes:** Always check the **release notes** for the new Ubuntu version we are moving to. This can be found on the Ubuntu Wiki Releases Page.
- **Fully update the current system:** The release upgrade process requires that the current system has all the latest updates installed. This is a standard **package upgrade**: **apt update** will refresh package index database, and **apt upgrade** will download and install the latest versions of installed packages.

  1. Run these commands to ensure everything is up to date:

     ```
     sudo apt update
     sudo apt upgrade
     ```

  2. Confirm both commands complete successfully and no further updates are available.
  3. After applying all updates, it may be necessary to **reboot your system.** The release upgrade process will let you know if that's needed, but you can also check manually before: if the file `/run/reboot-required` exists, then you will need to reboot.

- **Check that there is enough free space:** A release upgrade involves downloading hundreds of new packages, which can be several gigabytes. Make sure you have enough free disk space.
- **Dedicate time for the upgrade:** This is an interactive process. The release upgrade will sometimes stop and ask questions, so you should monitor the upgrade and be available to respond.
- **Understand Third-party repositories:** Third-party software repositories and Personal Package Archives (PPAs) are disabled during the release upgrade. While software installed from these sources will not be removed, it's the most common cause of upgrade issues. Be prepared to re-enable them or find updated versions after the upgrade.
- **Backup all your data:** Although upgrades are normally safe, there is always a chance that something could go wrong. It is extremely important that the data is safely copied to a backup location to allow restoration if any problems occur.

### Upgrade the system

We recommend upgrading the system using the `do-release-upgrade` command on Server edition and cloud images. This command can handle system configuration changes that are sometimes needed between releases.

To start the process, run this command:

```
sudo do-release-upgrade
```

> **Note**
>
> Upgrading to a development release of Ubuntu is available using the `-d` flag. However, using the development release (or the `-d` flag) is **not recommended** for production environments.

Upgrades from one LTS release to the next one are only available after the first point release. For example, Ubuntu 18.04 LTS will only upgrade to Ubuntu 20.04 LTS after the 20.04.1 point release. If users wish to update before the point release (e.g., on a subset of machines to evaluate the LTS upgrade) users can force the upgrade via the `-d` flag.

### Pre-upgrade summary

Before making any changes the command `do-release-upgrade` will first do some checks to verify the system is ready to upgrade, and provide a summary of the upgrade before proceeding. If you accept the changes, the process will begin to update the system's packages:

```
Do you want to start the upgrade?


5 installed packages are no longer supported by Canonical. You can
still get support from the community.

4 packages are going to be removed. 117 new packages are going to be
installed. 424 packages are going to be upgraded.

You have to download a total of 262 M. This download will take about
33 minutes with a 1Mbit DSL connection and about 10 hours with a 56k
modem.

Fetching and installing the upgrade can take several hours. Once the
download has finished, the process cannot be canceled.

Continue [yN]  Details [d]
```

### Configuration changes

During the upgrade process you may be presented with a message to make decisions about package updates. These prompts occur when there are existing configuration files (e.g. edited by the user) and the new package configuration file are different. Below is an example prompt:

```
Configuration file '/etc/ssh/ssh_config'
 ==> Modified (by you or by a script) since installation.
 ==> Package distributor has shipped an updated version.
   What would you like to do about it ?  Your options are:
```

(continues on next page)

(continued from previous page)

```
    Y or I  : install the package maintainer's version
    N or O  : keep your currently-installed version
      D     : show the differences between the versions
      Z     : start a shell to examine the situation
 The default action is to keep your current version.
*** ssh_config (Y/I/N/O/D/Z) [default=N] ?
```

You should look at the differences between the files and decide what to do. The default response is to keep the current version of the file. There are situations where accepting the new version, like with `/boot/grub/menu.lst`, is required for the system to boot correctly with the new kernel.

### Removing Obsolete Packages

After all packages are updated, you can choose to remove any obsolete packages that are no longer needed:

```
Remove obsolete packages?


30 packages are going to be removed.

Continue [yN]  Details [d]
```

### Reboot

Finally, when the upgrade is complete you are prompted to reboot the system. The system is not considered upgraded until this reboot occurs:

```
System upgrade is complete.

Restart required

To finish the upgrade, a restart is required.
If you select 'y' the system will be restarted.

Continue [yN]
```

### Further reading

- For a complete list of releases and current support status, see the List of releases page.

### How to use the Ubuntu snapshot service

The Ubuntu snapshot service allows you to access old packages in the Ubuntu archive based on past dates. Some of the use cases for this service include:

- Installing a superseded version of a package as it existed at a particular date and time, which is useful to troubleshoot bugs or regressions, or to allow reproducibility.

- Setting upgrades to known states, which can be useful to ensure homogeneity for a fleet of Ubuntu installations and to ensure predictability upon package upgrades (including unattended upgrades).
- Support a structured update workflow by validating snapshots in different environments or by rolling out package upgrades in stages for a large set of machines.

Here we show how to set up and use the snapshot service.

### Prerequisites

Snapshots are supported in Ubuntu 23.10 onwards and on updated installations of Ubuntu 20.04 LTS (starting from `apt` 2.0.10) and Ubuntu 22.04 LTS (starting from `apt` 2.4.11).

In Ubuntu 24.04 LTS and later, `apt` will automatically detect whether a repository supports snapshots. Therefore, there is no need for additional configuration before you can start using the Ubuntu snapshot service. For Ubuntu versions lower than 24.04, you need to set a `snapshot` option in the relevant entries of the `/etc/apt/sources.list` file. For instance:

If you have the following contents in the `/etc/apt/sources.list` file

```
deb http://archive.ubuntu.com/ubuntu jammy main universe restricted multiverse
deb http://archive.ubuntu.com/ubuntu jammy-updates main universe restricted
multiverse
deb http://archive.ubuntu.com/ubuntu jammy-backports main restricted universe
multiverse
deb http://security.ubuntu.com/ubuntu jammy-security main restricted universe
multiverse
```

you can enable the snapshot service for all components in all pockets by changing it to

```
deb [snapshot=yes] http://archive.ubuntu.com/ubuntu jammy main universe restricted
multiverse
deb [snapshot=yes] http://archive.ubuntu.com/ubuntu jammy-updates main universe
restricted multiverse
deb [snapshot=yes] http://archive.ubuntu.com/ubuntu jammy-backports main
restricted universe multiverse
deb [snapshot=yes] http://security.ubuntu.com/ubuntu jammy-security main
restricted universe multiverse
```

Without the setup shown above, `apt` will simply ignore snapshot related command line options in Ubuntu releases lower than 24.04.

The following table describes whether the snapshot service can be used with each supported Ubuntu release, and whether or not it is necessary to configure the `snapshot` option in the `sources.list` file.

| Ubuntu Release | Supports Snapshot Service? | Need to configure `sources.list`? |
|---|---|---|
| Next Ubuntu releases | Yes | No |
| Ubuntu 25.10 | Yes | No |
| Ubuntu 25.04 | Yes | No |
| Ubuntu 24.04 LTS | Yes | No |
| Ubuntu 22.04 LTS | Yes (with `apt >= 2.4.11`) | Yes |
| Ubuntu 20.04 LTS | Yes (with `apt >= 2.0.10`) | Yes |

### Quick start

You can start using the snapshot service right away by passing the `--snapshot` option to `apt` followed by a timestamp in the format shown below.

```
$ sudo apt install hello --update --snapshot 20240301T030400Z
```

The `--snapshot` option and the the timestamp format are discussed with more details in the next sections.

### Using the snapshot service

There are several options when it comes to setting up Ubuntu snapshot services. Here we present the most relevant, supported ones.

Below, we refer to the snapshot ID as `$SNAPSHOT_ID`. This is a UTC date and time formatted as YYYYMMDDTHHMMSSZ, e.g., 20240430T214500Z, which refers to the snapshot that represents the state of the Ubuntu archive on April 30th, 2024, at 21:45:00 UTC.

Currently, the Ubuntu snapshot service provides snapshots for any date and time **after 1 March 2023**.

### The `--snapshot` CLI option

The snapshot service can be used through the `--snapshot` (or `-S`) CLI option for `apt`, which receives a snapshot ID argument.

```
$ sudo apt update --snapshot $SNAPSHOT_ID
$ sudo apt install $PACKAGE_NAME --snapshot $SNAPSHOT_ID
```

You can also use the short, atomic version to merge the two actions above in a single command, which will download the package information from the repository, then download and install the package:

```
$ sudo apt install $PACKAGE_NAME --update --snapshot $SNAPSHOT_ID
```

or using the short version of the snapshot option:

```
$ sudo apt install $PACKAGE_NAME --update -S $SNAPSHOT_ID
```

For instance, if you want to install Docker as it was released when Ubuntu 24.04 LTS first came out, you could try a snapshot from May 2025:

```
$ sudo apt update --snapshot 20240501T120000Z
```

Note that you will need to keep using the snapshot option to let `apt` operate over the snapshot repositories. For example, if you want to check what versions of Docker are available with the snapshot:

```
$ apt policy docker.io --snapshot 20240501T120000Z
docker.io:
  Installed: (none)
  Candidate: 24.0.7-0ubuntu4
  Version table:
     24.0.7-0ubuntu4 500
        500 https://snapshot.ubuntu.com/ubuntu/20240501T120000Z noble/universe
amd64 Packages
```

If you forget to use the snapshot option, `apt` will operate over the regular archive:

```
$ apt policy docker.io
docker.io:
  Installed: (none)
  Candidate: 28.2.2-0ubuntu1~24.04.1
  Version table:
     28.2.2-0ubuntu1~24.04.1 500
        500 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages
     27.5.1-0ubuntu3~24.04.2 500
        500 http://security.ubuntu.com/ubuntu noble-security/universe amd64
Packages
     24.0.7-0ubuntu4 500
        500 http://archive.ubuntu.com/ubuntu noble/universe amd64 Packages
```

Then, you can install Docker from that snapshot:

```
$ sudo apt install docker.io --snapshot 20240501T120000Z
...
$ docker --version
Docker version 24.0.7, build 24.0.7-0ubuntu4
```

Now, let's say you checked the Docker package publishing history in Launchpad and (for some reason) decided you want to install version `26.1.3-0ubuntu1~24.04.1`, which, as inferred from the publishing history, superseded `docker.io 24.0.7-0ubuntu4.1` in `noble-updates` on `2024-11-25`. Then, `24.0.7-0ubuntu4.1` was removed from the `-updates` pocket on `2024-11-26`.

You could then use `20241126T230000Z` as the snapshot ID to get the target package (`26.1.3-0ubuntu1~24.04.1`):

```
$ sudo apt install docker.io --update --snapshot 20241126T230000Z
...
$ docker --version
Docker version 26.1.3, build 26.1.3-0ubuntu1~24.04.1
```

> **Note**
>
> Note that, at the time of writing, the version of docker.io in the `noble-security` pocket is `27.5.1-0ubuntu3~24.04.2`, i.e., the version used in the example above may be affected by known vulnerabilities. When using the snapshot service, do make sure to check the latest version available in your Ubuntu release's `-security` pocket to make sure you are not installing a vulnerable package.

### Configuring the snapshot service for specific repositories

In your `apt` repositories configuration files, it is possible to specify snapshots to be used for each specific repository.

For Ubuntu 24.04 LTS and later, which uses the deb822 style by default, we add a `Snapshot` option to the relevant sources file in the following format:

```
Snapshot: $SNAPSHOT_ID
```

For Ubuntu series lower than 24.04, you change the value of the `snapshot` option in your sources file, i.e., if you had the following entry in your `/etc/apt/sources.list` file

```
deb [snapshot=yes] http://archive.ubuntu.com/ubuntu ...
```

you replace the `yes` value with the snapshot ID:

```
deb [snapshot=$SNAPSHOT_ID] http://archive.ubuntu.com/ubuntu ...
```

Let's say you want to pin a specific snapshot for all the pockets in your Ubuntu 24.04 LTS server. Then, you add the `Snapshot` options to `/etc/apt/sources.list.d/ubuntu.sources`:

```
Types: deb
URIs: http://archive.ubuntu.com/ubuntu
Suites: noble noble-updates noble-backports
Components: main universe restricted multiverse
Signed-By: /usr/share/keyrings/ubuntu-archive-keyring.gpg
Snapshot: 20250530T223000Z

Types: deb
URIs: http://security.ubuntu.com/ubuntu
Suites: noble-security
Components: main universe restricted multiverse
Signed-By: /usr/share/keyrings/ubuntu-archive-keyring.gpg
Snapshot: 20250530T223000Z
```

Afterwards, you can run the following commands to install the Docker version present in that snapshot:

```
$ sudo apt install --update docker.io
```

to install the Docker version present in that snapshot.

```
$ docker --version
Docker version 27.5.1, build 27.5.1-0ubuntu3~24.04.1
```

Note that, when you configure a repository to use a snapshot using the method described above, `apt` will always ignore the `--snapshot` option:

```
$ sudo apt install --update docker.io --snapshot 20251013T2300Z
...
Hit:5 https://snapshot.ubuntu.com/ubuntu/20250530T223000Z noble InRelease
Hit:6 https://snapshot.ubuntu.com/ubuntu/20250530T223000Z noble-updates InRelease
Hit:7 https://snapshot.ubuntu.com/ubuntu/20250530T223000Z noble-backports
InRelease
Hit:8 https://snapshot.ubuntu.com/ubuntu/20250530T223000Z noble-security InRelease
...
docker.io is already the newest version (27.5.1-0ubuntu3~24.04.1).
0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
```

As you can see from the logs above, `apt` is still using the snapshot configured in `/etc/apt/sources.list.d/ubuntu.sources`, ignoring the `--snapshot` CLI option.

### Configuring the snapshot service globally

As an alternative to configuring a specific snapshot for each individual repository, as described in the previous section, you can configure a snapshot globally for all repositories that have snapshots enabled.

> **Warning**
>
> If you are following these examples until here, make sure to revert the changes to the sources configuration file made in the previous section.

Let's configure `apt` to default to a specific snapshot by setting the `APT::Snapshot` option:

```
$ echo 'APT::Snapshot "20250801T111111Z";' | sudo tee /etc/apt/apt.conf.d/
50snapshot
```

The system will now default to fetching packages from the configured snapshot:

```
$ sudo apt install docker.io --update
...
$ docker --version
Docker version 27.5.1, build 27.5.1-0ubuntu3~24.04.2
```

Note that, in contrast to repositories with snapshots configured in the sources file as shown in the previous section, configuring a snapshot globally will not make `apt` ignore the `--snapshot` CLI option:

```
$ sudo apt install docker.io --update --snapshot 20251013T120000Z
$ docker --version
Docker version 28.2.2, build 28.2.2-0ubuntu1~24.04.1
```

You can revert the global configuration by removing the file which added it:

```
$ sudo rm /etc/apt/apt.conf.d/50snapshot
```

### Further reading

- The Ubuntu snapshot service official page and documentation
- Securing multiple Ubuntu instances while maximising uptime
- Integrating the Ubuntu Snapshot Service into systems management and update tools
- *Automatic updates* shows you how to configure (or turn off) automatic updates
- *Upgrade your release* shows you how to upgrade from one Ubuntu release to the next one
- *Snapshot service* shows you how to use the Ubuntu Snapshot Service to update packages to time-specific archive states.

### Troubleshooting

- *Reporting bugs* shows you how to report a bug using the Apport utility
- *Kernel crash dump* shows how to use the kernel crash dump utility

### How to report a bug in Ubuntu Server

The Ubuntu project, including Ubuntu Server, uses Launchpad as its bug tracker. To file a bug, you will first need to create a Launchpad account.

### Report bugs with `apport-cli`

The preferred way to report a bug is with the `apport-cli` command. This command collects information from the machine on which it is run and publishes it to the bug report on Launchpad.

Getting this information to Launchpad can be a challenge if the system is not running a desktop environment with a browser (a common scenario with servers) or if it does not have Internet access. The steps to take in these situations are described below.

> **Note**
>
> The commands `apport-cli` and `ubuntu-bug` should give the same results on a command-line interface (CLI) server. The latter is actually a symlink to `apport-bug`, which is intelligent enough to know whether a desktop environment is in use, and will choose `apport-cli` if not. Since server systems tend to be CLI-only, `apport-cli` was chosen from the outset in this guide.

Bug reports in Ubuntu need to be filed against a specific software package, so the name of the package (source package or program name/path) affected by the bug needs to be supplied to `apport-cli`:

```
apport-cli PACKAGENAME
```

Once `apport-cli` has finished gathering information you will be asked what to do with it. For instance, to report a bug against vim using `apport-cli vim` produces output like this:

```
*** Collecting problem information

The collected information can be sent to the developers to improve the
application. This might take a few minutes.
...

*** Send problem report to the developers?

After the problem report has been sent, please fill out the form in the
automatically opened web browser.

What would you like to do? Your options are:
  S: Send report (2.8 KB)
  V: View report
  K: Keep report file for sending later or copying to somewhere else
  I: Cancel and ignore future crashes of this program version
  C: Cancel
Please choose (S/V/K/I/C):
```

The first three options are described below.

### S: Send report

Submits the collected information to Launchpad as part of the process of filing a new bug report. You will be given the opportunity to describe the bug in your own words.

```
*** Uploading problem information

The collected information is being sent to the bug tracking system.
This might take a few minutes.
94%

*** To continue, you must visit the following URL:

  https://bugs.launchpad.net/ubuntu/+source/vim/+filebug/09b2495a-e2ab-11e3-879b-
68b5996a96c8?

You can launch a browser now, or copy this URL into a browser on another computer.


Choices:
  1: Launch a browser now
  C: Cancel
Please choose (1/C):  1
```

The browser that will be used when choosing '1' will be the one known on the system as

`www-browser` via the `Debian alternatives system`. Examples of text-based browsers to install include links, *ELinks*, lynx, and w3m. You can also manually point an existing browser at the given URL.

### V: View

This displays the collected information on the screen for review. This can be a lot of information! Press Enter to scroll through the screens. Press q to quit and return to the choice menu.

### K: Keep

This writes the collected information to disk. The resulting file can be later used to file the bug report, typically after transferring it to another Ubuntu system.

```
What would you like to do? Your options are:
  S: Send report (2.8 KB)
  V: View report
  K: Keep report file for sending later or copying to somewhere else
  I: Cancel and ignore future crashes of this program version
  C: Cancel
Please choose (S/V/K/I/C): k
Problem report file: /tmp/apport.vim.1pg92p02.apport
```

To report the bug, get the file onto an Internet-enabled Ubuntu system and apply `apport-cli` to it. This will cause the menu to appear immediately (since the information is already collected). You should then press s to send:

```
apport-cli apport.vim.1pg92p02.apport
```

To directly save a report to disk (without menus) you can run:

```
apport-cli vim --save apport.vim.test.apport
```

Report names should end in `.apport`.

**Note**

If this Internet-enabled system is non-Ubuntu/Debian, `apport-cli` is not available so the bug will need to be created manually. An `apport` report is also not to be included as an attachment to a bug either so it is completely useless in this scenario.

## Reporting application crashes

The software package that provides the `apport-cli` utility, `apport`, can be configured to automatically capture the state of a crashed application. This is enabled by default in `/etc/default/apport`.

After an application crashes, if enabled, `apport` will store a crash report under `/var/crash`:

```
-rw-r----- 1 peter    whoopsie 150K Jul 24 16:17 _usr_lib_x86_64-linux-gnu_
libmenu-cache2_libexec_menu-cached.1000.crash
```

Use the `apport-cli` command with no arguments to process any pending crash reports. It will offer to report them one by one, as in the following example:

```
apport-cli
```

```
*** Send problem report to the developers?

After the problem report has been sent, please fill out the form in the
automatically opened web browser.

What would you like to do? Your options are:
  S: Send report (153.0 KB)
  V: View report
  K: Keep report file for sending later or copying to somewhere else
  I: Cancel and ignore future crashes of this program version
  C: Cancel
Please choose (S/V/K/I/C): s
```

If you send the report, as was done above, the prompt will be returned immediately and the `/var/crash` directory will then contain 2 extra files:

```
-rw-r----- 1 peter    whoopsie 150K Jul 24 16:17 _usr_lib_x86_64-linux-gnu_
libmenu-cache2_libexec_menu-cached.1000.crash
-rw-rw-r-- 1 peter    whoopsie    0 Jul 24 16:37 _usr_lib_x86_64-linux-gnu_
libmenu-cache2_libexec_menu-cached.1000.upload
-rw------- 1 whoopsie whoopsie    0 Jul 24 16:37 _usr_lib_x86_64-linux-gnu_
libmenu-cache2_libexec_menu-cached.1000.uploaded
```

Sending in a crash report like this will not immediately result in the creation of a new public bug. The report will be made private on Launchpad, meaning that it will be visible to only a limited set of bug triagers. These triagers will then scan the report for possible private data before creating a public bug.

### Further reading

- See the Reporting Bugs Ubuntu wiki page.
- Also, the Apport page has some useful information. Though some of it pertains to using a *GUI*.

### Kernel crash dump

A 'kernel crash dump' refers to a portion of the contents of volatile memory (RAM) that is copied to disk whenever the execution of the kernel is disrupted. The following events can cause a kernel disruption:

- Kernel panic
- Non-maskable interrupts (NMI)
- Machine check exceptions (MCE)
- Hardware failure
- Manual intervention

For some of these events (kernel panic, NMI) the kernel will react automatically and trigger the crash dump mechanism through *kexec*. In other situations a manual intervention is required in order to capture the memory. Whenever one of the above events occurs, it is important to find out the root cause in order to prevent it from happening again. The cause can be determined by inspecting the copied memory contents.

### Kernel crash dump mechanism

When a kernel panic occurs, the kernel relies on the *kexec* mechanism to quickly reboot a new instance of the kernel in a pre-reserved section of memory that had been allocated when the system booted (see below). This permits the existing memory area to remain untouched in order to safely copy its contents to storage.

### KDump enabled by default

Starting in Oracular Oriole (24.10) the kernel crash dump facility will be enabled by default during standard Ubuntu Desktop or Ubuntu Server installations on systems that meet the following requirements:

- the system has at least 4 CPU threads
- the system has at least 6GB of RAM, and less than 2TB of RAM
- the free space available in `/var` is more than 5 times the amount of RAM and swap space
- and the CPU architecture is
  - amd64 or s390x, or
  - arm64 and UEFI is used

On machines with it enabled (either by default or by manual installation), it can be disabled via the command:

```
sudo apt remove kdump-tools
```

On machines that do not meet these requirements and on pre-24.10 releases, the kernel crash dump facility can be enabled manually by following the installation instructions that follow.

### Installation

The kernel crash dump utility is installed with the following command:

```
sudo apt install kdump-tools
```

> **Note**
>
> Starting with 16.04, the kernel crash dump mechanism is enabled by default.

During the installation, you will be prompted with the following dialog:

```
|-----------------------| Configuring kdump-tools |-----------------------|
|                                                                         |
|                                                                         |
| If you choose this option, the kdump-tools mechanism will be enabled.  A  |
| reboot is still required in order to enable the crashkernel kernel      |
| parameter.                                                              |
|                                                                         |
| Should kdump-tools be enabled be default?                               |
|                                                                         |
|                    <Yes>                        <No>                    |
|                                                                         |
|-------------------------------------------------------------------------|
```

'Yes' should be selected to enable `kdump-tools`. If you want to revisit your choice, you can use the `dpkg-reconfigure kdump-tools` command and answer 'Yes' or 'No'.

As well, you will also need to edit `/etc/default/kdump-tools` to enable `kdump` by including the following line:

```
USE_KDUMP=1
```

If you're disabling `kdump-tools`, either set USE_KDUMP=0 or remove the line from the file.

If a reboot has not been done since installation of the `linux-crashdump` package, a reboot will be required in order to activate the `crashkernel= boot` parameter. Upon reboot, `kdump-tools` will be enabled and active.

If you enable `kdump-tools` after a reboot, you will only need to issue the `kdump-config load` command to activate the `kdump` mechanism.

You can view the current status of `kdump` via the command `kdump-config show`. This will display something like this:

```
DUMP_MODE:        kdump
USE_KDUMP:        1
KDUMP_SYSCTL:     kernel.panic_on_oops=1
KDUMP_COREDIR:    /var/crash
crashkernel addr:
   /var/lib/kdump/vmlinuz
kdump initrd:
   /var/lib/kdump/initrd.img
current state:    ready to kdump
kexec command:
  /sbin/kexec -p --command-line="..." --initrd=...
```

This tells us that we will find core dumps in `/var/crash`.

### Configuration

In addition to local dump, it is now possible to use the remote dump functionality to send the kernel crash dump to a remote server, using either the SSH or NFS protocols.

### Local kernel crash dumps

Local dumps are configured automatically and will remain in use unless a remote protocol is chosen. Many configuration options exist and are thoroughly documented in the `/etc/default/kdump-tools` file.

### Remote kernel crash dumps using the SSH protocol

To enable remote dumps using the SSH protocol, the `/etc/default/kdump-tools` must be modified in the following manner:

```
# ----------------------------------------------------------------------
# Remote dump facilities:
# SSH - username and hostname of the remote server that will receive the dump
#       and dmesg files.
# SSH_KEY - Full path of the ssh private key to be used to login to the remote
#           server. use kdump-config propagate to send the public key to the
#           remote server
# HOSTTAG - Select if hostname of IP address will be used as a prefix to the
#           timestamped directory when sending files to the remote server.
#           'ip' is the default.
SSH="ubuntu@kdump-netcrash"
```

The only mandatory variable to define is SSH. It must contain the username and *hostname* of the remote server using the format `{username}@{remote server}`.

`SSH_KEY` may be used to provide an existing private key to be used. Otherwise, the `kdump-config propagate` command will create a new keypair. The `HOSTTAG` variable may be used to use the hostname of the system as a prefix to the remote directory to be created instead of the IP address.

The following example shows how `kdump-config propagate` is used to create and propagate a new keypair to the remote server:

```
sudo kdump-config propagate
```

Which produces an output like this:

```
Need to generate a new ssh key...
The authenticity of host 'kdump-netcrash (192.168.1.74)' can't be established.
ECDSA key fingerprint is SHA256:iMp+5Y28qhbd+tevFCWrEXykDd4dI3yN4OVlu3CBBQ4.
Are you sure you want to continue connecting (yes/no)? yes
ubuntu@kdump-netcrash's password:
propagated ssh key /root/.ssh/kdump_id_rsa to server ubuntu@kdump-netcrash
```

The password of the account used on the remote server will be required in order to successfully send the public key to the server.

The `kdump-config show` command can be used to confirm that `kdump` is correctly configured to use the SSH protocol:

```
kdump-config show
```

Whose output appears like this:

```
DUMP_MODE:        kdump
USE_KDUMP:        1
KDUMP_SYSCTL:     kernel.panic_on_oops=1
KDUMP_COREDIR:    /var/crash
crashkernel addr: 0x2c000000
   /var/lib/kdump/vmlinuz: symbolic link to /boot/vmlinuz-4.4.0-10-generic
kdump initrd:
   /var/lib/kdump/initrd.img: symbolic link to /var/lib/kdump/initrd.img-4.4.0-10-
generic
SSH:              ubuntu@kdump-netcrash
SSH_KEY:          /root/.ssh/kdump_id_rsa
HOSTTAG:          ip
current state:    ready to kdump
```

### Remote kernel crash dumps using the NFS protocol

To enable remote dumps using the NFS protocol, the `/etc/default/kdump-tools` must be modified in the following manner:

```
# NFS -     Hostname and mount point of the NFS server configured to receive
#           the crash dump. The syntax must be {HOSTNAME}:{MOUNTPOINT}
#           (e.g. remote:/var/crash)
#
NFS="kdump-netcrash:/var/crash"
```

As with the SSH protocol, the `HOSTTAG` variable can be used to replace the IP address by the hostname as the prefix of the remote directory.

The `kdump-config show` command can be used to confirm that `kdump` is correctly configured to use the NFS protocol :

```
kdump-config show
```

Which produces an output like this:

```
DUMP_MODE:        kdump
USE_KDUMP:        1
KDUMP_SYSCTL:     kernel.panic_on_oops=1
KDUMP_COREDIR:    /var/crash
crashkernel addr: 0x2c000000
   /var/lib/kdump/vmlinuz: symbolic link to /boot/vmlinuz-4.4.0-10-generic
kdump initrd:
   /var/lib/kdump/initrd.img: symbolic link to /var/lib/kdump/initrd.img-4.4.0-10-
generic
NFS:              kdump-netcrash:/var/crash
HOSTTAG:          hostname
current state:    ready to kdump
```

### Verification

To confirm that the kernel dump mechanism is enabled, there are a few things to verify. First, confirm that the `crashkernel` boot parameter is present (note that the following line has been split into two to fit the format of this document):

```
cat /proc/cmdline

BOOT_IMAGE=/vmlinuz-3.2.0-17-server root=/dev/mapper/PreciseS-root ro
     crashkernel=384M-2G:64M,2G-:128M
```

The `crashkernel` parameter has the following syntax:

```
crashkernel=<range1>:<size1>[,<range2>:<size2>,...][@offset]
    range=start-[end] 'start' is inclusive and 'end' is exclusive.
```

So for the `crashkernel` parameter found in `/proc/cmdline` we would have :

```
crashkernel=384M-2G:64M,2G-:128M
```

The above value means:

- if the RAM is smaller than 384M, then don't reserve anything (this is the "rescue" case)
- if the RAM size is between 386M and 2G (exclusive), then reserve 64M
- if the RAM size is larger than 2G, then reserve 128M

Second, verify that the kernel has reserved the requested memory area for the `kdump` kernel by running:

```
dmesg | grep -i crash
```

Which produces the following output in this case:

```
...
[    0.000000] Reserving 64MB of memory at 800MB for crashkernel (System RAM:
1023MB)
```

Finally, as seen previously, the `kdump-config show` command displays the current status of the `kdump-tools` configuration :

```
kdump-config show
```

Which produces:

```
DUMP_MODE:        kdump
USE_KDUMP:        1
KDUMP_SYSCTL:     kernel.panic_on_oops=1
KDUMP_COREDIR:    /var/crash
crashkernel addr: 0x2c000000
   /var/lib/kdump/vmlinuz: symbolic link to /boot/vmlinuz-4.4.0-10-generic
kdump initrd:
      /var/lib/kdump/initrd.img: symbolic link to /var/lib/kdump/initrd.img-4.4.0-
10-generic
```

(continues on next page)

(continued from previous page)

```
current state:    ready to kdump

kexec command:
      /sbin/kexec -p --command-line="BOOT_IMAGE=/vmlinuz-4.4.0-10-generic root=/
dev/mapper/VividS--vg-root ro debug break=init console=ttyS0,115200 irqpoll
maxcpus=1 nousb systemd.unit=kdump-tools.service" --initrd=/var/lib/kdump/initrd.
img /var/lib/kdump/vmlinuz
```

### Testing the crash dump mechanism

> **Warning**
>
> Testing the crash dump mechanism **will cause a system reboot**. In certain situations, this can cause data loss if the system is under heavy load. If you want to test the mechanism, make sure that the system is idle or under very light load.

Verify that the *SysRQ* mechanism is enabled by looking at the value of the `/proc/sys/kernel/sysrq` kernel parameter:

```
cat /proc/sys/kernel/sysrq
```

If a value of *0* is returned, the dump and then reboot feature is disabled. A value greater than *1* indicates that a sub-set of `sysrq` features is enabled. See `/etc/sysctl.d/10-magic-sysrq.conf` for a detailed description of the options and their default values. Enable dump then reboot testing with the following command:

```
sudo sysctl -w kernel.sysrq=1
```

Once this is done, you must become root, as just using `sudo` will not be sufficient. As the *root* user, you will have to issue the command `echo c > /proc/sysrq-trigger`. If you are using a network connection, you will lose contact with the system. This is why it is better to do the test while being connected to the system console. This has the advantage of making the kernel dump process visible.

A typical test output should look like the following :

```
sudo -s
[sudo] password for ubuntu:
# echo c > /proc/sysrq-trigger
[   31.659002] SysRq : Trigger a crash
[   31.659749] BUG: unable to handle kernel NULL pointer dereference at
(null)
[   31.662668] IP: [<ffffffff8139f166>] sysrq_handle_crash+0x16/0x20
[   31.662668] PGD 3bfb9067 PUD 368a7067 PMD 0
[   31.662668] Oops: 0002 [#1] SMP
[   31.662668] CPU 1
....
```

The rest of the output is truncated, but you should see the system rebooting and somewhere in the log, you will see the following line :

```
Begin: Saving vmcore from kernel crash ...
```

Once completed, the system will reboot to its normal operational mode. You will then find the kernel crash dump file, and related subdirectories, in the `/var/crash` directory by running, e.g. `ls /var/crash` , which produces the following:

```
201809240744  kexec_cmd  linux-image-4.15.0-34-generic-201809240744.crash
```

If the dump does not work due to an 'out of memory' (OOM) error, then try increasing the amount of reserved memory by editing `/etc/default/grub.d/kdump-tools.cfg`. For example, to reserve 512 megabytes:

```
GRUB_CMDLINE_LINUX_DEFAULT="$GRUB_CMDLINE_LINUX_DEFAULT crashkernel=384M-:512M"
```

You can then run `sudo update-grub`, reboot afterwards, and then test again.

#### Resources

Kernel crash dump is a vast topic that requires good knowledge of the Linux kernel. You can find more information on the topic here:

- Kdump kernel documentation.
- Analyzing Linux Kernel Crash (Based on Fedora, it still gives a good walkthrough of kernel dump analysis)

#### See also

- Explanation: *Managing software*

*Managing software* provides guides on topics including:

- **Software updates** and configuration
- **Upgrading your Ubuntu release**
- **Bug reporting**

# 2.5. Data and storage

## 2.5.1. Data and storage

The following sections provide details on various topics related to storing, managing and accessing data.

#### Data management

- *OpenLDAP* shows how to set up and configure OpenLDAP
- *Databases* provides details on two of the most common databases found in Ubuntu: MySQL and PostgreSQL

## OpenLDAP

The Lightweight Directory Access Protocol, or LDAP, is a protocol for managing hierarchical data and accessing directories. The open source implementation used in Ubuntu is OpenLDAP.

These guides are intended to be sequential, so following them in the order presented below is suggested.

### Install and configure LDAP

Lightweight Directory Access Protocol (LDAP) is a protocol used for managing hierarchical data. OpenLDAP is the open-source implementation of LDAP used in Ubuntu. For information about OpenLDAP and explanations of the key terminology, see *Introduction to OpenLDAP*.

Installing slapd (the Stand-alone LDAP Daemon) creates a minimal working configuration with a top level entry, and an administrator's Distinguished Name (DN).

In particular, it creates a database instance that you can use to store your data. However, the **suffix** (or **base DN**) of this instance will be determined from the domain name of the host. If you want something different, you can change it right after the installation (before it contains any useful data).

> **Note**
>
> This guide will use a database suffix of `dc=example,dc=com`. You can change this to match your particular setup.

### Install slapd

You can install the server and the main command line utilities with the following command:

```
sudo apt install slapd ldap-utils
```

### Change the instance suffix (optional)

If you want to change your Directory Information Tree (*DIT*) suffix, now would be a good time since changing it discards your existing one. To change the suffix, run the following command:

```
sudo dpkg-reconfigure slapd
```

To switch your DIT suffix to `dc=example,dc=com`, for example, so you can follow this guide more closely, answer `example.com` when asked about the *DNS* domain name.

Throughout this guide we will issue many commands with the LDAP utilities. To save some typing, we can configure the OpenLDAP libraries with certain defaults in `/etc/ldap/ldap.conf` (adjust these entries for your server name and directory suffix):

```
BASE dc=example,dc=com
URI ldap://ldap01.example.com
```

### Default tree contents

slapd is designed to be configured within the service itself by dedicating a separate DIT for that purpose. This allows for dynamic configuration of slapd without needing to restart the service or edit config files. This configuration database consists of a collection of text-based LDIF files located under /etc/ldap/slapd.d, but these should never be edited directly. This way of working is known by several names: the "slapd-config" method, the "Real Time Configuration (RTC)" method, or the "cn=config" method. You can still use the traditional flat-file method (slapd.conf) but that will not be covered in this guide.

Right after installation, you will get two databases, or suffixes: one for your data, which is based on your host's domain (dc=example,dc=com), and one for your configuration, with its root at cn=config. To change the data on each we need different credentials and access methods:

- dc=example,dc=com The administrative user for this suffix is cn=admin,dc=example,dc=com and its password is the one selected during the installation of the slapd package.
- cn=config The configuration of slapd itself is stored under this suffix. Reading and writing to it can be done by the special *DN* gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth. This is how the local system's root user (uid=0/gid=0) is seen by the directory when using *SASL* EXTERNAL authentication through the ldapi:/// transport via the /run/slapd/ldapi Unix socket. Essentially what this means is that only the local root user can update the cn=config database. More details later.

### Default configuration tree

This is what the slapd-config DIT looks like via the LDAP protocol (listing only the DNs): To see what the slapd-config DIT looks like via the LDAP protocol, listing only the DNs, run this command:

```
sudo ldapsearch -Q -LLL -Y EXTERNAL -H ldapi:/// -b cn=config dn
```

The command-line options mean the following:

- -Q: Quiet mode for the SASL authentication setup.
- -LLL: Less verbose LDIF output. One "L" restricts output to LDIFv1; another disables comments; and the third one removes the LDIF version from the output.
- -Y EXTERNAL: Select the EXTERNAL SASL mechanism for authentication.
- -H ldapi:///: The URL to use to contact the server. In this case, it will use a local unix socket.
- -b cn=config: Start the search at the cn=config base.
- dn: Only retrieve the dn attribute.

The output will be the similar to the following:

```
dn: cn=config
dn: cn=module{0},cn=config
dn: cn=schema,cn=config
dn: cn={0}core,cn=schema,cn=config
dn: cn={1}cosine,cn=schema,cn=config
```

(continues on next page)

(continued from previous page)

```
dn: cn={2}nis,cn=schema,cn=config
dn: cn={3}inetorgperson,cn=schema,cn=config
dn: olcDatabase={-1}frontend,cn=config
dn: olcDatabase={0}config,cn=config
dn: olcDatabase={1}mdb,cn=config
```

Where the entries mean the following:

- `cn=config`: Global settings
- `cn=module{0},cn=config`: A dynamically loaded module
- `cn=schema,cn=config`: Contains hard-coded system-level schema
- `cn={0}core,cn=schema,cn=config`: The hard-coded *core* schema
- `cn={1}cosine,cn=schema,cn=config`: The Cosine schema
- `cn={2}nis,cn=schema,cn=config`: The Network Information Services (NIS) schema
- `cn={3}inetorgperson,cn=schema,cn=config`: The InetOrgPerson schema
- `olcDatabase={-1}frontend,cn=config`: *Frontend* database, default settings for other databases
- `olcDatabase={0}config,cn=config`: `slapd` configuration database (`cn=config`)
- `olcDatabase={1}mdb,cn=config`: Your database instance (`dc=example,dc=com`)

Since the configuration is located under the `cn=config` suffix, we can use LDAP commands to inspect or modify it.

To see all the configuration, run this command:

```
sudo ldapsearch -Q -LLL -Y EXTERNAL -H ldapi:/// -b cn=config
```

The output is too large to show here, but it will start like this:

```
dn: cn=config
objectClass: olcGlobal
cn: config
olcArgsFile: /var/run/slapd/slapd.args
olcLogLevel: none
...
```

### Default “data” tree

After installing the `slapd` package, a default data tree is configured, based on the detected domain name of the system. Assuming a domain of `example.com`, this command can be run to show what it looks like:

```
ldapsearch -x -LLL -H ldap:/// -b dc=example,dc=com dn
```

Here the only new command-line option is `-x`, and we have a new parameter for `-H`:

- `-x`: Use simple authentication instead of SASL, which is essentially a plain text authentication. Since no **Bind DN** was provided (via `-D`), this becomes an *anonymous* bind. Without `-x`, the default is to use a SASL bind.
- `-H ldap:///`: Use the LDAP protocol over the network (and not over a unix socket), and since no hostname was provided, it's assumed to be localhost. To access a server on another host, one would use `ldap://server.example.com/` as the URL, for example.

The output will be the top-level entry which represents the base of the DIT.

```
dn: dc=example,dc=com
```

### Who am I?

A very handy tool to verify the authentication is `ldapwhoami`, which can be used as follows:

```
ldapwhoami -x
```

The output will say who we connected as:

```
anonymous
```

Now let's perform an authenticated call, via simple authentication:

```
ldapwhoami -x -D cn=admin,dc=example,dc=com -W
```

This time we will be shown our authentication DN, after the password prompt:

```
Enter LDAP Password:
dn:cn=admin,dc=example,dc=com
```

When you use simple bind (`-x`) and specify a Bind DN with `-D` as your authentication DN, the server will look for a `userPassword` attribute in the entry, and use that to verify the credentials. In this particular case above, we used the database **Root DN** entry, i.e., the actual administrator, and that is a special case whose password is set in the configuration when the package is installed.

> **Note**
>
> A simple bind without some sort of transport security mechanism is **clear text**, meaning the credentials are transmitted in the clear. You should *add Transport Layer Security (TLS) support* to your OpenLDAP server as soon as possible.

Let's try some SASL EXTERNAL authentication commands:

```
ldapwhoami -Y EXTERNAL -H ldapi:/// -Q
```

The authentication DN is quite different from the simple bind one from before:

```
dn:gidNumber=1000+uidNumber=1000,cn=peercred,cn=external,cn=auth
```

Let's try as root:

```
sudo ldapwhoami -Y EXTERNAL -H ldapi:/// -Q
```

Notice how the `uidNumber` and `gidNumber` changed:

```
dn:gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
```

When using SASL EXTERNAL via the `ldapi:///` transport, the Bind DN becomes a combination of the *uid* and *gid* of the connecting user, followed by the suffix `cn=peercred,cn=external,cn=auth`. The server ACLs know about this, and grant the local root user complete write access to `cn=config` via the SASL mechanism.

> **Note**
>
> OpenLDAP ACLs are explained in *Set up access control*

### Populate the directory

Let's introduce some content to our directory. We will add the following:

- A node called **People**, to store users
  - An entry for a user called **john**
- A node called **Groups**, to store groups
  - An entry for a group called **miners**

Create the following LDIF file and call it `add_content.ldif`:

```
dn: ou=People,dc=example,dc=com
objectClass: organizationalUnit
ou: People

dn: ou=Groups,dc=example,dc=com
objectClass: organizationalUnit
ou: Groups

dn: cn=miners,ou=Groups,dc=example,dc=com
objectClass: posixGroup
cn: miners
gidNumber: 5000
memberUid: john

dn: uid=john,ou=People,dc=example,dc=com
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
uid: john
sn: Doe
givenName: John
cn: John Doe
displayName: John Doe
```

(continues on next page)

(continued from previous page)

```
uidNumber: 10000
gidNumber: 5000
userPassword: {CRYPT}x
gecos: John Doe
loginShell: /bin/bash
homeDirectory: /home/john
```

> **Note**
>
> It's important that `uid` and `gid` values in your directory do not collide with local Linux users' values as defined in `/etc/passwd` and `/etc/group`. You can use higher number ranges, such as starting at 5000 or even higher.

Add the content:

```
ldapadd -x -D cn=admin,dc=example,dc=com -W -f add_content.ldif
```

The command will ask for the admin password, and then show the entries as they are being added:

```
Enter LDAP Password: ********
adding new entry "ou=People,dc=example,dc=com"

adding new entry "ou=Groups,dc=example,dc=com"

adding new entry "cn=miners,ou=Groups,dc=example,dc=com"

adding new entry "uid=john,ou=People,dc=example,dc=com"
```

We can check that the information has been correctly added with the `ldapsearch` utility. For example, let's search for the "john" entry, and request the `cn` and `gidnumber` attributes:

```
ldapsearch -x -LLL -b dc=example,dc=com '(uid=john)' cn gidNumber
```

The output shows the DNs that matched the search criteria, and the requested attributes:

```
dn: uid=john,ou=People,dc=example,dc=com
cn: John Doe
gidNumber: 5000
```

Here we used an LDAP "filter": `(uid=john)`. LDAP filters are very flexible and can become complex. For example, to list the group names of which **john** is a member, we could use the following command:

```
ldapsearch -x -LLL -b dc=example,dc=com '(&
(objectClass=posixGroup)(memberUid=john))' cn gidNumber
```

And the result tells us that "john" is a member of the "miners" group:

```
dn: cn=miners,ou=Groups,dc=example,dc=com
cn: miners
gidNumber: 5000
```

That filter is a logical "AND" (signalled by the "&" character in the filter expression) between two attributes: `objectClass=posixGroup` AND `memberUid=john`. Filters are very important in LDAP and mastering their syntax is extremely helpful. They are used for simple queries like this, but can also select what content is to be replicated to a secondary server, or even in complex ACLs. The full specification is defined in RFC 4515.

Notice we set the `userPassword` field for the "john" entry to the cryptic value `{CRYPT}x`. This essentially is an invalid password, because no hashing will produce just `x`. It's a common pattern when adding a user entry without a default password. To change the password to something valid, you can now use `ldappasswd`:

```
ldappasswd -x -D cn=admin,dc=example,dc=com -W -S uid=john,ou=people,dc=example,
dc=com
```

We will be prompted for the new password twice, and at the end for the bind password corresponding to the bind DN specified via `-D`:

```
New password:
Re-enter new password:
Enter LDAP Password:
```

To verify the change, we can use `ldapwhoami` with simple bind authentication using john's DN as the bind DN:

```
ldapwhoami -x -D uid=john,ou=people,dc=example,dc=com -W
```

If the new password worked, the output will show that we authenticated as the `uid=john,ou=People,dc=example,dc=com` DN:

```
Enter LDAP Password:
dn:uid=john,ou=People,dc=example,dc=com
```

**Note**

Remember that simple binds are insecure and you should *add TLS support* to your server as soon as possible!

The `slapd-config` DIT can also be queried and modified. Here are some common operations.

### Which "admin" DN to use?

Throughout this guide so far, we have used two different authentication mechanisms to make changes to the directory. Which one is needed for what kind of change?

Each directory tree suffix has its own specific administrative DN. This is the DN that can make changes to the tree and is not subject to ACLs. It is stored in the `olcRootDN` attribute under the `cn=config` configuration tree, and the corresponding password is in the `olcRootPW` attribute.

Besides this specific administrator entry, ACLs can also grant such privileges to any other DN in the directory. All of this is setup by the `slapd` package when it is installed. This results in the following DNs that can be used to make changes to each directory suffix:

| Suffix | DN for making changes | Authentication mechanism |
|---|---|---|
| cn=config | cn=admin,cn=config | absent |
| cn=config | gidNumber=0+uidNumber=0,cn=peerc | SASL EXTERNAL as root via ldapi:// |
| dc=example,c | cn=admin,dc=example,dc=com | Simple bind with passwordset during install or reconfigure |

### Change the "admin" password

There is really only one administrative DN that has an associated password, and it's the one created at install (or reconfigure) time. To locate it in the `cn=config` suffix, run this command:

```
sudo ldapsearch -Q -LLL -Y EXTERNAL -H ldapi:/// -b cn=config
'(olcSuffix=dc=example,dc=com)' olcSuffix olcRootDN olcRootPW
```

The output will be the configuration entry for the `dc=example,dc=com` suffix, and show only the selected attributes in the response:

```
dn: olcDatabase={1}mdb,cn=config
olcSuffix: dc=example,dc=com
olcRootDN: cn=admin,dc=example,dc=com
olcRootPW: {SSHA}Y0UjBUUmf08TC25ePVc6waI/mfvPNktk
```

Since the `olcRootPW` password attribute we want to change is located under the `cn=config` suffix, we will also have to use the SASL EXTERNAL authentication to modify it, according to the table shown earlier.

**Note**

Even though this `olcRootDN` is the administrative DN for the `dc=example,dc=com` suffix, it is stored under the `cn=config` suffix!

To change the password associated with the `olcRootDN` administrative DN, we need to replace the value of the `olcRootPW` attribute. That value is not the literal password, but the hash of the password, using a specific hash algorithm.

To obtain the hash of a password, suitable to be used as the value of `olcRootPW`, run the `slappasswd` command and type the password you want, with a confirmation. The output will be the hash for that password, which we will need for the next step:

```
New password:
Re-enter new password:
{SSHA}VKrYMxlSKhONGRpC6rnASKNmXG2xHXFo
```

Now prepare a `changerootpw.ldif` file with this content, which includes the hashed password from the output above:

```
dn: olcDatabase={1}mdb,cn=config
changetype: modify
replace: olcRootPW
olcRootPW: {SSHA}VKrYMxlSKhONGRpC6rnASKNmXG2xHXFo
```

Finally, run the `ldapmodify` command on this file:

```
ldapmodify -Q -Y EXTERNAL -H ldapi:/// -f changerootpw.ldif
```

If successful, the output will show the entry that is being modified:

```
modifying entry "olcDatabase={1}mdb,cn=config"
```

### Add an index

Like in other database types, having an index for attributes commonly used in searches can speed up such searches dramatically, specially on large trees. The default installation of OpenLDAP already creates several common indexes, but depending on your data and queries, other indexes might be helpful.

Use `ldapmodify` to add an "Index" to your `{1}mdb,cn=config` database definition (for `dc=example,dc=com`). In this example, we will add an "equality" and a "substring" index to the `mail` attribute. Create a file called `add_index.ldif`, and add the following contents:

```
dn: olcDatabase={1}mdb,cn=config
add: olcDbIndex
olcDbIndex: mail eq,sub
```

Then issue the command:

```
sudo ldapmodify -Q -Y EXTERNAL -H ldapi:/// -f add_index.ldif
```

The output will show the modifications being done:

```
modifying entry "olcDatabase={1}mdb,cn=config"
```

You can confirm the change with a search:

```
ldapsearch -Q -LLL -Y EXTERNAL -H ldapi:/// -b cn=config '(olcDatabase={1}mdb)'
olcDbIndex
```

And the result will include all instances of the `olcDbIndex` attribute:

```
dn: olcDatabase={1}mdb,cn=config
olcDbIndex: objectClass eq
olcDbIndex: cn,uid eq
olcDbIndex: uidNumber,gidNumber eq
olcDbIndex: member,memberUid eq
olcDbIndex: mail eq,sub
```

**See also**

To learn more about OpenLDAP indexes, check the upstream documentation at https://www.openldap.org/doc/admin26/tuning.html#Indexes

### Add a schema

Schemas can only be added to `cn=config` if they are in LDIF format. If not, they will first have to be converted. You can find unconverted schemas in addition to converted ones in the `/etc/ldap/schema` directory.

**Note**

It is not trivial to remove a schema from the slapd-config database. Practice adding schemas on a test system.

In the following example we'll add one of the pre-installed policy schemas in `/etc/ldap/schema/`. The pre-installed schemas exists in both converted (`.ldif`) and native (`.schema`) formats, so we don't have to convert them and can use `ldapadd` directly:

```
sudo ldapadd -Q -Y EXTERNAL -H ldapi:/// -f /etc/ldap/schema/corba.ldif
```

The output will confirm the new schema being added:

```
adding new entry "cn=corba,cn=schema,cn=config"
```

If the schema you want to add does not exist in LDIF format, a nice conversion tool that can be used is provided in the `schema2ldif` package.

### Logging

Activity logging for `slapd` is very useful when implementing an OpenLDAP-based solution – and it must be manually enabled after software installation. Otherwise, only rudimentary messages will appear in the logs. Logging, like any other such configuration, is enabled via the `slapd-config` database.

OpenLDAP comes with multiple logging levels, with each level containing the lower one (additive). A good level to try is **stats**. The `slapd-config(5)` manual page has more to say on the different subsystems.

### Example logging with the stats level

Create the file `logging.ldif` with the following contents:

```
dn: cn=config
changetype: modify
replace: olcLogLevel
olcLogLevel: stats
```

Run `ldapmodify` to implement the change:

```
sudo ldapmodify -Q -Y EXTERNAL -H ldapi:/// -f logging.ldif
```

Depending on how active your OpenLDAP server is, this will produce a significant amount of logging. It is recommended to revert back to a less verbose level once the need for this detailed logging isn't there anymore.

While in this verbose mode your host's syslog engine (rsyslog) may have a hard time keeping up. If you see log message like this, it means some messages were dropped:

```
rsyslogd-2177: imuxsock lost 228 messages from pid 2547 due to rate-limiting
```

You may consider a change to rsyslog's configuration. In `/etc/rsyslog.conf`, add:

```
# Disable rate limiting
# (default is 200 messages in 5 seconds; below we make the 5 become 0)
$SystemLogRateLimitInterval 0
```

And then restart the rsyslog daemon:

```
sudo systemctl restart syslog.service
```

### Next steps

Now that you have successfully installed LDAP, you may want to *set up users and groups*, or find out more *about access control*.

### Set up LDAP access control

The management of what type of access (read, write, etc) users should be granted for resources is known as **access control**. The configuration directives involved are called **access control lists** or *ACL*s.

When we *installed the slapd package*, various ACLs were set up automatically. We will look at a few important consequences of those defaults and, in so doing, we'll get an idea of how ACLs work and how they're configured.

To get the effective ACL for an LDAP query we need to look at the ACL entries of both the database being queried, and those of the special *frontend* database instance. Note that the ACLs belonging to the frontend database are always appended to the database-specific ACLs, and the first match 'wins'.

### Getting the ACLs

The following commands will give, respectively, the ACLs of the `mdb` database (`dc=example,dc=com`) and those of the frontend database:

```
$ sudo ldapsearch -Q -LLL -Y EXTERNAL -H ldapi:/// -b \
cn=config '(olcDatabase={1}mdb)' olcAccess

dn: olcDatabase={1}mdb,cn=config
olcAccess: {0}to attrs=userPassword by self write by anonymous auth by * none
olcAccess: {1}to attrs=shadowLastChange by self write by * read
```

(continues on next page)

(continued from previous page)

```
olcAccess: {2}to * by * read

$ sudo ldapsearch -Q -LLL -Y EXTERNAL -H ldapi:/// -b \
cn=config '(olcDatabase={-1}frontend)' olcAccess

dn: olcDatabase={-1}frontend,cn=config
olcAccess: {0}to * by dn.exact=gidNumber=0+uidNumber=0,cn=peercred,cn=external
 ,cn=auth manage by * break
olcAccess: {1}to dn.exact="" by * read
olcAccess: {2}to dn.base="cn=Subschema" by * read
```

**Note**

The Root *DN* always has full rights to its database and does not need to be included in any ACL.

### Interpreting the results

The first two ACLs are crucial:

```
olcAccess: {0}to attrs=userPassword by self write by anonymous auth by * none
olcAccess: {1}to attrs=shadowLastChange by self write by * read
```

This can be represented differently for easier reading:

```
to attrs=userPassword
    by self write
    by anonymous auth
    by * none

to attrs=shadowLastChange
    by self write
    by * read
```

These ACLs enforce the following:

- Anonymous 'auth' access is provided to the **userPassword** attribute so that users can authenticate, or **bind**. Perhaps counter-intuitively, 'by anonymous auth' is needed even when anonymous access to the *DIT* is unwanted, otherwise this would be a chicken-and-egg problem: before authentication, all users are anonymous.
- The 'by self write' ACL grants write access to the **userPassword** attribute to users who authenticated as the DN where the attribute lives. In other words, users can update the **userPassword** attribute of their own entries.
- The **userPassword** attribute is otherwise inaccessible by all other users, with the exception of the Root DN, who always has access and doesn't need to be mentioned explicitly.
- In order for users to change their own password, using `passwd` or other utilities, the user's own **shadowLastChange** attribute needs to be writable. All other directory users get to read this attribute's contents.

This DIT can be searched anonymously because of `to * by * read` in this ACL, which grants read access to everything else, by anyone (including anonymous):

```
to *
    by * read
```

If this is unwanted then you need to change the ACL. To force authentication during a bind request you can alternatively (or in combination with the modified ACL) use the `olcRequire: authc` directive.

### SASL identity

There is no administrative account ("Root DN") created for the `slapd-config` database. There is, however, a SASL identity that is granted full access to it. It represents the localhost's superuser (`root`/`sudo`). Here it is:

```
dn.exact=gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
```

The following command will display the ACLs of the `slapd-config` database:

```
$ sudo ldapsearch -Q -LLL -Y EXTERNAL -H ldapi:/// -b \
cn=config '(olcDatabase={0}config)' olcAccess

dn: olcDatabase={0}config,cn=config
olcAccess: {0}to * by dn.exact=gidNumber=0+uidNumber=0,cn=peercred,
              cn=external,cn=auth manage by * break
```

Since this is a SASL identity we need to use a SASL **mechanism** when invoking the LDAP utility in question – the **EXTERNAL** mechanism (see the previous command for an example). Note that:

- You must use `sudo` to become the root identity in order for the ACL to match.
- The EXTERNAL mechanism works via **Interprocess Communication** (IPC, UNIX domain sockets). This means you must use the `ldapi` URI format.

A succinct way to get all the ACLs is like this:

```
$ sudo ldapsearch -Q -LLL -Y EXTERNAL -H ldapi:/// -b \
cn=config '(olcAccess=*)' olcAccess olcSuffix
```

### Next steps

See how to *set up LDAP users and groups*.

### Further reading

- See the manual page for `slapd.access(5)`
- The access control topic in the OpenLDAP administrator's guide.

### OpenLDAP replication

The LDAP service becomes increasingly important as more networked systems begin to depend on it. In such an environment, it is standard practice to build redundancy (*high availability*) into LDAP to prevent disruption should the LDAP server become unresponsive. This is done through **LDAP replication**.

Replication is achieved via the Sync replication engine, **syncrepl**. This allows changes to be synchronised using a *Consumer - Provider* model. A detailed description of this replication mechanism can be found in the OpenLDAP administrator's guide and in its defining RFC 4533.

There are two ways to use this replication:

- **Standard replication**: Changed entries are sent to the consumer in their entirety. For example, if the `userPassword` attribute of the `uid=john,ou=people,dc=example,dc=com` entry changed, then the whole entry is sent to the consumer.
- **Delta replication**: Only the actual change is sent, instead of the whole entry.

The delta replication sends less data over the network, but is more complex to set up. We will show both in this guide.

**Important**

You **must** have Transport Layer Security (TLS) enabled already before proceeding with this guide. Please consult the *LDAP with TLS guide* for details of how to set this up.

### Provider configuration - replication user

Both replication strategies will need a replication user, as well as updates to the *ACL*s and limits regarding this user. To create the replication user, save the following contents to a file called `replicator.ldif`:

```
dn: cn=replicator,dc=example,dc=com
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: replicator
description: Replication user
userPassword: {CRYPT}x
```

Then add it with `ldapadd`:

```
$ ldapadd -x -ZZ -H ldap://ldap01.example.com -D cn=admin,dc=example,dc=com -W -f
replicator.ldif
Enter LDAP Password:
adding new entry "cn=replicator,dc=example,dc=com"
```

Now set a password for it with `ldappasswd`:

```
$ ldappasswd -x -ZZ -H ldap://ldap01.example.com -D cn=admin,dc=example,dc=com -W
-S cn=replicator,dc=example,dc=com
New password:
Re-enter new password:
Enter LDAP Password:
```

> **Note**
>
> Please adjust the server URI in the `-H` parameter if needed to match your deployment.

The next step is to give this replication user the correct privileges, i.e.:

- Read access to the content that we want replicated
- No search limits on this content

For that we need to update the ACLs on the provider. Since ordering matters, first check what the existing ACLs look like on the `dc=example,dc=com` tree:

```
$ sudo ldapsearch -Q -Y EXTERNAL -H ldapi:/// -LLL -b cn=config
'(olcSuffix=dc=example,dc=com)' olcAccess
dn: olcDatabase={1}mdb,cn=config
olcAccess: {0}to attrs=userPassword by self write by anonymous auth by * none
olcAccess: {1}to attrs=shadowLastChange by self write by * read
olcAccess: {2}to * by * read
```

What we need is to insert a new rule before the first one, and also adjust the limits for the replicator user. Prepare the `replicator-acl-limits.ldif` file with this content:

```
dn: olcDatabase={1}mdb,cn=config
changetype: modify
add: olcAccess
olcAccess: {0}to *
  by dn.exact="cn=replicator,dc=example,dc=com" read
  by * break
-
add: olcLimits
olcLimits: dn.exact="cn=replicator,dc=example,dc=com"
  time.soft=unlimited time.hard=unlimited
  size.soft=unlimited size.hard=unlimited
```

And add it to the server:

```
$ sudo ldapmodify -Q -Y EXTERNAL -H ldapi:/// -f replicator-acl-limits.ldif
modifying entry "olcDatabase={1}mdb,cn=config"
```

### Provider configuration - standard replication

The remaining configuration for the provider using standard replication is to add the `syncprov` overlay on top of the `dc=example,dc=com` database.

Create a file called `provider_simple_sync.ldif` with this content:

```
# Add indexes to the frontend db.
dn: olcDatabase={1}mdb,cn=config
changetype: modify
add: olcDbIndex
olcDbIndex: entryCSN eq
```

(continues on next page)

(continued from previous page)

```
-
add: olcDbIndex
olcDbIndex: entryUUID eq

#Load the syncprov module.
dn: cn=module{0},cn=config
changetype: modify
add: olcModuleLoad
olcModuleLoad: syncprov

# syncrepl Provider for primary db
dn: olcOverlay=syncprov,olcDatabase={1}mdb,cn=config
changetype: add
objectClass: olcOverlayConfig
objectClass: olcSyncProvConfig
olcOverlay: syncprov
olcSpCheckpoint: 100 10
olcSpSessionLog: 100
```

**Warning**

The LDIF above has some parameters that you should review before deploying in production on your directory. In particular – `olcSpCheckpoint` and `olcSpSessionLog`. Please see the `slapo-syncprov(5)` manual page. In general, `olcSpSessionLog` should be equal to (or preferably larger than) the number of entries in your directory. Also see ITS #8125 for details on an existing bug.

Add the new content:

```
sudo ldapadd -Q -Y EXTERNAL -H ldapi:/// -f provider_simple_sync.ldif
```

The Provider is now configured.

### Consumer configuration - standard replication

Install the software by going through *the installation steps*. Make sure schemas and the database suffix are the same, and *enable TLS*.

Create an LDIF file with the following contents and name it `consumer_simple_sync.ldif`:

```
dn: cn=module{0},cn=config
changetype: modify
add: olcModuleLoad
olcModuleLoad: syncprov

dn: olcDatabase={1}mdb,cn=config
changetype: modify
add: olcDbIndex
olcDbIndex: entryUUID eq
```

(continues on next page)

(continued from previous page)

```
-
add: olcSyncrepl
olcSyncrepl: rid=0
  provider=ldap://ldap01.example.com
  bindmethod=simple
  binddn="cn=replicator,dc=example,dc=com" credentials=<secret>
  searchbase="dc=example,dc=com"
  schemachecking=on
  type=refreshAndPersist retry="60 +"
  starttls=critical tls_reqcert=demand
-
add: olcUpdateRef
olcUpdateRef: ldap://ldap01.example.com
```

Ensure the following attributes have the correct values:

- `provider`: Provider server's *hostname* – `ldap01.example.com` in this example – or IP address. It must match what is presented in the provider's SSL certificate.
- `binddn`: The bind *DN* for the replicator user.
- `credentials`: The password you selected for the replicator user.
- `searchbase`: The database suffix you're using, i.e., content that is to be replicated.
- `olcUpdateRef`: Provider server's hostname or IP address, given to clients if they try to write to this consumer.
- `rid`: Replica ID, a unique 3-digit ID that identifies the replica. Each consumer should have at least one `rid`.

**Note**

A successful encrypted connection via `START_TLS` is being enforced in this configuration, to avoid sending the credentials in the clear across the network. See *LDAP with TLS* for details on how to set up OpenLDAP with trusted SSL certificates.

Add the new configuration:

```
sudo ldapadd -Q -Y EXTERNAL -H ldapi:/// -f consumer_simple_sync.ldif
```

Now you're done! The `dc=example,dc=com` tree should now be synchronising.

### Provider configuration - delta replication

The remaining provider configuration for delta replication is:

- Create a new database called `accesslog`
- Add the `syncprov` overlay on top of the `accesslog` and `dc=example,dc=com` databases
- Add the `accesslog` overlay on top of the `dc=example,dc=com` database

### Add `syncprov` and `accesslog` overlays and DBs

Create an LDIF file with the following contents and name it `provider_sync.ldif`:

```
# Add indexes to the frontend db.
dn: olcDatabase={1}mdb,cn=config
changetype: modify
add: olcDbIndex
olcDbIndex: entryCSN eq
-
add: olcDbIndex
olcDbIndex: entryUUID eq

#Load the syncprov and accesslog modules.
dn: cn=module{0},cn=config
changetype: modify
add: olcModuleLoad
olcModuleLoad: syncprov
-
add: olcModuleLoad
olcModuleLoad: accesslog

# Accesslog database definitions
dn: olcDatabase={2}mdb,cn=config
objectClass: olcDatabaseConfig
objectClass: olcMdbConfig
olcDatabase: {2}mdb
olcDbDirectory: /var/lib/ldap/accesslog
olcSuffix: cn=accesslog
olcRootDN: cn=admin,dc=example,dc=com
olcDbIndex: default eq
olcDbIndex: entryCSN,objectClass,reqEnd,reqResult,reqStart
olcAccess: {0}to * by dn.exact="cn=replicator,dc=example,dc=com" read by * break
olcLimits: dn.exact="cn=replicator,dc=example,dc=com"
  time.soft=unlimited time.hard=unlimited
  size.soft=unlimited size.hard=unlimited

# Accesslog db syncprov.
dn: olcOverlay=syncprov,olcDatabase={2}mdb,cn=config
changetype: add
objectClass: olcOverlayConfig
objectClass: olcSyncProvConfig
olcOverlay: syncprov
olcSpNoPresent: TRUE
olcSpReloadHint: TRUE

# syncrepl Provider for primary db
dn: olcOverlay=syncprov,olcDatabase={1}mdb,cn=config
changetype: add
objectClass: olcOverlayConfig
objectClass: olcSyncProvConfig
```

(continues on next page)

(continued from previous page)

```
olcOverlay: syncprov
olcSpCheckpoint: 100 10
olcSpSessionLog: 100

# accesslog overlay definitions for primary db
dn: olcOverlay=accesslog,olcDatabase={1}mdb,cn=config
objectClass: olcOverlayConfig
objectClass: olcAccessLogConfig
olcOverlay: accesslog
olcAccessLogDB: cn=accesslog
olcAccessLogOps: writes
olcAccessLogSuccess: TRUE
# scan the accesslog DB every day, and purge entries older than 7 days
olcAccessLogPurge: 07+00:00 01+00:00
```

**Warning**

The LDIF above has some parameters that you should review before deploying in production on your directory. In particular – `olcSpCheckpoint`, `olcSpSessionLog`. Please see the `slapo-syncprov(5)` manpage. In general, `olcSpSessionLog` should be equal to (or preferably larger than) the number of entries in your directory. Also see ITS #8125 for details on an existing bug. For `olcAccessLogPurge`, please check the `slapo-accesslog(5)` manpage.

Create a directory:

```
sudo -u openldap mkdir /var/lib/ldap/accesslog
```

Add the new content:

```
sudo ldapadd -Q -Y EXTERNAL -H ldapi:/// -f provider_sync.ldif
```

The Provider is now configured.

### Consumer configuration

Install the software by going through *the installation steps*. Make sure schemas and the database suffix are the same, and *enable TLS*.

Create an LDIF file with the following contents and name it `consumer_sync.ldif`:

```
dn: cn=module{0},cn=config
changetype: modify
add: olcModuleLoad
olcModuleLoad: syncprov

dn: olcDatabase={1}mdb,cn=config
changetype: modify
add: olcDbIndex
olcDbIndex: entryUUID eq
```

(continues on next page)

(continued from previous page)

```
-
add: olcSyncrepl
olcSyncrepl: rid=0
  provider=ldap://ldap01.example.com
  bindmethod=simple
  binddn="cn=replicator,dc=example,dc=com" credentials=<secret>
  searchbase="dc=example,dc=com"
  logbase="cn=accesslog"
  logfilter="(&(objectClass=auditWriteObject)(reqResult=0))"
  schemachecking=on
  type=refreshAndPersist retry="60 +"
  syncdata=accesslog
  starttls=critical tls_reqcert=demand
-
add: olcUpdateRef
olcUpdateRef: ldap://ldap01.example.com
```

Ensure the following attributes have the correct values:

- `provider`: Provider server's hostname – `ldap01.example.com` in this example – or IP address. It must match what is presented in the provider's SSL certificate.
- `binddn`: The bind DN for the replicator user.
- `credentials`: The password you selected for the replicator user.
- `searchbase`: The database suffix you're using, i.e., content that is to be replicated.
- `olcUpdateRef`: Provider server's hostname or IP address, given to clients if they try to write to this consumer.
- **rid**: Replica ID, a unique 3-digit ID that identifies the replica. Each consumer should have at least one `rid`.

**Note**

Note that a successful encrypted connection via `START_TLS` is being enforced in this configuration, to avoid sending the credentials in the clear across the network. See *LDAP with TLS* for details on how to set up OpenLDAP with trusted SSL certificates.

Add the new configuration:

```
sudo ldapadd -Q -Y EXTERNAL -H ldapi:/// -f consumer_sync.ldif
```

You're done! The `dc=example,dc=com` tree should now be synchronising.

### Testing

Once replication starts, you can monitor it by running:

```
$ ldapsearch -z1 -LLL -x -s base -b dc=example,dc=com contextCSN
dn: dc=example,dc=com
contextCSN: 20200423222317.722667Z#000000#000#000000
```

On both the provider and the consumer. Once the `contextCSN` value for both match, both trees are in sync. Every time a change is done in the provider, this value will change and so should the one in the consumer(s).

If your connection is slow and/or your LDAP database large, it might take a while for the consumer's `contextCSN` match the provider's. But, you will know it is progressing since the consumer's `contextCSN` will be steadily increasing.

If the consumer's `contextCSN` is missing or does not match the provider, you should stop and figure out the issue before continuing. Try checking the `slapd` entries in `/var/log/syslog` in the provider to see if the consumer's authentication requests were successful, or that its requests to retrieve data return no errors. In particular, verify that you can connect to the provider from the consumer as the replicator BindDN using `START_TLS`:

```
ldapwhoami -x -ZZ -H ldap://ldap01.example.com -D cn=replicator,dc=example,dc=com
-W
```

For our example, you should now see the `john` user in the replicated tree:

```
$ ldapsearch -x -LLL -H ldap://ldap02.example.com -b dc=example,dc=com '(uid=john)
' uid
dn: uid=john,ou=People,dc=example,dc=com
uid: john
```

### References

- Replication types, OpenLDAP Administrator's Guide
- LDAP Sync Replication - OpenLDAP Administrator's Guide
- RFC 4533.

## How to set up LDAP users and groups

Once you *have a working LDAP server*, you will need to install libraries on the client that know how and when to contact it. On Ubuntu, this was traditionally done by installing the `libnss-ldap` package, but nowadays you should use the *System Security Services Daemon (SSSD)*. To find out how to use LDAP with SSSD, refer to *our SSSD and LDAP* guide.

### User and group management - `ldapscripts`

A common use case for an LDAP server is to store UNIX user and group information in the directory. There are many tools out there, and big deployments will usually develop their own. However, as a quick and easy way to get started with storing user and group information in OpenLDAP, you can use the `ldapscripts` package.

### Install ldapscripts

You can install `ldapscripts` by running the following command:

```
sudo apt install ldapscripts
```

Then edit the file `/etc/ldapscripts/ldapscripts.conf` to arrive at something similar to the following:

```
SERVER=ldap://ldap01.example.com
LDAPBINOPTS="-ZZ"
BINDDN='cn=admin,dc=example,dc=com'
BINDPWDFILE="/etc/ldapscripts/ldapscripts.passwd"
SUFFIX='dc=example,dc=com'
GSUFFIX='ou=Groups'
USUFFIX='ou=People'
MSUFFIX='ou=Computers'
```

> **Note**
>
> Adjust **SERVER** and related **SUFFIX** options to suit your directory structure. Here, we are forcing use of **START_TLS** (`-ZZ` parameter). Refer to *LDAP with TLS* to learn how to set up the server with TLS support.

Store the `cn=admin` password in the `/etc/ldapscripts/ldapscripts.passwd` file and make sure it's only readable by the *root* local user:

```
 echo -n 'password' | sudo tee /etc/ldapscripts/ldapscripts.passwd
sudo chmod 400 /etc/ldapscripts/ldapscripts.passwd
```

> **Note**: The password file must contain exactly and only the password characters, no end-of-line or anything else. The `echo` command above with the `-n` parameter achieves that by suppressing the *EOL* character `\n`. And in order to prevent the password from appearing in the shell history, the *echo* command line is prefixed by a space.

The scripts are now ready to help manage your directory.

### Manage users and groups with ldapscripts

Here are some brief examples you can use to manage users and groups using `ldapscripts`.

### Create a new user

```
sudo ldapaddgroup george
sudo ldapadduser george george
```

This will create a group and user with name "george" and set the user's primary group (gid) to "george" as well.

### Change a user's password

```
$ sudo ldapsetpasswd george

Changing password for user uid=george,ou=People,dc=example,dc=com
New Password:
Retype New Password:
Successfully set password for user uid=george,ou=People,dc=example,dc=com
```

### Delete a user

```
sudo ldapdeleteuser george
```

Note that this won't delete the user's primary group, but will remove the user from supplementary ones.

### Add a group

```
sudo ldapaddgroup qa
```

### Delete a group

```
sudo ldapdeletegroup qa
```

### Add a user to a group

```
sudo ldapaddusertogroup george qa
```

You should now see a `memberUid` attribute for the `qa` group with a value of `george`.

### Remove a user from a group

```
sudo ldapdeleteuserfromgroup george qa
```

The `memberUid` attribute should now be removed from the `qa` group.

### Manage user attributes with `ldapmodifyuser`

The `ldapmodifyuser` script allows you to add, remove, or replace a user's attributes. The script uses the same syntax as the `ldapmodify` utility. For example:

```
sudo ldapmodifyuser george
# About to modify the following entry :
dn: uid=george,ou=People,dc=example,dc=com
objectClass: account
objectClass: posixAccount
cn: george
uid: george
```

(continues on next page)

(continued from previous page)

```
uidNumber: 10001
gidNumber: 10001
homeDirectory: /home/george
loginShell: /bin/bash
gecos: george
description: User account
userPassword:: e1NTSEF9eXFsTFcyWlhwWkF1eGUybVdFWHZKRzJVMjFTSG9vcHk=

# Enter your modifications here, end with CTRL-D.
dn: uid=george,ou=People,dc=example,dc=com
replace: gecos
gecos: George Carlin
```

The user's `gecos` should now be "George Carlin".

### `ldapscripts` templates

A nice feature of `ldapscripts` is the template system. Templates allow you to customise the attributes of user, group, and machine objects. For example, to enable the `user` template, edit `/etc/ldapscripts/ldapscripts.conf` by changing:

```
UTEMPLATE="/etc/ldapscripts/ldapadduser.template"
```

There are sample templates in the `/usr/share/doc/ldapscripts/examples` directory. Copy or rename the `ldapadduser.template.sample` file to `/etc/ldapscripts/ldapadduser.template`:

```
sudo cp /usr/share/doc/ldapscripts/examples/ldapadduser.template.sample \
/etc/ldapscripts/ldapadduser.template
```

Edit the new template to add the desired attributes. The following will create new users with an `objectClass` of `inetOrgPerson`:

```
dn: uid=<user>,<usuffix>,<suffix>
objectClass: inetOrgPerson
objectClass: posixAccount
cn: <user>
sn: <ask>
uid: <user>
uidNumber: <uid>
gidNumber: <gid>
homeDirectory: <home>
loginShell: <shell>
gecos: <user>
description: User account
title: Employee
```

Notice the `<ask>` option used for the **sn** attribute. This will make `ldapadduser` prompt you for its value.

There are utilities in the package that were not covered here. This command will output a list of them:

```
dpkg -L ldapscripts | grep /usr/sbin
```

### Next steps

Now that you know how to set up and modify users and groups, it's a good idea to secure your LDAP communication by *setting up Transport Layer Security (TLS)*.

### LDAP and Transport Layer Security (TLS)

When authenticating to an OpenLDAP server it is best to do so using an encrypted session. This can be accomplished using Transport Layer Security (TLS).

Here, we will be our own Certificate Authority (CA) and then create and sign our LDAP server certificate as that CA. This guide will use the `certtool` utility to complete these tasks. For simplicity, this is being done on the OpenLDAP server itself, but your real internal CA should be elsewhere.

Install the `gnutls-bin` and `ssl-cert` packages:

```
sudo apt install gnutls-bin ssl-cert
```

Create a private key for the Certificate Authority:

```
sudo certtool --generate-privkey --bits 4096 --outfile /etc/ssl/private/mycakey.
pem
```

Create the template/file `/etc/ssl/ca.info` to define the CA:

```
cn = Example Company
ca
cert_signing_key
expiration_days = 3650
```

Create the self-signed CA certificate:

```
sudo certtool --generate-self-signed \
--load-privkey /etc/ssl/private/mycakey.pem \
--template /etc/ssl/ca.info \
--outfile /usr/local/share/ca-certificates/mycacert.crt
```

**Note**

Yes, the `--outfile` path is correct. We are writing the CA certificate to `/usr/local/share/ca-certificates`. This is where `update-ca-certificates` will pick up trusted local CAs from. To pick up CAs from `/usr/share/ca-certificates`, a call to `dpkg-reconfigure ca-certificates` is necessary.

Run `update-ca-certificates` to add the new CA certificate to the list of trusted CAs. Note the one added CA:

```
$ sudo update-ca-certificates
Updating certificates in /etc/ssl/certs...
1 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
```

This also creates a `/etc/ssl/certs/mycacert.pem` symlink pointing to the real file in `/usr/local/share/ca-certificates`.

Make a private key for the server:

```
sudo certtool --generate-privkey \
--bits 2048 \
--outfile /etc/ldap/ldap01_slapd_key.pem
```

**Note**

Replace `ldap01` in the filename with your server's *hostname*. Naming the certificate and key for the host and service that will be using them will help keep things clear.

Create the `/etc/ssl/ldap01.info` info file containing:

```
organization = Example Company
cn = ldap01.example.com
tls_www_server
encryption_key
signing_key
expiration_days = 365
```

The above certificate is good for 1 year, and it's valid only for the `ldap01.example.com` hostname. You can adjust this according to your needs.

Create the server's certificate:

```
sudo certtool --generate-certificate \
--load-privkey /etc/ldap/ldap01_slapd_key.pem \
--load-ca-certificate /etc/ssl/certs/mycacert.pem \
--load-ca-privkey /etc/ssl/private/mycakey.pem \
--template /etc/ssl/ldap01.info \
--outfile /etc/ldap/ldap01_slapd_cert.pem
```

Adjust permissions and ownership:

```
sudo chgrp openldap /etc/ldap/ldap01_slapd_key.pem
sudo chmod 0640 /etc/ldap/ldap01_slapd_key.pem
```

Your server is now ready to accept the new TLS configuration.

Create the file `certinfo.ldif` with the following contents (adjust paths and filenames accordingly):

```
dn: cn=config
add: olcTLSCACertificateFile
olcTLSCACertificateFile: /etc/ssl/certs/mycacert.pem
-
add: olcTLSCertificateFile
olcTLSCertificateFile: /etc/ldap/ldap01_slapd_cert.pem
-
add: olcTLSCertificateKeyFile
olcTLSCertificateKeyFile: /etc/ldap/ldap01_slapd_key.pem
```

Use the `ldapmodify` command to tell `slapd` about our TLS work via the `slapd-config` database:

```
sudo ldapmodify -Y EXTERNAL -H ldapi:/// -f certinfo.ldif
```

If you need access to **LDAPS** (LDAP over SSL), then you need to edit `/etc/default/slapd` and include `ldaps:///` in `SLAPD_SERVICES` like below:

```
SLAPD_SERVICES="ldap:/// ldapi:/// ldaps:///"
```

And restart `slapd` with:

```
sudo systemctl restart slapd
```

Note that *StartTLS* will be available without the change above, and does NOT need a `slapd` restart.

Test *StartTLS*:

```
$ ldapwhoami -x -ZZ -H ldap://ldap01.example.com
anonymous
```

Test LDAPS:

```
$ ldapwhoami -x -H ldaps://ldap01.example.com
anonymous
```

To generate a certificate pair for an OpenLDAP replica (consumer), create a holding directory (which will be used for the eventual transfer) and run the following:

```
mkdir ldap02-ssl
cd ldap02-ssl
certtool --generate-privkey \
--bits 2048 \
--outfile ldap02_slapd_key.pem
```

Create an info file, `ldap02.info`, for the Consumer server, adjusting its values according to your requirements:

```
organization = Example Company
cn = ldap02.example.com
tls_www_server
encryption_key
```

(continues on next page)

(continued from previous page)

```
signing_key
expiration_days = 365
```

Create the Consumer's certificate:

```
    sudo certtool --generate-certificate \
    --load-privkey ldap02_slapd_key.pem \
    --load-ca-certificate /etc/ssl/certs/mycacert.pem \
    --load-ca-privkey /etc/ssl/private/mycakey.pem \
    --template ldap02.info \
    --outfile ldap02_slapd_cert.pem
```

**Note**

We had to use `sudo` to get access to the CA's private key. This means the generated certificate file is owned by root. You should change that ownership back to your regular user before copying these files over to the Consumer.

Get a copy of the CA certificate:

```
cp /etc/ssl/certs/mycacert.pem .
```

We're done. Now transfer the `ldap02-ssl` directory to the Consumer. Here we use `scp` (adjust accordingly):

```
cd ..
scp -r ldap02-ssl user@consumer:
```

On the Consumer side, install the certificate files you just transferred:

```
sudo cp ldap02_slapd_cert.pem ldap02_slapd_key.pem /etc/ldap
sudo chgrp openldap /etc/ldap/ldap02_slapd_key.pem
sudo chmod 0640 /etc/ldap/ldap02_slapd_key.pem
sudo cp mycacert.pem /usr/local/share/ca-certificates/mycacert.crt
sudo update-ca-certificates
```

Create the file `certinfo.ldif` with the following contents (adjust accordingly regarding paths and filenames, if needed):

```
dn: cn=config
add: olcTLSCACertificateFile
olcTLSCACertificateFile: /etc/ssl/certs/mycacert.pem
-
add: olcTLSCertificateFile
olcTLSCertificateFile: /etc/ldap/ldap02_slapd_cert.pem
-
add: olcTLSCertificateKeyFile
olcTLSCertificateKeyFile: /etc/ldap/ldap02_slapd_key.pem
```

Configure the `slapd-config` database:

```
sudo ldapmodify -Y EXTERNAL -H ldapi:/// -f certinfo.ldif
```

Like before, if you want to enable LDAPS, edit `/etc/default/slapd` and add `ldaps:///` to `SLAPD_SERVICES`, and then restart `slapd`.

Test *StartTLS*:

```
$ ldapwhoami -x -ZZ -H ldap://ldap02.example.com
anonymous
```

Test LDAPS:

```
$ ldapwhoami -x -H ldaps://ldap02.example.com
anonymous
```

### Backup and restore OpenLDAP

Now we have LDAP running just the way we want, it is time to ensure we can save all of our work and restore it as needed.

What we need is a way to back up the directory database(s) – specifically the configuration backend (`cn=config`) and the *DIT* (`dc=example,dc=com`). If we are going to backup those databases into, say, `/export/backup`, we could use `slapcat` as shown in the following script, called `/usr/local/bin/ldapbackup`:

```
#!/bin/bash

set -e

BACKUP_PATH=/export/backup
SLAPCAT=/usr/sbin/slapcat

nice ${SLAPCAT} -b cn=config > ${BACKUP_PATH}/config.ldif
nice ${SLAPCAT} -b dc=example,dc=com > ${BACKUP_PATH}/example.com.ldif
chown root:root ${BACKUP_PATH}/*
chmod 600 ${BACKUP_PATH}/*.ldif
```

**Note**

These files are uncompressed text files containing everything in your directory including the tree layout, usernames, and every password. So, you might want to consider making `/export/backup` an encrypted partition and even having the script encrypt those files as it creates them. Ideally you should do both, but that depends on your security requirements.

Then, it is just a matter of having a cron script to run this program as often as you feel comfortable with. For many, once a day suffices. For others, more often is required. Here is an example of a cron script called `/etc/cron.d/ldapbackup` that is run every night at 22:45h:

```
MAILTO=backup-emails@domain.com
45 22 * * *  root    /usr/local/bin/ldapbackup
```

Now the files are created, they should be copied to a backup server.

Assuming we did a fresh reinstall of LDAP, the restore process could be something like this:

```
#!/bin/bash

set -e

BACKUP_PATH=/export/backup
SLAPADD=/usr/sbin/slapadd

if [ -n "$(ls -l /var/lib/ldap/* 2>/dev/null)" -o -n "$(ls -l /etc/ldap/slapd.d/*
2>/dev/null)" ]; then
    echo Run the following to remove the existing db:
    echo sudo systemctl stop slapd.service
    echo sudo rm -rf /etc/ldap/slapd.d/* /var/lib/ldap/*
    exit 1
fi
sudo systemctl stop slapd.service || :
sudo slapadd -F /etc/ldap/slapd.d -b cn=config -l /export/backup/config.ldif
sudo slapadd -F /etc/ldap/slapd.d -b dc=example,dc=com -l /export/backup/example.
com.ldif
sudo chown -R openldap:openldap /etc/ldap/slapd.d/
sudo chown -R openldap:openldap /var/lib/ldap/
sudo systemctl start slapd.service
```

This is a simplistic backup strategy, of course. It's being shown here as a reference for the basic tooling you can use for backups and restores.

### How to configure OpenLDAP with passthrough authentication

### Background

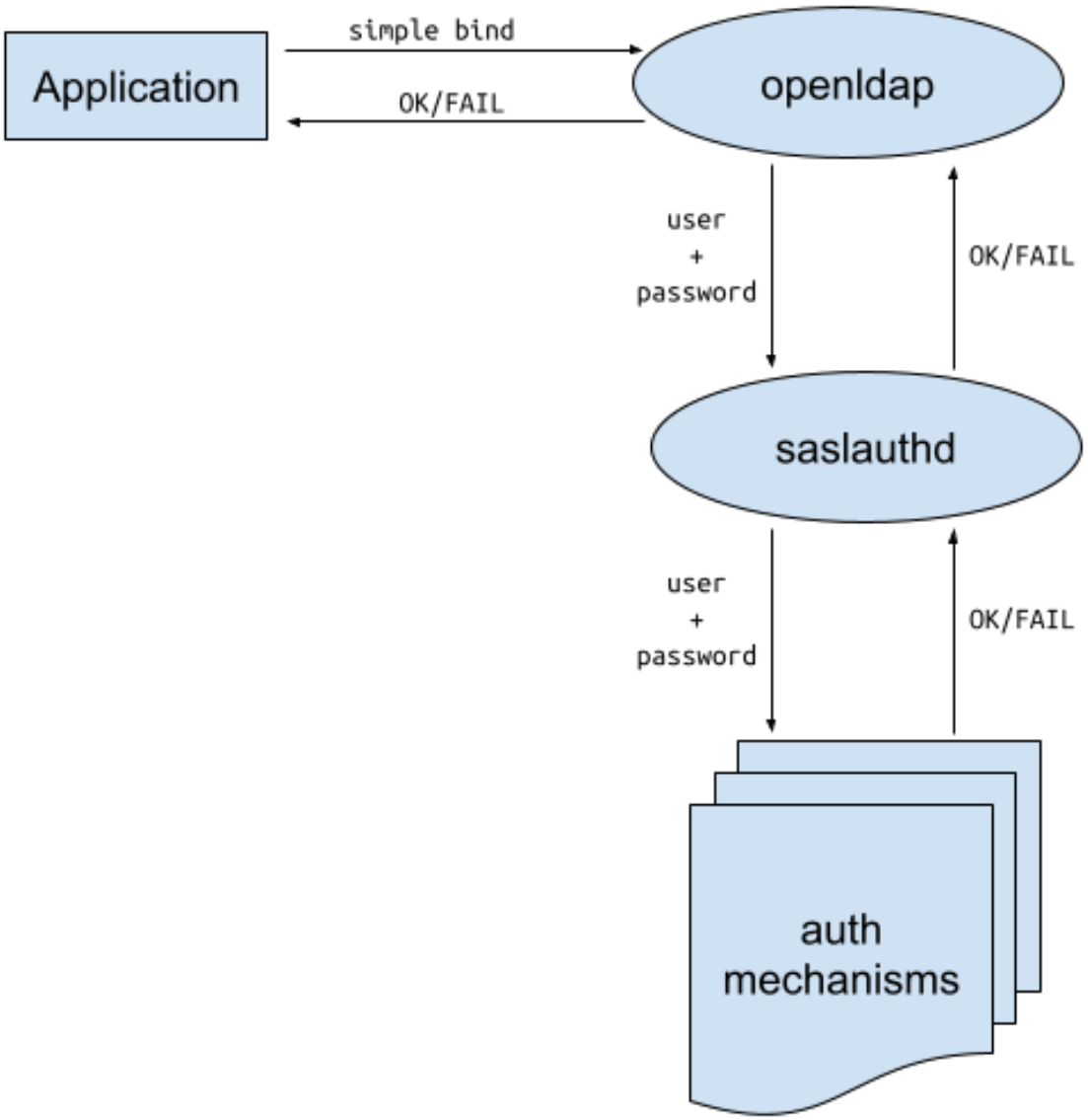
Managing large groups of users is great use case for OpenLDAP, and then allowing external applications to authenticate its users against this database allows the applications to offload user management to OpenLDAP. A sophisticated application can use a secure authentication method such as Generic Security Services Application Programming Interface (GSSAPI) to attempt to bind to the OpenLDAP server and obtain the authentication but a less sophisticated application may only be able to performs a simple bind request to OpenLDAP. This simple bind request supplies a username and password to OpenLDAP and then OpenLDAP checks the password against the hashed password it has stored internally and either accepts or rejects the bind. Obviously there is a security concern sending plain passwords over a network so this technique must only be used over a TLS connection see *setting up Transport Layer Security (TLS)*

### What is passthrough authentication

A complication occurs when the passwords storage mechanism is outside of OpenLDAP. How can this simple bind process be used without storing the password in the OpenLDAP database. One solution to this problem to use use the Simple Authentication and Security Layer (SASL) library to pass though this bind request to the mechanism where the password is actually stored. The application that does this is the saslauthd daemon. This daemon can

be configured to use various other authentication systems and will then pass back the authentication status to OpenLDAP which in turn passes it back to the calling application.

A graphic showing this process is as follows:




### Saslauthd authentication providers

Authentication providers that saslauthd can use are:

- getpwent – use the getpwent() library function
- kerberos5 – use Kerberos 5
- pam – use PAM
- rimap – use a remote IMAP server
- shadow – use the local shadow password file
- sasldb – use the local sasldb database file

- ldap – use LDAP (configuration is in /etc/saslauthd.conf)

Reference: https://www.openldap.org/doc/admin26/sasl.html

### How to configure OpenLDAP with passthrough SASL authentication using Kerberos

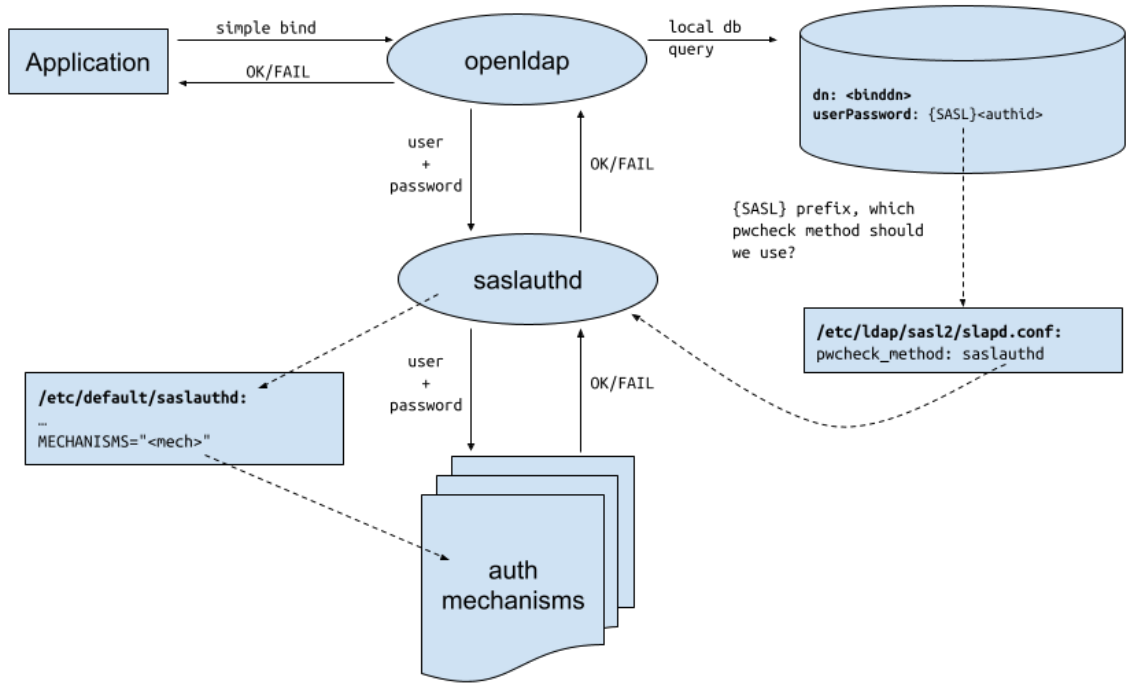
### Before you begin

It is assumed you are starting with a working OpenLDAP server, with a hostname of `ldap-server.example.com`. If not, follow this guide *Install and configure OpenLDAP* to set it up. It is also assumed that the `EXAMPLE.COM` realm is set up, and the Kerberos client tools (krb5-user) are installed on the ldap server. You will need to create an ubuntu principal. See *How to install a Kerberos server*. You should also know how to create service principals. See *How to configure Kerberos service principals*. All the following configuration will be on `ldap-server.example.com`.

> **Note**: This process is not the same as using Generic Security Services Application Programming Interface (GSSAPI) to log into the LDAP server. Rather it is using simple authentication with the OpenLDAP server so this should be over a Transport Layer Security (TLS) connection. The test user we will be using is `ubuntu@EXAMPLE.COM` which must exist in the Kerberos database

### How the passthrough authentication will work

Here is a diagram showing how all the different pieces work together:




We will go over all these details next.

### Package installation

Install `saslauthd` on the OpenLDAP server (`ldap-server.example.com` in this document):

```
sudo apt install sasl2-bin
```

### Check the hostname

Get the hostname from the server

```
hostname -f
```

Which should give you the hostname of:

```
ldap-server.example.com
```

Also check the hostname and domain using a reverse lookup with your IP. For example, if the IP address is `10.10.17.91`:

```
nslookup 10.10.17.91
```

The reply should look like this:

```
91.17.10.10.in-addr.arpa        name = ldap-server.example.com.
```

If the result is the same as your host's canonical name them all is well. If the domain is missing, the Fully Qualified Domain Name (FQDN) can be entered in the `/etc/hosts` file.

```
sudo vi /etc/hosts
```

Add the FQDN before the short hostname. Using the same IP as in the previous example, we would have:

```
10.10.17.91 ldap-server.example.com ldap-server
```

### Create the saslauthd principal

The `saslauthd` daemon needs a Kerberos service principal in order to authenticate itself to the Kerberos server. Such principals are created with a random password, and the resulting key is stored in `/etc/krb5.keytab`. For more information about Kerberos service principals, please consult *How to configure Kerberos service principals*.

The simplest way to create this principal, and extract the key safely, is to run the `kadmin` tool remotely, instead of on the Kerberos server. Since the key needs to be written to `/etc/krb5.keytab`, the tool needs to be run with root privileges. Additionally, since creating a new service principal, as well as extracting its key, are privileged operations, we need an `/admin` instance of a principal in order to be allowed these actions. In this example, we will use `ubuntu/admin`:

```
sudo kadmin -p ubuntu/admin
```

An interactive session will look like below. Note the two commands we are issuing at the `kadmin:` prompt: `addprinc` and `ktadd`:

```
Authenticating as principal ubuntu/admin with password.
Password for ubuntu/admin@EXAMPLE.COM
kadmin: addprinc -randkey host/ldap-server.example.com
No policy specified for host/ldap-server.example.com@EXAMPLE.COM; defaulting to no
policy
Principal "host/ldap-server.example.com@EXAMPLE.COM" created.
kadmin: ktadd host/ldap-server.example.com
Entry for principal host/ldap-server.example.com with kvno 2, encryption type
aes256-cts-hmac-sha1-96 added to keytab FILE:/etc/krb5.keytab.
Entry for principal host/ldap-server.example.com with kvno 2, encryption type
aes128-cts-hmac-sha1-96 added to keytab FILE:/etc/krb5.keytab.
```

Alternatively, we can issue the commands directly:

```
kadmin -p ubuntu/admin -q "addprinc -randkey host/ldap-server.example.com"
```

> *NOTE*
>
> `sudo` is not needed to remotely create a new principal.

And:

```
sudo kadmin -p ubuntu/admin -q "ktadd host/ldap-server.example.com"
```

To check that the service principal was added to `/etc/krb5.keytab`, run this command:

```
sudo klist -k
```

You should see the following:

```
Keytab name: FILE:/etc/krb5.keytab
KVNO Principal
---- --------------------------------------------------------------------------
   2 host/ldap-server.example.com@EXAMPLE.COM
   2 host/ldap-server.example.com@EXAMPLE.COM
```

### Configure saslauthd

We now need to configure `saslauthd` such that it uses Kerberos authentication. This is an option that is selected at startup time, via command-line options. The configuration file for such options is `/etc/default/saslauthd`. Only one change is needed in this file: update `MECHANISMS` to `kerberos5`:

```
sudo vi /etc/default/saslauthd
```

Make the following change:

```
...
# Which authentication mechanisms should saslauthd use? (default: pam)
#
# Available options in this Debian package:
# getpwent  -- use the getpwent() library function
```

(continues on next page)

(continued from previous page)

```
# kerberos5 -- use Kerberos 5
# pam       -- use PAM
# rimap     -- use a remote IMAP server
# shadow    -- use the local shadow password file
# sasldb    -- use the local sasldb database file
# ldap      -- use LDAP (configuration is in /etc/saslauthd.conf)
#
# Only one option may be used at a time. See the saslauthd man page
# for more information.
#
# Example: MECHANISMS="pam"
MECHANISMS="kerberos5"
...
```

**IMPORTANT**:

For Ubuntu version 22.04 and earlier "`START=yes`" must also be added to the default config file for `saslauthd` to start.

Save and exit the editor.

### Enable and start saslauthd

Continue by enabling and starting the saslauthd service.

```
sudo systemctl enable --now saslauthd
```

### Test saslauthd configuration

The `saslauthd` service can be tested with with the `testsaslauthd` command. For example, with the correct Kerberos password for the `ubuntu` principal:

```
testsaslauthd -u ubuntu -p ubuntusecret
0: OK "Success."
```

And with the wrong Kerberos password:

```
testsaslauthd -u ubuntu -p ubuntusecretwrong
0: NO "authentication failed"
```

**NOTE**:

In Ubuntu 22.04 LTS and earlier, the `/run/saslauthd` directory is restricted to members of the `sasl` group, so the `testsaslauthd` commands above need to be run as root (via `sudo`) or as a user who is in the `sasl` group.

### Configure OpenLDAP

In order for OpenLDAP to perform passthrough authentication using `saslauthd`, we need to create the configuration file `/etc/ldap/sasl2/slapd.conf` with the following content:

```
pwcheck_method: saslauthd
```

This will direct OpenLDAP to use `saslauthd` as the password checking mechanism when performing passthrough authentication on behalf of a user.

In Ubuntu 22.04 LTS and earlier, the `openldap` system user needs to be added to the `sasl` group, or else it will not have permission to contact the `saslauthd` unix socket in `/run/saslauthd/`. To make this change, run:

```
sudo gpasswd -a openldap sasl
```

Finally, restart the OpenLDAP service:

```
sudo systemctl restart slapd.service
```

### Change the `userPassword` attribute

What triggers OpenLDAP to perform a passthrough authentication when processing a simple bind authentication request, is the special content of the `userPassword` attribute. Normally, that attribute contains some form of password hash, which is used to authenticate the request. If, however, what it contains is in the format of `{SASL}username@realm`, then OpenLDAP will delegate the authentication to the SASL library, whose configuration is in that file we just created above.

For example, let's examine the directory entry below:

```
dn: uid=ubuntu,dc=example,dc=com
uid: ubuntu
objectClass: account
objectClass: simpleSecurityObject
userPassword: {SSHA}S+WlmGneLDFeCwErKnY4mJngnVJMZAM5
```

If a simple bind is performed using a binddn of `uid=ubuntu,dc=example,dc=com`, the password will be checked against the hashed `userPassword` value as normal. That is, no passthrough authentication will be done at all.

However, if the `userPassword` attribute is in this format:

```
userPassword: {SASL}ubuntu@EXAMPLE.COM
```

That will trigger the passthrough authentication, because the `userPassword` attribute starts with the special prefix `{SASL}`. This will direct OpenLDAP to use `saslauthd` for the authentication, and use the name provided in the `userPassword` attribute.

> **IMPORTANT**
>
> Note how the username present in the `userPassword` attribute is independent of the binddn used in the simple bind! If the `userPassword` attribute contained, say, `{SASL}anotheruser@EXAMPLE.COM`, OpenLDAP would ask `saslauthd` to authenticate `anotheruser@EXAMPLE.COM`, and not the user from the binddn! Therefore, it's important to use OpenLDAP ACLs to prevent users from changing the `userPassword` attribute when using passthrough authentication!

To continue with this how-to, let's create the `uid=ubuntu` entry in the directory, which will use passthrough authentication:

```
ldapadd -x -D cn=admin,dc=example,dc=com -W <<LDIF
dn: uid=ubuntu,dc=example,dc=com
uid: ubuntu
objectClass: account
objectClass: simpleSecurityObject
userPassword: {SASL}ubuntu@EXAMPLE.COM
LDIF
```

**NOTE**

Note how we don't need to add the posix attributes like user id, home directory, group, etc. All we really need is a directory entry that contains a `userPassword` attribute.

### Test the authentication

To test that the simple bind is working, and using the Kerberos password for the `ubuntu` user, let's use `ldapwhoami`:

```
ldapwhoami -D uid=ubuntu,dc=example,dc=com -W -x
Enter LDAP Password:
```

A successful bind will look like

```
dn:uid=ubuntu,dc=example,dc=com
```

A failed bind will look like

```
ldap_bind: Invalid credentials (49)
```

These LDAP bind DNs can now be used with external applications that only support "simple" username and password authentication, and they will be authenticated against Kerberos behind the scenes.

### Troubleshooting

Saslauthd can be run in debug mode for more verbose messages to aid in troubleshooting

```
sudo systemctl stop saslauthd
sudo /usr/sbin/saslauthd -a kerberos5 -d -m /var/run/saslauthd -n 1
```

Also the `/var/log/auth.log` file can be checked for saslauthd log entries

### Advanced options

Saslauthd can be configured in the `/etc/saslauthd.conf` file. The settings depend on the authorization mechanism configured in `/etc/default/saslauthd`, which in this case is `kerberos5`.

Some options are:

- `krb5_keytab`: Override the default keytab file location of `/etc/krb5.keytab`.
- `krb5_verify_principal`: Change the name of the Kerberos service principal used by `saslauthd`.

For example, with this content in `/etc/saslauthd.conf`:

```
krb5_keytab: /etc/saslauthd.keytab
krb5_verify_principal: saslauthd
```

We have changed the keytab file to `/etc/saslauthd.keytab`, and the service principal that `saslauthd` will use to authenticate itself with the Kerberos server becomes `saslauthd/ldap-server.example.com@EXAMPLE.COM` (following this how-to). Note how the domain and realm names remain the same in this service principal, and only the actual principal name is affected (changed from the default value of `host` to `saslauthd`).

### References

- OpenLDAP Passthrough Authentication
- Cyrus SASL Password Verification
- Cyrus SASL Slapd Configuration File
- Kerberos Client Hostname Requirements

### See also

- How-to: *How to set up SSSD with LDAP*
- Explanation: *Introduction to OpenLDAP*

## Databases

Ubuntu provides two popular database servers: MySQL and PostgreSQL. Both are popular choices with similar feature sets, and both are equally supported in Ubuntu.

These guides show you how to install and configure them.

### Install and configure a MySQL server

MySQL is a fast, multi-threaded, multi-user, and robust SQL database server. It is intended for mission-critical, heavy-load production systems and mass-deployed software.

### Install MySQL

To install MySQL, run the following command from a terminal prompt:

```
sudo apt install mysql-server
```

Once the installation is complete, the MySQL server should be started automatically. You can quickly check its current status via systemd:

```
sudo service mysql status
```

Which should provide an output like the following:

```
● mysql.service - MySQL Community Server
   Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset:
```

(continues on next page)

(continued from previous page)

```
enabled)
   Active: active (running) since Tue 2019-10-08 14:37:38 PDT; 2 weeks 5 days ago
 Main PID: 2028 (mysqld)
    Tasks: 28 (limit: 4915)
   CGroup: /system.slice/mysql.service
           └─2028 /usr/sbin/mysqld --daemonize --pid-file=/run/mysqld/mysqld.pid
 Oct 08 14:37:36 db.example.org systemd[1]: Starting MySQL Community Server...
Oct 08 14:37:38 db.example.org systemd[1]: Started MySQL Community Server.
```

The network status of the MySQL service can also be checked by running the `ss` command at the terminal prompt:

```
sudo ss -tap | grep mysql
```

When you run this command, you should see something similar to the following:

```
LISTEN    0         151              127.0.0.1:mysql             0.0.0.0:*
users:(("mysqld",pid=149190,fd=29))
LISTEN    0         70                     *:33060                 *:*
users:(("mysqld",pid=149190,fd=32))
```

If the server is not running correctly, you can type the following command to start it:

```
sudo service mysql restart
```

A good starting point for troubleshooting problems is the systemd journal, which can be accessed from the terminal prompt with this command:

```
sudo journalctl -u mysql
```

### Configure MySQL

You can edit the files in `/etc/mysql/` to configure the basic settings – log file, port number, etc. For example, to configure MySQL to listen for connections from network hosts, in the file `/etc/mysql/mysql.conf.d/mysqld.cnf`, change the `bind-address` directive to the server's IP address:

```
bind-address            = 192.168.0.5
```

**Note**

Replace `192.168.0.5` with the appropriate address, which can be determined via the `ip address show` command.

After making a configuration change, the MySQL daemon will need to be restarted with the following command:

```
sudo systemctl restart mysql.service
```

### User setup

By default, `mysql-server` initially provides a `'root'@'localhost'` user for managing the server locally. You can enter the MySQL command-line as this user by running:

```
sudo mysql -u root
```

No password is required by MySQL as it authenticates with auth_socket.

### Create a new user

From the command-line, you can create additional MySQL users with different privileges using the `CREATE USER` command. For authentication, the two main options are to use a password or use a socket like the root user.

To create a user authenticated with a password, you can use MySQL's provided `caching_sha2_password` plugin. It can be invoked in the following way, providing the password in plaintext:

```
CREATE USER 'username'@'localhost' IDENTIFIED WITH caching_sha2_password BY
'password';
```

A random password can also be generated here with:

```
CREATE USER 'username'@'localhost' IDENTIFIED WITH caching_sha2_password BY RANDOM
PASSWORD;
```

MySQL's upstream documentation provides an overview of additional options when creating accounts with passwords.

Socket-based authentication is used to allow a local system user to access an account without entering a password. Invoke this with:

```
CREATE USER 'username'@'localhost' IDENTIFIED WITH auth_socket;
```

By default, only the system user with the matching username can access this account. If you want the MySQL account username to differ from the system user username, then use the `AS` option:

```
CREATE USER 'username'@'localhost' IDENTIFIED WITH auth_socket AS 'system-user-
username';
```

### Adding user permissions

A newly created user will require privilege updates to interact with databases in any way. These are provided by the `GRANT` command alongside specified roles or operations. For example, to give your user the ability to view table entries using the `SELECT` operation on all databases, run the following:

```
GRANT SELECT on *.* TO 'username'@'localhost';
```

### Database engines

Whilst the default configuration of MySQL provided by the Ubuntu packages is perfectly functional and performs well there are things you may wish to consider before you proceed.

MySQL is designed to allow data to be stored in different ways. These methods are referred to as either database or storage engines. There are two main storage engines that you'll be interested in: InnoDB and MyISAM. Storage engines are transparent to the end user. MySQL will handle things differently under the surface, but regardless of which storage engine is in use, you will interact with the database in the same way.

Each engine has its own advantages and disadvantages.

While it is possible (and may be advantageous) to mix and match database engines on a table level, doing so reduces the effectiveness of the performance tuning you can do as you'll be splitting the resources between two engines instead of dedicating them to one.

### InnoDB

As of MySQL 5.5, InnoDB is the default engine, and is highly recommended over MyISAM unless you have specific needs for features unique to that engine.

InnoDB is a more modern database engine, designed to be ACID compliant which guarantees database transactions are processed reliably. To meet ACID compliance all transactions are journaled independently of the main tables. This allows for much more reliable data recovery as data consistency can be checked.

Write locking can occur on a row-level basis within a table. That means multiple updates can occur on a single table simultaneously. Data caching is also handled in memory within the database engine, allowing caching on a more efficient row-level basis rather than file block.

### MyISAM

MyISAM is the older of the two. It can be faster than InnoDB under certain circumstances and favours a read-only workload. Some web applications have been tuned around MyISAM (though that's not to imply that they will be slower under InnoDB).

MyISAM also supports the *FULLTEXT* index type, which allows very fast searches of large quantities of text data. However MyISAM is only capable of locking an entire table for writing. This means only one process can update a table at a time. As any application that uses the table scales this may prove to be a hindrance.

It also lacks journaling, which makes it harder for data to be recovered after a crash. The following link provides some points for consideration about using MyISAM on a production database.

### Backups

MySQL databases should be backed up regularly. Backups can be accomplished through several methods, of which we'll discuss three here.

*mysqldump* is included with `mysql-server`. It is useful for backing up smaller databases, allows backups to be edited prior to a restore, and can be used for exporting to CSV and XML.

*MySQL Shell's Dump Utility* allows for backups of specific schema and tables, both to local

files and remote secure servers. It is recommended for creating partial backups, and for integration with Python programs.

*Percona Xtrabackup* creates full backups with far greater performance than the former options. However, it lacks the ability to customize schema and tables. It is the recommended option for backing up large databases in a production environment.

### mysqldump

`mysqldump` is a built-in tool that performs logical backups for MySQL.

To dump the data of a publicly available database on the local MySQL server into a file, run the following:

```
mysqldump [database name] > dump.sql
```

For restricted databases, specify a user with the proper permissions using `-u`:

```
mysqldump -u root [database name] > dump.sql
```

To restore a database from the backup file, run the `mysql` command and pipe the file through stdin:

```
mysql -u root [database name] < dump.sql
```

See the upstream documentation for more information.

### MySQL Shell Dump Utility

MySQL Shell, supported in Ubuntu 24.04 LTS and later, contains a set of utilities for dumping, backing up, and restoring MySQL data. It provides a programmatic option for logical backups with filtering options.

To install MySQL Shell, run the following:

```
sudo apt install mysql-shell
```

Run the following to connect to the local MySQL server on Ubuntu with MySQL Shell in Python mode:

```
mysqlsh --socket=/var/run/mysqld/mysqld.sock --no-password --python
```

Initiate a local backup of all data in Python mode with:

```
util.dump_instance("/tmp/worlddump")
```

Dump a specific set of tables with `dump_tables`:

```
util.dump_tables("database name", ["table 1", "table 2"], "/tmp/tabledump")
```

To restore dumped data, use the dump loading utility.

```
util.load_dump("/tmp/worlddump")
```

> **Note**
>
> To restore data from a local file, `local_infile` needs to be enabled on the MySQL server. Activate this by accessing the server with the `mysql` command and entering `SET GLOBAL local_infile=1;`.

See the MySQL Shell dump documentation for more information.

### Percona Xtrabackup

Also supported in Ubuntu 24.04 LTS and later, Percona Xtrabackup is a tool for creating physical backups. It is similar to the commercial offering of MySQL Enterprise Backup.

To install Xtrabackup, run the following command from a terminal prompt:

```
sudo apt install percona-xtrabackup
```

Create a new backup with the `xtrabackup` command. This can be done while the server is running.

```
xtrabackup --backup --target-dir=/tmp/worlddump
```

To restore from a backup, service will need to be interrupted. This can be achieved with the following:

```
sudo systemctl stop mysql
xtrabackup --prepare --target-dir=/tmp/worlddump
sudo rm -rf /var/lib/mysql
sudo xtrabackup --copy-back --target-dir=/tmp/worlddump --datadir=/var/lib/mysql
sudo chown -R mysql:mysql /var/lib/mysql
sudo systemctl start mysql
```

For more information, see Percona's upstream documentation.

## Advanced configuration

### Creating a tuned configuration

There are a number of parameters that can be adjusted within MySQL's configuration files. This will allow you to improve the server's performance over time.

Many parameters can be adjusted with the existing database, however some may affect the data layout and thus need more care to apply.

First, if you have existing data, you will first need to carry out a `mysqldump` and reload:

```
mysqldump --all-databases --routines -u root -p > ~/fulldump.sql
```

This will then prompt you for the root password before creating a copy of the data. It is advisable to make sure there are no other users or processes using the database while this takes place. Depending on how much data you've got in your database, this may take a while. You won't see anything on the screen during the process.

Once the dump has been completed, shut down MySQL:

```
sudo service mysql stop
```

It's also a good idea to backup the original configuration:

```
sudo rsync -avz /etc/mysql /root/mysql-backup
```

Next, make any desired configuration changes. Then, delete and re-initialise the database space and make sure ownership is correct before restarting MySQL:

```
sudo rm -rf /var/lib/mysql/*
sudo mysqld --initialize
sudo chown -R mysql: /var/lib/mysql
sudo service mysql start
```

The final step is re-importation of your data by piping your SQL commands to the database.

```
cat ~/fulldump.sql | mysql
```

For large data imports, the 'Pipe Viewer' utility can be useful to track import progress. Ignore any ETA times produced by `pv`; they're based on the average time taken to handle each row of the file, but the speed of inserting can vary wildly from row to row with `mysqldumps`:

```
sudo apt install pv
pv ~/fulldump.sql | mysql
```

Once this step is complete, you are good to go!

> **Note**
>
> This is not necessary for all `my.cnf` changes. Most of the variables you can change to improve performance are adjustable even whilst the server is running. As with anything, make sure to have a good backup copy of your config files and data before making changes.

### MySQL Tuner

MySQL Tuner is a Perl script that connects to a running MySQL instance and offers configuration suggestions for optimising the database for your workload. The longer the server has been running, the better the advice `mysqltuner` can provide. In a production environment, consider waiting for at least 24 hours before running the tool. You can install `mysqltuner` with the following command:

```
sudo apt install mysqltuner
```

Then once it has been installed, simply run: `mysqltuner` – and wait for its final report.

The top section provides general information about the database server, and the bottom section provides tuning suggestions to alter in your `my.cnf`. Most of these can be altered live on the server without restarting; look through the official MySQL documentation for the relevant variables to change in production.

The following example is part of a report from a production database showing potential benefits from increasing the query cache:

```
-------- Recommendations ------------------------------------------------------
General recommendations:
    Run OPTIMIZE TABLE to defragment tables for better performance
    Increase table_cache gradually to avoid file descriptor limits
Variables to adjust:
    key_buffer_size (> 1.4G)
    query_cache_size (> 32M)
    table_cache (> 64)
    innodb_buffer_pool_size (>= 22G)
```

Obviously, performance optimisation strategies vary from application to application; what works best for WordPress might not be the best for Drupal or Joomla. Performance can depend on the types of queries, use of indexes, how efficient the database design is and so on.

You may find it useful to spend some time searching for database tuning tips based on the applications you're using. Once you've reached the point of diminishing returns from database configuration adjustments, look to the application itself for improvements, or invest in more powerful hardware and/or scale up the database environment.

### Further reading

- Full documentation is available in both online and offline formats from the MySQL Developers portal
- For general SQL information see the O'Reilly books Getting Started with SQL: A Hands-On Approach for Beginners by Thomas Nield as an entry point and SQL in a Nutshell as a quick reference.

## Install and configure PostgreSQL

PostgreSQL (commonly referred to as "Postgres") is an object-relational database system that has all the features of traditional commercial database systems, but with enhancements to be found in next-generation database management systems (DBMS).

### Install PostgreSQL

To install PostgreSQL, run the following command in the command prompt:

```
sudo apt install postgresql
```

The database service is automatically configured with viable defaults, but can be customised based on your specific needs.

### Configure PostgreSQL

PostgreSQL supports multiple client authentication methods. In Ubuntu, `peer` is the default authentication method used for `local` connections, while `scram-sha-256` is the default for `host` connections (this used to be `md5` until Ubuntu 21.10). Please refer to the PostgreSQL Administrator's Guide if you would like to configure alternatives like Kerberos.

The following discussion assumes that you wish to enable TCP/IP connections and use the `scram-sha-256` method for client authentication. PostgreSQL configuration files are stored

in the `/etc/postgresql/<version>/main` directory. For example, if you install PostgreSQL 14, the configuration files are stored in the `/etc/postgresql/14/main` directory.

> **Tip**
>
> To configure *IDENT* authentication, add entries to the `/etc/postgresql/*/main/pg_ident.conf` file. There are detailed comments in the file to guide you.

By default, only connections from the local system are allowed. To enable all other computers to connect to your PostgreSQL server, edit the file `/etc/postgresql/*/main/postgresql.conf`. Locate the line: *#listen_addresses = 'localhost'* and change it to `*`:

```
listen_addresses = '*'
```

> **Note**
>
> To listen on all IPv4 interfaces, set `listen_addresses` to '`0.0.0.0`', while '`::`' will listen on all IPv6 interfaces. '*' will cause PostgreSQL to listen on all available network interfaces, both IPv4 and IPv6.

For details on other parameters, refer to the configuration file or to the PostgreSQL documentation for information on how they can be edited.

Now that we can connect to our PostgreSQL server, the next step is to set a password for the `postgres` user. Run the following command at a terminal prompt to connect to the default PostgreSQL template database:

```
sudo -u postgres psql template1
```

The above command connects to PostgreSQL database `template1` as user `postgres`. Once you connect to the PostgreSQL server, you will be at an SQL prompt. You can run the following SQL command at the `psql` prompt to configure the password for the user `postgres`:

```
ALTER USER postgres with encrypted password 'your_password';
```

After configuring the password, edit the file `/etc/postgresql/*/main/pg_hba.conf` to use `scram-sha-256` authentication with the `postgres` user, allowed for the `template1` database, from any system in the local network (which in the example is `192.168.1.1/24`) :

```
hostssl template1       postgres        192.168.1.1/24        scram-sha-256
```

> **Note**
>
> The config statement `hostssl` used here will reject TCP connections that would not use SSL. PostgreSQL in Ubuntu has the SSL feature built in and configured by default, so it works right away. On your PostgreSQL server this uses the certificate created by `ssl-cert` package which is great, but for production use you should consider updating that with a proper certificate from a recognised Certificate Authority (CA).

Finally, you should restart the PostgreSQL service to initialise the new configuration. From a terminal prompt enter the following to restart PostgreSQL:

```
sudo systemctl restart postgresql.service
```

> **Warning**
>
> The above configuration is not complete by any means. Please refer to the PostgreSQL Administrator's Guide to configure more parameters.

You can test server connections from other machines by using the PostgreSQL client as follows, replacing the domain name with your actual server domain name or IP address:

```
sudo apt install postgresql-client
psql --host your-servers-dns-or-ip --username postgres --password --dbname
template1
```

### Streaming replication

PostgreSQL has a nice feature called **streaming replication** which provides the ability to continuously ship and apply the Write-Ahead Log (WAL) XLOG records to some number of standby servers to keep them current. Here is a simple way to replicate a PostgreSQL server (main) to a standby server.

First, create a replication user in the main server, to be used from the standby server:

```
sudo -u postgres createuser --replication -P -e replicator
```

Let's configure the main server to turn on the streaming replication. Open the file `/etc/postgresql/*/main/postgresql.conf` and make sure you have the following lines:

```
listen_addresses = '*'
wal_level = replica
```

Also edit the file `/etc/postgresql/*/main/pg_hba.conf` to add an extra line to allow the standby server connection for replication (that is a special keyword) using the `replicator` user:

```
host  replication   replicator   <IP address of the standby>      scram-sha-256
```

Restart the service to apply changes:

```
sudo systemctl restart postgresql
```

Now, in the standby server, let's stop the PostgreSQL service:

```
sudo systemctl stop postgresql
```

Edit the `/etc/postgresql/*/main/postgresql.conf` to set up hot standby:

```
hot_standby = on
```

Back up the current state of the main server (those commands are still issued on the standby system):

```
sudo su - postgres
# backup the current content of the standby server (update the version of your
postgres accordingly)
cp -R /var/lib/postgresql/14/main /var/lib/postgresql/14/main_bak
# remove all the files in the data directory
rm -rf /var/lib/postgresql/14/main/*
pg_basebackup -h <IP address of the main server> -D /var/lib/postgresql/14/main -U
replicator -P -v -R
```

After this, a full single pass will have been completed, copying the content of the main database onto the local system being the standby. In the `pg_basebackup` command the flags represent the following:

- `-h`: The *hostname* or IP address of the main server
- `-D`: The data directory
- `-U`: The user to be used in the operation
- `-P`: Turns on progress reporting
- `-v`: Enables verbose mode
- `-R`: Creates a `standby.signal` file and appends connection settings to `postgresql.auto.conf`

Finally, let's start the PostgreSQL service on standby server:

```
sudo systemctl start postgresql
```

To make sure it is working, go to the main server and run the following command:

```
sudo -u postgres psql -c "select * from pg_stat_replication;"
```

As mentioned, this is a very simple introduction, there are way more great details in the upstream documentation about the configuration of replication as well as further High Availability, Load Balancing, and Replication.

To test the replication you can now create a test database in the main server and check if it is replicated in the standby server:

```
sudo -u postgres createdb test # on the main server
sudo -u postgres psql -c "\l" # on the standby server
```

You need to be able to see the `test` database, that was created on the main server, in the standby server.

### Backups

PostgreSQL databases should be backed up regularly. Refer to the PostgreSQL Administrator's Guide for different approaches.

### Further reading

- As mentioned above, the PostgreSQL Administrator's Guide is an excellent resource. The guide is also available in the `postgresql-doc` package. Execute the following in a terminal to install the package:

  ```
  sudo apt install postgresql-doc
  ```

  This package provides further manpages on PostgreSQL *dblink* and "server programming interface" as well as the upstream HTML guide. To view the guide enter `xdg-open /usr/share/doc/postgresql-doc-*/html/index.html` or point your browser at it.

- For general SQL information see the O'Reilly books Getting Started with SQL: A Hands-On Approach for Beginners by Thomas Nield as an entry point and SQL in a Nutshell as a quick reference.

### See also

- Explanation: *Introduction to databases*

### Storage and backups

- *Storage* shows how to set up and manage Logical Volumes
- *Backups and version control* presents common options for backing up your data and your system

### Storage

- The Ubuntu Server installer can set up and install to *Logical Volume Management (LVM)* partitions.
- Ubuntu Server can be configured as both an *iSCSI initiator and an iSCSI target.*

### How to manage logical volumes

The Ubuntu Server installer has the ability to set up and install to LVM partitions, and this is the supported way of doing so. If you would like to know more about any of the topics in this page, refer to our *explanation of logical volume management (LVM)*.

### Create the physical volume

First, you need a physical volume. Typically you start with a hard disk, and create a regular partition whose type is "LVM" on it. You can create it with `gparted` or `fdisk`, and usually only want one LVM-type partition in the whole disk, since LVM will handle subdividing it into logical volumes. In `gparted`, you need to check the `lvm` flag when creating the partition, and with `fdisk`, tag the type with code `8e`.

Once you have your LVM partition, you need to initialise it as a physical volume. Assuming this partition is `/dev/sda1`:

```
sudo pvcreate /dev/sda1
```

### Create the volume group

After that, you can create a volume group; in our example, it will be named `foo` and uses only one physical volume:

```
sudo vgcreate foo /dev/sda1
```

### Create a logical volume

Now you want to create a logical volume from some of the free space in `foo`:

```
sudo lvcreate --name bar --size 5g foo
```

This creates a logical volume named `bar` in volume group `foo` using 5 GB of space. You can find the block device for this logical volume in `/dev/foo/bar or dev/mapper/foo-bar`.

You might also want to try the `lvs` and `pvs` commands, which list the logical volumes and physical volumes respectively, and their more detailed variants; `lvdisplay` and `pvdisplay`.

### Resize a partition

You can extend a logical volume with:

```
sudo lvextend --resizefs --size +5g foo/bar
```

This will add 5 GB to the `bar` logical volume in the `foo` volume group, and will automatically resize the underlying *filesystem* (if supported). The space is allocated from free space anywhere in the `bar` volume group. You can specify an absolute size instead of a relative size if you want by omitting the leading `+`.

If you have multiple physical volumes you can add the names of one (or more) of them to the end of the command to limit which ones are used to fulfil the request.

### Move a partition

If you only have one physical volume then you are unlikely to ever need to move, but if you add a new disk, you might want to. To move the logical volume `bar` off of physical volume `/dev/sda1`, you can run:

```
sudo pvmove --name bar /dev/sda1
```

If you omit the `--name bar` argument, then all logical volumes on the `/dev/sda1` physical volume will be moved. If you only have one other physical volume then that is where it will be moved to. Otherwise you can add the name of one or more specific physical volumes that should be used to satisfy the request, instead of any physical volume in the volume group with free space.

This process can be resumed safely if interrupted by a crash or power failure, and can be done while the logical volume(s) in question are in use. You can also add `--background` to perform the move in the background and return immediately, or `--interval s` to have it print how much progress it has made every `s` seconds. If you background the move, you can check its progress with the `lvs` command.

### Create a snapshot

When you create a snapshot, you create a new logical volume to act as a clone of the original logical volume.

The snapshot volume does not initially use any space, but as changes are made to the original volume, the changed blocks are copied to the snapshot volume before they are changed in order to preserve them. This means that the more changes you make to the origin, the more space the snapshot needs. If the snapshot volume uses all of the space allocated to it, then the snapshot is broken and can not be used any more, leaving you with only the modified origin.

The `lvs` command will tell you how much space has been used in a snapshot logical volume. If it starts to get full, you might want to extend it with the `lvextend` command. To create a snapshot of the bar logical volume and name it `lv_snapshot`, run:

```
sudo lvcreate --snapshot --name lv_snapshot --size 5g foo/bar
```

This will create a snapshot named `lv_snapshot` of the original logical volume `bar` and allocate 5 GB of space for it. Since the snapshot volume only stores the areas of the disk that have changed since it was created, it can be much smaller than the original volume.

While you have the snapshot, you can mount it if you wish to see the original filesystem as it appeared when you made the snapshot. In the above example you would mount the `/dev/foo/lv_snapshot` device. You can modify the snapshot without affecting the original, and the original without affecting the snapshot. For example, if you take a snapshot of your root logical volume, make changes to your system, and then decide you would like to revert the system back to its previous state, you can merge the snapshot back into the original volume, which effectively reverts it to the state it was in when you made the snapshot. To do this, you can run:

```
sudo lvconvert --merge foo/lv_snapshot
```

If the origin volume of `foo/lv_snapshot` is in use, it will inform you that the merge will take place the next time the volumes are activated. If this is the root volume, then you will need to reboot for this to happen. At the next boot, the volume will be activated and the merge will begin in the background, so your system will boot up as if you had never made the changes since the snapshot was created, and the actual data movement will take place in the background while you work.

### iSCSI initiator (or client)

> Wikipedia iSCSI Definition:
>
> iSCSI an acronym for **Internet Small Computer Systems Interface** , an Internet Protocol (IP)-based storage networking standard for linking data storage facilities. It provides block-level access to storage devices by carrying SCSI commands over a TCP/IP network.
>
> iSCSI is used to facilitate data transfers over intranets and to manage storage over long distances. It can be used to transmit data over local area networks (LANs), wide area networks (WANs), or the Internet and can enable location-independent data storage and retrieval.
>
> The protocol allows clients (called *initiators*) to send SCSI commands (*CDBs*) to

storage devices (*targets*) on remote servers. It is a storage area network (SAN) protocol, allowing organizations to consolidate storage into storage arrays while providing clients (such as database and web servers) with the illusion of locally attached SCSI disks.

It mainly competes with Fibre Channel, but unlike traditional *Fibre Channel*, which usually requires dedicated cabling, iSCSI can be run over long distances using existing network infrastructure.

Ubuntu Server can be configured as both: **iSCSI initiator** and **iSCSI target**. This guide provides commands and configuration options to setup an **iSCSI initiator** (or Client).

**Note**

It is assumed that **you already have an iSCSI target on your local network** and have the appropriate rights to connect to it. The instructions for setting up a target vary greatly between hardware providers, so consult your vendor documentation to configure your specific iSCSI target.

### Network Interfaces Configuration

Before start configuring iSCSI, make sure to have the network interfaces correctly set and configured in order to have open-iscsi package to behave appropriately, specially during boot time. In Ubuntu 20.04 LTS, the default network configuration tool is netplan.io.

For all the iSCSI examples below please consider the following netplan configuration for my iSCSI initiator:

*/etc/cloud/cloud.cfg.d/99-disable-network-config.cfg*

```
{config: disabled}
```

*/etc/netplan/50-cloud-init.yaml*

```
network:
    ethernets:
        enp5s0:
            match:
                macaddress: 00:16:3e:af:c4:d6
            set-name: eth0
            dhcp4: true
            dhcp-identifier: mac
        enp6s0:
            match:
                macaddress: 00:16:3e:50:11:9c
            set-name: iscsi01
            dhcp4: true
            dhcp-identifier: mac
            dhcp4-overrides:
              route-metric: 300
        enp7s0:
            match:
```

(continues on next page)

(continued from previous page)

```
            macaddress: 00:16:3e:b3:cc:50
        set-name: iscsi02
        dhcp4: true
        dhcp-identifier: mac
        dhcp4-overrides:
          route-metric: 300
    version: 2
    renderer: networkd
```

With this configuration, the interfaces names change by matching their mac addresses. This makes it easier to manage them in a server containing multiple interfaces.

From this point and beyond, 2 interfaces are going to be mentioned: **iscsi01** and **iscsi02**. This helps to demonstrate how to configure iSCSI in a multipath environment as well (check the Device Mapper Multipath session in this same Server Guide).

> If you have only a single interface for the iSCSI network, make sure to follow the same instructions, but only consider the **iscsi01** interface command line examples.

### iSCSI Initiator Install

To configure Ubuntu Server as an iSCSI initiator install the open-iscsi package. In a terminal enter:

```
$ sudo apt install open-iscsi
```

Once the package is installed you will find the following files:

- /etc/iscsi/iscsid.conf
- /etc/iscsi/initiatorname.iscsi

### iSCSI Initiator Configuration

Configure the main configuration file like the example bellow:

/etc/iscsi/iscsid.conf

```
### startup settings

## will be controlled by systemd, leave as is
iscsid.startup = /usr/sbin/iscsidnode.startup = manual

### chap settings

# node.session.auth.authmethod = CHAP

## authentication of initiator by target (session)
# node.session.auth.username = username
# node.session.auth.password = password
```

(continues on next page)

(continued from previous page)

```
# discovery.sendtargets.auth.authmethod = CHAP

## authentication of initiator by target (discovery)
# discovery.sendtargets.auth.username = username
# discovery.sendtargets.auth.password = password

### timeouts

## control how much time iscsi takes to propagate an error to the
## upper layer. if using multipath, having 0 here is desirable
## so multipath can handle path errors as quickly as possible
## (and decide to queue or not if missing all paths)
node.session.timeo.replacement_timeout = 0

node.conn[0].timeo.login_timeout = 15
node.conn[0].timeo.logout_timeout = 15

## interval for a NOP-Out request (a ping to the target)
node.conn[0].timeo.noop_out_interval = 5

## and how much time to wait before declaring a timeout
node.conn[0].timeo.noop_out_timeout = 5

## default timeouts for error recovery logics (lu & tgt resets)
node.session.err_timeo.abort_timeout = 15
node.session.err_timeo.lu_reset_timeout = 30
node.session.err_timeo.tgt_reset_timeout = 30

### retry

node.session.initial_login_retry_max = 8

### session and device queue depth

node.session.cmds_max = 128
node.session.queue_depth = 32

### performance

node.session.xmit_thread_priority = -20
```

and re-start the iSCSI daemon:

```
$ systemctl restart iscsid.service
```

This will set basic things up for the rest of configuration.

The other file mentioned:

/etc/iscsi/initiatorname.iscsi

```
InitiatorName=iqn.2004-10.com.ubuntu:01:60f3517884c3
```

contains this node's initiator name and is generated during open-iscsi package installation. If you modify this setting, make sure that you don't have duplicates in the same iSCSI SAN (Storage Area Network).

### iSCSI Network Configuration

Before configuring the Logical Units that are going to be accessed by the initiator, it is important to inform the iSCSI service what are the interfaces acting as paths.

A straightforward way to do that is by:

- configuring the following environment variables

```
$ iscsi01_ip=$(ip -4 -o addr show iscsi01 | sed -r 's:.* (([0-9]{1,3}\.){3}[0-9]
{1,3})/.*:\1:')
$ iscsi02_ip=$(ip -4 -o addr show iscsi02 | sed -r 's:.* (([0-9]{1,3}\.){3}[0-9]
{1,3})/.*:\1:')

$ iscsi01_mac=$(ip -o link show iscsi01 | sed -r 's:.*\s+link/ether (([0-f]{2}(\
:|)){6}).*:\1:g')
$ iscsi02_mac=$(ip -o link show iscsi02 | sed -r 's:.*\s+link/ether (([0-f]{2}(\
:|)){6}).*:\1:g')
```

- configuring **iscsi01** interface

```
$ sudo iscsiadm -m iface -I iscsi01 --op=new
New interface iscsi01 added
$ sudo iscsiadm -m iface -I iscsi01 --op=update -n iface.hwaddress -v $iscsi01_mac
iscsi01 updated.
$ sudo iscsiadm -m iface -I iscsi01 --op=update -n iface.ipaddress -v $iscsi01_ip
iscsi01 updated.
```

- configuring **iscsi02** interface

```
$ sudo iscsiadm -m iface -I iscsi02 --op=new
New interface iscsi02 added
$ sudo iscsiadm -m iface -I iscsi02 --op=update -n iface.hwaddress -v $iscsi02_mac
iscsi02 updated.
$ sudo iscsiadm -m iface -I iscsi02 --op=update -n iface.ipaddress -v $iscsi02_ip
iscsi02 updated.
```

- discovering the **targets**

```
$ sudo iscsiadm -m discovery -I iscsi01 --op=new --op=del --type sendtargets --
portal storage.iscsi01
10.250.94.99:3260,1 iqn.2003-01.org.linux-iscsi.storage.x8664:sn.2c084c8320ca

$ sudo iscsiadm -m discovery -I iscsi02 --op=new --op=del --type sendtargets --
portal storage.iscsi02
10.250.93.99:3260,1 iqn.2003-01.org.linux-iscsi.storage.x8664:sn.2c084c8320ca
```

- configuring **automatic login**

```
$ sudo iscsiadm -m node --op=update -n node.conn[0].startup -v automatic
$ sudo iscsiadm -m node --op=update -n node.startup -v automatic
```

- make sure needed **services** are enabled during OS initialization:

```
$ systemctl enable open-iscsi
Synchronizing state of open-iscsi.service with SysV service script with /lib/
systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable open-iscsi
Created symlink /etc/systemd/system/iscsi.service  /lib/systemd/system/open-
iscsi.service.
Created symlink /etc/systemd/system/sysinit.target.wants/open-iscsi.service  /
lib/systemd/system/open-iscsi.service.

$ systemctl enable iscsid
Synchronizing state of iscsid.service with SysV service script with /lib/systemd/
systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable iscsid
Created symlink /etc/systemd/system/sysinit.target.wants/iscsid.service  /lib/
systemd/system/iscsid.service.
```

- restarting **iscsid** service

```
$ systemctl restart iscsid.service
```

- and, finally, **login in** discovered logical units

```
$ sudo iscsiadm -m node --loginall=automatic
Logging in to [iface: iscsi02, target: iqn.2003-01.org.linux-iscsi.storage.
x8664:sn.2c084c8320ca, portal: 10.250.93.99,3260] (multiple)
Logging in to [iface: iscsi01, target: iqn.2003-01.org.linux-iscsi.storage.
x8664:sn.2c084c8320ca, portal: 10.250.94.99,3260] (multiple)
Login to [iface: iscsi02, target: iqn.2003-01.org.linux-iscsi.storage.x8664:sn.
2c084c8320ca, portal: 10.250.93.99,3260] successful.
Login to [iface: iscsi01, target: iqn.2003-01.org.linux-iscsi.storage.x8664:sn.
2c084c8320ca, portal: 10.250.94.99,3260] successful.
```

### Accessing the Logical Units (or LUNs)

Check *dmesg* to make sure that the new disks have been detected:

dmesg

```
[  166.840694] scsi 7:0:0:4: Direct-Access     LIO-ORG  TCMU device >
  0002 PQ: 0 ANSI: 5
[  166.840892] scsi 8:0:0:4: Direct-Access     LIO-ORG  TCMU device >
  0002 PQ: 0 ANSI: 5
[  166.841741] sd 7:0:0:4: Attached scsi generic sg2 type 0
[  166.841808] sd 8:0:0:4: Attached scsi generic sg3 type 0
[  166.842278] scsi 7:0:0:3: Direct-Access     LIO-ORG  TCMU device >
```

(continues on next page)

(continued from previous page)

```
  0002 PQ: 0 ANSI: 5
[  166.842571] scsi 8:0:0:3: Direct-Access     LIO-ORG  TCMU device >
  0002 PQ: 0 ANSI: 5
[  166.843482] sd 8:0:0:3: Attached scsi generic sg4 type 0
[  166.843681] sd 7:0:0:3: Attached scsi generic sg5 type 0
[  166.843706] sd 8:0:0:4: [sdd] 2097152 512-byte logical blocks: > (1.
07 GB/1.00 GiB)
[  166.843884] scsi 8:0:0:2: Direct-Access     LIO-ORG  TCMU device >
  0002 PQ: 0 ANSI: 5
[  166.843971] sd 8:0:0:4: [sdd] Write Protect is off
[  166.843972] sd 8:0:0:4: [sdd] Mode Sense: 2f 00 00 00
[  166.844127] scsi 7:0:0:2: Direct-Access     LIO-ORG  TCMU device >
  0002 PQ: 0 ANSI: 5
[  166.844232] sd 7:0:0:4: [sdc] 2097152 512-byte logical blocks: > (1.
07 GB/1.00 GiB)
[  166.844421] sd 8:0:0:4: [sdd] Write cache: enabled, read cache: >
enabled, doesn't support DPO or FUA
[  166.844566] sd 7:0:0:4: [sdc] Write Protect is off
[  166.844568] sd 7:0:0:4: [sdc] Mode Sense: 2f 00 00 00
[  166.844846] sd 8:0:0:2: Attached scsi generic sg6 type 0
[  166.845147] sd 7:0:0:4: [sdc] Write cache: enabled, read cache: >
enabled, doesn't support DPO or FUA
[  166.845188] sd 8:0:0:4: [sdd] Optimal transfer size 65536 bytes
[  166.845527] sd 7:0:0:2: Attached scsi generic sg7 type 0
[  166.845678] sd 8:0:0:3: [sde] 2097152 512-byte logical blocks: > (1.
07 GB/1.00 GiB)
[  166.845785] scsi 8:0:0:1: Direct-Access     LIO-ORG  TCMU device >
  0002 PQ: 0 ANSI: 5
[  166.845799] sd 7:0:0:4: [sdc] Optimal transfer size 65536 bytes
[  166.845931] sd 8:0:0:3: [sde] Write Protect is off
[  166.845933] sd 8:0:0:3: [sde] Mode Sense: 2f 00 00 00
[  166.846424] scsi 7:0:0:1: Direct-Access     LIO-ORG  TCMU device >
  0002 PQ: 0 ANSI: 5
[  166.846552] sd 8:0:0:3: [sde] Write cache: enabled, read cache: >
enabled, doesn't support DPO or FUA
[  166.846708] sd 7:0:0:3: [sdf] 2097152 512-byte logical blocks: > (1.
07 GB/1.00 GiB)
[  166.847024] sd 8:0:0:1: Attached scsi generic sg8 type 0
[  166.847029] sd 7:0:0:3: [sdf] Write Protect is off
[  166.847031] sd 7:0:0:3: [sdf] Mode Sense: 2f 00 00 00
[  166.847043] sd 8:0:0:3: [sde] Optimal transfer size 65536 bytes
[  166.847133] sd 8:0:0:2: [sdg] 2097152 512-byte logical blocks: > (1.
07 GB/1.00 GiB)
[  166.849212] sd 8:0:0:2: [sdg] Write Protect is off
[  166.849214] sd 8:0:0:2: [sdg] Mode Sense: 2f 00 00 00
[  166.849711] sd 7:0:0:3: [sdf] Write cache: enabled, read cache: >
enabled, doesn't support DPO or FUA
[  166.849718] sd 7:0:0:1: Attached scsi generic sg9 type 0
[  166.849721] sd 7:0:0:2: [sdh] 2097152 512-byte logical blocks: > (1.
```

(continues on next page)

(continued from previous page)

```
07 GB/1.00 GiB)
[  166.853296] sd 8:0:0:2: [sdg] Write cache: enabled, read cache: >
enabled, doesn't support DPO or FUA
[  166.853721] sd 8:0:0:2: [sdg] Optimal transfer size 65536 bytes
[  166.853810] sd 7:0:0:2: [sdh] Write Protect is off
[  166.853812] sd 7:0:0:2: [sdh] Mode Sense: 2f 00 00 00
[  166.854026] sd 7:0:0:3: [sdf] Optimal transfer size 65536 bytes
[  166.854431] sd 7:0:0:2: [sdh] Write cache: enabled, read cache: >
enabled, doesn't support DPO or FUA
[  166.854625] sd 8:0:0:1: [sdi] 2097152 512-byte logical blocks: > (1.
07 GB/1.00 GiB)
[  166.854898] sd 8:0:0:1: [sdi] Write Protect is off
[  166.854900] sd 8:0:0:1: [sdi] Mode Sense: 2f 00 00 00
[  166.855022] sd 7:0:0:2: [sdh] Optimal transfer size 65536 bytes
[  166.855465] sd 8:0:0:1: [sdi] Write cache: enabled, read cache: >
enabled, doesn't support DPO or FUA
[  166.855578] sd 7:0:0:1: [sdj] 2097152 512-byte logical blocks: > (1.
07 GB/1.00 GiB)
[  166.855845] sd 7:0:0:1: [sdj] Write Protect is off
[  166.855847] sd 7:0:0:1: [sdj] Mode Sense: 2f 00 00 00
[  166.855978] sd 8:0:0:1: [sdi] Optimal transfer size 65536 bytes
[  166.856305] sd 7:0:0:1: [sdj] Write cache: enabled, read cache: >
enabled, doesn't support DPO or FUA
[  166.856701] sd 7:0:0:1: [sdj] Optimal transfer size 65536 bytes
[  166.859624] sd 8:0:0:4: [sdd] Attached SCSI disk
[  166.861304] sd 7:0:0:4: [sdc] Attached SCSI disk
[  166.864409] sd 8:0:0:3: [sde] Attached SCSI disk
[  166.864833] sd 7:0:0:3: [sdf] Attached SCSI disk
[  166.867906] sd 8:0:0:2: [sdg] Attached SCSI disk
[  166.868446] sd 8:0:0:1: [sdi] Attached SCSI disk
[  166.871588] sd 7:0:0:1: [sdj] Attached SCSI disk
[  166.871773] sd 7:0:0:2: [sdh] Attached SCSI disk
```

In the output above you will find **8 x SCSI disks** recognized. The storage server is mapping **4 x LUNs** to this node, AND the node has **2 x PATHs** to each LUN. The OS recognizes each path to each device as 1 SCSI device.

> You will find different output depending on the storage server your node is mapping the LUNs from, and the amount of LUNs being mapped as well.

Although not the objective of this session, let's find the 4 mapped LUNs using multipath-tools.

> You will find further details about multipath in "Device Mapper Multipathing" session of this same guide.

```
$ apt-get install multipath-tools
```

```
$ sudo multipath -r
```

```
$ sudo multipath -ll
mpathd (360014051a042fb7c41c4249af9f2cfbc) dm-3 LIO-ORG,TCMU device
size=1.0G features='0' hwhandler='0' wp=rw
|-+- policy='service-time 0' prio=1 status=active
| `- 7:0:0:4 sde 8:64  active ready running
`-+- policy='service-time 0' prio=1 status=enabled
  `- 8:0:0:4 sdc 8:32  active ready running
mpathc (360014050d6871110232471d8bcd155a3) dm-2 LIO-ORG,TCMU device
size=1.0G features='0' hwhandler='0' wp=rw
|-+- policy='service-time 0' prio=1 status=active
| `- 7:0:0:3 sdf 8:80  active ready running
`-+- policy='service-time 0' prio=1 status=enabled
  `- 8:0:0:3 sdd 8:48  active ready running
mpathb (360014051f65c6cb11b74541b703ce1d4) dm-1 LIO-ORG,TCMU device
size=1.0G features='0' hwhandler='0' wp=rw
|-+- policy='service-time 0' prio=1 status=active
| `- 7:0:0:2 sdh 8:112 active ready running
`-+- policy='service-time 0' prio=1 status=enabled
  `- 8:0:0:2 sdg 8:96  active ready running
mpatha (36001405b816e24fcab64fb88332a3fc9) dm-0 LIO-ORG,TCMU device
size=1.0G features='0' hwhandler='0' wp=rw
|-+- policy='service-time 0' prio=1 status=active
| `- 7:0:0:1 sdj 8:144 active ready running
`-+- policy='service-time 0' prio=1 status=enabled
  `- 8:0:0:1 sdi 8:128 active ready running
```

Now it is much easier to understand each recognized SCSI device and common paths to same LUNs in the storage server. With the output above one can easily see that:

- **mpatha device** (/dev/mapper/mpatha) is a multipath device for:
  - /dev/sdj
  - /dev/dsi
- **mpathb device** (/dev/mapper/mpathb) is a multipath device for:
  - /dev/sdh
  - /dev/dsg
- **mpathc device** (/dev/mapper/mpathc) is a multipath device for:
  - /dev/sdf
  - /dev/sdd
- **mpathd device** (/dev/mapper/mpathd) is a multipath device for:
  - /dev/sde
  - /dev/sdc

  **Do not use this in production** without checking appropriate multipath configuration options in the **Device Mapper Multipathing** session. The *default multipath configuration* is less than optimal for regular usage.

Finally, to access the LUN (or remote iSCSI disk) you will:

- If accessing through a single network interface:
  - access it through /dev/sdX where X is a letter given by the OS
- If accessing through multiple network interfaces:
  - configure multipath and access the device through /dev/mapper/X

For everything else, the created devices are block devices and all commands used with local disks should work the same way:

- Creating a partition:

```
$ sudo fdisk /dev/mapper/mpatha

Welcome to fdisk (util-linux 2.34).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x92c0322a.

Command (m for help): p
Disk /dev/mapper/mpatha: 1 GiB, 1073741824 bytes, 2097152 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 65536 bytes
Disklabel type: dos
Disk identifier: 0x92c0322a

Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1):
First sector (2048-2097151, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-2097151, default 2097151):

Created a new partition 1 of type 'Linux' and of size 1023 MiB.

Command (m for help): w
The partition table has been altered.
```

- Creating a *filesystem*:

```
$ sudo mkfs.ext4 /dev/mapper/mpatha-part1
mke2fs 1.45.5 (07-Jan-2020)
Creating filesystem with 261888 4k blocks and 65536 inodes
Filesystem UUID: cdb70b1e-c47c-47fd-9c4a-03db6f038988
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376

Allocating group tables: done
```

(continues on next page)

(continued from previous page)

```
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done
```

- Mounting the block device:

```
$ sudo mount /dev/mapper/mpatha-part1 /mnt
```

- Accessing the data:

```
$ ls /mnt
lost+found
```

Make sure to read other important sessions in Ubuntu Server Guide to follow up with concepts explored in this one.

### References

1. iscsid
2. iscsi.conf
3. iscsid.conf
4. iscsi.service
5. iscsid.service
6. Open-iSCSI
7. Debian Open-iSCSI

### See also

- Explanation: *About Logical Volume Management (LVM)*

### Backups and version control

On Ubuntu, two primary ways of backing up your system are **backup utilities** and **shell scripts**. For additional protection, you can combine backup methods.

### Backup utilities

- *Bacula* has advanced features and customization support, which makes it a good choice for enterprise systems or complex setups.
- *rsnapshot* is a simple and efficient solution, well suited to individual users or small-scale organizations.

### How to install and configure Bacula

Bacula is a backup management tool that enables you to backup, restore, and verify data across your network. There are Bacula clients for Linux, Windows, and Mac OS X – making it a cross-platform and network-wide solution.

### Bacula components

Bacula is made up of several components and services that are used to manage backup files and locations:

- **Bacula Director**: A service that controls all backup, restore, verify, and archive operations.
- **Bacula Console**: An application that allows communication with the Director.
- **Bacula File**: This application is installed on machines to be backed up, and is responsible for handling data requested by the Director.
- **Bacula Storage**: The program that performs the storage of data onto, and recovery of data from, the physical media.
- **Bacula Catalog**: Responsible for maintaining the file indices and volume databases for all backed-up files. This enables rapid location and restoration of archived files. The Catalog supports three different databases: MySQL, PostgreSQL, and SQLite.
- **Bacula Monitor**: This is a graphical tray monitor for the Bacula backup system.

These services and applications can be run on multiple servers and clients, or they can be installed on one machine if backing up a single disk or volume.

In this documentation, we will deploy a Bacula Director, with a backup job for the Director itself, and also install the Bacule File service on a workstation, to remotely backup its data.

### Installing the Server Components

The Bacula components can be installed on multiple systems, or they can be grouped together where it makes sense. A fully distributed installation might be appealing and is more scalable, but is also harder to configure. Here we will pick something in between:

- A Bacula “server”, where we will install the following Bacula components: Director, Catalog with an SQL database, Storage, File, Console. The server itself should also be backed up, hence why components typically installed on clients are also installed here.
- A Bacula “client”, which is just a system to be backed up. It will have only the File component installed. Any system that needs to be backed up will have to have the File component installed.

To begin with, we have to start with installing the database that will be used by the Catalog. The choices are:

- sqlite: Should only be used for test or development deployments of Bacula.
- PostgreSQL
- MySQL

Either SQL database is suitable. For this document, we will use PostgreSQL:

```
sudo apt install postgresql
```

**Note**

Please take a look at *MySQL databases* and *PostgreSQL databases* for more details on these powerful databases.

Next we can install Bacula. The `bacula` package has the necessary dependencies and will pull in what is needed for our deployment scenario:

```
sudo apt install bacula
```

During the install process you will be asked to supply a password for the *database owner* of the *bacula database*. If left blank, a random password will be used.

### Configuring the Server

Several components were installed on this system by the `bacula` package. Each one has its own configuration file:

- Director: `/etc/bacula/bacula-dir.conf`
- Storage: `/etc/bacula/bacula-sd.conf`
- File: `/etc/bacula/bacula-fd.conf`
- Console: `/etc/bacula/bconsole.conf`

All these components need to eventually talk to each other, and the authentication is performed via passwords that were automatically generated at install time. These passwords are stored in the `/etc/bacula/common_default_passwords` file. If a new component is installed on this system, it can benefit from this file to automatically be ready to authenticate itself, but in general, after everything is installed and configured, this file isn't needed anymore.

### Director

The Bacula Director is the central component of the system. This is where we:

- Register the clients;
- Define the schedules;
- Define the file sets to be backed up;
- Define storage pools;
- Define the backup jobs;

Bacula configuration files are formatted based on **resources** composed of **directives** surrounded by curly "{}" braces. Each Bacula component has an individual file in the `/etc/bacula` directory.

The default installation of the several bacula components will create configuration files in `/etc/bacula/` with some choices and examples. That is a good reference, but it does not apply to all cases.

All the following sections involve the `/etc/bacula/bacula-dir.conf` Director configuration file, unless stated otherwise.

### The `Director` resource

This block defines the attributes of the Director service:

```
Director {
  Name = bacula-server-dir
  DIRport = 9101
  QueryFile = "/etc/bacula/scripts/query.sql"
  WorkingDirectory = "/var/lib/bacula"
  PidDirectory = "/run/bacula"
  Maximum Concurrent Jobs = 20
  Password = "<randomly generated>"
  Messages = Daemon
  #DirAddress = 127.0.0.1
}
```

What you should inspect and change:

- `Name`: it's common to use the format `$hostname-dir` for the director. For example, if the hostname is "`bacula-server`", the name here would be `bacula-server-dir`. By sticking to this pattern, less changes will have to be made to the config, as this is what the default installation already assumes.
- `DirAddress`: by default this is set to the localhost address. In order to be able to perform remote backups, though, the director needs to be accessible on the network. To do that, simply remove or comment this parameter: in that case, the service will listen on all network interfaces available on the system.
- `Password`: a random password will have been created for this installation, so it doesn't need to be changed, unless you would rather pick a different one.

**Tip**

For more details about all the options of the `Director` resource, please check the upstream Director Resource documentation.

### The `FileSet` resource

Let's define what we want to backup. There will likely be multiple file sets defined in a production server, but as an example, here we will define a set for backing up the home directory:

```
FileSet {
  Name = "Home Set"
  Include {
    Options {
      signature = SHA256
      aclsupport = yes
      xattrsupport = yes
    }
    Options {
        wilddir = "/home/*/Downloads"
        wildfile = "*.iso"
```

(continues on next page)

(continued from previous page)

```
        exclude = yes
    }
    File = /home
  }
}
```

This example illustrates some interesting points, and shows the type of flexibility he have in defining File Sets:

- `Name`: This defines the name of this file set, and will be referenced by other configuration blocks.
- *Include Options*: the `Options` block can be used several times inside the `Include` block. Here we define that we want to include POSIX ACLs support, and extended attributes, and use the SHA256 hash algorithm, but perhaps more interesting is how we select which files to exclude from the set:
  - `wilddir`, `wildfile`: these parameters specify globbing expressions of directories and files to exclude from the set (because we also set `exclude = yes`). In this example, we are excluding potentially large files. Unfortunately there is no direct way to exclude files from a set based on their size, so we have to make some educated guesses using standard file extensions and directory locations.
- `File`: This parameter can be specified multiple times, and it's additive. In this example we have it only used once, to select the `/home` directory and its subdirectories, subject to the exclusions defined in the `Options` block.

**Tip**

For more details about all the options of the `FileSet` resource, please check the upstream FileSet Resource documentation.

### The `Client` resource

The `Client` resource is used to define a system to be backed up. That system will have at least the File Daemon installed, and will be contacted by the Director whenever there is a backup job for it.

The default installation will have defined this resource already, and it should be similar to the following:

```
# Client (File Services) to backup
Client {
  Name = bacula-server-fd
  Address = bacula-server.lxd  # use the real hostname instead of "localhost"
  FDPort = 9102
  Catalog = MyCatalog
  Password = "<randomly generated>"
  File Retention = 60 days
  Job Retention = 6 months
  AutoPrune = yes
}
```

Of note in this definition we have:

- `Name`: As with other similar resources, the default name uses the format `$hostname-fd` (where `fd` stands for *File Daemon*).
- `Address`: The default will be `localhost`, but we should be in the habit of using a real hostname, because in a more distributed installation, these hostnames will be sent to other services in other machines, and "localhost" will then be incorrect.
- `Password`: The password was automatically generated, and should be kept as is unless you want to use another one.
- `File Retention`, `Job Retention`: these should be adjusted according to your particular needs for each client. By default, the backup job named `BackupClient1` is configured to archive the Bacula Catalog. If you plan on using the server to back up more than one client you should change the name of this job to something more descriptive. To change the name, edit `/etc/bacula/bacula-dir.conf`:
- `AutoPrune`: This setting makes Bacula automatically apply the retention parameters at the end of a backup job. It is enabled by default.

**Tip**

For more details about all the options of the `Client` resource, please check the upstream Client Resource documentation.

### The `Pool` resource

A *Pool* in Bacula represents a collection of volumes. A *Volume* is a single physical tape, or a file on disk, and is where Bacula will write the backup data.

The default configuration file will have defined several *Pools* already. For this documentation, we are interested in the `File` pool:

```
Pool {
  Name = File
  Pool Type = Backup
  Recycle = yes                  # Bacula can automatically recycle Volumes
  AutoPrune = yes                # Prune expired volumes
  Volume Retention = 365 days    # one year
  Maximum Volume Bytes = 50G     # Limit Volume size
  Maximum Volumes = 100          # Limit number of Volumes in Pool
  Label Format = "Vol-"          # Auto label
}
```

We will use this pool to backup to a directory on the server (which will usually be the mount point for a big storage device). The pool resource has some definitions that affect how large the backups can become, so these have to be checked:

- `Name`: The name of the pool, which will be referenced in other resources.
- `Volume Retention`: For how long volumes are kept.
- `Maximum Volume Bytes`: What is the maximum size of each volume file.
- `Maximum Volumes`: How many volume files are we going to keep at most.

- `Label Format`: The prefix that each volume file will get. In this example, the files will be automatically named `Vol-0001`, `Vol-0002`, and so on.

With the values in the example above, we will be storing at most 50G * 100 = 5000GB in this pool.

> **Tip**
>
> For more details about all the options of the `Pool` resource, please check the upstream Pool Resource documentation.

### The `Storage` resource

The *Storage* resource in the bacula Director configuration file points at the system where the Storage component is running.

In our current setup, that's the same system where the Director is running, but we **MUST NOT** use `localhost` in the definition, because this configuration is also sent to the File component on other systems. In another system, `localhost` will mean itself, but the Storage daemon is not running over there.

This time we will have to change two configuration files: the Director one, and the Storage one. Let's begin by defining a `Storage` resource on `/etc/bacula/bacula-dir.conf`, the Director configuration file:

```
Storage {
  Name = FileBackup
  Address = bacula-server.lxd
  SDPort = 9103
  # For this password, use:
  # sudo grep ^SDPASSWD /etc/bacula/common_default_passwords
  Password = "<SDPASSWD value>"
  Device = FileBackup
  Media Type = File
}
```

Here is what we have defined with the block above:

- `Name`: The name of this Storage resource, which will be referenced in other places.
- `Address`: The name of the system where the Storage daemon is running. Again, never use `localhost` here, even if it's the same system where the Director is running.
- `Password`: The password that should be used when connecting to the Storage daemon. The installation of the packages will have generated a random password. It can be found either in the existing `Autochanger` definitions in `/etc/bacula/bacula-dir.conf`, or in `/etc/bacula/common_default_passwords` in the line for `SDPASSWD`, or in the Storage daemon configuration file `/etc/bacula/bacula-sd.conf`.
- `Device`: This must match an existing `Device` definition in the Storage daemon's configuration file (which will be covered next).
- `Media Type`: Likewise, this must also match the same `Media Type` defined in the Storage daemon's configuration file.

> **Tip**
>
> For more details about all the options of the `Pool` resource, please check the upstream Storage Resource documentation.

Next we need to edit the corresponding Storage daemon configuration in `/etc/bacula/bacula-sd.conf`.

First, remove or comment out the `SDAddress` configuration, so that the daemon will listen on all network interfaces it finds:

```
Storage {
  Name = bacula-server-sd
  SDPort = 9103
  WorkingDirectory = "/var/lib/bacula"
  Pid Directory = "/run/bacula"
  Plugin Directory = "/usr/lib/bacula"
  Maximum Concurrent Jobs = 20
  Encryption Command = "/etc/bacula/scripts/key-manager.py getkey"
  #SDAddress = 127.0.0.1
}
```

Important points for the config above:

- `Name`: It's standard for Bacula systems to suffix the name of the system where a component is running with the abbreviation of that component. In this case, the name of the system is `bacule-server`, and the component we are defining is the Storage Daemon, hence the `-sd` suffix.
- `SDAddress`: We need this daemon to listen on all interfaces so it's reachable from other systems, so we comment this line out and rely on the default which it to listen on all interfaces.

Next, let's define a `Device`, also in `/etc/bacula/bacula-sd.conf`:

```
Device {
    Name = FileBackup
    Media Type = File
    Archive Device = /storage/backups
    Random Access = yes
    Automatic Mount = yes
    Removable Media = no
    Always Open = no
    Label Media = yes
}
```

What we need to pay close attention to here is:

- `Name`: This has to match the name this device will be referred to in other services. In our case, it matches the name we are using in the `Device` entry of the `Storage` definition we added to the Director configuration file `/etc/bacula/bacula-dir.conf` earlier.
- `Media Type`: Likewise, this has to match the entry we used in the `Storage` definition in the Director.

- `Archive Device`: Since we are going to store backups as files, and not as tapes, the `Archive Device` configuration points to a directory. Here we are using `/storage/backups`, which can be the mount point of an external storage for example. This is the target directory of all backup jobs what will refer to this device of this storage server.
- `Label Media`: Since we are using files and not real tapes, we want the Storage daemon to actually name the files for us. This configuration option allows it to do so.

Lastly, the Storage component needs to be told about the Director. This is done with a `Director` resource in `/etc/bacula/bacula-sd.conf`. No changes should be needed here because we installed the Storage component on the same host as the Director, but it's best to check:

```
Director {
    Name = bacula-server-dir
    Password = "<randomly generated>"
}
```

These two options need to match the following:

- `Name`: This names which Director is allowed to use this Storage component, and therefore needs to match the `Name` defined in the `Director` resource in `/etc/bacula/bacula-dir.conf` on the Director system.
- `Password`: The password that the Director needs to use to authenticate against this Storage component. This needs to match the `Password` set in the `Storage` resource in `/etc/bacula/bacula-dir.conf` on the Directory system.

After making all changes to the `bacula-sd.conf` configuration file, restart the Storage Daemon:

```
sudo systemctl restart bacula-sd.service
```

### The `Job` resource

The `Job` resource is the basic unit in Bacula, and ties everything together:

- Who is being backed up (`Client`).
- What should be backed up (`FileSet`).
- Where should the data be stored (`Storage`, `Pool`), and where to record the job (`Catalog`)
- When thouls the job run (`Schedule`)

The default Director configuration file includes a default Job resource, and more Jobs can inherit from that.

Let's go over the Default Job resource first in `/etc/bacula/bacula-dir.conf` and change it a little bit:

```
JobDefs {
  Name = "DefaultJob"
  Type = Backup
  Level = Incremental
  Client = bacula-server-fd
  FileSet = "Home Set"
```

(continues on next page)

(continued from previous page)

```
  Schedule = "WeeklyCycle"
  Storage = FileBackup
  Messages = Standard
  Pool = File
  SpoolAttributes = yes
  Priority = 10
  Write Bootstrap = "/var/lib/bacula/%c.bsr"
}
```

This configuration is selecting some defaults:

- `Name`: The name of this Job.
- `Client`: To which client it applies. This must match an existing `Client {}` resource definition.
- `FileSet`: The name of the `FileSet` resource that defines the data to be backed up. We changed it to `Home Set` in this example.
- `Schedule`: The name of the `Schedule` resource that defines when this job should run.
- `Storage`: Which `Storage` resource this job should use. We changed it to `FileBackup` in this example.
- `Pool`: Which `Pool` resource this job should use.

We can now take advantage of this set of defaults, and define a new Job resource with minimal config:

```
Job {
    Name = "DirectorHomeBackup"
    JobDefs = "DefaultJob"
}
```

**Tip**

A job has many attributes that can only be specified once. This means that jobs are pretty much specific to what is being backed up, and from where, among other things. It therefore helps to come up with a naming convention.

The upstream documentation has a section about Naming Resources with some suggestions.

We also need a Job definition for the restore task. The default configuration file will have a definition for this already, but it needs to be changed:

```
Job {
    Name = "RestoreFiles"
    Type = Restore
    Client = bacula-server-fd
    Storage = FileBackup
    # The FileSet and Pool directives are not used by Restore Jobs
    # but must not be removed
```

(continues on next page)

(continued from previous page)

```
    FileSet = "Home Set"
    Pool = File
    Messages = Standard
    Where = /storage/restore
}
```

Important parameters defined above:

- `Name`: The name of this job.
- `Type`: This is a job that restores backups (`Restore`).
- `Client`: Where the files should be restored to. This can be overridden when the job is invoked.
- `Storage`: The storage from where the backup should be restored. We changed it to `FileBackup` in this example.
- `FileSet` and `Pool`: These are not used, but must be present and point to valid resources. We changed `FileSet` to `Home Set` in this example.
- `Where`: The path where the restored files should be placed. This can also be overridden when the job is invoked. We changed it to `/storage/restore` in this example.

**Tip**

For more details about all the options of the `Job` resource, please check the upstream Job Resource documentation.

### Storage daemon

There isn't much more to configure for the Storage daemon after the Director configuration steps done earlier, but we still need to create the directories for the backup and restore jobs:

```
sudo mkdir -m 0700 /storage /storage/backups /storage/restore
sudo chown bacula: -R /storage
```

This will allow bacula, and only bacula, to read and write to the storage path. You can, of course, adjust the permissions and ownership to something that suits your deployment. Just be mindful that the bacula user needs to be able to create and remove files from the `/storage/backups` and `/storage/restore` paths, and that regular users should not be allowed to read those.

**Tip**

For more details about the Storage daemon configuration options, please check the upstream Storage Daemon documentation.

### File daemon

The File daemon configuration is located in the `/etc/bacula/bacula-fd.conf` file, and the only remaining task is to make sure it listens on the network. To be fair, in this particular deployment layout, this is not strictly needed, as both the Director and Storage daemons are located on the same system, but making this change allows for those components to be split off to different systems should that need arise.

To make this change, we are going to remove or comment out the `FDAddress` option in the `FileDaemon` resource in `/etc/bacula/bacula-fd.conf` file:

```
FileDaemon {
  Name = bacula-server-fd
  FDport = 9102
  WorkingDirectory = /var/lib/bacula
  Pid Directory = /run/bacula
  Maximum Concurrent Jobs = 20
  Plugin Directory = /usr/lib/bacula
  #FDAddress = 127.0.0.1
}
```

After making the change and saving the file, restart the File daemon service:

```
sudo systemctl restart bacula-fd.service
```

> **Tip**
>
> For more details about the File daemon configuration, please check the upstream File Daemon documentation.

### Console

There is no further configuration to be done for the Console at this time. The defaults selected and adjusted by the package install are sufficient. The configuration file is `/etc/bacula/bconsole.conf`, and more details are available in the upstream Console Configuration documentation.

The Console can be used to query the Director about jobs, but to use the Console with a *non-root* user, the user needs to be in the **Bacula group**. To add a user to the Bacula group, run the following command from a terminal:

```
sudo adduser <username> bacula
```

Replace `<username>` with the actual username. Also, if you are adding the current user to the group you should log out and back in for the new permissions to take effect.

> **Warning**
>
> Be mindful of who is added to the `bacula` group: members of this group are able to read all the data that is being backed up!

### Cleaning up

We have added new resources to some Bacula components, and changed some existing ones. There are also resources we didn't touch, but they will show up in the console or logs. Optionally, we can remove them to cleanup our config files.

In `/etc/bacula/bacula-dir.conf`:

- all the `Autochanger` resources can be removed, since they are not referred to by any other resource.

In `/etc/bacula/bacula-sd.conf`:

- The `Autochanger` resources can be removed, as it's not being used.
- The `FileChgr1-Dev1`, `FileChgr1-Dev2`, `FileChgr2-Dev1`, and `FileChgr2-Dev2` Devices, referred to by the Autochangers above, should then also be removed.

We made many changes to a few configuration files, so let's restart all the related services:

```
sudo systemctl restart bacula-dir.service bacula-fd.service bacula-sd.service
```

### Running a backup job

We now have everything in place to run our first backup job.

On the Bacula Director system, run the `bconsole` command as root to enter the Bacula Console:

```
sudo bconsole
```

The command will connect to the the local Director, and open up an interactive prompt:

```
Connecting to Director localhost:9101
1000 OK: 10002 bacula-server-dir Version: 15.0.3 (25 March 2025)
Enter a period to cancel a command.
*
```

You can type `help` for a full list of all the available commands, and `help <command>` for more detailed information about the specific `<command>`.

For example, to obtain help text about the `run` command, type `help run` to obtain the following output:

```
  Command       Description
  =======       ===========
  run           Run a job

Arguments:
  job=<job-name> client=<client-name>
  fileset=<FileSet-name> level=<level-keyword>
  storage=<storage-name> where=<directory-prefix>
  when=<universal-time-specification> pool=<pool-name>
  nextpool=<next-pool-name> comment=<text> accurate=<bool> spooldata=<bool> yes

When at a prompt, entering a period cancels the command.
```

Let's interactively run a backup job. The output below will show the `run` command and all the replies that were typed in response to the console prompts:

```
*run
Using Catalog "MyCatalog"
A job name must be specified.
The defined Job resources are:
     1: HomeBackup
     2: BackupCatalog
     3: RestoreFiles
Select Job resource (1-3): 1
Run Backup job
JobName:  HomeBackup
Level:    Incremental
Client:   bacula-server-fd
FileSet:  Home Set
Pool:     File (From Job resource)
Storage:  FileBackup (From Job resource)
When:     2025-10-20 20:21:03
Priority: 10
OK to run? (Yes/mod/no): yes
Job queued. JobId=7
```

Let's unpack this:

- `run`: This is the command. Since no parameters were given, Bacula will ask for what's missing.
- Which job should be run: `HomeBackup` is the job we defined in this how-to, and we select it by its index number.
- Summary: at the end, we are given a summary of the job. Here we can still change values via the `mod` reply, but for now let's just accept those values and reply `yes`.
- JobId: the job is accepted, and we are given an ID. In this case, it was "7".

To check the result of a job, there are several methods:

- Messages: right after scheduling the job, it's likely something will be logged. You can run the `messages` command, and it will show the latest unread messages (and also mark them as read, so you can only benefit from this once).
- The `list jobs` command, to list all jobs, or, more specifically, `list jobid=7` to list a particular job.
- Inspect the full log of that particular job, via the `list joblog jobid=<N>` command.
- Server log: you can inspect the server log at `/var/log/bacula/bacula.log`.

For example, if we run `list jobid=7`, this is the output:

```
+-------+-----------+---------------------+------+-------+----------+----------+-
----------+
| jobid | name      | starttime           | type | level | jobfiles | jobbytes |
jobstatus |
+-------+-----------+---------------------+------+-------+----------+----------+-
```

(continues on next page)

(continued from previous page)

```
----------+
|     7 | HomeBackup | 2025-10-20 20:21:11 | B    | I     |        0 |        0 |
T         |
+-------+------------+---------------------+------+-------+----------+----------+-
----------+
```

That tells us some details about this job, in particular that it finished correctly (the `T` code).

> **Tip**
>
> For a list of status and error codes, check the upstream Job status and Error codes tables.

To see the full log of this specific job, we can use the `list joblog jobid=7` command. This is quite detailed, and the output below is truncated for brevity:

```
+-------------------------------------------------------------------------------
------------------+
| logtext
                  |
+-------------------------------------------------------------------------------
------------------+
| bacula-server-dir JobId 7: Start Backup JobId 7, Job=HomeBackup.2025-10-20_20.
21.08_03              |
| bacula-server-dir JobId 7: Connected to Storage "FileBackup" at bacula-server.
lxd:9103 with TLS     |
| bacula-server-dir JobId 7: Using Device "FileBackup" to write.
                  |
...
  Build OS:               x86_64-pc-linux-gnu ubuntu 25.10
  JobId:                  7
  Job:                    HomeBackup.2025-10-20_20.21.08_03
  Backup Level:           Incremental, since=2025-10-20 18:11:00
  Client:                 "bacula-server-fd" 15.0.3 (25Mar25) x86_64-pc-linux-gnu,
ubuntu,25.10
  FileSet:                "Home Set" 2025-10-20 16:27:31
  Pool:                   "File" (From Job resource)
  Catalog:                "MyCatalog" (From Client resource)
  Storage:                "FileBackup" (From Job resource)
...
  Non-fatal FD errors:    0
  SD Errors:              0
  FD termination status:  OK
  SD termination status:  OK
  Termination:            Backup OK
                  |
...
```

If we inspect the backup target location on the Storage server (which in this deployment is the same as the Director), we can see that a volume file was created:

```
-rw-r----- 1 bacula tape 345K Oct 20 20:21 /storage/backups/Vol-0001
```

### Restoring a backup

So what is it that was backed up? This job used the `Home Set`, so we expect to see files from the `/home` directory. To see what are the contents of that backup job, we can use the `restore` command (the `RestoreFiles` job should never be executed directly). Below is the output of an interactive `restore` session where we selected the option "Select the most recent backup for a client":

```
First you select one or more JobIds that contain files
to be restored. You will be presented several methods
of specifying the JobIds. Then you will be allowed to
select which files from those JobIds are to be restored.

To select the JobIds, you have the following choices:
     1: List last 20 Jobs run
...
     5: Select the most recent backup for a client
...
Select item:  (1-14): 5
Automatically selected Client: bacula-server-fd
Automatically selected FileSet: Home Set
+-------+-------+----------+----------+---------------------+------------+
| jobid | level | jobfiles | jobbytes | starttime           | volumename |
+-------+-------+----------+----------+---------------------+------------+
|     4 | F     |      266 |  312,756 | 2025-10-20 17:34:43 | Vol-0001   |
|     5 | I     |        3 |       32 | 2025-10-20 18:11:00 | Vol-0001   |
+-------+-------+----------+----------+---------------------+------------+
You have selected the following JobIds: 4,5
...
You are now entering file selection mode where you add (mark) and
remove (unmark) files to be restored. No files are initially added, unless
you used the "all" keyword on the command line.
Enter "done" to leave this mode.

cwd is: /
$
```

Here we can navigate the filesystem and inspect which files are part of the backup:

```
$ dir
drwxr-xr-x   1 root     root              12  2025-10-20 14:03:44  /home/
$ cd home/ubuntu
cwd is: /home/ubuntu/
$ dir
-rw-------   1 ubuntu   ubuntu            32  2025-10-20 18:10:51  /home/ubuntu/.
bash_history
-rw-r--r--   1 ubuntu   ubuntu           220  2025-10-20 14:03:50  /home/ubuntu/.
bash_logout
```

(continues on next page)

(continued from previous page)

```
-rw-r--r--   1 ubuntu   ubuntu          3830  2025-10-20 14:03:50  /home/ubuntu/.
bashrc
-rw-r--r--   1 ubuntu   ubuntu           807  2025-10-20 14:03:44  /home/ubuntu/.
profile
drwx------   1 ubuntu   ubuntu            30  2025-10-20 14:03:45  /home/ubuntu/.
ssh/
-rw-rw-r--   1 ubuntu   ubuntu             0  2025-10-20 18:10:50  /home/ubuntu/
this-is-on-the-server.txt
```

To restore a file, we use the `mark` command on it. For example, let's restore `/home/ubuntu/.tmux.conf`:

```
$ mark .tmux.conf
1 file marked.
$ done
Bootstrap records written to /var/lib/bacula/bacula-server-dir.restore.1.bsr

The Job will require the following (*=>InChanger):
   Volume(s)                 Storage(s)                SD Device(s)
===========================================================================

    Vol-0001                  FileBackup                FileBackup

Volumes marked with "*" are in the Autochanger.


1 file selected to be restored.

Run Restore job
JobName:         RestoreFiles
Bootstrap:       /var/lib/bacula/bacula-server-dir.restore.1.bsr
Where:           /storage/restore
Replace:         Always
FileSet:         Home Set
Backup Client:   bacula-server-fd
Restore Client:  bacula-server-fd
Storage:         FileBackup
When:            2025-10-20 21:02:37
Catalog:         MyCatalog
Priority:        10
Plugin Options:  *None*
OK to run? (Yes/mod/no):
```

Now we have some choices. Notice how the `RestoreFiles` job was automatically selected. That's the only job of the type `Restore` that we defined in the Director configuration earlier. It has certain default values, and we can either accept those (by replying `yes`), or modify them (by replying `mod`).

If we accept these default, the marked files will be restored to the `/storage/restore` path on the `bacula-server-fd` system:

```
OK to run? (Yes/mod/no): yes
Job queued. JobId=8
*
```

And indeed, if we inspect that location, we see the file that we marked for restoration:

```
-rw-r--r-- 1 ubuntu ubuntu 2.4K Oct 20 14:03 /storage/restore/home/ubuntu/.tmux.
conf
```

If we wanted to restore it to its original place, for example, if the user mistakenly deleted it and wanted it back, we would select the `mod` option to change where the file should be placed:

```
OK to run? (Yes/mod/no): mod
Parameters to modify:
     1: Level
     2: Storage
     3: Job
     4: FileSet
     5: Restore Client
     6: When
     7: Priority
     8: Bootstrap
     9: Where
    10: File Relocation
    11: Replace
    12: JobId
    13: Plugin Options
Select parameter to modify (1-13): 9
Please enter the full path prefix for restore (/ for none): /
Run Restore job
JobName:         RestoreFiles
Bootstrap:       /var/lib/bacula/bacula-server-dir.restore.2.bsr
Where:
Replace:         Always
FileSet:         Home Set
Backup Client:   bacula-server-fd
Restore Client:  bacula-server-fd
Storage:         FileBackup
When:            2025-10-20 21:11:55
Catalog:         MyCatalog
Priority:        10
Plugin Options:  *None*
OK to run? (Yes/mod/no): yes
Job queued. JobId=9
```

By giving a restoration prefix of `/`, we are essentially asking to restore the file at its original full path.

### Adding a client

If we want to start backing up a new system, we need to install the File Daemon on that system and include it in the Bacula Director. In this example, the new system we are adding is called `workstation1`.

First, on the system that we want to add, let's install the client portion of Bacula, which is the File Daemon component:

```
sudo apt install bacula-fd
```

Next, update the Director resource in `/etc/bacula/bacula-fd.conf` to point at the existing Director we have already deployed:

```
Director {
  Name = bacula-server-dir # same as Director's Name on the Director server
  Password = "<randomly generated>"
}
```

Notes:

- `Name`: This has to be the same name set on the Director's `/etc/bacula/bacula-dir.conf` file, in the `Director` resource over there. It's not the hostname.
- `Password`: The password was randomly generated when the `bacula-fd` package was installed. This password has to match the password in the new `Client` resource that we will add to the Director next, so keep note of it.

Also in `/etc/bacula/bacula-fd.conf`, we have to remove or comment out the `FDAddress` parameter in the `FileDaemon` resource, so that this service will listen on all available network interfaces, and not just localhost:

```
FileDaemon {
  Name = workstation1-fd
  FDport = 9102           # where we listen for the director
  WorkingDirectory = /var/lib/bacula
  Pid Directory = /run/bacula
  Maximum Concurrent Jobs = 20
  Plugin Directory = /usr/lib/bacula
  #FDAddress = 127.0.0.1  # default is to listen on all interfaces
}
```

And finally, in the same file, update the `Messages` resource and update the Director name in there as well:

```
Messages {
    Name = Standard
    director = bacula-server-dir = all, !skipped, !restored, !verified, !saved
}
```

With these changes done, restart the File Daemon:

```
sudo systemctl restart bacula-fd.service
```

Now we switch to the Director system, where we have to let it know about this new Client that we just provisioned.

In `/etc/bacula/bacula-dir.conf`, add a new `Client` resource:

```
Client {
    Name = workstation1-fd
    Address = workstation1.lxd
    FDPort = 9102
    Catalog = MyCatalog
    Password = "<to be filled in>" # password from bacula-fd.conf on workstation1-
fd
    File Retention = 60 days
    Job Retention = 6 months
    AutoPrune = yes
}
```

Notes:

- `Name`: The name has to match the name defined in the `FileDaemon` resource from `/etc/bacula/bacula-fd.conf` of the system we just added.
- `Password`: The password has to be the same as the one defined in the `Director` resource from `/etc/bacula/bacula-fd.conf` of that system.
- `Address`: The hostname or IP of the system we added.

This makes the Director know how to reach the new client.

Now we have to define a new job to backup files from this new client. Again on `/etc/bacula/bacula-dir.conf` on the Director, let's add a new `Job` resource:

```
Job {
    Name = "BackupWorkstation"
    JobDefs = "DefaultJob"
    Client = workstation1-fd
}
```

This job inherits all parameters from the `DefaultJob`, and just overrides the client.

With this done, we can restart the Director:

```
sudo systemctl restart bacula-dir.service
```

If we now enter the Bacula console, we should be able to list the new client, and run its new backup job:

```
*list clients
Automatically selected Catalog: MyCatalog
Using Catalog "MyCatalog"
+----------+-----------------+--------------+--------------+
| clientid | name            | fileretention | jobretention |
+----------+-----------------+--------------+--------------+
|        1 | bacula-server-fd |     5,184,000 |   15,552,000 |
|        2 | workstation1-fd  |     5,184,000 |   15,552,000 |
+----------+-----------------+--------------+--------------+
```

Let's run the new `BackupWorkstation` job:

```
*run
Using Catalog "MyCatalog"
A job name must be specified.
The defined Job resources are:
     1: HomeBackup
     2: BackupWorkstation
     3: BackupCatalog
     4: RestoreFiles
Select Job resource (1-4): 2
Run Backup job
JobName:  BackupWorkstation
Level:    Incremental
Client:   workstation1-fd
FileSet:  Home Set
Pool:     File (From Job resource)
Storage:  FileBackup (From Job resource)
When:     2025-10-21 21:02:48
Priority: 10
OK to run? (Yes/mod/no): yes
Job queued. JobId=14
You have messages.
```

And for a quick check of the contents (for testing, there was a file called `this-is-workstation1.txt` in `/home/ubuntu` on that system):

```
*restore
...
     5: Select the most recent backup for a client
...
Select item:  (1-14): 5
Defined Clients:
     1: bacula-server-fd
     2: workstation1-fd
Select the Client (1-2): 2
...
$ cd home/ubuntu
cwd is: /home/ubuntu/
$ dir this*
-rw-rw-r--   1 ubuntu   ubuntu            0  2025-10-21 21:02:08  /home/ubuntu/
this-is-workstation1.txt
```

### Further reading

- For more Bacula configuration options, refer to the Bacula documentation.
- The Bacula home page contains the latest Bacula news and developments.

### How to install and configure rsnapshot

rsnapshot is an rsync-based *filesystem* snapshot utility. It can take incremental backups of local and remote filesystems for any number of machines. rsnapshot makes extensive use of hard links, so disk space is only used when absolutely necessary. It leverages the power of rsync to create scheduled, incremental backups.

### Install rsnapshot

To install `rsnapshot` open a terminal shell and run:

```
sudo apt-get install rsnapshot
```

If you want to backup a remote filesystem, the rsnapshot server needs to be able to access the target machine over SSH without password. For more information on how to enable this please see *OpenSSH documentation*. If the backup target is a local filesystem there is no need to set up OpenSSH.

### Configure rsnapshot

The `rsnapshot` configuration resides in `/etc/rsnapshot.conf`. Below you can find some of the options available there.

The root directory where all snapshots will be stored is found at:

```
snapshot_root       /var/cache/rsnapshot/
```

### Number of backups to keep

Since `rsnapshot` uses incremental backups, we can afford to keep older backups for a while before removing them. You set these up under the `BACKUP LEVELS / INTERVALS` section. You can tell `rsnapshot` to retain a specific number of backups of each kind of interval.

```
retain  daily   6
retain  weekly    7
retain  monthly   4
```

In this example we will keep 6 snapshots of our daily strategy, 7 snapshots of our weekly strategy, and 4 snapshots of our monthly strategy. These data will guide the rotation made by `rsnapshot`.

### Remote machine access

If you are accessing a remote machine over SSH and the port to bind is not the default (port 22), you need to set the following variable with the port number:

```
ssh_args        -p 22222
```

### What to backup

Now the most important part; you need to decide what you would like to backup.

If you are backing up locally to the same machine, this is as easy as specifying the directories that you want to save and following it with `localhost/` which will be a sub-directory in the `snapshot_root` that you set up earlier.

```
backup  /home/          localhost/
backup  /etc/           localhost/
backup  /usr/local/     localhost/
```

If you are backing up a remote machine you just need to tell `rsnapshot` where the server is and which directories you would like to back up.

```
backup root@example.com:/home/ example.com/    +rsync_long_args=--bwlimit=16,
exclude=core
backup root@example.com:/etc/  example.com/    exclude=mtab,exclude=core
```

As you can see, you can pass extra rsync parameters (the `+` appends the parameter to the default list – if you remove the `+` sign you override it) and also exclude directories.

You can check the comments in `/etc/rsnapshot.conf` and the `rsnapshot(1)` manual page for more options.

### Test configuration

After modifying the configuration file, it is good practice to check if the syntax is OK:

```
sudo rsnapshot configtest
```

You can also test your backup levels with the following command:

```
sudo rsnapshot -t daily
```

If you are happy with the output and want to see it in action you can run:

```
sudo rsnapshot daily
```

### Scheduling backups

With `rsnapshot` working correctly with the current configuration, the only thing left to do is schedule it to run at certain intervals. We will use cron to make this happen since `rsnapshot` includes a default cron file in `/etc/cron.d/rsnapshot`. If you open this file there are some entries commented out as reference.

```
0 4  * * *          root    /usr/bin/rsnapshot daily
0 3  * * 1          root    /usr/bin/rsnapshot weekly
0 2  1 * *          root    /usr/bin/rsnapshot monthly
```

The settings above added to `/etc/cron.d/rsnapshot` run:

- The **daily snapshot** everyday at 4:00 am
- The **weekly snapshot** every Monday at 3:00 am

- The **monthly snapshot** on the first of every month at 2:00 am

For more information on how to schedule a backup using cron please take a look at the `Executing with cron` section in *Backups - Shell Scripts*.

### Further reading

- rsnapshot offical web page
- `rsnapshot(1)` manual page
- `rsync(1)` manual page

### Shell scripts

If you are looking for full flexibility and customization, another option is to use shell scripts.

### How to back up using shell scripts

In general, a shell script configures which directories to backup, and passes those directories as arguments to the `tar` utility, which creates an archive file. The archive file can then be moved or copied to another location. The archive can also be created on a remote file system such as a *Network File System (NFS)* mount.

The `tar` utility creates one archive file out of many files or directories. `tar` can also filter the files through compression utilities, thus reducing the size of the archive file.

In this guide, we will walk through how to use a shell script for backing up files, and how to restore files from the archive we create.

### The shell script

The following shell script uses `tar` to create an archive file on a remotely mounted NFS file system. The archive filename is determined using additional command line utilities. For more details about the script, check out the `basic backup shell script <https://discourse.ubuntu.com/t/basic-backup-shell-script/36419>_`.

```
#!/bin/bash
###################################
#
# Backup to NFS mount script.
#
###################################

# What to backup.
backup_files="/home /var/spool/mail /etc /root /boot /opt"

# Where to backup to.
dest="/mnt/backup"

# Create archive filename.
day=$(date +%A)
hostname=$(hostname -s)
```

(continues on next page)

(continued from previous page)

```
archive_file="$hostname-$day.tgz"

# Print start status message.
echo "Backing up $backup_files to $dest/$archive_file"
date
echo

# Backup the files using tar.
tar czf $dest/$archive_file $backup_files

# Print end status message.
echo
echo "Backup finished"
date

# Long listing of files in $dest to check file sizes.
ls -lh $dest
```

### Running the script

#### Run from a terminal

The simplest way to use the above backup script is to copy and paste the contents into a file (called `backup.sh`, for example). The file must be made executable:

```
chmod u+x backup.sh
```

Then from a terminal prompt, run the following command:

```
sudo ./backup.sh
```

This is a great way to test the script to make sure everything works as expected.

#### Run with `cron`

The `cron` utility can be used to automate use of the script. The `cron` daemon allows scripts, or commands, to be run at a specified time and date.

`cron` is configured through entries in a `crontab` file. `crontab` files are separated into fields:

```
# m h dom mon dow   command
```

Where:

- `m`: The minute the command executes on, between 0 and 59.
- `h`: The hour the command executes on, between 0 and 23.
- `dom`: The day of the month the command executes on.
- `mon`: The month the command executes on, between 1 and 12.
- `dow`: The day of the week the command executes on, between 0 and 7. Sunday may be specified by using 0 or 7, both values are valid.

- `command`: The command to run.

To add or change entries in a `crontab` file the `crontab -e` command should be used. Also note the contents of a `crontab` file can be viewed using the `crontab -l` command.

To run the `backup.sh` script listed above using `cron`, enter the following from a terminal prompt:

```
sudo crontab -e
```

**Note**

Using `sudo` with the `crontab -e` command edits the *root* user's `crontab`. This is necessary if you are backing up directories only the root user has access to.

As an example, if we add the following entry to the `crontab` file:

```
# m h dom mon dow   command
0 0 * * * bash /usr/local/bin/backup.sh
```

The `backup.sh` script would be run every day at 12:00 pm.

**Note**

The `backup.sh` script will need to be copied to the `/usr/local/bin/` directory in order for this entry to run properly. The script can reside anywhere on the file system, simply change the script path appropriately.

### Restoring from the archive

Once an archive has been created, it is important to test the archive. The archive can be tested by listing the files it contains, but the best test is to **restore** a file from the archive.

- To see a listing of the archive contents, run the following command from a terminal:

  ```
  tar -tzvf /mnt/backup/host-Monday.tgz
  ```

- To restore a file from the archive back to a different directory, enter:

  ```
  tar -xzvf /mnt/backup/host-Monday.tgz -C /tmp etc/hosts
  ```

  The `-C` option to `tar` redirects the extracted files to the specified directory. The above example will extract the `/etc/hosts` file to `/tmp/etc/hosts`. `tar` recreates the directory structure that it contains. Also, notice the leading "/" is left off the path of the file to restore.

- To restore all files in the archive enter the following:

  ```
  cd /
  sudo tar -xzvf /mnt/backup/host-Monday.tgz
  ```

> **Note**
>
> This will overwrite the files currently on the file system.

### Further reading

- For more information on shell scripting see the Advanced Bash-Scripting Guide
- The Cron How-to Wiki Page contains details on advanced cron options.
- See the GNU tar Manual for more tar options.
- The Wikipedia Backup Rotation Scheme article contains information on other backup rotation schemes.
- The shell script uses tar to create the archive, but there many other command line utilities that can be used. For example:
  - `cpio`: used to copy files to and from archives.
  - `dd`: part of the coreutils package. A low level utility that can copy data from one format to another.
  - `rsnapshot`: a file system snapshot utility used to create copies of an entire file system. Also check the *Tools - rsnapshot* for some information.
  - `rsync(1)`: a flexible utility used to create incremental copies of files.

### Version control

- *etckeeper* stores the contents of `/etc` in a Version Control System (VCS) repository
- *Install gitolite* for a traditional source control management server for git, including multiple users and access rights management

### etckeeper

etckeeper allows the contents of `/etc` to be stored in a Version Control System (VCS) repository. It integrates with APT and automatically commits changes to `/etc` when packages are installed or upgraded.

Placing `/etc` under version control is considered an industry best practice, and the goal of etckeeper is to make this process as painless as possible.

### Install etckeeper

Install etckeeper by entering the following in a terminal:

```
sudo apt install etckeeper
```

### Initialise etckeeper

The main configuration file, `/etc/etckeeper/etckeeper.conf`, is fairly simple. The main option defines which VCS to use, and by default etckeeper is configured to use git.

The repository is automatically initialised (and committed for the first time) during package installation. It is possible to undo this by entering the following command:

```
sudo etckeeper uninit
```

### Configure autocommit frequency

By default, etckeeper will commit uncommitted changes made to `/etc` on a daily basis. This can be disabled using the `AVOID_DAILY_AUTOCOMMITS` configuration option.

It will also automatically commit changes before and after package installation. For a more precise tracking of changes, it is recommended to commit your changes manually, together with a commit message, using:

```
sudo etckeeper commit "Reason for configuration change"
```

The `vcs` etckeeper command provides access to any subcommand of the VCS that etckeeper is configured to run. It will be run in `/etc`. For example, in the case of git:

```
sudo etckeeper vcs log /etc/passwd
```

To demonstrate the integration with the package management system (APT), install `postfix`:

```
sudo apt install postfix
```

When the installation is finished, all the `postfix` configuration files should be committed to the repository:

```
[master 5a16a0d] committing changes in /etc made by "apt install postfix"
 Author: Your Name <xyz@example.com>
 36 files changed, 2987 insertions(+), 4 deletions(-)
 create mode 100755 init.d/postfix
 create mode 100644 insserv.conf.d/postfix
 create mode 100755 network/if-down.d/postfix
 create mode 100755 network/if-up.d/postfix
 create mode 100644 postfix/dynamicmaps.cf
 create mode 100644 postfix/main.cf
 create mode 100644 postfix/main.cf.proto
 create mode 120000 postfix/makedefs.out
 create mode 100644 postfix/master.cf
 create mode 100644 postfix/master.cf.proto
 create mode 100755 postfix/post-install
 create mode 100644 postfix/postfix-files
 create mode 100755 postfix/postfix-script
 create mode 100755 ppp/ip-down.d/postfix
 create mode 100755 ppp/ip-up.d/postfix
 create mode 120000 rc0.d/K01postfix
 create mode 120000 rc1.d/K01postfix
 create mode 120000 rc2.d/S01postfix
 create mode 120000 rc3.d/S01postfix
 create mode 120000 rc4.d/S01postfix
 create mode 120000 rc5.d/S01postfix
```

(continues on next page)

(continued from previous page)

```
    create mode 120000 rc6.d/K01postfix
    create mode 100755 resolvconf/update-libc.d/postfix
    create mode 100644 rsyslog.d/postfix.conf
    create mode 120000 systemd/system/multi-user.target.wants/postfix.service
    create mode 100644 ufw/applications.d/postfix
```

For an example of how `etckeeper` tracks manual changes, add new a host to `/etc/hosts`. Using git you can see which files have been modified:

```
sudo etckeeper vcs status
```

and how:

```
sudo etckeeper vcs diff
```

If you are happy with the changes you can now commit them:

```
sudo etckeeper commit "added new host"
```

### Resources

- See the etckeeper site for more details on using etckeeper.
- For documentation on the git VCS tool see the Git website.

### How to set up gitolite

*Gitolite* Gitolite allows you to setup git hosting on a central server, with fine-grained access control and many more powerful features.

You can use your served repositories as `git remote` in the form of `git@yourserver:some/repo/path`.

Gitolite stores "bare git" repositories at a location of your choice, usually `/home/git`. It has its independent user realm, each user is created by assigning their *SSH-key*. The repos itself are owned by one system user of your choice, usually `git`.

### Install a gitolite server

Gitolite can be installed with the following command.

The install automation will ask for a path or content of your `admin` ssh key. For a better understanding of your setup we recommend to leave the prompt empty for full control of your setup (so you can use `git` as the username, and customize the storage path).

```
sudo apt install gitolite3
```

### Configure gitolite

Gitolite stores its configuration in a git repository (called `gitolite-admin`), so there's no configuration in `/etc`. This configuration repository manages all other git repos, users and their permissions.

Create a `git` user for gitolite to use for the service (you can adjust the git repo storage path as the `--home` directory):

```
sudo useradd --system --home /home/git git
```

To access the config repository, we now add the administrator's public *SSH-key*. If you have not yet configured an SSH key, refer to the section on *SSH keys in our OpenSSH guide*.

We copy it to `/tmp` so our `git` user can read the file to import it. Please adjust the path to the desired admin user's *SSH-key* (and algorithm, like `id_rsa.pub`).

```
cp ~/.ssh/id_ed25519.pub /tmp/admin.pub
```

As the `git` user, let's import the administrator's key into gitolite (it will get the `admin` username due to that key's filename).

```
sudo -i -u git gitolite setup -pk /tmp/admin.pub
```

What this creates:

- the management repo in `~git/repositories/gitolite-admin.git`
- a global config in `~git/.gitolite.rc`
- `~git/projects.list` as repo overview
- `~git/.ssh/authorized_keys` with `command=` to force gitolite over `ssh`
  - later it will contain the `ssh` public key for each user you configured

To try if the setup worked, try `ssh` as the user owning the admin key we just added, so see the *gitolite repo overview*:

```
ssh git@yourserver
```

```
hello admin, this is git@your-gitolite-server running gitolite3

 R W        gitolite-admin
 R W        testing
```

### Managing gitolite users and repositories

To configure gitolite users, repositories and permissions, clone the configuration repository. `$yourserver` can be an ip-address, hostname, or just `localhost` for your current machine.

```
git clone git@$yourserver:gitolite-admin.git
```

To apply configuration change, commit them in the repo and **push the changes** back to the server with:

```
git commit -a
git push origin master
```

The `gitolite-admin` contains two subdirectories: `keydir` (which contains the list of users' public SSH keys) and `conf` (which contains configuration files). To **add a gitolite user** (it's virtual - not a system username), obtain their SSH public key (from `~user/.ssh/id_<name>.pub`) and

add it to the `keydir` directory as `<desired-username>.pub`. To **delete a gitolite user**, you only need to delete their public key files. To manage repositories and groups in `conf/gitolite.conf`, specify the the list of repositories followed by some access rules.

Have an example:

```
# gitolite config
# users are created by their public key in keydir/$username.pub

# group creation
@bestproject          = name1 name2
@projectwatchers      = name3 @bestproject

# this repo itself
repo    gitolite-admin
        RW+     =   admin
        R       =   alice

# a repo with access to anybody
repo    testing
        RW+     = @all

# a repo with special privileges, to tags and branches
repo    some/awesome/project
        RW                      =   alice @bestproject
        RW+                     =   bob
        RW+   dev/              =   @bestproject
        R                       =   @projectwatchers carol
# bestproject members and alice can push code (but not force-push)
# bestproject members can force-push branches starting with dev/
# bob can forcepush anything
# projectwatchers and carol have readonly access
```

For more advanced permission configuration (restricting tags, branches, …), please see the examples in the upstream documentation page 1 and page 2.

### Using your server

Now you can use your newly set up gitolite server as a regular `git remote`.

Once a user is created and has permissions, they can access the repositories.

As a fresh clone:

```
git clone git@$server:some/awesome/project.git
```

Or as a remote to an existing repository:

```
git remote add gitolite git@$server:some/awesome/project.git
```

#### Further reading

- Gitolite's code repository provides access to source code
- Gitolite's documentation includes more detailed configuration guides and a "fool-proof setup", with how-tos for common tasks
- Gitolite's maintainer has written a book, Gitolite Essentials, for more in-depth information about the software
- General information about `git` itself can be found at the Git homepage

#### See also

- Explanation: *Introduction to backups*

#### See also

- Explanation: *Data and storage*

*The data and storage section* covers the following:

- **Managing data** in the OpenLDAP and databases topics
- **Storage and backups**, including partitioning (with LVM), backup utilities, and version control

# 2.6. Mail services

Our *Mail services* section shows you how to set up:

- **Mail User Agents** (Thunderbird)
- **Mail Transfer Agents** (Postfix and Exim4)
- **Mail Delivery Agents** (Dovecot)

## 2.6.1. Mail services

Sending email from one person to another over a network or the Internet requires:

1. The sender's email client (**Mail User Agent**) sends the message.
2. One or more **Mail Transfer Agents** (MTA) to transfer the message.
3. The final MTA sends the message to a **Mail Delivery Agent** (MDA) for delivery to the recipient's inbox.
4. Finally, the recipient's email client retrieves the message, usually via a **POP3** or **IMAP** server.

These systems must all be configured correctly to successfully deliver a message.

#### Mail User Agent

Thunderbird is the default Mail User Agent (email client) used by Ubuntu. It comes pre-installed on all Ubuntu machines from Ubuntu 16.04 LTS (Xenial) onwards.

If you need to install Thunderbird manually, this short guide will walk you through the steps.

### Mail Transfer Agent

On Ubuntu, Postfix is the default supported MTA. It is compatible with the sendmail MTA.

- *Install Postfix* explains how to install and configure Postfix, including how to configure SMTP for secure communications.

Exim4 can be installed in place of sendmail, although its configuration is quite different.

- *Install Exim4* explains how to install and configure Exim4 on Ubuntu.

### Install and configure Postfix

> **Note**
>
> This guide does not cover setting up Postfix *Virtual Domains*. For information on Virtual Domains and other advanced configurations see the references list at the end of this page.

#### Install Postfix

To install Postfix run the following command:

```
sudo apt install postfix
```

It is OK to accept defaults initially by pressing return for each question. Some of the configuration options will be investigated in greater detail in the configuration stage.

> **Warning**
>
> The `mail-stack-delivery` metapackage has been deprecated in Focal. The package still exists for compatibility reasons, but won't setup a working email system.

#### Configure Postfix

There are four things you should decide before configuring:

- The <Domain> for which you'll accept email (we'll use `mail.example.com` in our example)
- The network and class range of your mail server (we'll use `192.168.0.0/24`)
- The username (we're using `steve`)
- Type of mailbox format (`mbox` is the default, but we'll use the alternative, `Maildir`)

To configure postfix, run the following command:

```
sudo dpkg-reconfigure postfix
```

The user interface will be displayed. On each screen, select the following values:

- Internet Site
- `mail.example.com`
- `steve`

- `mail.example.com`, `localhost.localdomain`, `localhost`
- No
- `127.0.0.0/8 \[::ffff:127.0.0.0\]/104 \[::1\]/128 192.168.0.0/24`
- 0
- +
- all

To set the mailbox format, you can either edit the configuration file directly, or use the `postconf` command. In either case, the configuration parameters will be stored in `/etc/postfix/main.cf` file. Later if you wish to re-configure a particular parameter, you can either run the command or change it manually in the file.

### Configure mailbox format

To configure the mailbox format for `Maildir`:

```
sudo postconf -e 'home_mailbox = Maildir/'
```

This will place new mail in `/home/<username>/Maildir` so you will need to configure your Mail Delivery Agent (MDA) to use the same path.

### SMTP authentication

SMTP-AUTH allows a client to identify itself through the Simple Authentication and Security Layer (SASL) authentication mechanism, using Transport Layer Security (TLS) to encrypt the authentication process. Once it has been authenticated, the SMTP server will allow the client to relay mail.

### Configure SMTP authentication

To configure Postfix for SMTP-AUTH using SASL (Dovecot SASL), run these commands at a terminal prompt:

```
sudo postconf -e 'smtpd_sasl_type = dovecot'
sudo postconf -e 'smtpd_sasl_path = private/auth'
sudo postconf -e 'smtpd_sasl_local_domain ='
sudo postconf -e 'smtpd_sasl_security_options = noanonymous'
sudo postconf -e 'broken_sasl_auth_clients = yes'
sudo postconf -e 'smtpd_sasl_auth_enable = yes'
sudo postconf -e 'smtpd_recipient_restrictions = \
permit_sasl_authenticated,permit_mynetworks,reject_unauth_destination'
```

> **Note**
>
> The `smtpd_sasl_path` config parameter is a path relative to the Postfix queue directory.

There are several SASL mechanism properties worth evaluating to improve the security of your deployment. The option “noanonymous” prevents the use of mechanisms that permit anonymous authentication.

### Configure TLS

Next, generate or obtain a digital certificate for TLS. MUAs connecting to your mail server via TLS will need to recognise the certificate used for TLS. This can either be done using a certificate from Let's Encrypt, from a commercial CA or with a self-signed certificate that users manually install/accept.

For MTA-to-MTA, TLS certificates are never validated without prior agreement from the affected organisations. For MTA-to-MTA TLS, there is no reason not to use a self-signed certificate unless local policy requires it. See our *guide on security certificates* for details about generating digital certificates and setting up your own Certificate Authority (CA).

Once you have a certificate, configure Postfix to provide TLS encryption for both incoming and outgoing mail:

```
sudo postconf -e 'smtp_tls_security_level = may'
sudo postconf -e 'smtpd_tls_security_level = may'
sudo postconf -e 'smtp_tls_note_starttls_offer = yes'
sudo postconf -e 'smtpd_tls_chain_files = /etc/ssl/private/server.key,/etc/ssl/
certs/server.crt'
sudo postconf -e 'smtpd_tls_loglevel = 1'
sudo postconf -e 'smtpd_tls_received_header = yes'
sudo postconf -e 'myhostname = mail.example.com'
```

If you are using your own Certificate Authority to sign the certificate, enter:

```
sudo postconf -e 'smtpd_tls_CAfile = /etc/ssl/certs/cacert.pem'
```

Again, for more details about certificates see our *security certificates guide*.

### Outcome of initial configuration

After running all the above commands, Postfix will be configured for SMTP-AUTH with a self-signed certificate for TLS encryption.

Now, the file `/etc/postfix/main.cf` should look like this:

```
# See /usr/share/postfix/main.cf.dist for a commented, more complete
# version

smtpd_banner = $myhostname ESMTP $mail_name (Ubuntu)
biff = no

# appending .domain is the MUA's job.
append_dot_mydomain = no

# Uncomment the next line to generate "delayed mail" warnings
#delay_warning_time = 4h

myhostname = server1.example.com
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
myorigin = /etc/mailname
```

(continues on next page)

(continued from previous page)

```
mydestination = server1.example.com, localhost.example.com, localhost
relayhost =
mynetworks = 127.0.0.0/8
mailbox_command = procmail -a "$EXTENSION"
mailbox_size_limit = 0
recipient_delimiter = +
inet_interfaces = all
smtpd_sasl_local_domain =
smtpd_sasl_auth_enable = yes
smtpd_sasl_security_options = noanonymous
broken_sasl_auth_clients = yes
smtpd_recipient_restrictions =
permit_sasl_authenticated,permit_mynetworks,reject _unauth_destination
smtpd_tls_auth_only = no
smtp_tls_security_level = may
smtpd_tls_security_level = may
smtp_tls_note_starttls_offer = yes
smtpd_tls_key_file = /etc/ssl/private/smtpd.key
smtpd_tls_cert_file = /etc/ssl/certs/smtpd.crt
smtpd_tls_CAfile = /etc/ssl/certs/cacert.pem
smtpd_tls_loglevel = 1
smtpd_tls_received_header = yes
smtpd_tls_session_cache_timeout = 3600s
tls_random_source = dev:/dev/urandom
```

The Postfix initial configuration is now complete. Run the following command to restart the Postfix daemon:

```
sudo systemctl restart postfix.service
```

### SASL

Postfix supports SMTP-AUTH as defined in RFC2554. It is based on SASL. However it is still necessary to set up SASL authentication before you can use SMTP-AUTH.

When using IPv6, the `mynetworks` parameter may need to be modified to allow IPv6 addresses, for example:

```
mynetworks = 127.0.0.0/8, [::1]/128
```

### Configure SASL

Postfix supports two SASL implementations: **Cyrus SASL** and **Dovecot SASL**.

To enable Dovecot SASL the `dovecot-core` package will need to be installed:

```
sudo apt install dovecot-core
```

Next, edit `/etc/dovecot/conf.d/10-master.conf` and change the following:

```
service auth {
  # auth_socket_path points to this userdb socket by default. It's typically
  # used by dovecot-lda, doveadm, possibly imap process, etc. Its default
  # permissions make it readable only by root, but you may need to relax these
  # permissions. Users that have access to this socket are able to get a list
  # of all usernames and get results of everyone's userdb lookups.
  unix_listener auth-userdb {
    #mode = 0600
    #user =
    #group =
  }

  # Postfix smtp-auth
  unix_listener /var/spool/postfix/private/auth {
    mode = 0660
    user = postfix
    group = postfix
  }
 }
```

To permit use of SMTP-AUTH by Outlook clients, change the following line in the **authentication mechanisms** section of `/etc/dovecot/conf.d/10-auth.conf` from:

```
auth_mechanisms = plain
```

to this:

```
auth_mechanisms = plain login
```

Once you have configured Dovecot, restart it with:

```
sudo systemctl restart dovecot.service
```

### Test your setup

SMTP-AUTH configuration is complete – now it is time to test the setup. To see if SMTP-AUTH and TLS work properly, run the following command:

```
telnet mail.example.com 25
```

After you have established the connection to the Postfix mail server, type:

```
ehlo mail.example.com
```

If you see the following in the output, then everything is working perfectly. Type `quit` to exit.

```
250-STARTTLS
250-AUTH LOGIN PLAIN
250-AUTH=LOGIN PLAIN
250 8BITMIME
```

## Troubleshooting

When problems arise, there are a few common ways to diagnose the cause.

### Escaping `chroot`

The Ubuntu Postfix package will, by default, install into a `chroot` environment for security reasons. This can add greater complexity when troubleshooting problems.

To turn off the `chroot` usage, locate the following line in the `/etc/postfix/master.cf` configuration file:

```
smtp      inet  n       -       -       -       -       smtpd
```

Modify it as follows:

```
smtp      inet  n       -       n       -       -       smtpd
```

You will then need to restart Postfix to use the new configuration. From a terminal prompt enter:

```
sudo service postfix restart
```

## SMTPS

If you need secure SMTP, edit `/etc/postfix/master.cf` and uncomment the following line:

```
smtps     inet  n       -       -       -       -       smtpd
  -o smtpd_tls_wrappermode=yes
  -o smtpd_sasl_auth_enable=yes
  -o smtpd_client_restrictions=permit_sasl_authenticated,reject
  -o milter_macro_daemon_name=ORIGINATING
```

## Log viewing

Postfix sends all log messages to `/var/log/mail.log`. However, error and warning messages can sometimes get lost in the normal log output so they are also logged to `/var/log/mail.err` and `/var/log/mail.warn` respectively.

To see messages entered into the logs in real time you can use the `tail -f` command:

```
tail -f /var/log/mail.err
```

## Increase logging detail

The amount of detail recorded in the logs can be increased via the configuration options. For example, to increase TLS activity logging set the `smtpd_tls_loglevel` option to a value from 1 to 4.

```
sudo postconf -e 'smtpd_tls_loglevel = 4'
```

Reload the service after any configuration change, to activate the new config:

```
sudo systemctl reload postfix.service
```

### Logging mail delivery

If you are having trouble sending or receiving mail from a specific domain you can add the domain to the `debug_peer_list` parameter.

```
sudo postconf -e 'debug_peer_list = problem.domain'
sudo systemctl reload postfix.service
```

### Increase daemon verbosity

You can increase the verbosity of any Postfix daemon process by editing the `/etc/postfix/master.cf` and adding a `-v` after the entry. For example, edit the `smtp` entry:

```
smtp      unix  -       -       -       -       -       smtp -v
```

Then, reload the service as usual:

```
sudo systemctl reload postfix.service
```

### Log SASL debug info

To increase the amount of information logged when troubleshooting SASL issues you can set the following options in `/etc/dovecot/conf.d/10-logging.conf`

```
auth_debug=yes
auth_debug_passwords=yes
```

As with Postfix, if you change a Dovecot configuration the process will need to be reloaded:

```
sudo systemctl reload dovecot.service
```

**Note**

Some of the options above can drastically increase the amount of information sent to the log files. Remember to return the log level back to normal after you have corrected the problem – then reload the appropriate daemon for the new configuration to take effect.

### References

Administering a Postfix server can be a very complicated task. At some point you may need to turn to the Ubuntu community for more experienced help.

- The Postfix website documents all available configuration options.
- O'Reilly's Postfix: The Definitive Guide is rather dated but provides deep background information about configuration options.
- The Ubuntu Wiki Postfix page has more information from an Ubuntu context.

- There is also a Debian Wiki Postfix page that's a bit more up to date; they also have a set of Postfix Tutorials for different Debian versions.
- Info on how to set up mailman3 with postfix.

### Install and configure Exim4

#### Install Exim4

To install Exim4, run the following command:

```
sudo apt install exim4
```

#### Configure Exim4

To configure Exim4, run the following command:

```
sudo dpkg-reconfigure exim4-config
```

This displays a "wizard" user interface for configuring the software. One important question in this configuration is whether Exim4 should split the configuration over multiple files, or use a single configuration file.

> **Note**
>
> The default configuration layout for Exim4 is the single configuration file one.

If using multiple configuration files, then the configuration will be split in a directory structure under `/etc/exim4/conf.d`, like so:

```
/etc/exim4/
└── conf.d
    ├── acl
    ├── auth
    ├── main
    ├── retry
    ├── rewrite
    ├── router
    └── transport
```

Each subdirectory contains one or more individual configuration files.

If, however, Exim4 was set up to use a single configuration file, then that file will be `/etc/exim4/exim4.conf.template`. It will essentially be as if all individual configuration files from the previous layout were concatenated into one file.

In any of these scenarios, after making a change to the configuration, the following command must be executed to update the actual configuration file that Exim4 will use:

```
sudo update-exim4.conf
```

The `update-exim4.conf` command will update the autogenerated configuration file stored in `/var/lib/exim4/config.autogenerated`. This is the actual configuration file that Exim4 uses.

**Warning**

You should never manually edit the configuration file `/var/lib/exim4/config.autogenerated`, because it is updated automatically every time you run `update-exim4.conf`. Any changes you make to it will eventually be lost.

If configuration changes were made, the service should also be restarted:

```
sudo systemctl restart exim4
```

All the choices made via `dpkg-reconfigure exim4-config` are stored in the `/etc/exim4/update-exim4.conf.conf` file. To re-configure the software you can either re-run `dpkg-reconfigure` as before, or manually edit this file using your preferred editor.

### Start the Exim4 daemon

The following command will start the Exim4 daemon:

```
sudo service exim4 start
```

### SMTP authentication

There are multiple authentication options available for Exim4. Here we will document two methods:

- Authenticate Linux users present in the local shadow file (`/etc/shadow`), via `saslauthd` and PAM.
- Authenticate arbitrary users against a custom Exim4 password database (`/etc/exim4/passwd`).

Both of these methods use clear text passwords transmitted over the network, so they need to be protected by Transport Layer Security (TLS).

**Warning**

All configuration steps shown from now on will assume a split-configuration mode for Exim4. If you have selected the non-split mode, then all commands that edit a configuration file under `/etc/exim4/conf.d` in the sections below should be replaced with editing the single file `/etc/exim4/exim4.conf.template`.

### Enabling TLS

First, enter the following into a terminal prompt to create a certificate for use with TLS:

```
sudo /usr/share/doc/exim4-base/examples/exim-gencert
```

This command will ask some questions about the certificate, like country, city, and others. The most important one, and that must be correct otherwise TLS won't work for this server, is the "Server name" one. It **MUST** match the fully qualified hostname (FQDN) of the system where Exim4 is deployed.

**Warning**

This will install a self-signed certificate. If deploying this system in production, you must get a proper certificate signed by a recognized Certificate Authority (CA), or, if using an internal, you will have to distribute the CA to all clients expected to connect to this server.

Configure Exim4 for TLS by editing the `/etc/exim4/conf.d/main/03_exim4-config_tlsoptions` file and adding the following:

```
MAIN_TLS_ENABLE = yes
```

### Authenticating existing Linux users

To authenticate existing Linux users, that is, users who already have accounts on this system, we will use the `saslauthd` service.

**Note**

To manage local Linux users, please refer to *User management*.

Configure Exim4 to use the `saslauthd` daemon for authentication by editing `/etc/exim4/conf.d/auth/30_exim4-config_examples` – uncomment the `plain_saslauthd_server` and `login_saslauthd_server` sections:

```
plain_saslauthd_server:
  driver = plaintext
  public_name = PLAIN
  server_condition = ${if saslauthd{{$auth2}{$auth3}}{1}{0}}
  server_set_id = $auth2
  server_prompts = :
  .ifndef AUTH_SERVER_ALLOW_NOTLS_PASSWORDS
  server_advertise_condition = ${if eq{$tls_cipher}{}{}{*}}
  .endif

login_saslauthd_server:
  driver = plaintext
  public_name = LOGIN
  server_prompts = "Username:: : Password::"
  # don't send system passwords over unencrypted connections
  server_condition = ${if saslauthd{{$auth1}{$auth2}}{1}{0}}
  server_set_id = $auth1
  .ifndef AUTH_SERVER_ALLOW_NOTLS_PASSWORDS
  server_advertise_condition = ${if eq{$tls_cipher}{}{}{*}}
  .endif
```

This enables the `PLAIN` and `LOGIN` authentication mechanisms via `saslauthd`.

For Ubuntu 22.04 and earlier, of it you plan to use authentication mechanisms that will need read access to `/etc/sasldb2` (not covered in this guide), you need to add the `Debian-exim` user to the `sasl` group:

```
sudo gpasswd -a Debian-exim sasl
```

To make all these changes effective, the main configuration file needs to be updated, and Exim4 restarted:

```
sudo update-exim4.conf
sudo systemctl restart exim4
```

This concludes the Exim4 side of the configuration. Next, the `sasl2-bin` package needs to be installed:

```
sudo apt install sasl2-bin
```

The main configuration for `saslauthd` is in the `/etc/default/saslauthd` file. What needs to be verified is the `MECHANISMS` setting, which we want to be `PAM`:

```
MECHANISMS="pam"
```

> **Note**
>
> In Ubuntu 22.04 Jammy and earlier, we also need to add `START="yes"` to `/etc/default/saslauthd`.

Finally, enable and start the `saslauthd` service:

```
sudo systemctl enable saslauthd
sudo systemctl start saslauthd
```

Exim4 is now configured with SMTP-AUTH using TLS authenticating local Linux users via PAM.

### Authenticating arbitrary users

Exim4 can also be configured to authenticate arbitrary users, that is, users that do note exist on the local system. These mechanisms are called `plain_server` and `login_server`. Edit `/etc/exim4/conf.d/auth/30_exim4-config_examples` and uncomment these sections:

```
plain_server:
  driver = plaintext
  public_name = PLAIN
  server_condition = "${if crypteq{$auth3}{${extract{1}{:}{${lookup{$auth2}lsearch
{CONFDIR/passwd}{$value}{*:*}}}}}{1}{0}}"
  server_set_id = $auth2
  server_prompts = :
  .ifndef AUTH_SERVER_ALLOW_NOTLS_PASSWORDS
  server_advertise_condition = ${if eq{$tls_in_cipher}{}{}{*}}
  .endif

login_server:
  driver = plaintext
  public_name = LOGIN
```

(continues on next page)

(continued from previous page)

```
  server_prompts = "Username:: : Password::"
  server_condition = "${if crypteq{$auth2}{${extract{1}{:}{${lookup{$auth1}lsearch
{CONFDIR/passwd}{$value}{*:*}}}}}{1}{0}}"
  server_set_id = $auth1
  .ifndef AUTH_SERVER_ALLOW_NOTLS_PASSWORDS
  server_advertise_condition = ${if eq{$tls_in_cipher}{}{}{*}}
  .endif
```

**Warning**

DO NOT enable both these and the `_saslauthd_server` variants (from "Authenticating existing Linux users" above) at the same time!

These mechanisms will lookup usernames and passwords in the `/etc/exim4/passwd` file, which has to be created and populated. The format of this file is:

```
username:crypted-password:cleartext-password
```

The Exim4 installation ships a helper script that can populate this file. It is a simple interactive script that can be run like this:

```
sudo /usr/share/doc/exim4-base/examples/exim-adduser
```

It will prompt for a username and password. In this example we are creating an `ubuntu` entry with the password `ubuntusecret`:

```
User: ubuntu
Password: ubuntusecret
```

After that, we will have a `/etc/exim4/passwd` file, owned by `root:root` and mode `0644`, with contents similar to this:

```
ubuntu:$1$ZvPA$HTddFobmJD1vURtJHBmbw/:ubuntusecret
```

Since this file contains secrets, it should be protected, and Exim4 has to be allowed to read it:

```
sudo chown root:Debian-exim /etc/exim4/passwd
sudo chmod 0640 /etc/exim4/passwd
```

The same script can also be used to manage users in this `passwd` file:

- To change the password of an existing user, edit the `passwd` file, delete the line corresponding to the user, save the file, and run the script again to provide the new password.
- To add another user, run the script and provide the new user name, and their password.
- To remove a user, edit the file with a text editor and delete the line corresponding to the user that should be removed.

**Warning**

The `/usr/share/doc/exim4-base/examples/exim-adduser` serves mostly as an example and is not able to handle many scenarios. For example, it won't check if the username you are providing already exists in the `passwd` file, which can lead to multiple entries for the same user, with unpredictable results.

Finally, update the Exim4 configuration and restart the service:

```
sudo update-exim4.conf
sudo systemctl restart exim4
```

**Note**

There is no need to restart Exim4 after making changes to the `/etc/exim4/passwd` file.

### Troubleshooting

Exim4 has logs in its own directory in `/var/log/exim4/mainlog`. Whenever troubleshooting the service, always look at that log file.

A quick test to verify if `saslauthd` is working can be performed with the `testsaslauthd` command. Assuming you have a local user called `ubuntu` with a password of `ubuntusecret`, this command can be used to test the authentication on the Exim4 server:

```
testsaslauthd -u ubuntu -p ubuntusecret
```

The result should be OK:

```
0: OK "Success."
```

Note that this tests only the `saslauthd` service, not the Exim4 integration with it. For that we need to actually connect to the SMTP service and try out the authentication. A good helper tool for this is shipped in the `cyrus-clients` package. Since this is part of another email system, it's best to install it on another machine, and not on the same machine as Exim4.

```
sudo apt install cyrus-clients --no-install-recommends
```

Here we are using the extra `--no-install-recommends` option because we don't need all the other components of the Cyrus email system.

The tool we are interested in is called `smtptest`, and its documentation can be inspected in its manual page at `cyrus-smtptest(1)`.

For our purposes, we will run it like this, assuming an `ubuntu` user with the `ubuntusecret` password, and that the Exim4 server is running on the `n-exim.lxd` system:

```
/usr/lib/cyrus/bin/smtptest -t "" -a ubuntu -w ubuntusecret n-exim.lxd
```

The command-line parameters are:

- `-t ""`: Enable TLS.

- `-a ubuntu`: Use `ubuntu` as the authenticating user.
- `-w ubuntusecret`: Authenticate using the `ubuntusecret` password.
- `n-exim.lxd`: The hostname of the Exim4 server to connect to.

If all works well, the output will be similar to this, showing that the connection was switched to TLS, and the authentication worked:

```
S: 220 n-exim ESMTP Exim 4.97 Ubuntu Mon, 23 Jun 2025 21:11:59 +0000
C: EHLO smtptest
S: 250-n-exim Hello n-exim.lxd [10.10.17.9]
S: 250-SIZE 52428800
S: 250-8BITMIME
S: 250-PIPELINING
S: 250-PIPECONNECT
S: 250-CHUNKING
S: 250-STARTTLS
S: 250-PRDR
S: 250 HELP
C: STARTTLS
S: 220 TLS go ahead
verify error:num=18:self-signed certificate
TLS connection established: TLSv1.3 with cipher TLS_AES_256_GCM_SHA384 (256/256
bits)
C: EHLO smtptest
S: 250-n-exim Hello n-exim.lxd [10.10.17.9]
S: 250-SIZE 52428800
S: 250-8BITMIME
S: 250-PIPELINING
S: 250-PIPECONNECT
S: 250-AUTH PLAIN LOGIN
S: 250-CHUNKING
S: 250-PRDR
S: 250 HELP
C: AUTH LOGIN
S: 334 VXNlcm5hbWU6
C: dWJ1bnR1
S: 334 UGFzc3dvcmQ6
C: dWJ1bnR1c2VjcmV0
S: 235 Authentication succeeded
Authenticated.
Security strength factor: 256
```

It will appear to hang at this point, but it's just waiting for the SMTP commands, i.e., receive an email. You can exit by typing `QUIT` followed by pressing enter.

Interesting points to note in the output above:

- No authentication was offered before the connection was switched to TLS. That's because the only mechanisms which are configured are plain-text ones. Without TLS, the password would be exposed on the network.
- Since this documentation used a self-signed certificate, that was highlighted right before the TLS session was established. A real email client would probably abort the

connnection at this point.

- After TLS was established, the `LOGIN` mechanism was chosen.
- The username and password are sent base64 encoded. Do not mistake that for encryption: this is just an encoding mechanism!

> **Tip**
>
> Want to obtain the original username and password back from the base64 encoded values? Feed those values to the `base64 -d` tool. Example, using the value from the session above:
>
> ```
> $ echo -n dWJ1bnR1c2VjcmV0 | base64 -d; echo
> ubuntusecret
> ```

To test the `PLAIN` mechanism, add the `-m plain` command-line option:

```
/usr/lib/cyrus/bin/smtptest -t "" -a ubuntu -w ubuntusecret -m plain n-exim.lxd
```

In the new output, `PLAIN` was selected:

```
S: 220 n-exim ESMTP Exim 4.97 Ubuntu Mon, 23 Jun 2025 21:15:39 +0000
C: EHLO smtptest
S: 250-n-exim Hello n-exim.lxd [10.10.17.9]
S: 250-SIZE 52428800
S: 250-8BITMIME
S: 250-PIPELINING
S: 250-PIPECONNECT
S: 250-CHUNKING
S: 250-STARTTLS
S: 250-PRDR
S: 250 HELP
C: STARTTLS
S: 220 TLS go ahead
verify error:num=18:self-signed certificate
TLS connection established: TLSv1.3 with cipher TLS_AES_256_GCM_SHA384 (256/256
bits)
C: EHLO smtptest
S: 250-n-exim Hello n-exim.lxd [10.10.17.9]
S: 250-SIZE 52428800
S: 250-8BITMIME
S: 250-PIPELINING
S: 250-PIPECONNECT
S: 250-AUTH PLAIN LOGIN
S: 250-CHUNKING
S: 250-PRDR
S: 250 HELP
C: AUTH PLAIN AHVidW50dQB1YnVudHVzZWNyZXQ=
S: 235 Authentication succeeded
Authenticated.
Security strength factor: 256
```

### Troubleshooting tips

Here are some troubleshooting tips.

### Permissions

- If using `saslauthd`: Can the `Debian-exim` user read and write to the `saslauthd` socket in `/run/saslauthd/mux` socket?
- If using `/etc/exim4/passwd`: Can the `Debian-exim` user read this file?

### Config

- If changing a configuration file under `/etc/exim4/conf.d/`, make sure to be using the split-config mode! Check the `/etc/exim4/update-exim4.conf.conf` file to see which mode is in use.
- Similarly, if changing the configuration file `/etc/exim4/exim4.conf.template`, make sure to be using the non-split mode.
- After any configuration file change, be it split mode or not, be sure to run `sudo update-exim4.conf` and restart the `exim4` service.

### References

- See exim.org for more information.
- Another resource is the Exim4 Ubuntu Wiki page.
- Further resources to set up mailman3 with Exim4.

### Mail Delivery Agent

Dovecot is an MDA written with security primarily in mind. It supports the mbox and Maildir mailbox formats.

- *Install Dovecot* explains how to set up Dovecot as an IMAP or POP3 server

### Install and configure Dovecot

### Install Dovecot

To install a basic Dovecot server with common POP3 and IMAP functions, run the following command:

```
sudo apt install dovecot-imapd dovecot-pop3d
```

There are various other Dovecot modules including `dovecot-sieve` (mail filtering), `dovecot-solr` (full text search), `dovecot-antispam` (spam filter training), `dovecot-ldap` (user directory).

### Configure Dovecot

To configure Dovecot, edit the file `/etc/dovecot/dovecot.conf` and its included config files in `/etc/dovecot/conf.d/`. By default, all installed protocols will be enabled via an *include* directive in `/etc/dovecot/dovecot.conf`.

```
!include_try /usr/share/dovecot/protocols.d/*.protocol
```

IMAPS and POP3S are more secure because they use SSL encryption to connect. A basic self-signed SSL certificate is automatically set up by package `ssl-cert` and used by Dovecot in `/etc/dovecot/conf.d/10-ssl.conf`.

`Mbox` format is configured by default, but you can also use `Maildir` if required. More details can be found in the comments in `/etc/dovecot/conf.d/10-mail.conf`. Also see the Dovecot web site to learn about further benefits and details.

Make sure to also configure your chosen Mail Transport Agent (MTA) to transfer the incoming mail to the selected type of mailbox.

### Restart the Dovecot daemon

Once you have configured Dovecot, restart its daemon in order to test your setup using the following command:

```
sudo service dovecot restart
```

Try to log in with the commands `telnet localhost pop3` (for POP3) or `telnet localhost imap2` (for IMAP). You should see something like the following:

```
bhuvan@rainbow:~$ telnet localhost pop3
Trying 127.0.0.1...
Connected to localhost.localdomain.
Escape character is '^]'.
+OK Dovecot ready.
```

### Dovecot SSL configuration

By default, Dovecot is configured to use SSL automatically using the package `ssl-cert` which provides a self signed certificate.

You can instead generate your own custom certificate for Dovecot using `openssh`, for example:

```
sudo openssl req -new -x509 -days 1000 -nodes -out "/etc/dovecot/dovecot.pem" \
    -keyout "/etc/dovecot/private/dovecot.pem"
```

Next, edit `/etc/dovecot/conf.d/10-ssl.conf` and amend following lines to specify that Dovecot should use these custom certificates :

```
ssl_cert = </etc/dovecot/private/dovecot.pem
ssl_key = </etc/dovecot/private/dovecot.key
```

You can get the SSL certificate from a Certificate Issuing Authority or you can create self-signed one. Once you create the certificate, you will have a key file and a certificate file that

you want to make known in the config shown above.

**See also**

For more details on creating custom certificates, see our guide on security certificates.

#### Configure a firewall for an email server

To access your mail server from another computer, you must configure your firewall to allow connections to the server on the necessary ports.

- IMAP - 143
- IMAPS - 993
- POP3 - 110
- POP3S - 995

#### References

- The Dovecot website has more general information about Dovecot.
- The Dovecot manual provides full documentation for Dovecot use.
- The Dovecot Ubuntu Wiki page has more details on configuration.

# 2.7. Web services

Our *Web services section* shows how to set up the different components of web servers, including:

- **Apache2** and **nginx**
- **Squid proxy servers**
- **Web programming** (PHP and Ruby)

## 2.7.1. Web services

Web servers are used to serve content over a network (or the Internet). If you want more of an introduction to the different types of web servers available in Ubuntu, check out our *Introduction to web servers*.

#### Proxy servers

This section shows how to set up a Squid proxy caching server.

#### How to install a Squid server

Squid is a filtering and caching mechanism for web servers that can optimise bandwidth and performance. For more information about Squid proxy servers, *refer to this guide*.

### Install Squid

At a terminal prompt, enter the following command to install the Squid server:

```
sudo apt install squid
```

### Configure Squid

Squid is configured by editing directives in the `/etc/squid/squid.conf` configuration file. The following examples illustrate a sample of directives that can be modified to configure the Squid server's behavior. For more in-depth configuration details, see the links at the bottom of the page.

### Protect the original config file

Before editing the configuration file, you should make a copy of the original and protect it from writing. You will then have the original settings as a reference, and can reuse it when needed. Run the following commands to make a copy of the original configuration file and protect it from being written to:

```
sudo cp /etc/squid/squid.conf /etc/squid/squid.conf.original
sudo chmod a-w /etc/squid/squid.conf.original
```

### Change TCP port

To set your Squid server to listen on TCP port 8888 instead of the default TCP port 3128, change the `http_port` directive as such:

```
http_port 8888
```

### Set the hostname

Change the `visible_hostname` directive to give the Squid server a specific *hostname*. This hostname does not need to be the same as the computer's hostname. In this example it is set to `weezie`:

```
visible_hostname weezie
```

### Configure the memory cache

The default setting is to use on-memory cache. This example tells squid to use up to 512MB of memory, erasing the last recently used content when the cache is full to free space for new items:

```
cache_mem 512 MB
maximum_memory_policy lru
```

### Configure on-disk cache

By changing the `cache_dir` directive you can configure use of an on-disk cache. The `cache_dir` directive takes the following arguments:

```
cache_dir <Type> <Directory-Name> <Size-in-MB> <L1-Dirs> <L2-Dirs> [options]
```

In this example we set the cache configuration to use a `ufs` storage in `/var/spool/squid`, up to 10GB, with 16 directories on the first level of the hierarchy, each of those containing 256 directories for organization.

```
cache_dir ufs /var/spool/squid 10000 16 256
```

The available storage types are:

- `ufs`: This is the common Squid storage format, good for general use.
- `aufs`: Uses the same storage format as `ufs`, using POSIX-threads to avoid blocking the main Squid process on disk-I/O. This was formerly known in Squid as `async-io`.
- `diskd`: Uses the same storage format as `ufs`, using a separate process to avoid blocking the main Squid process on disk-I/O.
- `rock`: This is a database-style storage. All cached entries are stored in a "database" file, using fixed-size slots. A single entry occupies one or more slots.

### Configure cached objects size limits

The following configuration directives control which objects get cached based on their size, for space optimization, both on disk and in memory:

```
maximum_object_size 512 MB
minimum_object_size 0 KB
maximum_object_size_in_memory 512 KB
```

### Configure cached objects lifetime

Using the `refresh_pattern` configuration directive controls how long cached objects stay fresh before they need to be revalidated with the origin server. It is configured as:

```
refresh_pattern regex min percent max [options]
```

where `regex` needs to match against the filename, `min` and `max` set the time limits in minutes for freshness, and a `percent` of the object's age to calculate the refresh threshold.

In the following example, static web assets (such as images, css and scripts) are configured to be kept for 7 to 30 days + 90% of their age, based on the `Last-Modified` header, while everything else is kept up to 3 days + 20% of their age.

```
refresh_pattern -i \.(gif|jpg|png|css|js)$  10080  90%  43200
refresh_pattern .                           0      20%  4320
```

### Caching HTTPS content

By default, Squid can't cache HTTPS because the traffic is encrypted. There are different strategies for enabling HTTPS caching, such as TLS interception via `CONNECT` requests, origin server caching based on `Cache-Control` and `ETag` headers, or access-control only solutions.

Please refer to the Squid HTTPS documentation to learn more.

### Other caching configuration options

Different options can be used to fine-tune the caching behavior overall, by determining how squid stores files in the hierarchy, the algorithm for the replacement policy, DNS cache settings, compatibility with different scenarios, and more.

For a full list of configuration entries, please refer to the Squid configuration guide.

### Access control

Using Squid's access control, you can configure use of Squid-proxied Internet services to be available only to users with certain Internet Protocol (IP) addresses. For example, we will illustrate access by users of the `192.168.42.0/24` subnetwork only:

- Add the following to the **bottom** of the *ACL* section of your `/etc/squid/squid.conf` file:

```
acl fortytwo_network src 192.168.42.0/24
```

- Then, add the following to the **top** of the `http_access` section of your `/etc/squid/squid.conf` file:

```
http_access allow fortytwo_network
```

Using Squid's access control features, you can configure Squid-proxied Internet services to only be available during normal business hours. As an example, we'll illustrate access by employees of a business which is operating between 9:00AM and 5:00PM, Monday through Friday, and which uses the `10.1.42.0/24` subnetwork:

- Add the following to the **bottom** of the ACL section of your `/etc/squid/squid.conf` file:

```
acl biz_network src 10.1.42.0/24
acl biz_hours time M T W T F 9:00-17:00
```

- Then, add the following to the **top** of the `http_access` section of your `/etc/squid/squid.conf` file:

```
http_access allow biz_network biz_hours
```

### Restart the Squid server

After making any changes to the `/etc/squid/squid.conf` file, you will need to save the file and restart the squid server application.

First, you can verify the syntax of your configuration file by running:

```
sudo squid -k parse
```

You can restart the server using the following command:

```
sudo systemctl restart squid.service
```

**Note**

If a formerly customised squid3 was used to set up the spool at `/var/log/squid3` to be a mount point, but otherwise kept the default configuration, the upgrade will fail. The upgrade tries to rename/move files as needed, but it can't do so for an active mount point. In that case you will need to adapt either the mount point or the config in `/etc/squid/squid.conf` so that they match. The same applies if the **include** config statement was used to pull in more files from the old path at `/etc/squid3/`. In those cases you should move and adapt your configuration accordingly.

### Troubleshooting

To monitor Squid behavior and check for potential errors and problems, there are useful commands to be executed and files to be checked.

Squid version and status can be checked with:

```
sudo squid -v
sudo systemctl status squid
```

Checking or watching the log files may be useful to see potential errors, and to verify cache hits and misses:

```
sudo cat /var/log/squid/cache.log
sudo cat /var/log/squid/access.log
```

For a status summary containing runtime statistics and congfiguration, run:

```
squidclient mgr:info
```

While monitoring, these cache status indicators can help identifying what is happening with requests:

- `TCP_MISS`: Content not in cache, fetched from origin
- `TCP_HIT`: Content served from disk cache
- `TCP_MEM_HIT`: Content served from memory cache
- `TCP_REFRESH_HIT`: Cached content revalidated with origin
- `TCP_TUNNEL`: HTTPS traffic (not cached by default)

A healthy cache should show increasing hit ratios over time, of non-zero size, and a growing number of cached objects.

### Further reading

- The Squid Website
- Ubuntu Wiki page on Squid.

### Web servers

Two of the most popular web servers in Ubuntu are Apache2 and nginx. This section covers the installation, configuration and extension of both.

### Apache2

### How to install Apache2

The Apache HTTP Server ("httpd") is the most commonly used web server on Linux systems, and is often used as part of the LAMP configuration. In this guide, we will show you how to install and configure Apache2, which is the current release of "httpd".

### Install apache2

To install Apache2, enter the following command at the terminal prompt:

```
sudo apt install apache2
```

### Configure apache2

Apache2 is configured by placing **directives** in plain text configuration files in `/etc/apache2/`. These *directives* are separated between the following files and directories:

### Files

- `apache2.conf` The main Apache2 configuration file. Contains settings that are **global** to Apache2.

  > **Note**
  >
  > Historically, the main Apache2 configuration file was `httpd.conf`, named after the "httpd" daemon. In other distributions (or older versions of Ubuntu), the file might be present. In modern releases of Ubuntu, all configuration options have been moved to `apache2.conf` and the below referenced directories and `httpd.conf` no longer exists.

- `envvars` File where Apache2 **environment** variables are set.
- `magic` Instructions for determining MIME type based on the first few bytes of a file.
- `ports.conf` Houses the directives that determine which TCP ports Apache2 is listening on.

In addition, other configuration files may be added using the **Include** directive, and wildcards can be used to include many configuration files. Any directive may be placed in any of these

configuration files. Changes to the main configuration files are only recognised by Apache2 when it is started or restarted.

The server also reads a file containing MIME document types; the filename is set by the `TypesConfig` directive, typically via `/etc/apache2/mods-available/mime.conf`, which might also include additions and overrides, and is `/etc/mime.types` by default.

### Directories

- `conf-available` This directory contains available configuration files. All files that were previously in `/etc/apache2/conf.d` should be moved to `/etc/apache2/conf-available`.
- `conf-enabled` Holds **symlinks** to the files in `/etc/apache2/conf-available`. When a configuration file is symlinked, it will be enabled the next time Apache2 is restarted.
- `mods-available` This directory contains configuration files to both load **modules** and configure them. Not all modules will have specific configuration files, however.
- `mods-enabled` Holds symlinks to the files in `/etc/apache2/mods-available`. When a module configuration file is symlinked it will be enabled the next time Apache2 is restarted.
- `sites-available` This directory has configuration files for Apache2 **Virtual Hosts**. Virtual Hosts allow Apache2 to be configured for multiple sites that have separate configurations.
- `sites-enabled` Like `mods-enabled`, `sites-enabled` contains symlinks to the `/etc/apache2/sites-available` directory. Similarly, when a configuration file in `sites-available` is symlinked, the site configured by it will be active once Apache2 is restarted.

### Detailed configuration

For more detailed information on configuring Apache2, check out our follow-up guides.

- Part 2: *Apache2 configuration settings*
- Part 3: *how to extend Apache2* with modules.

### Further reading

- Apache2 Documentation contains in depth information on Apache2 configuration directives. Also, see the `apache2-doc` package for the official Apache2 docs.
- O'Reilly's Apache Cookbook is a good resource for accomplishing specific Apache2 configurations.
- For Ubuntu-specific Apache2 questions, ask in the *#ubuntu-server* IRC channel on libera.chat.

### How to configure Apache2 settings

After you have *installed Apache2*, you will likely need to configure it. In this explanatory guide, we will explain the Apache2 server essential configuration parameters.

### Basic directives

Apache2 ships with a "virtual-host-friendly" default configuration – it is configured with a single default virtual host (using the **VirtualHost** directive) which can be modified or used as-is if you have a single site, or used as a template for additional virtual hosts if you have multiple sites.

If left alone, the default virtual host will serve as your default site, or the site users will see if the URL they enter does not match the **ServerName** directive of any of your custom sites. To modify the default virtual host, edit the file `/etc/apache2/sites-available/000-default.conf`.

> **Note**
>
> The directives set for a virtual host only apply to that particular virtual host. If a directive is set server-wide and not defined in the virtual host settings, the default setting is used. For example, you can define a Webmaster email address and not define individual email addresses for each virtual host.

If you want to configure a new virtual host or site, copy the `000-default.conf` file into the same directory with a name you choose. For example:

```
sudo cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-available/mynewsite.conf
```

Edit the new file to configure the new site using some of the directives described below:

### The ServerAdmin directive

Found in `/etc/apache2/sites-available`

Specifies the email address to be advertised for the server's administrator. The default value is `webmaster@localhost`. This should be changed to an email address that is delivered to you (if you are the server's administrator). If your website has a problem, Apache2 will display an error message containing this email address to report the problem to.

### The Listen directive

Found in `/etc/apache2/ports.conf`

Specifies the port, and optionally the IP address, Apache2 should listen on. If the IP address is not specified, Apache2 will listen on all IP addresses assigned to the machine it runs on. The default value for the **Listen** directive is `80`. Change this to:

- `127.0.0.1:80` to make Apache2 listen only on your loopback interface so that it will not be available to the Internet,
- to e.g. `81` to change the port that it listens on,
- or leave it as is for normal operation.

### The ServerName directive (optional)

Specifies what *FQDN* your site should answer to. The default virtual host has no **ServerName** directive specified, so it will respond to all requests that do not match a ServerName directive in another virtual host. If you have just acquired the domain name `mynewsite.com` and wish to host it on your Ubuntu server, the value of the ServerName directive in your virtual host configuration file should be `mynewsite.com`.

Add this directive to the new virtual host file you created earlier (`/etc/apache2/sites-available/mynewsite.conf`).

### The ServerAlias directive

You may also want your site to respond to `www.mynewsite.com`, since many users will assume the www prefix is appropriate – use the *ServerAlias* directive for this. You may also use wildcards in the ServerAlias directive.

For example, the following configuration will cause your site to respond to any domain request ending in *.mynewsite.com*.

```
ServerAlias *.mynewsite.com
```

### The DocumentRoot directive

Specifies where Apache2 should look for the files that make up the site. The default value is `/var/www/html`, as specified in `/etc/apache2/sites-available/000-default.conf`. If desired, change this value in your site's virtual host file, and remember to create that directory if necessary!

Enable the new *VirtualHost* using the a2ensite utility and restart Apache2:

```
sudo a2ensite mynewsite
sudo systemctl restart apache2.service
```

**Note**

Be sure to replace `mynewsite` with a more descriptive name for the VirtualHost. One method is to name the file after the **ServerName** directive of the VirtualHost.

Similarly, use the `a2dissite` utility to disable sites. This is can be useful when troubleshooting configuration problems with multiple virtual hosts:

```
sudo a2dissite mynewsite
sudo systemctl restart apache2.service
```

### Apache2 server default settings

This section explains configuration of the Apache2 server default settings. For example, if you add a virtual host, the settings you configure for the virtual host take precedence for that virtual host. For a directive not defined within the virtual host settings, the default value is used.

### The DirectoryIndex

The **DirectoryIndex** is the default page served by the server when a user requests an index of a directory by specifying a forward slash (/) at the end of the directory name.

For example, when a user requests the page `http://www.example.com/this_directory/`, they will get either the DirectoryIndex page (if it exists), a server-generated directory list (if it does not and the Indexes option is specified), or a Permission Denied page if neither is true.

The server will try to find one of the files listed in the DirectoryIndex directive and will return the first one it finds. If it does not find any of these files and if **Options Indexes** is set for that directory, the server will generate and return a list, in HTML format, of the subdirectories and files in the directory. The default value, found in `/etc/apache2/mods-available/dir.conf` is "index.html index.cgi index.pl index.php index.xhtml index.htm". Thus, if Apache2 finds a file in a requested directory matching any of these names, the first will be displayed.

### The ErrorDocument

The **ErrorDocument** directive allows you to specify a file for Apache2 to use for specific error events. For example, if a user requests a resource that does not exist, a 404 error will occur.

By default, Apache2 will return a HTTP 404 Return code. Read `/etc/apache2/conf-available/localized-error-pages.conf` for detailed instructions on using ErrorDocument, including locations of example files.

### CustomLog and ErrorLog

By default, the server writes the transfer log to the file `/var/log/apache2/access.log`. You can change this on a per-site basis in your virtual host configuration files with the **CustomLog** directive, or omit it to accept the default, specified in `/etc/apache2/conf-available/other-vhosts-access-log.conf`.

You can also specify the file to which errors are logged, via the **ErrorLog** directive, whose default is `/var/log/apache2/error.log`. These are kept separate from the transfer logs to aid in troubleshooting problems with your Apache2 server. You may also specify the **LogLevel** (the default value is "warn") and the **LogFormat** (see `/etc/apache2/apache2.conf` for the default value).

### The Options directive

Some options are specified on a per-directory basis rather than per-server. **Options** is one of these directives. A Directory stanza is enclosed in XML-like tags, like so:

```
<Directory /var/www/html/mynewsite>
...
</Directory>
```

The Options directive within a Directory stanza accepts one or more of the following values (among others), separated by spaces:

- **ExecCGI** Allow CGI scripts to be run. CGI scripts are not run if this option is not chosen.

> **Caution**
>
> Most files should not be run as CGI scripts. This would be very dangerous. CGI scripts should kept in a directory separate from and outside your **DocumentRoot**, and only this directory should have the ExecCGI option set. This is the default, and the default location for CGI scripts is `/usr/lib/cgi-bin`.

- **Includes** Allow **server-side includes**. Server-side includes allow an HTML file to *include* other files. See Apache SSI documentation (Ubuntu community) for more information.
- **IncludesNOEXEC** Allow server-side includes, but disable the `#exec` and `#include` commands in CGI scripts.
- **Indexes** Display a formatted list of the directory's contents, if no DirectoryIndex (such as `index.html`) exists in the requested directory.

> **Caution**
>
> For security reasons, this should usually not be set, and certainly should not be set on your DocumentRoot directory. Enable this option carefully on a per-directory basis **only** if you are certain you want users to see the entire contents of the directory.

- **Multiview** Support content-negotiated multiviews; this option is disabled by default for security reasons. See the Apache2 documentation on this option.
- **SymLinksIfOwnerMatch** Only follow symbolic links if the target file or directory has the same owner as the link.

### Apache2 daemon settings

This section briefly explains some basic Apache2 daemon configuration settings.

- **LockFile** The **LockFile** directive sets the path to the lockfile used when the server is compiled with either `USE_FCNTL_SERIALIZED_ACCEPT` or `USE_FLOCK_SERIALIZED_ACCEPT`. It must be stored on the local disk. It should be left to the default value unless the logs directory is located on an NFS share. If this is the case, the default value should be changed to a location on the local disk and to a directory that is readable only by root.
- **PidFile** The **PidFile** directive sets the file in which the server records its process ID (pid). This file should only be readable by root. In most cases, it should be left to the default value.
- **User** The **User** directive sets the userid used by the server to answer requests. This setting determines the server's access. Any files inaccessible to this user will also be inaccessible to your website's visitors. The default value for User is "www-data".

> **Warning**
>
> Unless you know exactly what you are doing, do not set the User directive to root. Using root as the User will create large security holes for your Web server.

- **Group** The **Group** directive is similar to the User directive. Group sets the group under which the server will answer requests. The default group is also “www-data”.

### Extending Apache2

Now that you know how to configure Apache2, you may also want to know *how to extend Apache2* with modules.

### Further reading

- The Apache2 Documentation contains in depth information on Apache2 configuration directives. Also, see the `apache2-doc` package for the official Apache2 docs.
- O’Reilly’s Apache Cookbook is a good resource for accomplishing specific Apache2 configurations.
- For Ubuntu specific Apache2 questions, ask in the `#ubuntu-server` IRC channel on libera.chat.

### How to use Apache2 modules

Apache2 is a modular server. This implies that only the most basic functionality is included in the core server. Extended features are available through modules which can be loaded into Apache2.

By default, a base set of modules is included in the server at compile-time. If the server is compiled to use dynamically loaded modules, then modules can be compiled separately, and added at any time using the **LoadModule** directive. Otherwise, Apache2 must be recompiled to add or remove modules.

Ubuntu compiles Apache2 to allow the dynamic loading of modules. Configuration directives may be conditionally included on the presence of a particular module by enclosing them in an `<IfModule>` block.

### Installing and handling modules

You can install additional Apache2 modules and use them with your web server. For example, run the following command at a terminal prompt to install the Python 3 WSGI module:

```
sudo apt install libapache2-mod-wsgi-py3
```

The installation will enable the module automatically, but we can disable it with `a2dismod`:

```
sudo a2dismod wsgi
sudo systemctl restart apache2.service
```

And then use the `a2enmod` utility to re-enable it:

```
sudo a2enmod wsgi
sudo systemctl restart apache2.service
```

See the `/etc/apache2/mods-available` directory for additional modules already available on your system.

### Configure Apache2 for HTTPS

The `mod_ssl` module adds an important feature to the Apache2 server - the ability to encrypt communications. Thus, when your browser is communicating using SSL, the `https://` prefix is used at the beginning of the Uniform Resource Locator (URL) in the browser navigation bar.

The `mod_ssl` module is available in the `apache2-common` package. Run the following command at a terminal prompt to enable the `mod_ssl` module:

```
sudo a2enmod ssl
```

There is a default HTTPS configuration file in `/etc/apache2/sites-available/default-ssl.conf`. In order for Apache2 to provide HTTPS, a **certificate** and **key** file are also needed. The default HTTPS configuration will use a certificate and key generated by the `ssl-cert` package. They are good for testing, but the auto-generated certificate and key should be replaced by a certificate specific to the site or server.

> **Note**
>
> For more information on generating a key and obtaining a certificate see *Certificates*.

To configure Apache2 for HTTPS, enter the following:

```
sudo a2ensite default-ssl
```

> **Note**
>
> The directories `/etc/ssl/certs` and `/etc/ssl/private` are the default locations. If you install the certificate and key in another directory make sure to change *SSLCertificateFile* and *SSLCertificateKeyFile* appropriately.

With Apache2 now configured for HTTPS, restart the service to enable the new settings:

```
sudo systemctl restart apache2.service
```

Note that depending on how you obtained your certificate, you may need to enter a passphrase when Apache2 restarts.

You can access the secure server pages by typing `https://your_hostname/url/` in your browser address bar.

### Sharing write permission

For more than one user to be able to write to the same directory you will need to grant write permission to a group they share in common. The following example grants shared write permission to `/var/www/html` to the group "webmasters".

```
sudo chgrp -R webmasters /var/www/html
sudo chmod -R g=rwX /var/www/html/
```

These commands recursively set the group permission on all files and directories in `/var/www/html` to allow reading, writing and searching of directories. Many admins find this useful for allowing multiple users to edit files in a directory tree.

**Warning**

The `apache2` daemon will run as the `www-data` user, which has a corresponding `www-data` group. These **should not** be granted write access to the document root, as this would mean that vulnerabilities in Apache or the applications it is serving would allow attackers to overwrite the served content.

### Further reading

- The Apache2 Documentation contains in depth information on Apache2 configuration directives. Also, see the apache2-doc package for the official Apache2 docs.
- O'Reilly's Apache Cookbook is a good resource for accomplishing specific Apache2 configurations.
- For Ubuntu specific Apache2 questions, ask in the `#ubuntu-server` IRC channel on libera.chat.
- *Install Apache2*
- *Configure Apache2*
- *Extend Apache2 with modules*

## Nginx

### How to install nginx

The nginx HTTP server is a powerful alternative to *Apache*. In this guide, we will demonstrate how to install and use nginx for web services.

### Install nginx

To install nginx, enter the following command at the terminal prompt:

```
$ sudo apt update
$ sudo apt install nginx
```

This will also install any required dependency packages, and some common mods for your server, and then start the nginx web server.

### Verify nginx is running

You can verify that nginx is running via this command:

```
$ sudo systemctl status nginx
  ● nginx.service - A high performance web server and a reverse proxy server
       Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset:
enabled)
```

(continues on next page)

(continued from previous page)

```
       Active: active (running) since Sun 2023-08-20 01:04:22 UTC; 53s ago
         Docs: man:nginx(8)
      Process: 28210 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_
process on; (code=exited, status=0/SU\
    CCESS)

      Process: 28211 ExecStart=/usr/sbin/nginx -g daemon on; master_process on;
(code=exited, status=0/SUCCESS)
     Main PID: 28312 (nginx)
        Tasks: 13 (limit: 76969)
       Memory: 13.1M
          CPU: 105ms
       CGroup: /system.slice/nginx.service
               ├─28312 "nginx: master process /usr/sbin/nginx -g daemon on;
master_process on;"
               ├─28314 "nginx: worker process" "" "" "" "" "" "" "" "" "" "" "" ""
"" "" "" "" ""
               ...
```

### Restarting nginx

To restart nginx, run:

```
$ sudo systemctl restart nginx
```

### Enable/disable nginx manually

By default, Nginx will automatically start at boot time. To disable this behaviour so that you can start it up manually, you can disable it:

```
$ sudo systemctl disable nginx
```

Then, to re-enable it, run:

```
$ sudo systemctl enable nginx
```

### The default nginx homepage

A default nginx home page is also set up during the installation process. You can load this page in your web browser using your web server's IP address; http://your_server_ip.

The default home page should look similar to:

```
    Welcome to nginx!
                                                Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working.
Further configuration is
required.
```

(continues on next page)

(continued from previous page)

```
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.
```

### Setting up nginx

For more information on customising nginx for your needs, see these follow-up guides:

- Part 2: {ref}`How to configure nginx
- Part 3: {ref}`How to use nginx modules

### Further reading

- The nginx documentation provides detailed explanations of configuration directives.
- O'Reilly's nginx cookbook provides guidance on solving specific needs
- For Ubuntu-specific nginx questions, ask in the `#ubuntu-server` IRC channel on libera.chat.

### How to configure nginx

Once you have *installed nginx*, you can customise it for your use with the configuration options explained in this guide.

### Server blocks

nginx organises sets of site-specific configuration details into **server blocks**, and by default comes pre-configured for single-site operation. This can either be used "as-is", or as a starting template for serving multiple sites.

The single-site configuration serves files out of `/var/www/html`, as defined by the server block and as provided by `/etc/nginx/sites-enabled/default`:

```
server {
        listen 80 default_server;
        listen [::]:80 default_server;

        root /var/www/html;

        # Add index.php to the list if you are using PHP
        index index.html index.htm index.nginx-debian.html;

        server_name _;

        location / {
                # First attempt to serve request as file, then
                # as directory, then fall back to displaying a 404.
                try_files $uri $uri/ =404;
```

(continues on next page)

(continued from previous page)

```
        }
}
```

Even for a single-site configuration, while you can place your website at `/var/www/html`, you may want to place the website's files at a different location in your *filesystem*. For example, if you were hosting `www.my-site.org` from `/srv/my-site/html` you might edit the above file to look like this:

```
server {
        listen                  80;
        root                    /srv/my-site/html;
        index                   index.html;
        server_name             my-site.org www.my-site.org;

        location / {
                try_files $uri $uri/ =404;
        }
}
```

Make sure to create your web root directory structure:

```
$ sudo mkdir -p /srv/my-site/html
$ sudo chmod -R 755 /srv/my-site/html
$ echo "<html><body><h1>My Site!</h1></body></html>" > /srv/my-site/html/index.
html
```

Then, to make nginx reload its configuration, run:

```
$ sudo systemctl reload nginx
```

Check that the settings have taken effect using your web browser:

```
$ www-browser www.my-site.org
```

### Multi-site hosting

Similar to Apache, nginx uses the `sites-available` and `sites-enabled` directories for the configurations of multiple websites. Unlike with Apache, you'll need to handle the enablement manually.

To do that, first create a new server block in a configuration file as above, and save it to `/etc/nginx/sites-available/<your-domain>`. Make sure to give each site a unique `server_name` and a different `listen` port number.

Next, enable the site by creating a symlink to it from the `sites-enabled` directory:

```
$ sudo ln -s /etc/nginx/sites-available/<your-domain> /etc/nginx/sites-enabled/
```

To disable a website, you can delete the symlink in `sites-enabled`. For example, once you have your new site(s) configured and no longer need the default site configuration:

```
$ sudo rm /etc/nginx/sites-available/default
```

### SSL and HTTPS

While establishing an HTTP website on port 80 is a good starting point (and perhaps adequate for static content), production systems will want HTTPS, such as serving on port 443 with SSL enabled via `cert` files. A server block with such a configuration might look like this, with HTTP-to-HTTPS redirection handled in the first block, and HTTPS in the second block:

```
server {
        listen                  80;
        server_name             our-site.org www.our-site.org;
        return                  301 https://$host$request_uri;
}

server {
        listen                  443 ssl;

        root                    /srv/our-site/html;
        index                   index.html;

        server_name             our-site.org www.our-site.org;

        ssl_certificate         our-site.org.crt;
        ssl_certificate_key     our-site.org.key;
        ssl_protocols           TLSv1 TLSv1.1 TLSv1.2 TLSv1.3;
        ssl_ciphers             HIGH:!aNULL:!MD5;
        ssl_session_timeout     15m;

        location / {
                try_files $uri $uri/ =404;
        }
}
```

Thanks to the `return 301` line in the above configuration, anyone visiting the site on port 80 via an HTTP URL will get automatically redirected to the equivalent secure HTTPS URL.

Refer to the *security - certificates* page in this manual for details on how to create and manage certificates, and the *OpenSSL* page for additional details on configuring and using that service. The *GnuTLS* section explains how to configure different SSL protocol versions and their associated ciphers.

For example, to generate a self-signed certificate, you might run a set of commands similar to these:

```
$ sudo openssl genrsa -out our-site.org.key 2048

$ openssl req -nodes -new -key our-site.org.key -out ca.csr

$ openssl x509 -req -days 365 -in our-site.org.csr -signkey our-site.org.key -out
our-site.org.crt
```

(continues on next page)

(continued from previous page)

```
$ mkdir /etc/apache2/ssl

$ cp our-site.org.crt our-site.org.key our-site.org.csr /etc/apache2/ssl/
```

### Setting up nginx

Beyond the settings outlined above, nginx can be further customised through the use of modules. Please see the next guide in this series for details of how to do that.

- Part 3: *How to use nginx modules*

### Further reading

- nginx's beginner's guide covers use cases such as proxy servers, *FastCGI* for use with PHP and other frameworks, and optimising the handling of static content.
- The nginx documentation describes HTTPS server configuration in greater detail, including certificate chains, disambiguating various multi-site certificate situations, performance optimisations and compatibility issues.
- For Ubuntu-specific nginx questions, ask in the `#ubuntu-server` IRC channel on libera.chat.

## How to use nginx modules

Like other web servers, nginx supports dynamically loaded modules to provide in-server support for programming languages, security mechanisms, and so on. Ubuntu provides a number of these modules as separate packages that are either installed simultaneously with nginx, or can be installed separately.

### Available modules

nginx will report the modules it has been built with via its `-V` option. A quick and dirty way to list the available modules is thus:

```
$ nginx -V 2>&1 | tr -- - '\n' | grep _module

http_ssl_module

http_stub_status_module

http_realip_module

...

http_image_filter_module=dynamic

http_perl_module=dynamic

http_xslt_module=dynamic
```

(continues on next page)

(continued from previous page)

```
stream_geoip_module=dynamic
```

Many of these modules are built-in and thus are always available with nginx, but some exist as separate packages whose installation status can be checked via `apt`. For example:

```
$ apt policy libnginx-mod-http-image-filter

libnginx-mod-http-image-filter:

  Installed: (none)

  Candidate: 1.24.0-1ubuntu1

  Version table:

     1.24.0-1ubuntu1 500

        500 http://archive.ubuntu.com/ubuntu mantic/main amd64 Packages
```

`apt` can also be used to install the desired dynamic module:

```
$ sudo apt install libnginx-mod-http-image-filter

...

The following NEW packages will be installed:

  libnginx-mod-http-image-filter

0 upgraded, 1 newly installed, 0 to remove and 34 not upgraded.

...

Triggering nginx reload

...
```

### Enabling and disabling dynamic modules

Dynamic modules are automatically enabled and get reloaded by nginx on installation. If you need to manually disable an installed module, remove its file from the `/etc/nginx/modules-enabled` directory, for example:

```
$ ls /etc/nginx/modules-*

/etc/nginx/modules-available:
```

(continues on next page)

(continued from previous page)

```
/etc/nginx/modules-enabled:

50-mod-http-image-filter.conf

$ sudo mv /etc/nginx/modules-enabled/50-mod-http-image-filter.conf /etc/nginx/
modules-available/

$ service nginx restart
```

Note that built-in modules cannot be disabled/enabled.

### Configuring modules

The installed configuration file for an nginx module mainly consists of the dynamically-loaded binary library:

```
## /etc/nginx/modules-enabled/50-mod-http-image-filter.conf
load_module modules/ngx_http_image_filter_module.so;
```

Note that you can also use the `load_module` parameter in your `/etc/nginx/nginx.conf` at the top level, if preferred for some reason.

To use a module for your website, its settings are specified in your server block. For example:

```
location /img/ {
    image_filter resize 240 360;
    image_filter rotate 180;
    image_filter_buffer 16M;
    error_page   415 = /415.html;
}
```

### Further reading

You've completed the nginx guide! See the following resources for more in-depth information on further extending nginx's capabilities:

- The nginx documentation provides detailed explanations of configuration directives.
- O'Reilly's nginx cookbook provides guidance on solving specific needs.
- For Ubuntu-specific nginx questions, ask in the `#ubuntu-server` IRC channel on libera.chat.
- *Install nginx*
- *Configure nginx*
- *Extend nginx with modules*

### Web programming

It is common to set up server-side scripting languages to support the creation of dynamic web content. Whichever scripting language you choose, you will need to have installed and configured your web and database servers beforehand.

### How to install and configure PHP

PHP is a general-purpose scripting language well-suited for Web development since PHP scripts can be embedded into HTML. This guide explains how to install and configure PHP in an Ubuntu System with Apache2 and MySQL.

### Prerequisites

Before installing PHP you should install Apache (or a preferred web server) and a database service such as MySQL.

- To install the Apache package, please refer to *our Apache guide*.
- To install and configure a MySQL database service, refer to *our MySQL guide*.

### Install PHP

PHP is available on Ubuntu Linux, but unlike Python (which comes pre-installed), must be manually installed.

To install PHP – and the Apache PHP module – you can enter the following command into a terminal prompt:

```
sudo apt install php libapache2-mod-php
```

### Install optional packages

The following packages are optional, and can be installed if you need them for your setup.

- **PHP-CLI** You can run PHP scripts via the Command Line Interface (CLI). To do this, you must first install the `php-cli` package. You can install it by running the following command:

  ```
  sudo apt install php-cli
  ```

- **PHP-CGI** You can also execute PHP scripts without installing the Apache PHP module. To accomplish this, you should install the `php-cgi` package via this command:

  ```
  sudo apt install php-cgi
  ```

- **PHP-MySQL** To use MySQL with PHP you should install the `php-mysql` package, like so:

  ```
  sudo apt install php-mysql
  ```

- **PHP-PgSQL** Similarly, to use PostgreSQL with PHP you should install the `php-pgsql` package:

  ```
  sudo apt install php-pgsql
  ```

### Configure PHP

If you have installed the `libapache2-mod-php` or `php-cgi` packages, you can run PHP scripts from your web browser. If you have installed the `php-cli` package, you can run PHP scripts at a terminal prompt.

By default, when `libapache2-mod-php` is installed, the Apache2 web server is configured to run PHP scripts using this module. First, verify if the files `/etc/apache2/mods-enabled/php8.*.conf` and `/etc/apache2/mods-enabled/php8.*.load` exist. If they do not exist, you can enable the module using the `a2enmod` command.

Once you have installed the PHP-related packages and enabled the Apache PHP module, you should restart the Apache2 web server to run PHP scripts, by running the following command:

```
sudo systemctl restart apache2.service
```

### Test your setup

To verify your installation, you can run the following PHP `phpinfo` script:

```
<?php
  phpinfo();
?>
```

You can save the content in a file – `phpinfo.php` for example – and place it under the *DocumentRoot* directory of the Apache2 web server. Pointing your browser to `http://hostname/phpinfo.php` will display the values of various PHP configuration parameters.

### Further reading

- For more in depth information see the php.net documentation.
- There are a plethora of books on PHP 7 and PHP 8. A good book from O'Reilly is "Learning PHP", which includes an exploration of PHP 7's enhancements to the language.
- Also, see the Apache MySQL PHP Ubuntu Wiki page for more information.

## How to install and configure Ruby on Rails

Ruby on Rails is an open source web framework for developing database-backed web applications. It is optimised for sustainable productivity of the programmer since it lets the programmer to write code by favouring convention over configuration. This guide explains how to install and configure Ruby on Rails for an Ubuntu system with Apache2 and MySQL.

### Prerequisites

Before installing Rails you should install Apache (or a preferred web server) and a database service such as MySQL.

- To install the Apache package, please refer to *our Apache guide*.
- To install and configure a MySQL database service, refer to *our MySQL guide*.

### Install `rails`

Once you have a web server and a database service installed and configured, you are ready to install the Ruby on Rails package, `rails`, by entering the following in the terminal prompt.

```
sudo apt install rails
```

This will install both the Ruby base packages, and Ruby on Rails.

### Configure the web server

You will need to modify the `/etc/apache2/sites-available/000-default.conf` configuration file to set up your domains.

The first thing to change is the *DocumentRoot* directive:

```
DocumentRoot /path/to/rails/application/public
```

Next, change the `<Directory "/path/to/rails/application/public">` directive:

```
<Directory "/path/to/rails/application/public">
        Options Indexes FollowSymLinks MultiViews ExecCGI
        AllowOverride All
        Order allow,deny
        allow from all
        AddHandler cgi-script .cgi
</Directory>
```

You should also enable the `mod_rewrite` module for Apache. To enable the `mod_rewrite` module, enter the following command into a terminal prompt:

```
sudo a2enmod rewrite
```

Finally, you will need to change the ownership of the `/path/to/rails/application/public` and `/path/to/rails/application/tmp` directories to the user that will be used to run the Apache process:

```
sudo chown -R www-data:www-data /path/to/rails/application/public
sudo chown -R www-data:www-data /path/to/rails/application/tmp
```

If you need to compile your application assets run the following command in your application directory:

```
RAILS_ENV=production rake assets:precompile
```

### Configure the database

With your database service in place, you need to make sure your app database configuration is also correct. For example, if you are using MySQL the your `config/database.yml` should look like this:

```
# Mysql
production:
```

(continues on next page)

(continued from previous page)

```
  adapter: mysql2
  username: user
  password: password
  host: 127.0.0.1
  database: app
```

To finally create your application database and apply its migrations you can run the following commands from your app directory:

```
RAILS_ENV=production rake db:create
RAILS_ENV=production rake db:migrate
```

That's it! Now your Server is ready for your Ruby on Rails application. You can *daemonize* your application as you want.

#### Further reading

- See the Ruby on Rails website for more information.
- Agile Development with Rails is also a great resource.

#### See also

- Explanation: *Web services*

# 2.8. Graphics

The *Graphics* section contains guides on how to set up both on-system and virtual GPU.

## 2.8.1. Graphics

#### On-system GPU

#### NVIDIA drivers installation

This page shows how to install the NVIDIA drivers from the command line, using either the `ubuntu-drivers` tool (recommended), or APT.

#### NVIDIA drivers releases

We package two types of NVIDIA drivers:

1. **Unified Driver Architecture (UDA)** drivers - which are recommended for the generic desktop use, and which you can also find on the NVIDIA website.
2. **Enterprise Ready Drivers** (*ERD*) - which are recommended on servers and for computing tasks. Their packages can be recognised by the `-server` suffix. You can read more about these drivers in the NVIDIA documentation.

Additionally, we package the **NVIDIA Fabric Manager** and the **NVIDIA Switch Configuration and Query (NSCQ) Library**, which you will only need if you have NVswitch hardware. The Fabric Manager and NSCQ library are only available with the ERDs or `-server` driver versions.

### Check driver versions

To check the version of your currently running driver:

```
cat /proc/driver/nvidia/version
```

### The recommended way (ubuntu-drivers tool)

The `ubuntu-drivers` tool relies on the same logic as the "Additional Drivers" graphical tool, and allows more flexibility on desktops and on servers.

The `ubuntu-drivers` tool is recommended if your computer uses Secure Boot, since it always tries to install signed drivers which are known to work with Secure Boot.

### Check the available drivers for your hardware

For desktop:

```
sudo ubuntu-drivers list
```

or, for servers:

```
sudo ubuntu-drivers list --gpgpu
```

You should see a list such as the following:

```
nvidia-driver-470
nvidia-driver-470-server
nvidia-driver-535
nvidia-driver-535-open
nvidia-driver-535-server
nvidia-driver-535-server-open
nvidia-driver-550
nvidia-driver-550-open
nvidia-driver-550-server
nvidia-driver-550-server-open
```

### Installing the drivers for generic use (e.g. desktop and gaming)

You can either rely on automatic detection, which will install the driver that is considered the best match for your hardware:

```
sudo ubuntu-drivers install
```

Or you can tell the `ubuntu-drivers` tool which driver you would like installed. If this is the case, you will have to use the driver version (such as `535`) that you saw when you used the `ubuntu-drivers list` command.

Let's assume we want to install the `535` driver:

```
sudo ubuntu-drivers install nvidia:535
```

### Installing the drivers on servers and/or for computing purposes

You can either rely on automatic detection, which will install the driver that is considered the best match for your hardware:

```
sudo ubuntu-drivers install --gpgpu
```

Or you can tell the `ubuntu-drivers` tool which driver you would like installed. If this is the case, you will have to use the driver version (such as `535`) and the `-server` suffix that you saw when you used the `ubuntu-drivers list --gpgpu` command.

Let's assume we want to install the `535-server` driver (listed as `nvidia-driver-535-server`):

```
sudo ubuntu-drivers install --gpgpu nvidia:535-server
```

You will also want to install the following additional components:

```
sudo apt install nvidia-utils-535-server
```

### Optional step

If your system comes with NVswitch hardware, then you will want to install Fabric Manager and the NVSwitch Configuration and Query library. You can do so by running the following:

```
sudo apt install nvidia-fabricmanager-535 libnvidia-nscq-535
```

**Note**

While `nvidia-fabricmanager` and `libnvidia-nscq` do not have the same `-server` label in their name, they are really meant to match the `-server` drivers in the Ubuntu archive. For example, `nvidia-fabricmanager-535` will match the `nvidia-driver-535-server` package version (not the `nvidia-driver-535` package).

### Manual driver installation (using APT)

Installing the NVIDIA driver manually means installing the correct kernel modules first, then installing the metapackage for the driver series.

### Installing the kernel modules

If your system uses Secure Boot (as most x86 modern systems do), your kernel will require the kernel modules to be signed. There are two (mutually exclusive) ways to achieve this.

### Installing the pre-compiled NVIDIA modules for your kernel

Install the metapackage for your kernel flavour (e.g. `generic`, `lowlatency`, etc) which is specific to the driver branch (e.g. `535`) that you want to install, and whether you want the compute vs. general display driver (e.g. `-server` or not):

```
sudo apt install linux-modules-nvidia-${DRIVER_BRANCH}${SERVER}-${LINUX_FLAVOUR}
```

(e.g. `linux-modules-nvidia-535-generic`)

Check that the modules for your specific kernel/*ABI* were installed by the metapackage:

```
sudo apt-cache policy linux-modules-nvidia-${DRIVER_BRANCH}${SERVER}-$(uname -r)
```

(e.g. `sudo apt-cache policy linux-modules-nvidia-535-$(uname -r)`)

If the modules were not installed for your current running kernel, upgrade to the latest kernel or install them by specifying the running kernel version:

```
sudo apt install linux-modules-nvidia-${DRIVER_BRANCH}${SERVER}-$(uname -r)
```

(e.g. `sudo apt install linux-modules-nvidia-535-$(uname -r)`)

### Building your own kernel modules using the NVIDIA DKMS package

Install the relevant NVIDIA *DKMS* package and `linux-headers` to build the kernel modules, and enroll your own key to sign the modules.

Install the `linux-headers` metapackage for your kernel flavour (e.g. `generic`, `lowlatency`, etc):

```
sudo apt install linux-headers-${LINUX_FLAVOUR}
```

Check that the headers for your specific kernel were installed by the metapackage:

```
sudo apt-cache policy linux-headers-$(uname -r)
```

If the headers for your current running kernel were not installed, install them by specifying the running kernel version:

```
sudo apt install linux-headers-$(uname -r)
```

Finally, install the NVIDIA DKMS package for your desired driver series (this may automatically guide you through creating and enrolling a new key for Secure Boot):

```
sudo apt install nvidia-dkms-${DRIVER_BRANCH}${SERVER}
```

### Installing the user-space drivers and the driver libraries

After installing the correct kernel modules (see the relevant section of this document), install the correct driver metapackage:

```
sudo apt install nvidia-driver-${DRIVER_BRANCH}${SERVER}
```

### (Optional) Installing Fabric Manager and the NSCQ library

If your system comes with NVswitch hardware, then you will want to install Fabric Manager and the NVSwitch Configuration and Query library. You can do so by running the following:

```
sudo apt install nvidia-fabricmanager-${DRIVER_BRANCH} libnvidia-nscq-${DRIVER_
BRANCH}
```

**Note**

While `nvidia-fabricmanager` and `libnvidia-nscq` do not have the same `-server` label in their name, they are really meant to match the `-server` drivers in the Ubuntu archive. For example, `nvidia-fabricmanager-535` will match the `nvidia-driver-535-server` package version (not the `nvidia-driver-535` package).

### Switching between pre-compiled and DKMS modules

1. Uninstalling the NVIDIA drivers (below)
2. Manual driver installation using APT

### Uninstalling the NVIDIA drivers

Remove any NVIDIA packages from your system:

```
sudo apt --purge remove '*nvidia*${DRIVER_BRANCH}*'
```

If you are unsure which `${DRIVER_BRANCH}` to pick for removal you might look at the installed nvidia packages and see the different `${DRIVER_BRANCH}` numbers that are present on your system. Since `autoremove` will take care of all indirect dependencies it is sufficient to list those that have been directly installed by using `apt-mark`.

```
apt-mark showmanual | grep nvidia`.
```

Remove any additional packages that may have been installed as a dependency (e.g. the `i386` libraries on amd64 systems) and which were not caught by the previous command:

```
sudo apt autoremove
```

### Transitional packages to new driver branches

When NVIDIA stops support on a driver branch, then Canonical will transition you to the next supported driver branch automatically if you try to install that driver branch.

See NVIDIA's current support matrix in their documentation.

### Troubleshooting

### Driver/library version mismatch error

If you encounter the following error when running the nvidia-smi command:

```
Failed to initialize NVML: Driver/library version mismatch
```

This typically indicates that the userspace driver packages were upgraded while the kernel module is still on the older version (for example, the client reports one driver version while the kernel module reports another). This situation often occurs after a system upgrade. To verify this, check the kernel logs:

```
sudo dmesg
```

Look for error messages similar to:

```
NVRM: API mismatch: the client has the version 570.172.08, but
NVRM: this kernel module has the version 570.158.01.  Please
NVRM: make sure that this kernel module and all NVIDIA driver
NVRM: components have the same version.
```

**Solution**: Rebooting the system will load the updated kernel module and bring the versions back in sync.

### No devices were found error

If you encounter the following error when running the nvidia-smi command:

```
No devices were found
```

This may occur if the open-source NVIDIA kernel driver nouveau is pre-installed and loaded, which conflicts with the proprietary NVIDIA driver. To check whether nouveau is loaded:

```
lsmod | grep nouveau
```

**Solution**: If nouveau kernel module is loaded, blacklist it and rebuild the initramfs:

```
echo "blacklist nouveau" | sudo tee /etc/modprobe.d/disable-nouveau.conf
echo "options nouveau modeset=0" | sudo tee -a /etc/modprobe.d/disable-nouveau.
conf
sudo rmmod nouveau || true
sudo update-initramfs -u
```

Then reboot the system for the changes to take effect.

## Virtual GPU

A virtual GPU (vGPU) partitions a physical GPU to enable GPU-accelerated workloads in virtualized environments.

### GPU virtualisation with QEMU/KVM

### Graphics

Graphics for QEMU/KVM always comes in two pieces: a *frontend* and a backend.

- `frontend`: Controlled via the `-vga` argument, which is provided to the guest. Usually one of `cirrus`, `std`, `qxl`, or `virtio`. The default these days is `qxl` which strikes a good balance between guest compatibility and performance. The guest needs a driver for whichever option is selected – this is the most common reason to not use the default (e.g., on very old Windows versions).
- `backend`: Controlled via the `-display` argument. This is what the host uses to actually display the graphical content, which can be an application window via `gtk` or a `vnc`.

- In addition, one can enable the `-spice` back end (which can be done in addition to `vnc`). This can be faster and provides more authentication methods than `vnc`.
- If you want no graphical output at all, you can save some memory and CPU cycles by setting `-nographic`.

If you run with `spice` or `vnc` you can use native `vnc` tools or virtualization-focused tools like `virt-viewer`. You can read more about these in the *libvirt section*.

All these options are considered basic usage of graphics, but there are also advanced options for more specific use-cases. Those cases usually differ in their ease-of-use and capability, such as:

- *Need 3D acceleration*: Use `-vga virtio` with a local display having a *GL* context `-display gtk,gl=on`. This will use virgil3d on the host, and guest drivers are needed (which are common in Linux since Kernels >= 4.4 but can be hard to come by for other cases). While not as fast as the next two options, the major benefit is that it can be used without additional hardware and without a proper input-output memory management unit (IOMMU) set up for device passthrough.
- *Need native performance*: Use PCI passthrough of additional *GPUs* in the system. You'll need an *IOMMU* set up, and you'll need to unbind the cards from the host before you can pass it through, like so:

```
-device vfio-pci,host=05:00.0,bus=1,addr=00.0,multifunction=on,x-vga=on -
device vfio-pci,host=05:00.1,bus=1,addr=00.1
```

- *Need native performance, but multiple guests per card*: Like with PCI passthrough, but using mediated devices to shard a card on the host into multiple devices, then passing those:

```
-display gtk,gl=on -device vfio-pci,sysfsdev=/sys/bus/pci/devices/0000:00:02.
0/4dd511f6-ec08-11e8-b839-2f163ddee3b3,display=on,rombar=0
```

  You can read more about vGPU at kraxel and Ubuntu GPU mdev evaluation. The sharding of the cards is driver-specific and therefore will differ per manufacturer – Intel, Nvidia, or *AMD*.

The advanced cases in particular can get pretty complex – it is recommended to use QEMU through *libvirt section* for those cases. libvirt will take care of all but the host kernel/BIOS tasks of such configurations. Below are the common basic actions needed for faster options (i.e., passthrough and mediated devices passthrough).

The initial step for both options is the same; you want to ensure your system has its IOMMU enabled and the device to pass should be in a group of its own. Enabling the VT-d and IOMMU is usually a BIOS action and thereby manufacturer dependent.

### Preparing the input-output memory management unit (IOMMU)

On the kernel side, there are various options you can enable/configure for the IOMMU feature. In recent Ubuntu kernels (>=5.4 => Focal or Bionic-HWE kernels) everything usually works by default, unless your hardware setup makes you need any of those tuning options.

**Note**

The card used in all examples below e.g. when filtering for or assigning PCI IDs, is an NVIDIA V100 on PCI ID 41.00.0

```
$ lspci | grep 3D
41:00.0 3D controller: NVIDIA Corporation GV100GL [Tesla V100 PCIe 16GB] (rev
a1)
```

You can check your boot-up kernel messages for IOMMU/*DMAR* messages or even filter it for a particular PCI ID.

To list all:

```
dmesg | grep -i -e DMAR -e IOMMU
```

Which produces an output like this:

```
[    3.509232] iommu: Default domain type: Translated
...
[    4.516995] pci 0000:00:01.0: Adding to iommu group 0
...
[    4.702729] perf/amd_iommu: Detected AMD IOMMU #0 (2 banks, 4 counters/bank).
```

To filter for the installed 3D card:

```
dmesg | grep -i -e DMAR -e IOMMU | grep $(lspci | awk '/ 3D / {print $1}' )
```

Which shows the following output:

```
[    4.598150] pci 0000:41:00.0: Adding to iommu group 66
```

If you have a particular device and want to check for its group you can do that via `sysfs`. If you have multiple cards or want the full list you can traverse the same `sysfs` paths for that.

For example, to find the group for our example card:

```
find /sys/kernel/iommu_groups/ -name "*$(lspci | awk '/ 3D / {print $1}')*"
```

Which it tells us is found here:

```
/sys/kernel/iommu_groups/66/devices/0000:41:00.0
```

We can also check if there are other devices in this group:

```
ll /sys/kernel/iommu_groups/66/devices/
lrwxrwxrwx 1 root root 0 Jan  3 06:57 0000:41:00.0 -> ../../../../devices/
pci0000:40/0000:40:03.1/0000:41:00.0/
```

Another useful tool for this stage (although the details are beyond the scope of this article) can be `virsh node*`, especially `virsh nodedev-list --tree` and `virsh nodedev-dumpxml <pcidev>`.

> **Note**
>
> Some older or non-server boards tend to group devices in one IOMMU group, which isn't very useful as it means you'll need to pass "all or none of them" to the same guest.

### Preparations for PCI and mediated devices pass-through – block host drivers

For both, you'll want to ensure the normal driver isn't loaded. In some cases you can do that at runtime via `virsh nodedev-detach <pcidevice>`. `libvirt` will even do that automatically if, on the passthrough configuration, you have set `<hostdev mode='subsystem' type='pci' managed='yes'>`.

This usually works fine for e.g. network cards, but some other devices like GPUs do not like to be unassigned, so there the required step usually is block loading the drivers you do not want to be loaded. In our GPU example the `nouveau` driver would load and that has to be blocked. To do so you can create a `modprobe` blacklist.

```
echo "blacklist nouveau" | sudo tee /etc/modprobe.d/blacklist-nouveau.conf

echo "options nouveau modeset=0" | sudo tee -a /etc/modprobe.d/blacklist-nouveau.
conf
sudo update-initramfs -u
sudo reboot
```

You can check which kernel modules are loaded and available via `lspci -v`:

```
lspci -v | grep -A 10 " 3D "
```

Which in our example shows:

```
41:00.0 3D controller: NVIDIA Corporation GV100GL [Tesla V100 PCIe 16GB] (rev a1)
...
Kernel modules: nvidiafb, nouveau
```

If the configuration did not work instead it would show:

```
Kernel driver in use: nouveau
```

### Preparations for mediated devices pass-through - driver

For PCI passthrough, the above steps would be all the preparation needed, but for mediated devices one also needs to install and set up the host driver. The example here continues with our NVIDIA V100 which is supported and available from Nvidia.

There is also an Nvidia document about the same steps available on installation and configuration of vGPU on Ubuntu.

Once you have the drivers from Nvidia, like `nvidia-vgpu-ubuntu-470_470.68_amd64.deb`, then install them and check (as above) that that driver is loaded. The one you need to see is `nvidia_vgpu_vfio`:

```
lsmod | grep nvidia
```

Which we can see in the output:

```
nvidia_vgpu_vfio       53248  38
nvidia              35282944  586 nvidia_vgpu_vfio
mdev                   24576  2 vfio_mdev,nvidia_vgpu_vfio
drm                   491520  6 drm_kms_helper,drm_vram_helper,nvidia
```

**Note**

While it works without a vGPU manager, to get the full capabilities you'll need to configure the vGPU manager (that came with above package) and a license server so that each guest can get a license for the vGPU provided to it. Please see Nvidia's documentation for the license server. While not officially supported on Linux (as of Q1 2022), it's worthwhile to note that it runs fine on Ubuntu with `sudo apt install unzip default-jre tomcat9 liblog4j2-java libslf4j-java` using `/var/lib/tomcat9` as the server path in the license server installer.

It's also worth mentioning that the Nvidia license server went *EOL* on 31 July 2023. At that time, it was replaced by the NVIDIA License System.

Here is an example of those when running fine:

```
# general status
$ systemctl status nvidia-vgpu-mgr
     Loaded: loaded (/lib/systemd/system/nvidia-vgpu-mgr.service; enabled; vendor
preset: enabled)
     Active: active (running) since Tue 2021-09-14 07:30:19 UTC; 3min 58s ago
    Process: 1559 ExecStart=/usr/bin/nvidia-vgpu-mgr (code=exited, status=0/
SUCCESS)
   Main PID: 1564 (nvidia-vgpu-mgr)
      Tasks: 1 (limit: 309020)
     Memory: 1.1M
     CGroup: /system.slice/nvidia-vgpu-mgr.service
             └─1564 /usr/bin/nvidia-vgpu-mgr

Sep 14 07:30:19 node-watt systemd[1]: Starting NVIDIA vGPU Manager Daemon...
Sep 14 07:30:19 node-watt systemd[1]: Started NVIDIA vGPU Manager Daemon.
Sep 14 07:30:20 node-watt nvidia-vgpu-mgr[1564]: notice: vmiop_env_log: nvidia-
vgpu-mgr daemon started

# Entries when a guest gets a vGPU passed
Sep 14 08:29:50 node-watt nvidia-vgpu-mgr[2866]: notice: vmiop_log: (0x0): gpu-
pci-id : 0x4100
Sep 14 08:29:50 node-watt nvidia-vgpu-mgr[2866]: notice: vmiop_log: (0x0): vgpu_
type : Quadro
Sep 14 08:29:50 node-watt nvidia-vgpu-mgr[2866]: notice: vmiop_log: (0x0):
Framebuffer: 0x1dc000000
Sep 14 08:29:50 node-watt nvidia-vgpu-mgr[2866]: notice: vmiop_log: (0x0): Virtual
Device Id: 0x1db4:0x1252
Sep 14 08:29:50 node-watt nvidia-vgpu-mgr[2866]: notice: vmiop_log: (0x0): FRL
```

(continues on next page)

(continued from previous page)

```
Value: 60 FPS
Sep 14 08:29:50 node-watt nvidia-vgpu-mgr[2866]: notice: vmiop_log: ######## vGPU
Manager Information: ########
Sep 14 08:29:50 node-watt nvidia-vgpu-mgr[2866]: notice: vmiop_log: Driver
Version: 470.68
Sep 14 08:29:50 node-watt nvidia-vgpu-mgr[2866]: notice: vmiop_log: (0x0): vGPU
supported range: (0x70001, 0xb0001)
Sep 14 08:29:50 node-watt nvidia-vgpu-mgr[2866]: notice: vmiop_log: (0x0): Init
frame copy engine: syncing...
Sep 14 08:29:50 node-watt nvidia-vgpu-mgr[2866]: notice: vmiop_log: (0x0): vGPU
migration enabled
Sep 14 08:29:50 node-watt nvidia-vgpu-mgr[2866]: notice: vmiop_log: display_init
inst: 0 successful

# Entries when a guest grabs a license
Sep 15 06:55:50 node-watt nvidia-vgpu-mgr[4260]: notice: vmiop_log: (0x0): vGPU
license state: Unlicensed (Unrestricted)
Sep 15 06:55:52 node-watt nvidia-vgpu-mgr[4260]: notice: vmiop_log: (0x0): vGPU
license state: Licensed

# In the guest the card is then fully recognized and enabled
$ nvidia-smi -a | grep -A 2 "Licensed Product"
    vGPU Software Licensed Product
        Product Name                    : NVIDIA RTX Virtual Workstation
        License Status                  : Licensed
```

A mediated device is essentially the partitioning of a hardware device using *firmware* and host driver features. This brings a lot of flexibility and options; in our example we can split our 16G GPU into 2x8G, 4x4G, 8x2G or 16x1G just as we need it. The following gives an example of how to split it into two 8G cards for a compute profile and pass those to guests.

Please refer to the NVIDIA documentation for advanced tunings and different card profiles.

The tool for listing and configuring these mediated devices is `mdevctl`:

```
sudo mdevctl types
```

Which will list the available types:

```
...
  nvidia-300
    Available instances: 0
    Device API: vfio-pci
    Name: GRID V100-8C
    Description: num_heads=1, frl_config=60, framebuffer=8192M, max_
resolution=4096x2160, max_instance=2
```

Knowing the PCI ID (`0000:41:00.0`) and the mediated device type we want (`nvidia-300`) we can now create those mediated devices:

```
$ sudo mdevctl define --parent 0000:41:00.0 --type nvidia-300
bc127e23-aaaa-4d06-a7aa-88db2dd538e0
$ sudo mdevctl define --parent 0000:41:00.0 --type nvidia-300
1360ce4b-2ed2-4f63-abb6-8cdb92100085
$ sudo mdevctl start --parent 0000:41:00.0 --uuid bc127e23-aaaa-4d06-a7aa-
88db2dd538e0
$ sudo mdevctl start --parent 0000:41:00.0 --uuid 1360ce4b-2ed2-4f63-abb6-
8cdb92100085
```

After that, you can check the UUID of your ready mediated devices:

```
$ sudo mdevctl list -d
bc127e23-aaaa-4d06-a7aa-88db2dd538e0 0000:41:00.0 nvidia-108 manual (active)
1360ce4b-2ed2-4f63-abb6-8cdb92100085 0000:41:00.0 nvidia-108 manual (active)
```

Those UUIDs can then be used to pass the mediated devices to the guest - which from here is rather similar to the pass through of a full PCI device.

### Passing through PCI or mediated devices

After the above setup is ready one can pass through those devices, in `libvirt` for a PCI passthrough that looks like:

```
<hostdev mode='subsystem' type='pci' managed='yes'>
  <source>
    <address domain='0x0000' bus='0x41' slot='0x00' function='0x0'/>
  </source>
</hostdev>
```

And for mediated devices it is quite similar, but using the UUID.

```
<hostdev mode='subsystem' type='mdev' managed='no' model='vfio-pci' display='on'>
  <source>
    <address uuid='634fc146-50a3-4960-ac30-f09e5cedc674'/>
  </source>
</hostdev>
```

Those sections can be part of the guest definition itself, to be added on guest startup and freed on guest shutdown. Or they can be in a file and used by for hot-add remove if the hardware device and its drivers support it `virsh attach-device`.

**Note**

This works great on Focal, but `type='none'` as well as `display='off'` weren't available on Bionic. If this level of control is required one would need to consider using the Ubuntu Cloud Archive or Server-Backports for a newer stack of the virtualisation components.

And finally, it might be worth noting that while mediated devices are becoming more common and known for vGPU handling, they are a general infrastructure also used (for example) for s390x vfio-ccw.

- *Virtual GPU (vGPU) with QEMU/KVM*

# 2.9. Virtualisation

Our *Virtualisation* section contains installation and usage guides for common virtualization tools available in Ubuntu, across various layers of abstraction, from Multipass to QEMU.

## 2.9.1. Virtualisation

In this section we show how to install, configure and use various options for creating virtual machines (VMs). For more information about these options, you may want to refer to our *Introduction to virtualization*

### Virtual machines

### How to create a VM with Multipass

Multipass is the recommended method for creating Ubuntu VMs on Ubuntu. It's designed for developers who want a fresh Ubuntu environment with a single command, and it works on Linux, Windows and macOS.

On Linux it's available as a snap:

```
sudo snap install multipass
```

If you're running an older version of Ubuntu where `snapd` isn't pre-installed, you will need to install it first:

```
sudo apt update
sudo apt install snapd
```

### Find available images

To find available images you can use the `multipass find` command, which will produce a list like this:

```
Image                       Aliases           Version          Description
snapcraft:core18            18.04             20201111         Snapcraft builder
for Core 18
snapcraft:core20            20.04             20210921         Snapcraft builder
for Core 20
snapcraft:core22            22.04             20220426         Snapcraft builder
for Core 22
snapcraft:devel                               20221128         Snapcraft builder
for the devel series
core                        core16            20200818         Ubuntu Core 16
core18                                        20211124         Ubuntu Core 18
18.04                       bionic            20221117         Ubuntu 18.04 LTS
20.04                       focal             20221115.1       Ubuntu 20.04 LTS
22.04                       jammy,lts         20221117         Ubuntu 22.04 LTS
22.10                       kinetic           20221101         Ubuntu 22.10
daily:23.04                 devel,lunar       20221127         Ubuntu 23.04
appliance:adguard-home                        20200812         Ubuntu AdGuard Home
Appliance
appliance:mosquitto                           20200812         Ubuntu Mosquitto
Appliance
```

(continues on next page)

(continued from previous page)

```
appliance:nextcloud                          20200812         Ubuntu Nextcloud
Appliance
appliance:openhab                            20200812         Ubuntu openHAB Home
Appliance
appliance:plexmediaserver                    20200812         Ubuntu Plex Media
Server Appliance
anbox-cloud-appliance                        latest           Anbox Cloud
Appliance
charm-dev                                    latest           A development and
testing environment for charmers
docker                                       latest           A Docker
environment with Portainer and related tools
jellyfin                                     latest           Jellyfin is a Free
Software Media System that puts you in control of managing and streaming your
media.
minikube                                     latest           minikube is local
Kubernetes
```

### Launch a fresh instance of the Ubuntu Jammy (22.04) LTS

You can launch a fresh instance by specifying either the image name from the list (in this example, 22.04) or using an alias, if the image has one.

```
$ multipass launch 22.04
Launched: cleansing-guanaco
```

This command is equivalent to: `multipass launch jammy` or `multipass launch lts` in the list above. It will launch an instance based on the specified image, and provide it with a random name – in this case, `cleansing-guanaco`.

### Check out the running instances

You can check out the currently running instance(s) by using the `multipass list` command:

```
$ multipass list
Name                    State             IPv4             Image
cleansing-guanaco       Running           10.140.26.17     Ubuntu 22.04 LTS
```

### Learn more about the VM instance you just launched

You can use the `multipass info` command to find out more details about the VM instance parameters:

```
$ multipass info cleansing-guanaco
Name:           cleansing-guanaco
State:          Running
IPv4:           10.140.26.17
Release:        Ubuntu 22.04.1 LTS
Image hash:     dc5b5a43c267 (Ubuntu 22.04 LTS)
```

(continues on next page)

(continued from previous page)

```
Load:           0.45 0.19 0.07
Disk usage:     1.4G out of 4.7G
Memory usage:   168.3M out of 969.5M
Mounts:         --
```

### Connect to a running instance

To enter the VM you created, use the `shell` command:

```
$ multipass shell cleansing-guanaco
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-53-generic x86_64)
(...)
ubuntu@cleansing-guanaco:~$
```

### Disconnect from the instance

Don't forget to log out (or Ctrl + D) when you are done, or you may find yourself heading all the way down the Inception levels…

### Run commands inside an instance from outside

```
$ multipass exec cleansing-guanaco -- lsb_release -a
No LSB modules are available.
Distributor ID:        Ubuntu
Description:        Ubuntu 22.04.1 LTS
Release:        22.04
Codename:        jammy
```

### Stop or start an instance

You can stop an instance to save resources using the `stop` command:

```
$ multipass stop cleansing-guanaco
```

You can start it back up again using the `start` command:

```
$ multipass start cleansing-guanaco
```

### Delete the instance

Once you are finished with the instance, you can delete it as follows:

```
$ multipass delete cleansing-guanaco
```

It will now show up as deleted when you use the `list` command:

```
$ multipass list
Name                    State             IPv4             Image
cleansing-guanaco       Deleted           --               Not Available
```

And when you want to completely get rid of it (and any other deleted instances), you can use the `purge` command:

```
$ multipass purge
```

Which we can check again using `list`:

```
$ multipass list
No instances found.
```

### Integrate with the rest of your virtualisation

If you already have a hypervisor interacting with *Libvirt*, such as *QEMU*, *KVM*, or *ESXi*, you might be managing virtual machines through tools like virt-manager or the older *uvtool*.

In that case, integrating Multipass with your existing setup would allow VMs to share the same network bridge for communication and be managed using virsh. However, Multipass runs as a headless system, so you don't have direct GUI access through virt-viewer. Follow this guide to set up a GUI.

To begin, integrate Multipass into your existing setup by selecting libvirt as your local driver:

```
$ sudo multipass set local.driver=libvirt
```

> **Note**
>
> If you are having issues interacting with Multipass after switching to the libvirt driver, check if there is a restriction by *AppArmor*, for example. AppArmor may have a default policy which restricts the multipass service from interacting with the libvert service. So you need to add an explicit permission that allows it.

Start a guest, and access it via tools like virt-manager or `virsh`:

```
$ multipass launch lts
Launched: engaged-amberjack

$ virsh list
 Id    Name                           State
----------------------------------------------------
 15    engaged-amberjack              running
```

For more detailed and comprehensive instructions on changing your drivers, refer to the Multipass drivers documentation.

### Get help

You can use the following commands on the CLI:

```
multipass help
multipass help <command>
multipass help --all
```

Or, check out the Multipass documentation for more details on how to use it.

### Create cloud image VMs with UVtool

With Ubuntu being one of the most popular operating systems on many cloud platforms, the availability of stable and secure cloud images has become very important. Since Ubuntu 12.04, the use of cloud images outside of a cloud infrastructure has been improved so that it is now possible to use those images to create a virtual machine without needing a complete installation.

### Creating virtual machines using `uvtool`

Starting with Ubuntu 14.04 LTS, a tool called `uvtool` has greatly facilitated the creation of virtual machines (VMs) using cloud images. `uvtool` provides a simple mechanism for synchronising cloud images locally and using them to create new VMs in minutes.

### Install `uvtool` packages

The following packages and their dependencies are required in order to use `uvtool`:

- `uvtool`
- `uvtool-libvirt`

To install `uvtool`, run:

```
sudo apt -y install uvtool
```

This will install `uvtool`'s main commands, `uvt-simplestreams-libvirt` and `uvt-kvm`.

### Get the Ubuntu cloud image with `uvt-simplestreams-libvirt`

This is one of the major simplifications that `uvtool` provides. It knows where to find the cloud images so you only need one command to get a new cloud image. For instance, if you want to synchronise all cloud images for the amd64 architecture, the `uvtool` command would be:

```
uvt-simplestreams-libvirt --verbose sync arch=amd64
```

After all the images have been downloaded from the Internet, you will have a complete set of locally-stored cloud images. To see what has been downloaded, use the following command:

```
uvt-simplestreams-libvirt query
```

Which will provide you with a list like this:

```
release=bionic arch=amd64 label=daily (20191107)
release=focal arch=amd64 label=daily (20191029)
...
```

In the case where you want to synchronise only one specific cloud image, you need to use the `release=` and `arch=` filters to identify which image needs to be synchronised.

```
uvt-simplestreams-libvirt sync release=DISTRO-SHORT-CODENAME arch=amd64
```

Furthermore, you can provide an alternative URL to fetch images from. A common case is the daily image, which helps you get the very latest images, or if you need access to the not-yet-released development release of Ubuntu. As an example:

```
uvt-simplestreams-libvirt sync --source http://cloud-images.ubuntu.com/daily [...
further options]
```

### Create a valid SSH key

To connect to the virtual machine once it has been created, you must first have a valid SSH key available for the Ubuntu user. If your environment does not have an SSH key, you can create one using the `ssh-keygen` command, which will produce similar output to this:

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_rsa.
Your public key has been saved in /home/ubuntu/.ssh/id_rsa.pub.
The key fingerprint is:
4d:ba:5d:57:c9:49:ef:b5:ab:71:14:56:6e:2b:ad:9b ubuntu@DISTRO-SHORT-CODENAMES
The key's randomart image is:
+--[ RSA 2048]----+
|               ..|
|              o.=|
|          .    **|
|         +    o+=|
|        S . ...=.|
|         o . .+ .|
|        . .  o o |
|             *   |
|            E    |
+-----------------+
```

### Create the VM using `uvt-kvm`

To create a new virtual machine using `uvtool`, run the following in a terminal:

```
uvt-kvm create firsttest
```

This will create a VM named 'firsttest' using the current locally-available LTS cloud image. If you want to specify a release to be used to create the VM, you need to use the `release=` filter, and the short codename of the release, e.g. "jammy":

```
uvt-kvm create secondtest release=DISTRO-SHORT-CODENAME
```

The `uvt-kvm wait` command can be used to wait until the creation of the VM has completed:

```
uvt-kvm wait secondttest
```

### Connect to the running VM

Once the virtual machine creation is completed, you can connect to it using SSH:

```
uvt-kvm ssh secondtest
```

You can also connect to your VM using a regular SSH session using the IP address of the VM. The address can be queried using the following command:

```
$ uvt-kvm ip secondtest
192.168.122.199
$ ssh -i ~/.ssh/id_rsa ubuntu@192.168.122.199
[...]
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@secondtest:~$
```

### Get the list of running VMs

You can get the list of VMs running on your system with the `uvt-kvm list` command.

### Destroy your VM

Once you are finished with your VM, you can destroy it with:

```
uvt-kvm destroy secondtest
```

> **Note**
>
> Unlike libvirt's `destroy` or `undefine` actions, this will (by default) also remove the associated virtual storage files.

### More `uvt-kvm` options

The following options can be used to change some of the characteristics of the VM that you are creating:

- `--memory` : Amount of RAM in megabytes. Default: 512.
- `--disk` : Size of the OS disk in gigabytes. Default: 8.
- `--cpu` : Number of CPU cores. Default: 1.

Some other parameters will have an impact on the cloud-init configuration:

- `--password <password>` : Allows logging into the VM using the Ubuntu account and this provided password.
- `--run-script-once <script_file>` : Run `script_file` as root on the VM the first time it is booted, but never again.
- `--packages <package_list>` : Install the comma-separated packages specified in `package_list` on first boot.

A complete description of all available modifiers is available in the `uvt-kvm(1)` manpages.

### Resources

If you are interested in learning more, have questions or suggestions, please contact the Ubuntu Server Team at:

- IRC: `#ubuntu-server` on Libera
- Mailing list: ubuntu-server at lists.ubuntu.com

## QEMU

> **Note**
>
> Please bear in mind that invoking QEMU manually may sometimes require your user to be part of the `kvm` group.

### Virtualisation with QEMU

QEMU is a machine emulator that can run operating systems and programs for one machine on a different machine. However, it is more often used as a virtualiser in collaboration with KVM kernel components. In that case it uses the hardware virtualisation technology to virtualise guests.

Although QEMU has a command line interface and a monitor to interact with running guests, they are typically only used for development purposes. On the other hand, *libvirt* provides an abstraction from specific versions and hypervisors, and encapsulates some workarounds and best practices.

### Install QEMU/KVM

The first step to using QEMU/KVM on Ubuntu is to check if your system supports KVM.

```
kvm-ok
```

You should get an output saying `KVM acceleration can be used`.

The next step is to install QEMU.

```
sudo apt-get install qemu-system
```

### Boot a VM

The quickest way to get started with QEMU is by booting a VM directly from a netboot ISO. You can achieve this by running the following command:

```
qemu-system-x86_64 -enable-kvm -cdrom http://archive.ubuntu.com/ubuntu/dists/
bionic-updates/main/installer-amd64/current/images/netboot/mini.iso
```

**Caution**

This example is just for illustration purposes - it is not generally recommended without verifying the checksums; *Multipass* and *UVTool* are much better ways to get actual guests easily.

If you are testing this example on a headless system, specify an alternative display method such as *VNC*.

### Create a virtual disk

The command in the previous sub-section boots the system entirely in RAM without persistent storage. To maintain the OS state across reboots, you should allocate space for the VM:

```
qemu-img create -f qcow2 disk.qcow 5G
```

And then we can use the disk space we have just allocated for storage by adding the argument: `-drive file=disk.qcow,format=qcow2`.

These tools can do much more, as you'll discover in their respective (long) manpages. They can also be made more consumable for specific use-cases and needs through a vast selection of auxiliary tools - for example virt-manager for UI-driven use through libvirt. But in general, it comes down to:

```
qemu-system-x86_64 options image[s]
```

So take a look at the `QEMU manpage`, `qemu-img(1)` and the QEMU documentation and see which options best suit your needs.

While a standard QEMU configuration works for most use cases, some scenarios demand high-vCPU VMs. In the next section, we'll cover how to create QEMU virtual machines with up to 1024 vCPUs.

### Create QEMU VMs with up to 1024 vCPUs

For a long time, QEMU only supported launching virtual machines with 288 vCPUs or fewer. While this was acceptable a decade ago, nowadays it is more common to see processors with 300+ physical cores available. For this reason, QEMU has been modified to support virtual machines with up to 1024 vCPUs. The caveat is that the user has to provide a few specific (and not trivial to guess) command line options to enable such a feature, and that is the gap that this document aims to fill.

### Requirement

To support more than 288 vCPUs, some QEMU versions are only compatible with special machine types.

| QEMU version | Ubuntu release | Supported machine types |
|---|---|---|
| QEMU 8.2.1+ | Ubuntu 24.04 LTS (Noble) | `ubuntu` (Native support, also supports Jammy and Mantic machine types) |
| QEMU 8.0.4 | Ubuntu 23.10 (Mantic) | `pc-q35-mantic-maxcpus`, `pc-i440fx-mantic-maxcpus` (Also supports Jammy machine types) |
| QEMU 6.2 | Ubuntu 22.04 LTS (Jammy) | `pc-q35-jammy-maxcpus`, `pc-i440fx-jammy-maxcpus` |

### Configuration by Ubuntu release

#### 24.04 Noble

From Noble onwards, the regular `ubuntu` machine type supports up to 1024 vCPUs out of the box, which simplifies the command used to create such virtual machines:

```
qemu-system-x86_64 -M ubuntu,accel=kvm,kernel-irqchip=split -device intel-iommu,
intremap=on -smp cpus=300,maxcpus=300 ...
```

Although the regular machine type can now be used to launch the virtual machine, it is still necessary to provide some special command line options to make sure that the VM is created with a virtual IOMMU with interrupt mapping.

Now that we've covered high-vCPU configurations for x86_64 VMs, let's look at how to boot ARM64 virtual machines on QEMU.

#### 23.10 Mantic

If you are using QEMU on Mantic, the special machine types are named in a similar fashion to Jammy's: `pc-q35-mantic-maxcpus` or `pc-i440fx-mantic-maxcpus`. Therefore, you command line to create a virtual machine with support for more than 288 vCPUs on Mantic should start with:

```
qemu-system-x86_64 -M pc-q35-mantic-maxcpus,accel=kvm,kernel-irqchip=split -device
intel-iommu,intremap=on -smp cpus=300,maxcpus=300 ...
```

In the example above, the virtual machine will be launched using 300 vCPUs and a `pc-q35-mantic-maxcpus` machine type. You can adjust the option according to your use case.

The `kernel-irqchip=split -device intel-iommu,intremap=on` command line options are required, to make sure that the VM is created with a virtual IOMMU with interrupt mapping. This is needed due to some idiosyncrasies present in this scenario.

Note that both machine types for Mantic are supported in subsequent versions of Ubuntu, so you should be able to migrate your virtual machines to newer versions of QEMU in Ubuntu without problems. As noted in the previous section, it is also possible to create virtual machines using the special Jammy machine types on Mantic.

### 22.04 Jammy

If you are using QEMU on Jammy and want to create VMs with more than 288 vCPUs, you will need to use either of the special `pc-q35-jammy-maxcpus` or `pc-i440fx-jammy-maxcpus` machine types.

The command line needs to start with:

```
qemu-system-x86_64 -M pc-q35-jammy-maxcpus,accel=kvm,kernel-irqchip=split -device
intel-iommu,intremap=on -smp cpus=300,maxcpus=300 ...
```

In the example above, the virtual machine will be launched using 300 vCPUs and a `pc-q35-jammy-maxcpus` machine type. You can adjust the option according to your use case.

The `kernel-irqchip=split -device intel-iommu,intremap=on` command line options are required, to make sure that the VM is created with a virtual IOMMU with interrupt mapping. This is needed due to some idiosyncrasies present in this scenario.

Note that both machine types for Jammy are supported in subsequent versions of Ubuntu, so you should be able to migrate your virtual machines to newer versions of QEMU in Ubuntu without problems.

## Boot ARM64 virtual machines on QEMU

Ubuntu ARM64 images can run inside QEMU. You can either do this fully emulated (e.g. on an x86 host) or accelerated with KVM if you have an ARM64 host. This page describes how to do both.

> **Note**
>
> This requires Ubuntu 20.04 or greater

## Install QEMU to run ARM64 virtual machines

The first step is to install the `qemu-system-arm` package, which needs to be done regardless of where the ARM64 virtual machine will run:

```
sudo apt install qemu-system-arm
```

## Create necessary support files

Next, create a VM-specific flash volume for storing NVRAM variables, which are necessary when booting *EFI* firmware:

```
truncate -s 64m varstore.img
```

We also need to copy the ARM UEFI firmware into a bigger file:

```
truncate -s 64m efi.img
dd if=/usr/share/qemu-efi-aarch64/QEMU_EFI.fd of=efi.img conv=notrunc
```

### Fetch the Ubuntu cloud image

You need to fetch the ARM64 variant of the Ubuntu cloud image you would like to use in the virtual machine. You can go to the official Ubuntu cloud image website, select the Ubuntu release, and then download the variant whose filename ends in `-arm64.img`. For example, if you want to use the latest Jammy cloud image, you should download the file named `jammy-server-cloudimg-arm64.img`.

### Run QEMU natively on an ARM64 host

If you have access to an ARM64 host, you should be able to create and launch an ARM64 virtual machine there. Note that the command below assumes that you have already set up a network bridge to be used by the virtual machine.

```
qemu-system-aarch64 \
 -enable-kvm \
 -m 1024 \
 -cpu host \
 -M virt \
 -nographic \
 -drive if=pflash,format=raw,file=efi.img,readonly=on \
 -drive if=pflash,format=raw,file=varstore.img \
 -drive if=none,file=jammy-server-cloudimg-arm64.img,id=hd0 \
 -device virtio-blk-device,drive=hd0 -netdev type=tap,id=net0 \
 -device virtio-net-device,netdev=net0
```

### Run an emulated ARM64 VM on x86

You can also emulate an ARM64 virtual machine on an x86 host. To do that:

```
qemu-system-aarch64 \
 -m 2048\
 -cpu max \
 -M virt \
 -nographic \
 -drive if=pflash,format=raw,file=efi.img,readonly=on \
 -drive if=pflash,format=raw,file=varstore.img \
 -drive if=none,file=jammy-server-cloudimg-arm64.img,id=hd0 \
 -device virtio-blk-device,drive=hd0 \
 -netdev type=tap,id=net0 \
 -device virtio-net-device,netdev=net0
```

### Troubleshooting

### No output and no response

If you get no output from the QEMU command above, aligning your host and guest release versions may help. For example, if you generated `efi.img` on Focal but want to emulate Jammy (with the Jammy cloud image), the firmware may not be fully compatible. Generating `efi.img` on Jammy when emulating Jammy with the Jammy cloud image may help.

### Resources

QEMU can be extended in many different ways. If you'd like to take QEMU further, you might want to explore these additional resources:

- *Virtualizing graphics using QEMU/KVM*
- *Using QEMU to create a microvm*.

### Confidential Computing with AMD

Secure Encrypted Virtualization - Secure Nested Paging (SEV-SNP) are a set of virtualization features available on the latest AMD EPIC CPUs (starting from "Rome"). These features enable *Confidential Computing* which is a way to better isolate a workload from the hypervisor and the host Operating System. Virtual Machines launched with SEV-SNP features enabled have their CPU registers protected from the host OS and their memory encrypted and integrity protected. The memory protection relies on the CPU's Memory Management Unit (MMU) which verifies the integrity of the secure pages and generates per-VM encryption keys to encrypt them and prevent the host OS from reading their content. While using Ubuntu as a guest OS on SEV-SNP VMs has been supported since Ubuntu 24.04 LTS, the host enablement (QEMU and OVMF support) was only added later with Ubuntu 25.04.

### Host configuration

AMD SEV-SNP is fully supported as of Ubuntu 25.04. To launch a VM with these features, enable memory encryption features and SNP in the firmware settings, and then assign Address-Space Identifiers (ASIDs) to SNP. For more details, refer to AMD's documentation and to the manuals for your specific mainboard or Baseboard Management Controller (BMC).

On the host OS, install `qemu-system-x86_64` and launch QEMU with the following parameters:

```
qemu-system-x86_64 \
  -enable-kvm \
  -nographic \
  -machine q35 -smp 6 -m 6G \
  -drive "if=virtio,format=qcow2,file=disk.img" \
  -net nic,model=e1000 -net user,hostfwd=tcp::2222-:22 \
  -cpu EPYC-v4 \
  -machine memory-encryption=sev0,vmport=off \
  -object memory-backend-memfd,id=ram1,size=6G,share=true,prealloc=false \
  -machine memory-backend=ram1 \
  -object sev-snp-guest,id=sev0,cbitpos=47,reduced-phys-bits=1,kernel-hashes=on \
  -kernel ./vmlinuz \
  -append "root=/dev/vda1 console=ttyS0" \
  -bios /usr/share/ovmf/OVMF.amdsev.fd
```

Here we are configuring the VM to use 6GB of encrypted memory:

- `cbitpos` and `reduced-phys-bits` have to be `47` and `1` respectively for all EPYC CPUs.
- `kernel-hashes=on` makes sure the kernel, initramfs and command line will be measured during the launch. It can be disabled but if enabled, the kernel needs to be provided with `-kernel`
- `OVMF.amdsev.fd` is a specific version of EDK II.

For more details about these parameters, refer to QEMU documentation pages for invocation and AMD SEV.

### Guest configuration

On the guest side, Ubuntu 24.04 LTS and newer fully support AMD SEV-SNP. You can download the disk image and kernel for your VM from `cloud-images.ubuntu.com`. The latest image and kernel can be found here:

- https://cloud-images.ubuntu.com/releases/noble/release/ubuntu-24.04-server-cloudimg-amd64.img
- https://cloud-images.ubuntu.com/releases/noble/release/unpacked/

Once the VM is launched, install `linux-generic` to get the `sev-guest` module and insert it with `modprobe sev-guest`. This will create a new character device on the guest that can be used to generate attestation reports from the TEE: `/dev/sev-guest`. Finally, you can use AMD's `snpguest` to generate an attestation report that can be used for a remote attestation:

```
sudo ./snpguest report --random attestation-report.bin request-file.txt
```

- *Create VMs with Multipass*
- *Create cloud image VMs with UVtool*
- *QEMU*
- *Confidential Computing with AMD*

## VM tooling

### Libvirt

The libvirt library is used to interface with many different virtualisation technologies. Before getting started with libvirt it is best to make sure your hardware supports the necessary virtualisation extensions for Kernel-based Virtual Machine (KVM). To check this, enter the following from a terminal prompt:

```
kvm-ok
```

A message will be printed informing you if your CPU *does* or *does not* support hardware virtualisation.

> **Note**
>
> On many computers with processors supporting hardware-assisted virtualisation, it is necessary to first activate an option in the BIOS to enable it.

### Virtual networking

There are a few different ways to allow a virtual machine access to the external network. The default virtual network configuration includes **bridging** and **iptables** rules implementing **usermode** networking, which uses the SLiRP protocol. Traffic is NATed through the host interface to the outside network.

To enable external hosts to directly access services on virtual machines, a different type of *bridge* than the default needs to be configured. This allows the virtual interfaces to connect to the outside network through the physical interface, making them appear as normal hosts to the rest of the network.

There is a great example of how to configure a bridge and combine it with libvirt so that guests will use it at the netplan.io documentation.

### Install libvirt

To install the necessary packages, from a terminal prompt enter:

```
sudo apt update
sudo apt install qemu-kvm libvirt-daemon-system
```

After installing `libvirt-daemon-system`, the user that will be used to manage virtual machines needs to be added to the *libvirt* group. This is done automatically for members of the *sudo* group, but needs to be done in addition for anyone else that should access system-wide libvirt resources. Doing so will grant the user access to the advanced networking options.

In a terminal enter:

```
sudo adduser $USER libvirt
```

> **Note**
>
> If the chosen user is the current user, you will need to log out and back in for the new group membership to take effect.

You are now ready to install a *Guest* operating system. Installing a virtual machine follows the same process as installing the operating system directly on the hardware.

You will need **one** of the following:

- A way to automate the installation.
- A keyboard and monitor attached to the physical machine.
- To use cloud images which are meant to self-initialise (see *Multipass* and *UVTool*).

In the case of virtual machines, a *Graphical User Interface (GUI)* is analogous to using a physical keyboard and mouse on a real computer. Instead of installing a GUI the `virt-viewer` or `virt-manager` application can be used to connect to a virtual machine's console using VNC. See *Virtual Machine Manager / Viewer* for more information.

### Virtual machine management

The following section covers `virsh`, a virtual machine management tool that is a part of libvirt. But there are other tools available at different levels of complexities and feature-sets, like:

- *Multipass*
- *UVTool*
- *virt-* tools*

- OpenStack

### Manage VMs with `virsh`

There are several utilities available to manage virtual machines and libvirt. The `virsh` utility can be used from the command line. Some examples:

- To list running virtual machines:

```
virsh list
```

- To start a virtual machine:

```
virsh start <guestname>
```

- Similarly, to start a virtual machine at boot:

```
virsh autostart <guestname>
```

- Reboot a virtual machine with:

```
virsh reboot <guestname>
```

- The **state** of virtual machines can be saved to a file in order to be restored later. The following will save the virtual machine state into a file named according to the date:

```
virsh save <guestname> save-my.state
```

  Once saved, the virtual machine will no longer be running.

- A saved virtual machine can be restored using:

```
virsh restore save-my.state
```

- To shut down a virtual machine you can do:

```
virsh shutdown <guestname>
```

- A CD-ROM device can be mounted in a virtual machine by entering:

```
virsh attach-disk <guestname> /dev/cdrom /media/cdrom
```

- To change the definition of a guest, `virsh` exposes the domain via:

```
virsh edit <guestname>
```

This will allow you to edit the XML representation that defines the guest. When saving, it will apply format and integrity checks on these definitions.

Editing the XML directly certainly is the most powerful way, but also the most complex one. Tools like *Virtual Machine Manager / Viewer* can help inexperienced users to do most of the common tasks.

**Note**

If `virsh` (or other `vir*` tools) connect to something other than the default `qemu-kvm`/system hypervisor, one can find alternatives for the `--connect` option using `man virsh` or the libvirt docs.

#### `system` and `session` scope

You can pass connection strings to `virsh` - as well as to most other tools for managing virtualisation.

```
virsh --connect qemu:///system
```

There are two options for the connection.

- `qemu:///system` - connect locally as **root** to the daemon supervising QEMU and KVM domains
- `qemu:///session` - connect locally as a **normal user** to their own set of QEMU and KVM domains

The *default* was always (and still is) `qemu:///system` as that is the behavior most users are accustomed to. But there are a few benefits (and drawbacks) to `qemu:///session` to consider.

`qemu:///session` is per user and can – on a multi-user system – be used to separate the people. Most importantly, processes run under the permissions of the user, which means no permission struggle on the just-downloaded image in your `$HOME` or the just-attached USB-stick.

On the other hand it can't access system resources very well, which includes network setup that is known to be hard with `qemu:///session`. It falls back to SLiRP networking which is functional but slow, and makes it impossible to be reached from other systems.

`qemu:///system` is different in that it is run by the global system-wide libvirt that can arbitrate resources as needed. But you might need to `mv` and/or `chown` files to the right places and change permissions to make them usable.

Applications will usually decide on their primary use-case. Desktop-centric applications often choose `qemu:///session` while most solutions that involve an administrator anyway continue to default to `qemu:///system`.

**See also**

There is more information about this topic in the libvirt FAQ and this blog post about the topic.

### Migration

There are different types of migration available depending on the versions of libvirt and the hypervisor being used. In general those types are:

- Offline migration
- Live migration

- Postcopy migration

There are various options to those methods, but the entry point for all of them is `virsh migrate`. Read the integrated help for more detail.

```
virsh migrate --help
```

Some useful documentation on the constraints and considerations of live migration can be found at the Ubuntu Wiki.

### CPU model and topology

libvirt abstracts CPU configuration and provides several options to specify the VM CPU model in the domain definition:

- custom
- host-model
- host-passthrough
- maximum

While most of thee modes are straightforward, the behavior of `host-model` is more subtle and the source of many common misunderstandings. There is no direct translation of `host-model` into a single QEMU `-cpu` argument. Instead, libvirt selects a baseline CPU model and appends a list of features:

```
# example snippet from qemu command line generated by libvirt
... -cpu Haswell-noTSX-IBRS,vmx=on,pdcm=off,...,vmx-entry-load-efer=on,vmx-eptp-
switching=on ...
```

Because libvirt can not include every known CPU model and variant, it chooses the model that shares the largest set of features with the host's physical CPU and then lists the remaining features explicitly. In many cases, libvirt therefore can not detect the exact host CPU model. At first this may seem like a flaw, but in practice, it is not necessary to know the exact model.

For example, running `virsh capabilities` on a host with an Intel **Broadwell CPU** may produce the following output, where libvirt uses **Haswell-noTSX-IBRS** as the baseline:

```
<capabilities>
  <host>
    <uuid>30303837-3831-584d-5135-323430354a38</uuid>
    <cpu>
      <arch>x86_64</arch>
      <model>Haswell-noTSX-IBRS</model>
      <vendor>Intel</vendor>
      <microcode version='73'/>
      <signature family='6' model='63' stepping='2'/>
      <counter name='tsc' frequency='2397195000' scaling='no'/>
      <topology sockets='1' dies='1' cores='6' threads='2'/>
      <maxphysaddr mode='emulate' bits='46'/>
      <feature name='vme'/>
      <feature name='ds'/>
```

(continues on next page)

(continued from previous page)

```
      <feature name='acpi'/>
      <feature name='ss'/>
```

This mismatch between the baseline model reported by libvirt and the actual physical CPU model is not a bug and can safely be ignored.

For more details, refer to the upstream documentation.

### Host CPU capabilities

### Device passthrough/hotplug

If you want to always pass through a device rather than using the hotplugging method described here, add the XML content of the device to your static guest XML representation via `virsh edit <guestname>`. In that case, you won't need to use *attach/detach*. There are different kinds of passthrough, and the types available to you depend on your hardware and software setup.

- USB *hotplug*/passthrough
- VF hotplug/Passthrough

Both kinds are handled in a very similar way and while there are various way to do it (e.g. also via QEMU monitor), driving such a change via libvirt is recommended. That way, libvirt can try to manage all sorts of special cases for you and also somewhat masks version differences.

In general, when driving hotplug via libvirt, you create an XML snippet that describes the device just as you would do in a static guest description. A USB device is usually identified by vendor/product ID:

```
<hostdev mode='subsystem' type='usb' managed='yes'>
  <source>
    <vendor id='0x0b6d'/>
    <product id='0x3880'/>
  </source>
</hostdev>
```

Virtual functions are usually assigned via their PCI ID (domain, bus, slot, and function).

```
<hostdev mode='subsystem' type='pci' managed='yes'>
  <source>
    <address domain='0x0000' bus='0x04' slot='0x10' function='0x0'/>
  </source>
</hostdev>
```

**Note**

Getting the virtual function in the first place is very device-dependent and can, therefore, not be fully covered here. But in general, it involves setting up an *IOMMU*, registering via VFIO and sometimes requesting a number of VFs.

Here is an example of configuring a ppc64el system to create four VFs on a device:

```
$ sudo modprobe vfio-pci
# identify device
$ lspci -n -s 0005:01:01.3
0005:01:01.3 0200: 10df:e228 (rev 10)
# register and request VFs
$ echo 10df e228 | sudo tee /sys/bus/pci/drivers/vfio-pci/new_id
$ echo 4 | sudo tee /sys/bus/pci/devices/0005\:01\:00.0/sriov_numvfs
```

You then attach or detach the device via libvirt by relating the guest with the XML snippet.

```
virsh attach-device <guestname> <device-xml>
# Use the Device in the Guest
virsh detach-device <guestname> <device-xml>
```

### Access QEMU Monitor via libvirt

The QEMU Monitor is the way to interact with QEMU/KVM while a guest is running. This interface has many powerful features for experienced users. When running under libvirt, the monitor interface is bound by libvirt itself for management purposes, but a user can still run QEMU monitor commands via libvirt. The general syntax is `virsh qemu-monitor-command [options] [guest] 'command'`.

Libvirt covers most use cases needed, but if you ever want/need to work around libvirt or want to tweak very special options you can e.g. add a device as follows:

```
virsh qemu-monitor-command --hmp focal-test-log 'drive_add 0 if=none,file=/var/
lib/libvirt/images/test.img,format=raw,id=disk1'
```

The monitor is a power tool, especially for debugging purposes. For example, one can use the monitor to show the guest registers:

```
$ virsh qemu-monitor-command --hmp y-ipns 'info registers'

RAX=00ffffc000000000 RBX=ffff8f0f5d5c7e48 RCX=0000000000000000
RDX=ffffea00007571c0
RSI=0000000000000000 RDI=ffff8f0fdd5c7e48 RBP=ffff8f0f5d5c7e18
RSP=ffff8f0f5d5c7df8
[...]
```

### Huge pages

Let's start with a summary of the allocation and structuring of memory in an OS, and its relationship to transparent huge pages and *huge pages*.

When you launch an application, the OS allocates virtual memory to it as a range of virtual addresses. The virtual memory is fake; it only allows the application to think it has more memory than what's physically available.

However, the CPU architecture will determine the amount of virtual memory allocated to the application; 64-bit CPUs support 16 EB, whereas 32-bit CPUs support 4 GB. x86_64, however, typically supports only 256 TB of virtual memory.

The OS splits the virtual memory into pages (4 KB each). A page contains several virtual addresses (1 byte each). These pages contain different parts of the application, like the code, data, stack, and heap. The OS maps these pages to real memory (RAM) because the virtual memory is a fake.

Although the OS allocates the virtual memory, the range of virtual addresses is generated by the CPU whenever the application accesses memory (e.g., reading a variable, fetching an instruction, or writing data). The Memory Management Unit (MMU) receives these virtual addresses and uses the page table to convert them into physical addresses.

With a 4 KB page size in the table, a large application having 200 MB in size, for example, could have thousands of pages. So, constantly looking up the page table in RAM would be slow. To briefly fix this, the CPU uses a *Translation Lookaside Buffer (TLB)* to cache recent page table entries to speed up memory access; however, the TLB can only hold a certain number of entries.

To reduce this overhead, you can use huge pages, which increase the page size from 4 KB to larger sizes (e.g., 2 MB or 1 GB). This reduces the number of page table entries in the TLB, making lookups faster.

Huge pages must frequently be *pre-allocated* on the host for Libvirt to *map the VMs memory to the host*. This is because VMs rely on two layers of address translation — one for the VM and one for the host - so expect memory lookups to be CPU-intensive. Since huge pages can be configured on Libvirt, the page table entries are reduced, hence, memory access from the VM speeds up.

It's now clear how huge pages can lessen page table entries and TLB overhead.

Huge pages can have some disadvantages too, as they frequently require manual setup. To address this, transparent huge pages are used to manage pages dynamically.

The dynamic page resizing can also be an issue when using libvirt since huge pages have to be pre-allocated. So, if it is clear that using huge pages is preferred, then making them explicit usually has some gains.

While huge pages are admittedly harder to manage (especially later in the system's lifetime if memory is fragmented), they provide a useful boost, especially for rather large guests.

> **Tip**
>
> When using device passthrough on very large guests, there is an extra benefit of using huge pages, as it is faster to do the initial memory clear on the VFIO *DMA* pin.

### Huge page allocation

Huge pages come in different sizes. A *normal* page is usually 4k and huge pages are either 2M or 1G, but depending on the architecture, other options are possible.

The simplest yet least reliable way to allocate some huge pages is to just echo a value to `sysfs`:

```
echo 256 | sudo tee /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages
```

Be sure to re-check if it worked:

```
$ cat /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages

256
```

There one of these sizes is the “default huge page size”, which will be used in the auto-mounted `/dev/hugepages`. Changing the default size requires a reboot and is set via default_hugepagesz.

You can check the current default size:

```
$ grep Hugepagesize /proc/meminfo

Hugepagesize:       2048 kB
```

But there can be more than one at the same time – so it’s a good idea to check:

```
$ tail /sys/kernel/mm/hugepages/hugepages-*/nr_hugepages`
==> /sys/kernel/mm/hugepages/hugepages-1048576kB/nr_hugepages <==
0
==> /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages <==
2
```

And even that could – on bigger systems – be further split per Numa node.

One can allocate huge pages at boot or runtime, but due to fragmentation there are no guarantees it works later. The kernel documentation lists details on both ways.

Huge pages need to be allocated by the kernel as mentioned above, but to be consumable, they also have to be mounted. By default, `systemd` will make `/dev/hugepages` available for the default huge page size.

Feel free to add more mount points if you need different sized ones. An overview can be queried with hugeadm:

```
$ apt install libhugetlbfs-bin
$ hugeadm --list-all-mounts

Mount Point          Options
/dev/hugepages       rw,relatime,pagesize=2M
```

A one-stop info for the overall huge page status of the system can be reported with:

```
hugeadm --explain
```

### Huge page usage in libvirt

With the above in place, libvirt can map guest memory to huge pages. In a guest definition add the most simple form of:

```
<memoryBacking>
  <hugepages/>
</memoryBacking>
```

That will allocate the huge pages using the default huge page size from an autodetected mount point. For more control, e.g. how memory is spread over Numa nodes or which page size to use, check out the details at the libvirt docs.

### Controlling addressing bits

This is a topic that rarely matters on a single computer with virtual machines for generic use; libvirt will automatically use the hypervisor default, which in the case of QEMU is 40 bits. This default aims for compatibility since it will be the same on all systems, which simplifies migration between them and usually is compatible even with older hardware.

However, it can be very important when driving more advanced use cases. If one needs bigger guest sizes with more than a terabyte of memory then controlling the addressing bits is crucial.

### -hpb machine types

Since Ubuntu 18.04, the QEMU in Ubuntu has provided special machine-types. These include machine types like `pc-q35-jammy` or `pc-i440fx-jammy`, but with a `-hpb` suffix. The "*HPB*" abbreviation stands for "host-physical-bits", which is the QEMU option that this represents.

For example, by using `pc-q35-jammy-hpb`, the guest would use the number of physical bits that the Host CPU has available.

Providing the configuration that a guest should use more address bits as a machine type has the benefit that many higher level management stacks like for example openstack, are already able to control it through libvirt.

One can check the bits available to a given CPU via the procfs:

```
$ cat /proc/cpuinfo | grep '^address sizes'
...
# an older server with a E5-2620
address sizes   : 46 bits physical, 48 bits virtual
# a laptop with an i7-8550U
address sizes   : 39 bits physical, 48 bits virtual
```

### maxphysaddr guest configuration

Since libvirt version 8.7.0 (>= Ubuntu 22.10 Lunar), `maxphysaddr` can be controlled via the CPU model and topology section of the guest configuration. If one needs just a large guest, like before when using the `-hpb` types, all that is needed is the following libvirt guest xml configuration:

```
  <maxphysaddr mode='passthrough' />
```

Since libvirt 9.2.0 and 9.3.0 (>= Ubuntu 23.10 Mantic), an explicit number of emulated bits or a limit to the passthrough can be specified. Combined, this pairing can be very useful for computing clusters where the CPUs have different hardware physical addressing bits. Without these features, guests could be large, but potentially unable to migrate freely between all nodes since not all systems would support the same amount of addressing bits.

But now, one can either set a fixed value of addressing bits:

```
  <maxphysaddr mode='emulate' bits='42'/>
```

Or use the best available by a given hardware, without going over a certain limit to retain some compute node compatibility.

```
  <maxphysaddr mode='passthrough' limit='41/>
```

### AppArmor isolation

By default, libvirt will spawn QEMU guests using AppArmor isolation for enhanced security. The AppArmor rules for a guest will consist of multiple elements:

- A static part that all guests share => `/etc/apparmor.d/abstractions/libvirt-qemu`
- A dynamic part created at guest start time and modified on hotplug/unplug => `/etc/apparmor.d/libvirt/libvirt-f9533e35-6b63-45f5-96be-7cccc9696d5e.files`

Of the above, the former is provided and updated by the `libvirt-daemon` package, and the latter is generated on guest start. Neither of the two should be manually edited. They will, by default, cover the vast majority of use cases and work fine. But there are certain cases where users either want to:

- Further lock down the guest, e.g. by explicitly denying access that usually would be allowed.
- Open up the guest isolation. Most of the time this is needed if the setup on the local machine does not follow the commonly used paths.

To do so there are two files. Both are local overrides which allow you to modify them without getting them clobbered or command file prompts on package upgrades.

- `/etc/apparmor.d/local/abstractions/libvirt-qemu` This will be applied to every guest. Therefore it is a rather powerful (if blunt) tool. It is a quite useful place to add additional deny rules.
- `/etc/apparmor.d/local/usr.lib.libvirt.virt-aa-helper` The above-mentioned *dynamic part* that is individual per guest is generated by a tool called `libvirt.virt-aa-helper`. That is under AppArmor isolation as well. This is most commonly used if you want to use uncommon paths as it allows one to have those uncommon paths in the guest XML (see `virsh edit`) and have those paths rendered to the per-guest dynamic rules.

### Sharing files between Host<->Guest

To be able to exchange data, the memory of the guest has to be allocated as “shared”. To do so you need to add the following to the guest config:

```
<memoryBacking>
  <access mode='shared'/>
</memoryBacking>
```

For performance reasons (it helps `virtiofs`, but also is generally wise to consider) it is recommended to use huge pages which then would look like:

```
<memoryBacking>
  <hugepages>
    <page size='2048' unit='KiB'/>
  </hugepages>
  <access mode='shared'/>
</memoryBacking>
```

In the guest definition, one then can add `filesystem` sections to specify host paths to share with the guest. The *target dir* is a bit special as it isn't really a directory – instead, it is a *tag* that in the guest can be used to access this particular `virtiofs` instance.

```
<filesystem type='mount' accessmode='passthrough'>
  <driver type='virtiofs'/>
  <source dir='/var/guests/h-virtiofs'/>
  <target dir='myfs'/>
</filesystem>
```

And in the guest, this can now be used based on the tag `myfs` like:

```
sudo mount -t virtiofs myfs /mnt/
```

Compared to other Host/Guest file sharing options – commonly Samba, NFS, or 9P – `virtiofs` is usually much faster and also more compatible with usual file system semantics.

See the libvirt domain/filesystem documentation for further details on these.

**Note**

While `virtiofs` works with >=20.10 (Groovy), with >=21.04 (Hirsute) it became more comfortable, especially in small environments (no hard requirement to specify guest Numa topology, no hard requirement to use huge pages). If needed to set up on 20.10 or just interested in those details - the libvirt knowledge-base about virtiofs holds more details about these.

### Resources

- See the KVM home page for more details.
- For more information on libvirt see the libvirt home page.
  - XML configuration of domains and storage are the most often used libvirt reference.
- Another good resource is the Ubuntu Wiki KVM page.
- For basics on how to assign VT-d devices to QEMU/KVM, please see the linux-kvm page.
- Introduction to Memory Management in Linux

### Virtual Machine Manager

The Virtual Machine Manager, through the `virt-manager` package, provides a graphical user interface (GUI) for managing local and remote virtual machines. In addition to the `virt-manager` utility itself, the package also contains a collection of other helpful tools like `virt-install`, `virt-clone` and `virt-viewer`.

### Install virt-manager

To install `virt-manager`, enter:

```
sudo apt install virt-manager
```

Since `virt-manager` requires a *Graphical User Interface (GUI)* environment we recommend installing it on a workstation or test machine instead of a production server. To connect to the local libvirt service, enter:

```
virt-manager
```

You can connect to the libvirt service running on another host by entering the following in a terminal prompt:

```
virt-manager -c qemu+ssh://virtnode1.mydomain.com/system
```

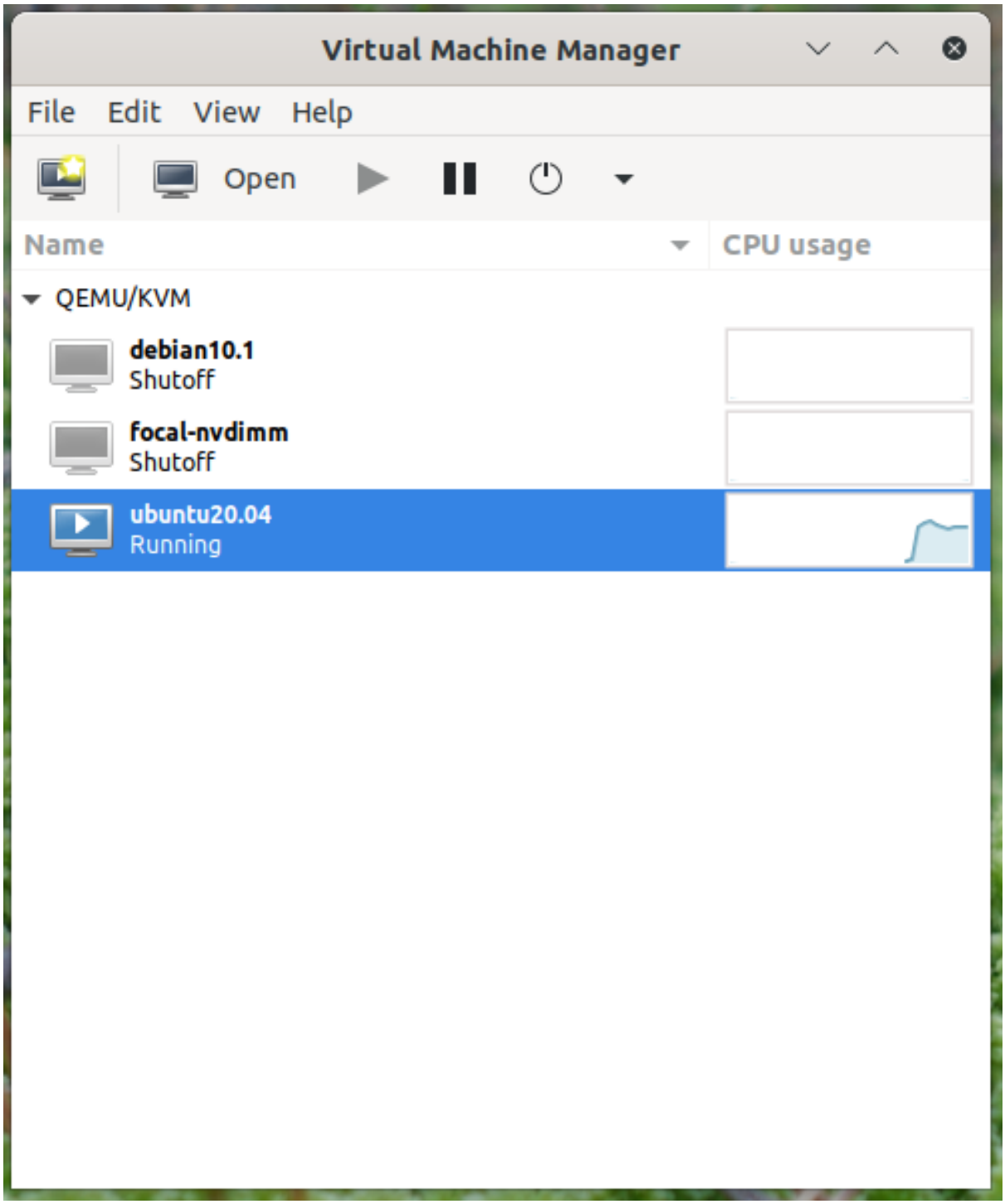
**Note**

The above example assumes that SSH connectivity between the management system and the target system has already been configured, and uses **SSH keys** for authentication. SSH keys are needed because libvirt sends the password prompt to another process. See our guide on OpenSSH for details on *how to set up SSH keys*.

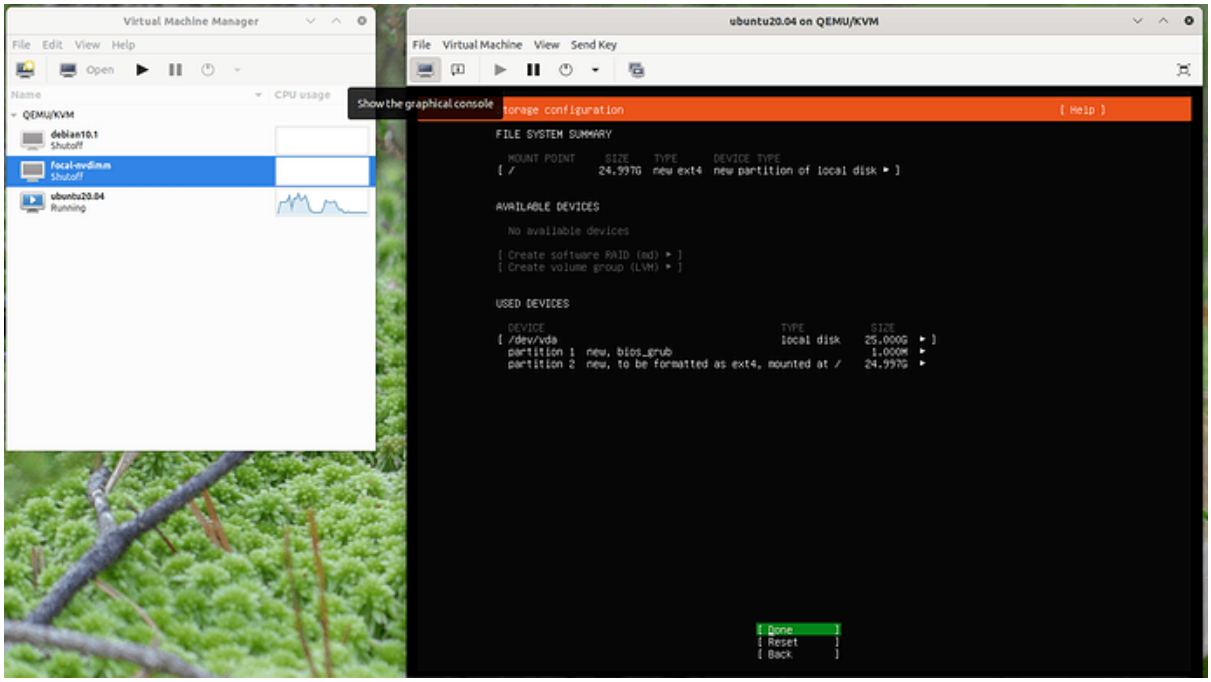
### Use virt-manager to manage guests

### Guest lifecycle

When using `virt-manager`, it is always important to know the context you're looking at. The main window initially lists only the currently-defined guests. You'll see their **name**, **state** (e.g., 'Shutoff' or 'Running') and a small chart showing the **CPU usage**.



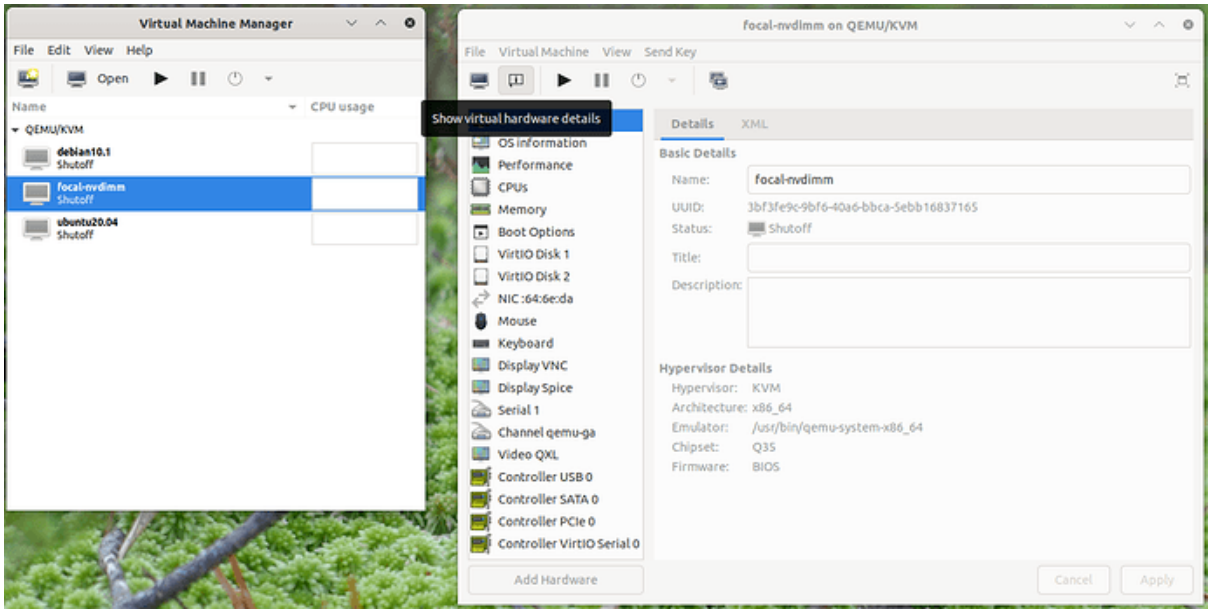

In this context, there isn't much to do except start/stop a guest. However, by double-clicking on a guest or by clicking the **Open** button at the top of the window, you can see the guest itself. For a running guest, that includes the guest's main-console/virtual-screen output.




If you are deeper in the guest configuration, clicking on “Show the graphical console” in the top left of the guest’s window will get you back to this output.

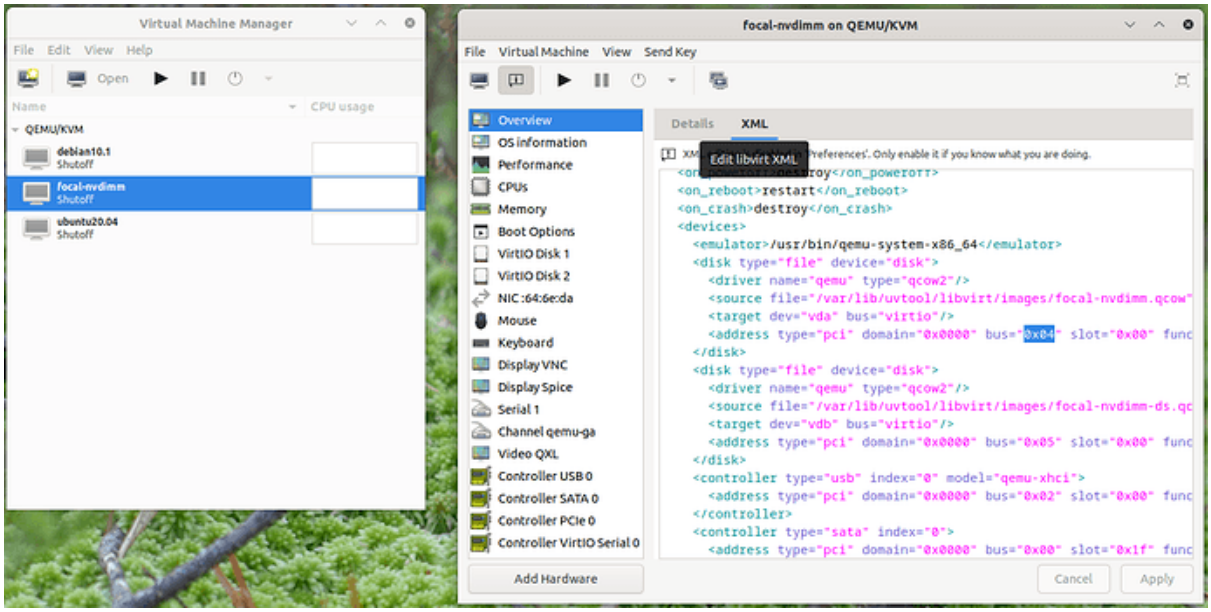
## Guest modification

`virt-manager` provides a handy, GUI-assisted way to edit guest definitions. To do so, the per-guest context view will have “Show virtual hardware details” at the top of the guest window. Here, you can edit the virtual hardware of the guest, which will alter the guest representation behind the scenes.




The UI edit ability is limited to the features known to (and supported by) that GUI feature. Not only does libvirt grow features faster than `virt-manager` can keep up – adding every feature would also overload the UI and render it unusable.

To strike a balance between the two, there also is the XML view which can be reached via the “Edit libvirt XML” button.




By default, this will be read-only and you can see what the UI-driven actions have changed. You can allow read-write access in this view via the “Preferences”. This is the same content that the `virsh edit` of the *libvirt-client* exposes.

### Virtual Machine Viewer (virt-viewer)

The Virtual Machine Viewer application, through `virt-viewer`, allows you to connect to a virtual machine’s console like `virt-manager` does, but reduced to the GUI functionality. `virt-viewer` requires a GUI to interface with the virtual machine.

### Install virt-viewer

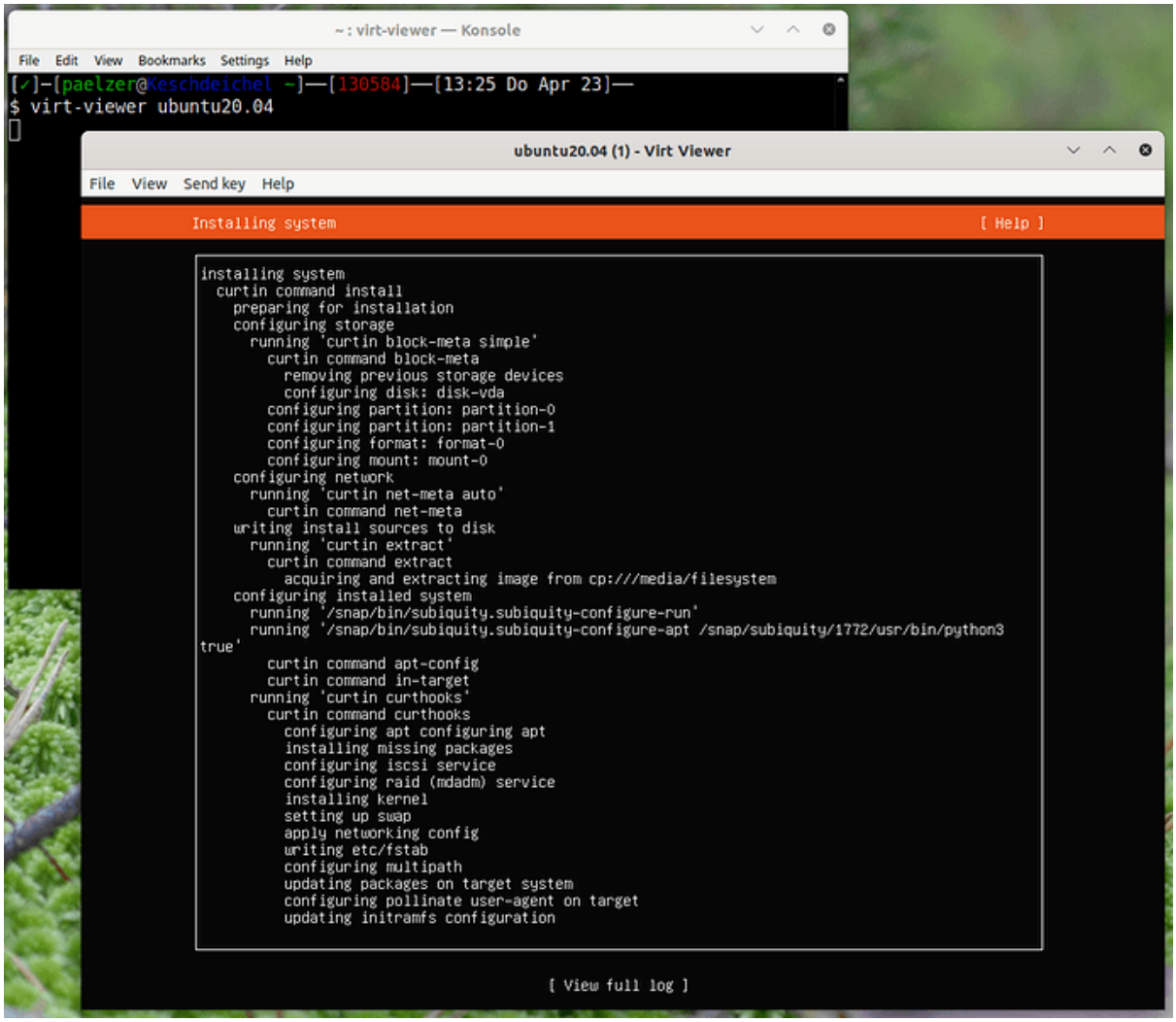
If `virt-viewer` is not already installed, you can install it from the terminal with the following command:

```
sudo apt install virt-viewer
```

Once a virtual machine is installed and running you can connect to the virtual machine’s console by using:

```
virt-viewer <guestname>
```

The UI will show a window representing the virtual screen of the guest, just like with `virt-manager` above, but without the extra buttons and features around it.




Similarly to `virt-manager`, `virt-viewer` can also connect to a remote host using SSH with key authentication:

```
virt-viewer -c qemu+ssh://virtnode1.mydomain.com/system <guestname>
```

Be sure to replace `web_devel` with the appropriate virtual machine name.

If configured to use a **bridged** network interface, you can also set up SSH access to the virtual machine.

### virt-install

`virt-install` is part of the `virtinst` package. It can help with installing classic ISO-based systems and provides a CLI for the most common options needed to do so.

### Install virt-install

To install `virt-install`, if it is not installed already, run the following from a terminal prompt:

```
sudo apt install virtinst
```

There are several options available when using `virt-install`. For example:

```
virt-install \
   --name web_devel \
   --ram 8192 \
   --disk path=/home/doug/vm/web_devel.img,bus=virtio,size=50 \
   --cdrom focal-desktop-amd64.iso \
   --network network=default,model=virtio \
   --graphics vnc,listen=0.0.0.0 \
   --noautoconsole \
   --hvm \
   --vcpus=4
```

There are many more arguments that can be found in the `virt-install(1)` manpage. However, here is an explanation of arguments used in the example above, one by one:

- `--name web_devel`: The name of the new virtual machine will be `web_devel`.
- `--ram 8192`: Specifies the amount of memory the virtual machine will use (in megabytes).
- `--disk path=/home/doug/vm/web_devel.img,bus=virtio,size=50`: Indicates the path to the virtual disk which can be a file, partition, or logical volume. In this example a file named `web_devel.img` in the current user's directory, with a size of 50 gigabytes, and using `virtio` for the disk bus. Depending on the disk path, `virt-install` may need to run with elevated privileges.
- `--cdrom focal-desktop-amd64.iso`: File to be used as a virtual CD-ROM. The file can be either an ISO file or the path to the host's CD-ROM device.
- `--network`: Provides details related to the VM's network interface. Here the default network is used, and the interface model is configured for `virtio`.
- `--graphics vnc,listen=0.0.0.0`: Exports the guest's virtual console using VNC and on all host interfaces. Typically servers have no GUI, so another GUI-based computer on the Local Area Network (LAN) can connect via VNC to complete the installation.
- `--noautoconsole` Will not automatically connect to the virtual machine's console.
- `--hvm`: creates a fully virtualised guest.
- `--vcpus=4`: allocate 4 virtual CPUs.

After launching `virt-install`, you can connect to the virtual machine's console either locally using a GUI (if your server has a GUI), or via a remote VNC client from a GUI-based computer.

#### `virt-clone`

The `virt-clone` application can be used to copy one virtual machine to another. For example:

```
virt-clone --auto-clone --original focal
```

Options used:

- `--auto-clone`: To have `virt-clone` create guest names and disk paths on its own.
- `--original`: Name of the virtual machine to copy.

You can also use the `-d` or `--debug` option to help troubleshoot problems with `virt-clone`.

Replace `focal` with the appropriate virtual machine names for your case.

**Caution**

Please be aware that this is a full clone, therefore any secrets, keys, and for example, `/etc/machine-id`, will be shared. This will cause issues with security and anything that needs to identify the machine like *DHCP*. You most likely want to edit those afterward and de-duplicate them as needed.

### Resources

- See the KVM home page for more details.
- For more information on libvirt see the libvirt home page
- The Virtual Machine Manager site has more information on `virt-manager` development.

### How to enable nested virtualisation

**Important**

Nested virtualisation is enabled by default on Ubuntu. If you are using Ubuntu, it's unlikely that you will need to manually enable the feature. If you check (using the steps below) and discover that nested virtualisation is enabled, then you will not need to do anything further.

There may be use cases where you need to enable nested virtualisation so that you can deploy instances inside other instances. The sections below explain how to check if nested virtualisation is enabled/available and how to enable it if that is not the case. Bear in mind that currently nested virtualisation is only supported in Ubuntu on `x86` machine architecture.

### Check if nested virtualisation is enabled

Check if the required kernel module for your CPU is already loaded. Hosts with Intel CPUs require the `kvm_intel` module while *AMD* hosts require `kvm_amd` instead:

```
$ lsmod | grep -i kvm
kvm_intel              204800  0
kvm                   1347584  1 kvm_intel
```

### If the module is loaded

If the module is already loaded, you can check if nested virtualisation is enabled by running the following command:

```
cat /sys/module/<module>/parameters/nested
```

As an example for AMD hosts:

```
$ cat /sys/module/kvm_amd/parameters/nested
1
```

If the output is either `1` or `Y` then nested virtualisation is enabled and you will not need to manually enable the feature (this should be the case for Ubuntu users).

### If the module is not loaded

If the module your host requires is not loaded, you can load it using `modprobe` in combination with the property `nested=1` to enable nested virtualisation, as shown below for Intel hosts:

```
modprobe kvm-intel nested=1
```

Or as follows for AMD hosts:

```
modprobe kvm-amd nested=1
```

### Enable nested virtualisation

If the above checks indicate that nested virtualisation is not enabled, you can follow the below steps to enable it.

- Create a file in `/etc/modprobe.d` -e.g., `/etc/modprobe.d/kvm.conf`- and add the line `options kvm-intel nested=1` to that file (replace `kvm-intel` with `kvm-amd` for AMD hosts).
- Reload the kernel module to apply the changes:

```
sudo modprobe -r <module>
```

Example for Intel hosts:

```
sudo modprobe -r kvm-intel
```

- You should now be able to see nested virtualisation enabled:

Example for Intel hosts:

```
$ cat /sys/module/kvm_intel/parameters/nested
Y
```

### Check and enable nested virtualisation inside an instance

Once the host is ready to use nested virtualisation, it is time to check if the guest instance can run additional nested VMs inside it.

To determine if an instance can host another instance on top, run the below command within the instance:

```
egrep "svm|vmx" /proc/cpuinfo
```

If any of these are present in the output (depending on whether the host is AMD or Intel respectively), then virtualisation is available in that instance. If this is not the case you will need to edit the instance CPU settings:

- Shut down the instance
- Edit the instance XML definition file executing: `virsh edit <instance>`
- Search the `cpu mode` parameter in and set its value to either `host-model` or `host-passthrough` (details about these modes can be found here).

  Sample `cpu mode` parameter in XML with nested virtualisation:

```
<cpu mode='host-model' check='partial'/>
```

- Save the modifications and start the instance

### Limitations of nested virtualisation

Nested virtualisation has some key limitations you'd need to consider, namely that not all KVM features will be available for instances running nested VMs, and actions such as migrating or saving the parent instance will not be possible until the nested instance is stopped.

- *How to use the libvirt library with virsh*
- *How to use virt-manager and other virt* tools*
- *How to enable nested virtualisation*

## Ubuntu in other virtual environments

### How to set up Ubuntu on Hyper-V

Hyper-V is a native type 1 hypervisor developed by Microsoft for the Windows family of operating systems, similar to Xen or VMWare *ESXi*. It was first released for Windows Server in 2008, and has been available without additional charge since Windows Server 2012 and Windows 8.

Hyper-V allows Ubuntu to be run in parallel or in isolation on Windows operating systems. There are several use-cases for running Ubuntu on Hyper-V:

- To introduce Ubuntu in a Windows-centric IT environment.
- To have access to a complete Ubuntu desktop environment without dual-booting a PC.
- To use Linux software on Ubuntu that is not yet supported on the Windows Subsystem for Linux.

### Hyper-V system requirements

The following are typical system requirements for Hyper-V:

- A 64-bit processor with Second Level Address Translation (SLAT)
- CPU support for virtualization extensions and virtualization enabled in the system BIOS/EFI
- Minimum of 4 GB of memory, recommended 8 GB
- Minimum of 5 GB of disk space, recommended 15 GB

### Install Hyper-V

Our first step in enabling Ubuntu is to install Hyper-V, which can be used on the Windows 11 Pro, Enterprise, Education, and Server operating systems.

Hyper-V is not included in Windows 11 Home, which would need to be upgraded to Windows 11 Pro.

### Install Hyper-V graphically

1. Right click on the Windows Start button and select 'Apps and Features'.
2. Select 'Programs and Features' under Related Settings.
3. Select 'Turn Windows Features on or off'.
4. Select 'Hyper-V' and click OK.
5. Restart when prompted.

### Install Hyper-V using PowerShell

1. Open a PowerShell console as Administrator.
2. Run the following command:

```
Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Hyper-V -All
```

3. Restart when prompted.

### Install Ubuntu on Hyper-V

There are two main methods for installing Ubuntu on Hyper-V depending on your use case. Read each of the descriptions of the following methods and then determine the best for your situation.

### Using Quick Create

The recommended method is to use the curated Ubuntu image from the Hyper-V Quick Create Gallery. This is ideal for desktop development on Ubuntu and for users interested in running a complete Ubuntu desktop environment. The Ubuntu image from the Quick Create Gallery includes pre-configured features, such as clipboard sharing, dynamic resolution display, and shared folders.

1. Enable Hyper-V as described above.
2. Open 'Hyper-V Manager' by either:
   - Selecting the Windows Start button, then -> Expanding the 'Windows Administrative Tools' folder -> Selecting 'Hyper-V Manager'

   or

   - Selecting the Windows key, then -> typing 'Hyper-V' -> selecting 'Hyper-V Manager'

   In the future, the Quick Create tool can be accessed directly using the above methods, but it is useful to know where Hyper-V Manager is because it is what you will use to manage your Ubuntu VM.

3. On the ‘Actions’ pane select ‘Quick Create’ and the Quick Create tool will open.
4. Select a version of Ubuntu from the versions on the list. A build of the most recent LTS version of Ubuntu and the most recent interim release are provided.
    - The **LTS version** is recommended if you are developing for Ubuntu Server or an enterprise environment.
    - The **interim release** is recommended if you would like to use the latest versions of software in Ubuntu.
5. Select ‘Create Virtual Machine’ and wait for the VM image to be downloaded.
6. Select ‘Connect’ to open a connection to your VM.
7. Select ‘Start’ to run your VM.
8. Complete the final stages of the Ubuntu install, including username selection.

### Using an Ubuntu CD image

It is possible to install Ubuntu on Hyper-V using a CD image ISO. This is useful if you are running Ubuntu Server and do not need an enhanced desktop experience. Note that the enhanced features of the Quick Create images are not enabled by default when you perform a manual install from an ISO.

1. Download an Ubuntu ISO from an official Ubuntu source.
2. Install Hyper-V as described above.
3. Open ‘Hyper-V Manager’ by either:
    - Selecting the Windows Start button, then -> Expanding the ‘Windows Administrative Tools’ folder -> Selecting ‘Hyper-V Manager’

      or

    - Selecting the Windows key, then -> Typing ‘Hyper-V’ -> Selecting ‘Hyper-V Manager’
4. On the ‘Actions’ pane click ‘Quick Create’ and the Quick Create tool will open.
5. Select ‘Change installation source’ and choose the ISO file you downloaded before.

   If you want to give your virtual machine a more descriptive name, select the ‘More options’ down-arrow and change ‘New Virtual Machine’ to something more useful, e.g. ‘Ubuntu Server 18.04 LTS’.
6. Select ‘Create Virtual Machine’ and wait for the virtual machine to be created.
7. Select ‘Connect’ to open a connection to your VM.
8. Select ‘File’ in the menu bar, then -> Choose ‘Settings’ and select the ‘Security’ tab -> Under Secure Boot choose ‘Microsoft UEFI Certificate Authority’ -> Then select ‘Apply’ and ‘OK’ to return to your VM.
9. Select ‘Start’ to run your VM.
10. Complete the manual installation of Ubuntu.

- *Setting up Ubuntu on Hyper-V* (Windows 11)

### See also

- Explanation: *Virtualisation and containers*

# 2.10. Containers

Our *Containers* section includes installation and usage guides for the most popular container tooling available in Ubuntu:

- **LXD**, which can also now be used to create virtual machines
- **Docker** and **rocks**

## 2.10.1. Containers

Containers are a lightweight, portable virtualization technology. They package software together with its dependencies so that applications can run consistently even across different environments.

- *How to use LXD*
- *Docker for sysadmins*
- *How to run rocks on your server*

### LXD containers

LXD (pronounced lex-dee) is a modern, secure, and powerful system container and virtual machine manager.

It provides a unified experience for running and managing full Linux systems inside containers or virtual machines. You can access it via the command line, its built-in graphical user interface, or a set of powerful REST APIs.

LXD scales from one instance on a single machine to a cluster in a full data center rack, making it suitable for both development and production workloads. You can even use LXD to set up a small, scalable private cloud, such as a MicroCloud.

This document will focus on how to configure and administer LXD on Ubuntu systems using the command line. On Ubuntu Server Cloud images, LXD comes pre-installed.

### Online resources

You can visit the official LXD documentation, or get in touch with the LXD team in their Ubuntu Discourse forum. The team also maintains a YouTube channel with helpful videos.

### Installation

LXD is pre-installed on Ubuntu Server cloud images. On other systems, the `lxd` package can be installed using:

```
sudo snap install lxd
```

This will install the self-contained LXD snap package.

### Kernel preparation

In general, Ubuntu should have all the desired features enabled by default. One exception to this is that in order to enable swap accounting, the boot argument `swapaccount=1` must be set. This can be done by appending it to the `GRUB_CMDLINE_LINUX_DEFAULT=`variable in /etc/default/grub, then running 'update-grub' as root and rebooting.

### Configuration

In order to use LXD, some basic settings need to be configured first. This is done by running `lxd init`, which will allow you to choose:

- Directory or ZFS container backend. If you choose ZFS, you can choose which block devices to use, or the size of a file to use as backing store.
- Availability over the network.
- A 'trust password' used by remote clients to vouch for their client certificate.

You must run 'lxd init' as root. 'lxc' commands can be run as any user who is a member of group lxd. If user joe is not a member of group 'lxd', you may run:

```
adduser joe lxd
```

as root to change it. The new membership will take effect on the next login, or after running `newgrp lxd` from an existing login.

See How to initialize LXD in the LXD documentation for more information on the configuration settings. Also, refer to the definitive configuration provided with the source code for the server, container, profile, and device configuration.

### Creating your first container

This section will describe the simplest container tasks.

### Creating a container

Every new container is created based on either an image, an existing container, or a container snapshot. At install time, LXD is configured with the following image servers:

- `ubuntu`: this serves official Ubuntu cloud image releases.
- `ubuntu-daily`: this serves official Ubuntu cloud images of the daily development releases.
- `ubuntu-minimal`: this serves official Ubuntu Minimal cloud image releases.
- `images`: this server provides unofficial images for a variety of Linux distributions. This is not the recommended server for Ubuntu images.

The command to create and start a container is

```
lxc launch remote:image containername
```

Images are identified by their hash, but are also aliased. The `ubuntu` remote knows many aliases such as `18.04` and `bionic`. A list of all images available from the Ubuntu Server can be seen using:

```
lxc image list ubuntu:
```

To see more information about a particular image, including all the aliases it is known by, you can use:

```
lxc image info ubuntu:bionic
```

You can generally refer to an Ubuntu image using the release name (`bionic`) or the release number (`18.04`). In addition, `lts` is an alias for the latest supported LTS release. To choose a different architecture, you can specify the desired architecture:

```
lxc image info ubuntu:lts/arm64
```

Now, let's start our first container:

```
lxc launch ubuntu:bionic b1
```

This will download the official current Bionic cloud image for your current architecture, then create a container named `b1` using that image, and finally start it. Once the command returns, you can see it using:

```
lxc list
lxc info b1
```

and open a shell in it using:

```
lxc exec b1 -- bash
```

A convenient alias for the command above is:

```
lxc shell b1
```

The try-it page mentioned above gives a full synopsis of the commands you can use to administer containers.

Now that the `bionic` image has been downloaded, it will be kept in sync until no new containers have been created based on it for (by default) 10 days. After that, it will be deleted.

### LXD server configuration

By default, LXD is socket activated and configured to listen only on a local UNIX socket. While LXD may not be running when you first look at the process listing, any LXC command will start it up. For instance:

```
lxc list
```

This will create your client certificate and contact the LXD server for a list of containers. To make the server accessible over the network you can set the http port using:

```
lxc config set core.https_address :8443
```

This will tell LXD to listen to port 8443 on all addresses.

### Authentication

By default, LXD will allow all members of group `lxd` to talk to it over the UNIX socket. Communication over the network is authorized using server and client certificates.

Before client `c1` wishes to use remote `r1`, `r1` must be registered using:

```
lxc remote add r1 r1.example.com:8443
```

The fingerprint of r1's certificate will be shown, to allow the user at c1 to reject a false certificate. The server in turn will verify that c1 may be trusted in one of two ways. The first is to register it in advance from any already-registered client, using:

```
lxc config trust add r1 certfile.crt
```

Now when the client adds r1 as a known remote, it will not need to provide a password as it is already trusted by the server.

The other step is to configure a 'trust password' with `r1`, either at initial configuration using `lxd init`, or after the fact using:

```
lxc config set core.trust_password PASSWORD
```

The password can then be provided when the client registers `r1` as a known remote.

### Backing store

LXD supports several backing stores. The recommended and the default backing store is `zfs`. If you already have a ZFS pool configured, you can tell LXD to use it during the `lxd init` procedure, otherwise a file-backed zpool will be created automatically. With ZFS, launching a new container is fast because the *filesystem* starts as a copy on write clone of the images' filesystem. Note that unless the container is privileged (see below) LXD will need to change ownership of all files before the container can start, however this is fast and change very little of the actual filesystem data.

The other supported backing stores are described in detail in the Storage configuration section of the LXD documentation.

### Container configuration

Containers are configured according to a set of profiles, described in the next section, and a set of container-specific configuration. Profiles are applied first, so that container specific configuration can override profile configuration.

Container configuration includes properties like the architecture, limits on resources such as CPU and RAM, security details including apparmor restriction overrides, and devices to apply to the container.

Devices can be of several types, including UNIX character, UNIX block, network interface, or disk. In order to insert a host mount into a container, a 'disk' device type would be used. For instance, to mount `/opt` in container `c1` at `/opt`, you could use:

```
lxc config device add c1 opt disk source=/opt path=opt
```

See:

```
lxc help config
```

for more information about editing container configurations. You may also use:

```
lxc config edit c1
```

to edit the whole of `c1`'s configuration. Comments at the top of the configuration will show examples of correct syntax to help administrators hit the ground running. If the edited configuration is not valid when the editor is exited, then the editor will be restarted.

### Profiles

Profiles are named collections of configurations which may be applied to more than one container. For instance, all containers created with `lxc launch`, by default, include the `default` profile, which provides a network interface `eth0`.

To mask a device which would be inherited from a profile but which should not be in the final container, define a device by the same name but of type 'none':

```
lxc config device add c1 eth1 none
```

### Nesting

Containers all share the same host kernel. This means that there is always an inherent trade-off between features exposed to the container and host security from malicious containers. Containers by default are therefore restricted from features needed to nest child containers. In order to run lxc or lxd containers under a lxd container, the `security.nesting` feature must be set to true:

```
lxc config set container1 security.nesting true
```

Once this is done, `container1` will be able to start sub-containers.

In order to run unprivileged (the default in LXD) containers nested under an unprivileged container, you will need to ensure a wide enough UID mapping. Please see the 'UID mapping' section below.

### Limits

LXD supports flexible constraints on the resources which containers can consume. The limits come in the following categories:

- CPU: limit cpu available to the container in several ways.
- Disk: configure the priority of I/O requests under load
- RAM: configure memory and swap availability
- Network: configure the network priority under load
- Processes: limit the number of concurrent processes in the container.

For a full list of limits known to LXD, see the configuration documentation.

### UID mappings and privileged containers

By default, LXD creates unprivileged containers. This means that root in the container is a non-root UID on the host. It is privileged against the resources owned by the container, but unprivileged with respect to the host, making root in a container roughly equivalent to an unprivileged user on the host. (The main exception is the increased attack surface exposed through the system call interface)

Briefly, in an unprivileged container, 65536 UIDs are 'shifted' into the container. For instance, UID 0 in the container may be 100000 on the host, UID 1 in the container is 100001, etc, up to 165535. The starting value for UIDs and *GIDs*, respectively, is determined by the 'root' entry the `/etc/subuid` and `/etc/subgid` files. (See the `subuid(5)`) manual page.)

It is possible to request a container to run without a UID mapping by setting the `security.privileged` flag to true:

```
lxc config set c1 security.privileged true
```

Note however that in this case the root user in the container is the root user on the host.

### Apparmor

LXD confines containers by default with an apparmor profile which protects containers from each other and the host from containers. For instance this will prevent root in one container from signaling root in another container, even though they have the same uid mapping. It also prevents writing to dangerous, un-namespaced files such as many sysctls and `/proc/sysrq-trigger`.

If the apparmor policy for a container needs to be modified for a container `c1`, specific apparmor policy lines can be added in the `raw.apparmor` configuration key.

### Seccomp

All containers are confined by a default seccomp policy. This policy prevents some dangerous actions such as forced umounts, kernel module loading and unloading, kexec, and the `open_by_handle_at` system call. The seccomp configuration cannot be modified, however a completely different seccomp policy – or none – can be requested using `raw.lxc` (see below).

### Raw LXC configuration

LXD configures containers for the best balance of host safety and container usability. Whenever possible it is highly recommended to use the defaults, and use the LXD configuration keys to request LXD to modify as needed. Sometimes, however, it may be necessary to talk to the underlying lxc driver itself. This can be done by specifying LXC configuration items in the 'raw.lxc' LXD configuration key. These must be valid items as documented in the `lxc.container.conf(5)` manual page.

### Snapshots

Containers can be renamed and live-migrated using the `lxc move` command:

```
lxc move c1 final-beta
```

They can also be snapshotted:

```
lxc snapshot c1 YYYY-MM-DD
```

Later changes to c1 can then be reverted by restoring the snapshot:

```
lxc restore u1 YYYY-MM-DD
```

New containers can also be created by copying a container or snapshot:

```
lxc copy u1/YYYY-MM-DD testcontainer
```

### Publishing images

When a container or container snapshot is ready for consumption by others, it can be published as a new image using;

```
lxc publish u1/YYYY-MM-DD --alias foo-2.0
```

The published image will be private by default, meaning that LXD will not allow clients without a trusted certificate to see them. If the image is safe for public viewing (i.e. contains no private information), then the 'public' flag can be set, either at publish time using

```
lxc publish u1/YYYY-MM-DD --alias foo-2.0 public=true
```

or after the fact using

```
lxc image edit foo-2.0
```

and changing the value of the public field.

### Image export and import

Image can be exported as, and imported from, tarballs:

```
lxc image export foo-2.0 foo-2.0.tar.gz
lxc image import foo-2.0.tar.gz --alias foo-2.0 --public
```

### Troubleshooting

To view debug information about LXD itself, on a systemd based host use

```
journalctl -u lxd
```

Container logfiles for container c1 may be seen using:

```
lxc info c1 --show-log
```

The configuration file which was used may be found under `/var/log/lxd/c1/lxc.conf` while apparmor profiles can be found in `/var/lib/lxd/security/apparmor/profiles/c1` and seccomp profiles in `/var/lib/lxd/security/seccomp/c1`.

### Docker for system admins

We are going to explore set-ups for configuring storage, networking, and logging in the subsequent sections. This will also help you get familiarized with Docker command line interface (CLI).

### Installation

First, install Docker if it's not already installed:

```
$ sudo apt-get install -y docker.io docker-compose-v2
```

### Configuring storage

### How to configure volumes

- Create a volume

```
$ docker volume create my-vol

my-vol
```

- List volumes

```
$ docker volume ls

DRIVER        VOLUME NAME
local         my-vol
```

- Inspect volume

```
$ docker volume inspect my-vol
```

```
[
  {
    "CreatedAt": "2023-10-25T00:53:24Z",
    "Driver": "local",
    "Labels": null,
    "Mountpoint": "/var/lib/docker/volumes/my-vol/_data",
    "Name": "my-vol",
    "Options": null,
    "Scope": "local"
  }
]
```

- Run a container and mount a volume

```
$ docker run --name web-server -d \
--mount source=my-vol,target=/app \
ubuntu/apache2

0709c1b632801fddd767deddda0d273289ba423e9228cc1d77b2194989e0a882
```

- Inspect your container to make sure the volume is mounted correctly:

```
docker inspect web-server --format '{{ json .Mounts }}' | jq .
```

```
[
  {
    "Type": "volume",
    "Name": "my-vol",
    "Source": "/var/lib/docker/volumes/my-vol/_data",
    "Destination": "/app",
    "Driver": "local",
    "Mode": "z",
    "RW": true,
    "Propagation": ""
  }
]
```

By default, all your volumes will be stored in `/var/lib/docker/volumes`.

- Stop and remove the container, then remove its volume.

```
docker stop web-server
docker rm web-server
docker volume rm my-vol
```

### How to configure bind mounts

- Create a Docker container and bind mount your host directory:

```
$ docker run -d \
--name web-server \
--mount type=bind,source="$(pwd)",target=/app \
ubuntu/apache2

6f5378e34d6c6811702e16d047a5a80f18adbd9d8a14b11050ae3c3353bf8d2a
```

- Inspect your container to check for the bind mount:

```
docker inspect web-server --format '{{ json .Mounts }}' | jq .
```

```
[
  {
    "Type": "bind",
    "Source": "/root",
    "Destination": "/app",
    "Mode": "",
    "RW": true,
    "Propagation": "rprivate"
  }
]
```

- Stop and remove the container

```
docker stop web-server
docker rm web-server
```

### How to configure Tmpfs

- Create a Docker container and mount a tmpfs:

```
$ docker run --name web-server -d \
--mount type=tmpfs,target=/app \
ubuntu/apache2

03483cc28166fc5c56317e4ee71904941ec5942071e7c936524f74d732b6a24c
```

- Inspect your container to check for the tmpfs mount:

```
docker inspect web-server --format '{{ json .Mounts }}' | jq .
```

```
[
  {
    "Type": "tmpfs",
    "Source": "",
    "Destination": "/app",
    "Mode": "",
    "RW": true,
    "Propagation": ""
  }
]
```

### Choosing the right storage drivers

Before changing the configuration and restarting the daemon, make sure that the specified filesystem (zfs, btrfs, or device mapper) is mounted at `/var/lib/docker`. Otherwise, if you configure the Docker daemon to use a storage driver different from the filesystem mounted at `/var/lib/docker`, a failure will happen. The Docker daemon expects that `/var/lib/docker` is correctly set up when it starts.

- Check the current storage driver

```
$ docker info | grep "Storage Driver"

Storage Driver: overlay2
```

- Ensure the required Filesystem is available. We will be using the ZFS Filesystem.

```
$ apt install zfsutils-linux -y # Install ZFS
$ fallocate -l 5G /zfs-pool.img  # Create a 5GB file
$ zpool create mypool /zfs-pool.img  # Create a ZFS pool
$ zfs create -o mountpoint=/var/lib/docker mypool/docker # Create a ZFS
dataset and mount it to dockers directory, "/var/lib/docker".
$ zfs list # Verify that it mounted successfully
```

(continues on next page)

(continued from previous page)

```
NAME            USED  AVAIL  REFER  MOUNTPOINT
mypool          162K  4.36G    24K  /mypool
mypool/docker    39K  4.36G    39K  /var/lib/docker
```

- Change the storage driver
  1. Stop the docker daemon

     ```
     systemctl stop docker
     ```

  2. Edit `/etc/docker/daemon.json` using your favorite editor, then update the storage driver value to `zfs`.

     ```
     vim /etc/docker/daemon.json
     ```

     ```
     {
       "storage-driver": "zfs"
     }
     ```

  3. Restart the docker daemon

     ```
     systemctl restart docker
     ```

- Verify the change

```
$ docker info | grep "Storage Driver"

Storage Driver: zfs
```

### Configuring networking

This is how you can create a user-defined network using the Docker CLI:

- Create a network

```
$ docker network create --driver bridge my-net

D84efaca11d6f643394de31ad8789391e3ddf29d46faecf0661849f5ead239f7
```

- List networks

```
$ docker network ls

NETWORK ID          NAME          DRIVER        SCOPE
1f55a8891c4a   bridge        bridge        local
9ca94be2c1a0   host          host          local
d84efaca11d6   my-net        bridge        local
5d300e6a07b1   none          null          local
```

- Inspect the network we created

```
docker network inspect my-net
```


```
[
  {
    "Name": "my-net",
    "Id": "d84efaca11d6f643394de31ad8789391e3ddf29d46faecf0661849f5ead239f7",
    "Created": "2023-10-25T22:18:52.972569338Z",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": {},
      "Config": [
        {
          "Subnet": "172.18.0.0/16",
          "Gateway": "172.18.0.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {},
    "Options": {},
    "Labels": {}
  }
]
```


Containers can connect to a defined network when they are created (via `docker run`) or at any time of its lifecycle.

### Connecting a new container to an existing network

```
$ docker run -d --name c1 --network my-net ubuntu/apache2

C7aa78f45ce3474a276ca3e64023177d5984b3df921aadf97e221da8a29a891e
```

- View the network connected to the container

```
docker inspect c1 --format '{{ json .NetworkSettings }}' | jq .
```


```
{
  "Bridge": "",
  "SandboxID":
"ee1cc10093fdfdf5d4a30c056cef47abbfa564e770272e1e5f681525fdd85555",
```


(continues on next page)

(continued from previous page)


```
  "HairpinMode": false,
  "LinkLocalIPv6Address": "",
  "LinkLocalIPv6PrefixLen": 0,
  "Ports": {
    "80/tcp": null
  },
  "SandboxKey": "/var/run/docker/netns/ee1cc10093fd",
  "SecondaryIPAddresses": null,
  "SecondaryIPv6Addresses": null,
  "EndpointID": "",
  "Gateway": "",
  "GlobalIPv6Address": "",
  "GlobalIPv6PrefixLen": 0,
  "IPAddress": "",
  "IPPrefixLen": 0,
  "IPv6Gateway": "",
  "MacAddress": "",
  "Networks": {
    "my-net": {
      "IPAMConfig": null,
      "Links": null,
      "Aliases": ["c7aa78f45ce3"],
      "NetworkID":
"d84efaca11d6f643394de31ad8789391e3ddf29d46faecf0661849f5ead239f7",
      "EndpointID":
"1cb76d44a484d302137bb4b042c8142db8e931e0c63f44175a1aa75ae8af9cb5",
      "Gateway": "172.18.0.1",
      "IPAddress": "172.18.0.2",
      "IPPrefixLen": 16,
      "IPv6Gateway": "",
      "GlobalIPv6Address": "",
      "GlobalIPv6PrefixLen": 0,
      "MacAddress": "02:42:ac:12:00:02",
      "DriverOpts": null
    }
  }
}
```


### Connecting a running container to an existing network

- Make a running container connect to the existing network
  1. Create the container

     ```
     $ docker run -d --name c2 ubuntu/nginx

     Fea22fbb6e3685eae28815f3ad8c8a655340ebcd6a0c13f3aad0b45d71a20935
     ```

  2. Connect the running container to the network and verify that it's connected.

```
docker network connect my-net c2
docker inspect c2 --format '{{ json .NetworkSettings }}' | jq .
```

```
{
  "Bridge": "",
  "SandboxID":
"82a7ea6efd679dffcc3e4392e0e5da61a8ccef33dd78eb5381c9792a4c01f366",
  "HairpinMode": false,
  "LinkLocalIPv6Address": "",
  "LinkLocalIPv6PrefixLen": 0,
  "Ports": {
    "80/tcp": null
  },
  "SandboxKey": "/var/run/docker/netns/82a7ea6efd67",
  "SecondaryIPAddresses": null,
  "SecondaryIPv6Addresses": null,
  "EndpointID":
"490c15cf3bcb149dd8649e3ac96f71addd13f660b4ec826dc39e266184b3f65b",
  "Gateway": "172.17.0.1",
  "GlobalIPv6Address": "",
  "GlobalIPv6PrefixLen": 0,
  "IPAddress": "172.17.0.3",
  "IPPrefixLen": 16,
  "IPv6Gateway": "",
  "MacAddress": "02:42:ac:11:00:03",
  "Networks": {
    "bridge": {
      "IPAMConfig": null,
      "Links": null,
      "Aliases": null,
      "NetworkID":
"1f55a8891c4a523a288aca8881dae0061f9586d5d91c69b3a74e1ef3ad1bfcf4",
      "EndpointID":
"490c15cf3bcb149dd8649e3ac96f71addd13f660b4ec826dc39e266184b3f65b",
      "Gateway": "172.17.0.1",
      "IPAddress": "172.17.0.3",
      "IPPrefixLen": 16,
      "IPv6Gateway": "",
      "GlobalIPv6Address": "",
      "GlobalIPv6PrefixLen": 0,
      "MacAddress": "02:42:ac:11:00:03",
      "DriverOpts": null
    },
    "my-net": {
      "IPAMConfig": {},
      "Links": null,
      "Aliases": ["fea22fbb6e36"],
      "NetworkID":
"d84efaca11d6f643394de31ad8789391e3ddf29d46faecf0661849f5ead239f7",
      "EndpointID":
```

(continues on next page)

(continued from previous page)

```
"17856b7f6902db39ff6ab418f127d75d8da597fdb8af0a6798f35a94be0cb805",
        "Gateway": "172.18.0.1",
        "IPAddress": "172.18.0.3",
        "IPPrefixLen": 16,
        "IPv6Gateway": "",
        "GlobalIPv6Address": "",
        "GlobalIPv6PrefixLen": 0,
        "MacAddress": "02:42:ac:12:00:03",
        "DriverOpts": {}
    }
  }
}
```

The container c2 is connected to two networks `bridge` and `my-net`.

The default network created by the Docker daemon is called `bridge` using the **bridge network driver**.

### Modifying the default network "bridge"

- A system administrator can modify this default networks IP address by editing `/etc/docker/daemon.json` and including the below into the JSON object

```
vim /etc/docker/daemon.json
```

```
{
  "bip": "192.168.1.1/24",
  "fixed-cidr": "192.168.1.0/25",
  "fixed-cidr-v6": "2001:db8::/64",
  "mtu": 1500,
  "default-gateway": "192.168.1.254",
  "default-gateway-v6": "2001:db8:abcd::89",
  "dns": ["10.20.1.2", "10.20.1.3"]
}
```

- Restart the Docker daemon

```
systemctl restart docker
```

- Verify your changes

```
docker network inspect bridge
```

### Exposing a container port to the host

After deciding how you are going to manage the network and selecting the most appropriate driver, there are some specific deployment details that a system administrator has to bear in mind when running containers.

Exposing ports of any system is always a concern, since it increases the surface for malicious attacks. For containers, we also need to be careful, analysing whether we really need to

publish ports to the host. For instance, if the goal is to allow containers to access a specific port from another container, there is no need to publish any port to the host. This can be solved by connecting all the containers to the same network. You should publish ports of a container to the host only if you want to make it available to non-Docker workloads. When a container is created no port is published to the host, the option `--publish` (or `-p`) should be passed to `docker run` or `docker create` listing which port will be exposed and how.

The `--publish` option of Docker CLI accepts the following options:

- First, the host port that will be used to publish the container's port. It can also contain the IP address of the host. For example, `0.0.0.0:8080`.
- Second, the container's port to be published. For example, `80`.
- Third (optional), the type of port that will be published which can be TCP or UDP. For example, `80/tcp` or `80/udp`.

An example of how to publish port `80` of a container to port `8080` of the host:

- Create a container and expose it's port to the host

```
$ docker run -d --name web-server --publish 8080:80 ubuntu/nginx

f451aa1990db7d2c9b065c6158e2315997a56a764b36a846a19b1b96ce1f3910
```

- View the containers network settings

```
docker inspect web-server --format '{{ json .NetworkSettings.Ports }}' | jq .
```

```
{
  "80/tcp": [
    {
      "HostIp": "0.0.0.0",
      "HostPort": "8080"
    },
    {
      "HostIp": "::",
      "HostPort": "8080"
    }
  ]
}
```

The `HostIp` values are `0.0.0.0` (IPv4) and `::` (IPv6), and the service running in the container is accessible to everyone in the network (reaching the host), if you want to publish the port from the container and let the service be available just to the host you can use `--publish 127.0.0.1:8080:80` instead. The published port can be TCP or UDP and one can specify that passing `--publish 8080:80/tcp` or `--publish 8080:80/udp`.

The system administrator might also want to manually set the IP address or the *hostname* of the container. To achieve this, one can use the `--ip` (IPv4), `--ip6` (IPv6), and `--hostname` options of the `docker network connect` command to specify the desired values.

Another important aspect of networking with containers is the *DNS* service. By default containers will use the DNS setting of the host, defined in `/etc/resolv.conf`. Therefore, if a container is created and connected to the default `bridge` network it will get a copy of host's `/etc/resolv.conf`. If the container is connected to a user-defined network, then it will use

Docker's embedded DNS server. The embedded DNS server forwards external DNS lookups to the DNS servers configured on the host. In case the system administrator wants to configure the DNS service, the `docker run` and `docker create` commands have options to allow that, such as `--dns` (IP address of a DNS server) and `--dns-opt` (key-value pair representing a DNS option and its value). For more information, check the manpages of those commands.

### Managing logs

The default logging driver is called `json file`, and the system administrator can change it to suite their needs.

### Modifying the logging driver via the docker daemon file

- Edit the docker daemon file and update the logging driver

```
vim /etc/docker/daemon.json
```

```
{
  "log-driver": "journald"
}
```

### Modifying the logging driver when creating a container

Another option is specifying the logging driver during container creation time:

- Specify a log driver when executing a `docker run`

```
$ docker run -d --name web-server --log-driver=journald ubuntu/nginx

1c08b667f32d8b834f0d9d6320721e07de5f22168cfc8a024d6e388daf486dfa
```

- Verify your configuration

```
docker inspect web-server --format '{{ json .HostConfig.LogConfig }}' | jq .
```

```
{
  "Type": "journald",
  "Config": {}
}
```

- View logs

```
$ docker logs web-server

/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform
configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
```

Depending on the driver you might also want to pass some options. You can do that via the CLI, passing `--log-opt` or in the daemon config file adding the key `log-opts`. For more information check the logging driver documentation.

Docker CLI also provides the `docker logs` and `docker service logs` commands which allows one to check for the logs produced by a given container or service (set of containers) in the host. However, those two commands are functional only if the logging driver for the containers is `json-file`, `local` or `journald`. They are useful for debugging in general, but there is the downside of increasing the storage needed in the host.

The remote logging drivers are useful to store data in an external service/host, and they also avoid spending more disk space in the host to store log files. Nonetheless, sometimes, for debugging purposes, it is important to have log files locally. Considering that, Docker has a feature called "dual logging", which is enabled by default, and even if the system administrator configures a logging driver different from `json-file`, `local` and `journald`, the logs will be available locally to be accessed via the Docker CLI. If this is not the desired behaviour, the feature can be disabled in the `/etc/docker/daemon.json` file:

```
{
  "log-driver": "syslog",
  "log-opts": {
    "cache-disabled": "true",
    "syslog-address": "udp://1.2.3.4:1111"
  }
}
```

The option `cache-disabled` is used to disable the "dual logging" feature. If you try to run `docker logs` with that configuration you will get the following error:

```
$ docker logs web-server

Error response from daemon: configured logging driver does not support reading
```

### Resources

To read an explanatory guide to Docker storage, networking, and logging see:

- *Docker storage, networking, and logging*

### How to run rocks on your server

### Deploying rocks with Docker

As with any other OCI-compliant container image, rocks can be deployed with your favourite container management tool. This section depicts a typical deployment workflow for a generic Grafana rock, using *Docker*.

First, install Docker if it's not already installed:

```
$ sudo apt-get install -y docker.io docker-compose-v2
```

We can deploy a container with the `docker run` command. This command has a number of possible parameters. The "Usage" section of the Grafana rock's documentation has a table with an overview of parameters specific to the image.

```
$ sudo docker run -d --name grafana-container -e TZ=UTC -p 3000:3000 ubuntu/
grafana:10.3.3-22.04_stable
```

(continues on next page)

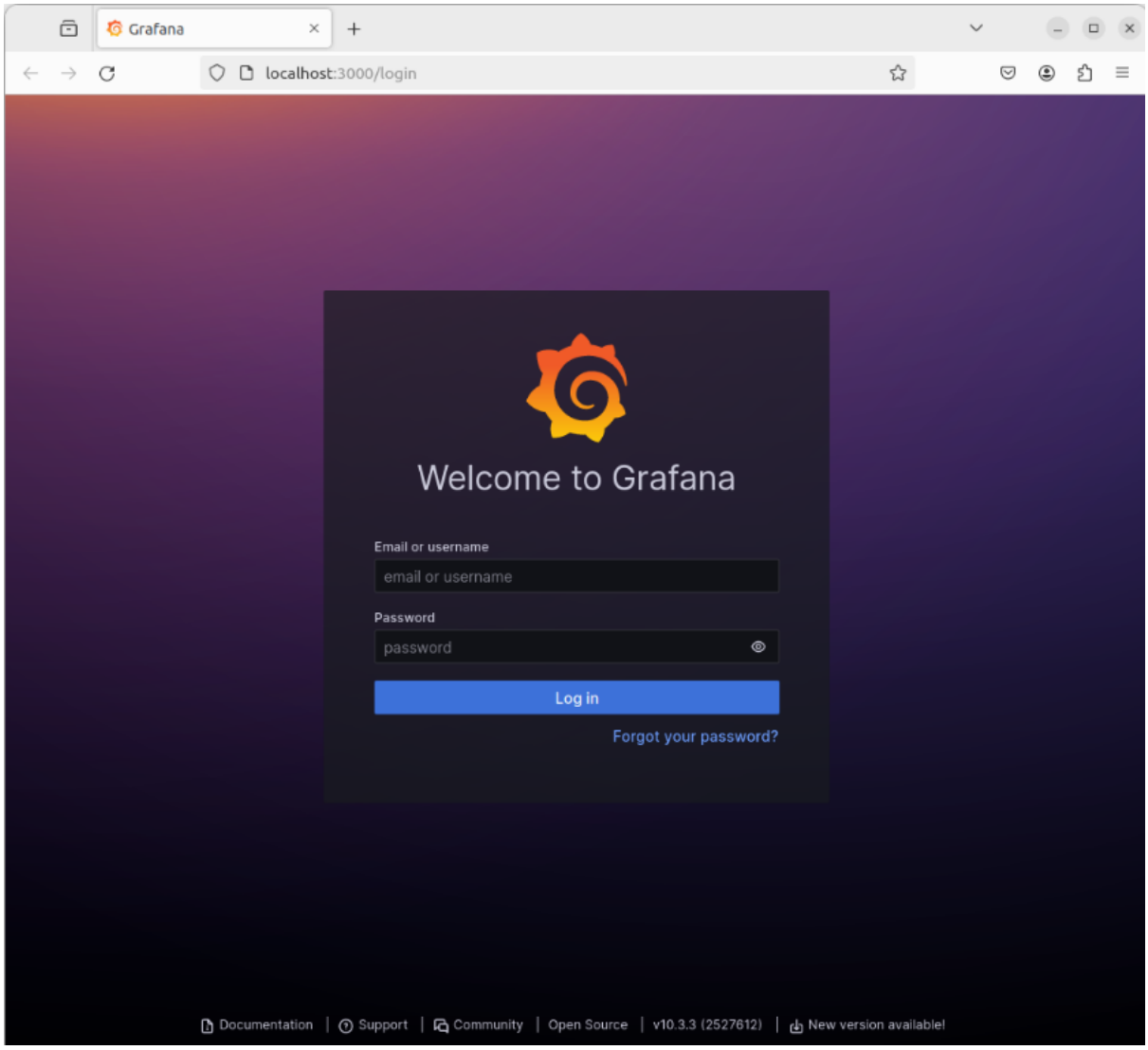
(continued from previous page)

```
Unable to find image 'ubuntu/grafana:10.3.3-22.04_stable' locally
10.3.3-22.04_stable: Pulling from ubuntu/grafana
bccd10f490ab: Already exists
549078d9d057: Pull complete
6ef870aa8500: Pull complete
2b475da7ccbd: Pull complete
Digest: sha256:df566ef90ecb14267a459081949ee7b6693fa573b97a7134a9a6722207275caa
Status: Downloaded newer image for ubuntu/grafana:10.3.3-22.04_stable
356e623ef2c16bc7d810bddccad8d7980f9c633aefc3a88bc8761eac4e1b1c50
```

In this particular case, we're using:

- `-d` to run the container in the background.
- We are also specifying a well-defined name for the container, with the `--name` parameter.
- With `-e` we are setting the container's timezone (`TZ`) environment variable to `UTC`.
- We also use `-p` to map port 3000 of the container to 3000 on localhost.
- The last parameter indicates the name of the rock, as listed in Docker Hub. Notice that the image tag we requested has the `_stable` suffix to indicate the image's risk. This is called a **Channel Tag** and it follows a similar convention to snap "channels".

This container, named `grafana-container`, serves Grafana 10.3.3 in an Ubuntu 22.04 LTS environment and can be accessed via local port 3000. Load the website up in your local web browser:




If you don't have Firefox handy, `curl` can be used instead:

```
$ curl -s http://localhost:3000/login | grep "<title>"
    <title>Grafana</title>
```

Now that we've tested the deployment of the Grafana rock as a single container, let's clean it up:

```
$ sudo docker ps
CONTAINER ID   IMAGE                                  COMMAND
CREATED          STATUS          PORTS                                       NAMES
356e623ef2c1   ubuntu/grafana:10.3.3-22.04_stable   "/bin/pebble enter -…"   17
minutes ago   Up 17 minutes   0.0.0.0:3000->3000/tcp, :::3000->3000/tcp   grafana-
container
```

We can stop and remove the container as follows:

```
$ sudo docker stop grafana-container
$ sudo docker rm grafana-container
```

The Grafana rock's documentation will also show you how to use Docker's `-v` bind mounts to

configure Grafana's provisioning directory and data persistence.

### Multi-container deployment

The section above explained the use of a single container for running a single software instance, but one of the benefits of using rocks is the ability to easily create and architecturally organise (or "orchestrate") them to operate together in a modular fashion.

This section will demonstrate use of `docker-compose` to set up two container services that inter-operate to implement a trivial observability stack with the Prometheus and Grafana rocks.

Start by creating a Prometheus configuration file called `prometheus.yml` with the following contents:

```
global:
  scrape_interval: 1m

scrape_configs:
  - job_name: 'prometheus'
    scrape_interval: 1m
    static_configs:
      - targets: ['localhost:9090']
```

Note that this is a very simplistic example, where Prometheus only collects metrics about itself. You could expand the above configuration to tell Prometheus to scrape metrics from other sources.

Then, create the Compose file `docker-compose.yml` and define both services:

```
services:
  grafana:
    image: ubuntu/grafana:10.3.3-22.04_stable
    container_name: grafana-container
    environment:
      TZ: UTC
    ports:
      - "3000:3000"
  prometheus:
    image: ubuntu/prometheus:2.49.1-22.04_stable
    container_name: prometheus-container
    environment:
      TZ: UTC
    ports:
      - "9090:9090"
    volumes:
      - ./prometheus.yml:/etc/prometheus/prometheus.yml
```

Note that the Prometheus configuration file is being given to the container via a Docker volume (of type "bind mount"). The above sample could also be improved to also use another volume for persisting data, and even a Grafana default configuration for the Prometheus datasource.

Since we already installed Docker in the section above, all that is needed is to create and start the containers defined in this Compose file. This can be achieved with:

```
$ sudo docker compose up -d
[+] Running 10/10
 ✓ grafana Pulled


   ✓ bccd10f490ab Already exists


   ✓ 549078d9d057 Pull complete


   ✓ 6ef870aa8500 Pull complete


   ✓ 2b475da7ccbd Pull complete


 ✓ prometheus Pulled


   ✓ a8b1c5f80c2d Already exists


   ✓ f021062473aa Pull complete


   ✓ 9c6122d12d1d Pull complete


   ✓ 274b56f68abe Pull complete


[+] Running 3/3
 ✓ Network compose_default         Created


 ✓ Container prometheus-container  Started
```

(continues on next page)

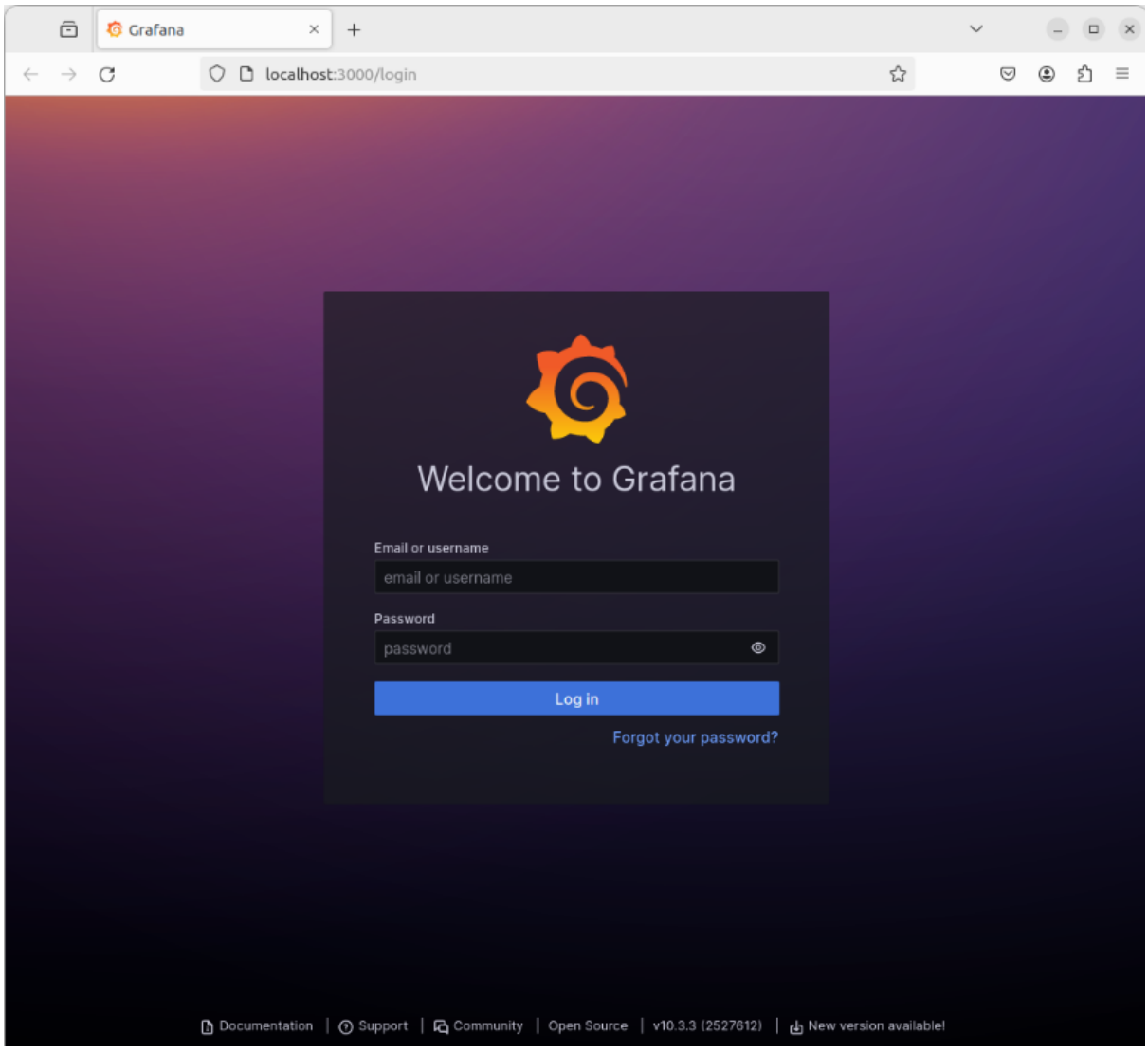
(continued from previous page)

```
 ✓ Container grafana-container      Started
```

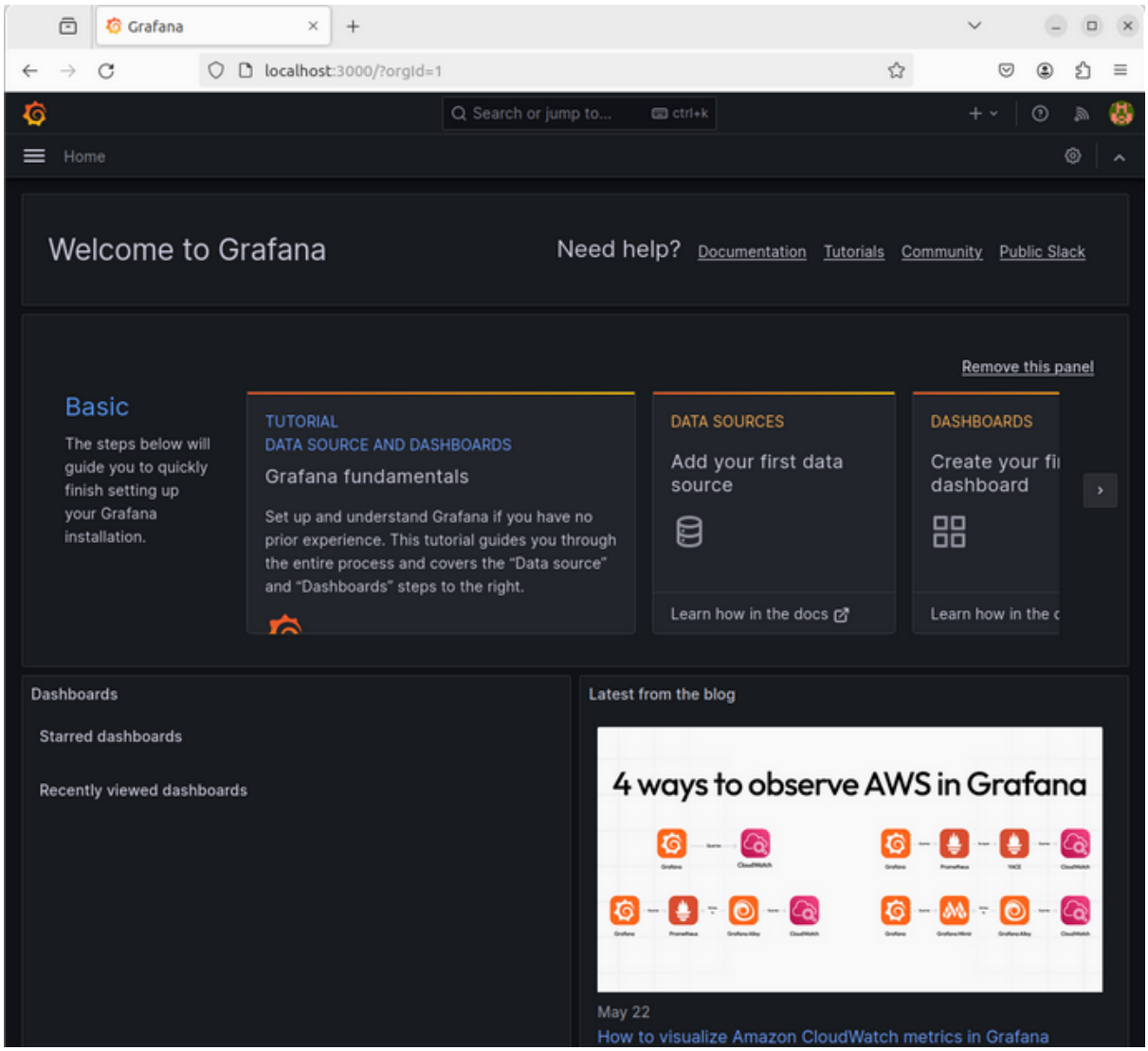
As before, the `-d` indicates that all containers in this stack should be started in the background. You can confirm they are live and running with:

```
$ sudo docker compose ps
NAME                   IMAGE                                      COMMAND
     SERVICE       CREATED          STATUS          PORTS
grafana-container      ubuntu/grafana:10.3.3-22.04_stable      "/bin/pebble enter
-…"   grafana       3 seconds ago    Up 3 seconds    0.0.0.0:3000->3000/tcp, :::3000-
>3000/tcp
prometheus-container   ubuntu/prometheus:2.49.1-22.04_stable   "/bin/pebble enter
-…"   prometheus    3 seconds ago    Up 3 seconds    0.0.0.0:9090->9090/tcp, :::9090-
>9090/tcp
```

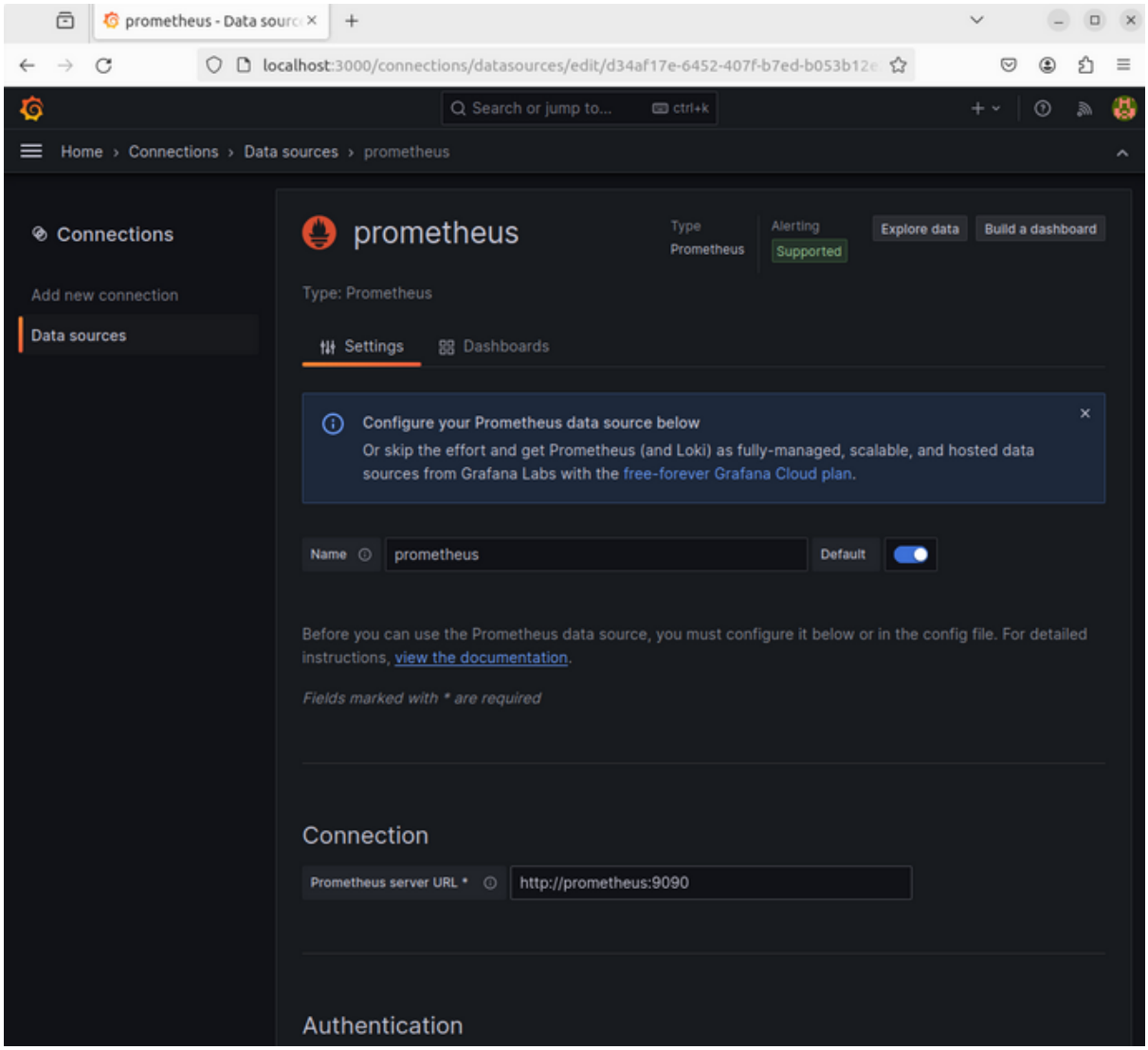
Opening `http://localhost:3000` will give you the same Grafana login page as before:



Use the default username `admin` and password `admin` to login:

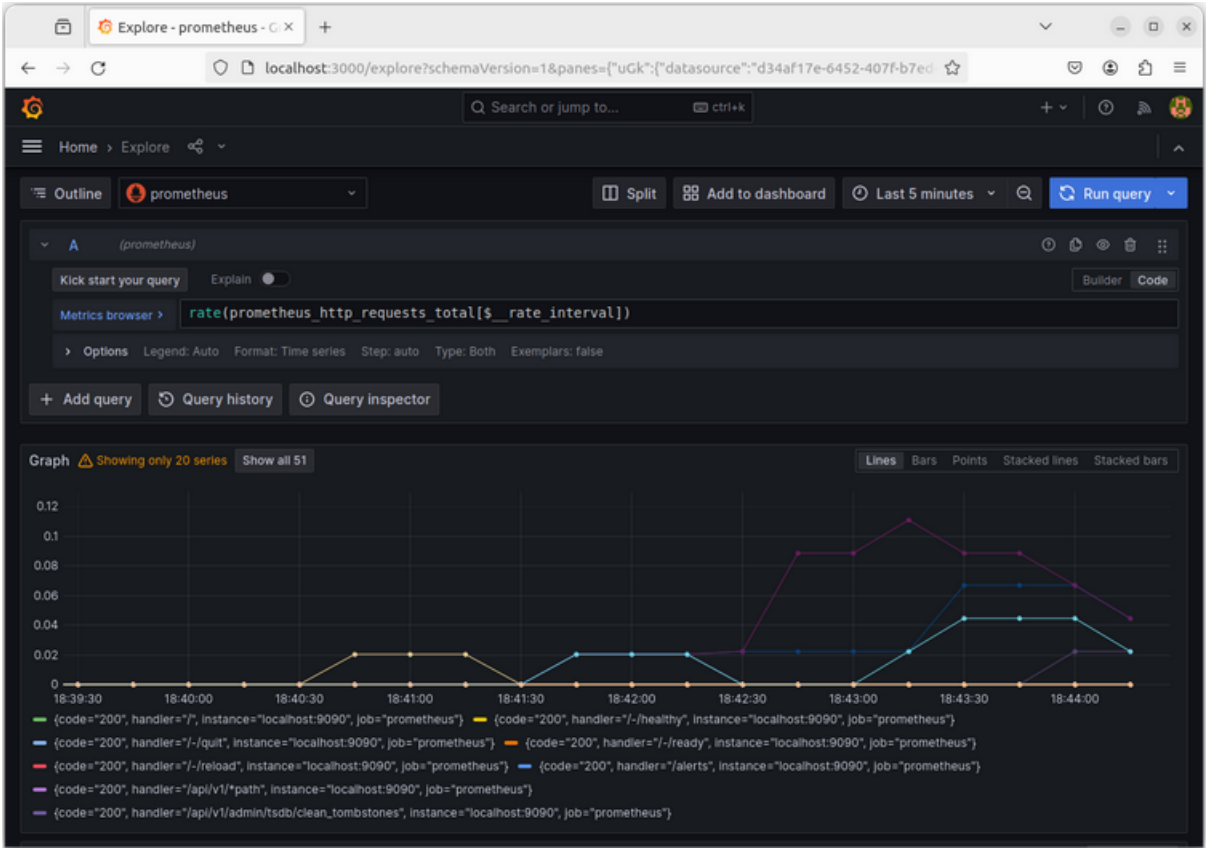



By clicking on "Data Sources" you can then add Prometheus and provide the server URL `http://prometheus:9090`:




This URL works because Docker Compose ensures both containers are on the same Docker network and that they can be discovered via their service name.

Finally, click on “Explore” from the Grafana menu, and select the `prometheus` datasource. You can now query and visualise the Prometheus metrics. For example:




### Next Steps

As you can see, `docker-compose` makes it convenient to set up multi-container applications without needing to perform runtime changes to the containers. As you can imagine, this can permit building a more sophisticated management system to handle fail-over, load-balancing, scaling, upgrading old nodes, and monitoring status. But rather than needing to implement all of this directly on top of `docker-container`, you may want to investigate Kubernetes-style cluster management software such as microk8s.

### See also

- Explanation: *Virtualisation and containers*

# 2.11. High Availability

*High Availability* is a method for clustering resources to ensure minimal downtime if a particular component fails. This section shows how to set up various components of a High Availability setup.

## 2.11.1. High availability

Distributed Replicated Block Device (DRBD) mirrors block devices between multiple hosts. This guide shows how to install and configure a DRBD.

- *Install a Distributed Replicated Block Device*

### Distributed Replicated Block Device (DRBD)

Distributed Replicated Block Device (DRBD) mirrors block devices between multiple hosts. The replication is transparent to other applications on the host systems. Any block device hard disks, partitions, RAID devices, logical volumes, etc can be mirrored.

### Install DRBD

To get started using DRBD, first install the necessary packages. In a terminal window, run the following command:

```
sudo apt install drbd-utils
```

> **Note**
>
> If you are using the **virtual kernel** as part of a virtual machine you will need to manually compile the `drbd` module. It may be easier to install the `linux-modules-extra-$(uname -r)` package inside the virtual machine.

### Configure DRBD

This section covers setting up a DRBD to replicate a separate `/srv` partition, with an `ext3` *filesystem* between two hosts. The partition size is not particularly relevant, but both partitions need to be the same size.

The two hosts in this example will be called `drbd01` and `drbd02`. They will need to have name resolution configured either through *DNS* or the `/etc/hosts` file. See our *guide to DNS* for details.

On the first host, edit `/etc/drbd.conf` as follows:

```
global { usage-count no; }
common { syncer { rate 100M; } }
resource r0 {
        protocol C;
        startup {
                wfc-timeout  15;
                degr-wfc-timeout 60;
        }
        net {
                cram-hmac-alg sha1;
                shared-secret "secret";
        }
        on drbd01 {
                device /dev/drbd0;
                disk /dev/sdb1;
                address 192.168.0.1:7788;
                meta-disk internal;
        }
        on drbd02 {
                device /dev/drbd0;
```

(continues on next page)

(continued from previous page)

```
                disk /dev/sdb1;
                address 192.168.0.2:7788;
                meta-disk internal;
        }
}
```

**Note**

There are many other options in `/etc/drbd.conf`, but for this example the default values are enough.

Now copy `/etc/drbd.conf` to the second host:

```
scp /etc/drbd.conf drbd02:~
```

And, on `drbd02`, move the file to `/etc`:

```
sudo mv drbd.conf /etc/
```

Now using the `drbdadm` utility, initialise the meta data storage. On both servers, run:

```
sudo drbdadm create-md r0
```

Next, on both hosts, start the `drbd` daemon:

```
sudo systemctl start drbd.service
```

On `drbd01` (or whichever host you wish to be the primary), enter the following:

```
sudo drbdadm -- --overwrite-data-of-peer primary all
```

After running the above command, the data will start syncing with the secondary host. To watch the progress, on `drbd02` enter the following:

```
watch -n1 cat /proc/drbd
```

To stop watching the output press Ctrl + C.

Finally, add a filesystem to `/dev/drbd0` and mount it:

```
sudo mkfs.ext3 /dev/drbd0
sudo mount /dev/drbd0 /srv
```

### Testing

To test that the data is actually syncing between the hosts copy some files on `drbd01`, the primary, to `/srv`:

```
sudo cp -r /etc/default /srv
```

Next, unmount `/srv`:

```
sudo umount /srv
```

Now demote the **primary** server to the **secondary** role:

```
sudo drbdadm secondary r0
```

Now on the **secondary** server, promote it to the **primary** role:

```
sudo drbdadm primary r0
```

Lastly, mount the partition:

```
sudo mount /dev/drbd0 /srv
```

Using `ls` you should see `/srv/default` copied from the former primary host `drbd01`.

### Further reading

- For more information on DRBD see the DRBD web site.
- The `drbd.conf(5)` manual page contains details on the options not covered in this guide.
- Also, see the `drbdadm(8)` manpage.

### See also

- Explanation: *High Availability*
- Reference: *High availability*

# 2.12. Observability

*Observability* is a name given to the collection of tools used to monitor your infrastructure. In Ubuntu, you can use the classic Logging, Monitoring, and Alerting (LMA) stack, or the newer Canonical Observability Stack.

This section focuses on the classic LMA stack.

## 2.12.1. Observability

In Ubuntu, it is recommended to use the Canonical Observability Stack to monitor your infrastructure.

However, you can also use the classic Logging, Monitoring and Alerting (LMA) stack.

### Set up your LMA stack

> **Note**
>
> **LMA to COS**
>
> The LMA stack is being succeeded by the Canonical Observability Stack (COS). While the current LMA still works, most users are recommended to consider COS instead. For more information, refer to this COS topic. In environments with more limited resources, there is also COS lite.

Logging, Monitoring, and Alerting (LMA) is a collection of tools that guarantee the availability of your running infrastructure. Your LMA stack will help point out issues in load, networking, and other resources before they become a failure point.

### Architectural overview

Canonical's LMA stack involves several discrete software services acting in concert.

**Telegraf** collects metrics from the operating system, running software, and other inputs. Its plugin system permits export of data in any arbitrary format; for this system we collect the data in a central data manager called **Prometheus**.

Prometheus works as a hub, polling data from different Telegraf nodes and sending it to various outputs, including persistent storage. For this LMA stack, visualisation is handled via **Grafana** and email/pager alerts are generated via the Prometheus *Alertmanager* plugin. See **Prometheus Alertmanager** for more details.

### Getting started

Let's set up a basic demonstration with two **nodes**, the first acting as a placeholder load with Telegraf installed - the "Workload", and the second acting as our data visualisation system - the "Monitor". This will help us familiarise ourselves with the various components and how they inter-operate.

> **Note**
>
> For clarity, we'll refer to these two hosts as named: `workload` and `monitor`. If you use other *hostnames*, substitute your preferred names as we go through this guide.

The Workload node will be running Telegraf to collect metrics from whatever load we're monitoring. For demonstration purposes we'll just read the CPU/memory data from the node. In a real environment, we'd have multiple hosts (each with their own Telegraf instance) collecting hardware, network, and software statuses particular to that node.

Our Monitor node will double as both a data store and a web UI, receiving data from the Workload, storing it to disk, and displaying it for analysis.

### Ports

As reference, here are the ports we'll be binding for each service:

| | |
|---|---|
| Prometheus | `monitor:9090` |
| Alertmanager | `monitor:9093` |
| Grafana | `monitor:3000` |
| Telegraf | `workload:9273` |

### Set up the Workload node

First, let's set up the Workload. We'll be using LXD as our container technology in this guide, but any VM, container, or bare metal host should work, so long as it's running Ubuntu 20.10. With LXD installed on our host we can use its `lxc` command line tool to create our containers:

```
$ lxc launch ubuntu:20.10 workload
Creating workload
Starting workload

$ lxc exec workload -- bash
workload:~#
```

On the Workload, install Telegraf:

```
workload:~# apt update
workload:~# apt install telegraf
```

Telegraf processes input data to transform, filter, and decorate it, and then performs selected aggregation functions on it such as tallies, averages, etc. The results are published for collection by external services; in our case Prometheus will be collecting the CPU/memory data from the Monitor node.

Open `/etc/telegraf/telegraf.conf` and scroll down to the "INPUT PLUGINS" section. What we'll need is the following configuration settings, which you should find already enabled by default:

```
[[inputs.cpu]]
  percpu = true
  totalcpu = true
  collect_cpu_time = false
  report_active = false
```

Looking at the config file you'll notice it's almost entirely commented out. There are three different types of sections in the file: `[[inputs]]`, which we set above; `[[outputs]]`, which we'll set up next; and the `[[agent]]` setting, with several performance tuning parameters such as the collection interval, which we're setting to 10 seconds. The agent defaults are fine for our example and for basic use.

Finally, we need to define where Telegraf will provide its output. Open `/etc/telegraf/telegraf.conf` and scroll down to the "OUTPUT PLUGINS" section and add the following output configuration:

```
[[outputs.prometheus_client]]
  listen = "workload:9273"
  metric_version = 2

#[[outputs.influxdb]]
```

We won't be using `Influxdb`, so you can comment that section out (if it's enabled).

Now restart the Telegraf service:

```
workload:~# systemctl restart telegraf
workload:~# systemctl status telegraf
● telegraf.service - The plugin-driven server agent for reporting metrics into
InfluxDB
         Loaded: loaded (/lib/systemd/system/telegraf.service; enabled; vendor
preset: enabled)
         Active: active (running) since Sat 2020-10-31 02:17:57 UTC; 6s ago
           Docs: https://github.com/influxdata/telegraf
   Main PID: 2562 (telegraf)
          Tasks: 17 (limit: 77021)
         Memory: 42.2M
         CGroup: /system.slice/telegraf.service
                         └─2562 /usr/bin/telegraf -config /etc/telegraf/telegraf.
conf -config-directory /etc/telegraf/telegraf.d

...I! Loaded inputs: swap system cpu disk diskio kernel mem processes
...I! Loaded outputs: http prometheus_client
...I! [agent] Config: Interval:10s, Quiet:false, Hostname:"workload", Flush
Interval:10s
...I! [outputs.prometheus_client] Listening on http://127.0.0.1:9273/metrics
```

Verify that it is collecting metrics by connecting to Telegraf's web interface:

```
workload:~# wget -O- http://workload:9273/metrics

# HELP cpu_usage_guest Telegraf collected metric
# TYPE cpu_usage_guest gauge
cpu_usage_guest{cpu="cpu-total",host="workload"} 0
cpu_usage_guest{cpu="cpu0",host="workload"} 0
cpu_usage_guest{cpu="cpu1",host="workload"} 0
cpu_usage_guest{cpu="cpu10",host="workload"} 0
...
cpu_usage_idle{cpu="cpu-total",host="workload"} 92.74914376428686
cpu_usage_idle{cpu="cpu0",host="workload"} 86.72897196325539
cpu_usage_idle{cpu="cpu1",host="workload"} 90.11857707405758
cpu_usage_idle{cpu="cpu10",host="workload"} 95.95141700494543
```

### Set up the Monitor node

Now let's create the Monitor. As before, we'll be using LXD as the container technology but feel free to adapt these steps to your chosen alternative:

```
$ lxc launch ubuntu:20.10 monitor
Creating monitor
Starting monitor
$ lxc exec monitor -- bash
monitor:~#
```

Make a note of the newly created container's IP address, which we'll need later on;

```
monitor:~# ip addr | grep 'inet .* global'
inet 10.69.244.104/24 brd 10.69.244.255 scope global dynamic eth0
```

Verify the Workload's Telegraf instance can be reached from the Monitor:

```
monitor:~# wget -O- http://workload:9273/metrics
```

We'll be setting up a few components to run on this node using their respective Snap packages. LXD images should normally have snap pre-installed, but if not, install it manually:

```
monitor:~# apt install snapd
```

### Install Prometheus

Prometheus will be our data manager. It collects data from external sources – Telegraf in our case – and distributes it to various destinations such as email/pager alerts, web UIs, API clients, remote storage services, etc. We'll get into those shortly.

Let's install Prometheus itself, and the Prometheus Alertmanager plugin for alerts, along with the required dependencies:

```
monitor:~# snap install prometheus
monitor:~# snap install prometheus-alertmanager
```

The snap will automatically configure and start the service. To verify this, run:

```
monitor:~# snap services
Service                Startup  Current   Notes
lxd.activate           enabled  inactive  -
lxd.daemon             enabled  inactive  socket-activated
prometheus.prometheus  enabled  active    -
prometheus-alertmanager.alertmanager  enabled  active    -
```

Verify that Prometheus is listening on the port as we expect:

```
visualizer:~# ss -tulpn | grep prometheus
tcp    LISTEN  0      128                    *:9090              *:*
users:(("prometheus",pid=618,fd=8))
```

`journalctl` can be also used to review the state of Snap services if more detail is needed. For example, to see where Prometheus is loading its config from:

```
monitor:~# journalctl | grep "prometheus.*config"
...
...msg="Completed loading of configuration file" filename=/var/snap/prometheus/32/
prometheus.yml
```

Although the file name points to a specific Snap revision (`32`, in this case), we can use the generic config file `/var/snap/prometheus/current/prometheus.yml` here in order to make things more general. Edit this config file to register the targets we'll be reading data from. This will go under the `scrape_configs` section of the file:

```
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped
from this config.
  - job_name: 'prometheus'

        # metrics_path defaults to '/metrics'
        # scheme defaults to 'http'.

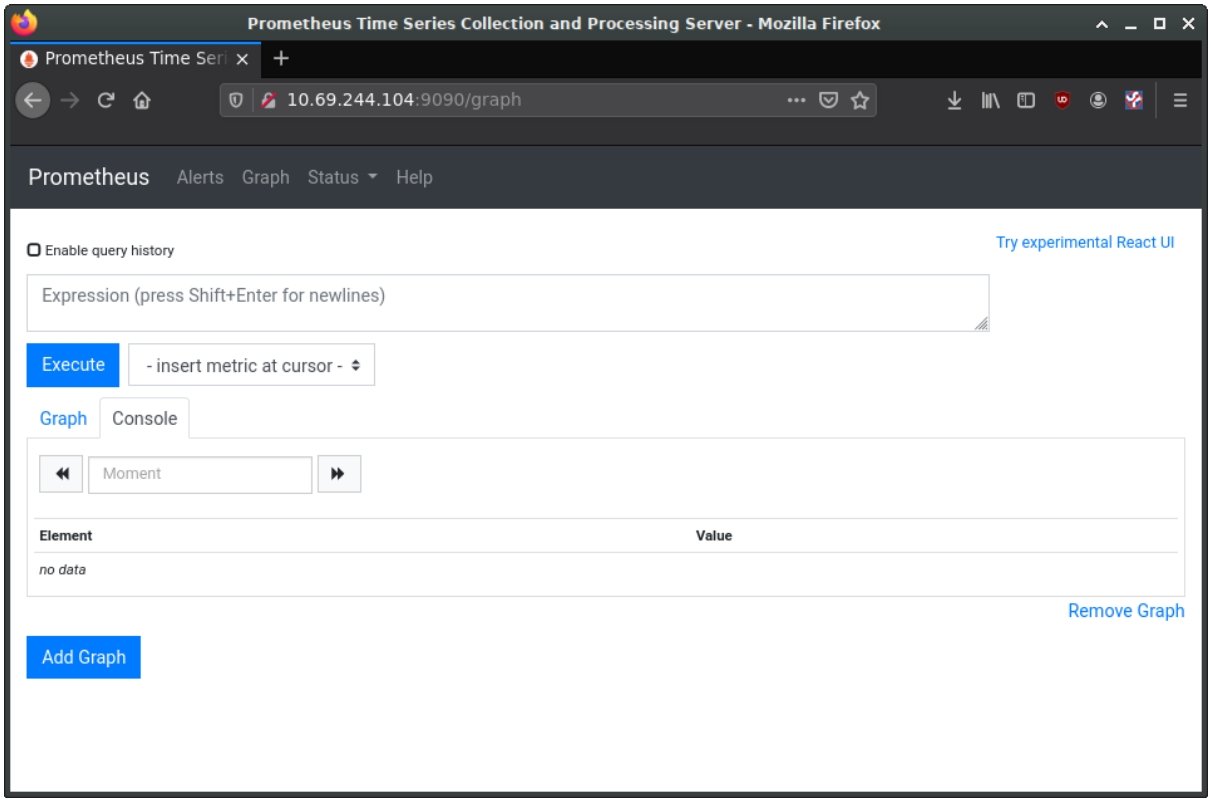
        static_configs:
        - targets: ['localhost:9090']

  - job_name: 'telegraf'
        static_configs:
        - targets: ['workload:9273']
```

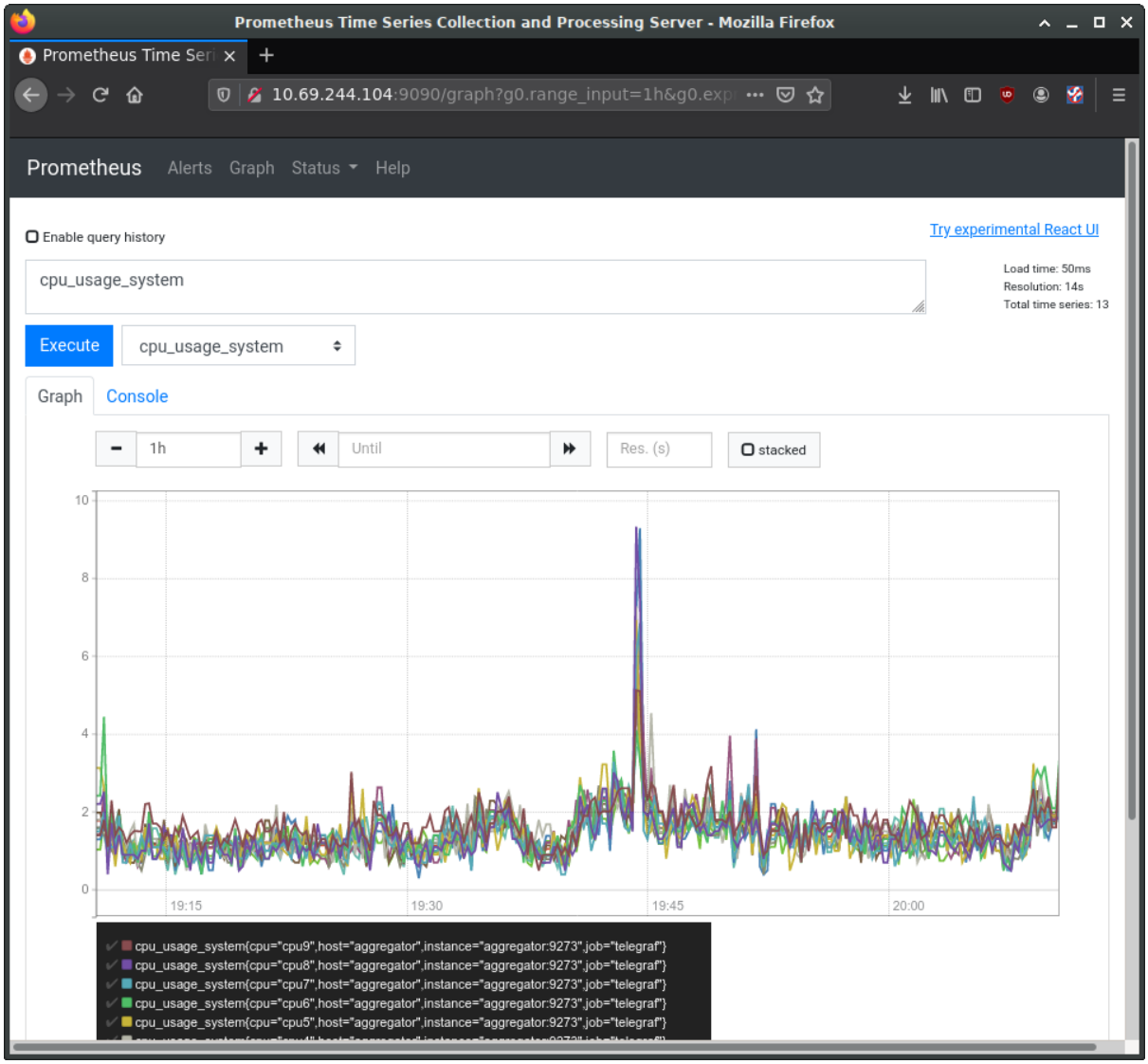
Then restart Prometheus:

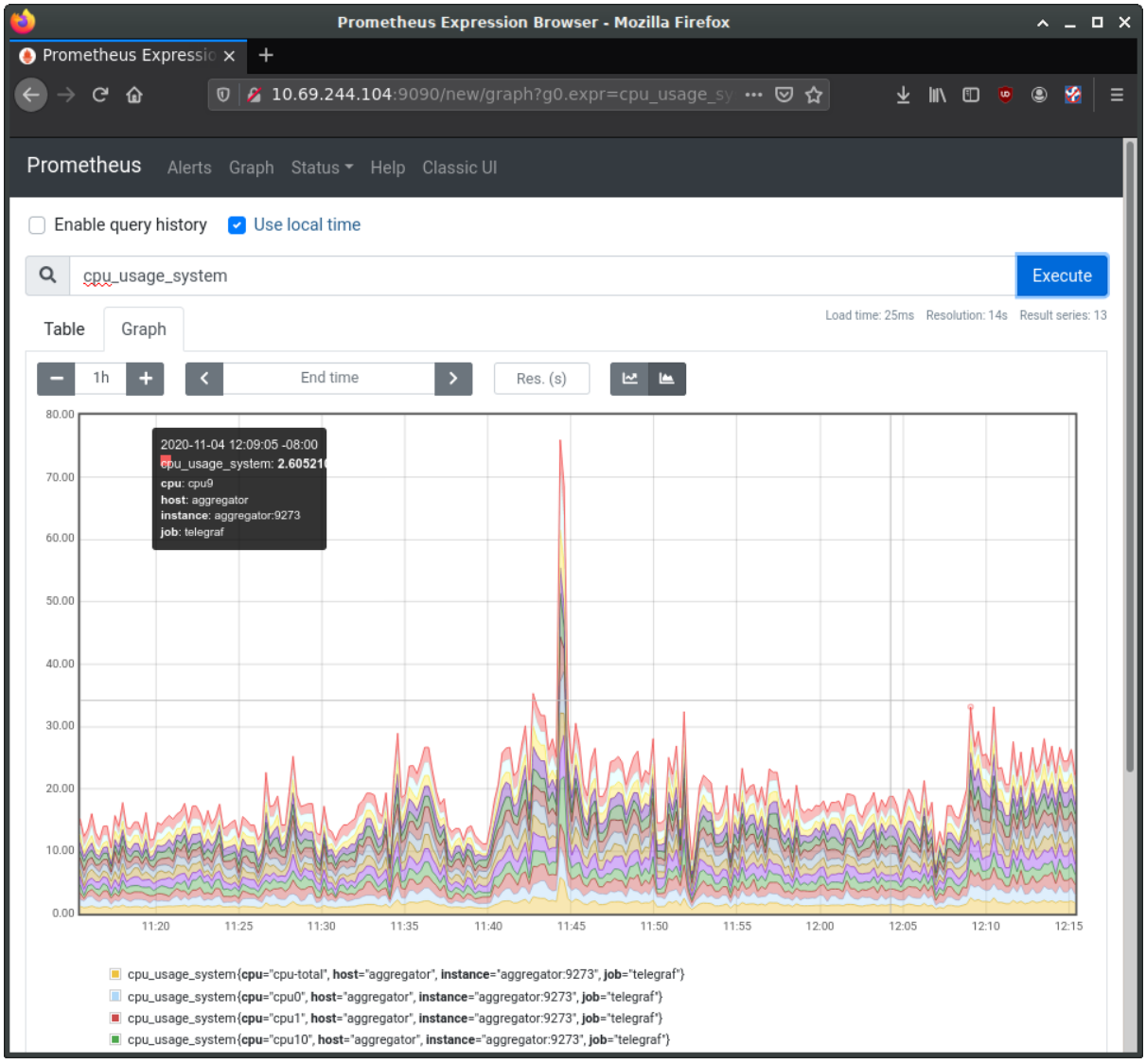
```
monitor:~# snap restart prometheus
```

While we'll be using Grafana for visualisation, Prometheus also has a web interface for viewing and interacting with the collected data. At this stage, we can load it to verify that our setup is working properly. In a web browser, navigate to the Monitor's IP address, and port `9090`. You should see Prometheus' interface, as in the following image:




In the entry box, enter `cpu_usage_system`, select the "Graph" tab and click "Execute". This should show a graph of our collected CPU data so far. Prometheus also has a secondary web UI using `React.js`.


Prometheus Time Series Collection and Processing Server - Mozilla Firefox
Prometheus Time Seri
10.69.244.104:9090/graph?g0.range_input=1h&g0.exp
Prometheus
Alerts
Graph
Status
Help
Enable query history
Try experimental React UI
cpu_usage_system
Load time: 50ms
Resolution: 14s
Total time series: 13
Execute
cpu_usage_system
Graph
Console
1h
Until
Res. (s)
stacked
10
8
6
4
2
0
19:15
19:30
19:45
20:00
cpu_usage_system{cpu="cpu9",host="aggregator",instance="aggregator:9273",job="telegraf"}
cpu_usage_system{cpu="cpu8",host="aggregator",instance="aggregator:9273",job="telegraf"}
cpu_usage_system{cpu="cpu7",host="aggregator",instance="aggregator:9273",job="telegraf"}
cpu_usage_system{cpu="cpu6",host="aggregator",instance="aggregator:9273",job="telegraf"}
cpu_usage_system{cpu="cpu5",host="aggregator",instance="aggregator:9273",job="telegraf"}




## Configure Alertmanager

Let's tackle the Alert Manager next. Edit `/var/snap/prometheus/current/prometheus.yml` again, adding the following to the `alerting` and `rules_files` sections:

```
## /var/snap/prometheus/current/prometheus.yml
#...
# Alertmanager configuration
alerting:
  alertmanagers:
  - static_configs:
        - targets:
          - 127.0.0.1:9093
rule_files:
  - 'alerts.yml'
```

Now create `/var/snap/prometheus/current/alerts.yml` with the following contents:

```
## /var/snap/prometheus/current/alerts.yml
```

(continues on next page)

(continued from previous page)

```
groups:
- name: demo-alerts
  rules:
  - alert: HighLoad
        expr: node_load1 > 2.0
        for: 60m
        labels:
          severity: normal
        annotations:
          description: '{{ $labels.instance }} of job {{ $labels.job }} is under
high load.'
          summary: Instance {{ $labels.instance }} under high load.
          value: '{{ $value }}'

  - alert: InstanceDown
        expr: up == 0
        for: 5m
        labels:
          severity: major
        annotations:
          summary: "Instance {{ $labels.instance }} down"
          description: "{{ $labels.instance }} of job {{ $labels.job }} has been
down for more than 5 minutes."
```

This adds two alerts: one for high processor load, and one to report if the node has been unreachable for over 5 minutes. We're considering high CPU to be a load of 2 or higher for an hour; this would need to be set to something more sensible for the style of workloads your production system experiences.

With the alerts themselves now defined, we next need to instruct Alertmanager how to handle them. There is a sample configuration installed to `/var/snap/prometheus-alertmanager/current/alertmanager.yml`, however it's full of example data. Instead, replace it entirely with this content:

```
## /var/snap/prometheus-alertmanager/current/alertmanager.yml
global:
  resolve_timeout: 5m

route:
  group_by: ['alertname']
  group_wait: 10s
  group_interval: 10s
  repeat_interval: 1h

inhibit_rules:
  - source_match:
          severity: 'critical'
        target_match:
          severity: 'warning'
        equal: ['alertname', 'dev', 'instance']
```

Restart Alertmanager after making the configuration change:

```
workload:~# snap restart prometheus-alertmanager
```

### Install Grafana

Grafana provides our main dashboard, from which we can generate graphs and other visuals to study the collected metrics. Grafana can read its data directly from log files, but we'll focus on using Prometheus as its principle data source. Grafana is available as a Snap and can be installed like this:

```
monitor:~# snap install grafana
grafana 6.7.4 from Alvaro Uría (aluria) installed
```

It uses port `3000`:

```
# ss -tulpn | grep grafana
tcp    LISTEN  0      128                    *:3000              *:*
users:(("grafana-server",pid=1449,fd=10))
```

We next need to know where it expects its configuration:

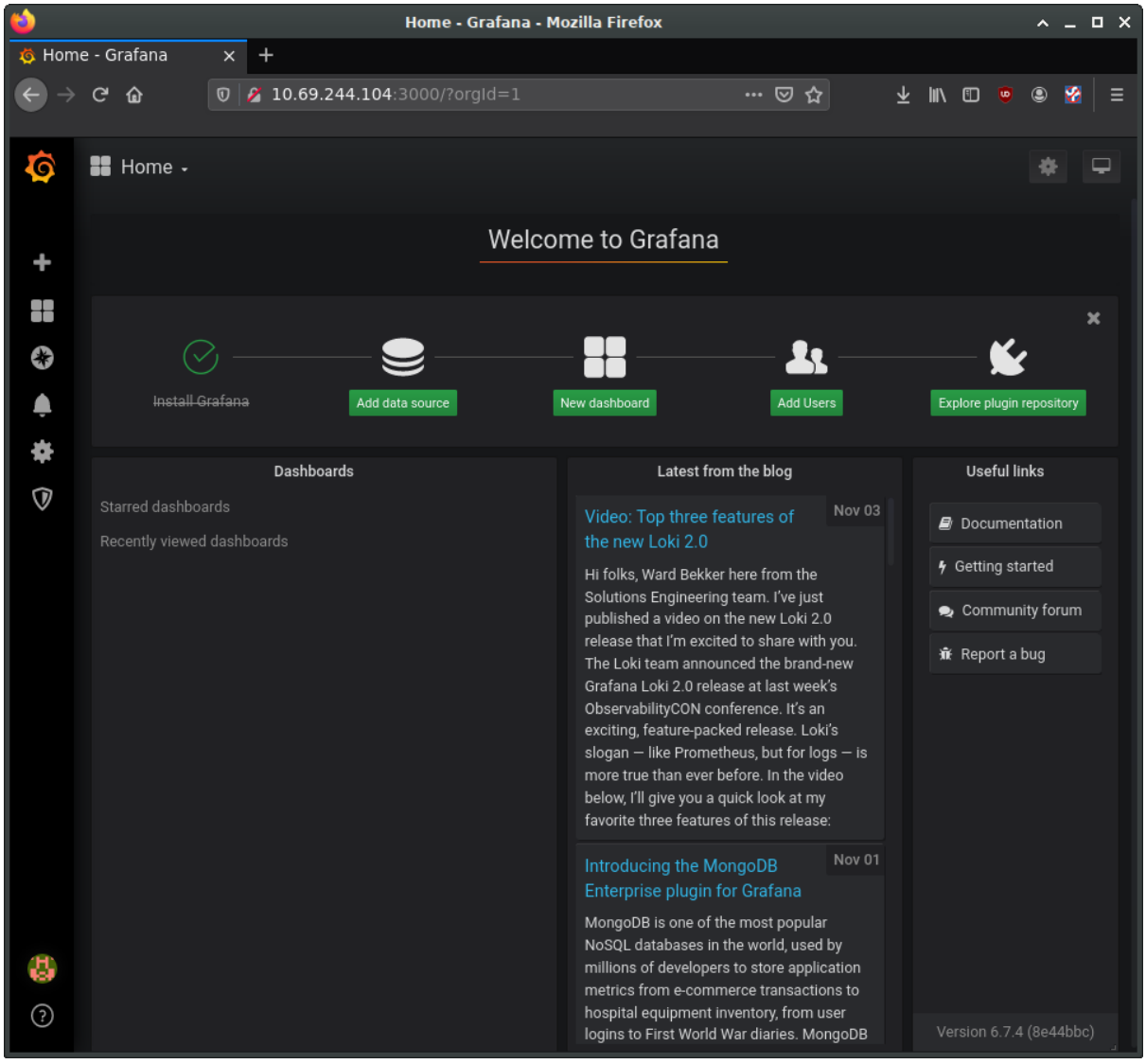
```
monitor:~# journalctl | grep "grafana.*conf"
... msg="Config loaded from" logger=settings file=/snap/grafana/36/conf/defaults.
ini
... msg="Config overridden from Environment variable" logger=settings var="GF_
PATHS_PROVISIONING=/var/snap/grafana/common/conf/provisioning"
... error="open /var/snap/grafana/common/conf/provisioning/datasources: no such
file or directory"
...
```

We can see it is getting its defaults from `/snap/grafana/36/conf/`, but `/snap/` is a read-only directory and therefore we cannot edit the file. Instead, we should put our customisations inside `/var/snap/grafana/36/conf/grafana.ini`. You can also use the generic path `/var/snap/grafana/current/conf/grafana.ini`.

For a production installation, the `defaults.ini` has numerous parameters we'd want to customise for our site, however for the demo we'll accept all the defaults. We do need to configure our data sources, but can do this via the web interface:

```
$ firefox http://10.69.244.104:3000
```

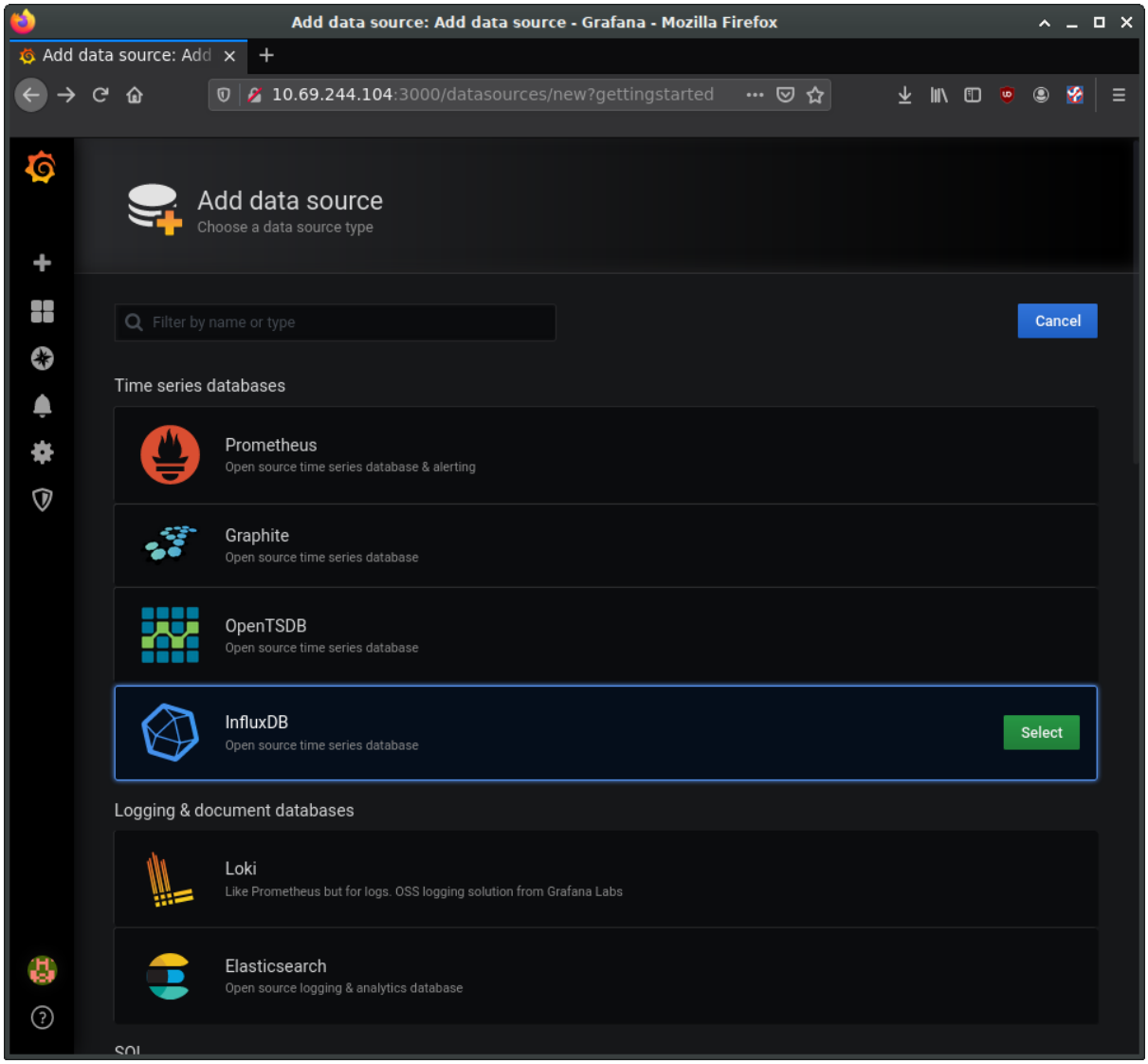
Log in with 'admin' and 'admin' as the username and password. This should bring you to the main Grafana page, where you can find links to tutorials and documentation. Delete any example data sources and/or dashboards.

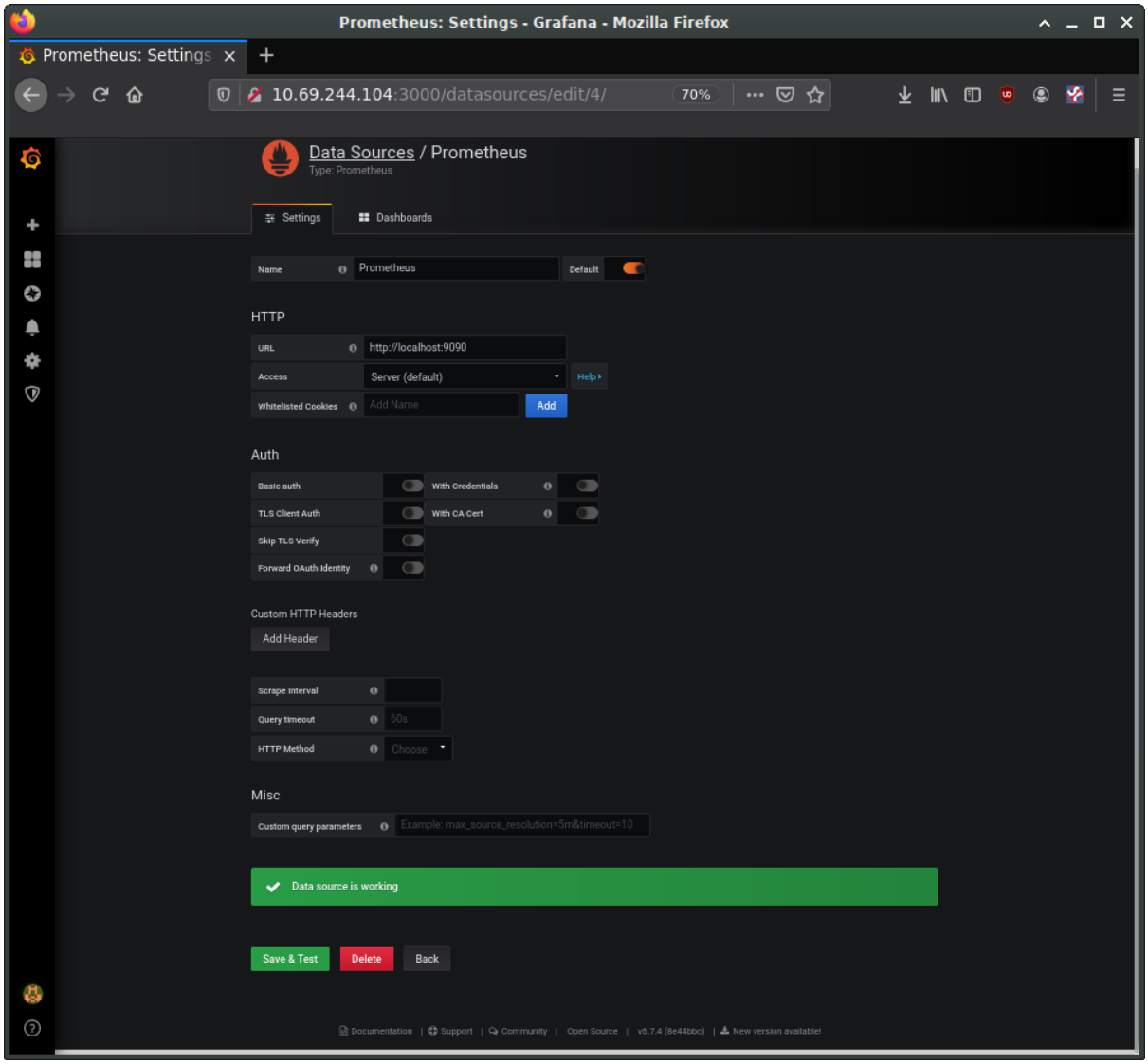



Select the button to add a new data source and select “Prometheus”. On the “Data Sources / Prometheus” edit page, set:

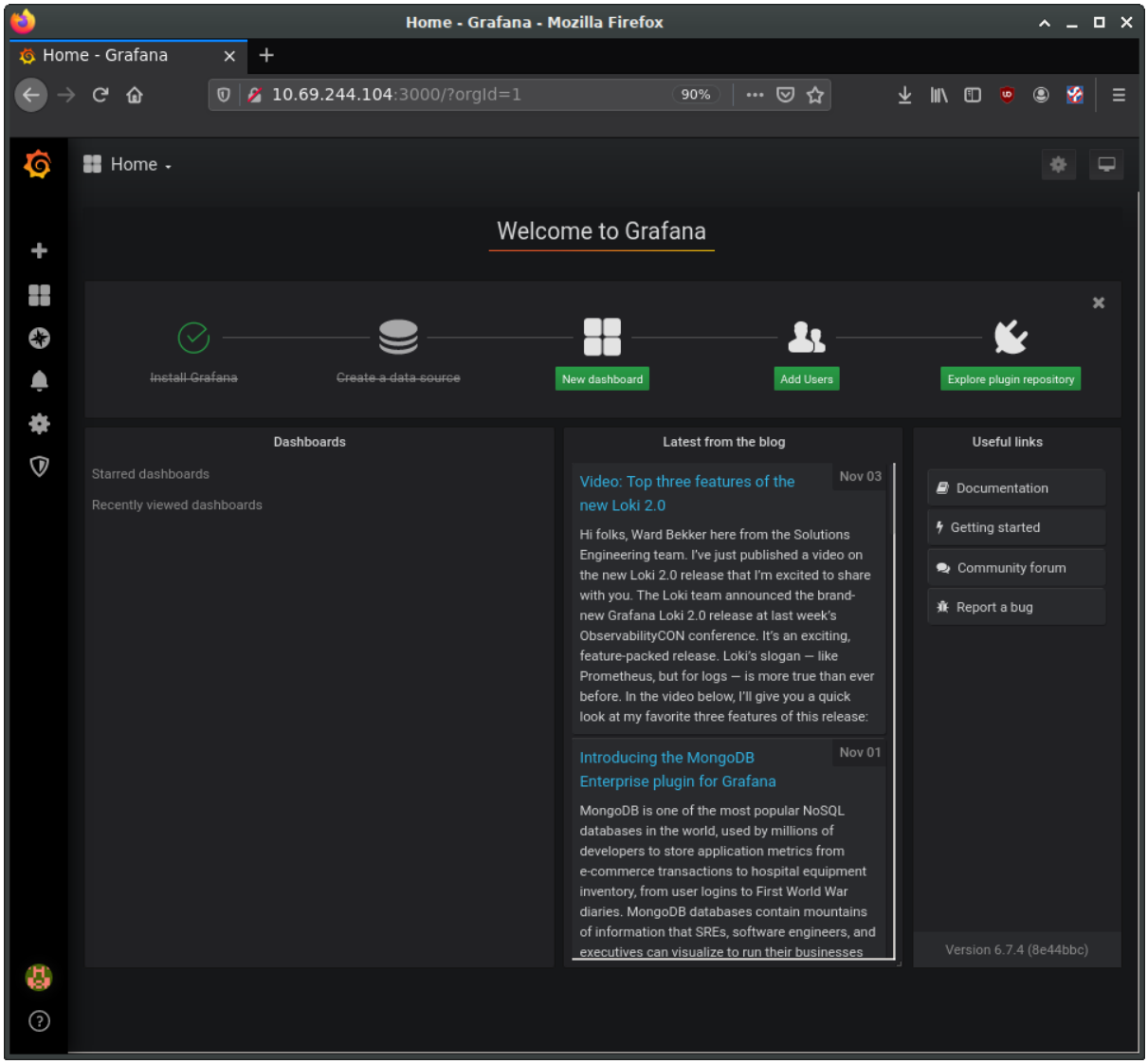
- the name to Prometheus
- the URL to `http://localhost:9090`
- ‘Access’ to “Server (default)” to make Grafana pull data from the Prometheus service we set up.

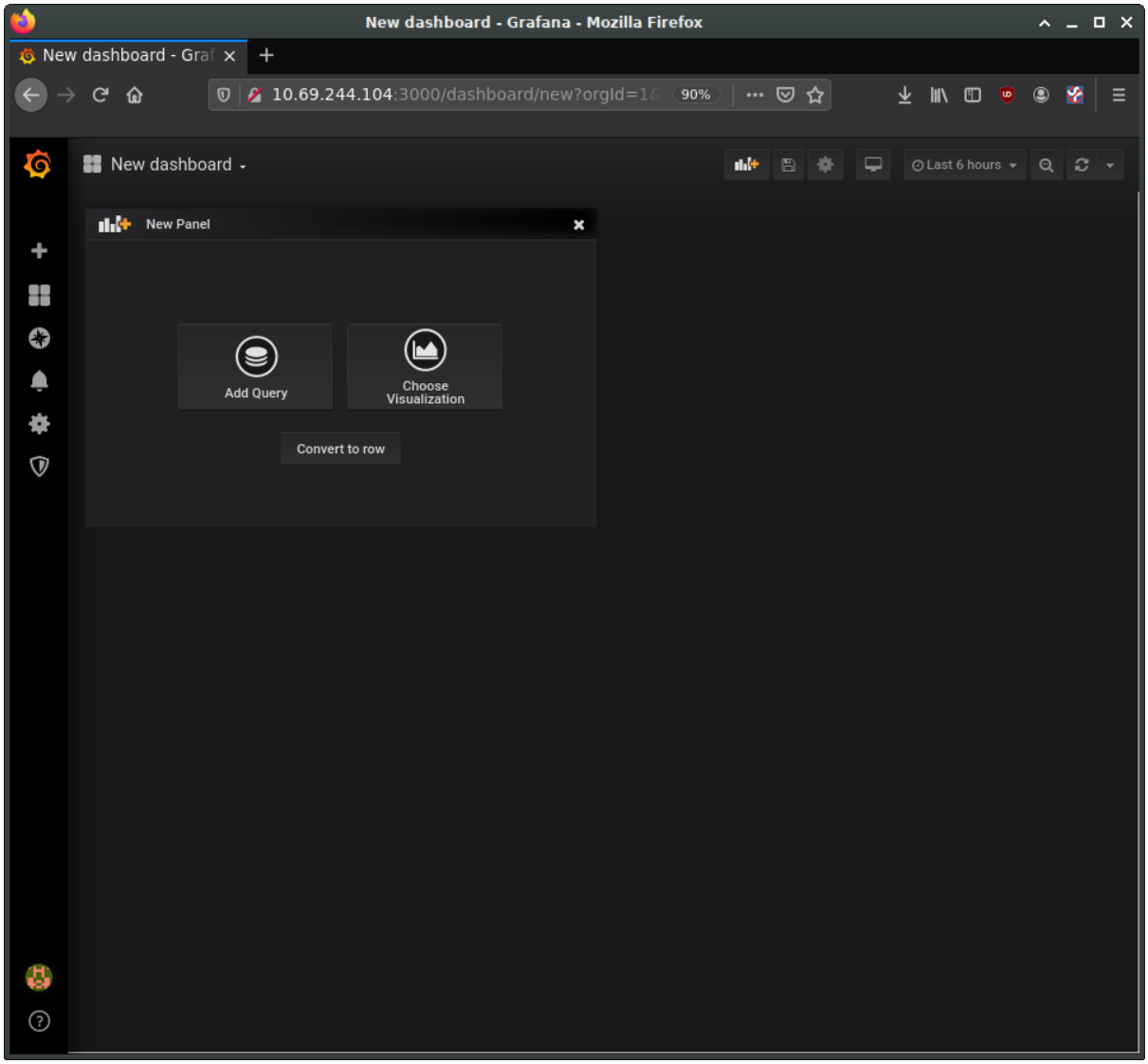
The remaining settings can be left as defaults. Click “Save & Test”.

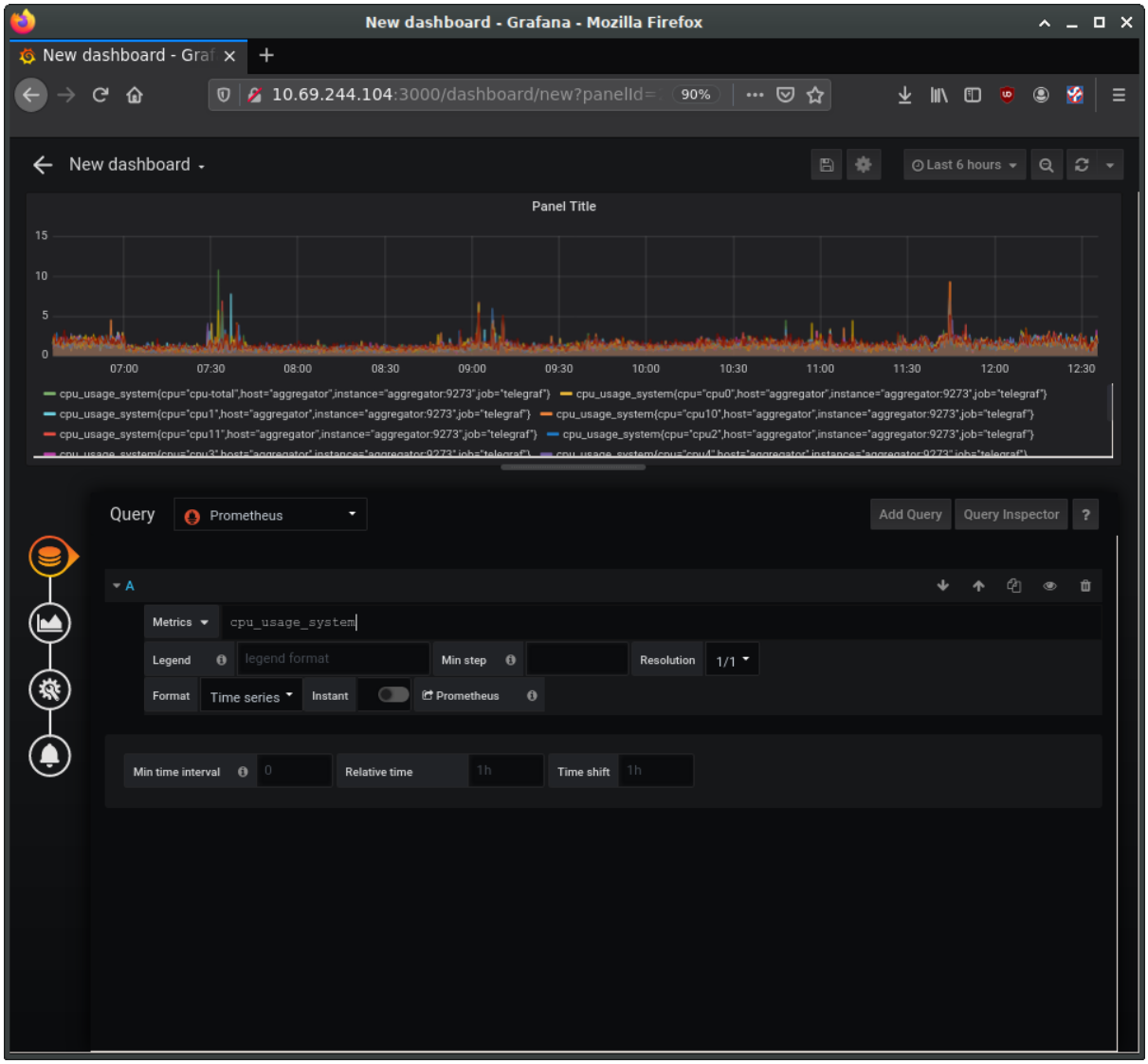

Add data source: Add data source - Grafana - Mozilla Firefox
Add data source: Add
10.69.244.104:3000/datasources/new?gettingstarted
Add data source
Choose a data source type
Filter by name or type
Cancel
Time series databases
Prometheus
Open source time series database & alerting
Graphite
Open source time series database
OpenTSDB
Open source time series database
InfluxDB
Open source time series database
Select
Logging & document databases
Loki
Like Prometheus but for logs. OSS logging solution from Grafana Labs
Elasticsearch
Open source logging & analytics database



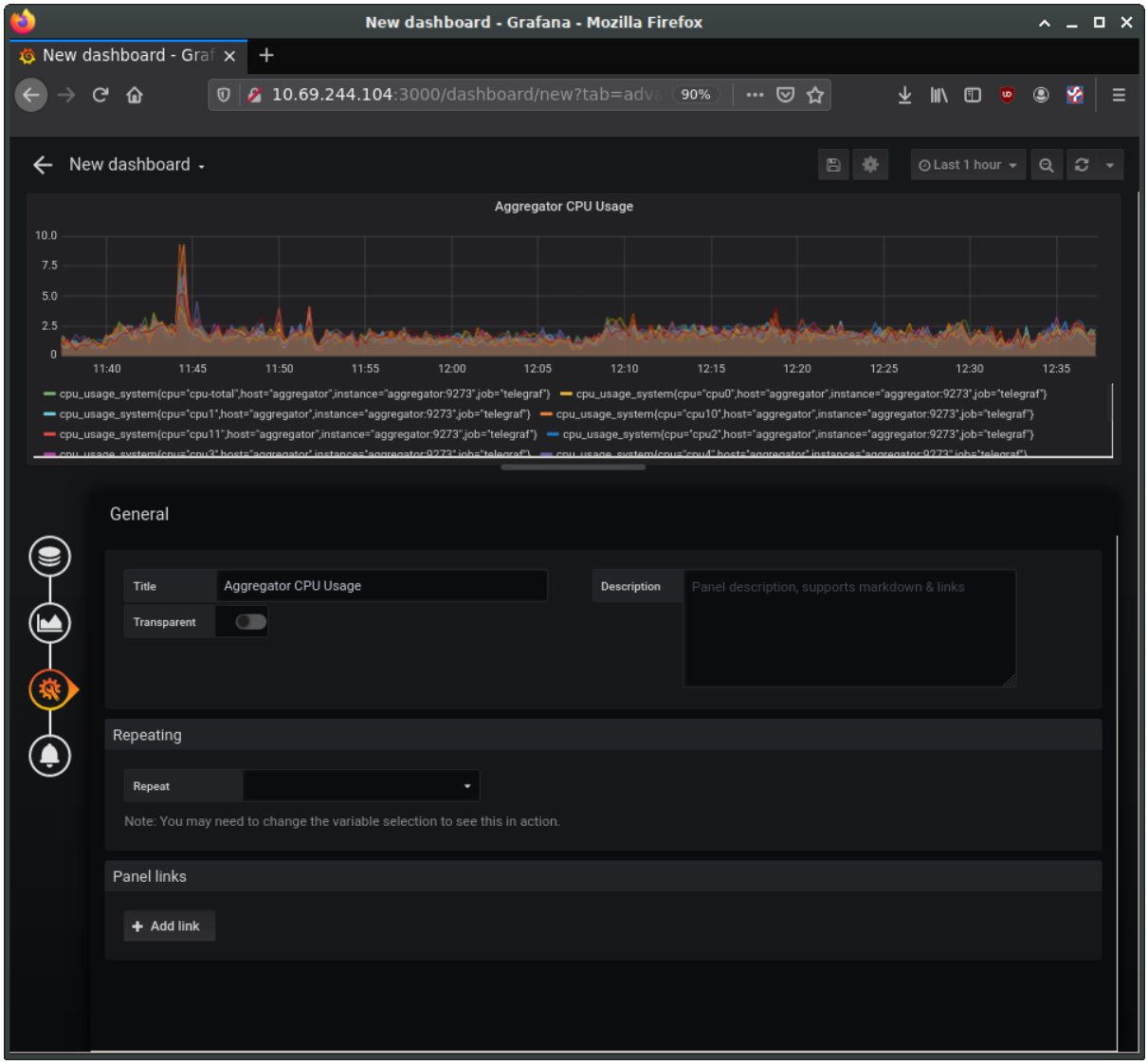

Returning to the Grafana home page, next set up a “New Dashboard”. A dashboard can hold one or more panels, each of which can be connected to one or more data queries. Let’s add a panel for CPU data. For the query, enter “cpu_usage_system” in the Metrics field.

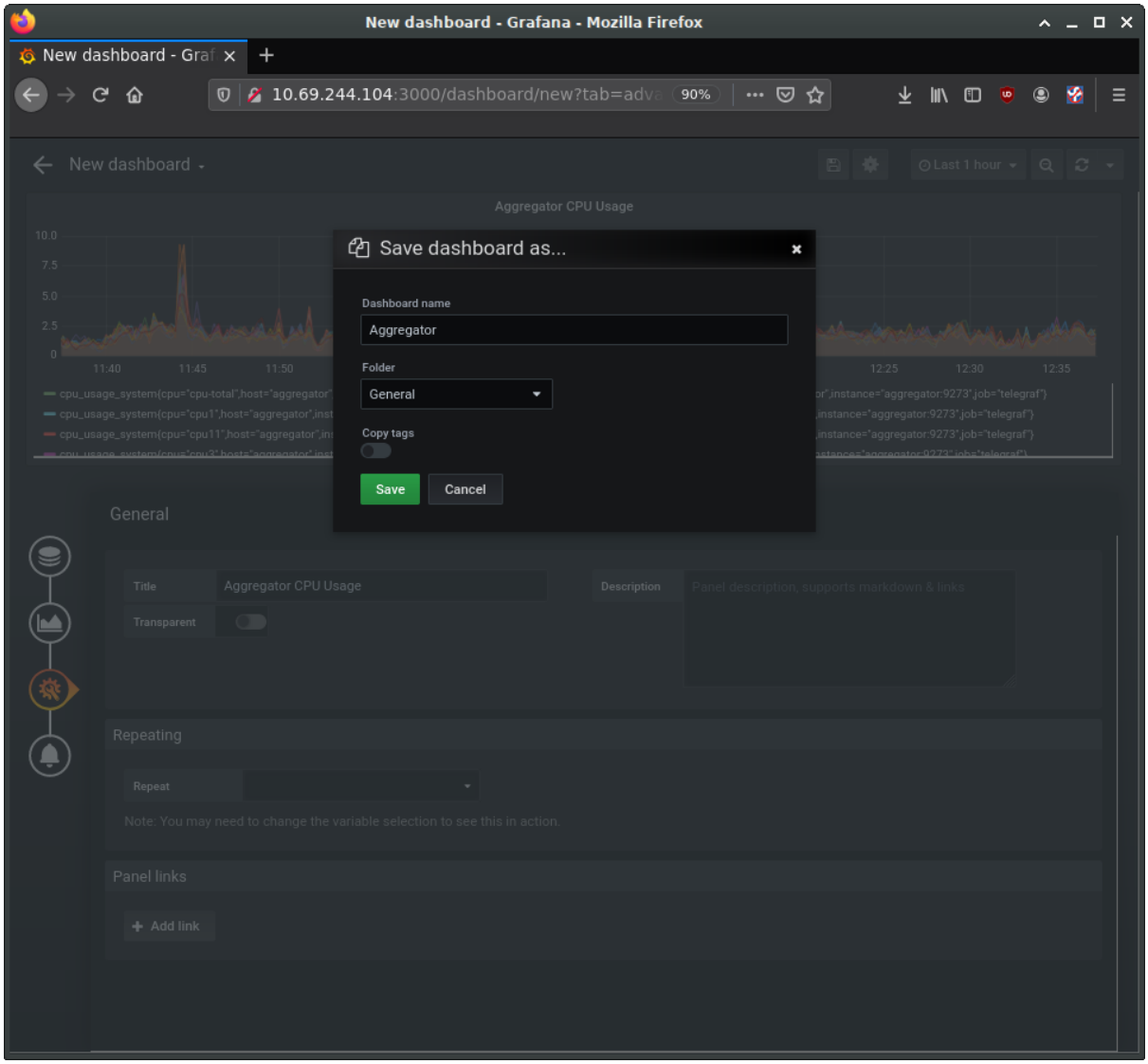

Home - Grafana - Mozilla Firefox
Home - Grafana
10.69.244.104:3000/?orgId=1
90%
Home
Welcome to Grafana
Install Grafana
Create a data source
New dashboard
Add Users
Explore plugin repository
Dashboards
Starred dashboards
Recently viewed dashboards
Latest from the blog
Video: Top three features of the new Loki 2.0
Nov 03
Hi folks, Ward Bekker here from the Solutions Engineering team. I've just published a video on the new Loki 2.0 release that I'm excited to share with you. The Loki team announced the brand-new Grafana Loki 2.0 release at last week's ObservabilityCON conference. It's an exciting, feature-packed release. Loki's slogan — like Prometheus, but for logs — is more true than ever before. In the video below, I'll give you a quick look at my favorite three features of this release:
Introducing the MongoDB Enterprise plugin for Grafana
Nov 01
MongoDB is one of the most popular NoSQL databases in the world, used by millions of developers to store application metrics from e-commerce transactions to hospital equipment inventory, from user logins to First World War diaries. MongoDB databases contain mountains of information that SREs, software engineers, and executives can visualize to run their businesses
Useful links
Documentation
Getting started
Community forum
Report a bug
Version 6.7.4 (8e44bbc)


New dashboard - Grafana - Mozilla Firefox
New dashboard - Graf
10.69.244.104:3000/dashboard/new?orgId=1&
90%
New dashboard
Last 6 hours
New Panel
Add Query
Choose Visualization
Convert to row

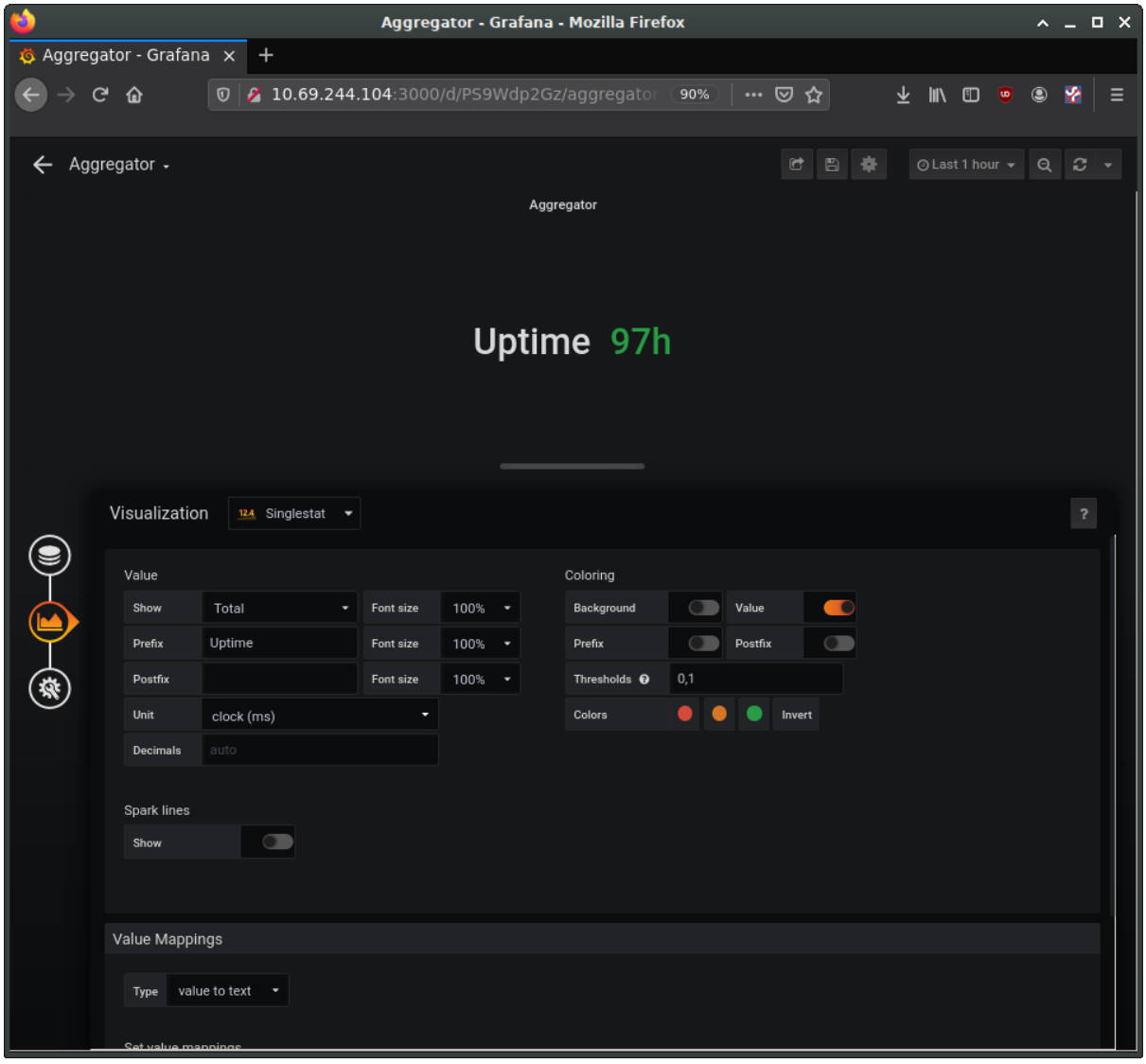



On the left you can see four buttons to configure four elements of the panel: data source, visualisation, general settings, and alerts. The general settings page allows us to set a title for the panel, for instance. Make any other customisations you want, and then save the dashboard using the save icon at the top of the page.

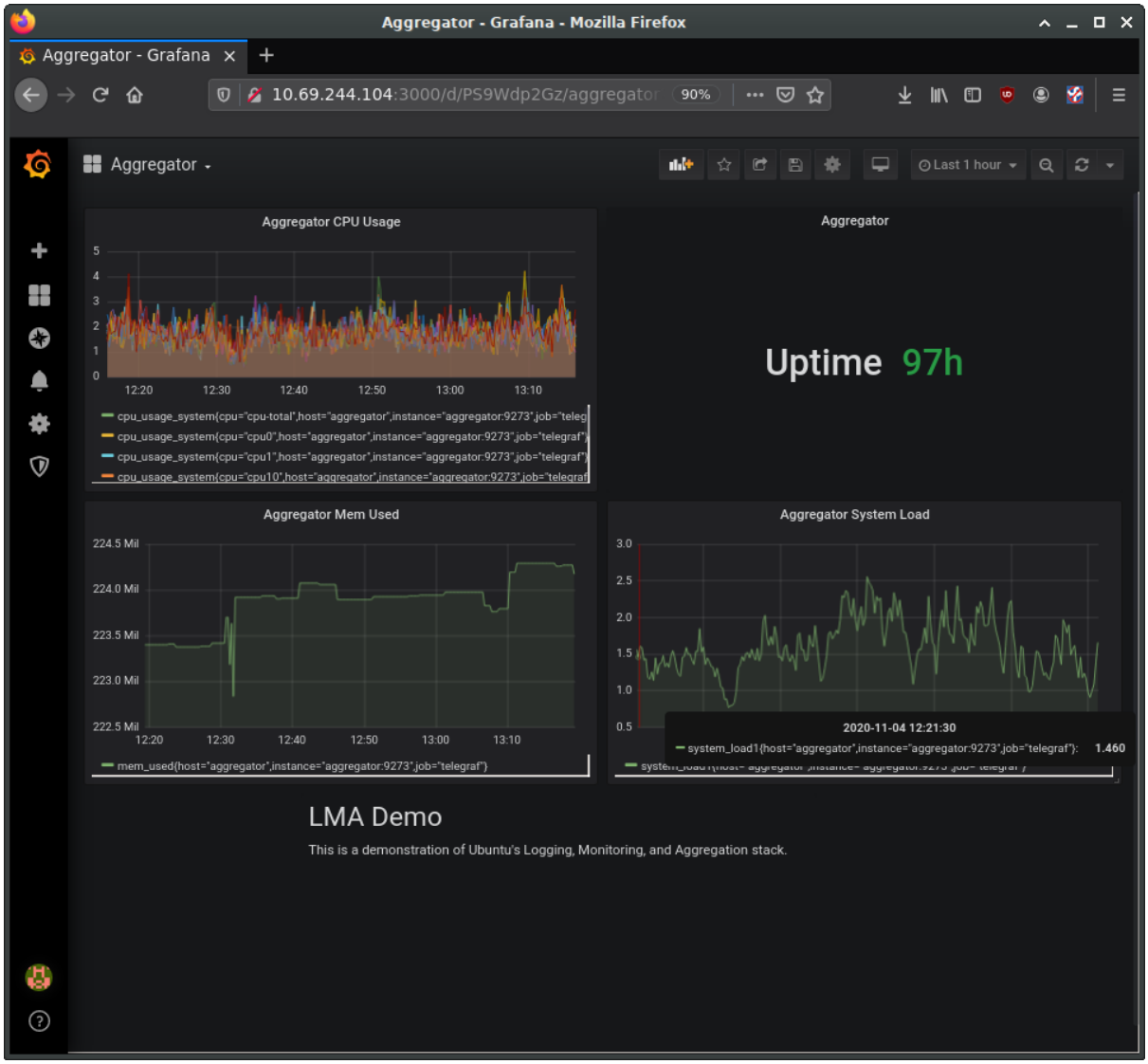

New dashboard - Grafana - Mozilla Firefox
New dashboard - Graf
10.69.244.104:3000/dashboard/new?tab=adva
90%
New dashboard
Last 1 hour
Aggregator CPU Usage
10.0
7.5
5.0
2.5
0
11:40
11:45
11:50
11:55
12:00
12:05
12:10
12:15
12:20
12:25
12:30
12:35
cpu_usage_system{cpu="cpu-total",host="aggregator",instance="aggregator:9273",job="telegraf"}
cpu_usage_system{cpu="cpu0",host="aggregator",instance="aggregator:9273",job="telegraf"}
cpu_usage_system{cpu="cpu1",host="aggregator",instance="aggregator:9273",job="telegraf"}
cpu_usage_system{cpu="cpu10",host="aggregator",instance="aggregator:9273",job="telegraf"}
cpu_usage_system{cpu="cpu11",host="aggregator",instance="aggregator:9273",job="telegraf"}
cpu_usage_system{cpu="cpu2",host="aggregator",instance="aggregator:9273",job="telegraf"}
General
Title
Aggregator CPU Usage
Transparent
Description
Panel description, supports markdown & links
Repeating
Repeat
Note: You may need to change the variable selection to see this in action.
Panel links
Add link




Using the same procedure, add additional panels for processor load and memory usage. Panels can be used to present other types of data as well, such as numerical indicators, logs, newsfeeds, or markdown-formatted documentation. For example, you can add a panel to display the system uptime, such as in the following image:




Try also adding a panel with the “Text” visualisation option for entering descriptive text about our demo. Save, and then view the final dashboard:




If you do not need a full LMA stack, there are some other supported tools that can provide similar capabilities.

## How to install and configure Logwatch

Logs are an invaluable source of information about problems that may arise in your server. Logwatch keeps an eye on your logs for you, flags items that may be of interest, and reports them via email.

## Install Logwatch

Install `logwatch` using the following command:

```
sudo apt install logwatch
```

You will also need to manually create a temporary directory in order for it to work:

```
sudo mkdir /var/cache/logwatch
```

### Configure `logwatch`

Logwatch's default configuration is kept in `/usr/share/logwatch/default.conf/logwatch.conf`. However, configuration changes made directly to that file can be overwritten during updates, so instead the file should be copied into `/etc` and modified there:

```
sudo cp /usr/share/logwatch/default.conf/logwatch.conf /etc/logwatch/conf/
```

With your favorite editor, open `/etc/logwatch/conf/logwatch.conf`. The uncommented lines indicate the default configuration values. First, lets customise some of the basics:

```
Output = mail
MailTo = me@mydomain.org
MailFrom = logwatch@host1.mydomain.org
Detail = Low
Service = All
```

This assumes you've already set up mail services on `host1` that will allow mail to be delivered to your `me@mydomain.org` address. These emails will be addressed from `logwatch@host1.mydomain.org`.

The **Detail** level defines how much information is included in the reports. Possible values are: `Low`, `Medium`, and `High`.

Logwatch will then monitor logs for all services on the system, unless specified otherwise with the **Service** parameter. If there are undesired services included in the reports, they can be disabled by removing them with additional **Service** fields. E.g.:

```
Service = "-http"
Service = "-eximstats"
```

Next, run `logwatch` manually to verify your configuration changes are valid:

```
sudo logwatch --detail Low --range today
```

The report produced should look something like this:

```
################### Logwatch 7.4.3 (12/07/16) ####################
       Processing Initiated: Fri Apr 24 16:58:14 2020
       Date Range Processed: today
                             ( 2020-Apr-24 )
                             Period is day.
       Detail Level of Output: 0
       Type of Output/Format: stdout / text
       Logfiles for Host: `host1.mydomain.org`
##################################################################

-------------------- pam_unix Begin ------------------------

sudo:
   Sessions Opened:
      bryce -> root: 1 Time(s)
```

(continues on next page)

(continued from previous page)

```
---------------------- pam_unix End ------------------------


--------------------- rsnapshot Begin ------------------------

ERRORS:
    /usr/bin/rsnapshot hourly: completed, but with some errors: 5 Time(s)
    /usr/bin/rsync returned 127 while processing root@host2:/etc/: 5 Time(s)
    /usr/bin/rsync returned 127 while processing root@host2:/home/: 5 Time(s)
    /usr/bin/rsync returned 127 while processing root@host2:/proc/uptime: 5
Time(s)
    /usr/bin/rsync returned 127 while processing root@host3:/etc/: 5 Time(s)
    /usr/bin/rsync returned 127 while processing root@host3:/home/: 5 Time(s)
    /usr/bin/rsync returned 127 while processing root@host3:/proc/uptime: 5
Time(s)


---------------------- rsnapshot End -------------------------


--------------------- SSHD Begin ------------------------


Users logging in through sshd:
   bryce:
      192.168.1.123 (`host4.mydomain.org`): 1 time

---------------------- SSHD End ------------------------


--------------------- Sudo (secure-log) Begin ------------------------


bryce => root
\-------------
/bin/bash                       -   1 Time(s).

---------------------- Sudo (secure-log) End ------------------------


--------------------- Disk Space Begin ------------------------

Filesystem      Size  Used Avail Use% Mounted on
/dev/sdc1       220G   19G  190G   9% /
/dev/loop1      157M  157M     0 100% /snap/gnome-3-28-1804/110
/dev/loop11     1.0M  1.0M     0 100% /snap/gnome-logs/81
/dev/md5        9.1T  7.3T  1.8T  81% /srv/Products
/dev/md6        9.1T  5.6T  3.5T  62% /srv/Archives
```

(continues on next page)

(continued from previous page)

```
/dev/loop14     3.8M  3.8M     0 100% /snap/gnome-system-monitor/127
/dev/loop17      15M   15M     0 100% /snap/gnome-characters/399
/dev/loop18     161M  161M     0 100% /snap/gnome-3-28-1804/116
/dev/loop6       55M   55M     0 100% /snap/core18/1668
/dev/md1        1.8T  1.3T  548G  71% /srv/Staff
/dev/md0        3.6T  3.5T   84G  98% /srv/Backup
/dev/loop2      1.0M  1.0M     0 100% /snap/gnome-logs/93
/dev/loop5       15M   15M     0 100% /snap/gnome-characters/495
/dev/loop8      3.8M  3.8M     0 100% /snap/gnome-system-monitor/135
/dev/md7        3.6T  495G  3.0T  15% /srv/Customers
/dev/loop9       55M   55M     0 100% /snap/core18/1705
/dev/loop10      94M   94M     0 100% /snap/core/8935
/dev/loop0       55M   55M     0 100% /snap/gtk-common-themes/1502
/dev/loop4       63M   63M     0 100% /snap/gtk-common-themes/1506
/dev/loop3       94M   94M     0 100% /snap/core/9066

/srv/Backup (/dev/md0) => 98% Used. Warning. Disk Filling up.

---------------------- Disk Space End -------------------------


###################### Logwatch End #########################
```

### Further reading

- The Ubuntu `logwatch(8)` manpage contains many more detailed options.

### How to install and configure Munin

The monitoring of essential servers and services is an important part of system administration. This guide will show you how to set up Munin for performance monitoring.

In this example, we will use two servers with *hostnames*: `server01` and `server02`.

`Server01` will be set up with the `munin` package to gather information from the network. Using the `munin-node` package, `server02` will be configured to send information to `server01`.

### Prerequisites

Before installing Munin on `server01`, *Apache2 will need to be installed*. The default configuration is fine for running a `munin` server.

### Install `munin` and `munin-node`

First, on `server01` install the `munin` package. In a terminal enter the following command:

```
sudo apt install munin
```

Now on `server02`, install the `munin-node` package:

```
sudo apt install munin-node
```

### Configure `munin`

On `server01` edit the `/etc/munin/munin.conf` file, adding the IP address for `server02`:

```
## First our "normal" host.
[server02]
       address 172.18.100.101
```

> **Note**
>
> Replace `server02` and `172.18.100.101` with the actual hostname and IP address for your server.

### Configure `munin-node`

Next, configure `munin-node` on `server02`. Edit `/etc/munin/munin-node.conf` to allow access by `server01`:

```
allow ^172\.18\.100\.100$
```

> **Note**
>
> Replace `^172\.18\.100\.100$` with the IP address for your `munin` server.

Now restart `munin-node` on `server02` for the changes to take effect:

```
sudo systemctl restart munin-node.service
```

### Test the setup

In a browser, go to `http://server01/munin`, and you should see links to nice graphs displaying information from the standard **munin-plugins** for disk, network, processes, and system. However, it should be noted that since this is a new installation, it may take some time for the graphs to display anything useful.

### Additional Plugins

The `munin-plugins-extra` package contains performance checks and additional services such as *DNS*, *DHCP*, and Samba, etc. To install the package, from a terminal enter:

```
sudo apt install munin-plugins-extra
```

Be sure to install the package on both the server and node machines.

### References

- See the Munin website for more details.
- Specifically the Munin Documentation page includes information on additional plugins, writing plugins, etc.

## How to install and configure Nagios Core 3

> **Note**
>
> Nagios Core 3 has been deprecated and is now replaced by Nagios Core 4. The `nagios3` package was last supported in Bionic, so subsequent releases should use `nagios4` instead.

The monitoring of essential servers and services is an important part of system administration. This guide walks through how to install and configure Nagios Core 3 for availability monitoring.

The example in this guide uses two servers with *hostnames*: `server01` and `server02`.

`Server01` will be configured with Nagios to monitor services on itself and on `server02`, while `server02` will be configured to send data to `server01`.

### Install `nagios3` on `server01`

First, on `server01`, install the `nagios3` package by entering the following command into your terminal:

```
sudo apt install nagios3 nagios-nrpe-plugin
```

You will be asked to enter a password for the **nagiosadmin** user. The user's credentials are stored in `/etc/nagios3/htpasswd.users`. To change the nagiosadmin password, or add more users to the Nagios CGI scripts, use the `htpasswd` that is part of the `apache2-utils` package.

For example, to change the password for the nagiosadmin user, enter:

```
sudo htpasswd /etc/nagios3/htpasswd.users nagiosadmin
```

To add a user:

```
sudo htpasswd /etc/nagios3/htpasswd.users steve
```

### Install `nagios-nrpe-server` on `server02`

Next, on `server02` install the `nagios-nrpe-server` package. From a terminal on `server02` enter:

```
sudo apt install nagios-nrpe-server
```

> **Note**

NRPE allows you to execute local checks on remote hosts. There are other ways of accomplishing this through other Nagios plugins, as well as other checks.

### Configuration overview

There are a couple of directories containing Nagios configuration and check files.

- `/etc/nagios3`: Contains configuration files for the operation of the Nagios daemon, CGI files, hosts, etc.
- `/etc/nagios-plugins`: Contains configuration files for the service checks.
- `/etc/nagios`: On the remote host, contains the `nagios-nrpe-server` configuration files.
- `/usr/lib/nagios/plugins/`: Where the check binaries are stored. To see the options of a check use the `-h` option. For example: `/usr/lib/nagios/plugins/check_dhcp -h`

There are multiple checks Nagios can be configured to execute for any given host. For this example, Nagios will be configured to check disk space, *DNS*, and a MySQL *hostgroup*. The DNS check will be on `server02`, and the MySQL hostgroup will include both `server01` and `server02`.

**Note**

See these guides for details on setting up Apache, Domain Name Service, and MySQL.

Additionally, there are some terms that once explained will hopefully make understanding Nagios configuration easier:

- *Host*: A server, workstation, network device, etc. that is being monitored.
- *Host Group*: A group of similar hosts. For example, you could group all web servers, file server, etc.
- *Service*: The service being monitored on the host, such as HTTP, DNS, NFS, etc.
- *Service Group*: Allows you to group multiple services together. This is useful for grouping multiple HTTP for example.
- *Contact*: Person to be notified when an event takes place. Nagios can be configured to send emails, SMS messages, etc.

By default, Nagios is configured to check HTTP, disk space, SSH, current users, processes, and load on the **localhost**. Nagios will also ping check the **gateway**.

Large Nagios installations can be quite complex to configure. It is usually best to start small, with one or two hosts, to get things configured the way you want before expanding.

### Configure Nagios

#### Create host config file for server02

First, create a **host** configuration file for `server02`. Unless otherwise specified, run all these commands on `server01`. In a terminal enter:

```
sudo cp /etc/nagios3/conf.d/localhost_nagios2.cfg \
/etc/nagios3/conf.d/server02.cfg
```

> **Note**
>
> In all command examples, replace "server01", "server02", 172.18.100.100, and 172.18.100.101 with the host names and IP addresses of your servers.

### Edit the host config file

Next, edit /etc/nagios3/conf.d/server02.cfg:

```
define host{
        use                     generic-host  ; Name of host template to use
        host_name               server02
        alias                   Server 02
        address                 172.18.100.101
}

# check DNS service.
define service {
        use                             generic-service
        host_name                       server02
        service_description             DNS
        check_command                   check_dns!172.18.100.101
}
```

Restart the Nagios daemon to enable the new configuration:

```
sudo systemctl restart nagio3.service
```

### Add service definition

Now add a service definition for the MySQL check by adding the following to /etc/nagios3/conf.d/services_nagios2.cfg:

```
# check MySQL servers.
define service {
        hostgroup_name          mysql-servers
        service_description     MySQL
        check_command           check_mysql_cmdlinecred!nagios!secret!$HOSTADDRESS
        use                     generic-service
        notification_interval 0 ; set > 0 if you want to be renotified
}
```

A **mysql-servers** hostgroup now needs to be defined. Edit /etc/nagios3/conf.d/hostgroups_nagios2.cfg and add the following:

```
# MySQL hostgroup.
define hostgroup {
        hostgroup_name  mysql-servers
                alias           MySQL servers
                members         localhost, server02
        }
```

The Nagios check needs to authenticate to MySQL. To add a `nagios` user to MySQL enter:

```
mysql -u root -p -e "create user nagios identified by 'secret';"
```

> **Note**
>
> The `nagios` user will need to be added to all hosts in the **mysql-servers** hostgroup.

Restart nagios to start checking the MySQL servers.

sudo systemctl restart nagios3.service

### Configure NRPE

Lastly configure NRPE to check the disk space on *server02*.

On `server01` add the service check to `/etc/nagios3/conf.d/server02.cfg`:

```
# NRPE disk check.
define service {
        use                     generic-service
        host_name               server02
        service_description     nrpe-disk
        check_command           check_nrpe_1arg!check_all_disks!172.18.100.101
}
```

Now on `server02` edit `/etc/nagios/nrpe.cfg` changing:

```
allowed_hosts=172.18.100.100
```

And below, in the command definition area, add:

```
command[check_all_disks]=/usr/lib/nagios/plugins/check_disk -w 20% -c 10% -e
```

Finally, restart `nagios-nrpe-server`:

```
sudo systemctl restart nagios-nrpe-server.service
```

Also, on `server01` restart Nagios:

```
sudo systemctl restart nagios3.service
```

You should now be able to see the host and service checks in the Nagios CGI files. To access them, point a browser to `http://server01/nagios3`. You will then be prompted for the **nagiosadmin** username and password.

### Further reading

This section has just scratched the surface of Nagios' features. The `nagios-plugins-extra` and `nagios-snmp-plugins` contain many more service checks.

- For more information about Nagios, see the Nagios website.
- The Nagios Core Documentation and Nagios Core 3 Documentation may also be useful.
- They also provide a list of books related to Nagios and network monitoring.
- The Nagios Ubuntu Wiki page also has more details.

## How to use Nagios with Munin

> **Note**
>
> Nagios Core 3 has been deprecated and is now replaced by Nagios Core 4. The `nagios3` package was last supported in Bionic, so subsequent releases should use `nagios4` instead.

The monitoring of essential servers and services is an important part of system administration. Most network services are monitored for performance, availability, or both. This section will cover installation and configuration of Nagios 3 for availability monitoring alongside Munin for performance monitoring.

The examples in this section will use two servers with hostnames `server01` and `server02`. `Server01` will be configured with Nagios 3 to monitor services on both itself and `server02`. `Server01` will also be set up with the Munin package to gather information from the network. Using the `munin-node` package, `server02` will be configured to send information to `server01`.

### Install Nagios 3

#### On server01

First, on `server01`, install the `nagios3` package. In a terminal, enter:

```
sudo apt install nagios3 nagios-nrpe-plugin
```

You will be asked to enter a password for the `nagiosadmin` user. The user's credentials are stored in `/etc/nagios3/htpasswd.users`. To change the `nagiosadmin` password, or add additional users to the Nagios CGI scripts, use the `htpasswd` that is part of the `apache2-utils` package.

For example, to change the password for the `nagiosadmin` user enter:

```
sudo htpasswd /etc/nagios3/htpasswd.users nagiosadmin
```

To add a user:

```
sudo htpasswd /etc/nagios3/htpasswd.users steve
```

### On server02

Next, on `server02` install the `nagios-nrpe-server` package. From a terminal on `server02`, enter:

```
sudo apt install nagios-nrpe-server
```

> **Note**
>
> NRPE allows you to run local checks on remote hosts. There are other ways of accomplishing this, including through other Nagios plugins.

### Configuration overview

There are a few directories containing Nagios configuration and check files.

- `/etc/nagios3`: contains configuration files for the operation of the Nagios daemon, CGI files, hosts, etc.
- `/etc/nagios-plugins`: houses configuration files for the service checks.
- `/etc/nagios`: is located on the remote host and contains the `nagios-nrpe-server` configuration files.
- `/usr/lib/nagios/plugins/`: where the check binaries are stored. To see the options of a check use the `-h` option.

  For example: `/usr/lib/nagios/plugins/check_dhcp -h`

There are many checks Nagios can be configured to run for any particular host. In this example, Nagios will be configured to check disk space, *DNS*, and a MySQL *host group*. The DNS check will be on `server02`, and the MySQL host group will include both `server01` and `server02`.

> **Note**
>
> See these additional guides for details on setting up *Apache*, *Domain Name Service (DNS)*, and *MySQL*.

Additionally, there are some terms that once explained will hopefully make understanding Nagios configuration easier:

- **Host**: a server, workstation, network device, etc that is being monitored.
- **Host group**: a group of similar hosts. For example, you could group all web servers, file servers, etc.
- **Service**: the service being monitored on the host, such as HTTP, DNS, NFS, etc.
- **Service group**: allows you to group multiple services together. This is useful for grouping, e.g., multiple HTTP.
- **Contact**: the person to be notified when an event takes place. Nagios can be configured to send emails, SMS messages, etc.

By default Nagios is configured to check HTTP, disk space, SSH, current users, processes, and load on the **localhost**. Nagios will also ping-check the **gateway**.

Large Nagios installations can be quite complex to configure. It is usually best to start small (i.e. with one or two hosts), get things configured the way you like, and then expand.

### Configure Nagios

First, create a **host** configuration file for `server02`. Unless otherwise specified, run all these commands on `server01`. In a terminal enter:

```
sudo cp /etc/nagios3/conf.d/localhost_nagios2.cfg \
/etc/nagios3/conf.d/server02.cfg
```

> **Note**
>
> In the above and following command examples, replace "`server01`", "`server02`", `172.18.100.100`, and `172.18.100.101` with the host names and IP addresses of your servers.

Next, edit `/etc/nagios3/conf.d/server02.cfg`:

```
define host{
        use                     generic-host  ; Name of host template to use
        host_name               server02
        alias                   Server 02
        address                 172.18.100.101
}

# check DNS service.
define service {
        use                             generic-service
        host_name                       server02
        service_description             DNS
        check_command                   check_dns!172.18.100.101
}
```

Restart the Nagios daemon to enable the new configuration:

```
sudo systemctl restart nagio3.service
```

Now add a service definition for the MySQL check by adding the following to `/etc/nagios3/conf.d/services_nagios2.cfg`:

```
# check MySQL servers.
define service {
        hostgroup_name          mysql-servers
        service_description     MySQL
        check_command           check_mysql_cmdlinecred!nagios!secret!$HOSTADDRESS
        use                     generic-service
        notification_interval 0 ; set > 0 if you want to be renotified
}
```

A **mysql-servers** host group now needs to be defined. Edit `/etc/nagios3/conf.d/hostgroups_nagios2.cfg` adding:

```
# MySQL hostgroup.
define hostgroup {
        hostgroup_name  mysql-servers
                alias           MySQL servers
                members         localhost, server02
}

The Nagios check needs to authenticate to MySQL. To add a `nagios` user to MySQL,
enter:

```bash
mysql -u root -p -e "create user nagios identified by 'secret';"

```{note}
The `nagios` user will need to be added all hosts in the `mysql-servers` host
group.
```

Restart Nagios to start checking the MySQL servers.

```
sudo systemctl restart nagios3.service
```

Lastly, configure NRPE to check the disk space on `server02`. On `server01` add the service check to `/etc/nagios3/conf.d/server02.cfg`:

```
# NRPE disk check.
define service {
        use                     generic-service
        host_name               server02
        service_description     nrpe-disk
        check_command           check_nrpe_1arg!check_all_disks!172.18.100.101
}
```

Now on `server02` edit `/etc/nagios/nrpe.cfg`, changing:

```
allowed_hosts=172.18.100.100
```

And below in the command definition area add:

```
command[check_all_disks]=/usr/lib/nagios/plugins/check_disk -w 20% -c 10% -e
```

Finally, restart `nagios-nrpe-server`:

```
sudo systemctl restart nagios-nrpe-server.service
```

Also, on `server01` restart `nagios3`:

```
sudo systemctl restart nagios3.service
```

You should now be able to see the host and service checks in the Nagios CGI files. To access them, point a browser to `http://server01/nagios3`. You will then be prompted for the `nagiosadmin` username and password.

### Install Munin

Before installing Munin on `server01` Apache2 will need to be installed. The default configuration is fine for running a Munin server. For more information see *setting up Apache*.

### On server01

First, on `server01` install `munin` by running the following command in a terminal:

```
sudo apt install munin
```

### On server02

Now on `server02` install the `munin-node` package:

```
sudo apt install munin-node
```

### Configure Munin on server01

On `server01` edit the `/etc/munin/munin.conf` to add the IP address for `server02`:

```
## First our "normal" host.
[server02]
       address 172.18.100.101
```

**Note**

Replace `server02` and `172.18.100.101` with the actual hostname and IP address of your server.

### Configure munin-node on server02

To configure `munin-node` on `server02`, edit `/etc/munin/munin-node.conf` to allow access by `server01`:

```
allow ^172\.18\.100\.100$
```

**Note**

Replace `^172\.18\.100\.100$` with IP address for your Munin server.

Now restart `munin-node` on `server02` for the changes to take effect:

```
sudo systemctl restart munin-node.service
```

Finally, in a browser go to `http://server01/munin`, and you should see links to some graphs displaying information from the standard `munin-plugins` for disk, network, processes, and system.

> **Note**
>
> Since this is a new install it may take some time for the graphs to display anything useful.

### Additional plugins

The `munin-plugins-extra` package contains performance checks and additional services such as DNS, *DHCP*, Samba, etc. To install the package, from a terminal enter:

```
sudo apt install munin-plugins-extra
```

Be sure to install the package on both the server and node machines.

### Further reading

- See the Munin and Nagios websites for more details on these packages.
- The Munin Documentation page includes information on additional plugins, writing plugins, etc.
- The Nagios Online Documentation site.
- There is also a list of books related to Nagios and network monitoring.
- The Nagios Ubuntu Wiki page also has more details.

# 3. Ubuntu Server reference

Our reference section is used for quickly checking what software and commands are available, and how to interact with various tools.

- Our *Glossary* contains definitions for common terminology used in this documentation.

## 3.1. Glossary

We are currently compiling and defining terms for this glossary. If you would like to help, please visit our *contributions page* for details on how to get involved.

**Jump to:**

*A – B – C – D – E – F – G – H – I – J – K – L – M – N – O – P – Q – R – S – T – U – V – W – X – Y – Z*

### 3.1.1. A

**ABI**
**Application Binary Interface**
An ABI is an interface that defines how two modules interact with each other at the machine code level. Most often, these modules are applications using external libraries. An ABI defines a low-level and hardware-dependent interface compared to an *API*, which is considered high-level and hardware-independent.

**ACL**
**Access Control List**
An ACL is a list of access permissions that defines entities and their access rights to resources. ACLs can specify access with varying levels of granularity, ranging from full access to a resource, to permission for a specific operation.

See also:

- `the ACL manual page`

Related topic(s):

- Security, *OpenLDAP*, and *Kerberos*

**AD**
**Active Directory**
*Work in Progress*

**ADSys**
ADSys is an Ubuntu-specific Active Directory client developed by Canonical. ADSys complements System Security Services Daemon (SSSD) by adding native Group Policy Object support, privilege management, and custom scripts execution.

See also:

- the ADSys documentation

Related topic(s):

- *Group Policy Object* and *SSSD*

**AES**
**Advanced Encryption Standard**
An AES is a symmetric encryption algorithm designed to encrypt data securely into an

unreadable format that can only be decrypted with the same key used for encryption.

Related topic(s):

- Security

**Alertmanager**

Alertmanager is an open-source monitoring system developed by the Prometheus project to monitor and handle alerts. It offers several key features, including *Grouping* to combine alerts, *Inhibition* to suppress certain alerts when others are already firing, and *Silencing* to temporarily mute specific alerts.

See also:

- the Alertmanager documentation

Related topic(s):

- Observability and *Prometheus*

**ALUA**
**Asymmetric Logical Unit Access**

It is a storage concept used in Small Computer System Interface (SCSI) environments, particularly in Multi-Path Input/Output (MPIO) setups for shared storage systems like Storage Area Networks (SANs). ALUA informs a system about which paths to a storage device are optimal and which are non-optimal, enabling it to make smarter decisions about accessing shared disks.

Related topic(s):

- *MPIO SCSI*, *SAN*

**AMD**
**Advanced Micro Devices**

AMD can refer to:

- The (AMD) company: semiconductor company that designs computer components
- An AMD processor: a microprocessor designed and produced by the AMD company
- All Intel/AMD 64-bit processors: the term “amd64” is commonly used to refer to 64-bit processors due to the company’s role in developing this architecture.

Related topic(s):

- Networking

**Ansible**

Ansible is an open-source IT automation tool developed by Red Hat. It offers several automation features, enabling developers and organizations to automate provisioning, configuration management, and application deployment.

See also:

- The Ansible website

Related topic(s):

- Automation

**Apache2**

A robust, open-source HTTP server software designed for the deployment and delivery of web-based applications and content. Functioning as a request-response service, Apache 2 processes HTTP requests from client applications, facilitating the transmission of static and dynamic web resources. It has a modular architecture, supporting a wide array of extensions, enabling customizable functionality including security protocols (e.g., *SSL*/*TLS*), server-side scripting, and content management.

Widely deployed in server environments, Apache 2 is a foundational component of numerous web infrastructure stacks, underpinning a substantial portion of internet-accessible services.

See also:

- The Apache project documentation

Related topic(s):

- Web servers

**API**
**Application Programming Interface**

An API is a type of software interface that acts as a connection between different software programs, allowing them to communicate and exchange data. APIs exist on multiple layers of abstraction, from low-level APIs closest to system hardware to high-level web APIs that enable clients and remote servers to communicate.

**AppArmor**

AppArmor is a Linux security module that provides *Mandatory Access Control (MAC)* for programs. AppArmor restricts what applications can do, even when they are compromised. It enforces a set of security policies (called profiles) that define what files, capabilities, and system resources a given program is allowed to access.

See also:

- The AppArmor website

Related topic(s):

- Security

**Apport**

Apport is a debugging tool and crash reporting system used in Ubuntu and Debian-based Linux distributions. It can automatically detect crashes in programs and system services, collect detailed diagnostic data, generate crash reports, and prompt the user to send the report to developers via systems like Launchpad. It is typically disabled by default on production systems because it can expose sensitive information in logs, but is used during development or testing.

See also:

- The Apport Wiki Page

Related topic(s):

- Debugging

**APT**
**Advanced Package Tool**

APT is a package management system used by Debian and Debian-based Linux distribu-

tions like Ubuntu. APT helps install, update, upgrade, and remove software packages from the command line.

See also:

- *Install and manage packages*

**armhf**
**ARM hard-float**
armhf is a designation used in Linux distributions to describe a 32-bit variant of the ARM architecture that has hardware-based floating-point support. armhf is typically used for lightweight systems or backward compatibility, especially in embedded environments.

Related topic(s):

- arm32, arm64

**ARP**
**Address Resolution Protocol**
ARP is a network protocol used to map an IP address to a physical machine (*MAC address*) on a local area network (LAN).

Related topic(s):

- Networking

**ASCII**
**American Standard Code for Information Interchange**
A character encoding standard.

**async**
**asynchronous**
A term commonly used in programming to describe operations that take place without blocking the main execution thread. Instead of waiting for a particular operation to finish (such as reading a file or making a network request), "async" programs can keep running other operations in the meantime. These operations are often dispatched to the background, allowing them to run in parallel. If needed, however, the program can still wait for the result of an asynchronous operation.

Related topic(s):

- Concurrency, parallelism, and threading

**attestation**
In Confidential or Trusted Computing contexts, attestation refers both to the process of generating verifiable evidence about the trustworthiness of a system, and to the artifact produced by that process (commonly called an attestation report or quote).

- As a process: A TEE or TPM measures critical system components (firmware, boot chain, enclave/VM code) and cryptographically signs these measurements with keys rooted in hardware trust. The evidence is then checked by a verifier against reference values or policies.
- As a document: The signed report (attestation) is the concrete proof presented to a verifier. It contains the measurements and cryptographic signatures used to establish trust.

Attestation may be local (between components on the same host) or remote (to an external verifier). It underpins measured boot, secure provisioning, and Confidential

Computing workloads.

**Authenticator**

An authenticator is any system, method, or mechanism used to verify a user's identity during the authentication process. It can range from something as simple as a password field (e.g., LDAP *bind*) to more advanced tools like biometric scanners or one-time code generators. Authenticators are essential components of authentication protocols and can be used in both single-factor and multi-factor authentication setups.

Related topic(s):

- OpenLDAP, authentication

**autocommit**

autocommit is a database feature that automatically commits every individual SQL statement as soon as it is executed. When autocommit is enabled, every SQL statement is treated as its own transaction and is applied immediately and permanently.

This means it is impossible to undo or roll back a statement executed with autocommit enabled. While autocommit is a common default in many systems, behavior can vary depending on the database or language bindings. For example, in Python's `sqlite3` module, Python 3.12 introduces changes to transaction control, allowing explicit control over autocommit mode.

See also:

- autocommit behavior in Python's `sqlite3` module

Related topic(s):

- Databases

**autodetect**

autodetect is the ability of a system to automatically detect and configure hardware or settings without user input. In Ubuntu Server and other Linux systems, this is used during boot or installation to identify devices like disks, network interfaces, or keyboard layouts. The kernel, installers, and configuration tools rely on autodetection to simplify setup by loading the right drivers and defaults based on the system's hardware and environment.

Related topic(s):

- Kernel modules

**autoinstall**

Autoinstall is a feature in Ubuntu Desktop and Ubuntu Server that provides fully automated installations using a pre-defined configuration file. This file describes how the system should be installed, including disk partitioning, user accounts, package selection, and network settings.

See also:

- The Autoinstall documentation

Related topic(s):

- cloud-init

**autorid**

autorid is a Samba ID mapping backend that automatically assigns *UID* and *GID* values to

security identifiers (SIDs) when integrating with Active Directory (AD). It ensures consistent and persistent Unix ID mapping without requiring manual configuration for each domain or user/group.

See also:

- The autorid Samba Wiki

Related topic(s):

- Samba, Active Directory

**AWS**
**Amazon Web Services**
AWS is a cloud computing platform that offers a wide range of on-demand services such as compute, storage, networking, machine learning, analytics and much more. It allows individuals and companies to run applications without owning physical hardware, to scale resources up or down as needed, and to pay only for what they use.

See also:

- The AWS documentation

Related topic(s):

- Clouds

### 3.1.2. B

**backend**
*Work in Progress*

**Backports**
*Work in Progress*

**Backtrace**
*Work in Progress*

**BDC**
**Backup Domain Controller**
*Work in Progress*

**bind**
*Work in Progress*

**BindDN**
*Work in Progress*

**BIOS**
*Work in Progress*

**BMC**
**Baseboard Management Controller**
*Work in Progress*

**bootloader**
*Work in Progress*

**BOOTP**
**Bootstrap Protocol**
A network protocol that assigns IP addresses and boot information to a diskless system.

**bootstrap**
*Work in Progress*

**btrfs**
**B-tree File System**
*Work in Progress*

### 3.1.3. C

**CA**
**Certificate Authority**
*Work in Progress*

**CAC**
**Common Access Card**
*Work in Progress*

**CARP**
**Cache Array Routing Protocol**
*Work in Progress*

**CCID**
**Chip Card Interface Device**
*Work in Progress*

**CDB**
**Command Descriptor Block**
*Work in Progress*

**CGNAT**
**Carrier-Grade Network Address Translation**
*Work in Progress*

**CGI**
**Common Gateway Interface**
*Work in Progress*

**checksums**
*Work in Progress*

**chrony**
*Work in Progress*

**chroot**
*Work in Progress*

**CIDR**
**Classless Inter-Domain Routing**
*Work in Progress*

**CIFS**
**Common Internet File System**
*Work in Progress*

**CIS**
**Center for Internet Security**
*Work in Progress*

**CLVM**
**Clustered Logical Volume Manager**
*Work in Progress*

**CMS**
**Configuration Management System**
*Work in Progress*

**CN**
**Common Name**
*Work in Progress*

**colocation**
*Work in Progress*

**conffile**
*Work in Progress*

**config**
*Work in Progress*

**connectionless**
*Work in Progress*

**containerization**
*Work in Progress*

**CPU**
**Central Processing Unit**
*Work in Progress*

**CRL**
**Certificate Revocation List**
*Work in Progress*

**crypto**
**cryptographic**
*Work in Progress*

**CSR**
**Certificate Signing Request**
*Work in Progress*

**CVE**
**Common Vulnerabilities and Exposures**
*Work in Progress*

### 3.1.4. D

**DAC**
**Discretionary access control**
A form of access control where the owner of a resource can grant/revoke permissions to other users.

Related topic(s):

- Security

**daemonize**
The process of converting a program to run in the background as a service, independent

of user sessions.

**DARPA**
**Defense Advanced Research Projects Agency**
A research and development agency of the United States Department of Defense responsible for the development of emerging technologies for use in the military.

**DASD**
**Direct Access Storage Device**
This term was coined by IBM to refer to a type of storage that allows random access to storage (hard-drives, optical discs, etc). It contrasts with sequential access storage such as magnetic tape or punched cards.

Related topic(s):

- Storage

**Datagram**
In networking, a self contained, independent packet sent over a network. It can be routed from source to destination without relying on earlier or subsequent transfers.

Related topic(s):

- Networking

**dblink**
**Database link**
A connection between two databases (mainly Oracle and PostgreSQL), allowing one database to query data from the other.

Related topic(s):

- Databases

**DC**
**Domain Component**
*Work in progress*

**DDNS**
**Dynamic Domain Name System**
A service that automatically updates DNS records when the underlying IP address changes (aka, dynamic IP).

Related topic(s):

- Networking

**deb822**
A structured configuration format based on RFC822.

See also:

- RFC 822
- The `deb822(5)` manual page

**debconf**
A *configuration management system* handling the configuration of software packages during installation or upgrades by prompting users for necessary settings and storing them for subsequent installations or updates.

**deduplication**
: Process of removing duplicate copies of data in storage spaces. The redundant data is then replaced with a reference to the original.

**denylist**
: In cyber-security, a denylist is a list of entities (IP, domains, emails, etc), that are explicitly denied access to a system or service.

  Related topic(s):

  - Security

**DER**
**Distinguished Encoding Rules**
: A standardized encoding format for data (mostly cryptographic certificates and keys) for transmission and storage.

**DGC**
**Distributed Garbage Collection**
: A process used in distributed systems to manage memory across multiple interconnected computers allowing identification and reclaiming of unused memory across nodes.

**DHCP**
**Dynamic Host Configuration Protocol**
: A network protocol used to automatically assign network configuration details (IP, DNS, gateway, etc) to devices allowing for easy network management and connections within the network.

**DHCPD**
**Dynamic Host Configuration Protocol Daemon**
: Server software responsible for assigning the network configuration via DHCP.

**DIT**
**Directory Information Tree**
: In directory services (LDAP) this is a hierarchical tree-like structure used to organize and store information.

  Related topic(s):

  - OpenLDAP

**DKMS**
**Dynamic Kernel Module Support**
: A framework used in Linux systems to automatically rebuild and install kernel modules when the kernel is updated.

  Related topic(s):

  - Kernel

**DMA**
**Direct Memory Access**
: DMA is a technology that allows peripheral devices (hard drives, network cards, etc) to access the system's memory directly, bypassing the CPU and thus improving performance.

**DMAR**
**Direct Memory Access Remapping**

DMAR is a technology used to control and secure *Direct Memory Access* operations and ensure that devices can only access memory regions they are authorized to. This helps to prevent unauthorized access, memory corruption, or security vulnerabilities. It is often used in virtualized environments to isolate devices between *virtual machines (VMs)* and the host system.

**dmesg**
: A command in Linux systems that displays system logs related to hardware, drivers, and kernel events, such as system startup, device detection, and errors. It is commonly used for troubleshooting hardware issues and system diagnostics.

**DN**
**Distinguished Name**
: In directory services (LDAP), this is a unique identifier used to represent an entry in a directory, such as a user or a group. It's often composed of sub-components like *CN*, *OU*, or *DC*.

**DNS**
**Domain Name System**
: A system that translates human-readable domain names (e.g. `ubuntu.com`) to their IP addresses (`185.125.190.20`).

  Related topic(s):

  - Networking

**dnsmasq**
: A lightweight, open-source *DNS* and *DHCP* server software.

**DNSSEC**
**Domain Name System Security Extensions**
: DNSSEC is a set of security extensions to *DNS* which allow DNS data to be verified for authenticity and integrity.

  Related topic(s):

  - Security

**Docker**
: One of the most popular containerization platforms, which allows developers to package applications – together with their dependencies – into lightweight containers. This provides a consistently reproducible environment for deploying applications.

  Related topic(s):

  - Containers

**DocumentRoot**
: A directive in web server configuration files that specifies the directory on the server where web files are stored (root location).

**DoT**
**DNS over TLS**
: DNS over TLS utilizes *Transport Layer Security (TLS)* to encrypt the entire DNS connection, rather than just the payload. DoT servers listen on TCP port 853.

  See also:

  - RFC 7858

Related topic(s):

- Networking
- Security

**DoH**
**DNS over HTTPS**
: DNS over HTTPS is tunneling DNS query data over encrypted *HTTPS* connections. It uses TCP port 443, and thus looks similar to web traffic.

  See also:

  - RFC 8484

  Related topic(s):

  - Networking
  - Security

**dpkg**
: dpkg is a package manager for Debian-based systems. It can install, remove, and build packages, but unlike other package management systems, it cannot automatically download and install packages -– or their dependencies.

**DRBD**
**Distributed Replicated Block Device**
: A software-based storage solution for Linux that allows for the mirroring of block devices between multiple hosts. The replication is transparent to other applications on the host systems. Any block device hard disks, partitions, RAID devices, logical volumes, etc can be mirrored.

  Related topic(s):

  - Storage

**DTLS**
**Datagram Transport Layer Security**
: A protocol that provides security for datagram-based communication, such as *UDP*. It is designed to offer similar security features as *TLS* but adapted for the connectionless nature of datagram protocols.

### 3.1.5. E

**EAL**
**Environment Abstraction Layer**
: A software layer that provides a standardized interface between an operating system and the underlying hardware. It abstracts hardware-specific details, allowing software to run on different hardware platforms without modification.

**ECKD**
**Extended Count Key Data**
: A disk storage format used by IBM mainframe systems, which provides advanced features such as better error detection and correction, as well as enhanced management of data records.

**EFI**
**Extensible Firmware Interface**
: A type of firmware interface designed to initialize hardware and load the operating

system during the boot process of a computer. Replacement for the older *BIOS* and ancestor of the *UEFI*.

**ELinks**
A text-based web browser for Unix-like operating systems. It allows users to browse the web in a terminal, making it ideal for environments without a *GUI*.

**Engenio**
A company that developed and manufactured storage systems including *SAN* and *NAS*. Later acquired by LSI Corporation and then by Seagate Technology.

**EOL**
**End of life**
When a product, service, software is no longer supported or maintained.

**ERD**
**Enterprise Ready Drivers**
Drivers that are specifically designed and optimized for use in enterprise environments, where stability, performance, and reliability are critical.

**ESM**
**Expanded Security Maintenance**
A service provided by Canonical to extend security updates and patches for older *LTS* releases of the Ubuntu operating system after the LTS standard support period has ended.

**ESXi**
A bare-metal virtualization platform created by VMWare that enables multiple virtual machines to operate on a single physical server.

### 3.1.6. F

**failover**
In a *Storage Area Network (SAN)* environment, this occurs when data flows into an alternative I/O path because a cable, switch, or controller in the current path failed.

It is a common feature in high availability environments and is handled (usually automatically) by multipathing software.

**fallbacks**
This is a manual or automatic switch to an alternative method, when the primary option fails or is less preferred.

**FastCGI**
**Fast Common Gateway Interface**
FastCGI is an extension of the *CGI* protocol that starts a persistent FastCGI application process, allowing it to handle multiple requests instead of starting a new process for each request as a traditional CGI does.

**FC**
**Fiber Channel**
FC is a storage networking protocol used for low-latency communication between a storage device and a node in a *Storage Area Network (SAN)*.

**FHS**
**Filesystem Hierarchy Standard**
FHS is a standard that defines the directory structure and contents in Linux and Unix-like operating systems.

**Fileset**
: A fileset defines a group of directories that will be included when performing a backup job using Bacula.

  Related topic(s):

  - Storage

**filesystem**
: A filesystem defines how data is organized, stored, and accessed on a storage device.

  Related topic(s):

  - Storage

**FIPS**
**Federal Information Processing Standard**
: A set of publicly-announced US government standards for codes, data security and encryption.

  See also:

  - NIST

**Fluentd**
: Fluentd is a data collection platform that gathers events from a container for later analysis on other platforms.

  Related topic(s):

  - Observability, Containers

**FQDN**
**Fully Qualified Domain Name**
: A FQDN represents a complete name that specifies the exact location of a host within the *DNS* hierarchy.

  Related topic(s):

  - Networking

**FreeIPA**
**Free Identity, Policy, and Audit**
: FreeIPA is an open-source security solution for Linux/Unix-like systems that stores user identities in an *LDAP* directory, manages a *CA*, and enables authentication, policy enforcement, and auditing through integrations with *SSSD* and *Kerberos*.

  Related topic(s): Security, OpenLDAP

**Freenode**
: Freenode is an open-source *Internet Relay Chat (IRC)* platform used by many open-source communities for real-time discussions.

**frontend**
: A frontend is a user-friendly interface for managing a complex system.

  - In firewall management, a frontend like `ufw` simplifies configuring `iptables`.
  - In QEMU/KVM graphics, a frontend is the virtual graphic adapter presented to the guest *OS*, allowing it to process and store graphical output in memory. The guest

OS treats it like a *GPU*, while the host determines how to display the output using the *backend*.

- In LDAP, the frontend is a unique database that defines global default settings, such as who the admin user is, who can access database entries, or the limit on the number of search results. These settings apply to all LDAP databases inside *slapd*, unless overridden.

Related topic(s):

- Virtualization and containers, Security, OpenLDAP.

**fsck**
**File System Check**

`fsck` is a Linux/Unix-like system utility tool that checks for, and repairs, any *filesystem* errors.

Related topic(s):

- Storage

**FULLTEXT**

FULLTEXT is an index type that allows for fast indexing and searching large quantities of text. It takes a sentence, splits it into words, and links them to row IDs. When a search query for a word is made, MySQL quickly looks up the row the word appear in, and retrieves all matching row IDs, rather than scanning the entire table. It can also find similar words using natural language processing.

See also:

- Full-Text Search Functions

Related topic(s):

- Databases

**FW**
**Firmware**

Firmware is a software that runs before an operating system (OS) boots.

- When a QEMU microvm starts, the firmware initializes minimal virtual hardware like allocating *RAM* to the OS, and then loads the Linux kernel into memory.
- In a physical device, firmware configures *PCIe* devices like *GPUs* or network cards.

## 3.1.7. G

**gcplogs**

A logging driver that allows logs to be forwarded from a Docker container running in Google Cloud to the Google Cloud Logging service.

Related topic(s):

- Cloud, Containers, Observability

**gcrypt**

A cryptographic library that supports encryption, hashing, etc. for applications.

Related topic(s):

- Cryptographic libraries

**GDB**
**GNU Debugger**
GDB traces the current execution of a program, with the aim of identifying any issues.

Related topic(s):

- Debugging

**gelf**
**GELF**
**Graylog Extended Log Format**
GELF is a logging driver that allow logs to be forwarded in *JSON* format, but with extra unique fields. These logs are sent from a Docker container to a data collector platform like *Graylog*, *Logstash*, and *Fluentd*.

Related topic(s):

- Containers, Observability

**GFS2**
A shared-disk *filesystem* that allows multiple servers to access a single disk. It uses a locking system to ensure that no two servers modify the same data simultaneously, thus preventing data corruption if one server fails. Additionally, fencing is used to isolate failed nodes, ensuring that their locks can be safely recovered.

Related topic(s):

- High availability, Storage

**GB**
Gigabyte (unit of measurement) 1 GB = 1024 bytes

**GID**
**Group ID**
A GID is an identifier for a collection of users. It helps administrators enforce system or file access permissions on multiple users at once.

Related topic(s):

- Active Directory integration, Samba, Security, SSSD

**gitolite**
*Gitolite* is a tool installed on a central server for managing git repositories and controlling access to them, all via the command line. The central server becomes a git server.

Related topic(s):

- Backups and version control

**GKE**
**Google Kubernetes Engine**
GKE is a managed Kubernetes service provided by Google cloud.

**GL**
**Graphics Library**
A GL is an *API* for interacting with a graphics card, enabling it to perform better rendering.

Related topic(s):

- Graphics

**GNU**
**GNU's Not Unix**
A recursive acronym, GNU, is an operating system containing several free software packages. It can be used in combination with the Linux kernel.

**GnuTLS**
**GNU's Not Unix Transport Layer Security**
GnuTLS is a GNU software package that secures data-in-transit by implementing the *SSL*, *TLS* and *DTLS* protocol.

Related topic(s):

- Cryptography, Web services, OpenLDAP

**GPG**
**GNU Privacy Guard**
GPG is a GNU software package that secures data-at-rest before sending it to a recepient.

Related topic(s):

- Security, Cryptography

**GPS**
**Global Positioning System**
GPS is a collection of satellites that provides accurate time using radio signals from their atomic clocks. A GPS receiver plugged into a computer can sync with these satellites and generate *PPS* signal, which delivers ultra-accurate time that applications can use as a time source.

Related topic(s):

- Networking

**GPSD**
**GPS daemon**
This reads data from a GPS receiver and makes it available as a shared resource to multiple applications (e.g., *Chrony*) to use for precise time synchronization.

Related topic(s):

- Networking

**GPU**
**Graphics Processing Unit**
A GPU enhances graphics rendering for a computer and any virtual machines running inside of it.

Related topic(s):

- Graphics, Virtualisation and containers

**Graylog**
A data collector platform for storing, analysing, and interpreting logs. These logs are received from a *gelf* logging driver in Docker.

Related topic(s):

- Containers

**GPO**
**Group Policy Object**
A set of configuration rules used to manage and enforce security and system behaviours across users or computers within an Active Directory (AD) object.

Related topic(s): Active Directory integration

**GRUB**
**GRand Unified Bootloader**
*Work in progress*

**GSSAPI**
**Generic Security Services Application Program Interface**
GSSAPI is a vendor-agnostic *API* that uses an existing communication protocol to establish a secure communication between applications. It does this securely by verifying user credentials, ensuring that data being transmitted remains unchanged, preventing unauthorized access, and securely negotiating encryption keys.

Related topic(s):

- Cryptography

**GTK**
**GIMP Toolkit**
GTK is a library used to create *graphical user interfaces (GUIs)*. It provides a visual interface for interacting with the Bacula Director when managing backup-related operations.

Related topic(s):

- Graphics, Backups and version control

**GUI**
**Graphical User Interface**
A GUI is a visual representation of operations within a computer. It is usually represented as icons rather than text only.

**GZIP**
**GNU Zip**
GZIP is a *GNU* software package used to reduce the file size of a backup.

- When applied directly to files, it replaces the original file type with a `.gz` type.
- When used in Bacula's *fileset*, it reduces the storage size of backed-up directories within Bacula's storage volumes.
- When used to reduce the size of a folder, it works in combination with a `tar` tool which first combines multiple files into a single archive, before applying GZIP's size-reduction technique.

Related topic(s):

- Backups and version control

### 3.1.8. H

**HA**
**High Availability**
HA is the process of ensuring that a system is always up. To achieve this, a redundant

system is set up that either takes over when the main system is down or runs alongside the main system to load-balance the workload.

Related topic(s):

- High availability

**HBA**
**Host Bus Adapter**
HBAs are interface cards that connect a server to a storage device.

Related topic(S):

- Device mapper multipathing

**HMAC**
**Hash-based Message Authentication Code**
A HMAC is a type of *Message Authentication Code*. While a general MAC may use various techniques during combination, HMAC follows a structured way. When a message and its HMAC are sent, the receiver verifies the integrity by computing the HMAC again – if the message is altered, the value will differ.

Related topic(S):

- Cryptography, Security

**HMC**
**Hardware Management Console**
A HMC is used to manage IBM servers. It can handle tasks like configuring network settings, loading Ubuntu installation files and installing the *OS*.

**hostgroup**
A group of backend web or database servers with similar configurations.

Related topic(S):

- Observability

**hostname**
A hostname identifies a server using a word rather than an *IP address*. This makes it easier to remember.

**HOTP**
**HMAC-based One-Time Password**
HOTP generates a one-time password by using the *HMAC* algorithm in combination with a counter. When a client presents the *OTP*, the server compares it with OTPs generated within a specific counter window to find a match.

**hotplug**
The process of adding or removing a device (USB, disks, etc.) while a virtual machine is running.

**HPB**
**Host Physical Bits**
HPB are appended to the name of an Ubuntu machine type. It signifies that a virtual machine will use the same number of bits the host *CPU* uses to point to physical memory.

**HPC**
**High Performance Computing**
HPC is the use of multiple servers to improve the performance of a task.

**HSG**
**High-availability Storage Group**
*Work in Progress*

**HSV**
**Highly-available Storage Virtualization**
*Work in Progress*

**HTCP**
**Hyper Text Caching Protocol**
*Work in Progress*

**HTML**
**HyperText Markup Language**
*Work in Progress*

**HTTP**
**HyperText Transfer Protocol**
*Work in Progress*

**HTTPD**
**HyperText Transfer Protocol Daemon**
*Work in Progress*

**HTTPS**
**HyperText Transfer Protocol Secure**
*Work in Progress*

**hugepage**
A huge page increases the page size on a host, and as a result, when virtual memory is allocated to an application, there are fewer page table entries required to map the virtual memory to physical memory. The page table entries are stored in Random Access Memory (RAM) and cached in the *Translation Lookaside Buffer (TLB)*.

**HWE**
**Hardware Enablement**
*Work in Progress*

### 3.1.9. I

**ICMP**
**Internet Control Message Protocol**
*Work in Progress*

**ICP**
**Internet Cache Protocol**
*Work in Progress*

**IDENT**
**Identification Protocol**
*Work in Progress*

**IMAP**
**Internet Messages Access Protocol**
*Work in Progress*

**init**
**initialization**
*Work in Progress*

**I/O**
**Input/Output**
*Work in Progress*

**IOMMU**
**Input-Output Memory Management Unit**
*Work in Progress*

**IoT**
**Internet of Things**
*Work in Progress*

**IP**
**Internet Protocol**
*Work in Progress*

**IP address**
*Work in Progress*

**IPC**
**Inter-Process Communication**
*Work in Progress*

**IPL**
**Initial Program Load**
*Work in Progress*

**IPMI**
**Intelligent Platform Management Interface**
*Work in Progress*

**IPP**
**Internet Printing Protocol**
*Work in Progress*

**IPSec**
**Internet Protocol Security**
*Work in Progress*

**iptables**
*Work in Progress*

**IPVS**
**IP Virtual Server**
*Work in Progress*

**IQN**
**iSCSI Qualified Name**
*Work in Progress*

**IRC**
**Internet Relay Chat**
*Work in Progress*

**ISC**
**Internet Systems Consortium**
*Work in Progress*

**iSCSI**
**Internet Small Computer System Interface**
*Work in Progress*

**ISO**
**International Organization for Standardization**
*Work in Progress*

**ISP**
**Internet Service Provider**
*Work in Progress*

### 3.1.10. J

**jitter**
Jitter is the variation in delay or latency between when data packets are sent and when they are received over a network, causing irregular arrival times at the destination. This variation is often caused by network congestion, packet loss, poor hardware performance or differences in the path packets take.

Related topic(s):

- Networking

**journald**
`journald`, also known as `systemd-journald`, is a logging service developed by the *systemd* project as part of the `systemd` suite. It collects and stores log messages from various sources, including `systemd` services, kernel messages, system logs, and application logs. `journald` stores logs in a binary format offering advantages, such as storage efficiency, searchability, and most especially structured logging. In containerized systems like Docker, it functions as a logging driver for containers.

See also:

- The `journald.conf(5)` manual page
- Docker `journald` documentation for details on using journald as a logging driver

Related topic(s):

- Logging, Observability

**JSON**
**JavaScript Object Notation**
This is a language-independent text format that uses conventions familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. Due to its simplicity, it is an ideal lightweight data interchange language.

See also:

- The JSON website

### 3.1.11. K

**KDC**
**Key Distribution Center**
*Work in Progress*

**keepalive**
*Work in Progress*

**Kerberos**
*Work in Progress*

**keypair**
*Work in Progress*

**keyring**
*Work in Progress*

**keysalt**
*Work in Progress*

**keyservers**
*Work in Progress*

**keytab**
*Work in Progress*

**Keytool**
*Work in Progress*

**KVM**
**Kernel-based Virtual Machine**
*Work in Progress*

### 3.1.12. L

**LAN**
**Local Area Network**
*Work in Progress*

**LDAP**
**Lightweight Directory Access Protocol**
*Work in Progress*

**LDIF**
**LDAP Data Interchange Format**
*Work in Progress*

**lightervisor**
*Work in Progress*

**LinuxONE**
IBM Linux-only enterprise server platform.

**Load-balancing**
*Work in Progress*

**localhost**
*Work in Progress*

**Logfiles**
*Work in Progress*

**Logstash**
*Work in Progress*

**Logwatch**
*Work in Progress*

**LPAR**
**Logical Partition**
*Work in Progress*

**LSI**
**Logic Systems Incorporated**
*Work in Progress*

**LTS**
**Long-Term Support**
*Work in Progress*

**LU**
**Logical Unit**
*Work in Progress*

**LUA**
**Lua Scripting Language**
*Work in Progress*

**LUN**
**Logical Unit Number**
*Work in Progress*

**LV**
**Logical Volume**
*Work in Progress*

**LVM**
**Logical Volume Manager**
A storage management framework in Linux that provides a layer of abstraction over physical storage devices. It allows disks or partitions to be pooled into a single storage space, enabling the creation of flexible logical volumes that can be easily resized or moved.

**LXC**
**Linux Containers**
*Work in Progress*

**LXD**
**Linux Container Daemon**
*Work in Progress*

### 3.1.13. M

**MAAS**
**Metal as a Service**
*Work in Progress*

**MAC**
**Message Authentication Code**
: A MAC verifies that a message hasn't been modified during transmission by combining a shared secret key between the sender and receiver, and a hash function.

**MAC address**
: *Work in Progress*

**manpage**
**manual page**
: *Work in Progress*

**MCE**
**Machine Check Exception**
: *Work in Progress*

**MD5**
**Message Digest Algorithm 5**
: A cryptographic hash function producing a 128-bit checksum.

**MDA**
**Mail Delivery Agent**
: *Work in Progress*

**mdev**
**Minimal Device Manager**
: *Work in Progress*

**metapackage**
: *Work in Progress*

**METAR**
**Meteorological Aerodrome Report**
: *Work in Progress*

**microservices**
: *Work in Progress*

**microVMs**
: *Work in Progress*

**MOTD**
**Message of the Day**
: *Work in Progress*

**mountpoints**
: *Work in Progress*

**MPIO**
**Multipath Input/Output**
: *Work in Progress*

**MSA**
**Modular Smart Array**
: *Work in Progress*

**MTA**
**Mail Transfer Agent**
: *Work in Progress*

**MTU**
**Maximum Transmission Unit**
*Work in Progress*

**MUA**
**Mail User Agent**
*Work in Progress*

**Multipass**
*Work in Progress*

**Multipath**
*Work in Progress*

**Multiview**
*Work in Progress*

**MySQL**
*Work in Progress*

### 3.1.14. N

**nameserver**
*Work in Progress*

**namespace**
*Work in Progress*

**NAS**
**Network Attached Storage**
*Work in Progress*

**NAT**
**Network Address Translation**
*Work in Progress*

**Netboot**
*Work in Progress*

**Netfilter**
*Work in Progress*

**Netplan**
*Work in Progress*

**NFS**
**Network File System**
*Work in Progress*

**NFV**
**Network Functions Virtualization**
*Work in Progress*

**nginx**
*Work in Progress*

**NIC**
**Network Interface Card**
*Work in Progress*

**NIS**
**Network Information Service**
*Work in Progress*

**NMI**
**Non-Maskable Interrupt**
*Work in Progress*

**NRPE**
**Nagios Remote Plugin Executor**
*Work in Progress*

**NSCQ**
**Network System Configuration Queue**
*Work in Progress*

**NSS**
**Name Service Switch**
*Work in Progress*

**NTP**
**Network Time Protocol**
*Work in Progress*

**NTS**
**Network Time Security**
*Work in Progress*

**NUMA**
**Non-Uniform Memory Access**
*Work in Progress*

**Nvidia**
*Work in Progress*

**NVMe**
**Non-Volatile Memory Express**
*Work in Progress*

**NVRAM**
**Non-Volative Random Access Memory**
*Work in Progress*

**NVSwitch**
**NVIDIA NVLink Switch**
*Work in Progress*

### 3.1.15. O

**OCF**
**Open Cluster Framework**
*Work in Progress*

**OCFS**
**Oracle Cluster File System**
*Work in Progress*

**OCI**
**Open Container Initiative**

*Work in Progress*

**OCSP**
**Online Certificate Status Protocol**
*Work in Progress*

**OpenLDAP**
*Work in Progress*

**OpenSC**
**Open Smart Card**
*Work in Progress*

**OpenSSH**
*Work in Progress*

**OpenSSL**
*Work in Progress*

**OpenStack**
*Work in Progress*

**OpenSUSE**
*Work in Progress*

**OpenVPN**
*Work in Progress*

**OpenVZ**
*Work in Progress*

**OpenWRT**
*Work in Progress*

**OS**
**Operating System**
*Work in Progress*

**OSA**
**Open Systems Adapter**
*Work in Progress*

**OSI**
**Open Systems Interconnection**
*Work in Progress*

**OTE**
**Operational Test and Evaluation**
*Work in Progress*

**OTP**
**One-Time Password**
*Work in Progress*

**OU**
**Organisational Unit**
*Work in Progress*

**OverlayFS**
*Work in Progress*

**OVS**
**Open vSwitch**
*Work in Progress*

### 3.1.16. P

**PAM**
**Pluggable Authentication Module**
*Work in Progress*

**passthrough**
*Work in Progress*

**PB**
Petabyte (unit of measurement) 1 PB = 1024 *TB*

**PCI**
**Peripheral Component Interconnect**
*Work in Progress*

**PCIe**
**Peripheral Component Interconnect Express**
*Work in Progress*

**PCS**
**Pacemaker/Corosync Stack**
*Work in Progress*

**PDC**
**Primary Domain Controller**
*Work in Progress*

**PEM**
**Privacy Enhanced Mail**
*Work in Progress*

**Petitboot**
*Work in Progress*

**PgSQL**
*Work in Progress*

**PHP**
**PHP: HyperText Preprocessor**
*Work in Progress*

**PID**
**Process Identifier**
*Work in Progress*

**pingable**
*Work in Progress*

**PIV**
**Personal Identity Verification**
*Work in Progress*

**PKCS**
**Public-Key Cryptography Standards**
*Work in Progress*

**PKI**
**Public Key Infrastructure**
*Work in Progress*

**pluggable**
*Work in Progress*

**PMD**
**Poll Mode Driver**
*Work in Progress*

**pockets**
Sub-repositories within the Ubuntu Package Archive.

See also:

- Ubuntu Project: pockets

**POSIX**
**Portable Operating System Interface (for UNIX)**
*Work in Progress*

**Postcopy**
*Work in Progress*

**Postfix**
*Work in Progress*

**Postgres**
**PostgreSQL**
*Work in Progress*

**PostScript**
*Work in Progress*

**Power 8**
**Power 9**
IMB POWER processor architectures used in enterprise servers.

**PowerShell**
*Work in Progress*

**PPA**
**Personal Package Archive**
*Work in Progress*

**ppc**
**PowerPC**
*Work in Progress*

**PPD**
**PostScript Printer Description**
*Work in Progress*

**PPS**
**Pulse Per Second**
*Work in Progress*

**Preboot**
*Work in Progress*

**preseed**
*Work in Progress*

**Prometheus**
*Work in Progress*

**proxy**
*Work in Progress*

**PTP**
**Precision Time Protocol**
*Work in Progress*

**PTR**
**Pointer Record**
*Work in Progress*

**PXE**
**Preboot Execution Environment**
*Work in Progress*

**PXELINUX**
**PXE Linux Loader**
*Work in Progress*

### 3.1.17. Q

**QA**
**Quality Assurance**
*Work in Progress*

**qdevice**
**Quorum Device**
*Work in Progress*

**QEMU**
**Quick Emulator**
*Work in Progress*

**qeth**
**QDIO Ethernet**
*Work in Progress*

**quickstart**
*Work in Progress*

### 3.1.18. R

**RAM**
**Random Access Memory**
*Work in Progress*

**rangesize**
*Work in Progress*

**rclone**
*Work in Progress*

**RDAC**
**Redundant Disk Array Controller**
*Work in Progress*

**RDBMS**
**Relational Database Management System**
*Work in Progress*

**RDN**
**Relative Distinguished Name**
*Work in Progress*

**Redbook**
*Work in Progress*

**renderer**
*Work in Progress*

**REXX**
**Restructured Extended Executor**
*Work in Progress*

**RFC**
**Request For Comments**
*Work in Progress*

**rid**
**Relative Identifier**
*Work in Progress*

**RISC-V**
**Reduced Instruction Set Computing - Version Five**
*Work in Progress*

**Rocks**
*Work in Progress*

**ROM**
**Read-Only Memory**
*Work in Progress*

**rootDN**
**Root Distinguished Name**
*Work in Progress*

**rootfs**
**Root File System**
*Work in Progress*

**routable**
*Work in Progress*

**RSA**
**Rivest–Shamir–Adleman**
RSA is an asymmetric encryption algorithm *Work in Progress*

**rsnapshot**
*Work in Progress*

**rsync**
*Work in Progress*

**rsyslog**
*Work in Progress*

**RTC**
**Real-Time Clock**
*Work in Progress*

**runtime**
*Work in Progress*

### 3.1.19. S

**Samba**
Open source software that implements the SMB/CIFS protocol for file and printer sharing with Windows systems.

See also:

- *Introduction to Samba*

**SAN**
**Storage Area Network**
*Work in Progress*

**sandboxed**
*Work in Progress*

**SANLOCK**
**SAN Locking Daemon**
*Work in Progress*

**SASL**
**Simple Authentication and Security Layer**
*Work in Progress*

**SBD**
**Storage-Based Death**
*Work in Progress*

**sbin**
**System Binaries**
*Work in Progress*

**schemas**
*Work in Progress*

**SCP**
**Secure Copy Protocol**
*Work in Progress*

**Scrollback**
*Work in Progress*

**SCSI**
**Small Computer System Interface**
*Work in Progress*

**SDN**
**Software-Defined Networking**
*Work in Progress*

**sdx**
**SCSI Desk (x)**
*Work in Progress*

**seccomp**
**Secure computing mode**
*Work in Progress*

**SFTP**
**SSH File Transfer Protocol**
*Work in Progress*

**SGI**
**Silicon Graphics Inc.**
*Work in Progress*

**SHA**
**Secure Hash Algorithm**
*Work in Progress*

**sharding**
*Work in Progress*

**Shell**
*Work in Progress*

**SHM**
**Shared Memory**
*Work in Progress*

**Shorewall**
**Shoreline Firewall**
*Work in Progress*

**SIDs**
**Security Identifiers**
*Work in Progress*

**SIMD**
**Single Instruction, Multiple Data**
*Work in Progress*

**slapd**
**Standalone LDAP Daemon**
*Work in Progress*

**SLAT**
**Second Level Address Translation**
*Work in progress*

**smartcard**
*Work in Progress*

**SMB**
**Server Message Block**
*Work in Progress*

**SMS**
**Short Message Service**
*Work in Progress*

**SMTP**
**Simple Mail Transfer Protocol**
*Work in Progress*

**SMTPS**
**SMTP Secure**
*Work in Progress*

**Snap**
*Work in Progress*

**snapd**
*Work in Progress*

**snapshot**
*Work in Progress*

**Snapstore**
*Work in Progress*

**SNMP**
**Simple Network Management Protocol**
*Work in Progress*

**SOA**
**Start of Authority**
*Work in Progress*

**Solaris**
*Work in Progress*

**SPC**
**SCSI Primary Commands**
*Work in Progress*

**Splunk**
*Work in Progress*

**SRU**
**Stable Release Update**
*Work in Progress*

**SSD**
**Solid State Drive**
*Work in Progress*

**SSH**
**Secure Shell**
*Work in Progress*

**SSH-key**
**Secure Shell Key**
A cryptographic encryption key pair for *SSH*, usually used and created with *OpenSSH*. It's split in two parts: a *public* and a *private* key files. The private key is secret and belogs to the owning user, and it's used to prove posession of that secret. The public key is not

secret and is used to securely identify only the private key holder. That way, entering a public key on a server for *SSH* access, only the private key holder can log in.

**SSI**
**Server-Side Includes**
*Work in Progress*

**SSL**
**Secure Sockets Layer**
*Work in Progress*

**SSO**
**Single Sign-On**
*Work in Progress*

**SSSD**
**System Security Services Daemon**
*Work in Progress*

**stateful**
*Work in Progress*

**STDIN**
**Standard Input**
*Work in Progress*

**STDOUT**
**Standard Output**
*Work in Progress*

**STDERR**
**Standard Error**
*Work in Progress*

**STK**
**Storake Tek**
*Work in Progress*

**storage**
*Work in Progress*

**subcommand**
*Work in Progress*

**Subiquity**
*Work in Progress*

**subnet**
**subnetwork**
*Work in Progress*

**substring**
*Work in Progress*

**subvolume**
*Work in Progress*

**sudo**
**superuser do**
*Work in Progress*

**superblock**
*Work in Progress*

**symlink**
*Work in Progress*

**syslog**
*Work in Progress*

**systemctl**
*Work in Progress*

**Systemd**
*Work in Progress*

### 3.1.20. T

**tasksel**
**Task selector**
*Work in Progress*

**TB**
Terabyte (unit of measurement) 1 TB = 1024 *GB*

**TCP**
**Transmission Control Protocol**
*Work in Progress*

**TEE**
**Trusted Execution Environment**
A secure processor enclave that isolates code and data to ensure confidentiality, integrity, and verifiable trust, forming the foundation of Confidential Computing.

**TFTP**
**Trivial File Transfer Protocol**
*Work in Progress*

**TGS**
**Ticket Granting Service**
*Work in Progress*

**TGT**
**Ticket Granting Ticket**
*Work in Progress*

**timedatectl**
*Work in Progress*

**timesyncd**
*Work in Progress*

**TLB**
**Translation Lookaside Buffer**
TLB is a *CPU* cache that stores recent page table entries. When the CPU translates a virtual address, it first checks the TLB. If the mapping is found, the translation is fast. If

it's missing, the CPU retrieves the mapping from the page table in memory, which takes longer.

**TLS**
**Transport Layer Security**
*Work in Progress*

**tmpfs**
**Temporary Filesystem**
*Work in Progress*

**tmux**
**Terminal Multiplexer**
*Work in Progress*

**topologies**
*Work in Progress*

**TOTP**
**Time-based One-Time Password**
*Work in Progress*

**TPM**
**Trusted Platform Module**
A hardware security chip that securely stores cryptographic keys, measures system integrity during boot, and enables features such as secure storage, encryption, and attestation.

**traceback**
*Work in Progress*

**Traceroute**
*Work in Progress*

**ttys**
**Teletype Terminals**
*Work in Progress*

**TXT**
**Trusted Execution Technology**
*Work in Progress*

### 3.1.21. U

**UDA**
**Unified Data Architecture**
*Work in Progress*

**UDP**
**User Datagram Protocol**
*Work in Progress*

**UEFI**
**Unified Extensible Firmware Interface**
*Work in Progress*

**ufw**
**Uncomplicated Firewall**
*Work in Progress*

**UID**
**User Identifier**
*Work in Progress*

**UI**
**User Interface**
*Work in Progress*

**unicast**
One-to-one communication

**unmount**
*Work in Progress*

**untrusted**
*Work in Progress*

**uptime**
*Work in Progress*

**URI**
**Uniform Resource Identifier**
*Work in Progress*

**URL**
**Uniform Resource Locator**
The web address of a resource on the internet.

**userspace**
*Work in Progress*

**USN**
**Update Sequence Number**
*Work in Progress*

**usr**
Refers to the `/usr/` directory and stands for “Unix System Resources”

**UUIDs**
**Universally Unique Identifiers**
*Work in Progress*

### 3.1.22. V

**vCPU**
**Virtual CPU**
*Work in Progress*

**VCS**
**Version Control System**
*Work in Progress*

**veth**
**Virtual Ethernet**
*Work in Progress*

**VFIO**
**Virtual Function I/O**
*Work in Progress*

**VFS**
**Virtual File System**
*Work in Progress*

**VFs**
**Virtual Functions**
*Work in Progress*

**VG**
**Volume Group**
*Work in Progress*

**vGPU**
**Virtual GPU**
*Work in Progress*

**virsh**
**Virtual Shell**
*Work in Progress*

**VirtIO**
**Virtual I/O**
*Work in Progress*

**virtual**
*Work in Progress*

**virtualization**
*Work in Progress*

**VLAN**
**Virtual Local Area Network**
*Work in Progress*

**VM**
**Virtual Machine**
*Work in Progress*

**VNC**
**Virtual Network Computing**
*Work in Progress*

**VPN**
**Virtual Private Network**
*Work in Progress*

**VRRP**
**Virtual Router Redundancy Protocol**
*Work in Progress*

**vsftpd**
**Very Secure FTP Daemon**
*Work in Progress*

### 3.1.23. W

**WAL**
**Write-Ahead Logging**
*Work in Progress*

**WAN**
**Wide Area Network**
*Work in Progress*

**WCCP**
**Web Cache Communication Protocol**
*Work in Progress*

**Webserver**
*Work in Progress*

**winbind**
**Windows Bind**
*Work in Progress*

**WireGuard**
*Work in Progress*

**WLAN**
**Wireless Local Area Network**
*Work in Progress*

**WSGI**
**Web Server Gateway Interface**
*Work in Progress*

**WWID**
**World Wide Identifier**
*Work in Progress*

### 3.1.24. X

**X.509**
Standard format for public key certificates used in TLS/SSL for secure communications.

**xhtml**
**Extensible HyperText Markup Language**
*Work in Progress*

**XML**
**Extensible Markup Language**
*Work in Progress*

### 3.1.25. Y

**YAML**
**YAML Ain't Markup Language**
*Work in Progress*

**Yubikey**
*Work in Progress*

### 3.1.26. Z

**zFCP**
**zSeries Fibre Channel Protocol**
*Work in progress*

**ZFS**
**Zettabyte File System**
*Work in progress*

**zpool**
**ZFS Pool**
*Work in progress*

**z/VM**
*Work in progress*

## 3.2. Server installation

Although Ubuntu Server is flexible and designed to run on a wide range of hardware, you can refer to the following requirements page to see the various architectures Ubuntu Server can be run on, and suggested minimal values for memory and storage.

### 3.2.1. System requirements

Ubuntu Server provides a flexible base for your solution that can run on a wide range of hardware, from small virtual machines to enterprise-scale computing.

Hard requirements depend on the scenario, but they're generally constrained by the following recommended values.

#### Architecture

Ubuntu Server supports various 64-bit architectures and 32-bit arm.

- amd64 (64-bit Intel/*AMD*)
- arm64 (64-bit Arm)
- armhf (32-bit Arm)
- ppc64el (64-bit Power)
- riscv64 (64-bit RISC-V)
- s390x (64-bit Mainframe)

For specific platforms, see our list of Ubuntu certified servers.

The numbers below are true for Ubuntu 24.04 Noble amd64. Other releases and architectures might differ slightly.

#### Memory

**Minimum RAM**: 1.5 GB (ISO installs) **Minimum RAM**: 1 GB (cloud images)

It's likely that your system might need more memory than that if you, for instance, have more hardware to initialise, have more complex setup plans, or are using other architectures. To cover better for any of those scenarios:

**Suggested minimum RAM**: 3 GB or more

Upper limits depend on the system hardware and setup.

#### Storage

**Minimum storage**: 5 GB (ISO installs) **Minimum storage**: 4 GB (cloud images)

In theory you could go even lower, like 2.5 GB for cloud image installs, but in practise it's quite likely that your system will need more disk storage than that to be really useful. Your setup plans could be more complex or you need more software to be installed, that could lead to increased storage needs. To cover better for any of those scenarios:

**Suggested minimum storage**: 25 GB or more

## 3.3. Data and storage

This section provides details on storing and backing up your data.

- *Backups and version control*

### 3.3.1. Data and storage

This section provides details on various topics related to storing, managing and accessing data.

#### Backups and version control

Example shell scripts for backing up your system.

- *Archive rotation shell script*

#### Archive rotation shell script

The *simple backup shell script* only allows for seven different archives. For a server whose data doesn't change often, this may be enough. If the server has a large amount of data, a more complex rotation scheme should be used.

#### Rotating NFS archives

Here, the shell script is slightly modified to implement a grandparent-parent-child rotation scheme (monthly-weekly-daily):

- The rotation will do a *daily* backup from Sunday to Friday.
- On Saturday, a *weekly* backup is done – giving four weekly backups per month.
- The *monthly* backup is done on the first day of the month, rotating two monthly backups based on whether the month is odd or even.

Here is the new script:

```
#!/bin/bash
####################################
#
# Backup to NFS mount script with
# grandparent-parent-child rotation.
#
####################################

# What to backup.
backup_files="/home /var/spool/mail /etc /root /boot /opt"
```

(continues on next page)

(continued from previous page)

```
# Where to backup to.
dest="/mnt/backup"

# Setup variables for the archive filename.
day=$(date +%A)
hostname=$(hostname -s)

# Find which week of the month 1-4 it is.
day_num=$(date +%-d)
if (( $day_num <= 7 )); then
        week_file="$hostname-week1.tgz"
elif (( $day_num > 7 && $day_num <= 14 )); then
        week_file="$hostname-week2.tgz"
elif (( $day_num > 14 && $day_num <= 21 )); then
        week_file="$hostname-week3.tgz"
elif (( $day_num > 21 && $day_num < 32 )); then
        week_file="$hostname-week4.tgz"
fi

# Find if the Month is odd or even.
month_num=$(date +%m)
month=$(expr $month_num % 2)
if [ $month -eq 0 ]; then
        month_file="$hostname-month2.tgz"
else
        month_file="$hostname-month1.tgz"
fi

# Create archive filename.
if [ $day_num == 1 ]; then
        archive_file=$month_file
elif [ $day != "Saturday" ]; then
        archive_file="$hostname-$day.tgz"
else
        archive_file=$week_file
fi

# Print start status message.
echo "Backing up $backup_files to $dest/$archive_file"
date
echo

# Backup the files using tar.
tar czf $dest/$archive_file $backup_files

# Print end status message.
echo
echo "Backup finished"
```

(continues on next page)

(continued from previous page)

```
date

# Long listing of files in $dest to check file sizes.
ls -lh $dest/
```

The script can be executed using the same methods as in this `basic backup shell script <https://discourse.ubuntu.com/t/basic-backup-shell-script/36419>_`.

As discussed in the introduction, a copy of the backup archives and/or media can then be transferred off-site.

### Backing up to tape drives

A tape drive attached to the server can be used instead of an NFS share. Using a tape drive simplifies archive rotation, and makes taking the media off-site easier as well.

When using a tape drive, the filename portions of the script aren't needed because the data is sent directly to the tape device. Some commands to manipulate the tape *are* needed, however. This is accomplished using `mt`, a magnetic tape control utility – part of the `cpio` package.

Here is the shell script modified to use a tape drive:

```
#!/bin/bash
###################################
#
# Backup to tape drive script.
#
###################################

# What to backup.
backup_files="/home /var/spool/mail /etc /root /boot /opt"

# Where to backup to.
dest="/dev/st0"

# Print start status message.
echo "Backing up $backup_files to $dest"
date
echo

# Make sure the tape is rewound.
mt -f $dest rewind

# Backup the files using tar.
tar czf $dest $backup_files

# Rewind and eject the tape.
mt -f $dest rewoffl

# Print end status message.
echo
```

(continues on next page)

(continued from previous page)

```
echo "Backup finished"
date
```

**Note**

The default device name for a SCSI tape drive is `/dev/st0`. Use the appropriate device path for your system.

Restoring from a tape drive is basically the same as restoring from a file. Simply rewind the tape and use the device path instead of a file path. For example, to restore the `/etc/hosts` file to `/tmp/etc/hosts`:

```
mt -f /dev/st0 rewind
tar -xzf /dev/st0 -C /tmp etc/hosts
```

#### See also

- How-to: *Backups and version control*
- Explanation: *Introduction to backups*

# 3.4. High Availability

The recommended tool for managing High Availability Pacemaker clusters is now *pcs*, as of 23.04 (Lunar). Here we provide a reference guide to help you migrate from *crmsh* to *pcs*.

## 3.4.1. High availability

In Ubuntu 23.04 (Lunar) and onwards, `pcs` became the recommended and supported tool for managing Pacemaker clusters. The 23.04 release is the last release where `crmsh` is available.

To migrate from `crmsh` to `pcs`, refer to our reference table of *corresponding commands*.

#### Migrate from crmsh to pcs

From Ubuntu 23.04 Lunar Lobster onwards, `pcs` is the recommended and supported tool for setting up and managing Corosync/Pacemaker clusters in Ubuntu. This is the final Ubuntu release where `crmsh` will be supported (but not recommended) so users will have time to migrate away from `crmsh`.

The migration from `crmsh` to `pcs` is not very complex since both have a similar command-line interface (CLI). Here is a direct mapping of some useful commands from `crmsh` to `pcs`.

| Action | `crmsh` |
|---|---|
| Show configuration (raw XML) | `crm configure show xml` |
| Show configuration (human-friendly) | `crm configure show` |
| Show cluster status | `crm status` |
| Put a node in standby mode | `crm node standby NODE` |
| Remove a node from standby mode | `crm node online NODE` |
| Set cluster property | `crm configure property PROPERTY=VALUE` |
| List resource agent classes | `crm ra classes` |

Table 1 – continued from p

| Action | crmsh |
|---|---|
| List available resource agents by standard | `crm ra list ocf` |
| List available resource agents by OCF provider | `crm ra list ocf pacemaker` |
| List available resource agent parameters | `crm ra info AGENT` |
| Show available fence agent parameters | `crm ra info stonith:AGENT` |
| Create a resource | `crm configure primitive NAME AGENT params PA` |
| Show configuration of all resources | `crm configure show` |
| Show configuration of one resource | `crm configure show RESOURCE` |
| Show configuration of fencing resources | `crm resource status` |
| Start a resource | `crm resource start RESOURCE` |
| Stop a resource | `crm resource stop RESOURCE` |
| Remove a resource | `crm configure delete RESOURCE` |
| Modify a resource's instance parameters | `crm resource param RESOURCE set PARAMETER=VA` |
| Delete a resource's instance parameters | `crm resource param RESOURCE delete PARAMETER` |
| List current resource defaults | `crm configure show type:rsc_defaults` |
| Set resource defaults | `crm configure rsc_defaults OPTION=VALUE` |
| List current operation defaults | `crm configure show type:op_defaults` |
| Set operation defaults | `crm configure op_defaults OPTION=VALUE` |
| Clear fail counts for a resource | `crm resource cleanup RESOURCE` |
| Create a colocation constraint | `crm configure colocation NAME INFINITY: RESO` |
| Create an ordering constraint | `crm configure order NAME mandatory: RESOURCE` |
| Create a location constraint | `crm configure location NAME RESOURCE 50: NOD` |
| Move a resource to a specific node | `crm resource move RESOURCE NODE` |
| Move a resource away from its current node | `crm resource ban RESOURCE NODE` |
| Remove any constraints created by moving a resource | `crm resource unmove RESOURCE` |

### See also

- How-to: *Set up a distributed replicated block device (DRBD)*
- Explanation: *High Availability*

## 3.5. Other tools

This section contains suggestions of other useful tools for system administrators.

### 3.5.1. Other tools

Byobu is a useful tool for system administrators who need to execute multiple shells in one terminal.

- *Byobu*

pam_motd is a PAM module that allows customised "Message Of The Day" (MOTD) messages to be shown.

- *pam_motd*

sudo-rs is a safety oriented and memory safe implementation of sudo written in Rust.

- *sudo-rs*

### Byobu

One of the most useful applications for any system administrator is an xterm multiplexer such as screen or tmux. It allows for the execution of multiple shells in one terminal. To make some of the advanced multiplexer features more user-friendly and provide some useful information about the system, the byobu package was created. It acts as a wrapper to these programs. By default Byobu is installed in Ubuntu server and it uses tmux (if installed) but this can be changed by the user.

Invoke it simply with:

```
byobu
```

Now bring up the configuration menu. By default this is done by pressing the *F9* key. This will allow you to:

- Help – Quick Start Guide
- Toggle status notifications
- Change the escape sequence
- Byobu currently does not launch at login (toggle on)

byobu provides a menu which displays the Ubuntu release, processor information, memory information, and the time and date. The effect is similar to a desktop menu.

Using the *"Byobu currently does not launch at login (toggle on)"* option will cause byobu to be executed any time a terminal is opened. Changes made to byobu are on a per user basis, and will not affect other users on the system.

One difference when using byobu is the *scrollback* mode. Press the *F7* key to enter scrollback mode. Scrollback mode allows you to navigate past output using *vi* like commands. Here is a quick list of movement commands:

- *h* - Move the cursor left by one character
- *j* - Move the cursor down by one line
- *k* - Move the cursor up by one line
- *l* - Move the cursor right by one character
- *0* - Move to the beginning of the current line
- *$* - Move to the end of the current line
- *G* - Moves to the specified line (defaults to the end of the buffer)
- */* - Search forward
- *?* - Search backward
- *n* - Moves to the next match, either forward or backward

### Resources

- For more information on screen see the screen web site.
- And the Ubuntu Wiki screen page.
- Also, see the byobu project page for more information.

### pam_motd

When logging into an Ubuntu server you may have noticed the informative Message Of The Day (MOTD). This information is obtained and displayed using a couple of packages:

- *landscape-common:* provides the core libraries of landscape-client, which is needed to manage systems with Landscape (proprietary). Yet the package also includes the landscape-sysinfo utility which is responsible for displaying core system data involving cpu, memory, disk space, etc. For instance:

```
      System load:  0.0               Processes:           76
      Usage of /:   30.2% of 3.11GB   Users logged in:     1
      Memory usage: 20%               IP address for eth0: 10.153.107.115
      Swap usage:   0%

      Graph this data and manage this system at https://landscape.canonical.
com/
```

> **Note**
>
> You can run landscape-sysinfo manually at any time.

- *update-notifier-common:* provides information on available package updates, impending *filesystem checks (fsck)*, and required reboots (e.g.: after a kernel upgrade).

pam_motd executes the scripts in `/etc/update-motd.d` in order based on the number prepended to the script. The output of the scripts is written to `/var/run/motd`, keeping the numerical order, then concatenated with `/etc/motd.tail`.

You can add your own dynamic information to the MOTD. For example, to add local weather information:

- First, install the weather-util package:

```
sudo apt install weather-util
```

- The weather utility uses METAR data from the National Oceanic and Atmospheric Administration and forecasts from the National Weather Service. In order to find local information you will need the 4-character ICAO location indicator. This can be determined by browsing to the National Weather Service site.

  Although the National Weather Service is a United States government agency there are weather stations available world wide. However, local weather information for all locations outside the U.S. may not be available.

- Create `/usr/local/bin/local-weather`, a simple shell script to use weather with your local ICAO indicator:

```
#!/bin/sh
#
#
# Prints the local weather information for the MOTD.
```

(continues on next page)

(continued from previous page)

```
#
#

# Replace KINT with your local weather station.
# Local stations can be found here: http://www.weather.gov/tg/siteloc.shtml

echo
weather KINT
echo
```

- Make the script executable:

```
sudo chmod 755 /usr/local/bin/local-weather
```

- Next, create a symlink to `/etc/update-motd.d/98-local-weather`:

```
sudo ln -s /usr/local/bin/local-weather /etc/update-motd.d/98-local-weather
```

- Finally, exit the server and re-login to view the new MOTD.

You should now be greeted with some useful information, and some information about the local weather that may not be quite so useful. Hopefully the local-weather example demonstrates the flexibility of pam_motd.

### Resources

- See the `update-motd(5)` manual page for more options available to update-motd.
- The Debian Package of the Day weather article has more details about using the weatherutility.

## sudo-rs

This page serves as a reference for the key differences between `sudo.ws` and `sudo-rs`.

Note: Both projects are under active development, so it is not possible to maintain a fully up-to-date list of differences. This is a list of major differences as of the Ubuntu 25.10 release.

For the most accurate and current information, refer to `sudo-rs --help` for a list of supported options in your installed version. Refer to `man sudoers-rs` for the `/etc/sudoers` configuration options supported by `sudo-rs`.

### Differences

1. Start with the official documentation from the `sudo-rs` project.
    - differences-from-original-sudo
    - aim-of-the-project
2. `sudo-rs` prompt.

This is the most common error users encounter when using Expect-based automation. The error often is a TIMEOUT because Expect is pattern matching on the `sudo.ws` prompt.

`sudo.ws` prompt for password says `[sudo] password for <USERNAME>`, whereas `sudo-rs` prompt says `[sudo: authenticate] <METHOD>:`. `sudo-rs` transparently prints whatever PAM says such as `Password:`, `PIN:`, etc.

You can use `--prompt ""` in Expect-based scripts to skip the regex-based matching of the prompt.

See more information

3. I/O logging and `sudoreplay` is not supported. The programs are `sudo_logsrvd`, `sudo_sendlog`, and `sudoreplay`.
4. There is no `sudoers.ldap`. You need to use LDAP authentication via PAM.
5. The `sudo-rs` team maintains the list of CLI flags parity with `sudo.ws`

# 4. Ubuntu Server explanation guides

Our explanatory and conceptual guides are written to provide a better understanding of how Ubuntu Server works and how it can be used and configured. They enable you to expand your knowledge, making the operating system easier to use.

If you're not sure how or where to get started with a topic, each section has "introduction to" pages to give you a high-level overview and relevant links to help you navigate to the guides and other materials of most interest to you.

## 4.1. Security

*Our security explanations* include high-level overviews of security configuration and good practices, and discussion of key concepts in the following topics:

- **Authentication**, with introductions to Kerberos and SSSD
- **Cryptography** and cryptographic libraries
- **Virtual Private Networks (VPNs)** with introductions to WireGuard VPN and its related concepts

### 4.1.1. Security

There are many steps you can take to strengthen the security posture of your system. In this section you will find explanations of various concepts related to security.

#### General configuration

- *Introduction to security* for a high-level overview of security in Ubuntu
- *Security suggestions* provides general recommendations for any Ubuntu system, from straightforward setups to more advanced and complex systems

#### Introduction to security

Security should always be considered when you install, deploy, and use an Ubuntu system. While a fresh installation of Ubuntu is usually safe for immediate use, your system's security posture will be very important depending on how it is used after it is deployed. You should always employ a layered approach to security so that that your system does not rely on a single point of defense.

#### Server security guidance

Given Ubuntu has a wide range of customization options, this introduction cannot provide a comprehensive security hardening guide. However, the security of nearly any Ubuntu system can be enhanced by implementing certain best practices and security-related packages. Our *security suggestions* page outlies those best practices and packages. While you may not choose to implement every suggestion – and this list is not exhaustive – each one can contribute an additional layer of security.

If your Ubuntu setup is more advanced or more complex than others, you may need to do more research into your security outlook. There are specific packages available for Ubuntu Servers that can provide additional security, and we suggest some packages in the *advanced*

*security* section that you might want to investigate. Again, the list in this section is not intended to be exhaustive, but rather a starting point.

For the most thorough treatment of security in Ubuntu, we also recommend reading the Ubuntu Security documentation.

### Ubuntu Pro

Canonical offers security, compliance and support services through the Ubuntu Pro subscription. Ubuntu Pro is available for free on up to 5 machines (for business or personal use). Although the compliance and certification features of Ubuntu Pro are likely to be of more interest to enterprise users, the enhanced security coverage is great for anyone using Ubuntu.

All of the Ubuntu Pro features can be managed on the command line via the Ubuntu Pro Client utility, which has an API that provides easier automation to users.

### Vulnerability management

In a standard Ubuntu LTS release, security support is provided for packages in the Main repository for 5 years. With Ubuntu Pro, this support is expanded to 10 years, and also includes patching for medium, high and critical severity vulnerabilities for the Universe repository. This service, known as Expanded Security Maintenance (*ESM*), is recommended for every Ubuntu system. Learn more about ESM here.

### Kernel application hardening

We also recommend Canonical's Livepatch service for every Ubuntu system, which applies kernel patches for high and critical severity vulnerabilities while the system is running, without the need for an immediate reboot – which reduces downtime. Learn more about Livepatch here.

### Security Compliance and Certification

For enterprise users who must ensure compliance with specific standards, such as FIPS, CIS and DISA STIG, Ubuntu also provides profile benchmarking. See our security and compliance documentation for more details.

### Reporting vulnerabilities

If you need to report a security issue, refer to the security disclosure policy.

### Security suggestions

Although a fresh install of Ubuntu is relatively safe for immediate use on the Internet, in this guide we'll take a look at some steps you can take to help keep your Ubuntu system safe and secure.

### For any Ubuntu system

The following suggestions are applicable generally to most Ubuntu systems. It is not necessary to use all of these steps – use the ones that are most relevant for your setup.

### Keep your system up-to-date

1. **Regularly update** your Ubuntu system to keep it protected from known vulnerabilities. Run the following command periodically to update your system software:

   ```
   sudo apt update && sudo apt upgrade
   ```

   You may want to use the `unattended-upgrade` package to fetch and install security updates and bug fixes automatically:

   ```
   sudo apt install unattended-upgrades
   ```

   By default, `unattended-upgrade` runs daily, but this can be configured. See the `unattended-upgrades(8)` manual page for details.

2. **Manage your software**:

   - Remove packages you don't need, to minimise the potential attack surface you are exposing. See our article on *Package management* for more details.
   - Avoid using third party repositories. If you need to download a package from a third party repository, make sure you *understand the risks and how to minimize them.*.

3. **Use the most up-to-date release** of Ubuntu. If you are on an older Ubuntu release we have instructions on *how to upgrade*.

4. **Use Ubuntu Pro**, particularly if you are on an older release of Ubuntu. Pro provides Enterprise-level security patching, but is free for personal/business use on up to 5 machines. The most useful Pro features for *any* Ubuntu Server are:

   - Expanded Security Maintenance (ESM) which expands the Ubuntu LTS commitment on packages in Main from 5 years to 10 years – and now also covers packages in Ubuntu Universe.
   - Livepatch applies kernel patches for high and critical severity vulnerabilities while the system is running. This avoids the need for an immediate reboot.

   Most security patches can be fetched and applied automatically through the `unattended-upgrade` package. For more details on using and monitoring Ubuntu Pro via the command line, refer to the official documentation.

### Access Control

1. **Use and enforce** the principle of least privilege:

   - This means creating non-root user accounts with as few privileges as possible.
   - Not using `sudo` (root access) except for administration tasks.
   - For more details on basic access control, see our *guide on user management*.

### Network security

1. **Use a firewall**. In Ubuntu, the uncomplicated firewall (`ufw`) tool is used to configure firewalls. `ufw` is a wrapper around the `iptables` utility (which experienced system admins may prefer to use directly). To get started with `ufw`, check out our *Firewall* guide.

2. **Use the Secure Shell (SSH)** protocol to secure remote access. In Ubuntu, this is managed through OpenSSH. For details on setting up OpenSSH, refer to our *guide to OpenSSH*.

### Physical security

There are also steps you can take to protect the physical security of your system. These how-to guides will help you set up these additional precautions:

- *Smart card authentication*.
- *Smart card authentication with SSH*.
- *Console security*.

### Suggestions for complex setups

The following section will help direct you to the security-related packages for which we provide documentation. For more discussion about advanced security considerations, refer to the Ubuntu Security documentation.

### Advanced Access Control

1. **Lightweight Directory Access Protocol (LDAP)** is the usual way to gate access control for larger or more complex setups. In Ubuntu, this is implemented through OpenLDAP. Refer to our *introduction to OpenLDAP* for more details, or see our section *on how to set up OpenLDAP*.
2. **Kerberos** is a network authentication protocol that provides identity verification for distributed environments, commonly used in enterprise systems. Learn more in our *introduction to Kerberos*, or see our section on how to *set up and use Kerberos*.
3. **System Security Services Daemon (SSSD)** is a collection of daemons that handle authentication, authorisation and user/group information from disparate network sources. It integrates with OpenLDAP, Kerberos, and Active Directory as we discuss in more detail in our *introduction to SSSD* or get started setting it up with our *how-to section*.
4. **Active Directory**, the directory service for Windows domain networks, can be set up to allow your Ubuntu Server to integrate with a Windows network. Find out more about *Active Directory integration*, or *learn how to set it up*. You may also find *ADSys*, the Group Policy client for Ubuntu, helpful. See the ADSys documentation for more details.
5. **AppArmor** is strongly recommended in order to limit the system access and capabilities of the software running on your systems. It enforces Mandatory Access Control (MAC) policies for individual applications to ensure that even if an application is compromised, the amount of damage caused is limited. We have a how-to guide that will show you *how to set up AppArmor*.

### Security of communications

VPNs are an important tool for system administrators. They provide encrypted, secure connections between a network and the users connecting to it. Two of the most popular choices in Ubuntu are **WireGuard VPN** and **OpenVPN**.

1. **WireGuard VPN** is a modern and performant option, and it removes much of the complexity from the configuration and deployment of a VPN. To get an overview, see our *Introduction to WireGuard VPN*. You can then find out how to *set up WireGuard VPN*.
2. **OpenVPN** is a well-established and widely supported option with a large user base. It supports many platforms besides Linux, including Windows, macOS and iOS. Find out more *about the available clients*, or see our guide on *how to install OpenVPN*.

It's also important to consider **Transport Layer Security (TLS)** and/or **Secure Sockets Layer (SSL)** for securely encrypting data in transit. These cryptographic protocols provide privacy, integrity and authenticity to the communications being passed between two clients, or between a client and server. The exact implementation you choose will depend upon your setup, and there are many options available.

### Cryptography

There are many **cryptographic libraries** available in Ubuntu. For an overview of the most common ones, including some more details about TLS and SSL, refer to our page *about crypto libraries*. For a more high level overview of cryptographic libraries in general, see our *Introduction to cryptographic libraries*

Of course, no discussion of cryptography would be complete without including **certificates**. For more details about what certificates are and how they are used, see our *About certificates* page. Alternatively, if you are familiar with the concepts of certificates and Certification Authorities (CA), our how-to guide will show you how to *Install root CA certificate in the trust store*

### Compliance and auditing

If you need to adhere to specific industry standards, or are otherwise operating in a high security environment, refer to the Ubuntu Security documentation.

### Authentication

- *Introduction to Kerberos*, the network authentication system
- *Introduction to SSSD*, the collection of daemons that handle authentication from various network sources
- *About DNSSEC*, the security extension for the Domain Name System (DNS).

### Introduction to Kerberos

Kerberos is a network authentication system based on the principal of a trusted third party. The other two parties being the user and the service the user wishes to authenticate to. Not all services and applications can use Kerberos, but for those that can, it brings the network environment one step closer to being Single Sign On (SSO).

This section covers installation and configuration of a Kerberos server, and some example client configurations.

## Overview

If you are new to Kerberos there are a few terms that are good to understand before setting up a Kerberos server. Most of the terms will relate to things you may be familiar with in other environments:

- *Principal:* any users, computers, and services provided by servers need to be defined as Kerberos Principals.
- *Instances:* are a variation for service principals. For example, the principal for an NFS service will have an instance for the *hostname* of the server, like `nfs/server.example.com@REALM`. Similarly admin privileges on a principal use an instance of `/admin`, like `john/admin@REALM`, differentiating it from `john@REALM`. These variations fit nicely with *ACL*s.
- *Realms:* the unique realm of control provided by the Kerberos installation. Think of it as the domain or group your hosts and users belong to. Convention dictates the realm should be in uppercase. By default, Ubuntu will use the *DNS* domain converted to uppercase (`EXAMPLE.COM`) as the realm.
- *Key Distribution Center:* (KDC) consist of three parts: a database of all principals, the authentication server, and the ticket granting server. For each realm there must be at least one KDC.
- *Ticket Granting Ticket:* issued by the Authentication Server (AS), the Ticket Granting Ticket (TGT) is encrypted in the user's password which is known only to the user and the KDC. This is the starting point for a user to acquire additional tickets for the services being accessed.
- *Ticket Granting Server:* (TGS) issues service tickets to clients upon request.
- *Tickets:* confirm the identity of the two principals. One principal being a user and the other a service requested by the user. Tickets establish an encryption key used for secure communication during the authenticated session.
- *Keytab Files:* contain encryption keys for a service or host extracted from the KDC principal database.

To put the pieces together, a Realm has at least one KDC, preferably more for redundancy, which contains a database of Principals. When a user principal logs into a workstation that is configured for Kerberos authentication, the KDC issues a Ticket Granting Ticket (TGT). If the user supplied credentials match, the user is authenticated and can then request tickets for Kerberized services from the Ticket Granting Server (TGS). The service tickets allow the user to authenticate to the service without entering another username and password.

## Resources

- For more information on MIT's version of Kerberos, see the MIT Kerberos site.
- Also, feel free to stop by the *#ubuntu-server* and *#kerberos* IRC channels on Libera.Chat if you have Kerberos questions.
- Another guide for installing Kerberos on Debian, includes PKINIT

### Introduction to network user authentication with SSSD

The System Security Services Daemon (SSSD) is a collection of daemons that handle authentication, authorisation, and user and group information from a variety of network sources. It's a useful tool for administrators of Linux and UNIX-based systems, particularly if enterprise systems need to integrate with other directory, access control and authentication services.

### Common deployment scenarios

The SSSD supports a variety of authorisation and identity services, such as Active Directory, LDAP, and Kerberos. The following guides will help you set up SSSD for:

- *Active Directory*
- *LDAP*
- *LDAP and Kerberos*

### Integration with PAM and NSS

If you need to integrate remote sources into your system, SSSD's Pluggable Authentication Modules (PAM) and Name Service Switch (NSS) modules allow you to recognize remote users as valid users and identify them as members in user groups. In the event of network failure or other related problems, SSSD also can also cache this information, so that users can continue to login to the system.

### Troubleshooting

Canonical's *SSSD troubleshooting guide* can provide you with information on managing problems with SSSD.

### DNS Security Extensions (DNSSEC)

DNSSEC is a security extension for the Domain Name System (DNS). DNS is a mapping between names and Internet Protocol (IP) addresses, allowing the use of friendly names instead of sequences of numbers when reaching web sites. Additionally, it stores extra information about a given domain, such as:

- who the point of contact is
- when it was last updated
- what are the authoritative name servers for the domain
- what are the mail exchangers for the domain (i.e., which systems are responsible for email for this domain)
- how host names are mapped to IP addresses, and vice-versa
- what information about other services, such as kerberos realms, LDAP servers, etc (usually on internal domains only) are there
- and more

When DNS was first conceived, security wasn't a top priority. At its origins, DNS is susceptible to multiple vulnerabilities, and has many weaknesses. Most of them are a consequence of

spoofing: there is no guarantee that the reply you received to a DNS query was not tampered with or that it came from the true source.

This is not news, and other mechanisms on top of DNS and around it are in place to counteract that weakness. For example, the famous *HTTPS* padlock that can be seen when accessing most websites nowadays, which uses the TLS protocol to both authenticate the website, and encrypt the connection. It doesn't prevent DNS spoofing, and your web browser might still be tricked into attempting a connection with a fraudulent website, but the moment the TLS certificate is inspected, a warning will be issued to the user. Depending on local policies, the connection might be even immediately blocked. Still, DNS spoofing is a real problem, and TLS itself is subject to other types of attacks.

### What is it?

DNSSEC, which stands for Domain Name System Security Extensions, is an extension to DNS that introduces digital signatures. This allows each DNS response to be verified for:

- integrity: The answer was not tampered with and did not change during transit.
- authenticity: The data came from the true source, and not another entity impersonating the source.

It's important to note that DNSSEC, however, will NOT encrypt the data: it is still sent in the clear.

DNSSEC is based on public key cryptography, meaning that every DNS zone has a public/private key pair. The private key is used to sign the zone's DNS records, and the corresponding public key can be used to verify those signatures. This public key is also published in the zone. Anyone querying the zone can also fetch the public key to verify the signature of the data.

A crucial question arises: How can we trust the authenticity of this public key? The answer lies in a hierarchical signing process. The key is signed by the parent zone's key, which, in turn, is signed by its parent, and so on, forming the "chain of trust", ensuring the integrity and authenticity of DNS records. The **root zone keys** found at the root DNS zone are implicitly trusted, serving as the foundation of the trust chain.

The public key cryptography underpinning SSL/TLS operates in a similar manner, but relies on Certificate Authority (CA) entities to issue and vouch for certificates. Web browsers and other SSL/TLS clients must maintain a list of trusted CAs, typically numbering in the dozens. In contrast, DNSSEC simplifies this process by requiring only a single trusted root zone key.

For a more detailed explanation of how the DNSSEC validation is performed, please refer to the Simplified 12-step DNSSEC validation process guide from ISC.

### DNS daemons

Ubuntu supports multiple DNS resolvers, covering a variety of use cases. Most of them support DNSSEC validation, but it might not be activated and set up with valid trust-anchors automatically.

| Daemon | Type | DNSSEC support |
|---|---|---|
| sys-temd□resolv | Stub Resolver (local) | Yes. Disabled by default. Controlled via `DNSSEC=...` setting in `/etc/systemd/resolved.conf.d`. |
| dnsmasq | Stub Resolver | Yes. Disabled by default. Controlled via `dnssec` and `conf-file=../trust-anchors.conf` settings. |
| bind9 | Recursive Resolver | Yes. Enabled by default via `dnssec-validation auto;` setting. |

The **systemd-resolved** stub resolver is pre-installed in Ubuntu but ships with DNSSEC validation disabled by default. It supports two optional DNSSEC modes: fallback or strict validation. Either mode can be configured using the `DNSSEC=` setting, e.g. in a new `/etc/systemd/resolved.conf.d/10-dnssec.conf` drop-in configuration (see example below).

- You can enable the **fallback mode**, using the `DNSSEC=allow-downgrade` setting, which tries to validate the DNSSEC records whenever possible, but at the same time accepts unsigned responses for backwards compatibility with unsigned zones.
- Should authenticity of DNS records be a concern to you, it's advised to opt-in to the **strict mode** by using the `DNSSEC=yes` setting and reloading the configuration.

For example, here is how the strict mode can be enabled. To use the fallback mode instead, replace `DNSSEC=yes` with `DNSSEC=allow-downgrade`:

```
$ sudo mkdir -p /etc/systemd/resolved.conf.d
$ sudo tee /etc/systemd/resolved.conf.d/10-dnssec.conf >/dev/null <<EOF
[Resolve]
DNSSEC=yes
EOF

$ sudo systemctl reload systemd-resolved.service
```

Once reloaded, the functionality of DNSSEC in **systemd-resolved** can be confirmed, using the `dig` command on the local stub resolver at `127.0.0.53`, by checking for the existence of the **ad** (Authenticated Data) flag:

```
$ dig @127.0.0.53 isc.org +dnssec
[...]
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 1
```

DNSSEC validation will reject DNS resource records, originating from a DNS resolver that indicates DNSSEC support through the `EDNS0` "DNSSEC OK" (`DO`) flag (c.f. RFC 3225), but at the same time does not properly respond to DNSSEC queries (e.g. the missing "authenticated data" = `AD` flag or missing `RRSIG` or `NSEC` records). In the past this has led to issues, especially in virtualization environments or when edge routers are involved, e.g. those provided by ISPs for home connectivity (ref1, ref2, ref3, ref4). Therefore, DNSSEC should only be enabled in controlled environments, where the upstream DNS server is correctly configured to handle DNSSEC queries.

**Warning**

Be aware that enforcing DNSSEC in strict mode can lead to errors, especially for local,

unsigned domains and you would only be able to reach such services by accessing them through their IP address directly. For example this could manifest itself by errors like `DNS_PROBE_FINISHED_NXDOMAIN` in your browser, when trying to access services in the local network.

Once strict DNSSEC validation is enabled, you should also be able to confirm it through higher level checks in your browser, e.g. using those 3rd party services:

- https://internet.nl/test-connection/
- https://wander.science/projects/dns/dnssec-resolver-test/

**Note**

In case of issues with Domain Name resolution, make sure to remove relevant drop-in configs in `/etc/resolved.conf.d` and execute `systemctl reload systemd-resolved.service`. This will reset any DNSSEC configuration to the default `DNSSEC=no` and can be confirmed by executing `resolvectl dnssec`.

### New Resource Records (RRs)

DNSSEC introduces a set of new Resource Records. Here are the most important ones:

- RRSIG: Resource Record Signature. Each RRSIG record matches a corresponding Resource Record, i.e., it's the digital cryptographic signature of that Resource Record.
- DNSKEY: There are several types of keys used in DNSSEC, and this record is used to store the public key in each case.
- DS: Delegation Signer. This stores a secure delegation, and is used to build the authentication chain to child DNS zones. This makes it possible for a parent zone to "vouch" for its child zone.
- NSEC, NSEC3, NSEC3PARAM: Next Secure record. These records are used to prove that a DNS name does not exist.

For instance, when a DNSSEC-aware client queries a Resource Record that is signed, the corresponding RRSIG record is also returned:

```
$ dig @1.1.1.1 +dnssec -t MX isc.org

; <<>> DiG 9.18.28-0ubuntu0.24.04.1-Ubuntu <<>> @1.1.1.1 +dnssec -t MX isc.org
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 51256
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 1232
;; QUESTION SECTION:
;isc.org.                       IN      MX
```

(continues on next page)

(continued from previous page)

```
;; ANSWER SECTION:
isc.org.                300     IN      MX      5 mx.pao1.isc.org.
isc.org.                300     IN      MX      10 mx.ams1.isc.org.
isc.org.                300     IN      RRSIG   MX 13 2 300 20241029080338
20241015071431 27566 isc.org. LG/
cvFmZ8jLz+CM14foaCtwsyCTwKXfVBZV2jcl2UV8zV79QRLs0YXJ3 sjag1vYCqc+Q5AwUi2DB8L/
wZR6EJQ==

;; Query time: 199 msec
;; SERVER: 1.1.1.1#53(1.1.1.1) (UDP)
;; WHEN: Tue Oct 22 16:44:33 UTC 2024
;; MSG SIZE  rcvd: 187
```

### Other uses

DNSSEC makes it more attractive and secure to store other types of information in DNS zones. Although it has always been possible to store SRV, TXT and other generic records in DNS, now these can be signed, and can thus be relied upon to be true. A well known initiative that leverages DNSSEC for this purpose is DANE: DNS-based Authentication of Named Entities (RFC 6394, RFC 6698, RFC 7671, RFC 7672, RFC 7673).

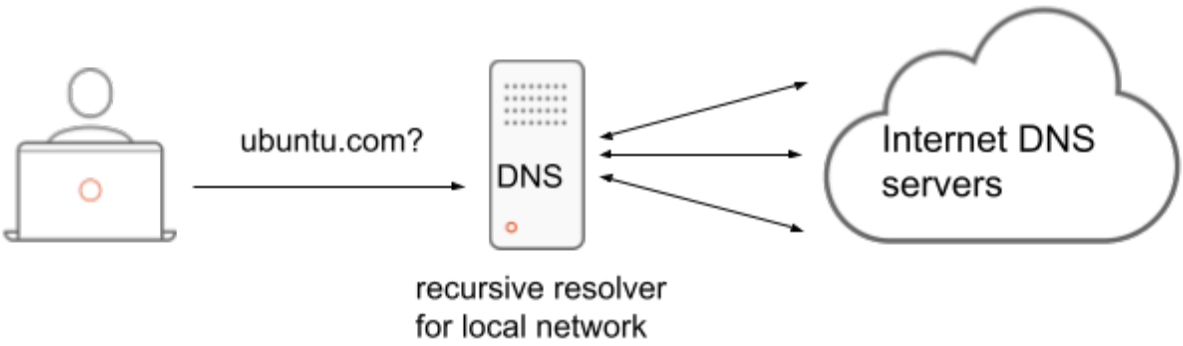
For example, consider a scenario where SSH host keys are stored and secured in DNS using DNSSEC. Rather than manually verifying a host key fingerprint, the verification process could be automated using DNSSEC and the SSHFP Resource Record published in the DNS zone for that host. OpenSSH already supports this feature through the `VerifyHostKeyDNS` configuration option.

### Where does the DNSSEC validation happen?

DNSSEC validation involves fetching requested DNS data, retrieving their corresponding digital signatures, and cryptographically verifying them. Who is responsible for this process?

It depends.

Let's analyze the simple scenario of a system on a local network performing a DNS query for a domain.

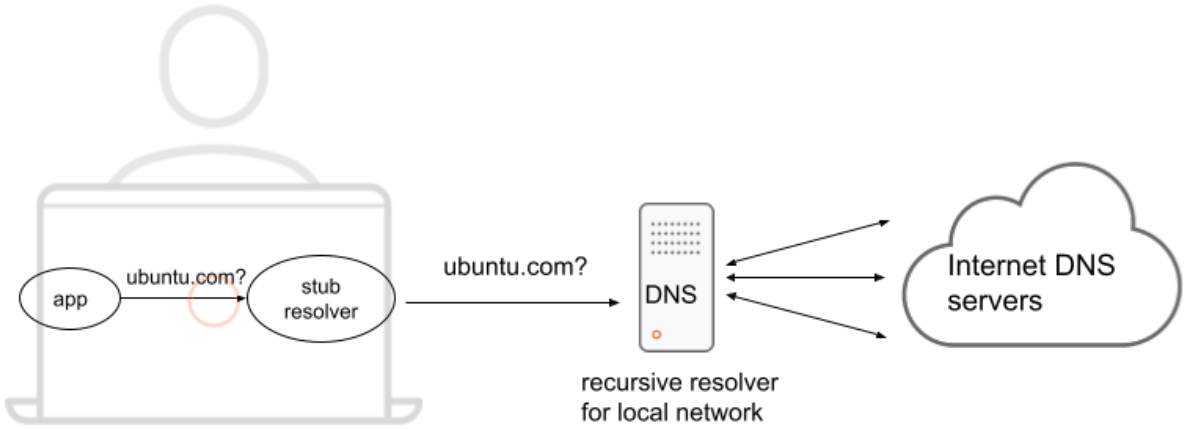



Here we have:

- An Ubuntu system, like a desktop, configured to use a DNS server in the local network.

- A DNS server configured to perform recursive queries on behalf of the clients from the local network.

Let's zoom in a little bit on that Ubuntu system:




To translate a *hostname* into an IP address, applications typically rely on standard glibc functions. This process involves a stub resolver, often referred to as a DNS client. A stub resolver is a simple client that doesn't perform recursive queries itself; instead, it delegates the task to a recursive DNS server, which handles the complex query resolution.

In Ubuntu, the default stub resolver is `systemd-resolved`. That's a daemon, running locally, and listening on port 53/udp on IP 127.0.0.53. The system is configured to use that as its nameserver via `/etc/resolv.conf`:

```
nameserver 127.0.0.53
options edns0 trust-ad
```

This stub resolver has its own configuration for which recursive DNS servers to use. That can be seen with the command `resolvectl`. For example:

```
Global
         Protocols: -LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported
  resolv.conf mode: stub

Link 12 (eth0)
    Current Scopes: DNS
         Protocols: +DefaultRoute -LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported
Current DNS Server: 10.10.17.1
       DNS Servers: 10.10.17.1
        DNS Domain: lxd
```

This configuration is usually provided via *DHCP*, but could also be set via other means. In this particular example, the DNS server that the stub resolver (`systemd-resolved`) will use for all queries that go out on that network interface is 10.10.17.1. The output above also has `DNSSEC=no/unsupported`: we will get back to that in a moment, but it means that `systemd-resolved` is not performing the DNSSEC cryptographic validation.

Given what we have:

- an application

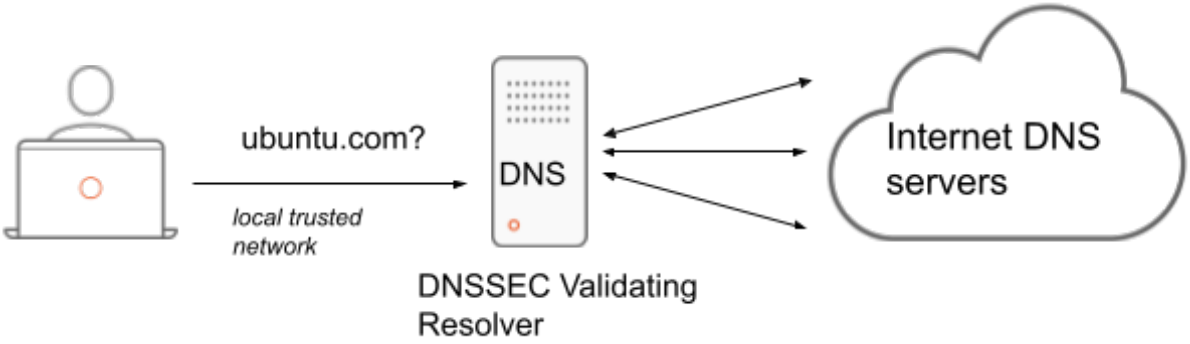
- stub resolver (“DNS client”)
- recursive DNS server in the local network
- several other DNS servers in the internet that will be queried by our recursive DNS server

Where does the DNSSEC validation happen? Who is responsible?

Well, any DNS server can perform the validation. Let’s look at two scenarios, and what it means in each case.

### Validating Resolver

When a recursive DNS server is also performing DNSSEC validation, it’s called a *Validating Resolver*. That will typically be the DNS server on your local network, at your company, or in some cases at your ISP.




This is the case if you install the BIND9 DNS server: the default configuration is to act as a Validating Resolver. This can be seen in `/etc/bind/named.conf.options` after installing the `bind9` package:

```
options {
    ...
    dnssec-validation auto;
    ...
};
```

**Note**

Starting with version `1:9.18.34-1` in Ubuntu 24.10 and above, the `dnssec-validation auto` setting became the implicit default and does not need to be set explicitly in `named.conf.options` anymore.

A critical aspect of this deployment model is the trust in the network segment between the stub resolver and the Validating Resolver. If this network is compromised, the security benefits of DNSSEC can be undermined. While the Validating Resolver performs DNSSEC checks and returns only verified responses, the response could still be tampered with on the final (“last mile”) network segment.

This is where the `trust-ad` setting from `/etc/resolv.conf` comes into play:

```
nameserver 127.0.0.53
options edns0 trust-ad
```

The `trust-ad` setting is documented in the `resolv.conf(5)` manpage. It means that the local resolver will:

- Set the `ad` bit (Authenticated Data) in the outgoing queries.
- Trust the `ad` bit in the responses from the specified nameserver.

When the `ad` bit is set in a DNS response, it means that DNSSEC validation was performed and successful. The data was authenticated.

Specifying `trust-ad` in `/etc/resolv.conf` implies in these assumptions:

- The 127.0.0.53 name server is trusted to set the `ad` flag correctly in its responses. If it performs DNSSEC validation, it is trusted to perform this validation correctly, and set the `ad` flag accordingly. If it does not perform DNSSEC validation, then the `ad` flag will always be unset in the responses.
- The network path between localhost and 127.0.0.53 is trusted.

When using `systemd-resolved` as a stub resolver, as configured above, the network path to the local DNS resolver is inherently trusted, as it is a localhost interface. However, the actual nameserver used is not 127.0.0.53; it depends on `systemd-resolved`'s configuration. Unless local DNSSEC validation is enabled, `systemd-resolved` will strip the ad bit from queries sent to the Validating Resolver and from the received responses.

This is the default case in Ubuntu systems.

Another valid configuration is to not use `systemd-resolved`, but rather point at the Validating Resolver of the network directly, like in this example:

```
nameserver 10.10.17.11
options edns0 trust-ad
```

The `trust-ad` configuration functions similarly to the previous scenario. The `ad` bit is set in outgoing queries, and the resolver trusts the `ad` bit in incoming responses. However, in this case, the nameserver is located at a different IP address on the network. This configuration relies on the same assumptions as before:

- The 10.10.17.11 name server is trusted to perform DNSSEC validation and set the `ad` flag accordingly in its responses.
- The network path between localhost and 10.10.17.11 is trusted.

As these assumptions have a higher chance of not being true, this is not the default configuration.

In any case, having a Validating Resolver in the network is a valid and very useful scenario, and good enough for most cases. And it has the extra benefit that the DNSSEC validation is done only once, at the resolver, for all clients on the network.

### Local DNSSEC validation

Some stub resolvers, such as systemd-resolved, can perform DNSSEC validation locally. This eliminates the risk of network attacks between the resolver and the client, as they reside on the same system. However, local DNSSEC validation introduces additional overhead in the form of multiple DNS queries. For each DNS query, the resolver must fetch the desired record, its digital signature, and the corresponding public key. This process can increase latency, and with multiple clients on the same network requesting the same records, that's duplicated work.

In general, local DNSSEC validation is still the more secure approach, validating and authenticating the DNS resource records end-to-end, without the need to trust any DNS server along the way. Besides this, *DNS-over-TLS (DoT)* or *DNS-over-HTTPS (DoH)* could be used to increase privacy, by encrypting the DNS connection between your local client and the remote Recursive Resolver.

As an example, let's perform the same query using `systemd-resolved` with and without local DNSSEC validation enabled.

Without local DNSSEC validation. First, let's show it's disabled indeed:

```
$ resolvectl dnssec
Global: no
Link 44 (eth0): no
```

Now we perform the query:

```
$ resolvectl query --type=MX isc.org
isc.org IN MX 10 mx.ams1.isc.org                              -- link: eth0
isc.org IN MX 5 mx.pao1.isc.org                               -- link: eth0

-- Information acquired via protocol DNS in 37.2ms.
-- Data is authenticated: no; Data was acquired via local or encrypted transport:
no
-- Data from: network
```

Notice the `Data is authenticated: no` in the result.

Now we enable local DNSSEC validation:

```
$ sudo resolvectl dnssec eth0 true
```

And repeat the query:

```
$ resolvectl query --type=MX isc.org
isc.org IN MX 5 mx.pao1.isc.org                               -- link: eth0
isc.org IN MX 10 mx.ams1.isc.org                              -- link: eth0

-- Information acquired via protocol DNS in 3.0ms.
-- Data is authenticated: yes; Data was acquired via local or encrypted transport:
no
-- Data from: network
```

There is a tiny difference in the output:

```
-- Data is authenticated: yes
```

This shows that local DNSSEC validation was applied, and the result is authenticated.

### What happens when DNSSEC validation fails

When DNSSEC validation fails, how this error is presented to the user depends on multiple factors.

For example, if the DNS client is not performing DNSSEC validation, and relying on a Validating Resolver for that, typically what the client will see is a generic failure. For example:

```
$ resolvectl query www.dnssec-failed.org
www.dnssec-failed.org: resolve call failed: Could not resolve 'www.dnssec-failed.
org', server or network returned error SERVFAIL
```

The Validating Resolver logs, however, will have more details about what happened:

```
Oct 22 17:14:50 n-dns named[285]: validating dnssec-failed.org/DNSKEY: no valid
signature found (DS)
Oct 22 17:14:50 n-dns named[285]: no valid RRSIG resolving 'dnssec-failed.org/
DNSKEY/IN': 68.87.68.244#53
...
Oct 22 17:14:52 n-dns named[285]: broken trust chain resolving 'www.dnssec-failed.
org/AAAA/IN': 68.87.72.244#53
```

In contrast, when DNSSEC validation is being performed locally, the error is more specific:

```
$ sudo resolvectl dnssec eth0 true
$ resolvectl query www.dnssec-failed.org
www.dnssec-failed.org: resolve call failed: DNSSEC validation failed: no-signature
```

But even when the validation is local, simpler clients might not get the full picture, and still just return a generic error:

```
$ host www.dnssec-failed.org
Host www.dnssec-failed.org not found: 2(SERVFAIL)
```

### Further reading

- DNSSEC - What Is It and Why Is It Important
- Tool to visualize the DNSSEC chain of trust of a domain
- DANE
- RFC 4255 - Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints
- *Basic DNSSEC troubleshooting*

### Cryptography

- *Our cryptography section* explains in detail about the different cryptographic libraries and configurations you might encounter.
- *Certificates* are issued, stored and signed by a Certificate Authority (CA) to create trusted connections.

## Cryptography

### Introduction to cryptographic libraries

When choosing a crypto library, the following aspects should be considered to maintain security and compliance. Typically, you will want to answer questions such as: How can we ensure legacy crypto systems with known vulnerabilities are not being used? How can we enforce minimum key size requirements? And what criteria should we use to accept X.509 certificates when connecting to remote servers?

However, the cryptographic library landscape is vast and complex, and there are many crypto libraries available on an Ubuntu system. When an application developer chooses a crypto library, they will consider many aspects, such as:

- Technical requirements
- Language bindings
- License
- Community
- Ease of use
- General availability
- Upstream maintenance

Among the most popular and widely used libraries and frameworks are:

- OpenSSL
- *GnuTLS*
- NSS
- GnuPG
- *gcrypt*

Each one of these has its own implementation details, API, behavior, configuration file, and syntax.

### Determining which libraries are being used by an application

Ultimately, the only reliable way to determine how an application uses cryptography is by reading its documentation or inspecting its source code. But even then, you may discover that the code is not available and sometimes the documentation might lack this information. If you are having problems finding answers to your crypto-related questions, there are some practical checks that can be made.

To find out what libraries are being used, one generally needs to check all of the installed crypto implementations and their respective configurations. To make things even more complicated, sometimes an application implements its own crypto, without using anything external.

If you are having problems finding answers to your crypto-related questions, there are some practical checks that can be made.

### Follow dynamic links

This is the most common way, and very easy to spot via package dependencies and helper tools. A detailed example of this approach to determining crypto is discussed later in this page.

### Check static linking

This is harder, as there is no dependency information in the binary package, and this usually requires inspection of the source package to see build dependencies. A detailed example of this approach is shown later in this page.

### Look at the plugins

The main binary of an application can not depend directly on a crypto library, but it could load dynamic plugins which do. Usually these would be packaged separately, and then we fall under the dynamic or static linking cases above. Note that via such a plugin mechanism, an application could depend on multiple external cryptographic libraries.

### Examine the execution of external binaries

The application could just plain call external binaries at runtime for its cryptographic operations, like calling out to `openssl` or `gnupg` to encrypt/decrypt data. This will hopefully be expressed in the dependencies of the package. If it's not, then it's a bug that should be reported.

### Indirect use of libraries or executables

The application could be using a third party library or executable which in turn could fall into any of the above categories.

### Check documentation more carefully

Read the application documentation again. It might have crypto options directly in its own configuration files, or point at specific crypto configuration files installed on the system. This may also clarify whether the application uses external crypto libraries or it has its own implementation of crypto.

## Detailed examples of how to locate crypto libraries

### Use dpkg to check package dependencies

Since package dependencies are a good way to check what is needed at runtime by the application, you can find out which package owns what file with the `dpkg -S` command. For example:

```
$ dpkg -S /usr/bin/lynx
lynx: /usr/bin/lynx
```

Now that you have the package name, check the package's dependencies. You should also look for `Recommends`, as they are installed by default. Using the current example, we can now do the following:

```
$ dpkg -s lynx | grep -E "^(Depends|Recommends)"
Depends: libbsd0 (>= 0.0), libbz2-1.0, libc6 (>= 2.34), libgnutls30 (>= 3.7.0),
libidn2-0 (>= 2.0.0), libncursesw6 (>= 6), libtinfo6 (>= 6), zlib1g (>= 1:1.1.4),
lynx-common
Recommends: mime-support
```

Now we can see that `lynx` links with `libgnutls30`, which answers our question: `lynx` uses the GnuTLS library for its cryptography operations.

### Using ldd to list dynamic libraries

If a dynamic library is needed by an application, it should always be correctly identified in the list of application package dependencies. When that is not the case, or you need to identify what is needed by some plugin that is not part of the package, you can use some system tools to help identify the dependencies.

In this situation, you can use the `ldd` tool, which is installed in all Ubuntu systems. It lists all the dynamic libraries needed by the given binary, including dependencies of dependencies, i.e. the command is recursive. Going back to the `lynx` example:

```
$ ldd /usr/bin/lynx
    linux-vdso.so.1 (0x00007ffffd2df000)
    libz.so.1 => /lib/x86_64-linux-gnu/libz.so.1 (0x00007feb69d77000)
    libbz2.so.1.0 => /lib/x86_64-linux-gnu/libbz2.so.1.0 (0x00007feb69d64000)
    libidn2.so.0 => /lib/x86_64-linux-gnu/libidn2.so.0 (0x00007feb69d43000)
    libncursesw.so.6 => /lib/x86_64-linux-gnu/libncursesw.so.6
(0x00007feb69d07000)
    libtinfo.so.6 => /lib/x86_64-linux-gnu/libtinfo.so.6 (0x00007feb69cd5000)
    libgnutls.so.30 => /lib/x86_64-linux-gnu/libgnutls.so.30 (0x00007feb69aea000)
    libbsd.so.0 => /lib/x86_64-linux-gnu/libbsd.so.0 (0x00007feb69ad0000)
    libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007feb698a8000)
    libunistring.so.2 => /lib/x86_64-linux-gnu/libunistring.so.2
(0x00007feb696fe000)
    libp11-kit.so.0 => /lib/x86_64-linux-gnu/libp11-kit.so.0 (0x00007feb695c3000)
    libtasn1.so.6 => /lib/x86_64-linux-gnu/libtasn1.so.6 (0x00007feb695ab000)
    libnettle.so.8 => /lib/x86_64-linux-gnu/libnettle.so.8 (0x00007feb69565000)
    libhogweed.so.6 => /lib/x86_64-linux-gnu/libhogweed.so.6 (0x00007feb6951b000)
    libgmp.so.10 => /lib/x86_64-linux-gnu/libgmp.so.10 (0x00007feb69499000)
    /lib64/ld-linux-x86-64.so.2 (0x00007feb69fe6000)
    libmd.so.0 => /lib/x86_64-linux-gnu/libmd.so.0 (0x00007feb6948c000)
    libffi.so.8 => /lib/x86_64-linux-gnu/libffi.so.8 (0x00007feb6947f000)
```

We again see the GnuTLS library (via `libgnutls.so.30`) in the list, and can reach the same conclusion.

Another way to check for such dependencies (without recursion) is via `objdump`. You may need to install it with the `binutils` package, as it's not mandatory. The way to use it is to grep for the `NEEDED` string:

```
$ objdump -x /usr/bin/lynx|grep NEEDED
  NEEDED               libz.so.1
  NEEDED               libbz2.so.1.0
  NEEDED               libidn2.so.0
  NEEDED               libncursesw.so.6
  NEEDED               libtinfo.so.6
  NEEDED               libgnutls.so.30
  NEEDED               libbsd.so.0
  NEEDED               libc.so.6
```

Finally, if you want to see the dependency *tree*, you can use `lddtree` from the `pax-utils` package:

```
$ lddtree /usr/bin/lynx
lynx => /usr/bin/lynx (interpreter => /lib64/ld-linux-x86-64.so.2)
    libz.so.1 => /lib/x86_64-linux-gnu/libz.so.1
    libbz2.so.1.0 => /lib/x86_64-linux-gnu/libbz2.so.1.0
    libidn2.so.0 => /lib/x86_64-linux-gnu/libidn2.so.0
        libunistring.so.2 => /lib/x86_64-linux-gnu/libunistring.so.2
    libncursesw.so.6 => /lib/x86_64-linux-gnu/libncursesw.so.6
    libtinfo.so.6 => /lib/x86_64-linux-gnu/libtinfo.so.6
    libgnutls.so.30 => /lib/x86_64-linux-gnu/libgnutls.so.30
        libp11-kit.so.0 => /lib/x86_64-linux-gnu/libp11-kit.so.0
            libffi.so.8 => /lib/x86_64-linux-gnu/libffi.so.8
        libtasn1.so.6 => /lib/x86_64-linux-gnu/libtasn1.so.6
        libnettle.so.8 => /lib/x86_64-linux-gnu/libnettle.so.8
        libhogweed.so.6 => /lib/x86_64-linux-gnu/libhogweed.so.6
        libgmp.so.10 => /lib/x86_64-linux-gnu/libgmp.so.10
        ld-linux-x86-64.so.2 => /lib64/ld-linux-x86-64.so.2
    libbsd.so.0 => /lib/x86_64-linux-gnu/libbsd.so.0
        libmd.so.0 => /lib/x86_64-linux-gnu/libmd.so.0
    libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6
```

### Check package headers for static linking

Identifying which libraries were used in a static build is a bit more involved. There are two ways, and they are complementary most of the time:

- look for the `Built-Using` header in the binary package
- inspect the `Build-Depends` header in the source package

For example, let's try to discover which crypto libraries, if any, the `rclone` tool uses. First, let's try the packaging dependencies:

```
$ dpkg -s rclone | grep -E "^(Depends|Recommends)"
Depends: libc6 (>= 2.34)
```

Uh, that's a short list. But `rclone` definitely supports encryption, so what is going on? Turns

out this is a tool written in the Go language, and that uses static linking of libraries. So let's try to inspect the package data more carefully, and this time look for the `Built-Using` header:

```
$ dpkg -s rclone | grep Built-Using
Built-Using: go-md2man-v2 (= 2.0.1+ds1-1), golang-1.18 (= 1.18-1ubuntu1), golang-
bazil-fuse (= 0.0~git20160811.0.371fbbd-3), ...
```

Ok, this time we have a lot of information (truncated above for brevity, since it's all in one very long line). If we look at the full output carefully, we can see that `rclone` was built statically using the `golang-go.crypto` package, and documentation about that package and its crypto implementations is what we should look for.

If the `Built-Using` header was not there, or didn't yield any clues, we could try one more step and look for the build dependencies. These can be found in the `debian/control` file of the source package. In the case of `rclone` for Ubuntu Jammy, that can be seen at https://git.launchpad.net/ubuntu/+source/rclone/tree/debian/control?h=ubuntu/jammy-devel#n7, and a quick look at the `Build-Depends` list shows us the `golang-golang-x-crypto-dev` build dependency, whose source package is `golang-go.crypto` as expected:

```
$ apt-cache show golang-golang-x-crypto-dev | grep ^Source:
Source: golang-go.crypto
```

**Note**

If there is no `Source:` line, then it means the name of the source package is the same as the binary package that was queried.

### What's next?

Once you have uncovered which library your application is using, the following guides may help you to understand the associated configuration files and what options you have available (including some handy examples).

- *OpenSSL guide*
- *GnuTLS guide*
- *Troubleshooting TLS/SSL*

## Common crypto libraries

These pages discuss various popular cryptographic libraries that are commonly found in Ubuntu.

### OpenSSL

OpenSSL is probably the most well known cryptographic library, used by thousands of projects and applications.

The OpenSSL configuration file is located at `/etc/ssl/openssl.cnf` and is used both by the library itself and the command-line tools included in the package. It is simple in structure,

but quite complex in the details, and it won't be fully covered here. In particular, we will only cover the settings that control which cryptographic algorithms will be allowed by default.

### Structure of the config file

The OpenSSL configuration file is very similar to a standard INI file. It starts with a nameless default section, not inside any `[section]` block, and after that we have the traditional `[section-name]` followed by the `key = value` lines. The SSL config manpage has all the details.

This is what it looks like:

```
openssl_conf = <name-of-conf-section>

[name-of-conf-section]
ssl_conf = <name-of-ssl-section>

[name-of-ssl-section]
server = <name of section>
client = <name of section>
system_default = <name of section>
```

See how it's like a chain, where a key (`openssl_conf`) points at the name of a section, and that section has a key that points to another section, and so on.

To adjust the algorithms and ciphers used in a SSL/TLS connection, we are interested in the "SSL Configuration" section of the library, where we can define the behavior of server, client, and the library defaults.

For example, in an Ubuntu Jammy installation, we have (omitting unrelated entries for brevity):

```
openssl_conf = openssl_init

[openssl_init]
ssl_conf = ssl_sect

[ssl_sect]
system_default = system_default_sect

[system_default_sect]
CipherString = DEFAULT:@SECLEVEL=2
```

This gives us our first information about the default set of ciphers and algorithms used by OpenSSL in an Ubuntu installation: `DEFAULT:@SECLEVEL=2`. What that means is detailed inside the `SSL_CTX_set_security_level(3)` manpage.

> **Note**
>
> In Ubuntu Jammy, TLS versions below 1.2 are **disabled** in OpenSSL's `SECLEVEL=2` due to this patch.

That default is also set at package building time, and in the case of Ubuntu, it's set to `SECLEVEL=2`.

The list of allowed ciphers in a security level can be obtained with the `openssl ciphers` command (output truncated for brevity):

```
$ openssl ciphers -s -v DEFAULT:@SECLEVEL=2
TLS_AES_256_GCM_SHA384         TLSv1.3 Kx=any      Au=any   Enc=AESGCM(256)
    Mac=AEAD
TLS_CHACHA20_POLY1305_SHA256   TLSv1.3 Kx=any      Au=any   Enc=CHACHA20/
POLY1305(256) Mac=AEAD
TLS_AES_128_GCM_SHA256         TLSv1.3 Kx=any      Au=any   Enc=AESGCM(128)
    Mac=AEAD
ECDHE-ECDSA-AES256-GCM-SHA384  TLSv1.2 Kx=ECDH     Au=ECDSA Enc=AESGCM(256)
    Mac=AEAD
(...)
```

**Note**

The `openssl ciphers` command will output even ciphers that are not allowed, unless the `-s` switch is given. That option tells the command to list only **supported** ciphers.

All the options that can be set in the `system_default_sect` section are detailed in the `SSL_CONF_cmd(3)` manpage.

### Cipher strings, cipher suites, cipher lists

Encrypting data (or signing it) is not a one step process. The whole transformation applied to the source data (until it is in its encrypted form) has several stages, and each stage typically uses a different cryptographic algorithm. The combination of these algorithms is called a cipher suite.

Similar to GnuTLS, OpenSSL also uses the concept of cipher strings to group several algorithms and cipher suites together. The full list of cipher strings is shown in the `openssl ciphers` manpage.

OpenSSL distinguishes the ciphers used with TLSv1.3, and those used with TLSv1.2 and older. Specifically for the `openssl ciphers` command, we have:

- `-ciphersuites`: used for the TLSv1.3 ciphersuites. So far, there are only five listed in the upstream documentation, and the defaults are:

  TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256

- *cipherlist*: this is a plain argument in the command line of the `openssl ciphers` command, without a specific parameter, and is expected to be a list of cipher strings used in TLSv1.2 and lower. The default in Ubuntu Jammy 22.04 LTS is `DEFAULT:@SECLEVEL=2`.

These defaults are built-in in the library, and can be set in `/etc/ssl/openssl.cnf` via the corresponding configuration keys `CipherString` for TLSv1.2 and older, and `CipherSuites` for TLSv1.3. For example:

```
[system_default_sect]
CipherString = DEFAULT:@SECLEVEL=2
CipherSuites = TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256
```

In the end, without other constraints, the library will merge both lists into one set of supported crypto algorithms. If the crypto negotiation in a connection settles on TLSv1.3, then the list of *CipherSuites* is considered. If it's TLSv1.2 or lower, then *CipherString* is used.

### `openssl ciphers` examples

This will list all supported/enabled ciphers, with defaults taken from the library and `/etc/ssl/openssl.cnf`. Since no other options were given, this will include TLSv1.3 ciphersuites and TLSv1.2 and older cipher strings:

```
$ openssl ciphers -s -v
TLS_AES_256_GCM_SHA384          TLSv1.3 Kx=any      Au=any   Enc=AESGCM(256)
    Mac=AEAD
TLS_CHACHA20_POLY1305_SHA256    TLSv1.3 Kx=any      Au=any   Enc=CHACHA20/
POLY1305(256) Mac=AEAD
TLS_AES_128_GCM_SHA256          TLSv1.3 Kx=any      Au=any   Enc=AESGCM(128)
    Mac=AEAD
ECDHE-ECDSA-AES256-GCM-SHA384   TLSv1.2 Kx=ECDH     Au=ECDSA Enc=AESGCM(256)
    Mac=AEAD
ECDHE-RSA-AES256-GCM-SHA384     TLSv1.2 Kx=ECDH     Au=RSA   Enc=AESGCM(256)
    Mac=AEAD
DHE-RSA-AES256-GCM-SHA384       TLSv1.2 Kx=DH       Au=RSA   Enc=AESGCM(256)
    Mac=AEAD
ECDHE-ECDSA-CHACHA20-POLY1305   TLSv1.2 Kx=ECDH     Au=ECDSA Enc=CHACHA20/
POLY1305(256) Mac=AEAD
ECDHE-RSA-CHACHA20-POLY1305     TLSv1.2 Kx=ECDH     Au=RSA   Enc=CHACHA20/
POLY1305(256) Mac=AEAD
DHE-RSA-CHACHA20-POLY1305       TLSv1.2 Kx=DH       Au=RSA   Enc=CHACHA20/
POLY1305(256) Mac=AEAD
ECDHE-ECDSA-AES128-GCM-SHA256   TLSv1.2 Kx=ECDH     Au=ECDSA Enc=AESGCM(128)
    Mac=AEAD
ECDHE-RSA-AES128-GCM-SHA256     TLSv1.2 Kx=ECDH     Au=RSA   Enc=AESGCM(128)
    Mac=AEAD
DHE-RSA-AES128-GCM-SHA256       TLSv1.2 Kx=DH       Au=RSA   Enc=AESGCM(128)
    Mac=AEAD
ECDHE-ECDSA-AES256-SHA384       TLSv1.2 Kx=ECDH     Au=ECDSA Enc=AES(256)
    Mac=SHA384
ECDHE-RSA-AES256-SHA384         TLSv1.2 Kx=ECDH     Au=RSA   Enc=AES(256)
    Mac=SHA384
DHE-RSA-AES256-SHA256           TLSv1.2 Kx=DH       Au=RSA   Enc=AES(256)
    Mac=SHA256
ECDHE-ECDSA-AES128-SHA256       TLSv1.2 Kx=ECDH     Au=ECDSA Enc=AES(128)
    Mac=SHA256
ECDHE-RSA-AES128-SHA256         TLSv1.2 Kx=ECDH     Au=RSA   Enc=AES(128)
    Mac=SHA256
DHE-RSA-AES128-SHA256           TLSv1.2 Kx=DH       Au=RSA   Enc=AES(128)
```

(continues on next page)

(continued from previous page)

```
    Mac=SHA256
ECDHE-ECDSA-AES256-SHA          TLSv1   Kx=ECDH     Au=ECDSA Enc=AES(256)
    Mac=SHA1
ECDHE-RSA-AES256-SHA            TLSv1   Kx=ECDH     Au=RSA   Enc=AES(256)
    Mac=SHA1
DHE-RSA-AES256-SHA              SSLv3   Kx=DH       Au=RSA   Enc=AES(256)
    Mac=SHA1
ECDHE-ECDSA-AES128-SHA          TLSv1   Kx=ECDH     Au=ECDSA Enc=AES(128)
    Mac=SHA1
ECDHE-RSA-AES128-SHA            TLSv1   Kx=ECDH     Au=RSA   Enc=AES(128)
    Mac=SHA1
DHE-RSA-AES128-SHA              SSLv3   Kx=DH       Au=RSA   Enc=AES(128)
    Mac=SHA1
AES256-GCM-SHA384               TLSv1.2 Kx=RSA      Au=RSA   Enc=AESGCM(256)
    Mac=AEAD
AES128-GCM-SHA256               TLSv1.2 Kx=RSA      Au=RSA   Enc=AESGCM(128)
    Mac=AEAD
AES256-SHA256                   TLSv1.2 Kx=RSA      Au=RSA   Enc=AES(256)
    Mac=SHA256
AES128-SHA256                   TLSv1.2 Kx=RSA      Au=RSA   Enc=AES(128)
    Mac=SHA256
AES256-SHA                      SSLv3   Kx=RSA      Au=RSA   Enc=AES(256)
    Mac=SHA1
AES128-SHA                      SSLv3   Kx=RSA      Au=RSA   Enc=AES(128)
    Mac=SHA1
```

Let's filter this a bit, and just as an example, remove all *AES*128 ciphers and SHA1 hashes:

```
$ openssl ciphers -s -v 'DEFAULTS:-AES128:-SHA1'
TLS_AES_256_GCM_SHA384          TLSv1.3 Kx=any      Au=any   Enc=AESGCM(256)
    Mac=AEAD
TLS_CHACHA20_POLY1305_SHA256    TLSv1.3 Kx=any      Au=any   Enc=CHACHA20/
POLY1305(256) Mac=AEAD
TLS_AES_128_GCM_SHA256          TLSv1.3 Kx=any      Au=any   Enc=AESGCM(128)
    Mac=AEAD
ECDHE-ECDSA-AES256-GCM-SHA384   TLSv1.2 Kx=ECDH     Au=ECDSA Enc=AESGCM(256)
    Mac=AEAD
ECDHE-RSA-AES256-GCM-SHA384     TLSv1.2 Kx=ECDH     Au=RSA   Enc=AESGCM(256)
    Mac=AEAD
DHE-RSA-AES256-GCM-SHA384       TLSv1.2 Kx=DH       Au=RSA   Enc=AESGCM(256)
    Mac=AEAD
ECDHE-ECDSA-CHACHA20-POLY1305   TLSv1.2 Kx=ECDH     Au=ECDSA Enc=CHACHA20/
POLY1305(256) Mac=AEAD
ECDHE-RSA-CHACHA20-POLY1305     TLSv1.2 Kx=ECDH     Au=RSA   Enc=CHACHA20/
POLY1305(256) Mac=AEAD
DHE-RSA-CHACHA20-POLY1305       TLSv1.2 Kx=DH       Au=RSA   Enc=CHACHA20/
POLY1305(256) Mac=AEAD
ECDHE-ECDSA-AES256-SHA384       TLSv1.2 Kx=ECDH     Au=ECDSA Enc=AES(256)
    Mac=SHA384
```

(continues on next page)

(continued from previous page)

```
ECDHE-RSA-AES256-SHA384         TLSv1.2 Kx=ECDH     Au=RSA    Enc=AES(256)
    Mac=SHA384
DHE-RSA-AES256-SHA256           TLSv1.2 Kx=DH       Au=RSA    Enc=AES(256)
    Mac=SHA256
AES256-GCM-SHA384               TLSv1.2 Kx=RSA      Au=RSA    Enc=AESGCM(256)
    Mac=AEAD
AES256-SHA256                   TLSv1.2 Kx=RSA      Au=RSA    Enc=AES(256)
    Mac=SHA256
```

Since we didn't use `-ciphersuites`, the TLSv1.3 list was unaffected by our filtering, and still contains the **AES128** cipher. But TLSv1.2 and older no longer have **AES128** or **SHA1**. This type of filtering with '+', '-' and '!' can be done with the TLSv1.2 and older protocols and is detailed in the `openssl ciphers` manpage.

To filter out TLSv1.3 algorithms, there is no such mechanism, and we must list explicitly what we want by using `-ciphersuites`:

```
$ openssl ciphers -s -v -ciphersuites TLS_AES_256_GCM_SHA384:TLS_CHACHA20_
POLY1305_SHA256 'DEFAULTS:-AES128:-SHA1'
TLS_AES_256_GCM_SHA384          TLSv1.3 Kx=any      Au=any    Enc=AESGCM(256)
    Mac=AEAD
TLS_CHACHA20_POLY1305_SHA256    TLSv1.3 Kx=any      Au=any    Enc=CHACHA20/
POLY1305(256) Mac=AEAD
ECDHE-ECDSA-AES256-GCM-SHA384   TLSv1.2 Kx=ECDH     Au=ECDSA  Enc=AESGCM(256)
    Mac=AEAD
ECDHE-RSA-AES256-GCM-SHA384     TLSv1.2 Kx=ECDH     Au=RSA    Enc=AESGCM(256)
    Mac=AEAD
DHE-RSA-AES256-GCM-SHA384       TLSv1.2 Kx=DH       Au=RSA    Enc=AESGCM(256)
    Mac=AEAD
ECDHE-ECDSA-CHACHA20-POLY1305   TLSv1.2 Kx=ECDH     Au=ECDSA  Enc=CHACHA20/
POLY1305(256) Mac=AEAD
ECDHE-RSA-CHACHA20-POLY1305     TLSv1.2 Kx=ECDH     Au=RSA    Enc=CHACHA20/
POLY1305(256) Mac=AEAD
DHE-RSA-CHACHA20-POLY1305       TLSv1.2 Kx=DH       Au=RSA    Enc=CHACHA20/
POLY1305(256) Mac=AEAD
ECDHE-ECDSA-AES256-SHA384       TLSv1.2 Kx=ECDH     Au=ECDSA  Enc=AES(256)
    Mac=SHA384
ECDHE-RSA-AES256-SHA384         TLSv1.2 Kx=ECDH     Au=RSA    Enc=AES(256)
    Mac=SHA384
DHE-RSA-AES256-SHA256           TLSv1.2 Kx=DH       Au=RSA    Enc=AES(256)
    Mac=SHA256
AES256-GCM-SHA384               TLSv1.2 Kx=RSA      Au=RSA    Enc=AESGCM(256)
    Mac=AEAD
AES256-SHA256                   TLSv1.2 Kx=RSA      Au=RSA    Enc=AES(256)
    Mac=SHA256
```

### Config file examples

Let's see some practical examples of how we can use the configuration file to tweak the default cryptographic settings of an application linked with OpenSSL.

Note that applications can still override these settings: what is set in the configuration file merely acts as a default that is used when nothing else in the application command line or its own config says otherwise.

### Only use TLSv1.3

To configure the OpenSSL library to consider TLSv1.3 as the minimum acceptable protocol, we add a `MinProtocol` parameter to the `/etc/ssl/openssl.cnf` configuration file like this:

```
[system_default_sect]
CipherString = DEFAULT:@SECLEVEL=2
MinProtocol = TLSv1.3
```

If you then try to connect securely to a server that only offers, say TLSv1.2, the connection will fail:

```
$ curl https://j-server.lxd/stats
curl: (35) error:0A00042E:SSL routines::tlsv1 alert protocol version

$ wget https://j-server.lxd/stats
--2023-01-06 13:41:50--  https://j-server.lxd/stats
Resolving j-server.lxd (j-server.lxd)... 10.0.100.87
Connecting to j-server.lxd (j-server.lxd)|10.0.100.87|:443... connected.
OpenSSL: error:0A00042E:SSL routines::tlsv1 alert protocol version
Unable to establish SSL connection.
```

### Use only AES256 with TLSv1.3

As an additional constraint, besides forcing TLSv1.3, let's only allow AES256. This would do it for OpenSSL applications that do not override this elsewhere:

```
[system_default_sect]
CipherString = DEFAULT:@SECLEVEL=2
CipherSuites = TLS_AES_256_GCM_SHA384
MinProtocol = TLSv1.3
```

Since we are already forcing TLSv1.3, there is no need to tweak the `CipherString` list, since that applies only to TLSv1.2 and older.

The OpenSSL `s_server` command is very handy to test this (see *the Troubleshooting section* for details on how to use it):

```
$ sudo openssl s_server -cert j-server.pem -key j-server.key -port 443 -www
```

**Note**

Be sure to use another system for this server, or else it will be subject to the same `/etc/ssl/openssl.cnf` constraints you are testing on the client, and this can lead to very confusing results.

As expected, a client will end up selecting TLSv1.3 and the `TLS_AES_256_GCM_SHA384` cipher suite:

```
$ wget https://j-server.lxd/stats -O /dev/stdout -q | grep Cipher -w
New, TLSv1.3, Cipher is TLS_AES_256_GCM_SHA384
    Cipher    : TLS_AES_256_GCM_SHA384
```

To be sure, we can tweak the server to only offer `TLS_CHACHA20_POLY1305_SHA256` for example:

```
$ sudo openssl s_server -cert j-server.pem -key j-server.key -port 443 -www -
ciphersuites TLS_CHACHA20_POLY1305_SHA256
```

And now the client will fail:

```
$ wget https://j-server.lxd/stats -O /dev/stdout
--2023-01-06 14:20:55--  https://j-server.lxd/stats
Resolving j-server.lxd (j-server.lxd)... 10.0.100.87
Connecting to j-server.lxd (j-server.lxd)|10.0.100.87|:443... connected.
OpenSSL: error:0A000410:SSL routines::sslv3 alert handshake failure
Unable to establish SSL connection.
```

### Drop AES128 entirely

If we want to still allow TLS v1.2, but just drop AES128, then we need to configure the ciphers separately for TLS v1.3 and v1.2 or lower:

```
[system_default_sect]
CipherString = DEFAULT:@SECLEVEL=2:!AES128
CipherSuites = TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256
MinProtocol = TLSv1.2
```

To test, let's force our test `s_server` server to only offer TLSv1.2:

```
$ sudo openssl s_server -cert j-server.pem -key j-server.key -port 443 -www -tls1_
2
```

And our client picks AES256:

```
$ wget https://j-server.lxd/stats -O /dev/stdout -q | grep Cipher -w
New, TLSv1.2, Cipher is ECDHE-RSA-AES256-GCM-SHA384
    Cipher    : ECDHE-RSA-AES256-GCM-SHA384
```

But that could also be just because AES256 is stronger than AES128. Let's not offer AES256 on the server, and also jump ahead and also remove CHACHA20, which would be the next one preferable to AES128:

```
$ sudo openssl s_server -cert j-server.pem -key j-server.key -port 443 -www -tls1_
2 -cipher 'DEFAULT:!AES256:!CHACHA20'
```

Surely `wget` should fail now. Well, turns out it does select AES128:

```
$ wget https://j-server.lxd/stats -O /dev/stdout -q | grep Cipher -w
New, TLSv1.2, Cipher is ECDHE-RSA-AES128-GCM-SHA256
    Cipher    : ECDHE-RSA-AES128-GCM-SHA256
```

It's unclear why. Maybe it's a safeguard, or maybe AES128 is always allowed in TLSv1.2 and we produced an invalid configuration. This case shows how crypto is complex, and also applications can override any such configuration setting that comes from the library. As a counter example, OpenSSL's `s_client` tool follows the library config, and fails in this case:

```
$ echo | openssl s_client -connect j-server.lxd:443  | grep -w -i cipher
4007F4F9D47F0000:error:0A000410:SSL routines:ssl3_read_bytes:sslv3 alert handshake
failure:../ssl/record/rec_layer_s3.c:1584:SSL alert number 40
New, (NONE), Cipher is (NONE)
```

But we can override that as well with a command-line option and force `s_client` to allow AES128:

```
$ echo | openssl s_client -connect j-server.lxd:443 --cipher DEFAULT:AES128 2>&1|
grep -w -i cipher
New, TLSv1.2, Cipher is ECDHE-RSA-AES128-GCM-SHA256
    Cipher    : ECDHE-RSA-AES128-GCM-SHA256
```

### References

- OpenSSL home page
- SECLEVEL description:
  - https://www.openssl.org/docs/man3.0/man3/SSL_CTX_set_security_level.html
  - https://www.feistyduck.com/library/openssl-cookbook/online/openssl-command-line/understanding-security-levels.html
- Configuration directives that can be used in the `system_default_sect` section are in the `SSL_CONF_cmd(3)` manual page

### GnuTLS

When initialised, the *GnuTLS* library tries to read its system-wide configuration file `/etc/gnutls/config`. This file is present in Ubuntu Noble 24.04 LTS and later, but previous releases did not ship it. In that case, if you want to make configuration changes, it has to be created manually.

This config file can be used to disable (or mark as insecure) algorithms and protocols in a system-wide manner, overriding the library defaults. Note that, intentionally, any algorithms or protocols that were disabled or marked as insecure cannot then be re-enabled or marked as secure.

There are many configuration options available for GnuTLS, and we strongly recommend that you carefully read the upstream documentation listed in the References section at the end of this page if creating this file or making changes to it.

### Structure of the config file

The GnuTLS configuration file is structured as an INI-style text file. There are three sections, and each section contains `key = values` lines. For example:

```
[global]
override-mode = blocklist

[priorities]
SYSTEM = NORMAL:-MD5

[overrides]
tls-disabled-mac = sha1
```

### The `global` section

The `[global]` section sets the override mode used in the `[overrides]` section:

- `override-mode = blocklist`: the algorithms listed in `[overrides]` are disabled
- `override-mode = allowlist`: the algorithms listed in `[overrides]` are enabled.

Note that in the `allowlist` mode, all algorithms that should be enabled must be listed in `[overrides]`, as the library starts with marking all existing algorithms as disabled/insecure. In practice, this means that using `allowlist` tends to make the list in `[overrides]` quite large. Additionally, GnuTLS automatically constructs a `SYSTEM` keyword (that can be used in `[priorities]`) with all the allowed algorithms and ciphers specified in `[overrides]`.

When using `allowlist`, all options in `[overrides]` will be of the `enabled` form. For example:

```
[global]
override-mode = allowlist

[overrides]
secure-hash = sha256
enabled-curve = secp256r1
secure-sig = ecdsa-secp256r1-sha256
enabled-version = tls1.3
tls-enabled-cipher = aes-128-gcm
tls-enabled-mac = aead
tls-enabled-group = secp256r1
```

And when using `blocklist`, all `[override]` options have the opposite meaning (i.e. `disabled`):

```
[global]
override-mode = blocklist

[overrides]
tls-disabled-cipher = aes-128-cbc
```

(continues on next page)

(continued from previous page)

```
tls-disabled-cipher = aes-256-cbc
tls-disabled-mac = sha1
tls-disabled-group = group-ffdhe8192
```

For other examples and a complete list of the valid keys in the `[overrides]` section, please refer to disabling algorithms and protocols.

### Priority strings

The `[priorities]` section is used to construct **priority strings**. These strings are a way to specify the TLS session's handshake algorithms and options in a compact, easy-to-use, format. Note that priority strings are not guaranteed to imply the same set of algorithms and protocols between different GnuTLS versions.

The default priority string is selected at package build time by the vendor, and in the case of Ubuntu Jammy it's defined in `debian/rules` as:

```
NORMAL:-VERS-ALL:+VERS-TLS1.3:+VERS-TLS1.2:+VERS-DTLS1.2:%PROFILE_MEDIUM
```

A priority string can start with a single initial keyword, and then add or remove algorithms or special keywords. The `NORMAL` priority string is defined in this table in the upstream documentation reference, which also includes many other useful keywords that can be used.

To see the resulting list of ciphers and algorithms from a priority string, one can use the `gnutls-cli` command-line tool. For example, to list all the ciphers and algorithms allowed with the priority string `SECURE256`:

```
$ gnutls-cli --list --priority SECURE256
Cipher suites for SECURE256
TLS_AES_256_GCM_SHA384                                0x13, 0x02      TLS1.3
TLS_CHACHA20_POLY1305_SHA256                          0x13, 0x03      TLS1.3
TLS_ECDHE_ECDSA_AES_256_GCM_SHA384                    0xc0, 0x2c      TLS1.2
TLS_ECDHE_ECDSA_CHACHA20_POLY1305                     0xcc, 0xa9      TLS1.2
TLS_ECDHE_ECDSA_AES_256_CCM                           0xc0, 0xad      TLS1.2
TLS_ECDHE_RSA_AES_256_GCM_SHA384                      0xc0, 0x30      TLS1.2
TLS_ECDHE_RSA_CHACHA20_POLY1305                       0xcc, 0xa8      TLS1.2
TLS_RSA_AES_256_GCM_SHA384                            0x00, 0x9d      TLS1.2
TLS_RSA_AES_256_CCM                                   0xc0, 0x9d      TLS1.2
TLS_DHE_RSA_AES_256_GCM_SHA384                        0x00, 0x9f      TLS1.2
TLS_DHE_RSA_CHACHA20_POLY1305                         0xcc, 0xaa      TLS1.2
TLS_DHE_RSA_AES_256_CCM                               0xc0, 0x9f      TLS1.2

Protocols: VERS-TLS1.3, VERS-TLS1.2, VERS-TLS1.1, VERS-TLS1.0, VERS-DTLS1.2, VERS-
DTLS1.0
Ciphers: AES-256-GCM, CHACHA20-POLY1305, AES-256-CBC, AES-256-CCM
MACs: AEAD
Key Exchange Algorithms: ECDHE-ECDSA, ECDHE-RSA, RSA, DHE-RSA
Groups: GROUP-SECP384R1, GROUP-SECP521R1, GROUP-FFDHE8192
PK-signatures: SIGN-RSA-SHA384, SIGN-RSA-PSS-SHA384, SIGN-RSA-PSS-RSAE-SHA384,
SIGN-ECDSA-SHA384, SIGN-ECDSA-SECP384R1-SHA384, SIGN-EdDSA-Ed448, SIGN-RSA-SHA512,
```

(continues on next page)

(continued from previous page)

```
SIGN-RSA-PSS-SHA512, SIGN-RSA-PSS-RSAE-SHA512, SIGN-ECDSA-SHA512, SIGN-ECDSA-
SECP521R1-SHA512
```

You can manipulate the resulting set by manipulating the priority string. For example, to remove `CHACHA20-POLY1305` from the `SECURE256` set:

```
$ gnutls-cli --list --priority SECURE256:-CHACHA20-POLY1305
Cipher suites for SECURE256:-CHACHA20-POLY1305
TLS_AES_256_GCM_SHA384                                0x13, 0x02      TLS1.3
TLS_ECDHE_ECDSA_AES_256_GCM_SHA384                    0xc0, 0x2c      TLS1.2
TLS_ECDHE_ECDSA_AES_256_CCM                           0xc0, 0xad      TLS1.2
TLS_ECDHE_RSA_AES_256_GCM_SHA384                      0xc0, 0x30      TLS1.2
TLS_RSA_AES_256_GCM_SHA384                            0x00, 0x9d      TLS1.2
TLS_RSA_AES_256_CCM                                   0xc0, 0x9d      TLS1.2
TLS_DHE_RSA_AES_256_GCM_SHA384                        0x00, 0x9f      TLS1.2
TLS_DHE_RSA_AES_256_CCM                               0xc0, 0x9f      TLS1.2

Protocols: VERS-TLS1.3, VERS-TLS1.2, VERS-TLS1.1, VERS-TLS1.0, VERS-DTLS1.2, VERS-
DTLS1.0
Ciphers: AES-256-GCM, AES-256-CBC, AES-256-CCM
MACs: AEAD
Key Exchange Algorithms: ECDHE-ECDSA, ECDHE-RSA, RSA, DHE-RSA
Groups: GROUP-SECP384R1, GROUP-SECP521R1, GROUP-FFDHE8192
PK-signatures: SIGN-RSA-SHA384, SIGN-RSA-PSS-SHA384, SIGN-RSA-PSS-RSAE-SHA384,
SIGN-ECDSA-SHA384, SIGN-ECDSA-SECP384R1-SHA384, SIGN-EdDSA-Ed448, SIGN-RSA-SHA512,
SIGN-RSA-PSS-SHA512, SIGN-RSA-PSS-RSAE-SHA512, SIGN-ECDSA-SHA512, SIGN-ECDSA-
SECP521R1-SHA512
```

And you can give this a new name by adding the following to the `[priorities]` section:

```
[priorities]
MYSET = SECURE256:-CHACHA20-POLY1305
```

Which allows the `MYSET` priority string to be used like this:

```
$ gnutls-cli --list --priority @MYSET
```

### Verification profile (overrides)

When verifying a certificate, or TLS session parameters, GnuTLS uses a set of profiles associated with the session to determine whether the parameters seen in the session are acceptable. These profiles are normally set using the `%PROFILE` priority string, but it is also possible to set a low bar that applications cannot override. This is done with the `min-verification-profile` setting in the `[overrides]` section.

For example:

```
[overrides]
# do not allow applications use the LOW or VERY-WEAK profiles.
min-verification-profile = legacy
```

The list of values that can be used, and their meaning, is shown in the key sizes and security parameters table in the upstream documentation.

### Practical examples

Let's see some practical examples of how we can use the configuration file to tweak the default cryptographic settings of an application linked with GnuTLS.

Contrary to OpenSSL, GnuTLS does not allow a cipher that was once removed to be allowed again. So if you have a setting in the GnuTLS config file that prohibits `CHACHA20`, an application using GnuTLS will not be able to allow it.

### Only use TLSv1.3

One way to do it is to set a new default priority string that removes all TLS versions and then adds back just TLS 1.3:

```
[global]
override-mode = blocklist

[overrides]
default-priority-string = NORMAL:-VERS-TLS-ALL:+VERS-TLS1.3
```

With our test server providing everything but TLSv1.3:

```
$ sudo openssl s_server -cert j-server.pem -key j-server.key -port 443 -no_tls1_3
-www
```

Connections will fail:

```
$ gnutls-cli j-server.lxd
Processed 125 CA certificate(s).
Resolving 'j-server.lxd:443'...
Connecting to '10.0.100.87:443'...
*** Fatal error: A TLS fatal alert has been received.
*** Received alert [70]: Error in protocol version
```

An application linked with GnuTLS will also fail:

```
$ lftp -c "cat https://j-server.lxd/status"
cat: /status: Fatal error: gnutls_handshake: A TLS fatal alert has been received.
```

But an application can override these settings, because it's only the priority string that is being manipulated in the GnuTLS config:

```
$ lftp -c "set ssl:priority NORMAL:+VERS-TLS-ALL; cat https://j-server.lxd/status"
| grep ^New
New, TLSv1.2, Cipher is ECDHE-RSA-AES256-GCM-SHA384
```

Another way to limit the TLS versions is via specific protocol version configuration keys:

```
[global]
override-mode = blocklist
```

(continues on next page)

(continued from previous page)

```
[overrides]
disabled-version = tls1.1
disabled-version = tls1.2
disabled-version = tls1.0
```

Note that setting the same key multiple times will append the new value to the previous value(s).

In this scenario, the application cannot override the config anymore:

```
$ lftp -c "set ssl:priority NORMAL:+VERS-TLS-ALL; cat https://j-server.lxd/status"
| grep ^New
cat: /status: Fatal error: gnutls_handshake: A TLS fatal alert has been received.
```

### Use only AES256 with TLSv1.3

TLSv1.3 has a small list of ciphers, but it includes *AES*128. Let's remove it:

```
[global]
override-mode = blocklist

[overrides]
disabled-version = tls1.1
disabled-version = tls1.2
disabled-version = tls1.0
tls-disabled-cipher = AES-128-GCM
```

If we now connect to a server that was brought up with this config:

```
$ sudo openssl s_server -cert j-server.pem -key j-server.key -port 443 -
ciphersuites TLS_AES_128_GCM_SHA256 -www
```

Our GnuTLS client will fail:

```
$ gnutls-cli j-server.lxd
Processed 126 CA certificate(s).
Resolving 'j-server.lxd:443'...
Connecting to '10.0.100.87:443'...
*** Fatal error: A TLS fatal alert has been received.
*** Received alert [40]: Handshake failed
```

And given GnuTLS' behaviour regarding re-enabling a cipher that was once removed, we cannot allow AES128 from the command line either:

```
$ gnutls-cli --priority="NORMAL:+AES-128-GCM"  j-server.lxd
Processed 126 CA certificate(s).
Resolving 'j-server.lxd:443'...
Connecting to '10.0.100.87:443'...
*** Fatal error: A TLS fatal alert has been received.
*** Received alert [40]: Handshake failed
```

### References

- System-wide configuration
- Priority strings
- `min-verification-profile` values
- Disabling algorithms and protocols
- Invoking the `gnutls-cli` command line tool

## Network Security Services (NSS)

Network Security Services, or NSS, is a set of libraries that was originally developed by Netscape and later inherited by Mozilla. In Ubuntu, it's used mainly in Mozilla products such as Firefox and Thunderbird, but there are modules and language bindings available for other packages to use.

Given its origins in the Netscape browser, this library used to be bundled together with the applications that required it. Up to this day, for example, the Debian package of Mozilla Thunderbird has its own copy of `libnss3`, ignoring the system-wide one shipped by the `libnss3` Debian package.

### Config file

NSS doesn't have a system-wide policy configuration file in Ubuntu (see #2016303 for details). That leaves the remaining location for the configuration file to be in the NSS "database" directory. Depending on the application, it can be in the following places by default:

- `~/.pki/nssdb/pkcs11.txt` This is where the system-provided `libnss3` library will look by default.
- `~/snap/firefox/common/.mozilla/firefox/<random>.default/pkcs11.txt` This is where the Firefox snap will look.
- `~/.thunderbird/<random>.default-release/pkcs11.txt` Mozilla Thunderbird ships with its own copy of `libnss3`, and is configured to look into this directory to find it.
- `~/.netscape/pkcs11.txt` This is the default used by the NSS tools shipped in the `libnss3-tools` Debian package.

The directory where `pkcs11.txt` is looked up is also the NSS database directory. NSS will store the certificates and private keys it imports or generates here, and the directory will typically contain these SQLITE3 database files:

- `cert9.db`: certificates database
- `key4.db`: private key database

With the `pkcs11.txt` file we can load PKCS#11 modules, including the one built into NSS itself. Other examples of modules that can be loaded from there are modules for smart cards or other hardware-based cryptographic devices. Of interest to us here, though, is the policy module.

### Configuring the NSS policy module

The policy module is defined like this in `pkcs11.txt`:

```
library=
name=Policy
NSS=flags=policyOnly,moduleDB
config="disallow=<list> allow=<list> flags=<flags>"
```

It's via the `config=` line that we can list which cryptographic algorithms we want to allow and disallow. The terms in the list are separated with a colon (":") and consist of the following:

- **The special keyword "ALL"**, meaning all possible values and algorithms. It's mostly used with `disallow`, so that a clean slate can be constructed with a following `allow` list. For example, `disallow=ALL allow=<list of allowed>` would only allow the algorithms explicitly listed in the `allow` list.
- **Algorithm name**: Standard names like `sha256`, `hmac-sha256`, `chacha20-poly1305`, `aes128-gcm` and others.
- **Version specifiers**: A minimum or maximum version for a protocol. These are the available ones:
  - `tls-version-min`, `tls-version-max`: Minimum and maximum version for the TLS protocol. For example, `tls-version-min=tls1.2`.
  - `dtls-version-min`, `dtls-version-max`: As above, but for *DTLS* (TLS over UDP)
- **Key sizes**: Minimum size for a key:
  - `DH-MIN`: Diffie-Helman minimum key size. For example, `DH-MIN=2048` specifies a minimum of 2048 bits.
  - `DSA-MIN`: Digital Signature Algorithm minimum key size. For example, `DSA-MIN=2048` specifies a minimum of 2048 bits.
  - `RSA-MIN`: RSA minimum key size. For example, `RSA-MIN=2048` specifies a minimum of 2048 bits.
- **Signature qualifier**: Selects the specified algorithm with a specific type of signature. For example, `sha256/cert-signature`. Here are some of the qualifiers that are available:
  - `/cert-signature`: Used in certificate signatures, certificate revocation lists (CRLs) and Online Certificate Status Protocol (OCSP).
  - `/signature`: Used in any signature.
  - `/all`: Combines SSL, SSL key exchange, and signatures.
  - `/ssl-key-exchange`: Used in the SSL key exchange.
  - `/ssl`: Used in the SSL record protocol.

The `disallow` rules are always parsed first, and then the `allow` ones, independent of the order in which they appear.

There are extra flags that can be added to the `config` line as well, in a comma-separated list if more than one is specified:

- `policy-lock`: Turn off the ability for applications to change policy with API calls.

- `ssl-lock`: Turn off the ability to change the SSL defaults.

### Practical examples

Let's see some practical examples of how we can use the configuration file to tweak the default cryptographic settings of an application linked with the system NSS libraries.

For these examples, we will be using the configuration file located in `~/.pki/nssdb/pkcs11.txt`. As noted before, depending on the application this file can be in another directory.

The examples will use the `tstclnt` test application that is part of the `libnss3-tools` Debian package. For the server part, we will be using the OpenSSL test server on the same system. Since it uses the OpenSSL library, it won't be affected by the changes we make to the NSS configuration.

### Bootstrapping the NSS database

Install the `libnss3-tools` package which has the necessary tools we will need:

```
sudo apt install libnss3-tools
```

If you don't have a `~/.pki/nssdb` directory yet, it will have to be created first. For that, we will use the `certutil` command, also part of the `libnss3-tools` package. This will bootstrap the NSS database in that directory, and also create the initial `pkcs11.txt` file we will tweak in the subsequent examples:

```
mkdir -p ~/.pki/nssdb
certutil -d ~/.pki/nssdb -N
```

If you already have a populated `~/.pki/nssdb` directory, there is no need to run the above commands.

When running the `certutil` command as shown, you will be asked to choose a password. That password will protect the NSS database, and will be requested whenever certain changes are made to it.

In the following examples we will make changes to the `pkcs11.txt` file inside the NSS database directory. The bootstrap process above will have created this file for us already. The changes that we will make should be *added* to the file, and not replace it. For example, these are the contents of `~/.pki/nssdb/pkcs11.txt` right after the bootstrap process:

```
library=
name=NSS Internal PKCS #11 Module
parameters=configdir='/home/ubuntu/.pki/nssdb' certPrefix='' keyPrefix='' secmod=
'secmod.db' flags= updatedir='' updateCertPrefix='' updateKeyPrefix='' updateid=''
updateTokenDescription=''
NSS=Flags=internal,critical trustOrder=75 cipherOrder=100 slotParams=(1=
{slotFlags=[ECC,RSA,DSA,DH,RC2,RC4,DES,RANDOM,SHA1,MD5,MD2,SSL,TLS,AES,Camellia,
SEED,SHA256,SHA512] askpw=any timeout=30})
```

When an example asks to configure the policy module, its block should be appended to the existing configuration block in the file. For example:

```
library=
name=NSS Internal PKCS #11 Module
parameters=configdir='/home/ubuntu/.pki/nssdb' certPrefix='' keyPrefix='' secmod=
'secmod.db' flags= updatedir='' updateCertPrefix='' updateKeyPrefix='' updateid=''
updateTokenDescription=''
NSS=Flags=internal,critical trustOrder=75 cipherOrder=100 slotParams=(1=
{slotFlags=[ECC,RSA,DSA,DH,RC2,RC4,DES,RANDOM,SHA1,MD5,MD2,SSL,TLS,AES,Camellia,
SEED,SHA256,SHA512] askpw=any timeout=30})

library=
name=Policy
NSS=flags=policyOnly,moduleDB
config="allow=tls-version-min=tls1.3"
```

### Test setup

For these examples, we will be using a simple OpenSSL server on the same system as the NSS client we are testing. For that we will have to generate a certificate and key for the OpenSSL server to use, and then import that into the NSS database so it can be trusted.

First, generate a keypair for OpenSSL:

```
openssl req -new -x509 -days 30 -nodes -subj "/CN=localhost" -out localhost.pem -
keyout localhost.key
```

To avoid telling `tstclnt` to ignore certification validation errors, which might mask the crypto policy changes we are trying to demonstrate, it's best to import this certificate into the NSS database and mark it as trusted:

```
certutil -d ~/.pki/nssdb -A -a -i localhost.pem -t TCP -n localhost
```

This command will ask you for the NSS database password that you supplied when bootstrapping it. The command line options that were used have the following meanings:

- `-d ~/.pki/nssdb`: The path to the NSS database.
- `-A`: Import a certificate.
- `-a`: The certificate is in ASCII mode (PEM).
- `-i localhost.pem`: The file to read (the actual certificate).
- `-t TCP`: Trust flags (see the `-t trustargs` argument in the `certutil manpage` for a full list).
  - `T`: Trusted CA for client authentication.
  - `C`: Trusted CA.
  - `P`: Trusted peer.
- `-n localhost`: A nickname for this certificate, like a label. It can be used later on to select this certificate.

We are now ready to begin our tests. Unless otherwise noted, this is how it's expected that the server will be run:

```
openssl s_server -accept 4443 -cert localhost.pem -key localhost.key -www
```

### The `tstclnt` tool

The `libnss3-tools` package also contains the `tstclnt` tool, which is what we will use in the following examples to test our NSS configuration changes.

This is the typical command we will use:

```
tstclnt -d ~/.pki/nssdb -h localhost -p 4443
```

Where the options have the following meanings:

- `-d ~/.pki/nssdb`: Use the NSS database located in the `~/.pki/nssdb` directory.
- `-h localhost`: The server to connect to.
- `-p 4443`: The TCP port to connect to.

To make things a bit easier to see, since this tool prints a lot of information about the connection, we will wrap it like this:

```
echo "GET / HTTP/1.0" | tstclnt -d ~/.pki/nssdb -h localhost -p 4443 2>&1 | grep ^
New

New, TLSv1.3, Cipher is TLS_AES_128_GCM_SHA256
^C
```

The above tells us that the connection was completed and that it is using `TLSv1.3`, with a `TLS_AES_128_GCM_SHA256` cipher suite.

It will not exit on its own, so it's necessary to press Ctrl+C (`^C`) to get back to the shell prompt.

### Only use TLSv1.3

Here is how we can restrict the TLS protocol version to 1.3 at a minimum:

```
library=
name=Policy
NSS=flags=policyOnly,moduleDB
config="allow=tls-version-min=tls1.3"
```

If we then start the OpenSSL server without TLSv1.3 support, like this (note the extra `no_tls1_3` at the end):

```
openssl s_server -accept 4443 -cert localhost.pem -key localhost.key -www -no_
tls1_3
```

The `tstclnt` tool will fail to connect:

```
echo "GET / HTTP/1.0" | tstclnt -d ~/.pki/nssdb -h localhost -p 4443 2>&1 | grep ^
New
echo $?

1
```

To see the actual error, we can remove the `grep` at the end:

```
echo "GET / HTTP/1.0" | tstclnt -d ~/.pki/nssdb -h localhost -p 4443 2>&1

tstclnt: write to SSL socket failed: SSL_ERROR_PROTOCOL_VERSION_ALERT: Peer
reports incompatible or unsupported protocol version.
```

If we allow the server to offer TLSv1.3:

```
openssl s_server -accept 4443 -cert localhost.pem -key localhost.key -www
```

Then the connection completes:

```
echo "GET / HTTP/1.0" | tstclnt -d ~/.pki/nssdb -h localhost -p 4443 2>&1 | grep ^
New

New, TLSv1.3, Cipher is TLS_AES_128_GCM_SHA256
^C
```

### Use only AES256 with TLSv1.3

In the previous example, the connection ended up using TLSv1.3 as expected, but *AES*128. To enforce AES256, we can disallow the 128-bit version:

```
library=
name=Policy
NSS=flags=policyOnly,moduleDB
config="disallow=aes128-gcm allow=tls-version-min=tls1.3"
```

This time the client selects something else:

```
echo "GET / HTTP/1.0" | tstclnt -d ~/.pki/nssdb -h localhost -p 4443  2>&1 | grep
^New

New, TLSv1.3, Cipher is TLS_CHACHA20_POLY1305_SHA256
```

We can remove that one from the list as well:

```
config="disallow=aes128-gcm:chacha20-poly1305 allow=tls-version-min=tls1.3"
```

And now we get AES256:

```
echo "GET / HTTP/1.0" | tstclnt -d ~/.pki/nssdb -h localhost -p 4443  2>&1 | grep
^New

New, TLSv1.3, Cipher is TLS_AES_256_GCM_SHA384
```

### References

Unfortunately most of the upstream Mozilla documentation is either outdated or deprecated, and the best reference available about the policy module at the moment is in the source code and tests.

- In the source code
- In the tests (policy)
- In the tests (SSL policy)

### Java cryptography configuration

The Java cryptographic settings are large and complex, with many layers and policies. Here we will focus on one aspect of it, which is how to apply some basic filters to the set of cryptographic algorithms available to applications. The references section at the end contains links to more information.

There are many versions of Java available in Ubuntu. It's best to install the "default" one, which is represented by the `default-jre` (for the Runtime Environment) or `default-jdk` (for the Development Kit). And their non-*GUI* counterparts `default-jre-headless` and `default-jdk-headless`, respectively.

To install the default Java Runtime on Ubuntu Server, run the following command:

```
sudo apt install default-jre-headless
```

### Config file

The Java installation in Ubuntu ships a system-wide configuration tree under `/etc/java-<VERSION>-openjdk`. In Ubuntu Jammy 22.04 LTS, the default Java version is 11, so this directory will be `/etc/java-11/openjdk`. In that directory, the file that defines Java security settings, including cryptographic algorithms, is `/etc/java-11-openjdk/security/java.security`.

This is a very large file, with many options and comments. Its structure is simple, with configuration keys and their values. For crypto algorithms, we will be looking into the following settings:

- `jdk.certpah.disabledAlgorithms`: Restrictions on algorithms and key lengths used in certificate path processing.
- `jdk.tls.disabledAlgorithms`: Restrictions on algorithms and key lengths used in SSL/TLS connections.

The list of restrictions has its own format which allows for constructs that disable whole families of algorithms, key sizes, usage, and more. The `java.security` configuration file has comments explaining this syntax with some examples.

Changes to these security settings can be made directly in the `/etc/java-11-openjdk/security/java.security` file, or in an alternate file that can be specified to a Java application by setting the `java.security.properties` value. For example, if your java application is called `myapp.java`, you can invoke it as shown below to specify an additional security properties file:

```
java -Djava.security.properties=file://$HOME/java.security
```

When using just one equals sign ("=") as above, the settings from the specified file are appended to the existing ones. If, however, we use two equals signs:

```
java -Djava.security.properties==file://$HOME/java.security
```

Then the settings from `$HOME/java.security` completely override the ones from the main file at `/etc/java-11-openjdk/security/java.security`.

To disable the ability to specify an additional properties file in the command line, set the key `security.overridePropertiesFile` to `false` in `/etc/java-11-openjdk/security/java.security`.

### Practical examples

Let's see some practical examples of how we can use the configuration file to tweak the default cryptographic settings of a Java application.

The examples will use the Java `keytool` utility for the client part, and a simple OpenSSL test server on localhost for the server part. Since OpenSSL has its own separate configuration, it won't be affected by the changes we make to the Java security settings.

### Test setup

To use the test OpenSSL server, we will have to generate a certificate and key for it to use, and then import that into the Java Certificate Authority (CA) database so it can be trusted.

First, generate a keypair for OpenSSL:

```
openssl req -new -x509 -days 30 -nodes -subj "/CN=localhost" -out localhost.pem -
keyout localhost.key
```

Now let's import this new certificate into the system-wide CA database. Execute the following commands:

```
sudo cp localhost.pem /usr/local/share/ca-certificates/localhost-test.crt
sudo update-ca-certificates
```

For our testing purposes, this is how we will launch our OpenSSL test server:

```
openssl s_server -accept 4443 -cert localhost.pem -key localhost.key | grep ^
CIPHER
```

This will show the cipher that was selected for each connection, as it occurs.

The client part of our setup will be using the `keytool` utility that comes with Java, but any Java application that is capable of using SSL/TLS should suffice. We will be running the client as below:

```
keytool -J-Djava.security.properties=file://$HOME/java.security -printcert -
sslserver localhost:4443 > /dev/null;echo $?
```

These are the parameters:

- `-J-Djava.security.properties=...` This is used to point at the configuration file snippet that has our changes. It is NOT NEEDED if you are modifying `/etc/java-11-openjdk/security/java.security` instead.
- `-printcert -sslserver localhost:4443` Connect to a server on localhost (`-sslserver` is a parameter to `-printcert`, so we need the latter even though we are not interested in the certificate).

The rest is just to ignore all non-error output, and show us the exit status (`0` for success, anything else for an error).

> **Note**
>
> `keytool` is not really intended as a tool to test SSL/TLS connections, but being part of the Java packaging makes it convenient and it's enough for our purposes.

Let's see some examples now.

### Only use TLSv1.3

Create `$HOME/java.security` with the following content:

```
jdk.tls.disabledAlgorithms=TLSv1, TLSv1.1, TLSv1.2, SSLv3, SSLv2
```

Notice that TLSv1.3 is absent.

When you then run the `keytool` utility:

```
$ keytool -J-Djava.security.properties=file://$HOME/java.security -printcert -
sslserver localhost:4443 > /dev/null;echo $?

0
```

The server should log:

```
$ openssl s_server -accept 4443 -key localhost.key -cert localhost.pem   | grep ^
CIPHER

CIPHER is TLS_AES_256_GCM_SHA384
```

That is a TLSv1.3 cipher. To really test that TLSv1.3 is the only protocol available, we can force some failures:

Force the client to try to use TLSv1.2:

```
$ keytool \
  -J-Djava.security.properties=file://$HOME/java.security \
  -J-Djdk.tls.client.protocols=TLSv1.2 \
  -printcert -sslserver localhost:4443

keytool error: java.lang.Exception: No certificate from the SSL server
```

Restart the server with the `no_tls1_3` option, disabling TLSv1.3, and run the client again as originally (without the extra TLSv1.2 option we added above):

**Server**:

```
$ openssl s_server -accept 4443 -key localhost.key -cert localhost.pem -no_tls1_3

Using default temp DH parameters
ACCEPT
```

(continues on next page)

(continued from previous page)

```
ERROR
40676E75B37F0000:error:0A000102:SSL routines:tls_early_post_process_client_
hello:unsupported protocol:../ssl/statem/statem_srvr.c:1657:
shutting down SSL
CONNECTION CLOSED
```

**Client**:

```
$ keytool -J-Djava.security.properties=file://$HOME/java.security -printcert -
sslserver localhost:4443

keytool error: java.lang.Exception: No certificate from the SSL server
```

To get a little bit more verbosity in the `keytool` output, you can add the `-v` option. Then, inside the traceback that we get back, we can see an error message about an SSL protocol version:

```
$ keytool -J-Djava.security.properties=file://$HOME/java.security -printcert -
sslserver localhost:4443  -v

keytool error: java.lang.Exception: No certificate from the SSL server
java.lang.Exception: No certificate from the SSL server
        at java.base/sun.security.tools.keytool.Main.doPrintCert(Main.java:2981)
        at java.base/sun.security.tools.keytool.Main.doCommands(Main.java:1292)
        at java.base/sun.security.tools.keytool.Main.run(Main.java:421)
        at java.base/sun.security.tools.keytool.Main.main(Main.java:414)
Caused by: javax.net.ssl.SSLHandshakeException: Received fatal alert: protocol_
version
...
```

### Prevent a specific cipher

The Java Security Standard Algorithm Names page lists the names of all the cryptographic algorithms recognised by Java. If you want to prevent a specific algorithm from being used, you can list it in the `java.security` file.

In the previous example where we allowed only TLSv1.3 we saw that the negotiated algorithm was `TLS_AES_256_GCM_SHA384`. But what happens if we block it?

Add `TLS_AES_256_GCM_SHA384` to `jdk.tls.disabledAlgorithms` in `$HOME/java.security` like this:

```
jdk.tls.disabledAlgorithms=TLSv1, TLSv1.1, TLSv1.2, SSLv3, SSLv2, TLS_AES_256_GCM_
SHA384
```

If we run our client now:

```
$ keytool -J-Djava.security.properties=file://$HOME/java.security -printcert -
sslserver localhost:4443 > /dev/null; echo $?

0
```

The server will show the new selected cipher:

```
$ openssl s_server -accept 4443 -key localhost.key -cert localhost.pem | grep ^
CIPHER

CIPHER is TLS_AES_128_GCM_SHA256
```

### Blocking cipher "elements"

With TLSv1.3 ciphers, we must list the exact cipher name. With TLSv1.2 ciphers, however, there is a bit more flexibility and we can list just an "element".

For example, let's check out a case where we only allow TLSv1.2 for simplicity by once again modifying `$HOME/java.security`:

```
jdk.tls.disabledAlgorithms=TLSv1, TLSv1.1, TLSv1.3, SSLv2, SSLv3
```

When we run the client:

```
$ keytool -J-Djava.security.properties=file://$HOME/java.security -printcert -
sslserver localhost:4443  > /dev/null; echo $?

0
```

The server reports:

```
$ openssl s_server -accept 4443 -key localhost.key -cert localhost.pem | grep ^
CIPHER

CIPHER is ECDHE-RSA-AES256-GCM-SHA384
```

We can block just the AES256 component by using:

```
jdk.tls.disabledAlgorithms=TLSv1, TLSv1.1, TLSv1.3, SSLv2, SSLv3, AES_256_GCM
```

And now the server reports:

```
$ openssl s_server -accept 4443 -key localhost.key -cert localhost.pem | grep ^
CIPHER

CIPHER is ECDHE-RSA-CHACHA20-POLY1305
```

### References

- Additional information on Java's Cryptographic Algorithms settings
- Java Security Standard Algorithm Names
- Keytool upstream documentation
- `java.security` file with comments – links to the section which explains the crypto algorithm restrictions)

## BIND 9 DNSSEC cryptography selection

Domain Name System Security Extensions (DNSSEC), which provides a set of security features to *DNS*, is a broad topic. In this article, we will briefly show DNSSEC validation happening on a `bind9` DNS server, and then introduce the topic of how we can disable certain cryptographic algorithms from being used in this validation.

### DNSSEC validation

Out of the box, the BIND 9 DNS server is configured to try to use DNSSEC whenever it's available, doing all the validation checks automatically. This is done via the `dnssec-validation auto` setting in `/etc/bind/named.conf.options`, which became the implicit default as of version `1:9:18.34-1` in Ubuntu 24.10 and above.

DNSSEC can be quickly checked with the help of `dig`. Right after you installed `bind9`, you can run `dig` and ask it about the `isc.org` domain:

```
$ dig @127.0.0.1 isc.org +dnssec +multiline

; <<>> DiG 9.18.12-0ubuntu0.22.04.1-Ubuntu <<>> @127.0.0.1 isc.org +dnssec
+multiline
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57669
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 1232
; COOKIE: 71aa6b4e4ca6bb4b01000000643fee81edf0840b48d28d44 (good)
;; QUESTION SECTION:
;isc.org.                IN A

;; ANSWER SECTION:
isc.org.                300 IN A 149.20.1.66
isc.org.                300 IN RRSIG A 13 2 300 (
                                20230510161711 20230410161439 27566 isc.org.
                                EUA5QPEjtVC0scPsvf1c/EIBKHRpS8ektiWiOqk6nb3t
                                JhJAt9uCr3e0KNAcc3WDU+wJzEvqDyJrlZoELqT/pQ== )

;; Query time: 740 msec
;; SERVER: 127.0.0.1#53(127.0.0.1) (UDP)
;; WHEN: Wed Apr 19 13:37:05 UTC 2023
;; MSG SIZE  rcvd: 183
```

We can see that a `RRSIG` DNSSEC record was returned, but the important information in this output is the `ad` flag near the top. That stands for "authenticated data", and means that the DNSSEC records in the response were validated.

To see an example where this verification fails, we can use the `www.dnssec-failed.org` domain, which is specially crafted for this:

```
$ dig @127.0.0.1 www.dnssec-failed.org +dnssec +multiline

; <<>> DiG 9.18.12-0ubuntu0.22.04.1-Ubuntu <<>> @127.0.0.1 www.dnssec-failed.org
+dnssec +multiline
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 56056
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 1232
; COOKIE: 541f6c66a216acdb01000000643fef9ebb21307fee2ea0e3 (good)
;; QUESTION SECTION:
;www.dnssec-failed.org.         IN A

;; Query time: 1156 msec
;; SERVER: 127.0.0.1#53(127.0.0.1) (UDP)
;; WHEN: Wed Apr 19 13:41:50 UTC 2023
;; MSG SIZE  rcvd: 78
```

Here we see that:

- There is no `IN A` IP address shown in the reply
- The status is `SERVFAIL`
- There is no `ad` flag

In the `bind9` logs, we will see DNSSEC validation errors:

```
$ journalctl -u named.service -n 10

Apr 19 13:41:50 j named[3018]: validating dnssec-failed.org/DNSKEY: no valid
signature found (DS)
Apr 19 13:41:50 j named[3018]: no valid RRSIG resolving 'dnssec-failed.org/DNSKEY/
IN': 68.87.76.228#53
Apr 19 13:41:50 j named[3018]: broken trust chain resolving 'www.dnssec-failed.
org/A/IN': 68.87.85.132#53
(...)
```

We can run `dig` with the `+cd` command line parameter which disables this verification, but notice that still we don't get the `ad` flag in the reply:

```
$ dig @127.0.0.1 www.dnssec-failed.org +dnssec +multiline +cd

; <<>> DiG 9.18.12-0ubuntu0.22.04.1-Ubuntu <<>> @127.0.0.1 www.dnssec-failed.org
+dnssec +multiline +cd
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 42703
;; flags: qr rd ra cd; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1
```

(continues on next page)

(continued from previous page)

```
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 1232
; COOKIE: 3d6a4f4ff0014bdc01000000643ff01c3229ed7d798c5f8d (good)
;; QUESTION SECTION:
;www.dnssec-failed.org.        IN A

;; ANSWER SECTION:
www.dnssec-failed.org.        7031 IN        A 68.87.109.242
www.dnssec-failed.org.        7031 IN        A 69.252.193.191
www.dnssec-failed.org.        7074 IN        RRSIG A 5 3 7200 (
                                20230505145108 20230418144608 44973 dnssec-failed.
org.
                                R6/u+5Gv3rH93gO8uNvz3sb9ErQNuvFKu6W5rtUleXF/
                                vkqJXbNe8grMuiV6Y+CNEP6jRBu0jOBPncb5cXbfcmfo
                                CoVOjpsLySxt4D1EUl4yByWm2ZAdXRrk6A8SaldIdDv8
                                9t+FguTdQrZv9Si+afKrLyC7L/mltXMllq3stDI= )

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1) (UDP)
;; WHEN: Wed Apr 19 13:43:56 UTC 2023
;; MSG SIZE  rcvd: 287
```

### Restricting DNSSEC algorithms

It's possible to limit the cryptographic algorithms used by BIND to validate DNSSEC records. This is done via two configuration settings, located inside the `options { }` block of `/etc/named/named.conf.options`:

- `disable-algorithms "<domain>" { a; b; ... };` Disables the listed algorithms for the specified domain and all subdomains of it.
- `disable-ds-digests "<domain>" { a; b; ... };` Disables the listed digital signature digests for the specified domain and all subdomains of it.

For example, the following disables `RSAMD5`, `DSA` and `GOST` for all zones:

```
disable-algorithms "." {
    RSAMD5;
    DSA;
};
disable-ds-digest "." {
    GOST;
};
```

The list of algorithm names can be obtained at DNSSEC Algorithm Numbers, in the **Mnemonic** column of the **Available Formats** table. The algorithm number is also standardised, and is part of the DNSSEC records.

For example, if we go back to the `dig` result from before where we inspected the `isc.org` domain, the `RRSIG` record had this (showing just the first line for brevity):

```
isc.org.        300 IN RRSIG A 13 2 300 (
```

In that record, the number `13` is the algorithm number, and in this case it means the algorithm `ECDSAP256SHA256` was used.

Just to see how BIND would react to an algorithm being disabled, let's temporarily add `ECDSAP256SHA256` to the list of disabled algorithms:

```
disable-algorithms "." {
   RSAMD5;
   DSA;
   ECDSAP256SHA256;
};
```

And restart BIND:

```
sudo systemctl restart bind9.service
```

Now the `ad` flag is gone, meaning that this answer wasn't validated:

```
$ dig @127.0.0.1 isc.org +dnssec +multiline

; <<>> DiG 9.18.1-1ubuntu1-Ubuntu <<>> @127.0.0.1 isc.org +dnssec +multiline
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 43893
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 1232
; COOKIE: 6527ce585598025d01000000643ff8fa02418ce38af13fa7 (good)
;; QUESTION SECTION:
;isc.org.               IN A

;; ANSWER SECTION:
isc.org.                300 IN A 149.20.1.66
isc.org.                300 IN RRSIG A 13 2 300 (
                                20230510161711 20230410161439 27566 isc.org.
                                EUA5QPEjtVC0scPsvf1c/EIBKHRpS8ektiWiOqk6nb3t
                                JhJAt9uCr3e0KNAcc3WDU+wJzEvqDyJrlZoELqT/pQ== )

;; Query time: 292 msec
;; SERVER: 127.0.0.1#53(127.0.0.1) (UDP)
;; WHEN: Wed Apr 19 14:21:46 UTC 2023
;; MSG SIZE  rcvd: 183
```

The logs only say there was no valid signature found:

```
Apr 19 14:23:01 j-bind9 named[2786]: validating isc.org/A: no valid signature
found
```

Note this is different from rejecting the response: it just means that this response is being treated as if it didn't have any DNSSEC components in it, or in other words, it's treated as "insecure".

In general, as always with cryptography, be careful with which algorithms you decide to disable and remove from DNSSEC validation, as such errors can be hard to diagnose. To help with troubleshooting, the Internet Systems Consortium (ISC) has published a very extensive DNSSEC guide, which contains a detailed troubleshooting section (see below).

**Note**

Remember now to remove the disabling of `ECDSAP256SHA256` from `/etc/bind/named.conf.options` and restart BIND 9. This change was just a quick test!

### References

- ISC's DNSSEC Guide
- DNSSEC troubleshooting section of the ISC DNSSEC guide
- Standard algorithms used for DNSSEC

### OpenSSH crypto configuration

Establishing an SSH connection to a remote service involves multiple stages. Each one of these stages will use some form of encryption, and there are configuration settings that control which cryptographic algorithms can be used at each step.

The default selection of algorithms for each stage should be good enough for the majority of deployment scenarios. Sometimes, however, a compliance rule, or a set of legacy servers, or something else, requires a change in this selection. Perhaps a legacy system or piece of hardware that is still in production is not compatible with the current encryption schemes and requires legacy algorithms to be enabled again. Or a compliance rule that isn't up-to-date with the current crypto standards doesn't allow a more advanced cipher.

**Warning**

Be careful when restricting cryptographic algorithms in SSH, specially on the server side. You can inadvertently lock yourself out of a remote system!

### Algorithm configuration general rules

Most of the configuration options that take a list of cryptographic algorithms follow a defined set of rules. The first algorithm in the list (that the **client** offers to the server) that matches an offer from the server, is what will be selected. The rules are as follows:

- The lists are algorithm names separated by commas. For example, `Ciphers aes128-gcm@openssh.com,aes256-gcm@openssh.com` will replace the current set of ciphers with the two named algorithms.
- Instead of specifying the full list, which will replace the existing default one, some manipulations are allowed. If the list starts with:

- `+` The specified algorithm(s) will be appended to the end of the default set. For example, `MACs +hmac-sha2-512,hmac-sha2-256` will append *both* Message Authentication Code (MAC) algorithms to the end of the current set.
- `-` The specified algorithm(s) will be removed from the default set. For example, `KexAlgorithms -diffie-hellman-group1-sha1,diffie-hellman-group14-sha1` will remove *both* key exchange algorithms from the current set.
- `^` The specified ciphers will be placed at the beginning of the default set. For example, `PubkeyAcceptedAlgorithms ^ssh-ed25519,ecdsa-sha2-nistp256` will move *both* signature algorithms to the start of the set.
- Wildcards (`*`) are also allowed, but be careful to not inadvertently include or exclude something that wasn't intended.

With rare exceptions, the list of algorithms can be queried by running `ssh -Q <config>`, where `<config>` is the configuration setting name. For example, `ssh -Q ciphers` will show the available list of ciphers.

**Note**

The output of the `ssh -Q <name>` command will not take into consideration the configuration changes that may have been made. It cannot therefore be used to test the crypto configuration changes.

### Configuration settings

It's not the goal of this documentation to repeat the excellent upstream documentation (see the References section at the end of this page). Instead, we will show the configuration options, and some examples of how to use them.

Here are the configuration settings that control the cryptographic algorithms selection. Unless otherwise noted, they apply to both the server and the client.

- `Ciphers` List of symmetric ciphers. Examples include `aes256-ctr` and `chacha20-poly1305@openssh.com`.
- `MACs` List of Message Authentication Code algorithms, used for data integrity protection. The `-etm` versions calculate the MAC after encryption and are considered safer. Examples include `hmac-sha2-256` and `hmac-sha2-512-etm@openssh.com`.
- `GSSAPIKexAlgorithms` This option is not available in OpenSSH upstream, and is provided via a patch that Ubuntu and many other Linux Distributions carry. It lists the key exchange (kex) algorithms that are offered for *Generic Security Services Application Program Interface (GSSAPI)* key exchange, and only applies to connections using GSSAPI. Examples include `gss-gex-sha1-` and `gss-group14-sha256-`.
- `KexAlgorithms` List of available key exchange (kex) algorithms. Examples include `curve25519-sha256` and `sntrup761x25519-sha512@openssh.com`.
- `HostKeyAlgorithms` This is a **server-only** configuration option. It lists the available host key signature algorithms that the server offers. Examples include `ssh-ed25519-cert-v01@openssh.com` and `ecdsa-sha2-nistp521-cert-v01@openssh.com`.
- `PubkeyAcceptedAlgorithms` List of signature algorithms that will be accepted for public key authentication. Examples include `ssh-ed25519-cert-v01@openssh.com` and

`rsa-sha2-512-cert-v01@openssh.com`.

- `CASignatureAlgorithms` List of algorithms that certificate authorities (CAs) are allowed to use to sign certificates. Certificates signed using any other algorithm will not be accepted for public key or host-based authentication. Examples include `ssh-ed25519` and `ecdsa-sha2-nistp384`.

To check what effect a configuration change has on the server, it's helpful to use the `-T` parameter and `grep` the output for the configuration key you want to inspect. For example, to check the current value of the `Ciphers` configuration setting after having set `Ciphers ^3des-cbc` in `sshd_config`:

```
$ sudo sshd -T | grep ciphers

ciphers 3des-cbc,chacha20-poly1305@openssh.com,aes128-ctr,aes192-ctr,aes256-ctr,
aes128-gcm@openssh.com,aes256-gcm@openssh.com
```

The output will include changes made to the configuration key. There is no need to restart the service.

### OpenSSH examples

Here are some examples of how the cryptographic algorithms can be selected in OpenSSH.

### Which cipher was used?

One way to examine which algorithm was selected is to add the `-v` parameter to the `ssh` client.

For example, assuming password-less public key authentication is being used (so no password prompt), we can use this command to initiate the connection and exit right away:

```
$ ssh -v <server> exit 2>&1 | grep "cipher:"

debug1: kex: server->client cipher: chacha20-poly1305@openssh.com MAC: <implicit>
compression: none
debug1: kex: client->server cipher: chacha20-poly1305@openssh.com MAC: <implicit>
compression: none
```

In the above case, the `chacha20` cipher was automatically selected. We can influence this decision and only offer one algorithm:

```
$ ssh -v -c aes128-ctr <server> exit 2>&1 | grep "cipher:"

debug1: kex: server->client cipher: aes128-ctr MAC: umac-64-etm@openssh.com
compression: none
debug1: kex: client->server cipher: aes128-ctr MAC: umac-64-etm@openssh.com
compression: none
```

For the other stages in the `ssh` connection, like key exchange, or public key authentication, other expressions for the `grep` command have to be used. In general, it will all be visible in the full `-v` output.

### Remove AES 128 from server

Let's configure an OpenSSH server to only offer the *AES* 256-bit variant of symmetric ciphers for an ssh connection.

First, let's see what the default is:

```
$ sudo sshd -T | grep ciphers

ciphers chacha20-poly1305@openssh.com,aes128-ctr,aes192-ctr,aes256-ctr,aes128-
gcm@openssh.com,aes256-gcm@openssh.com
```

Now let's make our change. On the server, we can edit `/etc/ssh/sshd_config` and add this line:

```
Ciphers -aes128*
```

And then check what is left:

```
$ sudo sshd -T | grep ciphers

ciphers chacha20-poly1305@openssh.com,aes192-ctr,aes256-ctr,aes256-gcm@openssh.com
```

To activate the change, ssh has to be restarted:

```
$ sudo systemctl restart ssh.service
```

After we restart the service, clients will no longer be able to use AES 128 to connect to it:

```
$ ssh -c aes128-ctr <server>

Unable to negotiate with 10.0.102.49 port 22: no matching cipher found. Their
offer: chacha20-poly1305@openssh.com,aes192-ctr,aes256-ctr,aes256-gcm@openssh.com
```

### Prioritise AES 256 on the client

If we just want to prioritise a particular cipher, we can use the "^" character to move it to the front of the list, without disabling any other cipher:

```
$ ssh -c ^aes256-ctr -v <server> exit 2>&1 | grep "cipher:"

debug1: kex: server->client cipher: aes256-ctr MAC: umac-64-etm@openssh.com
compression: none
debug1: kex: client->server cipher: aes256-ctr MAC: umac-64-etm@openssh.com
compression: none
```

In this way, if the server we are connecting to does not support AES 256, the negotiation will pick up the next one from the list. If we do that on the server via `Ciphers -aes256*`, this is what the same client, with the same command line, now reports:

```
$ ssh -c ^aes256-ctr -v <server> exit 2>&1 | grep "cipher:"

debug1: kex: server->client cipher: chacha20-poly1305@openssh.com MAC: <implicit>
```

(continues on next page)

(continued from previous page)

```
compression: none
debug1: kex: client->server cipher: chacha20-poly1305@openssh.com MAC: <implicit>
compression: none
```

### References

- OpenSSH upstream documentation index
- Ubuntu `sshd_config(5)` manual page
- Ubuntu `ssh_config(5)` manual page

### Troubleshooting TLS/SSL

Debugging TLS/SSL connections and protocols can be daunting due to their complexity. Here are some troubleshooting tips.

### Separate the client and server

Whenever testing TLS/SSL connections over the network, it's best to really separate the client and the server. Remember that the crypto library configuration file is read by the library, not just by a server or a client. It's read by both. Therefore having separate systems acting as clients and servers, with their own configuration files, makes things simpler to analyse.

### Tools

Here are some tools to help troubleshooting a TLS/SSL configuration.

### OpenSSL server and client apps

The OpenSSL server and client tools are very handy to quickly bring up a server with a selection of ciphers and protocols and test it with a client. Being part of OpenSSL, these tools will also initialise the library defaults directly from the OpenSSL config file, so they are very useful to test your configuration changes.

To bring up an OpenSSL server, a certificate with a private key is needed. There are many ways to generate a pair, and here is a quick one:

```
$ openssl req -new -x509 -nodes -days 30 -out myserver.pem -keyout myserver.key
```

Answer the questions as you prefer, but the one that needs special attention is the `commonName` (`CN`) one, which should match the *hostname* of this server. Then bring up the OpenSSL server with this command:

```
$ openssl s_server -cert myserver.pem -key myserver.key
```

That will bring up a TLS/SSL server on port 4433. Extra options that can be useful:

- `-port N`: Set a port number. Remember that ports below 1024 require root privileges, so use `sudo` if that's the case.
- `-www`: Will send back a summary of the connection information, like ciphers used, protocols, etc.

- `-tls1_2`, `-tls1_3`, `-no_tls1_3`, `-no_tls1_2`: Enable only the mentioned protocol version, or, with the `no_` prefix variant, disable it.
- `-cipher <string>`: Use the specified cipher string for TLS1.2 and lower.
- `-ciphersuite <string>`: Use the specified string for TLS1.3 ciphers.

The client connection tool can be used like this when connecting to `server`:

```
$ echo | openssl s_client -connect server:port 2>&1 | grep ^New
```

That will generally show the TLS version used, and the selected cipher:

```
$ echo | openssl s_client -connect j-server.lxd:443 2>&1  | grep ^New
New, TLSv1.3, Cipher is TLS_AES_256_GCM_SHA384
```

The ciphers and protocols can also be selected with the same command line options as the server:

```
$ echo | openssl s_client -connect j-server.lxd:443 -no_tls1_3 2>&1  | grep ^New
New, TLSv1.2, Cipher is ECDHE-RSA-AES256-GCM-SHA384

$ echo | openssl s_client -connect j-server.lxd:443 -no_tls1_3 2>&1 -cipher
DEFAULT:-AES256 | grep ^New
New, TLSv1.2, Cipher is ECDHE-RSA-CHACHA20-POLY1305
```

### The `sslscan` tool

The `sslscan` tool comes from a package with the same name, and it will scan a server and list the supported algorithms and protocols. It's super useful for determining if your configuration has really disabled or enabled a particular cipher or TLS version.

To use the tool, point it at the server you want to scan:

```
$ sslscan j-server.lxd
```

And you will get a report of the ciphers and algorithms supported by that server. Consult the `sslscan(1)` manpage.

### References

- `OpenSSL s_server`
- `OpenSSL s_client`
- `sslscan(1)`
- `https://badssl.com`: excellent website that can be used to test a client against a multitude of certificates, algorithms, key sizes, protocol versions, and more.

### About certificates

One of the most common forms of cryptography today is **public-key** cryptography. Public-key cryptography uses a **public key** and a **private key**. The system works by encrypting information using the public key. The information can then only be decrypted using the private key.

A common use for public-key cryptography is encrypting application traffic using a Secure Socket Layer (SSL) or Transport Layer Security (TLS) connection. One example: configuring Apache to provide HTTPS, the HTTP protocol over SSL/TLS. This allows a way to encrypt traffic using a protocol that does not itself provide encryption.

A **certificate** is a way to distribute a public key and other information about a server and the organisation responsible for it. Certificates can be digitally signed by a **Certification Authority** (CA), which is a trusted third party that has confirmed the information contained in the certificate is accurate.

### Types of certificates

To set up a secure server using public-key cryptography, in most cases, you send your certificate request (including your public key), proof of your company's identity, and payment to a CA. The CA verifies the certificate request and your identity, and then sends back a certificate for your secure server. Alternatively, you can create your own **self-signed** certificate.

> **Note**
>
> Self-signed certificates should not be used in most production environments.

Continuing the HTTPS example, a CA-signed certificate provides two important capabilities that a self-signed certificate does not:

- Browsers will (usually) automatically recognise the CA signature and allow a secure connection to be made without prompting the user.
- When a CA issues a signed certificate, it is guaranteeing the identity of the organisation providing the web pages to the browser.

Most software supporting SSL/TLS has a list of CAs whose certificates they automatically accept. If a browser encounters a certificate whose authorising CA is not in the list, the browser asks the user to either accept or decline the connection. Also, other applications may generate an error message when using a self-signed certificate.

The process of getting a certificate from a CA is fairly straightforward. A quick overview is as follows:

1. Create a private and public encryption key pair.
2. Create a certificate signing request based on the public key. The certificate request contains information about your server and the company hosting it.
3. Send the certificate request, along with documents proving your identity, to a CA. We cannot tell you which certificate authority to choose. Your decision may be based on your past experiences, or on the experiences of your friends or colleagues, or purely on monetary factors.

   Once you have decided upon a CA, you need to follow the instructions they provide on how to obtain a certificate from them.
4. When the CA is satisfied that you are indeed who you claim to be, they send you a digital certificate.
5. Install this certificate on your secure server, and configure the appropriate applications to use the certificate.

### Generate a Certificate Signing Request (CSR)

Whether you are getting a certificate from a CA or generating your own self-signed certificate, the first step is to generate a key.

If the certificate will be used by service daemons, such as Apache, Postfix, Dovecot, etc., a key without a passphrase is often appropriate. Not having a passphrase allows the services to start without manual intervention, usually the preferred way to start a daemon.

This section will cover generating a key both with or without a passphrase. The non-passphrase key will then be used to generate a certificate that can be used with various service daemons.

> **Warning**
>
> Running your secure service without a passphrase is convenient because you will not need to enter the passphrase every time you start your secure service. But it is insecure – a compromise of the key means a compromise of the server as well.

To generate the keys for the Certificate Signing Request (CSR) run the following command from a terminal prompt:

```
openssl genrsa -des3 -out server.key 2048

Generating RSA private key, 2048 bit long modulus
.........................++++++
.......++++++
e is 65537 (0x10001)
Enter pass phrase for server.key:
```

You can now enter your passphrase. For best security, it should contain **at least** eight characters. The minimum length when specifying `-des3` is four characters. As a best practice it should also include numbers and/or punctuation and not be a word in a dictionary. Also remember that your passphrase is case-sensitive.

Re-type the passphrase to verify. Once you have re-typed it correctly, the server key is generated and stored in the `server.key` file.

Now create the insecure key, the one without a passphrase, and shuffle the key names:

```
openssl rsa -in server.key -out server.key.insecure
mv server.key server.key.secure
mv server.key.insecure server.key
```

The insecure key is now named `server.key`, and you can use this file to generate the CSR without a passphrase.

To create the CSR, run the following command at a terminal prompt:

```
openssl req -new -key server.key -out server.csr
```

It will prompt you to enter the passphrase. If you enter the correct passphrase, it will prompt you to enter 'Company Name', 'Site Name', 'Email ID', etc. Once you enter all these details, your CSR will be created and it will be stored in the `server.csr` file.

You can now submit this CSR file to a CA for processing. The CA will use this CSR file and issue the certificate. Alternatively, you can create self-signed certificate using this CSR.

### Creating a self-signed certificate

To create the self-signed certificate, run the following command at a terminal prompt:

```
openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt
```

The above command will prompt you to enter the passphrase. Once you enter the correct passphrase, your certificate will be created and it will be stored in the `server.crt` file.

> **Warning**
>
> If your secure server is to be used in a production environment, you probably need a CA-signed certificate. It is not recommended to use self-signed certificates in production environments.

### Install the certificate

You can install the key file `server.key` and certificate file `server.crt`, or the certificate file issued by your CA, by running following commands at a terminal prompt:

```
sudo cp server.crt /etc/ssl/certs
sudo cp server.key /etc/ssl/private
```

Now configure any applications that have the ability to use public-key cryptography so that they use the certificate and key files. For example, Apache can provide HTTPS, Dovecot can provide IMAPS and POP3S, etc.

### Certification Authority

If the services on your network require more than a few self-signed certificates it may be worth the additional effort to setup your own internal Certification Authority (CA). Using certificates signed by your own CA allows the various services using the certificates to easily trust other services using certificates issued from the same CA.

First, create the directories to hold the CA certificate and related files:

```
sudo mkdir /etc/ssl/CA
sudo mkdir /etc/ssl/newcerts
```

The CA needs a few additional files to operate; one to keep track of the last serial number used by the CA (each certificate must have a unique serial number), and another file to record which certificates have been issued:

```
sudo sh -c "echo '01' > /etc/ssl/CA/serial"
sudo touch /etc/ssl/CA/index.txt
```

The third file is a CA configuration file. Though not strictly necessary, it is convenient when issuing multiple certificates. Edit `/etc/ssl/openssl.cnf`, and in the `[ CA_default ]`, change:

```
dir             = /etc/ssl              # Where everything is kept
database        = $dir/CA/index.txt     # database index file.
certificate     = $dir/certs/cacert.pem # The CA certificate
serial          = $dir/CA/serial        # The current serial number
private_key     = $dir/private/cakey.pem# The private key
```

Next, create the self-signed root certificate:

```
openssl req -new -x509 -extensions v3_ca -keyout cakey.pem -out cacert.pem -days
3650
```

You will then be asked to enter the details about the certificate. Next, install the root certificate and key:

```
sudo mv cakey.pem /etc/ssl/private/
sudo mv cacert.pem /etc/ssl/certs/
```

You are now ready to start signing certificates. The first item needed is a Certificate Signing Request (CSR) – see the "Generating a CSR" section above for details. Once you have a CSR, enter the following to generate a certificate signed by the CA:

```
sudo openssl ca -in server.csr -config /etc/ssl/openssl.cnf
```

After entering the password for the CA key, you will be prompted to sign the certificate, and again to commit the new certificate. You should then see a somewhat large amount of output related to the certificate creation.

There should now be a new file, `/etc/ssl/newcerts/01.pem`, containing the same output. Copy and paste everything beginning with the `-----BEGIN CERTIFICATE-----` line and continuing through to the `----END CERTIFICATE-----` lines to a file named after the *hostname* of the server where the certificate will be installed. For example `mail.example.com.crt`, is a nice descriptive name.

Subsequent certificates will be named `02.pem`, `03.pem`, etc.

**Note**

Replace `mail.example.com.crt` with your own descriptive name.

Finally, copy the new certificate to the host that needs it, and configure the appropriate applications to use it. The default location to install certificates is `/etc/ssl/certs`. This enables multiple services to use the same certificate without overly complicated file permissions.

For applications that can be configured to use a CA certificate, you should also copy the `/etc/ssl/certs/cacert.pem` file to the `/etc/ssl/certs/` directory on each server.

### Further reading

- The Wikipedia HTTPS page has more information regarding HTTPS.
- For more information on OpenSSL see the OpenSSL Home Page.
- Also, O'Reilly's "Network Security with OpenSSL" is a good in-depth reference.

### Virtual Private Network (VPN)

VPNs are commonly used to provide encrypted, secure access to a network.

- *Introduction to WireGuard VPN*, a popular and modern VPN implementation
- *OpenVPN clients* provides a list of client implementations that can be used with a GUI across platforms.

### Introduction to WireGuard VPN

WireGuard is a simple, fast and modern VPN implementation. It is widely deployed and can be used cross-platform.

VPNs have traditionally been hard to understand, configure and deploy. WireGuard removed most of that complexity by focusing on its single task, and leaving out things like key distribution and pushed configurations. You get a network interface which encrypts and verifies the traffic, and the remaining tasks like setting up addresses, routing, etc, are left to the usual system tools like `ip-route(8)` and `ip-address(8)`.

Setting up the cryptographic keys is very much similar to configuring SSH for key based authentication: each side of the connection has its own private and public key, and the peers' public key, and this is enough to start encrypting and verifying the exchanged traffic.

For more details on how WireGuard works, and information on its availability on other platforms, please see the references section.

### WireGuard concepts

It helps to think of WireGuard primarily as a network interface, like any other. It will have the usual attributes, like IP address, CIDR, and there will be some routing associated with it. But it also has WireGuard-specific attributes, which handle the VPN part of things.

All of this can be configured via different tools. WireGuard itself ships its own tools in the user-space package `wireguard-tools`: `wg(8)` and `wg-quick(8)`. But these are not strictly needed: any user space with the right privileges and kernel calls can configure a WireGuard interface. For example, `systemd-networkd` and `network-manager` can do it on their own, without the WireGuard user-space utilities.

Important attributes of a WireGuard interface are:

- **Private key**: together with the corresponding public key, they are used to authenticate and encrypt data. This is generated with the `wg genkey` command.
- **Listen port**: the UDP port that WireGuard will be listening to for incoming traffic.
- List of **peers**, each one with:
  - **Public key**: the public counterpart of the private key. Generated from the private key of that peer, using the `wg pubkey` command.
  - **Endpoint**: where to send the encrypted traffic to. This is optional, but at least one of the corresponding peers must have it to bootstrap the connection.
  - **Allowed IPs**: list of inner tunnel destination networks or addresses for this peer when sending traffic, or, when receiving traffic, which source networks or addresses are allowed to send traffic to us.

> **Note**
>
> Cryptography is not simple. When we say that, for example, a private key is used to decrypt or sign traffic, and a public key is used to encrypt or verify the authenticity of traffic, this is a simplification and is hiding a lot of important details. WireGuard has a detailed explanation of its protocols and cryptography handling on its website.

These parameters can be set with the low-level `wg(8)` tool, directly via the command line or with a configuration file. This tool, however, doesn't handle the non-WireGuard settings of the interface. It won't assign an IP address to it, for example, nor set up routing. For this reason, it's more common to use `wg-quick(8)`.

`wg-quick` will handle the lifecycle of the WireGuard interface. It can bring it up or down, set up routing, execute arbitrary commands before or after the interface is up, and more. It augments the configuration file that `wg` can use, with its own extra settings, which is important to keep in mind when feeding that file to `wg`, as it will contain settings `wg` knows nothing about.

The `wg-quick` configuration file can have an arbitrary name, and can even be placed anywhere on the system, but the best practice is to:

- Place the file in `/etc/wireguard`.
- Name it after the interface it controls.

For example, a file called `/etc/wireguard/wg0.conf` will have the needed configuration settings for a WireGuard network interface called `wg0`. By following this practice, you get the benefit of being able to call `wg-quick` with just the interface name:

```
$ sudo wg-quick up wg0
```

That will bring the `wg0` interface up, give it an IP address, set up routing, and configure the WireGuard-specific parameters for it to work. This interface is usually called `wg0`, but can have any valid network interface name, like `office` (it doesn't need an index number after the name), `home1`, etc. It can help to give it a meaningful name if you plan to connect to multiple peers.

Let's go over an example of such a configuration file:

```
[Interface]
PrivateKey = eJdSgoS7BZ/uWkuSREN+vhCJPPr3M3UlB3v1Su/amWk=
ListenPort = 51000
Address = 10.10.11.10/24

[Peer]
# office
PublicKey = xeWmdxiLjgebpcItF1ouRo0ntrgFekquRJZQO+vsQVs=
Endpoint = wg.example.com:51000 # fake endpoint, just an example
AllowedIPs = 10.10.11.0/24, 10.10.10.0/24
```

In the `[Interface]` section:

- `Address`: this is the IP address, and CIDR, that the WireGuard interface will be set up with.
- `ListenPort`: the UDP port WireGuard will use for traffic (listening and sending).

- `PrivateKey`: the secret key used to decrypt traffic destined for this interface.

The **peers** list, each one in its own `[Peer]` section (example above has just one), comes next:

- `PublicKey`: the key that will be used to encrypt traffic to this peer.
- `Endpoint`: where to send encrypted traffic to.
- `AllowedIPs`: when sending traffic, this is the list of target addresses that identify this peer. When receiving traffic, it's the list of addresses that are allowed to be the source of the traffic.

To generate the keypairs for each peer, the `wg` command is used:

```
$ umask 077
$ wg genkey > wg0.key
$ wg pubkey < wg0.key > wg0.pub
```

And then the contents of `wg0.key` and `wg0.pub` can be used in the configuration file.

This is what it looks like when this interface is brought up by `wg-quick`:

```
$ sudo wg-quick up wg0
[#] ip link add wg0 type wireguard
[#] wg setconf wg0 /dev/fd/63
[#] ip -4 address add 10.10.11.10/24 dev wg0
[#] ip link set mtu 1420 up dev wg0
[#] ip -4 route add 10.10.10.0/24 dev wg0
```

This is what `wg-quick`:

- Created the WireGuard `wg0` interface.
- Configured it with the data from the configuration file.
- Added the IP/CIDR from the `Address` field to the `wg0` interface.
- Calculated a proper MTU (which can be overridden in the config if needed).
- Added a route for `AllowedIPs`.

Note that in this example `AllowedIPs` is a list of two CIDR network blocks, but `wg-quick` only added a route for `10.10.10.0/24` and skipped `10.10.11.0/24`. That's because the `Address` was already specified as a `/24` one. Had we specified the address as `10.10.11.10/32` instead, then `wg-quick` would have added a route for `10.10.11.0/24` explicitly.

To better understand how `AllowedIPs` work, let's go through a quick example.

Let's say this system wants to send traffic to `10.10.10.201/24`. There is a route for it which says to use the `wg0` interface for that:

```
$ ip route get 10.10.10.201
10.10.10.201 dev wg0 src 10.10.11.10 uid 1000
    cache
```

Since `wg0` is a WireGuard interface, it will consult its configuration to see if any peer has that target address in the `AllowedIPs` list. Turns out one peer has it, in which case the traffic will:

a) Be authenticated as us, and encrypted for that peer. b) Sent away via the configured `Endpoint`.

Now let's picture the reverse. This system received traffic on the `ListenPort` UDP port. If it can be decrypted, and verified as having come from one of the listed peers using its respective public key, and if the source IP matches the corresponding `AllowedIPs` list, then the traffic is accepted.

What if there is no `Endpoint`? Well, to bootstrap the VPN, at least one of the peers must have an `Endpoint`, or else it won't know where to send the traffic to, and you will get an error saying "Destination address required" (see the *troubleshooting section* for details).

But once the peers know each other, the one that didn't have an `Endpoint` setting in the interface will remember where the traffic came from, and use that address as the current endpoint. This has a very nice side effect of automatically tracking the so called "road warrior" peer, which keeps changing its IP. This is very common with laptops that keep being suspended and awakened in a new network, and then try to establish the VPN again from that new address.

### Peers

You will notice that the term "peers" is used preferably to "server" or "client". Other terms used in some VPN documentation are "left" and "right", which is already starting to convey that the difference between a "server" and a "client" is a bit blurry. It only matters, if at all, at the start of the traffic exchange: who sends the first packet of data?

In that sense, "servers" expect to sit idle and wait for connections to be initiated to them, and "clients" are the initiators. For example, a laptop in a public cafe initiating a connection to the company VPN peer. The laptop needs to know the address of that peer, because it's initiating the exchange. But the "server" doesn't need to know the IP of the laptop beforehand.

On a site-to-site VPN, however, when two separate networks are connected through the tunnel, who is the server and who is the client? Both! So it's best to call them "peers" instead.

### Putting it all together

Key takeaways from this introduction:

- Each peer participating in the WireGuard VPN has a private key and a public key.
- `AllowedIPs` is used as a routing key when sending traffic, and as an *ACL* when receiving traffic.
- To establish a VPN with a remote peer, you need its public key. Likewise, the remote peer will need your public key.
- At least one of the peers needs an `Endpoint` configured in order to be able to initiate the VPN.

To help better understand these (and other) concepts, we will create some WireGuard VPNs in the next sections, illustrating some common setups.

### Peer-to-site

- *About peer-to-site*
- *Set up peer-to-site "on router"*
- *Set up peer-to-site on an internal device*

### Site-to-site

- *Set up site-to-site*

### Default gateway

- *Using the VPN as the default gateway*

### Other common tasks, hints and tips

- *Common tasks*
- *Security tips*
- *Troubleshooting*

> **Note**
>
> Throughout this guide, we will sometimes mention a VPN "connection". This is technically false, as WireGuard uses UDP and there is no persistent connection. The term is used just to facilitate understanding, and means that the peers in the examples know each other and have completed a handshake already.

### Further reading

- See the WireGuard website for more detailed information.
- The WireGuard Quickstart has a good introduction and demo.
- `wg(8)` manual page
- `wg-quick(8)` manual page
- Detailed explanation of the algorithms used by WireGuard.

### OpenVPN client implementations

#### Linux Network-Manager GUI for OpenVPN

Many Linux distributions (including Ubuntu Desktop variants) come with Network Manager; a *GUI* to configure your network settings. It also can manage your VPN connections. It is the default, but if in doubt make sure you have the `network-manager-openvpn` package installed.

- Open the Network Manager GUI, select the VPN tab and then the 'Add' button
- Select OpenVPN as the VPN type in the opening requester and press 'Create'
- In the next window, add the OpenVPN's server name as the 'Gateway'
  - Set 'Type' to 'Certificates (TLS)'
  - Point 'User Certificate' to your user certificate
  - Point 'CA Certificate' to your CA certificate
  - Point 'Private Key' to your private key file.

- Use the 'advanced' button to enable compression (e.g. `comp-lzo`), dev tap, or other special settings you want to set on the server. Now try to establish your VPN.

#### OpenVPN with GUI for Mac OS X

Tunnelblick is an excellent free, open source implementation of a GUI for OpenVPN for OS X. Download the latest OS X installer from there and install it.

It also is recommended by upstream, which has an alternative of their own.

Then put your `client.ovpn` config file together with the certificates and keys in `/Users/username/Library/Application Support/Tunnelblick/Configurations/` and launch Tunnelblick from your 'Application' folder.

Instead of downloading manually, if you have brew set up on MacOS this is as easy as running:

```
brew cask install tunnelblick
```

#### OpenVPN with GUI for Win

First, download and install the latest OpenVPN Windows Installer. As of this writing, the management GUI is included with the Windows binary installer.

You need to start the OpenVPN service. Go to Start > Computer > Manage > Services and Applications > Services. Find the OpenVPN service and start it. Set its startup type to 'automatic'.

When you start the OpenVPN MI GUI the first time you need to run it as an administrator. You have to right click on it and you will see that option.

There is an updated guide by the upstream project for the client on Windows.

#### Further reading

- See the OpenVPN website for additional information.
- OpenVPN hardening security guide
- Also, Packt's OpenVPN: Building and Integrating Virtual Private Networks is a good resource.

#### See also

- How-to: *Security*

## 4.2. Networking

*Our networking section* will give you an introduction to networking and details on some of the key topics, such as:

- **Network tooling and configuration**
- **Network shares**

### 4.2.1. Networking

This section contains introductions to (and more details about) various aspects of networking in Ubuntu.

## Introduction

If you are new to networking, you should start with these pages to obtain an understanding of the key terms and concepts.

### Introduction to networking

Networks consist of two or more devices which are connected by either physical cabling or wireless links for the purpose of sharing information. Devices can be computer systems, printers, and related equipment.

In this overview, we'll take a look at some of the key principles involved in networks, and some of the most popular tools available to help you manage your networks.

### Networking key concepts

If you're new to networking, our explanatory *Networking key concepts* section provides an overview of some important concepts. It includes detailed discussion of the popular network protocols: TCP/IP; IP routing; TCP and UDP; and ICMP.

### Network configuration with Netplan

Ubuntu uses Netplan to configure networks. Netplan is a high-level, distribution-agnostic tool that uses a YAML configuration file to define your network setup. Read *more about Netplan*.

If you are a server administrator, check out our explanatory guide on *configuring networks"*.

### Network tools and services

#### DHCP

The Dynamic Host Configuration Protocol (DHCP) enables host computers to be automatically assigned settings from a server. To learn more about DHCP and how configuration works, we have *an explanatory guide*.

There are two DHCP servers available on Ubuntu.

- **isc-kea** (available from 23.04 onwards)
- **isc-dhcp-server** (no longer supported by vendor)

Learn how to *install isc-kea* or *install and configure isc-dhcp-server*.

### Time synchronization

Network Time Protocol (NTP) synchronizes time between all devices on a network to within a few milliseconds of Coordinated Universal Time (UTC). Learn more about *time synchronization*.

Time is primarily synchronized in Ubuntu by `chrony`. To find out how to configure this service, including Network Time Security (NTS) *read our how-to guide*.

If you want to set up a server to *provide* NTP information, we suggest `chrony`. Learn *how to serve NTP using chrony* with this guide.

### The DPDK library

The Data Plane Development Kit (DPDK) is a set of libraries that improve network performance. Learn more *about DPDK and its use in Ubuntu*.

One popular piece of software that uses DPDK is Open vSwitch (OVS). It functions as a virtual switch in virtual machine (VM) environments, providing connectivity between VMs. OVS can run inside a VM or at the hypervisor level. Check out our guide to find out *how to use DPDK with Open vSwitch*.

### Other networking functionality

### Samba

If you need to network Ubuntu and Microsoft devices together, use Samba. To get started, check out our *introduction to Samba*.

### Networking key concepts

This section provides high-level information pertaining to networking, including an overview of network concepts and detailed discussion of popular network protocols.

### The Transmission Control Protocol and Internet Protocol (TCP/IP)

The Transmission Control Protocol and Internet Protocol is a standard set of protocols developed in the late 1970s by the Defense Advanced Research Projects Agency (*DARPA*) as a means of communication between different types of computers and computer networks. TCP/IP is the driving force of the Internet, and thus it is the most popular set of network protocols on Earth.

### TCP/IP overview

The two protocol components of TCP/IP deal with different aspects of computer networking.

- **Internet Protocol** – the "IP" of TCP/IP – is a connectionless protocol that deals only with network packet routing using the *IP Datagram* as the basic unit of networking information. The IP *Datagram* consists of a header followed by a message.
- **Transmission Control Protocol** – the "TCP" of TCP/IP – enables network hosts to establish connections that may be used to exchange data streams. TCP also guarantees that data sent between connections is delivered, and that it arrives at one network host in the same order as sent from another network host.

### TCP/IP configuration

The TCP/IP protocol configuration consists of several elements that must be set by editing the appropriate configuration files, or by deploying solutions such as the Dynamic Host Configuration Protocol (*DHCP*) server which can, in turn, be configured to provide the proper TCP/IP configuration settings to network clients automatically. These configuration values must be set correctly in order to facilitate the proper network operation of your Ubuntu system.

The common configuration elements of TCP/IP and their purposes are as follows:

- **IP address**: The IP address is a unique identifying string expressed as four decimal numbers ranging from zero (0) to two-hundred and fifty-five (255), separated by periods, with each of the four numbers representing eight (8) bits of the address for a total length of thirty-two (32) bits for the whole address. This format is called *dotted quad notation*.
- **Netmask**: The subnet mask (or simply, *netmask*) is a local bit mask, or set of flags, which separate the portions of an IP address significant to the network from the bits significant to the *subnetwork*. For example, in a Class C network, the standard netmask is 255.255.255.0 which masks the first three bytes of the IP address and allows the last byte of the IP address to remain available for specifying hosts on the subnetwork.
- **Network address**: The network address represents the bytes comprising the network portion of an IP address. For example, the host 12.128.1.2 in a Class A network would use 12.0.0.0 as the network address, where twelve (12) represents the first byte of the IP address, (the network part) and zeroes (0) in all of the remaining three bytes to represent the potential host values. A network host using the private IP address 192.168.1.100 would in turn use a network address of 192.168.1.0, which specifies the first three bytes of the Class C 192.168.1 network and a zero (0) for all the possible hosts on the network.
- **Broadcast address**: The broadcast address is an IP address that allows network data to be sent simultaneously to all hosts on a given subnetwork, rather than specifying a particular host. The standard general broadcast address for IP networks is 255.255.255.255, but this broadcast address cannot be used to send a broadcast message to every host on the Internet because routers block it. A more appropriate broadcast address is set to match a specific subnetwork. For example, on the private Class C IP network, 192.168.1.0, the broadcast address is 192.168.1.255. Broadcast messages are typically produced by network protocols such as the Address Resolution Protocol (ARP) and the Routing Information Protocol (RIP).
- **Gateway address**: A gateway address is the IP address through which a particular network, or host on a network, may be reached. If one network host wishes to communicate with another network host, and that host is not located on the same network, then a *gateway* must be used. In many cases, the gateway address will be that of a router on the same network, which will in turn pass traffic on to other networks or hosts, such as Internet hosts. The value of the Gateway Address setting must be correct, or your system will not be able to reach any hosts beyond those on the same network.
- **Nameserver address**: Nameserver addresses represent the IP addresses of Domain Name Service (DNS) systems, which resolve network hostnames into IP addresses. There are three levels of nameserver addresses, which may be specified in order of precedence: The *primary* nameserver, the *secondary* nameserver, and the *tertiary* nameserver. So that your system can resolve network hostnames into their corresponding IP addresses, you must specify valid nameserver addresses that you are authorized to use in your system's TCP/IP configuration. In many cases, these addresses can and will be provided by your network service provider, but many free and publicly accessible nameservers are available for use, such as the Level3 (Verizon) servers with IP addresses from 4.2.2.1 to 4.2.2.6.

**Tip**

The IP address, netmask, network address, broadcast address, gateway address, and

nameserver addresses are typically specified via the appropriate directives in the file `/etc/network/interfaces`. For more information, view the system manual page for `interfaces`, with the following command typed at a terminal prompt:

```
man interfaces
```

### IP routing

IP routing is a way of specifying and discovering paths in a TCP/IP network that network data can be sent along. Routing uses a set of *routing tables* to direct the forwarding of network data packets from their source to the destination, often via many intermediary network nodes known as *routers*. There are two primary forms of IP routing: *static routing* and *dynamic routing*.

Static routing involves manually adding IP routes to the system's routing table, and this is usually done by manipulating the routing table with the `route` command. Static routing enjoys many advantages over dynamic routing, such as simplicity of implementation on smaller networks, predictability (the routing table is always computed in advance, and thus the route is precisely the same each time it is used), and low overhead on other routers and network links due to the lack of a dynamic routing protocol. However, static routing does present some disadvantages as well. For example, static routing is limited to small networks and does not scale well. Static routing also fails completely to adapt to network outages and failures along the route due to the fixed nature of the route.

Dynamic routing depends on large networks with multiple possible IP routes from a source to a destination and makes use of special routing protocols, such as the Router Information Protocol (RIP), which handle the automatic adjustments in routing tables that make dynamic routing possible. Dynamic routing has several advantages over static routing, such as superior scalability and the ability to adapt to failures and outages along network routes. Additionally, there is less manual configuration of the routing tables, since routers learn from one another about their existence and available routes. This trait also prevents mistakes being introduced into the routing tables via human error. Dynamic routing is not perfect, however, and presents disadvantages such as heightened complexity and additional network overhead from router communications, which does not immediately benefit the end users but still consumes network bandwidth.

### About TCP and UDP

TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) are the most common protocols used to transfer data over networks.

TCP is a connection-based protocol, offering error correction and guaranteed delivery of data via what is known as *flow control*. Flow control determines when the flow of a data stream needs to be stopped, and previously-sent data packets should be re-sent due to problems such as *collisions*, for example, thus ensuring complete and accurate delivery of the data. TCP is typically used in the exchange of important information such as database transactions.

UDP on the other hand, is a *connectionless* protocol which seldom deals with the transmission of important data because it lacks flow control or any other method to ensure reliable delivery of the data. UDP is commonly used in such applications as audio and video streaming, where it is considerably faster than TCP due to the lack of error correction and flow control, and where the loss of a few packets is not generally catastrophic.

### Internet Control Messaging Protocol (ICMP)

The Internet Control Messaging Protocol is an extension to the Internet Protocol (IP) as defined in the Request For Comments (RFC) #792, and supports network packets containing control, error, and informational messages. ICMP is used by such network applications as the ping utility, which can determine the availability of a network host or device. Examples of some error messages returned by ICMP which are useful to both network hosts and devices such as routers, include *Destination Unreachable* and *Time Exceeded*.

### About daemons

Daemons are special system applications which typically execute continuously in the background and await requests for the functions they provide from other applications. Many daemons are network-centric; that is, a large number of daemons executing in the background on an Ubuntu system may provide network-related functionality. Such network daemons include the *Hyper Text Transport Protocol Daemon* (`httpd`), which provides web server functionality; the *Secure SHell Daemon* (`sshd`), which provides secure remote login shell and file transfer capabilities; and the *Internet Message Access Protocol Daemon* (`imapd`), which provides E-Mail services.

### Resources

- There are manual pages for `TCP` and `IP` that contain more useful information.
- Also, see the TCP/IP Tutorial and Technical Overview IBM Redbook.
- Another resource is O'Reilly's "TCP/IP Network Administration".

### Configuration

Network configuration in Ubuntu is handled through Netplan. Find out more *About Netplan* or get started with our walkthrough on *network configuration* which gives a practical demonstration.

### About Netplan

Network configuration on Ubuntu is handled through Netplan, which provides a high-level, distribution-agnostic way to define how the network on your system should be set up via a YAML configuration file.

It is just as useful for configuring networking connectivity on a personal Raspberry Pi project as it is for enterprise systems administrators, who may need to configure and deploy complex networking setups in a consistent way across servers.

It is also flexible enough to be used in virtual environments and containerised deployments where network requirements might be more dynamic. Network bridges for VMs and containers can be straightforwardly defined in the YAML configuration file, and changed without needing to restart the entire network.

Netplan integrates with both of the primary Linux network management daemons: NetworkManager and systemd-networkd. For more general information about Netplan and how it works, see the introduction to Netplan in the official Netplan documentation.

Server admins may want to get started by checking out our guide to *configuring networks*. For more specific networking tasks with Netplan, we recommend checking out the list of how-to

guides in their documentation.

### Configuring networks

Network configuration on Ubuntu is handled through Netplan, which provides a high-level, distribution-agnostic way to define how the network on your system should be set up via a YAML configuration file.

While Netplan is a configuration abstraction renderer that covers all aspects of network configuration, here we will outline the underlying system elements like IP addresses, ethernet devices, name resolution and so on. We will refer to the related Netplan settings where appropriate, but we do recommend studying the Netplan documentation in general.

### Ethernet interfaces

Ethernet interfaces are identified by the system using predictable network interface names. These names can appear as `eno1` or `enp0s25`. However, in some cases an interface may still use the kernel *eth#* style of naming.

### Identify Ethernet interfaces

To quickly identify all available Ethernet interfaces, you can use the `ip` command as shown below.

```
ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s25: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
group default qlen 1000
    link/ether 00:16:3e:e2:52:42 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.102.66.200/24 brd 10.102.66.255 scope global dynamic eth0
       valid_lft 3257sec preferred_lft 3257sec
    inet6 fe80::216:3eff:fee2:5242/64 scope link
       valid_lft forever preferred_lft forever
```

Another application that can help identify all network interfaces available to your system is the `lshw` command. This command provides greater details around the hardware capabilities of specific adapters. In the example below, `lshw` shows a single Ethernet interface with the logical name of *eth4* along with bus information, driver details and all supported capabilities.

```
sudo lshw -class network
  *-network
       description: Ethernet interface
       product: MT26448 [ConnectX EN 10GigE, PCIe 2.0 5GT/s]
       vendor: Mellanox Technologies
       physical id: 0
       bus info: pci@0004:01:00.0
```

(continues on next page)

(continued from previous page)

```
       logical name: eth4
       version: b0
       serial: e4:1d:2d:67:83:56
       slot: U78CB.001.WZS09KB-P1-C6-T1
       size: 10Gbit/s
       capacity: 10Gbit/s
       width: 64 bits
       clock: 33MHz
       capabilities: pm vpd msix pciexpress bus_master cap_list ethernet physical
fibre 10000bt-fd
       configuration: autonegotiation=off broadcast=yes driver=mlx4_en
driverversion=4.0-0 duplex=full firmware=2.9.1326 ip=192.168.1.1 latency=0
link=yes multicast=yes port=fibre speed=10Gbit/s
       resources: iomemory:24000-23fff irq:481 memory:3fe200000000-3fe2000fffff
memory:240000000000-240007ffffff
```

### Ethernet Interface logical names

Interface logical names can also be configured via a Netplan configuration. If you would like control which interface receives a particular logical name use the `match` and `set-name` keys. The `match` key is used to find an adapter based on some criteria like MAC address, driver, etc. The `set-name` key can be used to change the device to the desired logical name.

```
network:
  version: 2
  renderer: networkd
  ethernets:
    eth_lan0:
      dhcp4: true
      match:
        macaddress: 00:11:22:33:44:55
      set-name: eth_lan0
```

### Ethernet Interface settings

`ethtool` is a program that displays and changes Ethernet card settings such as auto-negotiation, port speed, duplex mode, and Wake-on-LAN. The following is an example of how to view the supported features and configured settings of an Ethernet interface.

```
sudo ethtool eth4
Settings for eth4:
    Supported ports: [ FIBRE ]
    Supported link modes:   10000baseT/Full
    Supported pause frame use: No
    Supported auto-negotiation: No
    Supported FEC modes: Not reported
    Advertised link modes:  10000baseT/Full
    Advertised pause frame use: No
    Advertised auto-negotiation: No
```

(continues on next page)

(continued from previous page)

```
    Advertised FEC modes: Not reported
    Speed: 10000Mb/s
    Duplex: Full
    Port: FIBRE
    PHYAD: 0
    Transceiver: internal
    Auto-negotiation: off
    Supports Wake-on: d
    Wake-on: d
    Current message level: 0x00000014 (20)
                   link ifdown
    Link detected: yes
```

### IP addressing

The following section describes the process of configuring your system's IP address and default gateway needed for communicating on a local area network and the Internet.

### Temporary IP address assignment

For temporary network configurations, you can use the `ip` command which is also found on most other *GNU*/Linux operating systems. The `ip` command allows you to configure settings which take effect immediately – however they are not persistent and will be lost after a reboot.

To temporarily configure an IP address, you can use the `ip` command in the following manner. Modify the IP address and subnet mask to match your network requirements.

```
sudo ip addr add 10.102.66.200/24 dev enp0s25
```

The `ip` can then be used to set the link up or down.

```
ip link set dev enp0s25 up
ip link set dev enp0s25 down
```

To verify the IP address configuration of `enp0s25`, you can use the `ip` command in the following manner:

```
ip address show dev enp0s25
10: enp0s25: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
group default qlen 1000
    link/ether 00:16:3e:e2:52:42 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.102.66.200/24 brd 10.102.66.255 scope global dynamic eth0
       valid_lft 2857sec preferred_lft 2857sec
    inet6 fe80::216:3eff:fee2:5242/64 scope link
       valid_lft forever preferred_lft forever6
```

To configure a default gateway, you can use the `ip` command in the following manner. Modify the default gateway address to match your network requirements.

```
sudo ip route add default via 10.102.66.1
```

You can also use the `ip` command to verify your default gateway configuration, as follows:

```
ip route show
default via 10.102.66.1 dev eth0 proto dhcp src 10.102.66.200 metric 100
10.102.66.0/24 dev eth0 proto kernel scope link src 10.102.66.200
10.102.66.1 dev eth0 proto dhcp scope link src 10.102.66.200 metric 100
```

If you require *DNS* for your temporary network configuration, you can add DNS server IP addresses in the file `/etc/resolv.conf`. In general, editing `/etc/resolv.conf` directly is not recommended, but this is a temporary and non-persistent configuration. The example below shows how to enter two DNS servers to `/etc/resolv.conf`, which should be changed to servers appropriate for your network. A more lengthy description of the proper (persistent) way to do DNS client configuration is in a following section.

```
nameserver 8.8.8.8
nameserver 8.8.4.4
```

If you no longer need this configuration and wish to purge all IP configuration from an interface, you can use the `ip` command with the flush option:

```
ip addr flush eth0
```

**Note**

Flushing the IP configuration using the `ip` command does not clear the contents of `/etc/resolv.conf`. You must remove or modify those entries manually (or re-boot), which should also cause `/etc/resolv.conf`, which is a symlink to `/run/systemd/resolve/stub-resolv.conf`, to be re-written.

### Dynamic IP address assignment (DHCP client)

To configure your server to use DHCP for dynamic address assignment, create a Netplan configuration in the file `/etc/netplan/99_config.yaml`. The following example assumes you are configuring your first Ethernet interface identified as `enp3s0`.

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp3s0:
      dhcp4: true
```

The configuration can then be applied using the `netplan` command:

```
sudo netplan apply
```

### Static IP address assignment

To configure your system to use static address assignment, create a `netplan` configuration in the file `/etc/netplan/99_config.yaml`. The example below assumes you are configuring your first Ethernet interface identified as `eth0`. Change the `addresses`, `routes`, and `nameservers` values to meet the requirements of your network.

```
network:
  version: 2
  renderer: networkd
  ethernets:
    eth0:
      addresses:
        - 10.10.10.2/24
      routes:
        - to: default
          via: 10.10.10.1
      nameservers:
          search: [mydomain, otherdomain]
          addresses: [10.10.10.1, 1.1.1.1]
```

The configuration can then be applied using the `netplan` command.

```
sudo netplan apply
```

**Note**

`netplan` in Ubuntu Bionic 18.04 LTS doesn't understand the "`to: default`" syntax to specify a default route, and should use the older `gateway4: 10.10.10.1` key instead of the whole `routes:` block.

The loopback interface is identified by the system as `lo` and has a default IP address of 127.0.0.1. It can be viewed using the `ip` command.

```
ip address show lo
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
```

### Name resolution

Name resolution (as it relates to IP networking) is the process of mapping *hostnames* to IP addresses, and vice-versa, making it easier to identify resources on a network. The following section will explain how to properly configure your system for name resolution using DNS and static hostname records.

Traditionally, the file `/etc/resolv.conf` was a static configuration file that rarely needed to

be changed, or it automatically changed via DHCP client hooks. `systemd-resolved` handles nameserver configuration, and it should be interacted with through the `systemd-resolve` command. Netplan configures `systemd-resolved` to generate a list of nameservers and domains to put in `/etc/resolv.conf`, which is a symlink:

```
/etc/resolv.conf -> ../run/systemd/resolve/stub-resolv.conf
```

To configure the resolver, add the IP addresses of the appropriate nameservers for your network to the `netplan` configuration file. You can also add optional DNS suffix search-lists to match your network domain names. The resulting file might look like the following:

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s25:
      addresses:
        - 192.168.0.100/24
      routes:
        - to: default
          via: 192.168.0.1
      nameservers:
          search: [mydomain, otherdomain]
          addresses: [1.1.1.1, 8.8.8.8, 4.4.4.4]
```

The *search* option can also be used with multiple domain names so that DNS queries will be appended in the order in which they are entered. For example, your network may have multiple sub-domains to search; a parent domain of `example.com`, and two sub-domains, `sales.example.com` and `dev.example.com`.

If you have multiple domains you wish to search, your configuration might look like the following:

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s25:
      addresses:
        - 192.168.0.100/24
      routes:
        - to: default
          via: 192.168.0.1
      nameservers:
          search: [example.com, sales.example.com, dev.example.com]
          addresses: [1.1.1.1, 8.8.8.8, 4.4.4.4]
```

If you try to ping a host with the name `server1`, your system will automatically query DNS for its *Fully Qualified Domain Name (FQDN)* in the following order:

1. `server1.example.com`
2. `server1.sales.example.com`

3. `server1.dev.example.com`

If no matches are found, the DNS server will provide a result of *notfound* and the DNS query will fail.

### Static hostnames

Static hostnames are locally defined hostname-to-IP mappings located in the file `/etc/hosts`. Entries in the `hosts` file will have precedence over DNS by default. This means that if your system tries to resolve a hostname and it matches an entry in `/etc/hosts`, it will not attempt to look up the record in DNS. In some configurations, especially when Internet access is not required, servers that communicate with a limited number of resources can be conveniently set to use static hostnames instead of DNS.

The following is an example of a `hosts` file where a number of local servers have been identified by simple hostnames, aliases and their equivalent Fully Qualified Domain Names (FQDN's):

```
127.0.0.1   localhost
127.0.1.1   ubuntu-server
10.0.0.11   server1 server1.example.com vpn
10.0.0.12   server2 server2.example.com mail
10.0.0.13   server3 server3.example.com www
10.0.0.14   server4 server4.example.com file
```

**Note**

In this example, notice that each of the servers were given aliases in addition to their proper names and FQDN's. *Server1* has been mapped to the name *vpn*, *server2* is referred to as *mail*, *server3* as *www*, and *server4* as *file*.

### Name Service Switch (NSS) configuration

The order in which your system selects a method of resolving hostnames to IP addresses is controlled by the Name Service Switch (NSS) configuration file `/etc/nsswitch.conf`. As mentioned in the previous section, typically static hostnames defined in the systems `/etc/hosts` file have precedence over names resolved from DNS. The following is an example of the line responsible for this order of hostname lookups in the file `/etc/nsswitch.conf`.

```
hosts:          files mdns4_minimal [NOTFOUND=return] dns mdns4
```

- `files` first tries to resolve static hostnames located in `/etc/hosts`.
- `mdns4_minimal` attempts to resolve the name using Multicast DNS.
- `[NOTFOUND=return]` means that any response of `notfound` by the preceding `mdns4_minimal` process should be treated as authoritative and that the system should not try to continue hunting for an answer.
- `dns` represents a legacy unicast DNS query.
- **mdns4** represents a multicast DNS query.

To modify the order of these name resolution methods, you can simply change the `hosts:` string to the value of your choosing. For example, if you prefer to use legacy unicast DNS versus multicast DNS, you can change the string in `/etc/nsswitch.conf` as shown below:

```
hosts:          files dns [NOTFOUND=return] mdns4_minimal mdns4
```

### Bridging multiple interfaces

Bridging is a more advanced configuration, but is very useful in multiple scenarios. One scenario is setting up a bridge with multiple network interfaces, then using a firewall to filter traffic between two network segments. Another scenario is using bridge on a system with one interface to allow virtual machines direct access to the outside network. The following example covers the latter scenario:

Configure the bridge by editing your `netplan` configuration found in `/etc/netplan/`, entering the appropriate values for your physical interface and network:

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp3s0:
      dhcp4: no
  bridges:
    br0:
      dhcp4: yes
      interfaces:
        - enp3s0
```

Now apply the configuration to enable the bridge:

```
sudo netplan apply
```

The new bridge interface should now be up and running. The `brctl` provides useful information about the state of the bridge, controls which interfaces are part of the bridge, etc. See `man brctl` for more information.

### networkd-dispatcher for hook scripts

Users of the former `ifupdown` may be familiar with using hook scripts (e.g., pre-up, post-up) in their interfaces file. Netplan configuration does not currently support hook scripts in its configuration definition.

Instead, to achieve this functionality with the `networkd` renderer, users can use `networkd-dispatcher(8)`. The package provides both users and packages with hook points when specific network states are reached, to aid in reacting to network state.

**Note**

If you are on Desktop (not Ubuntu Server) the network is driven by Network Manager - in that case you need NM Dispatcher scripts instead.

The Netplan FAQ has a great table that compares event timings between `ifupdown`/`systemd-networkd`/`network-manager`.

It is important to be aware that these hooks run asynchronously; i.e. they will not block transition into another state.

The Netplan FAQ also has an example on converting an old `ifupdown` hook to `networkd-dispatcher`.

### Resources

- The Ubuntu Wiki Network page has links to articles covering more advanced network configuration.
- The Netplan website has additional examples and documentation.
- The Netplan `manual page` has more information on Netplan.
- The `systemd-resolved(8)` manual page has more information on `systemd-resolved` service.
- For more information on *bridging* see the netplan.io examples page

### Network tools

In our how-to section we show to set up virtual switching using Open vSwitch (OVS) and the Data Plane Development Kit (DPDK) library. This page discusses DPDK in more detail.

- *About DPDK* discusses DPDK in more detail

### About DPDK

The Data Plane Development Kit (DPDK) is a set of libraries and drivers for fast packet processing, which runs mostly in Linux userland. This set of libraries provides the so-called "Environment Abstraction Layer" (*EAL*). The EAL hides the details of the environment and provides a standard programming interface. Common use cases are around special solutions, such as network function virtualisation and advanced high-throughput network switching.

The DPDK uses a run-to-completion model for fast data plane performance and accesses devices via polling to eliminate the latency of interrupt processing, albeit with the tradeoff of higher CPU consumption. It was designed to run on any processor. The first supported CPU was Intel x86 and it is now extended to IBM PPC64 and ARM64.

Ubuntu provides some additional infrastructure to increase DPDK's usability.

### Prerequisites

This package is currently compiled for the lowest possible CPU requirements allowed by upstream. Starting with DPDK 17.08, that means it requires at least SSE4_2 and for anything else activated by -march=corei7 (in GCC) to be supported by the CPU.

The list of upstream DPDK-supported network cards can be found at supported NICs. However, a lot of those are disabled by default in the upstream project as they are not yet in a stable state. The subset of network cards that DPDK has enabled in the package (as available in Ubuntu 16.04) is:

DPDK has "userspace" drivers for the cards called PMDs. The packages for these follow the pattern of `librte-pmd-<type>-<version>`. Therefore the example for an Intel e1000 in 18.11 would be `librte-pmd-e1000-18.11`.

The more commonly used, tested and fully supported drivers are installed as dependencies of `dpdk`. But there are many more "in-universe" that follow the same naming pattern.

### Unassign the default kernel drivers

Cards must be unassigned from their kernel driver and instead be assigned to `uio_pci_generic` of `vfio-pci`. `uio_pci_generic` is older and it's (usually) easier to get it to work. However, it also has fewer features and less isolation.

The newer VFIO-PCI requires that you activate the following kernel parameters to enable the input-output memory management unit (IOMMU):

```
iommu=pt intel_iommu=on
```

Alternatively, on *AMD*:

```
amd_iommu=pt
```

On top of VFIO-PCI, you must also configure and assign the IOMMU groups accordingly. This is mostly done in *firmware* and by hardware layout – you can check the group assignment the kernel probed in `/sys/kernel/iommu_groups/`.

**Note**

VirtIO is special. DPDK can directly work on these devices without `vfio_pci/uio_pci_generic`. However, to avoid issues that might arise from the kernel and DPDK managing the device, you still need to unassign the kernel driver.

Manual configuration and status checks can be done via `sysfs`, or with the tool `dpdk_nic_bind`:

```
dpdk_nic_bind.py --help
```

### Usage

```
dpdk-devbind.py [options] DEVICE1 DEVICE2 ....

where DEVICE1, DEVICE2 etc, are specified via PCI "domain:bus:slot.func" syntax
or "bus:slot.func" syntax. For devices bound to Linux kernel drivers, they may
also be referred to by Linux interface name e.g. eth0, eth1, em0, em1, etc.

Options:
--help, --usage:
    Display usage information and quit

-s, --status:
    Print the current status of all known network, crypto, event
    and mempool devices.
```

(continues on next page)

(continued from previous page)

```
    For each device, it displays the PCI domain, bus, slot and function,
    along with a text description of the device. Depending upon whether the
    device is being used by a kernel driver, the igb_uio driver, or no
    driver, other relevant information will be displayed:
    * the Linux interface name e.g. if=eth0
    * the driver being used e.g. drv=igb_uio
    * any suitable drivers not currently using that device
        e.g. unused=igb_uio
    NOTE: if this flag is passed along with a bind/unbind option, the
    status display will always occur after the other operations have taken
    place.

--status-dev:
    Print the status of given device group. Supported device groups are:
    "net", "crypto", "event", "mempool" and "compress"

-b driver, --bind=driver:
    Select the driver to use or "none" to unbind the device

-u, --unbind:
    Unbind a device (Equivalent to "-b none")

--force:
    By default, network devices which are used by Linux - as indicated by
    having routes in the routing table - cannot be modified. Using the
    --force flag overrides this behavior, allowing active links to be
    forcibly unbound.
    WARNING: This can lead to loss of network connection and should be used
    with caution.

Examples:
---------

To display current device status:
    dpdk-devbind.py --status

To display current network device status:
    dpdk-devbind.py --status-dev net

To bind eth1 from the current driver and move to use igb_uio
    dpdk-devbind.py --bind=igb_uio eth1

To unbind 0000:01:00.0 from using any driver
    dpdk-devbind.py -u 0000:01:00.0

To bind 0000:02:00.0 and 0000:02:00.1 to the ixgbe kernel driver
    dpdk-devbind.py -b ixgbe 02:00.0 02:00.1
```

### DPDK device configuration

The package `dpdk` provides *init* scripts that ease configuration of device assignment and huge pages. It also makes them persistent across reboots.

The following is an example of the file `/etc/dpdk/interfaces` configuring two ports of a network card: one with `uio_pci_generic` and the other with `vfio-pci`.

```
# <bus>         Currently only "pci" is supported
# <id>          Device ID on the specified bus
# <driver>      Driver to bind against (vfio-pci or uio_pci_generic)
#
# Be aware that the two DPDK compatible drivers uio_pci_generic and vfio-pci are
# part of linux-image-extra-<VERSION> package.
# This package is not always installed by default - for example in cloud-images.
# So please install it in case you run into missing module issues.
#
# <bus> <id>     <driver>
pci 0000:04:00.0 uio_pci_generic
pci 0000:04:00.1 vfio-pci
```

Cards are identified by their PCI-ID. If you are need to check, you can use the tool `dpdk_nic_bind.py` to show the currently available devices – and the drivers they are assigned to. For example, running the command `dpdk_nic_bind.py --status` provides the following details:

```
Network devices using DPDK-compatible driver
============================================
0000:04:00.0 'Ethernet Controller 10-Gigabit X540-AT2' drv=uio_pci_generic
unused=ixgbe

Network devices using kernel driver
===================================
0000:02:00.0 'NetXtreme BCM5719 Gigabit Ethernet PCIe' if=eth0 drv=tg3 unused=uio_
pci_generic *Active*
0000:02:00.1 'NetXtreme BCM5719 Gigabit Ethernet PCIe' if=eth1 drv=tg3 unused=uio_
pci_generic
0000:02:00.2 'NetXtreme BCM5719 Gigabit Ethernet PCIe' if=eth2 drv=tg3 unused=uio_
pci_generic
0000:02:00.3 'NetXtreme BCM5719 Gigabit Ethernet PCIe' if=eth3 drv=tg3 unused=uio_
pci_generic
0000:04:00.1 'Ethernet Controller 10-Gigabit X540-AT2' if=eth5 drv=ixgbe
unused=uio_pci_generic

Other network devices
=====================
<none>
```

### DPDK hugepage configuration

DPDK makes heavy use of hugepages to eliminate pressure on the translation lookaside buffer (TLB). Therefore, hugepages need to be configured in your system. The `dpdk` package has a config file and scripts that try to ease hugepage configuration for DPDK in the form of `/etc/dpdk/dpdk.conf`.

If you have more consumers of hugepages than just DPDK in your system – or very special requirements for how your hugepages will be set up – you likely want to allocate/control them yourself. If not, this can be a great simplification to get DPDK configured for your needs.

As an example, we can specify a configuration of 1024 hugepages of 2M each and four 1G pages in `/etc/dpdk/dpdk.conf` by adding:

```
NR_2M_PAGES=1024
NR_1G_PAGES=4
```

This supports configuring 2M and the larger 1G hugepages (or a mix of both). It will make sure there are proper `hugetlbfs` mountpoints for DPDK to find both sizes – no matter what size your default hugepage is. The config file itself holds more details on certain corner cases and a few hints if you want to allocate hugepages manually via a kernel parameter.

The size you want depends on your needs: 1G pages are certainly more effective regarding TLB pressure, but there have been reports of them fragmenting inside the DPDK memory allocations. Also, it can be harder to find enough free space to set up a certain number of 1G pages later in the life-cycle of a system.

### Compile DPDK applications

Currently, there are not many consumers of the DPDK library that are stable and released. Open vSwitch DPDK is an exception to that (see below) and more are appearing, but in general it may be that you will want to compile an app against the library.

You will often find guides that tell you to fetch the DPDK sources, build them to your needs and eventually build your application based on DPDK by setting values `RTE_*` for the build system. Since Ubuntu provides an already-compiled DPDK for you can can skip all that.

DPDK provides a valid pkg-config file to simplify setting the proper variables and options:

```
sudo apt-get install dpdk-dev libdpdk-dev
gcc testdpdkprog.c $(pkg-config --libs --cflags libdpdk) -o testdpdkprog
```

An example of a complex (auto-configure) user of pkg-config of DPDK including *fallbacks* to older non pkg-config style can be seen in the Open vSwitch build system.

Depending on what you are building, it may be a good idea to install all DPDK build dependencies before the make. On Ubuntu, this can be done automatically with the following command:

```
sudo apt-get install build-dep dpdk
```

### DPDK in KVM guests

Even if you have no access to DPDK-supported network cards, you can still work with DPDK by using its support for VirtIO. To do so, you must create guests backed by hugepages (see above). In addition, you will also need to have *at least* Streaming SIMD Extensions 3 (SSE3).

The default CPU model used by QEMU/libvirt is only up to SSE2. So, you will need to define a model that passes the proper feature flags (or use `host-passthrough`). As an example, you can add the following snippet to your virsh XML (or the equivalent virsh interface you use).

```
<cpu mode='host-passthrough'>
```

Nowadays, VirtIO supports multi-queue, which DPDK in turn can exploit for increased speed. To modify a normal VirtIO definition to have multiple queues, add the following snippet to your interface definition.

```
<driver name="vhost" queues="4"/>
```

This will enhance a normal VirtIO NIC to have multiple queues, which can later be consumed by e.g., DPDK in the guest.

### Use DPDK

Since DPDK itself is only a (massive) library, you most likely will continue to *Open vSwitch DPDK* as an example to put it to use.

### Resources

- DPDK documentation
- Release Notes matching the version packages in Ubuntu 16.04
- Linux DPDK user getting started
- EAL command-line options
- DPDK API documentation
- Open vSwitch DPDK installation
- Wikipedia definition of DPDK

## DHCP

The Dynamic Host Configuration Protocol (DHCP) handles automatic IP address assignment.

- *About DHCP* gives detail about DHCP and how it works

### About DHCP

The Dynamic Host Configuration Protocol (DHCP) is a network service that enables host computers to be automatically assigned settings from a server as opposed to manually configuring each network host. Computers configured to be DHCP clients have no control over the settings they receive from the DHCP server, and the configuration is transparent to the computer's user.

The most common settings provided by a DHCP server to DHCP clients include:

- IP address and netmask
- IP address of the default-gateway to use
- IP addresses of the *DNS* servers to use

However, a DHCP server can also supply configuration properties such as:

- *Hostname*
- Domain name
- Time server
- Print server

The advantage of using DHCP is that any changes to the network, such as a change in the DNS server address, only need to be changed at the DHCP server, and all network hosts will be reconfigured the next time their DHCP clients poll the DHCP server. As an added advantage, it is also easier to integrate new computers into the network, as there is no need to check for the availability of an IP address. Conflicts in IP address allocation are also reduced.

### DHCP configuration

A DHCP server can provide configuration settings using the following methods:

#### Manual allocation (MAC address)

This method uses DHCP to identify the unique hardware address of each network card connected to the network, and then supplies a static configuration each time the DHCP client makes a request to the DHCP server using that network device. This ensures that a particular address is assigned automatically to that network card, based on its MAC address.

#### Dynamic allocation (address pool)

In this method, the DHCP server assigns an IP address from a pool of addresses (sometimes also called a range or scope) for a period of time (known as a lease) configured on the server, or until the client informs the server that it doesn't need the address anymore. This way, the clients receive their configuration properties dynamically and on a "first come, first served" basis. When a DHCP client is no longer on the network for a specified period, the configuration is expired and released back to the address pool for use by other DHCP clients. After the lease period expires, the client must renegotiate the lease with the server to maintain use of the same address.

#### Automatic allocation

Using this method, the DHCP automatically assigns an IP address permanently to a device, selecting it from a pool of available addresses. Usually, DHCP is used to assign a temporary address to a client, but a DHCP server can allow an infinite lease time.

The last two methods can be considered "automatic" because in each case the DHCP server assigns an address with no extra intervention needed. The only difference between them is in how long the IP address is leased; in other words, whether a client's address varies over time.

### Available servers

Ubuntu makes two DHCP servers available:

- `isc-dhcp-server`: This server installs `dhcpd`, the dynamic host configuration protocol daemon. Although Ubuntu still supports `isc-dhcp-server`, this software is no longer supported by its vendor.

  Find out *how to install and configure isc-dhcp-server*.

- `isc-kea`: Kea was created by ISC to replace `isc-dhcp-server` – It is supported in Ubuntu releases from 23.04 onwards.

  Find out *how to install and configure isc-kea*.

### References

- The isc-dhcp-server Ubuntu Wiki page has more information.
- For more `/etc/dhcp/dhcpd.conf` options see the `dhcpd.conf(5)` manual page
- ISC dhcp-server
- ISC Kea Documentation

### Time synchronisation

- *About time synchronisation* discusses the Network Time Protocol (NTP) and how it works

### About time synchronisation

Network Time Protocol (NTP) is a networking protocol for synchronising time over a network. Basically, a client requests the current time from a server, and uses it to set its own clock.

Behind this simple description, there is a lot of complexity. There are three tiers of NTP servers; tier one NTP servers are connected to atomic clocks, while tier two and tier three three servers spread the load of actually handling requests across the Internet.

The client software is also a lot more complex than you might expect. It must factor in communication delays and adjust the time in a way that does not upset all the other processes that run on the server. Luckily, all that complexity is hidden from you!

By default, Ubuntu uses `chrony` to synchronise time, which is installed by default. See our guides, if you would like to know how to *synchronise time with Chrony* or *use chrony to serve NTP*.

Users can optionally use *timedatectl and timesyncd*.

### How time synchronisation works

Since Ubuntu 25.10, `chrony` replaces most of `systemd-timesyncd` but still uses systemd's `timedatectl`.

### About chrony

chrony replaces not only ntpdate, but also the Simple Network Time Protocol (SNTP) client of timesyncd (formerly ntpd). So, on top of the one-shot action that ntpdate provided on boot and network activation, chrony now regularly checks and keeps your local time in sync. It also stores time updates locally, so that after reboots the time monotonically advances (if applicable).

### About timedatectl

If chrony is installed, timedatectl can still be used to configure basic settings, such as setting the timezone. But more complex configuration needs to be set up using the chronyc command. This ensures that no two time-syncing services can conflict with each other.

ntpdate is now considered deprecated in favor of chrony (or timesyncd) and is no longer installed by default. chrony will generally keep your time in sync and can be set up as a time server, timesyncd can help with simpler cases. But if you had one of a few known special ntpdate use cases, consider the following:

- If you require a one-shot sync, use: `chronyd -q`
- If you require a one-shot time check (without setting the time), use: `chronyd -Q`

While use of ntpd is no longer recommended, this also still applies to ntpd being installed to retain any previous behaviour/config that you had through an upgrade. However, it also implies that on an upgrade from a former release, ntp/ntpdate might still be installed and therefore renders the new systemd-based services disabled.

### Further reading

- ntp.org: home of the Network Time Protocol project
- pool.ntp.org: project of virtual cluster of timeservers
- Freedesktop.org info on timedatectl
- Freedesktop.org info on systemd-timesyncd service
- Chrony FAQ
- Feeding chrony from GPSD
- Also see the Ubuntu Time wiki page for more information.

### Network shares

Samba and Active Directory are two tools you will need if you want to share resources and directories between Linux and Windows systems.

- *Introduction to Samba* will talk you through the key concepts you will need to be able to follow our how-to guides on Samba
- Our *Active Directory integration* section has both an introduction and in-depth discussion about Active Directory integration

## Introduction to Samba

Computer networks are often comprised of diverse systems. While operating a network made up entirely of Ubuntu desktop and server computers would definitely be fun, some network environments require both Ubuntu and Microsoft Windows systems working together in harmony.

This is where Samba comes in! Samba provides various tools for configuring your Ubuntu Server to share network resources with Windows clients. In this overview, we'll look at some of the key principles, how to install and configure the tools available, and some common Samba use cases.

## Samba functionality

There are several services common to Windows environments that your Ubuntu system needs to integrate with in order to set up a successful network. These services share data and configuration details of the computers and users on the network between them, and can each be classified under one of three main categories of functionality.

### File and printer sharing services

These services use the Server Message Block (SMB) protocol to facilitate the sharing of files, folders, volumes, and the sharing of printers throughout the network.

- **File server** Samba can be *configured as a file server* to share files with Windows clients - our guide will walk you through that process.
- **Print server** Samba can also be *configured as a print server* to share printer access with Windows clients, as detailed in this guide.

### Directory services

These services share vital information about the computers and users of the network. They use technologies like the Lightweight Directory Access Protocol (LDAP) and Microsoft Active Directory.

- **Microsoft Active Directory** An Active Directory domain is a collection of users, groups, or hardware components within a Microsoft Active Directory network. This guide will show you how to set up your server as a *member of an Active Directory domain*.
- NT4 Domain Controller *(deprecated)* This guide will show you how to configure your Samba server to appear *as a Windows NT4-style domain controller*.
- OpenLDAP backend *(deprecated)* This guide will show you how to integrate Samba with *LDAP in Windows NT4 mode*.

### Authentication and access

These services establish the identity of a computer or network user, and determine the level of access that should be granted to the computer or user. The services use such principles and technologies as file permissions, group policies, and the Kerberos authentication service.

- **Share access controls** This article provides more details on *controlling access to shared directories*.

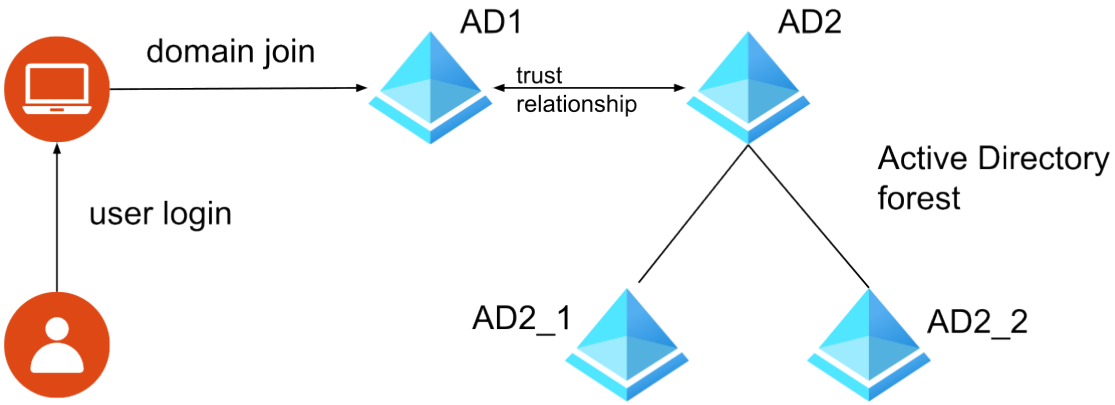
- **AppArmor profile for Samba** This guide will briefly cover how to *set up a profile for Samba* using the Ubuntu security module, AppArmor.
- **Mounting CIFS shares permanently** This guide will show you *how to set up Common Internet File System (CIFS) shares* to automatically provide access to network files and resources.

### Active Directory integration

If you have a Microsoft Active Directory domain set up, you can join your Ubuntu Server to it.

### Introduction

### Introduction to Active Directory integration




Active Directory deployments can range from single-domain, one tree, with one or more servers, up to multiple domains and servers geographically dispersed spawning a structure that is referred to as a “forest”. Furthermore, such a forest is not necessarily static, allowing its multiple delegated administrators to add and remove domains from it. Depending on the desired level of integration and the complexity of the domain or forest, joining an Ubuntu system to Active Directory requires different tooling, configuration, and planning.

Joining an Ubuntu system to an Active Directory domain (or a forest) means that the Ubuntu system will get an account in that domain, and be able to identify and authenticate users from that domain. In other words, a joined Ubuntu system should be able to:

- authenticate Active Directory users, including changing their passwords
- recognize the Active Directory users as valid users on the Ubuntu system, with linux-compatible user and group identifiers (more on that later)
- recognize group memberships

Depending on how the join was performed, and the software stack available on the Ubuntu system, the following is also possible:

- authenticate and recognize users from different domains that make up the forest
- apply certain *group policy objects* (not covered here)

- provide file and print services to users from the domain

To set up your Active Directory integrations, we suggest first familiarising yourself with the following key topics:

- *Choosing an integration method*
- *Security identifiers*
- *Identity mapping backends*
- *The rid idmap backend*
- *The autorid idmap backend*

### References

- About Active Directory:
  - Security Identifiers (SIDs)
  - Active Directory Domain Services Overview
- Samba Wiki pages:
  - Choosing an identity mapping backend
  - rid identity mapping backend
  - autorid identity mapping backend
- Manual pages:
  - `idmap_tdb(8)`
  - `idmap_rid(8)`
  - `idmap_autorid(8)`
  - `wbinfo(1)`

### Configuration

This section will talk you through the options available and help you make the appropriate choices for your setup.

### Choosing an integration method

There are multiple mechanisms to join an Ubuntu system to an Active Directory tree (single domain) or a forest (multiple domains with trust relationships). It's important to understand the features, pros, and cons, of each method, and cross reference those with what is the objective of the integration.

Three main criteria need to be evaluated:

- forest or single domain: Is the Active Directory deployment being joined composed of a single domain, or multiple domains (forest)?
- member server or workstation: Is the Ubuntu system that is being joined to Active Directory going to be a member server (it provides authentication and shares), or just a

workstation/desktop which will not share files via the SMB protocol? If a workstation is also going to share files via SMB, then consider it a member server.

- Deterministic Linux ID: is it important that the same Active Directory user obtains the same Linux ID on all joined systems? For example, if NFS exports will be used between the joined systems.

| Requirement / Method | sssd | idmap_rid | idmap_autorid |
|---|---|---|---|
| Deterministic ID | □ | □ | □ |
| Undetermined ID | □ | □ | □ |
| Member server | □ | □ | □ |
| Not a member server | □ | □ | □ |
| AD multidomain (forest) | □ | □(*) | □ |
| AD single domain | □ | □ | □ |

(*) The *idmap_rid* choice for multidomain Active Directory is doable, but with important caveats that are better explained in the *Joining a forest with the rid backend* howto.

Before going into the details of each integration method, it's a good idea to familiarise yourself with the other basic concepts of Active Directory integration. These are shown next:

- *Security identifiers*
- *Identity mapping backends*
- *The rid idmap backend*
- *The autorid idmap backend*

### Security identifiers (SIDs)

One of the more challenging aspects of integrating with Active Directory is called Identity Mapping. Windows entities (users, groups, computers, etc) have identifiers called SID, which stands for *Security IDentifier*. This is not just a number: it has a structure and is composed of several values. Linux users and groups identifiers, on the other hand, are a single number..

**Note**

For a full explanation of what an SID is, and how they are structured and allocated, please refer to Understand Security Identifiers.

For the purposes of integration, the more important fields of a SID are the *Domain Identifier* and the *Relative Identifier*:

```
S-1-5-<Domain-Identifier>-<Relative-Identifier>
```

Here are some examples of what an SID looks like:

- `S-1-5-32-544`: this is the so called well-known SID of the built-in (`32`) Administrators (`544`) group
- `S-1-5-21-1004336348-1177238915-682003330-512`: a Domain Admins group (`512`) in a specific domain (`21-1004336348-1177238915-682003330`)

- `S-1-5-21-1132786674-3270659749-157314226-512`: also a Domain Admins group (`512`), but in another domain (`21-1132786674-3270659749-157314226`)
- `S-1-5-21-1132786674-3270659749-157314226-1103`: a user with *Relative Identifier* (RID) of `1103`, in the domain `21-1132786674-3270659749-157314226`

Within a domain, the *Domain Identifier* remains the same, and the *Relative Identifier* is what distinguishes the users and groups of that domain. In other words, the *Domain Admins* group will always have a *Relative Identifier* (RID) of `512` in all domains, but the *Domain Identifier* will be different. And thus we have a unique global *Secure Identifier*.

In order to integrate an Ubuntu system with Active Directory, some sort of mapping is needed between the SIDs on the AD site, and the numeric identifiers on the Linux side. At first glance, the RID component of the SID looks like a very good candidate, and indeed it is used by most mapping mechanisms, but it's not truly unique since it misses the Domain Identifier.

The *winbind* package, part of the Samba suite of applications, is capable of performing a mapping between SIDs and Linux IDs, and has several mechanisms to do so, each with some pros and cons. These mechanisms are called *Identity Mapping Backends*, and how to choose one will depend on the type of domain being joined (single, forest, etc), and what level of integration we want on the Ubuntu System.

See next:

- *Identity mapping backends*
- *The rid idmap backend*
- *The autorid idmap backend*

### Identity Mapping (idmap) backends

In order to have Domain users and groups from an Active Directory system appear in an Ubuntu system as valid, they need to have Linux specific attributes. Some of these attributes map directly to equivalent ones in Active Directory, some need to be translated to something else, and others do not exist at all. This is the problem that the Identity Mapping Backends (`idmap`) try to solve.

There are basically three idmap backends to chose from:

- `idmap_ad(8)`: requires that Active Directory be augmented with RFC-2307 attributes
- `idmap_rid(8)`: algorithmic user/group id generation, manual domain configuration
- `idmap_autorid(8)`: algorithmic user/group ID generation, automatic domain configuration

Of these, the simplest ones to use are `rid` and `autorid`, because they require no changes to Active Directory. These are the ones we will examine next.

There is another idmap backend that we need to introduce, not related to Active Directory integration, but necessary in some cases: the `tdb` (Trivial Data Base) backend. The `idmap_tdb(8)` backend is an *allocating backend* that stores the mappings on a persistent database on disk. It is needed whenever the mapping is not deterministic, and is instead done on a first come, first serve, order. Configurations using the `idmap_rid` backend need to be supported by the `idmap_tdb` backend as well, as will be shown later.

To better understand how these mapping mechanisms work, it helps to have a quick refresher on the typical user ID ranges on an Ubuntu/Debian system:

- `0-99`: builtin global allocations, shipped in the `base-passwd` package
- `100-999`: dynamically allocated system users and groups, typically created by packages for services as they are installed
- `1000-59999`: dynamically allocated normal user/group accounts (*)
- `60000-64999`: other global allocations
- `65000-65533`: reserved
- `65534`: the `nobody` user, and corresponding `nogroup` group (*)
- `65536-4294967293`: dynamically allocated user/group accounts
- `4294967294` and `4294967295`: do not use

Most of these ranges are configured in `/etc/adduser.conf`, and the above are the default values.

The Active Directory domain users and groups need to fit somewhere, and the largest block available is `65536-4294967293`, so that is typically what is used.

See next:

- *The rid idmap backend*
- *The autorid idmap backend*

### The rid idmap backend

The `idmap_rid(8)` idmap backend provides an algorithmic mapping between Linux uids/*gids* and Active Directory SIDs. That means that a given SID will always map to the same uid/gid, and vice-versa, within the same domain.

To use this backend, we have to choose two or more ID ranges:

- a range for the domain we are joining
- another range to serve as a “catch all”, which will store mappings for users and groups that might come from other domains, as well as the default built-in entries

Let’s analyse an example configuration:

```
[global]
    ...
    security = ads
    realm = EXAMPLE.INTERNAL
    workgroup = EXAMPLE
    ...
    idmap config * : backend       = tdb
    idmap config * : range         = 100000 - 199999

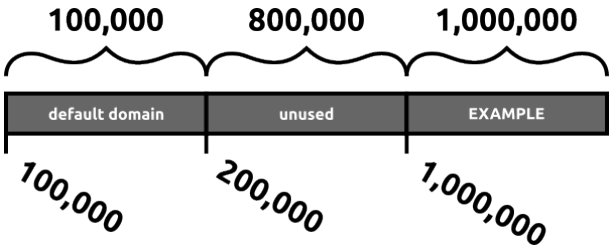
    idmap config EXAMPLE : backend = rid
    idmap config EXAMPLE : range   = 1000000 - 1999999
```

**Note**

This is not yet a complete configuration file, and is just illustrating the idmap configuration. More is needed to join an Active Directory domain.

This configuration is using two idmap backends, and carving out two ranges:

- * domain, or "default domain": any SID that is not mapped via another more specific idmap configuration will use this backend. Since this mapping is not deterministic, a database is needed to keep a record, hence the `tdb` backend is used.
- `EXAMPLE` domain: uses the *rid* idmap backend, and users from the `EXAMPLE` domain will be allocated IDs in the range of 1.000.000 to 1.999.999, that is, there is space for 1 million IDs. Since the mapping is deterministic, there is no need for a database.




**Important**

Planning a range of IDs to be used for the mapping critically important. Such a range can never be reduced, just expanded (carefully!), and it must **NEVER** overlap with another allocated range.

Once this system is joined to the `EXAMPLE.INTERNAL` domain, users from that domain will be allocated corresponding Linux uids and gids from the specified range in a deterministic way, following a formula. As long as the above configuration is used in all Ubuntu systems joined to the domain, the same Active Directory user will always get the same Linux IDs in all those systems.

Things start to get more complicated if the `EXAMPLE.INTERNAL` domain establishes a trust relationship with another Active Directory domain. The correct way to handle this is to, *before*, add a new idmap configuration for that domain. For example:

```
[global]
    ...
    security = ads
    realm = EXAMPLE.INTERNAL
    workgroup = EXAMPLE
    ...
    idmap config * : backend       = tdb
    idmap config * : range         = 100000 - 199999

    idmap config EXAMPLE : backend = rid
    idmap config EXAMPLE : range   = 1000000 - 1999999
```

(continues on next page)

(continued from previous page)

```
    idmap config COMPANY : backend = rid
    idmap config COMPANY : range   = 2000000 - 2999999
```

This change is allocating a new range for the new `COMPANY` domain. Then, when the domain trust relationship is established between the Active Directory domains, the Ubuntu systems with this extra idmap configuration will know that users from the `COMPANY` belong to the range `2000000 - 2999999`.

If, however, the domain trust relationship is established between `EXAMPLE` and `COMPANY` before the new idmap range is configured, then the users and groups from `COMPANY` will have their ID allocations taken from the default domain "*", which is NOT deterministic and is done on a first come, first serve, basis. This means that the **same** Active Directory user from domain `COMPANY` connecting to different Ubuntu systems will likely get a **different** Linux ID.

### Pros and Cons of the rid backend

We now have enough information to understand the pros and cons of the `idmap_rid` backend, and which scenarios are a better fit for it.

Pros:

- Stable mapping of SIDs to IDs within a single domain: all Ubuntu systems sharing the same configuration will arrive at the same mapping for Active Directory users.

Cons:

- extra config for trusted domains: you must add *idmap config* entries for all trusted domains, and deploy these changes to all joined systems before the domains establish a new trust relationship, or else you risk having users of the new trusted domain be allocated IDs from the default backend ("*") range.

With that in mind, `idmap_rid` is best used in the following scenarios:

- Single domain with no trust relationships
- If there are trust relationships, they are fairly static, and well planned in advance, and there is a configuration file management system in place to easily update the `smb.conf` config file with the new idmap config lines across all joined systems.
- Stability of Linux IDs across multiple joined systems is important. For example, NFSv3 is being used.

Next:

- *The autorid idmap backend*

### The autorid idmap backend

The `idmap_autorid(8)` identity mapping backend, like the `rid` one, also provides an algorithmic mapping between SIDs and Linux IDs, with the advantage that it can also automatically cope with Active Directory deployed across multiple domains. There is no need to pre-allocate ranges for each specific existing or future domain. Some planning is required, though:

- range: the first and last ID that will be allocated on the Linux side by this backend. This includes all possible domains.

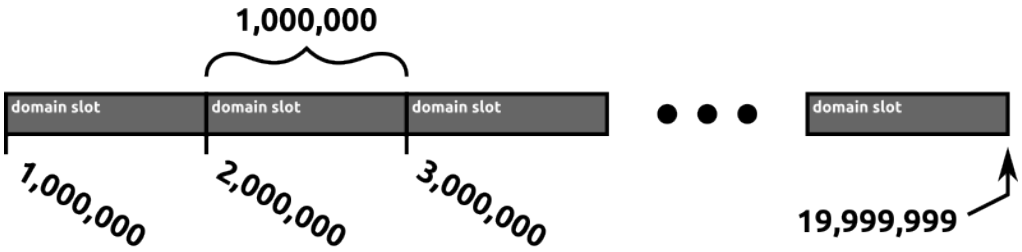
- rangesize: how many IDs to allocate to each domain.

Let's see an example:

```
[global]
    ...
    security = ads
    realm = EXAMPLE.INTERNAL
    workgroup = EXAMPLE
    ...
    idmap config * : backend = autorid
    # 1,000,000 - 19,999,999
    idmap config * : range   = 1000000 - 19999999
    # 1,000,000
    idmap config * : rangesize = 1000000
```

The configuration above gives us 19 domains (or slots) with the capacity of 1 million IDs in each:

- first slot: IDs from `1000000` up to `1999999`
- second slot: IDs from `2000000` up to `2999999`
- …
- 19th slot (last): IDs from `19000000` up to `19999999`




Which domain will get which slot? That is **not deterministic**. It will basically be a first come, first serve. Furthermore, if a domain exhausts the available IDs from a slot, an extension slot will be used, in which case the domain will be using two (possibly non-consecutive even) slots.

This also means that a persistent database is required to record which domain goes into which slot. This is managed automatically by the autorid backend in the `autorid.tdb` file.

> **Note**
>
> The `autorid.tdb` domain mapping database file is kept in `/var/lib/samba/` and should be backed up regularly.

### Pros and Cons of the autorid backend

Let's examine the Pros and Cons of the `idmap_autorid` backend, and which scenarios are a better fit for it.

Pros:

- automatic handling of trusted domains

- simple initial planning, only done once

Cons:

- non-deterministic SID to ID mapping with multiple domains, even if the same idmap config settings are used for all domain-joined systems
- extra concern to backup the domain mapping database file

So when to use the `idmap_autorid` backend?

- Multiple domains are involved via trust relationships, and the stability of IDs is not required on the joined systems.
- Single domain systems can also benefit from the easier and simpler up-front configuration.
- If you need to share files owned by domain users via NFS or another mechanism which relies on stability of IDs, then this backend must not be used.

#### See also

- How-to: *Active Directory integration*

#### See also

- How-to: *Networking section*

## 4.3. Managing software

*Managing software* is an integral part of system maintenance. In this section we discuss the following topics in detail:

- **Software updates**, why updates are sometimes phased, and testing updates before they're released in your production environment
- **Third party repositories**
- **Changing package files**

### 4.3.1. Managing software

In this section we go into detail on package management and best practices.

#### Package sources

- *Third party repository usage* gives some best practices and guidance if you need to use Third Party software

#### Third party repository usage

Ubuntu is an operating system with thousands of packages and snaps available to its users, but it is humanly (and sometimes technically!) impossible to make all software out there available in the official repositories. There are situations where you may want to install a package that is not maintained by Ubuntu, but *is* maintained by a third party entity.

## Why not use third party software?

While having access to the software you want to use is great, it is crucial to understand the risks involved in using third party software - whether it's an individual deb package, or an APT repository. Although this page will focus on third party APT repositories, the same risks are inherent in third party packages as well.

Although we don't recommend using third party software, we know that users sometimes have no other option – so let's take a look at some of the pitfalls, alternatives, and mitigations.

### Security risk

When using any software that you have not audited yourself, you must implicitly trust the publisher of that software with your data. However, with third party APT repositories, there are additional implications of this that are less obvious.

Unlike more modern packaging systems, APT repositories run code that is not sandboxed. When using software from more than one publisher, such as from your distribution as well as a third party, APT and *dpkg* provide no security boundary between them.

This is important because in addition to trusting the publisher's intentions, you are also implicitly trusting the quality and competence of the publisher's own information security, since an adversary can compromise your system indirectly by compromising the software publisher's infrastructure.

For example, consider users who use applications such as games where system security isn't much of a concern, but also use their computers for something more security-sensitive such as online banking. A properly sandboxed packaging system would mitigate an adversary compromising the game publisher in order to take over their users' online banking sessions, since the games wouldn't have access to those sessions. But with APT repositories, the game can access your online banking session as well. Your system's security – as a whole – has been downgraded to the level of the app publisher that has the worst security; they may not consider their information security important because they aren't a bank.

### System integrity

Even if you are certain that the third party APT repository can be trusted, you also need to take into account possible conflicts that having an external package may bring to your system. Some third party packagers – but not all – are careful to integrate their packages into Ubuntu in a way that they don't conflict with official packages from the distribution, but it is technically impossible to predict future changes that might happen in future Ubuntu releases. This means that fundamentally there always is the possibility of conflict. The most common cause of system upgrade failure is the use of third party repositories that worked at the time but later conflicted with a subsequent upgrade.

One of the most common conflicts occurs when a third party package ships with a file that is also shipped by an official Ubuntu package. In this case, having both packages installed simultaneously is impossible because dpkg will prevent managed files from being overwritten. Another possible (and more subtle) issue can happen when the third party software interacts in a problematic way with an official package from Ubuntu. This can be harder to diagnose and might cause more serious problems in the system, such as data loss and service unavailability.

As a general rule, if the third party package you are installing is interacting with or is a modified version of an existing Ubuntu package, you need to be more careful and do some pre-

liminary research before using it in your system.

### Lack of official Ubuntu support

If you decide to install a third party package on your Ubuntu system, the Ubuntu community will struggle to offer support for whatever failures you may encounter as a consequence, since it is out of their control and they are unlikely to be familiar with it. In fact, if you experience a bug in an official Ubuntu package but it is later determined that the bug was caused by a third party package, the Ubuntu community may not be able to help you.

In other words, if you use a third party software you will have to contact its packagers for help if you experience any problem with it.

### A better solution to third party APT repositories: snaps

As we have seen, third party APT repositories are not simple and should be handled carefully. But there is an alternative that is natively supported by Ubuntu and solves some of the issues affecting third party APT repositories: snaps.

Due to the way they are architected, snaps already carry all of their dependencies inside them. When they are installed, they are placed in an isolated directory in the system, which means that they cannot conflict with existing Ubuntu packages (or even with other snaps).

When executed, a snap application is sandboxed and has limited access to the system resources. While still vulnerable to some security threats, snaps offer a better isolation than third party APT repositories when it comes to the damage that can be done by an application.

Finally, if a snap is published in the snapstore, you will not need to go through the hassle of modifying `sources.list` or adding a new *GPG* key to the keyring. Everything will work "out of the box" when you run `snap install`.

### Mitigating the risks

If the software you want is not available as a snap, you may still need to use a third party APT repository. In that case, there are some mitigating steps you can take to help protect your system.

### Security risk mitigation

- If the package you want to install is Free Software/Open Source, then the risk can be reduced by carefully examining the source code of the entire software, including the packaging parts. The amount of work required to do this assessment will depend on the size and complexity of the software, and is something that needs to be performed by an expert whenever an update is available. Realistically, this kind of evaluation almost never happens due to the efforts and time required.
- The availability and cadence of fixes to security vulnerabilities should also be taken into account when assessing the quality and reliability of the third party APT repository. It is important to determine whether these fixes are covered by the third party entity, and how soon they are released once they have been disclosed.
- In addition, you must ensure that the packages are cryptographically signed with the repository's GPG key. This requirement helps to confirm the integrity of the package you are about to install on your system.

### System integrity mitigation

- Avoid release upgrades whenever possible, favouring redeployment onto a newer release instead. Third party APT repositories will often break at release time, and the only way to avoid this is to wait until the maintainers of the repository have upgraded the software to be compatible with the release.
- Configure pinning (we show how to do this below). Pinning is a way to assign a preference level to some (or all) packages from a certain source; in this particular case, the intention is to reduce the preference of packages provided by an external repository so that official Ubuntu packages are not overwritten by mistake.

### Dealing with third party APT repositories in Ubuntu

Now that we have discussed the risks and mitigations of using third party APT repositories, let's take a look at how we can work with them in Ubuntu. Unless otherwise noted, all commands below are to be executed as the `root` user (or using `sudo` with your regular user).

### Add the repository

Several third party entities provide their own instructions on how to add their repositories to a system, but more often than not they don't follow best practices when doing so.

### Fetch the GPG key

The first step before adding a third party APT repository to your system is to fetch the GPG key for it. This key must be obtained from the third party entity; it should be available at the root of the repository's URL, but you might need to contact them and ask for the key file.

Although several third party guides instruct the user to use `apt-key` in order to add the GPG key to `apt`'s keyring, this is no longer recommended. Instead, you should explicitly list the key in the `sources.list` entry by using the `signed-by` option (see below).

Third party APT repositories should also provide a special package called `REPONAME-archive-keyring` whose purpose is to provide updates to the GPG key used to sign the archive. Because this package is signed using the GPG key that is not present in the system when we are initially configuring the repository, we need to manually download and put it in the right place the first time. Assuming that `REPONAME` is `externalrepo`, something like the following should work:

```
wget -O /usr/share/keyrings/externalrepo-archive-keyring.pgp https://
thirdpartyrepo.com/ubuntu/externalrepo-archive-keyring.pgp
```

### Sources.list entry

To add a third party APT repository to your system, you will need to create a file under `/etc/apt/sources.list.d/` with information about the external archive. This file is usually named after the repository (in our example, `externalrepo`). There are two standards the file can follow:

- A one-line entry, which is the most common. In this case, the extension of the file should be `.list`.

- The deb822 format, which is more descriptive but less common. In this case, the extension of the file should be .sources.

An example of a one-line entry would be the following:

```
deb [signed-by=/usr/share/keyrings/externalrepo-archive-keyring.pgp] https://
thirdpartyrepo.com/ubuntu/ jammy main
```

An example of a deb822 file for the same case would be the following:

```
Types: deb
URIs: https://thirdpartyrepo.com/ubuntu
Suites: jammy
Components: main
Signed-By: /usr/share/keyrings/externalrepo-archive-keyring.pgp
```

There are cases when the third party APT repository may be served using HTTPS, in which case you will also need to install the apt-transport-https package.

After adding the repository information, you need to run apt update in order to install the third party packages. Also, now that you have everything configured you should be able to install the externalrepo-archive-keyring package to automate the update of the GPG key.

### Configure pinning for the repository

In order to increase the security of your system and to prevent the conflict issues discussed in the "System integrity" section, we recommend that you configure pinning for the third party APT repository.

You can configure this preference level by creating a file under /etc/apt/preferences.d/ that is usually named after the repository name (externalrepo in this case).

In our example, a file named /etc/apt/preferences.d/externalrepo should be created with the following contents:

```
Package: *
Pin: origin thirdpartyrepo.com
Pin-Priority: 100
```

There are several levels of pinning you can choose here; the Debian Reference guide has good documentation about the topic. The level 100 used above means that users will be able to install packages from the repository and that automatic package upgrades are also enabled. If you want to be able to install packages but don't want them to be considered for automatic upgrades, you should use the level 1.

### How to remove a repository

If you have enabled a third party APT repository but found yourself in a situation where you would like to remove it from the system, there are a few steps you need to take to make sure that the third party packages are also uninstalled.

The first step is to remove the files created in the steps above. These are:

- The sources.list file, under /etc/apt/sources.list.d/.
- The package pinning preference, under /etc/apt/preferences.d/.

- If the third party APT repository does not provide the GPG key in a package, then you can also remove it manually from `/usr/share/keyrings/`.

Before you run `apt update`, you might want to also remove the third party packages that were installed from the repository. The following one-liner will list all those packages:

```
apt remove --purge \
    $(grep "^Package: " /var/lib/apt/lists/#<SELECT_THE_FILE_FOR_YOUR_REPOSITORY>
#_*_Packages \
        | cut -d " " -f2 | sort -u | \
        xargs dpkg-query -W -f='${binary:Package}\t${db:Status-Abbrev}\n' 2> /dev/
null | \
        awk '/\tii $/{print $1}')
```

Make sure to replace `#<SELECT_THE_FILE_FOR_YOUR_REPOSITORY>#` with the right file for the third party APT repository.

After that, you can safely run `apt update`.

### A special case: Ubuntu PPAs

Ubuntu PPAs can be considered as a special case of third party APT repositories. In fact, there are upstream projects that choose to ship their software through PPAs because of the existing tooling that allows users to easily add them to their Ubuntu systems.

It is important to mention that the same points raised above regarding security, system integrity and lack of official Ubuntu support also apply to PPAs.

If you would like to install packages from a PPA, first you will need to add it to your system. For that, you can use the `add-apt-repository` command. Suppose you want to add a PPA from user `thirdparty` named `externalrepo`. You can run:

```
add-apt-repository ppa:thirdparty/externalrepo
```

This command will automatically set up the GPG key, as discussed above. After that, you can run `apt update` and install the third party packages provided by the PPA. Note that `add-apt-repository` will not adjust the repository pinning, so it is recommended that you go through that process manually.

If you decide you do not want to use the PPA anymore and would like to remove it (and its packages) from your system, the easiest way to do it is by installing the `ppa-purge` package. You can then execute it and provide the PPA reference as its argument. In our example, that would be:

```
ppa-purge ppa:thirdparty/externalrepo
```

### Configuration

There are many different ways to configure the software on your machine.

- *Changing package files*
- *Configuration managers*

### Changing package files

Many packages in Ubuntu will create extra files when installed. These files can contain metadata, configurations, rules for operating system interaction, and so on. In many cases, these files will be fully managed by updates to a package, leading to issues when they are modified manually. This page goes over some methods for changing the behavior of a package without causing conflicts in maintained files.

### Configuration files

Configuration files are often provided by packages. They come in many forms, but the majority can be found in the `/etc/` directory with either a `.conf` or `.cnf` extension. Most of the time, these files are managed by the package and editing them could lead to a conflict when updating. To get around this, packages will check in additional `<config>.d/` directories where you can place personal changes.

For example, if you would like `mysql-server` to run on port 3307 instead of 3306, you can open the file `/etc/mysql/mysql.conf.d/mysqld.cnf`, and edit the port option.

```
[mysqld]
#
# * Basic Settings
#
user            = mysql
# pid-file      = /var/run/mysqld/mysqld.pid
# socket        = /var/run/mysqld/mysqld.sock
port            = 3307
```

> **Note**
>
> Some packages do not automatically create files for you to edit in their `.d` directories. In these cases it is often acceptable to just create an additional config file by any name there. When in doubt, check the package's documentation to confirm.

After saving the file, restart the service.

```
systemctl restart mysql
```

The `netstat` command shows that this was successful:

```
netstat -tunpevaW | grep -i 3307
tcp        0      0 127.0.0.1:3307          0.0.0.0:*               LISTEN
106        416022     1730/mysqld
```

### Systemd files

Many packages ship service unit files for interacting with Systemd. Unit files allow packages to define background tasks, initialisation behaviour, and interactions with the operating system. The files, or symlinks of them, will automatically be placed in the `/lib/systemd/system/` directory. Likewise, the files can also show up in `/etc/systemd/system`. If these are edited manually they can cause major issues when updating or even running in general.

Instead, if you would like to modify a unit file, do so through Systemd. It provides the command `systemctl edit <service>` which creates an override file and brings up a text editor for you to edit it.

For example, if you want to edit Apache2 such that it restarts after a failure instead of just when it aborts, you can run the following:

```
sudo systemctl edit apache2
```

This will open a text editor containing:

```
### Editing /etc/systemd/system/apache2.service.d/override.conf
### Anything between here and the comment below will become the new contents of
the file



### Lines below this comment will be discarded

### /lib/systemd/system/apache2.service
# [Unit]
# Description=The Apache HTTP Server
# After=network.target remote-fs.target nss-lookup.target
# Documentation=https://httpd.apache.org/docs/2.4/
#
# [Service]
# Type=forking
# Environment=APACHE_STARTED_BY_SYSTEMD=true
# ExecStart=/usr/sbin/apachectl start
# ExecStop=/usr/sbin/apachectl graceful-stop
# ExecReload=/usr/sbin/apachectl graceful
# KillMode=mixed
# PrivateTmp=true
# Restart=on-abort
...
```

Override the on-abort option by adding a new line in the designated edit location.

```
### Editing /etc/systemd/system/apache2.service.d/override.conf
### Anything between here and the comment below will become the new contents of
the file

[Service]
Restart=on-failure

### Lines below this comment will be discarded
...
```

**Note**

Some options, such as `ExecStart` are additive. If you would like to fully override them add an extra line that clears it (e.g. `ExecStart=`) before providing new options. See Systemd's

man page for more information.

Once the changes are saved, the override file will be created in `/etc/systemd/system/apache2.service.d/override.conf`. To apply changes, run

```
sudo systemctl daemon-reload
```

To verify the change was successful, you can run the status command.

```
systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; preset:
enabled)
    Drop-In: /etc/systemd/system/apache2.service.d
             └─override.conf
             /run/systemd/system/service.d
             └─zzz-lxc-service.conf
     Active: active (running) since Fri 2023-02-17 16:39:22 UTC; 27min ago
       Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 4735 (apache2)
      Tasks: 55 (limit: 76934)
     Memory: 6.5M
        CPU: 65ms
     CGroup: /system.slice/apache2.service
             ├─4735 /usr/sbin/apache2 -k start
             ├─4736 /usr/sbin/apache2 -k start
             └─4737 /usr/sbin/apache2 -k start
...
```

### AppArmor

Packages that use AppArmor will install AppArmor profiles in the `/etc/apparmor.d/` directory. These files are often named after the process being protected, such as `usr.bin.firefox` and `usr.sbin.libvirtd`.

When these files are modified manually, it can lead to a conflict during updates. This will show up in `apt` with something like:

```
Configuration file '/etc/apparmor.d/usr.bin.swtpm'
 ==> Modified (by you or by a script) since installation.
 ==> Package distributor has shipped an updated version.
   What would you like to do about it ?  Your options are:
    Y or I  : install the package maintainer's version
    N or O  : keep your currently-installed version
      D     : show the differences between the versions
      Z     : start a shell to examine the situation
 The default action is to keep your current version.
*** usr.bin.swtpm (Y/I/N/O/D/Z) [default=N] ?
```

Updating to the maintainer's version will override your changes, which could cause problems with your setup. However, using your version could cause security issues.

If you would like to modify these rules to provide the application with additional permissions, you can instead update the local profile, most often found in `/etc/apparmor.d/local/`.

For example, if you would like `swtpm` to access a custom directory called `/var/customtpm`, you can append the following line to `/etc/apparmor.d/local/usr.bin.swtpm`:

```
/var/customtpm/** rwk,
```

This method will work for all AppArmor syntax.

> **Note**
>
> Although most local profiles have the same name as the maintainer's, you can often check what file is included based on the main profile's contents. In `swtpm`'s case, `/etc/apparmor.d/usr.bin.swtpm` contains the lines:
>
> ```
> # Site-specific additions and overrides. See local/README for details.
> #include <local/usr.bin.swtpm>
> ```
>
> showing that the local profile is located at `/etc/apparmor.d/local/usr.bin.swtpm`

### Restoring configuration files

Since config files are meant to be intentional changes by the user/admin, they are not overwritten by updates or even re-installs of the package. However, it's possible you might change it by accident or may just want to go back to step one of a trial-and-error phase that you are in. In those situations you can use `apt` to restore the original config files. Note that while we call `apt`, it is *dpkg* that actually handles the restoration.

If you have a particular config file, like in the example `/etc/rsyslog.conf`, you first want to find out which package owns that config file:

```
$ dpkg -S /etc/rsyslog.conf

rsyslog: /etc/rsyslog.conf
```

So we now know that the package `rsyslog` owns the config file `/etc/rsyslog.conf`. This command just queries package metadata and works even if the file has been deleted.

```
$ rm /etc/rsyslog.conf
$ dpkg -S /etc/rsyslog.conf

rsyslog: /etc/rsyslog.con
```

To restore that file you can re-install the package, telling `dpkg` to bring any missing files back. To do so you pass `dpkg` options through `apt` using `-o Dpkg::Options::="` and then set `--force-...` depending on what action you want. For example:

```
$ sudo apt install --reinstall -o Dpkg::Options::="--force-confmiss" rsyslog
...
Preparing to unpack .../rsyslog_8.2302.0-1ubuntu3_amd64.deb ...
Unpacking rsyslog (8.2302.0-1ubuntu3) over (8.2302.0-1ubuntu3) ...
```

(continues on next page)

(continued from previous page)

```
Setting up rsyslog (8.2302.0-1ubuntu3) ...

Configuration file '/etc/rsyslog.conf', does not exist on system.
Installing new config file as you requested.
```

More details on these options can be found in the `dpkg(1)` manual page, but the most common and important ones are:

- `confmiss` Always install the missing conffile without prompting.
- `confnew` If a conffile has been modified and the version in the package changed, always install the new version without prompting.
- `confold` If a conffile has been modified and the version in the package changed, always keep the old version without prompting.
- `confdef` If a conffile has been modified and the version in the package changed, always choose the default action without prompting.
- `confask` If a conffile has been modified, always offer to replace it with the version in the package, even if the version in the package did not change.

So in the case of an accidental bad config entry, if you want to go back to the package default you could use `--force-confask` to check the difference and consider restoring the content.

```
$ echo badentry >> /etc/rsyslog.conf
$ sudo apt install --reinstall -o Dpkg::Options::="--force-confask" rsyslog
...
Preparing to unpack .../rsyslog_8.2302.0-1ubuntu3_amd64.deb ...
Unpacking rsyslog (8.2302.0-1ubuntu3) over (8.2302.0-1ubuntu3) ...
Setting up rsyslog (8.2302.0-1ubuntu3) ...

Configuration file '/etc/rsyslog.conf'
 ==> Modified (by you or by a script) since installation.
     Version in package is the same as at last installation.
   What would you like to do about it ?  Your options are:
    Y or I  : install the package maintainer's version
    N or O  : keep your currently-installed version
      D     : show the differences between the versions
      Z     : start a shell to examine the situation
 The default action is to keep your current version.
*** rsyslog.conf (Y/I/N/O/D/Z) [default=N] ? D
--- /etc/rsyslog.conf   2023-04-18 07:11:50.427040350 +0000
+++ /etc/rsyslog.conf.dpkg-new  2023-02-23 16:58:03.000000000 +0000
@@ -51,4 +51,3 @@
 # Include all config files in /etc/rsyslog.d/
 #
 $IncludeConfig /etc/rsyslog.d/*.conf
-badentry

Configuration file '/etc/rsyslog.conf'
 ==> Modified (by you or by a script) since installation.
```

(continues on next page)

(continued from previous page)

```
    Version in package is the same as at last installation.
  What would you like to do about it ?  Your options are:
   Y or I  : install the package maintainer's version
   N or O  : keep your currently-installed version
     D     : show the differences between the versions
     Z     : start a shell to examine the situation
 The default action is to keep your current version.
*** rsyslog.conf (Y/I/N/O/D/Z) [default=N] ? y
Installing new version of config file /etc/rsyslog.conf ...
```

The same can be used if you removed a whole directory by accident, to detect and re-install all related packages config files.

```
$ rm -rf /etc/vim
$ dpkg -S /etc/vim
vim-common, vim-tiny: /etc/vim
$ sudo apt install --reinstall -o Dpkg::Options::="--force-confmiss" vim-common
vim-tiny
...
Configuration file '/etc/vim/vimrc', does not exist on system.
Installing new config file as you requested.
...
Configuration file '/etc/vim/vimrc.tiny', does not exist on system.
Installing new config file as you requested.
```

### Configuration managers

There are several configuration management tools that can help manage IT infrastructure, typically through automating configuration and deployment processes. The most popular options are Ansible, Chef, and Puppet.

Although they can be complex to set up initially, they have a clear advantage in environments where scalability and consistency are important. All three of these tools are available from the Universe repository. This page will give a short overview of each of them, along with their suggested use cases.

### Ansible

*Ansible* remains one of the most popular options due to its simplicity. It uses SSH to orchestrate nodes rather than through an installed agent. It is therefore most often used in cases where an agentless architecture is desired, or in small to medium-sized environments where the use of easy-to-write YAML playbooks can reduce the steepness of the initial learning curve, and requirements may be less complex.

- **Language**: YAML (for playbooks) and Python.

### Chef

Chef is a flexible tool that can run either in a client-server configuration, or in "chef-solo" mode. It integrates with cloud-based platforms such as EC2 and Azure, and is often used to streamline a company's servers (in both large or small-scale environments). It has an active community that contributes "cookbooks", which are collections of "recipes" that control individual tasks. This can help to alleviate some of the complexity in using the tool.

- **Language**: Ruby.

### Puppet

Puppet uses a client-server architecture; the Puppet server (the "master") is installed one one or more servers, and the client (Puppet Agent) is installed on every machine Puppet is to manage. It's most often used to manage IT infrastructure lifecycles; although it can be complicated to set up, it is useful in particularly complex or large-scale environments where detailed reporting is desired.

- **Language**: Puppet domain-specific language (DSL), based on Ruby.

### Further reading

- For a comparison of all open source configuration management software, refer to this Wikipedia table.

### Updates

- *About apt upgrade and phased updates*; you may occasionally see messages about phased updates.
- *Advance testing of updates in best practice server deployments* helps you to avoid updates being rolled out on your production systems without them being tested first.

### About apt upgrade and phased updates

You may have noticed recently that updating your system with `apt upgrade` sometimes produces a weird message about packages being kept back…like this one:

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following packages have been kept back:
  (Names of <X> held back packages listed here)
0 upgraded, 0 newly installed, 0 to remove and <X> not upgraded.
```

If you've ever used combinations of packages from different releases or third party repos, you may be familiar with this message already. However, it has become a much more common occurrence due to something called "phased updates".

### What are phased updates?

Phased updates are software updates that are rolled out in stages, rather than being provided to everyone at the same time. Initially, the update is provided only to a small subset of Ubuntu machines. As the update proves to be stable, it is provided to an ever-increasing number of users until everyone has received it (i.e., when the update is "fully phased").

The good news is, you don't need to do anything about the "packages kept back" message – you can safely ignore it. Once the update has been deemed safe for release, you will receive the update automatically.

### Why is Ubuntu doing this?

Although updates are thoroughly tested before they get released at all, sometimes bugs can be hidden well enough to escape our attention and make it into a release – especially in highly specific use cases that we didn't know we needed to test. This can obviously cause problems for our users, and used to be the norm before we phased updates through `apt`.

Update phasing makes it much easier for us to detect serious breakages early on – before they have a chance to cause problems for the majority of our users. It gives us the opportunity to hold back the update until the bugs are fixed.

In other words, it directly benefits our users by increasing the safety, stability and reliability of Ubuntu.

The phasing system makes it so that different sets of users are chosen to be the first to get the updates, so that there isn't one group of unlucky people who always get potentially broken updates soon after release.

> **Note**
>
> It should be mentioned here that security updates are *never* phased.

### Can I turn off phased updates?

That depends on how stable you need your system to be. If you just want to avoid any notices about packages being held back during `apt` updates, and you're willing to be one of the first people to get updates whenever they're released, you can turn off phased updates. Be warned, though – if an update *is* broken, you will almost always be in the first set of people to get it (i.e., you're basically volunteering yourself as a guinea pig for the early update releases!). It will get rid of the "held back packages" in `apt` message, though.

If that doesn't sound like something you want, leave phased updates on (this is the default). You will still temporarily get the "held back packages" message, but your machine will be more protected from updates that might otherwise break it – and once the packages are ready to be safely installed on your system, they will no longer be held back.

### Can I `apt upgrade` the individual packages? (and should I?)

While you can *technically* get around phased updates by running `apt install` on individual held back packages, it's not recommended. You're unlikely to break your machine by doing this – as long as the package is being held back due to update phasing.

If you want to `apt upgrade` a package, you should first carefully examine the proposed changes that `apt` would make before you proceed. If the package update was kept back for a reason unrelated to phasing, `apt` may be forced to remove packages in order to complete your request, which could then cause problems elsewhere.

### How do I turn off phased updates?

If you're sure that you want to disable phased updates, reverting to the old behaviour, you can change `apt`'s configuration by creating a file in `/etc/apt/apt.conf.d` called `99-Phased-Updates` (if `/etc/apt/apt.conf.d/99-Phased-Updates` doesn't already exist). In the file, simply add the following lines:

```
Update-Manager::Always-Include-Phased-Updates true;
APT::Get::Always-Include-Phased-Updates true;
```

Again, please only do this if you really know what you're doing and are absolutely sure you need to do it (for instance, if you are intentionally installing all the latest packages to help test them – and don't mind if your system breaks). We definitely don't recommend turning off phased updates if you're a newer user.

### Why is this suddenly happening now?

Phased updates have been part of the update-manager on Ubuntu Desktop for quite a while (since 13.04, in fact!), but were implemented in APT in 21.04. It now works on all versions of Ubuntu (including Ubuntu Server, Raspberry Pi, and containers). Since this includes the 22.04 LTS, it's now getting a lot more attention as a result!

### How does it actually work?

Phased updates depend on a value derived from your machine's "Machine ID", as well as the package name and package version. The neat thing about this is that phasing is determined completely at the client end; no identifiable information (or indeed any new information at all) is ever sent to the server to achieve update phasing.

When the software update is released, the initial subset of machines to receive the update first is chosen at random. Only if there are no problems detected by the first set of users will the update be made available to everyone.

For more detailed information, including about how changes to phasing are timed, you can check the Ubuntu wiki page on phased updates.

### How can I find out more information about currently phased packages?

You can find out the phasing details of a package by using the `apt policy` command:

```
apt policy <package>
```

For example, at the time of writing, the package `libglapi-mesa` has a phased update. Running `apt policy libglapi-mesa` then produces an output like this:

```
libglapi-mesa:
  Installed: 22.0.5-0ubuntu0.3
```

(continues on next page)

(continued from previous page)

```
 Candidate: 22.2.5-0ubuntu0.1~22.04.1
 Version table:
        22.2.5-0ubuntu0.1~22.04.1 500 (phased 20%)
           500 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages
*** 22.0.5-0ubuntu0.3 100
           100 /var/lib/dpkg/status
        22.0.1-1ubuntu2 500
           500 http://archive.ubuntu.com/ubuntu jammy/main amd64 Packages
```

In this output you can see that this package is 20% phased.

You can see the status of all packages currently being phased in Ubuntu at https://people.canonical.com/~ubuntu-archive/phased-updates.html

### Further reading

- The details in this page are based on this excellent post on AskUbuntu by AskUbuntu user *ArrayBolt3*. This page is therefore licensed under Creative Commons Attribution-ShareAlike license, distributed under the terms of CC BY-SA 4.0
- You can check on the progress of the current phasing Ubuntu Stable Release Updates.
- There is also more detail on how phased updates work in the Ubuntu wiki, the Error Tracker, and the `apt preferences` manpage.

### Advance testing of updates in best-practice server deployments

If you manage production server deployments with demanding reliability needs, you probably follow best practices with automated testing of Ubuntu updates in your environment before you deploy them. If an update causes a problem for you, you can hold it back from your production deployments while the problem is resolved.

However, if a problem is identified with a package update *after* release, holding it back in a deployment also holds back security updates for that package. Fixing the issue can also take longer because we also have to consider users who *have* upgraded. Ideally, we would like to identify problems with an update before release, which would allow us to fix it more rapidly, keeping deployments more reliable and secure.

We usually make the final builds of our proposed updates publicly available for at least a week before we publish them. We therefore encourage you to test these proposed updates before we release them so that if you discover an unanticipated problem we can delay the release until the issue is fixed.

So, if you already have automated tests for your deployment, please help us to help you by also running your testing against our proposed updates!

### Information about proposed updates

The status of our proposed update pipeline is available on the Pending SRU page. This displays the packages with prepared updates, and links to the bugs they are intended to fix.

### Testing a single proposed update

Calls for testing are made on individual bugs that are being fixed by a proposed update. You can enable proposed testing as follows.

On Ubuntu 24.04 and later, run `sudo add-apt-repository -yp proposed`. This makes proposed updates visible to APT, but will not install them by default.

On releases prior to Ubuntu 23.04, in addition to `sudo add-apt-repository -yp proposed` it is also necessary to apply a "pin" to APT so that it does not attempt to automatically upgrade all packages to -proposed. See the Enable proposed wiki page for instructions.

Once enabled, you can upgrade just the package `foo` and its necessary dependencies to a proposed update using, for example, `sudo apt-get install -t noble-proposed foo` for 24.04 "Noble Numbat".

After the appropriate packages are upgraded to their proposed versions, you can test your deployment as normal.

### Test all proposed updates together

**Warning:** in the general case, upgrading to "everything" in proposed is not considered safe. Dependencies may not be fully resolvable while an update is still being prepared, and this may lead to APT mistakenly removing packages to make everything resolved. For this reason, this method is not generally recommended.

Further, some packages in proposed may already have been reported to cause a regression and been left there pending a fix, in which case your time would be wasted on duplicate regression reports.

However, in the specific case of best practice automated testing of server deployments, we can assume that the testing is taking place in a sandboxed environment, so this caveat does not cause any damage. You may see intermittent CI failures, however, so we suggest only taking heed of a failure if it persists for more than a day or so. This still leaves plenty of time to alert us to a problem!

To upgrade to all of proposed, enable proposed as described above, then run (for example) `apt-get upgrade -t noble-proposed` for 24.04 "Noble Numbat".

After the appropriate packages are upgraded to their proposed versions, you can test your deployment as normal.

After testing, the test environment should be considered "spent" and not be used again. Instead, re-create and re-upgrade to proposed for any subsequent test run. This prevents proposed updates that were not ultimately released from accidentally being included, and minimises occurrences of the "inconsistent pocket" problem as described above.

### How to report a regression

- File a bug against the package if a suitable bug report doesn't already exist.
- In your report, please specify the version that worked and the version that does not. The output of `apt policy some-package` is helpful to identify this.
- If the regression is in an update that was already released, please tag the bug `regression-update`.

- If the regression is caused by an update that was proposed to be released, but not yet released, please tag the bug `regression-proposed`. Please also identify one of the bugs being fixed by the update from the pending SRU report and note the regression there.

General information on reporting bugs is also available.

#### See also

- How-to: *Managing software*

# 4.4. Storage

*In data and storage* we discuss:

- **Managing data**
- **Storage and backups**

## 4.4.1. Data and storage

The following sections provide details on various topics related to storing, managing and accessing data.

#### Data management

OpenLDAP is a popular implementation of the Lightweight Directory Access Protocol (LDAP), used for managing hierarchical data. It offers a way to store, organise and manage an organisation's data such as employee accounts and computers. It facilitates centralised authentication and authorisation management.

#### Introduction to OpenLDAP

Lightweight Directory Access Protocol (LDAP) is a protocol used for managing hierarchical data. OpenLDAP is the open-source implementation of LDAP used in Ubuntu.

It offers an LDAP server that provides directory services, a client for managing them, and client libraries used by hundreds of applications. OpenLDAP contains some terminology and concepts that new users may want to familiarise themselves with before attempting to set it up. Thanks to its high configurability and flexibility, OpenLDAP can be tailored to suit various needs and is a pertinent choice for those with specific requirements.

LDAP is a protocol for querying and modifying an X.500-based directory service running over TCP/IP. The current LDAP version is LDAPv3, as defined in RFC 4510.

The LDAP protocol *accesses* directories. It's common to refer to a directory as an *LDAP directory* or *LDAP database* as a shorthand – although technically incorrect, this shorthand is so widely used that it's understood as such.

#### Key concepts and terms

- A **directory** is a tree of data **entries** that is hierarchical in nature; it is called the Directory Information Tree (*DIT*).
- An **entry** consists of a set of **attributes**.
- An **attribute** has a **key** (a name or description) and one or more **values**. Every attribute must be defined in at least one `objectClass`.

- Attributes and `objectClasses` are defined in **schemas** (an `objectClass` is considered a special kind of attribute).
- Each entry has a unique identifier: its **Distinguished Name** (*DN* or dn). This, in turn, consists of a **Relative Distinguished Name** (RDN) followed by the parent entry's DN.
- The entry's DN is not an attribute. It is not considered part of the entry itself.

> **Note**
>
> The terms **object**, **container**, and **node** have certain connotations but they all essentially mean the same thing as **entry** (the technically correct term).

For example, below we have a single entry consisting of 11 attributes where the following is true:

- DN is `cn=John Doe,dc=example,dc=com`
- RDN is `cn=John Doe`
- parent DN is `dc=example,dc=com`

```
dn: cn=John Doe,dc=example,dc=com
cn: John Doe
givenName: John
sn: Doe
telephoneNumber: +1 888 555 6789
telephoneNumber: +1 888 555 1232
mail: john@example.com
manager: cn=Larry Smith,dc=example,dc=com
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
objectClass: top
```

The above entry is in **LDAP Data Interchange Format** format (LDIF). Any information that you feed into your DIT must also be in such a format. It is defined in RFC 2849.

A directory accessed via LDAP is good for anything that involves a large number of access requests to a mostly-read, attribute-based (name:value) backend, and that can benefit from a hierarchical structure. Examples include an address book, company directory, a list of email addresses, and a mail server's configuration.

### Our OpenLDAP guide

For users who want to set up OpenLDAP, we recommend following our series of guides in this order:

- *Install and configure LDAP*
- *LDAP Access Control*
- *LDAP users and groups*
- *SSL/TLS*

- *Replication*
- *Backup and restore*

### References

- The OpenLDAP administrators guide
- RFC 4515: LDAP string representation of search filters
- Zytrax's LDAP for Rocket Scientists; a less pedantic but comprehensive treatment of LDAP Older references that might still be useful:
- O'Reilly's LDAP System Administration (textbook; 2003)
- Packt's Mastering OpenLDAP (textbook; 2007)

Databases are also a common data management tool in any setup.

### Introduction to databases

Ubuntu provides two popular database servers. They are MySQL and PostgreSQL.

Both are popular choices among developers, with similar feature sets and performance capabilities. Historically, Postgres tended to be a preferred choice for its attention to standards conformance, features, and extensibility, whereas MySQL may be more preferred for higher performance requirements. However, over time each has made good strides catching up with the other. Specialised needs may make one a better option for a certain application, but in general both are good, strong options.

They are available in the Main repository and equally supported by Ubuntu. In our *how-to section* we have guides that explain how to install and configure these database servers.

- *How to install and configure MySQL*
- *How to install and configure PostgreSQL*

### Storage and backups

- Our *Storage* section contains more detail about LVM and iSCSI
- Our *Multipath* section explains the key concepts and common configuration setups for Device Mapper Multipathing
- *Introduction to backups* discusses various backup strategies to protect your data

### Storage

- *About LVM* provides more detail about Logical Volume Management

### About Logical Volume Management (LVM)

Logical Volume Management, or LVM, provides a method of allocating and managing space on mass-storage devices that is more advanced and flexible than the traditional method of partitioning storage volumes.

To find out how to set up LVM in Ubuntu, *refer to this guide*, which will also show you how to resize and move partitions, and how to create snapshots.

### Key concepts

There are 3 concepts that LVM manages:

- **Physical volumes**: correspond to disks. They represent the lowest abstraction level of LVM, and are used to create a volume group.
- **Volume groups**: are collections of physical volumes. They are pools of disk space that logical volumes can be allocated from.
- **Logical volumes**: correspond to partitions – they usually hold a *filesystem*. Unlike partitions though, they can span multiple disks (because of the way volume groups are organised) and do not have to be physically contiguous.

### Resizing partitions

LVM can expand a partition while it is mounted, if the filesystem used on it also supports that. When expanding a partition, LVM can use free space anywhere in the volume group, even on another disk.

When resizing LVM partitions, and especially when shrinking them, it is important to take the same precautions you would as if you were dealing with regular partitions. Namely, always make a backup of your data before actually executing the commands. Although LVM will try hard to determine whether a partition can be expanded or shrunk before actually performing the operation, there is always the possibility of data loss.

### Moving Partitions

Moving regular partitions is usually only necessary because of the requirement that partitions be physically contiguous, so you probably will not need to do this with LVM. If you do, LVM can move a partition while it is in use, and will not corrupt your data if it is interrupted. In the event that your system crashes or loses power during the move, you can simply restart it after rebooting and it will finish normally. Another reason you might want to move an LVM partition is to replace an old disk with a new, larger one. You can migrate the system to the new disk while using it, and then remove the old one later.

### Snapshots

LVM allows you to freeze an existing logical volume in time, at any moment, even while the system is running. You can continue to use the original volume normally, but the snapshot volume appears to be an image of the original, frozen in time at the moment you created it. You can use this to get a consistent filesystem image to back up, without shutting down the system. You can also use it to save the state of the system, so that you can later return to that state if needed. You can also mount the snapshot volume and make changes to it, without affecting the original.

### See also

- How-to: *Storage*

## Multipath

Device Mapper Multipathing (which we call “Multipath”) allows you to create a single virtual device to aggregate different input/output (I/O) paths between server nodes and storage arrays.

### Introduction

- *Introduction to Device mapper multipathing*

### Configuration

- *Configuration options and overview*
- *Configuration examples*
- *Common tasks and procedures*

### Introduction to device mapper multipathing (“multipath”)

Device mapper multipathing (which will be called **multipath** in this document) allows you to create a virtual device that aggregrates multiple input/output (I/O) paths between server nodes and storage arrays. These I/O paths are physical storage area network (SAN) connections that can include separate cables, switches, and controllers.

### How a multipath device works in practice

If you are not using multipath, your system treats every path from a server node to a storage controller as a separate device, even when the I/O path connects the same server node to the same storage controller. Multipath allows you to organise the I/O paths logically, by creating a single device on top of the underlying paths.

### What identifier does a multipath device use?

Every multipath device has a World Wide Identifier (WWID), which is guaranteed to be globally unique and unchanging. By default, the name of a multipath device is set to its WWID. However, you can force multipath to give each multipath device a name that is a **node-unique alias** of the form `mpathn` by setting the `user_friendly_names` option in `multipath.conf`.

### What does multipath provide?

When you use multipath, it can provide:

- **Redundancy** Multipath provides *failover* in an active/passive configuration. In an active/passive configuration, only half the paths are used at any time for I/O. If any element of an I/O path (the cable, switch, or controller) fails, multipath switches to an alternate path.
- **Improved performance** Multipath can be configured in active/active mode, where I/O is spread over the paths in a round-robin fashion. In some configurations, multipath can detect loading on the I/O paths and dynamically re-balance the load.

### Configuring multipath for a storage system

Before you decide to use multipath, check your storage vendor's installation guide for the multipath configuration variables that are recommended for your storage model. The default multipath configuration will probably work but will likely need adjustments based on your storage setup.

### Multipath components

| Component | Description |
|---|---|
| `dm_multipa` kernel module | Reroutes I/O and supports **failover** for paths and path groups. |
| `multipath` command | Lists and configures multipath devices. While this command is normally started up with `/etc/rc.sysinit`, it can also be started up by a `udev` program whenever a block device is added, or it can be run by the `initramfs` file system. |
| `multipathd` daemon | Monitors paths; as paths fail and come back, it may initiate path group switches. Provides for interactive changes to multipath devices. This daemon must be restarted for any changes to the `/etc/multipath.conf` file to take effect. |
| `kpartx` command | Creates device-mapper devices for the partitions on a device. It is necessary to use this command for DOS-based partitions with multipath. The `kpartx` is provided in its own package, but the `multipath-tools` package depends on it. |

### Multipath setup overview

The multipath setup process is usually simple because it has compiled-in default settings that are suitable for common multipath configurations. The basic procedure for configuring your system with multipath is:

1. Install the `multipath-tools` and `multipath-tools-boot` packages.
2. Create an empty config file called `/etc/multipath.conf`.
3. Edit the `multipath.conf` file to modify default values and save the updated file.
4. Start the multipath daemon.
5. Update initial RAM disk.

For detailed setup instructions for multipath configuration see *DM-Multipath configuration* and *DM-Multipath setup*.

### Example of a multipath device in use

For example, a node with two host bus adapters (*HBAs*) attached to a storage controller, with two ports, via a single un-zoned *FC* switch, sees four devices: `/dev/sda`, `/dev/sdb`, `/dev/sdc`, and `/dev/sdd`. Multipath creates a single device with a unique WWID that reroutes I/O to those four underlying devices according to the multipath configuration.

When the `user_friendly_names` configuration option is set to 'yes', the name of the multipath device is set to `mpathn`. When new devices are brought under the control of multipath,

the new devices may be seen in two different places under the `/dev` directory: `/dev/mapper/mpathn` and `/dev/dm-n`.

- The devices in `/dev/mapper` are created early in the boot process. **Use these devices to access the multipathed devices.**
- Any devices of the form `/dev/dm-n` are for **internal use only** and should never be used directly.

You can also set the name of a multipath device to a name of your choosing by using the `alias` option in the `multipaths` section of the multipath configuration file.

**See also**

For information on the multipath configuration defaults, including the `user_friendly_names` and `alias` configuration options, see *DM-Multipath configuration*.

### Consistent multipath device names in a cluster

When the `user_friendly_names` configuration option is set to 'yes', the name of the multipath device is unique to a node, but it is not guaranteed to be the same on all nodes using the multipath device. Similarly, if you set the `alias` option for a device in the `multipaths` section of `/etc/multipath.conf`, the name is not automatically consistent across all nodes in the cluster.

This should not cause any difficulties if you *use LVM* to create logical devices from the multipath device, but if you require that your multipath device names be consistent in every node it is recommended that you leave the `user_friendly_names` option set to 'no' and that you **do not** configure aliases for the devices.

If you configure an alias for a device that you would like to be consistent across the nodes in the cluster, you should ensure that the `/etc/multipath.conf` file is the same for each node in the cluster by following the same procedure:

1. Configure the aliases for the multipath devices in the in the `multipath.conf` file on one machine.
2. Disable all of your multipath devices on your other machines by running the following commands as root:

```
systemctl stop multipath-tools.service
multipath -F
```

3. Copy the `/etc/multipath.conf` file from the first machine to all other machines in the cluster.
4. Re-enable the `multipathd` daemon on all the other machines in the cluster by running the following command as root:

```
systemctl start multipath-tools.service
```

Whenever you add a new device you will need to repeat this process.

### Multipath device attributes

In addition to the `user_friendly_names` and `alias` options, a multipath device has numerous attributes. You can modify these attributes for a specific multipath device by creating an entry for that device in the `multipaths` section of `/etc/multipath.conf`.

For information on the `multipaths` section of the multipath configuration file, see *DM-Multipath configuration*.

### Multipath devices in logical volumes

After creating multipath devices, you can use the multipath device names just as you would use a physical device name when you are creating an LVM physical volume.

For example, if `/dev/mapper/mpatha` is the name of a multipath device, the following command (run as root) will mark `/dev/mapper/mpatha` as a physical volume:

```
pvcreate /dev/mapper/mpatha
```

You can use the resulting LVM physical device when you create an LVM volume group just as you would use any other LVM physical device.

> **Note**
>
> If you try to create an LVM physical volume on a whole device on which you have configured partitions, the `pvcreate` command will fail.

Once you create an LVM logical volume that has active/passive multipath arrays as the underlying physical devices, you must add filters in the `lvm.conf` file to exclude the disks that underlie the multipath devices. This is because if the array automatically changes the active path to the passive path when it receives I/O, multipath will failover and *fallbacks* whenever LVM scans the passive path if these devices are not filtered.

If an active/passive array requires a command to activate the passive path, LVM will print a warning message. To filter all SCSI devices in the LVM configuration file (`lvm.conf`), include the following filter in the devices section of the file:

```
filter = [ "r/block/", "r/disk/", "r/sd.*/", "a/.*/" ]
```

After updating `/etc/lvm.conf`, it's necessary to update the `initrd` so that this file will be copied there, where the filter matters the most – during boot. Perform:

```
update-initramfs -u -k all
```

> **Note**
>
> Every time either `/etc/lvm.conf` or `/etc/multipath.conf` is updated, the `initrd` should be rebuilt to reflect these changes. This is imperative when *denylists* and filters are necessary to maintain a stable storage configuration.

### Multipath configuration options and overview

It is recommended that you first read the *device mapper multipathing introduction* if you are unfamiliar with the concepts and terms. For consistency, we refer to device mapper multipathing as **multipath**.

Multipath usually works out-of-the-box with most common storages. This doesn't mean the default configuration variables should be used in production: they don't treat important parameters your storage might need.

It's a good idea to consult your storage manufacturer's install guide for the Linux Multipath configuration options. Storage vendors often provide the most adequate options for Linux, including minimal versions required for kernel and multipath-tools.

Default multipath configuration values can be overridden by editing the `/etc/multipath.conf` file and restarting the `multipathd` service. This page provides information on parsing and modifying the `multipath.conf` file.

### Configuration file overview

The `multipath.conf` configuration file contains entries of the form:

```
<section> {
       <attribute> <value>
       ...
       <subsection> {
              <attribute> <value>
              ...
       }
}
```

The following keywords are recognised:

- `defaults`: Defines default values for attributes used whenever no values are given in the appropriate device or multipath sections.
- `blacklist`: Defines which devices should be excluded from the multipath topology discovery.
- `blacklist_exceptions`: Defines which devices should be included in the multipath topology discovery, despite being listed in the blacklist section.
- `multipaths`: Defines the multipath topologies. They are indexed by a World Wide Identifier (WWID). Attributes set in this section take precedence **over all others**.
- `devices`: Defines the device-specific settings. Devices are identified by vendor, product, and revision.
- `overrides`: This section defines values for attributes that should override the device-specific settings for all devices.

### Configuration file defaults

Currently, the multipath configuration file ONLY includes a minor `defaults` section that sets the `user_friendly_names` parameter to 'yes':

```
defaults {
    user_friendly_names yes
}
```

This overwrites the default value of the `user_friendly_names` parameter.

All the multipath attributes that can be set in the `defaults` section of the `multipath.conf` file can be found `in the manual pages` with an explanation of what they mean. The attributes are:

- `verbosity`
- `polling_interval`
- `max_polling_interval`
- `reassign_maps`
- `multipath_dir`
- `path_selector`
- `path_grouping_policy`
- `uid_attrs`
- `uid_attribute`
- `getuid_callout`
- `prio`
- `prio_args`
- `features`
- `path_checker`
- `alias_prefix`
- `failback`
- `rr_min_io`
- `rr_min_io_rq`
- `max_fds`
- `rr_weight`
- `no_path_retry`
- `queue_without_daemon`
- `checker_timeout`
- `flush_on_last_del`
- `user_friendly_names`
- `fast_io_fail_tmo`
- `dev_loss_tmo`
- `bindings_file`

- wwids_file
- prkeys_file
- log_checker_err
- reservation_key
- all_tg_pt
- retain_attached_hw_handler
- detect_prio
- detect_checker
- force_sync
- strict_timing
- deferred_remove
- partition_delimiter
- config_dir
- san_path_err_threshold
- san_path_err_forget_rate
- san_path_err_recovery_time
- marginal_path_double_failed_time
- marginal_path_err_sample_time
- marginal_path_err_rate_threshold
- marginal_path_err_recheck_gap_time
- delay_watch_checks
- delay_wait_checks
- marginal_pathgroups
- find_multipaths
- find_multipaths_timeout
- uxsock_timeout
- retrigger_tries
- retrigger_delay
- missing_uev_wait_timeout
- skip_kpartx
- disable_changed_wwids
- remove_retries
- max_sectors_kb
- ghost_delay

- `enable_foreign`

> **Note**
>
> Previously, the `multipath-tools` project provided a complete configuration file with all the most commonly used options for each of the most-used storage devices. Currently, you can see all those default options by running `sudo multipath -t`. This will dump a used configuration file including all the embedded default options.

### Configuration file blacklist and exceptions

The blacklist section is used to exclude specific devices from the multipath topology. It is most commonly used to exclude local disks, non-multipathed devices, or non-disk devices.

#### By `devnode`

The default blacklist consists of the regular expressions `"^(ram|zram|raw|loop|fd|md|dm-|sr|scd|st|dcssblk)[0-9]"` and `"^(td|hd|vd)[a-z]"`. This causes virtual devices, non-disk devices, and some other device types to be excluded from multipath handling by default.

```
blacklist {
    devnode "^(ram|zram|raw|loop|fd|md|dm-|sr|scd|st|dcssblk)[0-9]"
    devnode "^(td|hd|vd)[a-z]"
    devnode "^cciss!c[0-9]d[0-9]*"
}
```

#### By `wwid`

Regular expression for the World Wide Identifier of a device to be excluded/included

#### By device

Subsection for the device description. This subsection recognises the `vendor` and `product` keywords. Both are regular expressions.

```
device {
    vendor "LENOVO"
    product "Universal Xport"
}
```

#### By property

Regular expression for an udev property. All devices that have matching udev properties will be excluded/included. The handling of the property keyword is special, because devices must have at least one whitelisted udev property; otherwise they're treated as blacklisted, and the message "blacklisted, udev property missing" is displayed in the logs.

### Blacklist by protocol

The protocol strings that multipath recognises are `scsi:fcp`, `scsi:spi`, `scsi:ssa`, `scsi:sbp`, `scsi:srp`, `scsi:iscsi`, `scsi:sas`, `scsi:adt`, `scsi:ata`, `scsi:unspec`, `ccw`, `cciss`, `nvme`, and `undef`. The protocol that a path is using can be viewed by running:

```
multipathd show paths format "%d %P"
```

### Blacklist exceptions

The `blacklist_exceptions` section is used to revert the actions of the blacklist section. This allows one to selectively include ("whitelist") devices which would normally be excluded via the blacklist section.

```
blacklist_exceptions {
    property "(SCSI_IDENT_|ID_WWN)"
}
```

> **Note**
>
> A common use is to blacklist "everything" using a catch-all regular expression, and create specific `blacklist_exceptions` entries for those devices that should be handled by `multipath-tools`.

### Configuration file multipath section

The `multipaths` section allows setting attributes of **multipath maps**. The attributes set via the multipaths section (see list below) take precedence over all other configuration settings, including those from the overrides section.

The only recognised attribute for the multipaths section is the multipath subsection. If there is more than one multipath subsection matching a given WWID, the contents of these sections are merged, and settings from later entries take precedence.

The multipath subsection recognises the following attributes:

- `wwid`: (Mandatory) World Wide Identifier. Detected multipath maps are matched against this attribute. Note that, unlike the `wwid` attribute in the blacklist section, this is not a regular expression or a substring; WWIDs must match exactly inside the multipaths section.
- `alias`: Symbolic name for the multipath map. This takes precedence over an entry for the same WWID in the `bindings_file`.

### Optional attributes

The following attributes are optional; if not set, the default values are taken from the overrides, devices, or defaults section of the `multipath.conf(5)` manual page:

- `path_grouping_policy`
- `path_selector`
- `prio`

- prio_args
- failback
- rr_weight
- no_path_retry
- rr_min_io
- rr_min_io_rq
- flush_on_last_del
- features
- reservation_key
- user_friendly_names
- deferred_remove
- san_path_err_threshold
- san_path_err_forget_rate
- san_path_err_recovery_time
- marginal_path_err_sample_time
- marginal_path_err_rate_threshold
- marginal_path_err_recheck_gap_time
- marginal_path_double_failed_time
- delay_watch_checks
- delay_wait_checks
- skip_kpartx
- max_sectors_kb
- ghost_delay

### Example

```
multipaths {
    multipath {
        wwid                    3600508b4000156d700012000000b0000
        alias                   yellow
        path_grouping_policy    multibus
        path_selector           "round-robin 0"
        failback                manual
        rr_weight               priorities
        no_path_retry           5
    }
    multipath {
        wwid                    1DEC_____321816758474
        alias                   red
```

(continues on next page)

(continued from previous page)

```
    }
}
```

### Configuration file devices section

`multipath-tools` has a built-in **device table** with reasonable defaults for more than 100 known multipath-capable storage devices. The `devices` section can be used to override these settings. If there are multiple matches for a given device, the attributes of all matching entries are applied to it. If an attribute is specified in several matching device subsections, later entries take precedence.

The only recognised attribute for the `devices` section is the `device` subsection. Devices detected in the system are matched against the device entries using the vendor, product, and revision fields.

The vendor, product, and revision fields that multipath or `multipathd` detect for devices in a system depend on the device type. For SCSI devices, they correspond to the respective fields of the "SCSI INQUIRY" page. In general, the command `multipathd show paths format "%d %s"` command can be used to see the detected properties for all devices in the system.

The device subsection recognises the following attributes:

1. `vendor`: (Mandatory) Regular expression to match the vendor name.
2. `product`: (Mandatory) Regular expression to match the product name.
3. `revision`: Regular expression to match the product revision.
4. `product_blacklist`: Products with the given vendor matching this string are blacklisted.
5. `alias_prefix`: The `user_friendly_names` prefix to use for this device type, instead of the default `mpath`.
6. `hardware_handler`: The hardware handler to use for this device type. The following hardware handlers are implemented (all of these are hardware-dependent):
    - `1 emc`: Hardware handler for *DGC* class arrays as CLARiiON CX/AX and EMC VNX and Unity families.
    - `1 rdac`: Hardware handler for LSI / *Engenio* / NetApp RDAC class as NetApp SANtricity E/EF Series, and OEM arrays from IBM DELL SGI STK and SUN.
    - `1 hp_sw`: Hardware handler for HP/COMPAQ/DEC HSG80 and MSA/HSV arrays with Active/Standby mode exclusively.
    - `1 alua`: Hardware handler for SCSI-3 ALUA-compatible arrays.
    - `1 ana`: Hardware handler for NVMe ANA-compatible arrays.

### Optional attributes

The following attributes are optional – if not set, the default values are taken from the defaults section:

- `path_grouping_policy`
- `uid_attribute`

- getuid_callout
- path_selector
- path_checker
- prio
- prio_args
- features
- failback
- rr_weight
- no_path_retry
- rr_min_io
- rr_min_io_rq
- fast_io_fail_tmo
- dev_loss_tmo
- flush_on_last_del
- user_friendly_names
- retain_attached_hw_handler
- detect_prio
- detect_checker
- deferred_remove
- san_path_err_threshold
- san_path_err_forget_rate
- san_path_err_recovery_time
- marginal_path_err_sample_time
- marginal_path_err_rate_threshold
- marginal_path_err_recheck_gap_time
- marginal_path_double_failed_time
- delay_watch_checks
- delay_wait_checks
- skip_kpartx
- max_sectors_kb
- ghost_delay
- all_tg_pt

### Example

```
devices {
    device {
        vendor "3PARdata"
        product "VV"
        path_grouping_policy "group_by_prio"
        hardware_handler "1 alua"
        prio "alua"
        failback "immediate"
        no_path_retry 18
        fast_io_fail_tmo 10
        dev_loss_tmo "infinity"
    }
    device {
        vendor "DEC"
        product "HSG80"
        path_grouping_policy "group_by_prio"
        path_checker "hp_sw"
        hardware_handler "1 hp_sw"
        prio "hp_sw"
        no_path_retry "queue"
    }
}
```

### Multipath configuration examples

Before moving on with this section we suggesting reading or being familiar with the topics covered in:

1. *Introduction to device mapper multipathing*
2. *Configuration options and overview*

For consistency with those sections, we will refer here to device mapper multipathing as **multipath**.

### Basic setup

Before setting up multipath on your system, ensure that your system has been updated and includes the `multipath-tools` package. If you want to boot from the storage area network (SAN), then the `multipath-tools-boot` package is also required.

A very simple `/etc/multipath.conf` file exists, as explained in *the configuration overview*. All attributes not declared in `multipath.conf` are taken from the `multipath-tools` internal database and its internal blacklist.

The **internal attributes** database can be acquired by running the following on the command line:

```
sudo multipath -t
```

Multipath usually works out-of-the-box with most common storages. This **does not mean** the default configuration variables should be used in production: the default variables don't

treat important parameters your storage might need.

With the internal attributes (described above), and the example below, you will likely be able to create your `/etc/multipath.conf` file by squashing the code blocks below. Make sure to read the `defaults` section attribute comments and make any changes based on what your environment needs.

### Example of a defaults section

```
defaults {
    #
    # name    : polling_interval
    # scope   : multipathd
    # desc    : interval between two path checks in seconds. For
    #           properly functioning paths, the interval between checks
    #           will gradually increase to (4 * polling_interval).
    # values  : n > 0
    # default : 5
    #
    polling_interval 10

    #
    # name    : path_selector
    # scope   : multipath & multipathd
    # desc    : the default path selector algorithm to use
    #           these algorithms are offered by the kernel multipath target
    # values  : "round-robin 0"  = Loop through every path in the path group,
    #                              sending the same amount of IO to each.
    #           "queue-length 0" = Send the next bunch of IO down the path
    #                              with the least amount of outstanding IO.
    #           "service-time 0" = Choose the path for the next bunch of IO
    #                              based on the amount of outstanding IO to
    #                              the path and its relative throughput.
    # default : "service-time 0"
    #
    path_selector "round-robin 0"

    #
    # name    : path_grouping_policy
    # scope   : multipath & multipathd
    # desc    : the default path grouping policy to apply to unspecified
    #           multipaths
    # values  : failover           = 1 path per priority group
    #           multibus           = all valid paths in 1 priority group
    #           group_by_serial    = 1 priority group per detected serial
    #                                number
    #           group_by_prio      = 1 priority group per path priority
    #                                value
    #           group_by_node_name = 1 priority group per target node name
    # default : failover
    #
```

(continues on next page)

(continued from previous page)

```
    path_grouping_policy multibus

    #
    # name    : uid_attribute
    # scope   : multipath & multipathd
    # desc    : the default udev attribute from which the path
    #       identifier should be generated.
    # default : ID_SERIAL
    #
    uid_attribute "ID_SERIAL"

    #
    # name    : getuid_callout
    # scope   : multipath & multipathd
    # desc    : the default program and args to callout to obtain a unique
    #           path identifier. This parameter is deprecated.
    #           This parameter is deprecated, superseded by uid_attribute
    # default : /lib/udev/scsi_id --whitelisted --device=/dev/%n
    #
    getuid_callout "/lib/udev/scsi_id --whitelisted --device=/dev/%n"

    #
    # name    : prio
    # scope   : multipath & multipathd
    # desc    : the default function to call to obtain a path
    #           priority value. The ALUA bits in SPC-3 provide an
    #           exploitable prio value for example.
    # default : const
    #
    # prio "alua"

    #
    # name    : prio_args
    # scope   : multipath & multipathd
    # desc    : The arguments string passed to the prio function
    #           Most prio functions do not need arguments. The
    #       datacore prioritizer need one.
    # default : (null)
    #
    # prio_args "timeout=1000 preferredsds=foo"

    #
    # name    : features
    # scope   : multipath & multipathd
    # desc    : The default extra features of multipath devices.
    #           Syntax is "num[ feature_0 feature_1 ...]", where `num' is the
    #           number of features in the following (possibly empty) list of
    #           features.
    # values  : queue_if_no_path = Queue IO if no path is active; consider
```

(continues on next page)

(continued from previous page)

```
#                               using the `no_path_retry' keyword instead.
#           no_partitions     = Disable automatic partitions generation via
#                               kpartx.
# default : "0"
#
features    "0"
#features   "1 queue_if_no_path"
#features   "1 no_partitions"
#features   "2 queue_if_no_path no_partitions"

#
# name    : path_checker, checker
# scope   : multipath & multipathd
# desc    : the default method used to determine the paths' state
# values  : readsector0|tur|emc_clariion|hp_sw|directio|rdac|cciss_tur
# default : directio
#
path_checker directio

#
# name    : rr_min_io
# scope   : multipath & multipathd
# desc    : the number of IO to route to a path before switching
#           to the next in the same path group for the bio-based
#           multipath implementation. This parameter is used for
#           kernels version up to 2.6.31; newer kernel version
#           use the parameter rr_min_io_rq
# default : 1000
#
rr_min_io 100

#
# name    : rr_min_io_rq
# scope   : multipath & multipathd
# desc    : the number of IO to route to a path before switching
#           to the next in the same path group for the request-based
#           multipath implementation. This parameter is used for
#           kernels versions later than 2.6.31.
# default : 1
#
rr_min_io_rq 1

#
# name    : flush_on_last_del
# scope   : multipathd
# desc    : If set to "yes", multipathd will disable queueing when the
#           last path to a device has been deleted.
# values  : yes|no
# default : no
```

(continues on next page)

(continued from previous page)

```
#
flush_on_last_del yes

#
# name    : max_fds
# scope   : multipathd
# desc    : Sets the maximum number of open file descriptors for the
#           multipathd process.
# values  : max|n > 0
# default : None
#
max_fds 8192

#
# name    : rr_weight
# scope   : multipath & multipathd
# desc    : if set to priorities the multipath configurator will assign
#           path weights as "path prio * rr_min_io"
# values  : priorities|uniform
# default : uniform
#
rr_weight priorities

#
# name    : failback
# scope   : multipathd
# desc    : tell the daemon to manage path group failback, or not to.
#           0 means immediate failback, values >0 means deffered
#           failback expressed in seconds.
# values  : manual|immediate|n > 0
# default : manual
#
failback immediate

#
# name    : no_path_retry
# scope   : multipath & multipathd
# desc    : tell the number of retries until disable queueing, or
#           "fail" means immediate failure (no queueing),
#           "queue" means never stop queueing
# values  : queue|fail|n (>0)
# default : (null)
#
no_path_retry fail

#
# name    : queue_without_daemon
# scope   : multipathd
# desc    : If set to "no", multipathd will disable queueing for all
```

(continues on next page)

(continued from previous page)

```
#           devices when it is shut down.
# values  : yes|no
# default : yes
queue_without_daemon no

#
# name    : user_friendly_names
# scope   : multipath & multipathd
# desc    : If set to "yes", using the bindings file
#           /etc/multipath/bindings to assign a persistent and
#           unique alias to the multipath, in the form of mpath<n>.
#           If set to "no" use the WWID as the alias. In either case
#           this be will be overriden by any specific aliases in this
#           file.
# values  : yes|no
# default : no
user_friendly_names yes

#
# name    : mode
# scope   : multipath & multipathd
# desc    : The mode to use for the multipath device nodes, in octal.
# values  : 0000 - 0777
# default : determined by the process
mode 0644

#
# name    : uid
# scope   : multipath & multipathd
# desc    : The user id to use for the multipath device nodes. You
#           may use either the numeric or symbolic uid
# values  : <user_id>
# default : determined by the process
uid 0

#
# name    : gid
# scope   : multipath & multipathd
# desc    : The group id to user for the multipath device nodes. You
#           may use either the numeric or symbolic gid
# values  : <group_id>
# default : determined by the process
gid disk

#
# name    : checker_timeout
# scope   : multipath & multipathd
# desc    : The timeout to use for path checkers and prioritizers
#           that issue scsi commands with an explicit timeout, in
```

(continues on next page)

(continued from previous page)

```
#           seconds.
# values  : n > 0
# default : taken from /sys/block/sd<x>/device/timeout
checker_timeout 60

#
# name    : fast_io_fail_tmo
# scope   : multipath & multipathd
# desc    : The number of seconds the scsi layer will wait after a
#           problem has been detected on a FC remote port before failing
#           IO to devices on that remote port.
# values  : off | n >= 0 (smaller than dev_loss_tmo)
# default : determined by the OS
fast_io_fail_tmo 5

#
# name    : dev_loss_tmo
# scope   : multipath & multipathd
# desc    : The number of seconds the scsi layer will wait after a
#           problem has been detected on a FC remote port before
#           removing it from the system.
# values  : infinity | n > 0
# default : determined by the OS
dev_loss_tmo 120

#
# name    : bindings_file
# scope   : multipath
# desc    : The location of the bindings file that is used with
#           the user_friendly_names option.
# values  : <full_pathname>
# default : "/var/lib/multipath/bindings"
# bindings_file "/etc/multipath/bindings"

#
# name    : wwids_file
# scope   : multipath
# desc    : The location of the wwids file multipath uses to
#           keep track of the created multipath devices.
# values  : <full_pathname>
# default : "/var/lib/multipath/wwids"
# wwids_file "/etc/multipath/wwids"

#
# name    : reservation_key
# scope   : multipath
# desc    : Service action reservation key used by mpathpersist.
# values  : <key>
# default : (null)
```

(continues on next page)

(continued from previous page)

```
    # reservation_key "mpathkey"

    #
    # name    : force_sync
    # scope   : multipathd
    # desc    : If set to yes, multipath will run all of the checkers in
    #           sync mode, even if the checker has an async mode.
    # values  : yes|no
    # default : no
    force_sync yes

    #
    # name    : config_dir
    # scope   : multipath & multipathd
    # desc    : If not set to an empty string, multipath will search
    #           this directory alphabetically for files ending in ".conf"
    #           and it will read configuration information from these
    #           files, just as if it was in /etc/multipath.conf
    # values  : "" or a fully qualified pathname
    # default : "/etc/multipath/conf.d"

    #
    # name    : delay_watch_checks
    # scope   : multipathd
    # desc    : If set to a value greater than 0, multipathd will watch
    #           paths that have recently become valid for this many
    #           checks.  If they fail again while they are being watched,
    #           when they next become valid, they will not be used until
    #           they have stayed up for delay_wait_checks checks.
    # values  : no|<n> > 0
    # default : no
    delay_watch_checks 12

    #
    # name    : delay_wait_checks
    # scope   : multipathd
    # desc    : If set to a value greater than 0, when a device that has
    #           recently come back online fails again within
    #           delay_watch_checks checks, the next time it comes back
    #           online, it will marked and delayed, and not used until
    #           it has passed delay_wait_checks checks.
    # values  : no|<n> > 0
    # default : no
    delay_wait_checks 12
    }
```

### Example of a multipaths section

> **Note**
>
> You can obtain the WWIDs for your LUNs by running: `multipath -ll` after the service `multipath-tools.service` has been restarted.

```
multipaths {
    multipath {
        wwid 360000000000000000e00000000030001
        alias yellow
    }
    multipath {
        wwid 360000000000000000e00000000020001
        alias blue
    }
    multipath {
        wwid 360000000000000000e00000000010001
        alias red
    }
    multipath {
        wwid 360000000000000000e00000000040001
        alias green
    }
    multipath {
        wwid 360000000000000000e00000000050001
        alias purple
    }
}
```

### Example of a devices section

```
# devices {
#     device {
#         vendor "IBM"
#         product "2107900"
#         path_grouping_policy group_by_serial
#     }
# }
#
```

### Example of a blacklist section

```
# name    : blacklist
# scope   : multipath & multipathd
# desc    : list of device names to discard as not multipath candidates
#
# Devices can be identified by their device node name "devnode",
```

(continues on next page)

(continued from previous page)

```
# their WWID "wwid", or their vender and product strings "device"
# default : fd, hd, md, dm, sr, scd, st, ram, raw, loop, dcssblk
#
# blacklist {
#     wwid 26353900f02796769
#     devnode "^(ram|raw|loop|fd|md|dm-|sr|scd|st)[0-9]\*"
#     devnode "^hd[a-z]"
#     devnode "^dcssblk[0-9]\*"
#     device {
#         vendor DEC.\*
#         product MSA[15]00
#     }
# }
```

### Example of a blacklist exception section

```
# name    : blacklist_exceptions
# scope   : multipath & multipathd
# desc    : list of device names to be treated as multipath candidates
#           even if they are on the blacklist.
#
# Note: blacklist exceptions are only valid in the same class.
# It is not possible to blacklist devices using the devnode keyword
# and to exclude some devices of them using the wwid keyword.
# default : -
#
# blacklist_exceptions {
#        devnode "^dasd[c-d]+[0-9]\*"
#        wwid    "IBM.75000000092461.4d00.34"
#        wwid    "IBM.75000000092461.4d00.35"
#        wwid    "IBM.75000000092461.4d00.36"
# }
```

### Common tasks and procedures

This section shows examples of common configuration procedures and tasks. Before moving on with this section it's a good idea to read or be familiar with the topics in these pages:

1. *Introduction to device mapper multipathing*
2. *Configuration options and overview*
3. *Configuration examples*

For consistency, we will refer to device mapper multipathing as **multipath**.

### Resizing online multipath devices

To resize online multipath devices, first find all the paths to the logical unit number (LUN) that is to be resized by running the following command:

```
$ sudo multipath -ll

mpathb (360014056eee8ec6e1164fcb959086482) dm-0 LIO-ORG,lun01
size=1.0G features='0' hwhandler='1 alua' wp=rw
|-+- policy='service-time 0' prio=50 status=active
| `- 7:0:0:1 sde 8:64 active ready running
`-+- policy='service-time 0' prio=50 status=enabled
  `- 8:0:0:1 sdf 8:80 active ready running
mpatha (36001405e3c2841430ee4bf3871b1998b) dm-1 LIO-ORG,lun02
size=1.0G features='0' hwhandler='1 alua' wp=rw
|-+- policy='service-time 0' prio=50 status=active
| `- 7:0:0:2 sdc 8:32 active ready running
`-+- policy='service-time 0' prio=50 status=enabled
  `- 8:0:0:2 sdd 8:48 active ready running
```

Now, reconfigure `mpathb` (with `wwid = 360014056eee8ec6e1164fcb959086482`) to have 2 GB instead of just 1 GB and check if it has changed:

```
$ echo 1 | sudo tee /sys/block/sde/device/rescan

1

$ echo 1 | sudo tee /sys/block/sdf/device/rescan

1

$ sudo multipath -ll

mpathb (360014056eee8ec6e1164fcb959086482) dm-0 LIO-ORG,lun01
size=1.0G features='0' hwhandler='1 alua' wp=rw
|-+- policy='service-time 0' prio=50 status=active
| `- 7:0:0:1 sde 8:64 active ready running
`-+- policy='service-time 0' prio=50 status=enabled
  `- 8:0:0:1 sdf 8:80 active ready running
mpatha (36001405e3c2841430ee4bf3871b1998b) dm-1 LIO-ORG,lun02
size=1.0G features='0' hwhandler='1 alua' wp=rw
|-+- policy='service-time 0' prio=50 status=active
| `- 7:0:0:2 sdc 8:32 active ready running
`-+- policy='service-time 0' prio=50 status=enabled
  `- 8:0:0:2 sdd 8:48 active ready running
```

Not yet! We still need to re-scan the multipath map:

```
$ sudo multipathd resize map mpathb

ok
```

And then we are good:

```
$ sudo multipath -ll

mpathb (360014056eee8ec6e1164fcb959086482) dm-0 LIO-ORG,lun01
size=2.0G features='0' hwhandler='1 alua' wp=rw
|-+- policy='service-time 0' prio=50 status=active
| `- 7:0:0:1 sde 8:64 active ready running
.`-+- policy='service-time 0' prio=50 status=enabled
  `- 8:0:0:1 sdf 8:80 active ready running
mpatha (36001405e3c2841430ee4bf3871b1998b) dm-1 LIO-ORG,lun02
size=1.0G features='0' hwhandler='1 alua' wp=rw
|-+- policy='service-time 0' prio=50 status=active
| `- 7:0:0:2 sdc 8:32 active ready running
`-+- policy='service-time 0' prio=50 status=enabled
  `- 8:0:0:2 sdd 8:48 active ready running
```

Make sure to run `resize2fs /dev/mapper/mpathb` to resize the *filesystem*.

### Move root file system from a single path device to a multipath device

This is greatly simplified by the use of UUIDs to identify devices with an intrinsic label. To do this, install `multipath-tools-boot` and reboot your system. This will rebuild the initial RAM disk and afford multipath the opportunity to build its paths before the root filesystem is mounted by UUID.

> **Note**
>
> Whenever `multipath.conf` is updated, initrd should be updated as well by running: `update-initramfs -u -k all` The reason for this is that `multipath.conf` is copied to the RAM disk, and is integral to determining the available devices to map via its `denylist`and `devices` sections.

### The multipathd daemon

If you have trouble implementing a multipath configuration, you should ensure the multipath daemon is running as described in *the example configuration page*. The `multipathd` daemon must be running in order to use multipath devices.

### Multipath command output

When you create, modify, or list a multipath device, you get a printout of the current device setup. The format is as follows for each multipath device:

```
action_if_any: alias (wwid_if_different_from_alias) dm_device_name_if_known
vendor,product
   size=size features='features' hwhandler='hardware_handler' wp=write_permission_
if_known
```

For each path group:

```
 -+- policy='scheduling_policy' prio=prio_if_known
 status=path_group_status_if_known
```

For each path:

```
  `- host:channel:id:lun devnode major:minor dm_status_if_known path_status
 online_status
```

For example, the output of a multipath command might appear as follows:

```
mpathb (360014056eee8ec6e1164fcb959086482) dm-0 LIO-ORG,lun01
size=2.0G features='0' hwhandler='1 alua' wp=rw
|-+- policy='service-time 0' prio=50 status=active
| `- 7:0:0:1 sde 8:64 active ready running
`-+- policy='service-time 0' prio=50 status=enabled
  `- 8:0:0:1 sdf 8:80 active ready running
```

If the path is up and ready for I/O, the status of the path is `ready` or `ghost`. If the path is down, the status is `faulty` or `shaky`. The path status is updated periodically by the `multipathd` daemon based on the **polling interval** defined in the `/etc/multipath.conf` file.

The `dm_status` is similar to the `path` status, but from the kernel's point of view. The `dm_status` has two states: `failed`, which is analogous to `faulty`, and `active`, which covers all other path states. Occasionally, the `path` state and the `dm` state of a device will temporary not agree.

The possible values for `online_status` are `running` and `offline`. A status of `offline` means that the SCSI device has been disabled.

### Multipath queries with the multipath command

You can use the `-l` and `-ll` options of the `multipath` command to display the current multipath configuration.

- `-l` : Displays multipath topology gathered from information in `sysfs` and the device mapper
- `-ll` : Displays the information the `-l` displays in addition to all other available components of the system

When displaying the multipath configuration, there are three verbosity levels you can specify with the `-v` option of the `multipath` command:

- `-v0` : Yields no output
- `-v1` : Outputs only the created or updated multipath names, which you can then feed to other tools such as `kpartx`
- `-v2` : Prints all detected paths, multipaths, and device maps

**Note**

The default verbosity level of multipath is 2 and can be globally modified by defining the verbosity attribute in the `defaults` section of `multipath.conf`

The following example shows the output of a `sudo multipath -l` command:

```
mpathb (360014056eee8ec6e1164fcb959086482) dm-0 LIO-ORG,lun01
size=2.0G features='0' hwhandler='1 alua' wp=rw
|-+- policy='service-time 0' prio=0 status=active
| `- 7:0:0:1 sde 8:64 active undef running
`-+- policy='service-time 0' prio=0 status=enabled
  `- 8:0:0:1 sdf 8:80 active undef running
mpatha (36001405e3c2841430ee4bf3871b1998b) dm-1 LIO-ORG,lun02
size=1.0G features='0' hwhandler='1 alua' wp=rw
|-+- policy='service-time 0' prio=0 status=active
| `- 7:0:0:2 sdc 8:32 active undef running
`-+- policy='service-time 0' prio=0 status=enabled
  `- 8:0:0:2 sdd 8:48 active undef running
```

### Determining device mapper entries with dmsetup

You can use the `dmsetup` command to find out which device mapper entries match the multipathed devices. The following command displays all the device mapper devices and their major and minor numbers. The minor numbers determine the name of the **dm** device. For example, a minor number of 1 corresponds to the `multipathd` device `/dev/dm-1`.

```
$ sudo dmsetup ls
mpathb  (253:0)
mpatha  (253:1)

$ ls -lahd /dev/dm*
brw-rw---- 1 root disk 253, 0 Apr 27 14:49 /dev/dm-0
brw-rw---- 1 root disk 253, 1 Apr 27 14:47 /dev/dm-1
```

### Troubleshooting with the multipathd interactive console

The `multipathd -k` command is an interactive interface to the `multipathd` daemon. Running this command brings up an interactive multipath console where you can enter `help` to get a list of available commands, you can enter an interactive command, or you can enter Ctrl+D to quit.

The `multipathd` interactive console can be used to troubleshoot problems with your system. For example, the following command sequence displays the multipath configuration, including the defaults, before exiting the console.

```
$ sudo multipathd -k
  > show config
  > CTRL-D
```

The following command sequence ensures that multipath has picked up any changes to the `multipath.conf`:

```
$ sudo multipathd -k
> reconfigure
> CTRL-D
```

Use the following command sequence to ensure that the path checker is working properly:

```
$ sudo multipathd -k
> show paths
> CTRL-D
```

Commands can also be streamed into `multipathd` using STDIN like so:

```
$ echo 'show config' | sudo multipathd -k
```

### Introduction to backups

It's important to back up your Ubuntu installation so you can recover quickly if you experience data loss. You can create redundancy by using multiple back-up methods. Redundant data is useful if the primary back-up fails.

It is important to develop a backup plan that consists of:

- What should be backed up
- How often to back it up
- Where backups should be stored
- How to restore your backups

It is good practice to store important backup media off-site in case of a disaster. Physical backup media like removable hard drives or tape can be moved off-site manually. When it is either impossible or impractical to move media, backups can be copied over a WAN link to a server in another location.

### Backup options

There are many ways to back up an Ubuntu installation. On Ubuntu, two primary ways of backing up your system are **backup utilities** and **shell scripts**. For additional protection, you can combine backup methods so you don't rely on a single system.

### Backup utilities

The easiest way to create backups is to use a dedicated tool like Bacula or rsnapshot. These tools have easy-to-use interface or CLI to help simplify the backup management process. They have powerful features such as automation, compression, data recovery, encryption and incremental backups. Incremental backups only store changes made since the last backup which can significantly decrease storage space needs and backup time.

Bacula's advanced features and support for additional customization make it a good choice for enterprise systems or users with complex needs. rsnapshot is ideal for individual users or small-scale organizations who want a simple and efficient solution.

| Tool | Back up | Backup Method | Install and configure |
|---|---|---|---|
| **Bacula** | Multiple systems over a network | Incremental backups | *how to install and configure Bacula* |
| **rsnapshot** | Single system | Periodic "snapshots" of files locally or remotely with SSH | *how to install and configure rsnapshot* |

#### Shell scripts

You can fully tailor the backup process to your specific requirements with shell scripts. The advantage of shell scripts over using backup utilities is they offer full flexibility and customization.

Refer to this guide for instructions on *how to use shell scripts for backups* – or you can take a look at these examples:

- Basic backup shell script
- *An example of archive rotation with shell scripts*

#### See also

- How-to: *Data and storage*

# 4.5. Web services

Our *Web services* section includes details about web servers and how they work, as well as related topics like Squid proxy servers.

## 4.5.1. Web services

Web servers are used to serve web pages when requested by client computers.

#### Introduction

#### Introduction to web servers

Web servers are used to serve web pages requested by client computers. Clients typically request and view web pages using web browser applications such as Firefox, Opera, Chromium, or Internet Explorer.

If you're new to web servers, see this page for more information *on the key concepts*.

#### Squid proxy server

Squid is a popular, open-source, proxy caching server that can help optimise network efficiency and improve response times by saving local copies of frequently accessed content. Read more *about Squid proxy servers* and what they can do, or find out *how to install a Squid server*.

#### Web server

Apache is the most commonly used web server on Linux systems, and the current version is Apache2. It is robust, reliable, and highly configurable. This set of guides will show you:

- *How to install and configure Apache2*
- *How to configure Apache2 for your needs*
- *How to extend Apache2's functionality with modules*

Nginx is a popular alternative web server also widely used on Linux, with a focus on static file serving performance, ease of configuration, and use as both a web server and reverse proxy server.

- *How to install Nginx*

- *How to configure Nginx*
- *How to use Nginx modules*

### Database server

The database server, when included in the LAMP stack, allows data for web applications to be stored and managed. MySQL is one of the most popular open source Relational Database Management Systems (RDBMS) available, and you can find out in this guide *how to install MySQL* – or *PostgreSQL*, as another popular alternative.

### Scripting languages

Server-side scripting languages allow for the creation of dynamic web content, processing of web forms, and interacting with databases (amongst other crucial tasks). PHP is most often used, and we can show you *how to install PHP*, or if you prefer, we can show you *how to install Ruby on Rails*.

Whichever scripting language you choose, you will need to have installed and configured your web and database servers beforehand.

## Key concepts

### About web servers

The primary function of a **web server** is to store, process and deliver **web pages** to clients. The clients communicate with the server by sending HTTP requests.

Clients, mostly via **web browsers**, request specific resources and the server responds with the content of that resource (or an error message). The response is usually a web page in the form of HTML documents – which may include images, style sheets, scripts, and text.

### URLs

Users enter a Uniform Resource Locator (URL) to point to a web server by means of its *Fully Qualified Domain Name (FQDN)* and a path to the required resource. For example, to view the home page of the Ubuntu Web site a user will enter only the FQDN:

```
www.ubuntu.com
```

To view the community sub-page, a user will enter the FQDN followed by a path:

```
www.ubuntu.com/community
```

### Transfer protocols

The most common protocol used to transfer web pages is the Hyper Text Transfer Protocol (HTTP). Protocols such as Hyper Text Transfer Protocol over Secure Sockets Layer (HTTPS), and File Transfer Protocol (FTP), a protocol for uploading and downloading files, are also supported.

### HTTP status codes

When accessing a web server, every HTTP request received is responded to with content and a HTTP status code. HTTP status codes are three-digit codes, which are grouped into five different classes. The class of a status code can be quickly identified by its first digit:

- **1xx** : *Informational* - Request received, continuing process
- **2xx** : *Success* - The action was successfully received, understood, and accepted
- **3xx** : *Redirection* - Further action must be taken in order to complete the request
- **4xx** : *Client Error* - The request contains bad syntax or cannot be fulfilled
- **5xx** : *Server Error* - The server failed to fulfill an apparently valid request

For more information about status codes, check the RFC 2616.

### Implementation

Web servers are heavily used in the deployment of websites, and there are two different implementations:

- **Static web server**: The content of the server's response will be the hosted files "as-is".
- **Dynamic web server**: Consists of a web server plus additional software (usually an *application server* and a *database*).

  For example, to produce the web pages you see in your web browser, the application server might fill an HTML template with contents from a database. We can therefore say that the content of the server's response is generated dynamically.

### About Squid proxy servers

Squid is a proxy caching server which provides proxy and cache services for Hyper Text Transport Protocol (HTTP), File Transfer Protocol (FTP), and other popular network protocols.

It acts as an intermediary between web servers and clients. When a client sends a request for content, Squid fetches the content from the web server and creates a local copy. Then, if a request is made again, it shows the local, cached copy instead of making another request to the web server. In this way, performance is improved and network bandwidth is optimised. It can also filter web traffic, helping to improve security.

### Features

The Squid proxy cache server scales from the branch office to enterprise level networks. It provides extensive, granular access controls, and monitoring of critical parameters via the Simple Network Management Protocol (SNMP).

When selecting a computer system for use as a dedicated Squid caching proxy server, it is helpful to configure it with a large amount of physical memory as Squid maintains an in-memory cache for increased performance.

### Caching

Squid can implement caching and proxying of Secure Sockets Layer (SSL) requests and caching of Domain Name Server (*DNS*) lookups, and perform transparent caching. Squid also supports a wide variety of caching protocols, such as Internet Cache Protocol (ICP), the Hyper Text Caching Protocol (HTCP), the Cache Array Routing Protocol (CARP), and the Web Cache Coordination Protocol (WCCP).

Squid caching can be done in-RAM for fastest access, on disk for larger capacity, and may use a hierarchical directory structure for efficient file organization.

If you would like to know how to install and configure your own Squid server, refer to *our installation guide*.

### Further reading

- The Squid website
- Ubuntu Wiki page on Squid.

### See also

- How-to: *Web services*

# 4.6. Virtualisation and containers

Our *Virtualisation and containers* section includes overviews of the available virtualisation and container tooling in the Ubuntu space, as well as more detail about topics like rock images, Docker, eBPF and more!

## 4.6.1. Virtualisation and containers

- *Introduction to virtualization* discusses and disambiguates between virtualization and containers

### Introduction to virtualization

Virtualization is a technology that allows you to create safe, isolated environments on your server. Developers can use virtualization to create self-contained sandboxes for development and testing that cannot negatively affect the host machine. System administrators can use virtualization to scale network resources to meet changing demand, giving greater control and flexibility in managing infrastructure.

The virtualisation stack is made using layers of **abstraction**. Each layer hides some of the complexity of the layer (or layers) beneath, presenting an increasingly high-level view of the technology. This makes the underlying technology progressively easier to understand and work with.

### Virtual machines

Virtual machines (VMs) are essentially computers-within-computers. Every VM includes its own operating system and simulated resources, making it completely independent of the host machine and any other VM. Although more resource-intensive than a container, a VM provides strong isolation and reduces the need for additional hardware to run different operating system environments. To find out more, see this overview of *the different VM tools and technologies* available for Ubuntu.

### Containers

Containers, on the other hand, are a more lightweight virtualization technology. They share the operating system of the host machine, so they are much quicker to provision when demand for resources is high. They are often used for packaging and running applications. They contain everything the application needs including any required dependencies and libraries. This ensures consistency across different environments. Containers come in two main flavors: **system** containers, and **application** containers.

### System containers

System containers simulate a full machine in a similar way to a VM. However, since containers run on the host kernel they don't need to install an operating system, making them quick to start and stop. They are often used for separating user spaces.

### Application containers

Application containers package all of the components needed for a specific application or process to run, including any required dependencies and libraries. This means the application can run consistently, even across different environments, without running into problems of missing dependencies. Application containers are particularly useful for running microservices.

For more details about container tools available in the Ubuntu space, *take a look at this overview*.

### Virtualization

- *VM tools overview* gives more details about the virtual machines available in Ubuntu

We also give further details on some specific usage scenarios here:

- *QEMU microvm*
- *Upgrading the machine type of your VM*

### VM tools in the Ubuntu space

Let's take a look at some of the major tools and technologies available in the Ubuntu virtualization stack, in order of increasing abstraction.

### KVM

**Abstraction layer**: Hardware virtualization

Kernel-Based Virtual Machine (KVM) is a Linux kernel module that enables hardware-assisted virtualization. It is the default virtualization technology supported by Ubuntu.

For Intel and *AMD* hardware, KVM requires virtualization extensions in order to run. KVM is also available for IBM Z and LinuxONE, IBM POWER, and ARM64.

### QEMU

**Abstraction layer**: Emulation

Quick Emulator (QEMU) is a versatile and powerful open source machine emulator. It emulates complete virtual machines, which allows users to run machines with different operating systems than the underlying host system – without needing to purchase dedicated hardware.

QEMU primarily functions as the user-space backend for KVM. When used in collaboration with KVM kernel components, it harnesses the hardware virtualization capability that KVM provides in order to efficiently virtualize guests.

It also has a command line interface and a monitor for interacting with running guests. However, these are typically only used for development purposes.

To find out how to get started with QEMU quickly, check out this guide on *how to set up QEMU*.

### libvirt

**Abstraction layer**: API and toolkit

libvirt provides an abstraction layer away from specific versions and hypervisors, giving users a command-line toolkit and API for managing virtualizations.

By providing an abstraction away from the underlying technologies (such as QEMU/KVM), libvirt makes it possible to manage all kinds of virtual resources – across different platforms and hypervisors – using one single, common interface. This can greatly simplify administration and automation tasks.

For details of how to get libvirt set up and the basics of how to use it, see this guide on *how to use libvirt*.

### Multipass and UVtool

**Abstraction layer**: User-friendly, CLI-based VM management

Multipass and UVtool provide an abstraction layer away from libvirt, using command-line interfaces to simplify VM management. Both Multipass and UVtool are widely used in development and testing; they are lightweight and straightforward to use, and can greatly simplify the process of creating and managing VMs.

UVtool is essentially a wrapper around libvirt, providing an additional abstraction layer to simplify its use. Multipass is not based on libvirt, but can be integrated with it. This means that both tools can be used as part of a virtualization "stack" based around QEMU and libvirt.

If you want to get started with either of these tools, you can see our guides on *how to use Multipass* or *how to use UVtool*.

### virt-manager

**Abstraction layer**: GUI-based VM management

Virt-manager, the Virtual Machine Manager, provides another high-level way to manage VMs. Like UVtool, virt-manager uses libvirt on the backend. However, unlike UVtool, its abstraction is presented in the form of a *graphical user interface (GUI)*.

Although in many ways this makes virt-manager easier to use than Multipass and UVtool, it also introduces more complex tooling that supports more advanced users.

To get started with virt-manager, *this how-to guide* showcases all the basic functionality and tooling.

### QEMU microvm

QEMU microvm is a special case of virtual machine (VMs), designed to be optimised for initialisation speed and minimal resource use.

The underlying concept of a microvm is based on giving up some capabilities of standard QEMU in order to reduce complexity and gain speed. Maybe - for your use-case - you do not need e.g. the hypervisor to be able to pretend to have a network card from the 90s, nor to emulate a CPU of a foreign architecture, nor live migrate with external I/O going on. In such cases, a lot of what QEMU provides is not needed and a less complex approach like a microvm might be interesting to you.

All of that is a balance that needs to be decided by the needs of your use case. There will surely be arguments and examples going both ways - so be careful. Giving up on unnecessary features to gain speed is great, but it is not so great if - some time after deploying your project - you realise that you now need a feature only available in the more complete solution.

QEMU provides additional components that were added to support this special use case:

1. The `microvm` machine type
2. Alternative simple *firmware (FW)* that can boot Linux called `qboot`
3. Ubuntu has a QEMU build with reduced features matching these use cases called `qemu-system-x86-microvm`

### Why a special workload?

One has to understand that minimising the QEMU initialisation time only yields a small gain, by shaving off parts of a task that usually do not take a long time. That is only worth it if the workload you run is not taking much longer anyway. For example, by booting a fully generic operating system, followed by more time to completely initialise your workload.

There are a few common ways to adapt a workload to match this:

- Use faster bootloaders and virtual firmware (see `qboot` below) with a reduced feature set, not as generally capable but sufficient for a particular use case.
- Even the fastest bootloader is slower than no bootloader, so often the kernel is directly passed from the host *filesystem*. A drawback of this solution is the fact that the guest system will not have control over the kernel anymore, thus restricting what can be done inside the guest system.
- Sometimes a simpler user space like busybox or a container-like environment is used.
- In a similar fashion, you could use a customised kernel build with a reduced feature set with only what is needed for a given use case.

A common compromise of the above options is aligning virtualization with container paradigms. While behaving mostly like a container, those tools will use virtualization instead of namespaces for the isolation. Examples of that are:

- container-like, as in kata containers,
- function-based services as in Firecracker,
- system containers as in *LXD*.

In particular, *LXD* added VM mode to allow the very same UX with namespaces and virtualizaton.

Other related tools are more about creating VMs from containers like:

- slim from dockerfiles or
- krunvm from OCI images.

There are more of these out there, but the point is that one can mix and match to suit their needs. At the end of the day, many of the above use the same underlying technology of namespaces or QEMU/KVM.

This page tries to stick to the basics and not pick either higher level system mentioned above. Instead, it sticks to just QEMU to show how it's ideas of reduced firmware and microvms play into all of this.

### Create the example workload artifact

To create an example of such a small workload, we will follow the tutorial on how to build a sliced rock.

Out of these tutorials one gets an OCI-compatible artifact. It will be called `chiselled-hello_latest_amd64.rock`. That is now converted to a disk image for use as virtual disk in our later example.

```
# Convert the artifact of the ROCK tutorial into OCI format
$ sudo rockcraft.skopeo --insecure-policy copy oci-archive:chiselled-hello_latest_
amd64.rock oci:chiselled-hello.oci:latest
# Unpack that to a local directory
$ sudo apt install oci-image-tool
$ oci-image-tool  unpack --ref name=latest chiselled-hello.oci /tmp/chiselled-
hello
# Create some paths the kernel would be unhappy if they would be missing
$ mkdir /tmp/chiselled-hello/{dev,proc,sys,run,var}
# Convert the directory to a qcow2 image
$ sudo apt install guestfs-tools
$ sudo virt-make-fs --format=qcow2 --size=50M /tmp/chiselled-hello chiselled-
hello.qcow2
```

### Run the stripped workload in QEMU

Now that we have a stripped-down workload as an example, we can run it in standard QEMU and see that this is much quicker than booting a full operating system.

```
$ sudo qemu-system-x86_64 -m 128M -machine accel=kvm \
    -kernel /boot/vmlinuz-$(uname -r) \
    -append 'console=ttyS0 root=/dev/vda fsck.mode=skip init=/usr/bin/hello' \
    -nodefaults -no-user-config \
```

(continues on next page)

(continued from previous page)

```
    -display none -serial mon:stdio \
    -drive file=chiselled-hello.qcow2,index=0,format=qcow2,media=disk,if=none,
id=virtio1 \
    -device virtio-blk-pci,drive=virtio1
...
[    2.116207] Run /usr/bin/hello as init process
Hello, world!
```

Breaking down the command-line elements and their purpose in this context:

| command-line element | Explanation |
|---|---|
| `sudo` | `sudo` is a simple way for this example to work, but not recommended. Scenarios outside of an example should use separate kernel images and a user that is a member of the `kvm` group to access `/dev/kvm`. |
| `qemu-system-x` | Call the usual binary of QEMU used for system virtualization. |
| `-m 128M` | Allocate 128 megabytes of RAM for the guest. |
| `-machine accel=kvm` | Enable KVM. |
| `-kernel /boot/vmlinuz-$(una-r)` | Load the currently running kernel for the guest. |
| `-append '...'` | This passes four arguments to the kernel explained one by one in the following rows. |
| `console=ttyS0` | Tells the kernel which serial console it should send its output to. |
| `root=/dev/vda` | Informs it where to expect the root device matching the `virtio-block` device we provide. |
| `fsck.mode=skip` | Instructs the kernel to skip filesystem checks, which saves time. |
| `init=/usr/bin/hello` | Tell the kernel to directly start our test workload. |
| `-nodefaults` | Do not create the default set of devices. |
| `-no-user-conf` | Do not load any user provided config files. |
| `-display none` | Disable video output (due to `-nodefaults` and `-display none` we do not also need `-nographic`). |
| `-serial mon:stdio` | Map the virtual serial port and the monitor (for debugging) to stdio |
| `-drive ... -device ...` | Provide our test image as virtio based block device. |

After running this example we notice that, by changing the workload to something small and streamlined, the execution time went down from about 1 minute (when booting a bootloader into a full OS into a workload) to about 2 seconds (from when the kernel started accounting time), as expected.

For the purpose of what this page wants to explain, it is not important to be perfectly accurate and stable. We are now in the right order of magnitude (seconds instead of a minute)

in regard to the *overall time spent* to begin focusing on the time that the initialisation of firmware and QEMU consume.

### Using `qboot` and `microvm`

In the same way as `qemu-system-x86-microvm` is a reduced QEMU, qboot is a simpler variant to the extended feature set of seabios or UEFI that can do less, but therefore is faster at doing what it can.

If your system does not need the extended feature sets you can try `qboot` if this gives you an improvement for your use case. To do so add `-bios /usr/share/qemu/qboot.rom` to the QEMU command line.

QEMU microvm is a machine type inspired by Firecracker and constructed after its machine model.

In Ubuntu, we provide this on x86 as `qemu-system-x86_64-microvm` alongside the *standard* QEMU in the package `qemu-system-x86`.

Microvm aims for maximum compatibility by default; this means that you will probably want to switch off some more legacy devices that are not shown in this example. But for what is shown here, we want to keep it rather comparable to the non-microvm invocation. For more details on what else could be disabled see microvm.

Run the guest in `qemu-system-x86_64-microvm`:

```
$ sudo qemu-system-x86_64-microvm -m 128M -machine accel=kvm,rtc=on \
    -bios /usr/share/qemu/qboot.rom \
    -kernel /boot/vmlinuz-$(uname -r) \
    -append 'console=ttyS0 root=/dev/vda fsck.mode=skip init=/usr/bin/hello' \
    -nodefaults -no-user-config \
    -display none -serial mon:stdio \
    -drive file=chiselled-hello.qcow2,index=0,format=qcow2,media=disk,if=none,
id=virtio1 -device virtio-blk-device,drive=virtio1
```

Breaking down the changes to the command-line elements and their purpose:

| command-line element | Explanation |
|---|---|
| `qemu-syste` | Call the lighter, feature-reduced QEMU binary. |
| `-bios /usr/share/qemu/qboot.rom` | Running QEMU as `qemu-system-x86_64-microvm` will auto-select `/usr/share/seabios/bios-microvm.bin` which is a simplified SeaBIOS for this purpose. But, for the example shown here we want the even simpler `qboot`, so in addition we set `-bios /usr/share/qemu/qboot.rom`. |
| *info* | QEMU will auto-select the microvm machine type, equivalent to `-M microvm` which therefore doesn't need to be explicitly included here. |
| `... virtio-blk ...` | This feature-reduced QEMU only supports `virtio-bus`, so we need to switch the type `virtio-blk-pci` to `virtio-blk-device`. |

> Sadly, polluting this nice showcase there is currently an issue with the RTC initialisation not working in this mode - which makes the guest kernel wait ~1.3 + ~1.4 seconds. See this qemu bug if you are curious about that.
>
> But these changes were not about making the guest faster once it runs, instead it mostly is about the initialisation time (and kernel init by having less virtual hardware). And that we can check despite this issue.

On average, across a few runs (albeit not in a very performance-controlled environment) we can see the kernel start time to be 282ms faster comparing *normal QEMU* to `microvm` and another 526ms faster comparing `microvm` to `microvm+qboot`.

As mentioned, one could go further from here by disabling more legacy devices, using `hvc-console`, customising the guest CPU, switching off more subsystems like ACPI, or customising the kernel that is used. But this was meant to be an example on how `microvm` can be used in general, so we won't make it more complex for now.

### Alternative - using virtiofs

Another common path not fully explored in the example above is sharing the content with the guest via `virtiofsd`.

Doing so for our example could start with a conversion of the container artifact above to a shareable directory:

```
# Copy out the example the tutorial had in OCI format
$ sudo rockcraft.skopeo --insecure-policy copy oci-archive:chiselled-hello_latest_
amd64.rock oci:chiselled-hello.oci:latest
# Unpack that to a directory
$ sudo apt install oci-image-tool
$ oci-image-tool  unpack --ref name=latest chiselled-hello.oci /tmp/chiselled-
hello
```

Exposing that directory to a guest via `virtiofsd`:

```
$ sudo apt install virtiofsd
$ /usr/libexec/virtiofsd --socket-path=/tmp/vfsd.sock --shared-dir /tmp/chiselled-
hello
...
[INFO  virtiofsd] Waiting for vhost-user socket connection...
```

To the QEMU command-line one would then add the following options:

```
...
-object memory-backend-memfd,id=mem,share=on,size=128M \
-numa node,memdev=mem -chardev socket,id=char0,path=/tmp/vfsd.sock \
-device vhost-user-fs-pci,queue-size=1024,chardev=char0,tag=myfs
...
```

Which allows the user to mount it from inside the guest via `$ mount -t virtiofs myfs /mnt` or if you want to use it as root, you can pass it via kernel parameter `rootfstype=virtiofs root=myfs rw`.

### Upgrading the machine type of your VM

Upgrading the machine type of a virtual machine (VM) can be thought of in the same way as buying (virtual) hardware of the same spec but with a newer release date. Whereas to upgrade a physical machine you might buy an improved CPU, more RAM, or increased storage, with a virtual machine you can change the configuration to achieve the same results.

### Why should you do this for a VM?

There are several reasons why you might want to update the machine type of an existing VM. For example, to:

- Improve performance with additional computing power
- Add a virtual *GPU*
- Scale up the allocated resources to cope with increased workloads
- Obtain the latest security fixes and features
- Continue using a guest created on a now-unsupported release
- Prepare for future expansion by upgrading in advance

### How does this work?

It is generally recommended to update machine types when upgrading QEMU/KVM to a new major version. However, this can likely never be an automated task as the change is "guest visible"; the guest devices might change in appearance, new features will be announced to the guest, and so on.

Linux is usually very good at tolerating such changes – but, it depends so heavily on the setup and workload of the guest that this has to be evaluated by the owner/admin of the system.

Other operating systems were known to often be severely impacted by changing the hardware. Consider a machine type change as similar to replacing all devices and firmware of a physical machine to the latest revision. **All** of the considerations that apply to firmware upgrades apply to evaluating a machine type upgrade as well.

### Backing up guest definitions

As usual, with major configuration changes it is wise to back up your guest definition and disk state to be able to do a rollback – just in case something goes wrong.

### Upgrade the machine type

There is no integrated single command to update the machine type via `virsh` or similar tools. It is a normal part of your machine definition, and therefore updated the same way as most others.

### Shut down the VM

First shutdown your machine and wait until it has reached that state:

```
virsh shutdown <your_machine>
```

You can check the status of the machine with the following command:

```
virsh list --inactive
```

### Edit the guest definition

Once the machine is listed as “shut off”, you can then edit the machine definition and find the type in the `type` tag given at the machine attribute.

```
virsh edit <your_machine>
<type arch='x86_64' machine='pc-i440fx-bionic'>hvm</type>
```

Change this to the value you want. If you need to check what machine types are available via the `kvm -M ?` command first, then note that while upstream types are provided for convenience, only Ubuntu types are supported. There you can also see what the current default would be, as in this example:

```
$ kvm -M ?
pc-i440fx-xenial       Ubuntu 16.04 PC (i440FX + PIIX, 1996) (default)
...
pc-i440fx-bionic       Ubuntu 18.04 PC (i440FX + PIIX, 1996) (default)
...
```

We strongly recommend that you change to newer types (if possible), not only to take advantage of newer features, but also to benefit from bug fixes that only apply to the newer device virtualisation.

### Restart the guest

After this you can start your guest again. You can check the current machine type from guest and host depending on your needs.

```
virsh start <your_machine>
# check from host, via dumping the active xml definition
virsh dumpxml <your_machine> | xmllint --xpath "string(//domain/os/type/@machine)"
-
# or from the guest via dmidecode (if supported)
sudo dmidecode | grep Product -A 1
        Product Name: Standard PC (i440FX + PIIX, 1996)
        Version: pc-i440fx-bionic
```

If you keep non-live definitions around – such as `.xml` files – remember to update those as well.

### Further reading

- This process is also documented along with some more constraints and considerations at the Ubuntu Wiki

### Containers

- *Container tools overview* gives more details about the popular container technologies available in Ubuntu

There are more detailed discussions on specific technologies here:

- *About rock images*
- *Docker storage, networking, and logging*

### Container tools in the Ubuntu space

Let's take a look at some of the most commonly used tools and technologies available in the Ubuntu container space.

#### LXC

**Container type**: System containers

Linux Containers, or LXC (pronounced "lex-see"), is a program that creates and administers containers on your local system. It is the foundation of several other system container technologies and provides both an API (to allow higher-level managers like LXD to administer containers), and an interface through which the user can interact with kernel containment features (often called the "userspace interface"). LXC interacts directly with the kernel to isolate processes, resources, etc, and provides the necessary tools - and a container runtime - for creating and managing system containers.

To get started with LXC containers, check out the LXC Introduction and LXC Getting started pages.

#### LXD

**Container type**: System containers

The Linux Containers Daemon, or LXD (pronounced "lex-dee") is the lightervisor, or lightweight container hypervisor. It is a system container management tool built on top of LXC. Since it is an abstraction layer away from LXC it offers a more user-friendly interface, including both a REST API and a command-line interface. The LXD API deals with "remotes", which serve images and containers. In fact, it comes with a built-in image store, so that containers can be created more quickly.

To get started with LXD from an Ubuntu Server administrator's point of view, check out our *how to get started with LXD* guide. For a more general beginner's introduction to LXD, we recommend this tutorial from the LXD team.

In addition to creating and managing containers, LXD can also be used to create virtual machines.

#### Docker

**Container type**: Application containers

Docker is one of the most popular containerization platforms, which allows developers to package applications - together with their dependencies - into lightweight containers. This provides a consistently reproducible environment for deploying applications, which makes

it easy to build, ship, and run them even in different environments. Docker includes a command-line interface and a daemon to create and manage containers.

Although Docker is widely used by developers, it can also be used by system administrators to manage resources and applications. For instance, by encapsulating applications (and their libraries and dependencies) in a single package, and providing version control, deployment of software and updates can be simplified. It also helps to optimise resource use - particularly through its alignment with microservices architecture.

To get started with Docker from a system administrator's point of view, check out our *Docker guide for sysadmins*.

### About rock images

A rock is an Ubuntu-based container image. Rocks are OCI-compliant and thus compatible with all popular container management tools (such as *Docker* and Kubernetes) and container registries (such as Docker Hub and Amazon ECR). They are built to be secure and stable by design.

Rocks are created using Rockcraft, which in turn uses Chisel to extract the relevant parts of Debian packages needed to form a minimal container image. By keeping rocks small and specific, their exposure to vulnerabilities is minimised.

Although rocks can be useful for anyone using containers, in the Ubuntu Server context they are particularly helpful for system administrators who want to use containers to manage their infrastructure. To find out how to run rocks on your server, refer to *our how-to guide*. Alternatively, to find out more about rocks and Rockcraft, refer to the official Rockcraft documentation.

### Docker storage, networking, and logging

Containers are widely used across multiple server workloads (databases and web servers, for instance), and understanding how to properly set up your server to run them is becoming more important for systems administrators. In this explanatory page, we are going to discuss some of the most important factors a system administrator needs to consider when setting up the environment to run Docker containers.

Understanding the options available to run Docker containers is key to optimising the use of computational resources in a given scenario/workload, which might have specific requirements. Some aspects that are important for system administrators are: **storage**, **networking** and **logging**.

### Storage

The first thing we need to keep in mind is that containers are ephemeral, and unless configured otherwise, so are their data. Docker images are composed of one or more layers which are read-only, and once you run a container based on an image a new writable layer is created on top of the topmost image layer; the container can manage any type of data there. The content changes in the writable container layer are not persisted anywhere, and once the container is gone all the changes disappear. This behaviour presents some challenges to us: How can the data be persisted? How can it be shared among containers? How can it be shared between the host and the containers?

There are some important concepts in the Docker world that are the answer for some of

those problems: they are **volumes**, **bind mounts** and **tmpfs**. Another question is how all those layers that form Docker images and containers will be stored, and for that we are going to talk about **storage drivers** (more on that later).

When we want to persist data we have two options:

- Volumes are the preferred way to persist data generated and used by Docker containers if your workload will generate a high volume of data, such as a database.
- Bind mounts are another option if you need to access files from the host, for example system files.

If what you want is to store some sensitive data in memory, like credentials, and do not want to persist it in either the host or the container layer, we can use tmpfs mounts.

### Volumes

The recommended way to persist data to and from Docker containers is by using volumes. Docker itself manages them, they are not OS-dependent and they can provide some interesting features for system administrators:

- Easier to back up and migrate when compared to bind mounts;
- Managed by the Docker CLI or API;
- Safely shared among containers;
- Volume drivers allow one to store data in remote hosts or in public cloud providers (also encrypting the data).

Moreover, volumes are a better choice than persisting data in the container layer, because volumes do not increase the size of the container, which can affect the life-cycle management performance.

Volumes can be created before or at the container creation time. There are two CLI options you can use to mount a volume in the container during its creation (`docker run` or `docker create`):

- `--mount`: it accepts multiple key-value pairs (`<key>=<value>`). This is the preferred option to use.
  - `type`: for volumes it will always be `volume`;
  - `source` or `src`: the name of the volume, if the volume is anonymous (no name) this can be omitted;
  - `destination`, `dst` or `target`: the path inside the container where the volume will be mounted;
  - `readonly` or `ro` (optional): whether the volume should be mounted as read-only inside the container;
  - `volume-opt` (optional): a comma separated list of options in the format you would pass to the `mount` command.
- `-v` or `--volume`: it accepts 3 parameters separated by colon (`:`):
  - First, the name of the volume. For the default `local` driver, the name should use only: letters in upper and lower case, numbers, `.`, `_` and `-`;
  - Second, the path inside the container where the volume will be mounted;

  - Third (optional), a comma-separated list of options in the format you would pass to the `mount` command, such as `rw`.

### Bind mounts

Bind mounts are another option for persisting data, however, they have some limitations compared to volumes. Bind mounts are tightly associated with the directory structure and with the OS, but performance-wise they are similar to volumes in Linux systems.

In a scenario where a container needs to have access to any host system's file or directory, bind mounts are probably the best solution. Some monitoring tools make use of bind mounts when executed as Docker containers.

Bind mounts can be managed via the Docker CLI, and as with volumes there are two options you can use:

- `--mount`: it accepts multiple key-value pairs (`<key>=<value>`). This is the preferred option to use.
  - `type`: for bind mounts it will always be `bind`;
  - `source` or `src`: path of the file or directory on the host;
  - `destination`, `dst` or `target`: container's directory to be mounted;
  - `readonly` or `ro` (optional): the bind mount is mounted in the container as read-only;
  - `volume-opt` (optional): it accepts any `mount` command option;
  - `bind-propagation` (optional): it changes the bind propagation. It can be `rprivate`, `private`, `rshared`, `shared`, `rslave`, `slave`.
- `-v` or `--volume`: it accepts 3 parameters separated by colon (`:`):
  - First, path of the file or directory on the host;
  - Second, path of the container where the volume will be mounted;
  - Third (optional), a comma separated of option in the format you would pass to `mount` command, such as `rw`.

### Tmpfs

Tmpfs mounts allow users to store data temporarily in RAM memory, not in the host's storage (via bind mount or volume) or in the container's writable layer (with the help of storage drivers). When the container stops, the tmpfs mount will be removed and the data will not be persisted in any storage.

This is ideal for accessing credentials or security-sensitive information. The downside is that a tmpfs mount cannot be shared with multiple containers.

Tmpfs mounts can be managed via the Docker CLI with the following two options:

- `--mount`: it accepts multiple key-value pairs (`<key>=<value>`). This is the preferred option to use.
  - `type`: for volumes it will always be `tmpfs`;
  - `destination`, `dst` or `target`: container's directory to be mounted;

  - `tmpfs-size` and `tmpfs-mode` options (optional). For a full list see the Docker documentation.

- `--tmpfs`: it accepts no configurable options, just mount the tmpfs for a standalone container.

### Storage drivers

Storage drivers are used to store image layers and to store data in the writable layer of a container. In general, storage drivers are implemented trying to optimise the use of space, but write speed might be lower than native *filesystem* performance depending on the driver in use. To better understand the options and make informed decisions, take a look at the Docker documentation on how layers, images and containers work.

The default storage driver is the `overlay2` which is backed by `OverlayFS`. This driver is recommended by upstream for use in production systems. The following storage drivers are available and are supported in Ubuntu (as at the time of writing):

- **OverlayFS**: it is a modern union filesystem. The Linux kernel driver is called `OverlayFS` and the Docker storage driver is called `overlay2`. **This is the recommended driver**.
- **ZFS**: it is a next generation filesystem that supports many advanced storage technologies such as volume management, snapshots, checksumming, compression and *deduplication*, replication and more. The Docker storage driver is called `zfs`.
- **Btrfs**: it is a copy-on-write filesystem included in the Linux kernel mainline. The Docker storage driver is called `btrfs`.
- **Device Mapper**: it is a kernel-based framework that underpins many advanced volume management technologies on Linux. The Docker storage driver is called `devicemapper`.
- **VFS**: it is not a union filesystem, instead, each layer is a directory on disk, and there is no copy-on-write support. To create a new layer, a "deep copy" is done of the previous layer. This driver does not perform well compared to the others, however, it is robust, stable and works in any environment. The Docker storage driver is called `vfs`.

The storage drivers accept some options via `storage-opts`, check the storage driver documentation for more information. Keep in mind that this is a *JSON* file and all lines should end with a comma (`,`) except the last one.

### Networking

Networking in the context of containers refers to the ability of containers to communicate with each other and with non-Docker workloads. The Docker networking subsystem was implemented in a pluggable way, and we have different network drivers available to be used in different scenarios:

- **Bridge**: This is the default network driver. This is widely used when containers need to communicate among themselves in the same host.
- **Overlay**: It is used to make containers managed by different docker daemons (different hosts) communicate among themselves.
- **Host**: It is used when the networking isolation between the container and the host is not desired, the container will use the host's networking capabilities directly.
- **IPvlan**: It is used to provide full control over the both IPv4 and IPv6 addressing.

- **Macvlan**: It is used to allow the assignment of Mac addresses to containers, making them appear as a physical device in the network.
- **None**: It is used to make the container completely isolated from the host.

### Logging

Monitoring what is happening in the system is a crucial part of systems administration, and with Docker containers it is no different. Docker provides the logging subsystem (which is pluggable) and there are many drivers that can forward container logs to a file, an external host, a database, or another logging back-end. The logs are basically everything written to `STDOUT` and `STDERR`. When building a Docker image, the relevant data should be forwarded to those I/O stream devices.

The following logging drivers are available (at the time of writing):

- **json-file**: it is the default logging driver. It writes logs in a file in JSON format.
- **local**: write logs to an internal storage that is optimised for performance and disk use.
- **journald**: send logs to systemd journal.
- **syslog**: send logs to a syslog server.
- **logentries**: send container logs to the Logentries server.
- **gelf**: write logs in a *Graylog* Extended Format which is understood by many tools, such as Graylog, Logstash, and Fluentd.
- **awslogs**: send container logs to Amazon CloudWatch Logs.
- **etwlogs**: forward container logs as ETW events. ETW stands for Event Tracing in Windows, and is the common framework for tracing applications in Windows. Not supported in Ubuntu systems.
- **fluentd**: send container logs to the Fluentd collector as structured log data.
- **gcplogs**: send container logs to Google Cloud Logging Logging.
- **splunk**: sends container logs to HTTP Event Collector in Splunk Enterprise and Splunk Cloud.

### Resources

To get a hands-on tutorial on using Docker for storage, networking, and logging, see:

- *Docker for system admins*

### Other virtualisation tools

- *About OpenStack* provides an introduction to OpenStack
- *Introduction to eBPF* discusses eBPF, which is a lightweight VM that runs in the kernel space

## About OpenStack

OpenStack is the most popular open source cloud computing platform that enables the management of distributed compute, network and storage resources in the data centre.

While the reference virtualisation stack (consisting of *QEMU/KVM* and *libvirt*) enables hardware virtualisation and the management of virtual machines (VMs) on a single host, in most cases the computing, network and storage resources are distributed across multiple hosts in the data centre.

This creates an obvious challenge with centralised management of those resources, scheduling VMs, etc. OpenStack solves this problem by aggregating distributed pools of resources, allocating them to VMs on-demand and enabling automated VM provisioning through a self-service portal.

OpenStack consists of the following primary components:

- **Keystone**: Serves as an identity service, providing authentication and authorisation functions for the users and enabling multi-tenancy.
- **Glance**: This is an image service, responsible for uploading, managing and retrieving cloud images for VMs running on OpenStack.
- **Nova**: This is the primary compute engine of OpenStack, responsible for VM scheduling, creation and termination.
- **Neutron**: Provides network connectivity between VMs, enabling multi-VM deployments.
- **Cinder**: This is a storage component that is responsible for provisioning, management and termination of persistent block devices.
- **Swift**: This is another storage component that provides a highly available and scalable object storage service.

There are also many other OpenStack components and supporting services available in the OpenStack ecosystem, enabling more advanced functions, such as load balancing, secrets management, etc.

## OpenStack installation

The most straightforward way to get started with OpenStack on Ubuntu is to use MicroStack since the entire installation process requires only 2 commands and takes around 20 minutes.

Apart from MicroStack, multiple different installation methods for OpenStack on Ubuntu are available. These include:

- OpenStack Charms
- OpenStack Ansible
- Manual Installation
- DevStack

## Introduction to eBPF

eBPF is a powerful tool for server and system administrators, often described as a lightweight, sandboxed virtual machine within the kernel. It is commonly used for performance monitoring, security, and network traffic processing without the need to modify or rebuild the kernel.

Since it runs in the kernel space, there is no need for context-switching, making it very fast compared to solutions implemented in user-space. It also has access to the kernel data structures, providing more capabilities than tools limited to the interfaces exposed to user-space.

BPF, which stands for "Berkeley Packet Filter", was originally designed to perform network packet filtering. Over time, it has evolved into *extended Berkeley Packet Filter* (eBPF), a tool which contains many additional capabilities, including the use of more registers, support for 64-bit registers, data stores (Maps), and more.

As a result, eBPF has been extended beyond the kernel networking subsystem and it not only enhances the networking experience, but also provides tracing, profiling, observability, security, etc. The terms eBPF and BPF are used interchangeably but both refer to eBPF now.

## How eBPF works

User-space applications can load eBPF programs into the kernel as eBPF bytecode. Although you could write eBPF programs in bytecode, there are several tools which provide abstraction layers on top of eBPF so you do not need to write bytecode manually. These tools will then generate the bytecode which will in turn be loaded into the kernel.

Once an eBPF program is loaded into the kernel, it is then verified by the kernel before it can run. These checks include:

- verifying if the eBPF program halts and will never get stuck in a loop,
- verifying that the program will not crash by checking the registers and stack state validity throughout the program, and
- ensuring the process loading the eBPF program has all the capabilities required by the eBPF program to run.

After the verification step, the bytecode is Just-In-Time (JIT) compiled into machine-code to optimize the program's execution.

## eBPF in Ubuntu Server

Since Ubuntu 24.04, `bpftrace` and `bpfcc-tools` (the BPF Compiler Collection or BCC) are available in every Ubuntu Server installation by default as part of our efforts to enhance the application developer's and sysadmin's experience in Ubuntu.

In Ubuntu, you can use tools like the BCC to identify bottlenecks, investigate performance degradation, trace specific function calls, and create custom monitoring tools to collect data on specific kernel or user-space processes without disrupting running services.

Both `bpftrace` and `bpfcc-tools` install sets of tools to handle these different functionalities. Apart from the `bpftrace` tool itself, you can fetch a comprehensive list of these tools with the following command:

```
$ dpkg -L bpftrace bpfcc-tools | grep -E '/s?bin/.*$' | xargs -n1 basename
```

Most of the tools listed above are quite well documented. Their manpages usually include good examples you can try immediately. The tools ending in `.bt` are installed by `bpftrace` and refer to `bpftrace` scripts (they are text files, hence, you could read them to understand how they are achieving specific tasks). The ones ending in `-bpfcc` are BCC tools (from `bpfcc`) written in Python (you can also inspect those as you would inspect the `.bt` files).

These `bpftrace` scripts often demonstrate how a complex task can be achieved in simple ways. The `-bpfcc` variants are often a bit more advanced, often providing more options and customizations.

You will also find several text files describing use-case examples and the output for `bpftrace` tools in `/usr/share/doc/bpftrace/examples/` and for `bpfcc-tools` in `/usr/share/doc/bpfcc-tools/examples/`.

For instance,

```
# bashreadline.bt
```

will print bash commands for all running bash shells in your system. Since the information you can get via eBPF may be confidential, you will need to run any of it as root. You may notice that there is also a `bashreadline-bpfcc` tool available from `bpfcc-tools`. Both of them provide similar features. The former is implemented in python with BCC while the latter is a `bpftrace` script, as described above. You will see that many of these tools have both a `-bpfcc` and a `.bt` version. Do read their manpages (and perhaps the scripts) to choose which suits you best.

### Example - Determine what commands are executed

One tool that is trivial but powerful is `execsnoop-bpfcc`, which allows you to answer common questions that should not be common - like "what other programs is this action eventually calling?". It can be used to determine if a program calls another tool too often or calls something that you'd not expect.

It has become a common way to help you understand what happens when a maintainer script is executed in a .deb package in Ubuntu.

For this task, you'd run `execsnoop-bpfcc` with the following arguments:

- `-Uu root` - to reduce the noisy output only to things done in root context (like here the package install)
- `-T` - to get time info along the log

```
# In one console run:
$ sudo execsnoop-bpfcc -Uu root -T

# In another trigger what you want to watch for
$ sudo apt install --reinstall vim-nox

# Execsnoop in the first console will now report probably more than you expected:
TIME     UID   PCOMM            PID     PPID    RET ARGS
10:58:07 1000  sudo             1323101 1322857   0 /usr/bin/sudo apt install --
reinstall vim-nox
10:58:10 0     apt              1323107 1323106   0 /usr/bin/apt install --
reinstall vim-nox
```

(continues on next page)

(continued from previous page)

```
10:58:10 0     dpkg             1323108 1323107   0 /usr/bin/dpkg --print-foreign-
architectures
...
10:58:12 0     sh               1323134 1323107   0 /bin/sh -c /usr/sbin/dpkg-
preconfigure --apt || true
...
10:58:13 0     tar              1323155 1323152   0 /usr/bin/tar -x -f  --
warning=no-timestamp
10:58:14 0     vim-nox.prerm    1323157 1323150   0 /var/lib/dpkg/info/vim-nox.
prerm upgrade 2:9.1.0016-1ubuntu7.3
10:58:14 0     dpkg-deb         1323158 1323150   0 /usr/bin/dpkg-deb --fsys-
tarfile /var/cache/apt/archives/vim-nox_2%3a9.1.0016-1ubuntu7.3_amd64.deb
...
10:58:14 0     update-alternat  1323171 1323163   0 /usr/bin/update-alternatives -
-install /usr/bin/vimdiff vimdiff /usr/bin/vim.nox 40
...
10:58:17 0     snap             1323218 1323217   0 /usr/bin/snap advise-snap --
from-apt
10:58:17 1000  git              1323224 1323223   0 /usr/bin/git rev-parse --
abbrev-ref HEAD
10:58:17 1000  git              1323226 1323225   0 /usr/bin/git status --
porcelain
10:58:17 1000  vte-urlencode-c  1323227 1322857   0 /usr/libexec/vte-urlencode-cwd
```

### eBPF can be modified to your needs

Let us look at another practical application of eBPF. This example is meant to show another use-case with eBPF then evolve this case into a more complex one by modifying it.

### Example - Find out which files QEMU is loading

Let's say you want to verify which binary files are loaded when running a particular QEMU command. QEMU is a truly complex program and sometimes it can be hard to make the connection from a command line to the files used from /usr/share/qemu. This is hard to determine when you define the QEMU command line, but becomes problematic when more useful layers of abstraction are used like libvirt or LXD or even things on top like OpenStack.

While `strace` could be used, it would add additional overhead to the investigation process since `strace` uses `ptrace`, and context switching may be required. Furthermore, if you need to monitor a system to produce this answer, especially on a host running many VMs, `strace` quickly reaches its limits.

Instead, `opensnoop` could be used to trace `open()` syscalls. In this case, we use `opensnoop-bpfcc` to have more parameters to tune it to our needs. The example will use the following arguments:

- `--full-path` - Show full path for open calls using a relative path.
- `--name qemu-system-x86` - only care about files opened by QEMU; The mindful reader will wonder why this isn't qemu-system-x86_64, but you'd see in unfiltered output of opensnoop that it is length limited, so only the shorter qemu-system-x86 can be used.

```
# This will collect a log of files opened by QEMU
$ sudo /usr/sbin/opensnoop-bpfcc --full-path --name qemu-system-x86
#
# If now you in another console or anyone on this system in general runs QEMU,
# this would log the files opened
#
# For example calling LXD for an ephemeral VM
$ lxc launch ubuntu-daily:n n-vm-test --ephemeral --vm
#
# Will in opensnoop deliver a barrage of files opened
1308728 qemu-system-x86    -1   2 PID    COMM               FD ERR PATH
/snap/lxd/current/zfs-2.2/lib/glibc-hwcaps/x86-64-v3/libpixman-1.so.0
1308728 qemu-system-x86    -1   2 /snap/lxd/current/zfs-2.2/lib/glibc-hwcaps/x86-
64-v2/libpixman-1.so.0
1308728 qemu-system-x86    -1   2 /snap/lxd/current/zfs-2.2/lib/tls/haswell/x86_
64/libpixman-1.so.0
...
1313104 qemu-system-x86    58   0 /sys/dev/block/230:16/queue/zoned
1313104 qemu-system-x86    20   0 /dev/fd/4
```

Of course the QEMU process opens plenty of things: shared libraries, config files, entries in /{sys,dev,proc}, and much more. But with opensnoop-bpfcc, we can see them all as they happen across the whole system.

### Focusing on a specific file type

Imagine you only wanted to verify which .bin files this is loading. Of course, we could just use grep on the output, but this whole section is about showing eBPF examples to get you started. So here we make the simplest change – modifying the python wrapper around the tracing eBPF code. Once you understand how to do this, you can go further in adapting them to your own needs by delving into the eBPF code itself, and from there to create your very own eBPF solutions from scratch.

So while opensnoop-bpfcc as of right now has no option to filter on the file names, it could …

```
$ sudo cp /usr/sbin/opensnoop-bpfcc /usr/sbin/opensnoop-bpfcc.new
$ sudo vim /usr/sbin/opensnoop-bpfcc.new
...
$ diff -Naur /usr/sbin/opensnoop-bpfcc /usr/sbin/opensnoop-bpfcc.new
--- /usr/sbin/opensnoop-bpfcc        2024-11-12 09:15:17.172939237 +0100
+++ /usr/sbin/opensnoop-bpfcc.new        2024-11-12 09:31:48.973939968 +0100
@@ -40,6 +40,7 @@
     ./opensnoop -u 1000               # only trace UID 1000
     ./opensnoop -d 10                 # trace for 10 seconds only
     ./opensnoop -n main               # only print process names containing
"main"
+    ./opensnoop -c path               # only print paths containing "fname"
     ./opensnoop -e                    # show extended fields
     ./opensnoop -f O_WRONLY -f O_RDWR  # only print calls for writing
     ./opensnoop -F                    # show full path for an open file with
relative path
```

(continues on next page)

(continued from previous page)

```
@@ -71,6 +72,9 @@
 parser.add_argument("-n", "--name",
     type=ArgString,
     help="only print process names containing this name")
+parser.add_argument("-c", "--contains",
+    type=ArgString,
+    help="only print paths containing this string (implies --full-path)")
 parser.add_argument("--ebpf", action="store_true",
     help=argparse.SUPPRESS)
 parser.add_argument("-e", "--extended_fields", action="store_true",
@@ -83,6 +87,8 @@
     help="size of the perf ring buffer "
         "(must be a power of two number of pages and defaults to 64)")
 args = parser.parse_args()
+if args.contains is not None:
+    args.full_path = True
 debug = 0
 if args.duration:
     args.duration = timedelta(seconds=int(args.duration))
@@ -440,6 +446,12 @@
         if args.name and bytes(args.name) not in event.comm:
             skip = True

+        paths = entries[event.id]
+        paths.reverse()
+        entire_path = os.path.join(*paths)
+        if args.contains and bytes(args.contains) not in entire_path:
+            skip = True
+
         if not skip:
             if args.timestamp:
                 delta = event.ts - initial_ts
@@ -458,9 +470,7 @@
             if not args.full_path:
                 printb(b"%s" % event.name)
             else:
-                paths = entries[event.id]
-                paths.reverse()
-                printb(b"%s" % os.path.join(*paths))
+                printb(b"%s" % entire_path)

         if args.full_path:
             try:
```

Running the modified version now allows you to probe for specific file names, like all the `.bin` files:

```
$ sudo /usr/sbin/opensnoop-bpfcc.new --contains '.bin' --name qemu-system-x86
PID    COMM               FD ERR PATH
```

(continues on next page)

(continued from previous page)

```
1316661 qemu-system-x86    21   0 /snap/lxd/current/share/qemu//kvmvapic.bin
1316661 qemu-system-x86    39   0 /snap/lxd/current/share/qemu//vgabios-virtio.bin
1316661 qemu-system-x86    39   0 /snap/lxd/current/share/qemu//vgabios-virtio.bin
```

### Use for other purposes, the limit is your imagination

And just like with all the other tools and examples, the limit is your imagination. Wanted to know which files in `/etc` your complex intertwined apache config is really loading?

```
$ sudo /usr/sbin/opensnoop-bpfcc.new --name 'apache2' --contains '/etc'
PID     COMM               FD ERR PATH
1319357 apache2             3   0 /etc/apache2/apache2.conf
1319357 apache2             4   0 /etc/apache2/mods-enabled
1319357 apache2             4   0 /etc/apache2/mods-enabled/access_compat.load
...
1319357 apache2             4   0 /etc/apache2/ports.conf
1319357 apache2             4   0 /etc/apache2/conf-enabled
1319357 apache2             4   0 /etc/apache2/conf-enabled/charset.conf
1319357 apache2             4   0 /etc/apache2/conf-enabled/localized-error-pages.
conf
...
1319357 apache2             4   0 /etc/apache2/sites-enabled/000-default.conf
```

### eBPF's limitations

When you read the example code change above, it is worth noticing - just like the existing `--name` option - this is filtering on the reporting side, not on event generation. So you should be aware and understand why you might receive eBPF messages like:

```
Possibly lost 84 samples
```

Since eBPF programs produce events on a ring buffer, if the rate of events exceeds the pace that the userspace process can consume the events, a program will lose some events (overwritten since it's a ring). The "Possibly lost .. samples" message hints that this is happening. This is conceptually the same for almost all kernel tracing facilities. Since they are not allowed to slow down the kernel, you can't tell the tool to "wait until I've consumed". Most of the time this is fine, but advanced users might need to aggregate on the eBPF side to reduce what needs to be picked up by the userspace. And despite having lower overhead, eBPF tools still need to find their balance between buffering, dropping events and consuming CPU. See the same discussion in the examples of tools shown above.

### Conclusion

eBPF offers a vast array of options to monitor, debug, and secure your systems directly in kernel-space (i.e., fast and omniscient), with no need to disrupt running services. It is an invaluable tool for system administrators and software engineers.

#### References

- Introduction to eBPF video, given at the Ubuntu summit 2024, eventually presenting an eBPF based framework for Kubernetes called Inspector Gadget.
- For a deeper introduction into eBPF concepts consider reading what is eBPF by the eBPF community.
- For a complete documentation on eBPF, its internals and interfaces, please check the upstream documentation.
- The eBPF community also has a list of eBPF based solutions, many of which are related to the Kubernetes ecosystem.

#### See also

- How-to: *Virtualisation*
- How-to: *Containers*

## 4.7. Clouds

Our *clouds section* provides details on finding cloud images for various public clouds, and about the popular cloud initialization tool, cloud-init.

### 4.7.1. Clouds

Clouds are networks of remote servers providing services via the internet. They have become a popular way to manage storage, software and processing without the need to host physical servers yourself.

#### Cloud-init

Cloud-init is the industry standard tool for provisioning clouds and is supported by most cloud providers. Here we provide a high-level introduction to the tool.

#### Introduction to cloud-init

Managing and configuring cloud instances and servers can be a complex and time-consuming task. Cloud-init is the industry-standard open source tool designed to automate getting systems up and running with preconfigured settings in a repeatable way across instances and platforms.

Although it's commonly used to automatically configure public or private cloud instances, it can also be used to deploy virtual machines and physical machines on a network. By automating all the routine setup tasks, systems can be initialized efficiently and reliably, whether you're a developer spinning up virtual machines or containers, or a system administrator managing infrastructure.

#### How does it work?

Cloud-init works by taking the initial configuration that you supply, and applying it at boot time so that when the instance is launched it's already configured the way you want.

Your configuration can be used and re-used as often as you want to get the exact same VM environment every time, or to deploy an entire fleet of machines in exactly the same way. You get consistent results with a fraction of the time and effort it would take to do so manually.

It can handle a range of tasks that normally happen when a new instance is created, such as setting the *hostname*, configuring network interfaces, creating user accounts, and even running custom scripts.

### Related products

Cloud-init can automatically detect the source platform it is being run on (the datasource). It is widely-supported and works with:

- Most public cloud offerings (including Amazon EC2, Azure, Google Compute Engine)
- Private clouds
- MAAS and OpenStack
- Common virtualization and VM software such as LXD, libvirt, and QEMU

You can also use it on other Linux distributions (such as RedHat, OpenSUSE, and Alpine), or in concert with popular configuration managers (like Ansible, Chef, and Puppet). It even supports the Windows Subsystem for Linux (WSL)!

To learn more about cloud-init and try it out for yourself, check out their tutorials.

### Cloud images

Whichever cloud you use, you will need to use images compatible with that cloud. We call these **cloud images**.

- *Cloud images overview* provides a high level introduction to the concept, along with links to the official sources of cloud images.

For information related to public clouds in Ubuntu, you may also want to refer to the official Public Cloud documentation.

For ease of reference, these links in the Public Cloud documentation will take you to the cloud images for:

- Amazon EC2 | Google Compute Engine (GCE) | Microsoft Azure

### Cloud images

Canonical produces a variety of cloud-specific images, which are available directly via the clouds themselves, as well as on https://cloud-images.ubuntu.com.

For expanded documentation, please see the separate public-cloud documentation.

### Public clouds

### Compute offerings

Users can find Ubuntu images for virtual machines and bare-metal offerings published directly to the following clouds:

- Amazon Elastic Compute Cloud (EC2)
- Google Compute Engine (GCE)
- IBM Cloud

- Microsoft Azure
- Oracle Cloud

### Container offerings

Ubuntu images are also produced for a number of container offerings:

- Amazon Elastic Kubernetes Service (EKS)
- *Google Kubernetes Engine (GKE)*

### Private clouds

On cloud-images.ubuntu.com, users can find standard and minimal images for the following:

- Hyper-V
- KVM
- OpenStack
- Vagrant
- VMware

### Release support

Cloud images are published and supported throughout the lifecycle of an Ubuntu release. During this time images can receive all published security updates and bug fixes.

For users wanting to upgrade from one release to the next, the recommended path is to launch a new image with the desired release and then migrate any workload or data to the new image.

Some cloud image customisation must be applied during image creation, which would be missing if an in-place upgrade were performed. For that reason, in-place upgrades of cloud images are not recommended.

# 4.8. System tuning

*Our system tuning* section provides details on system performance and optimization, covering concepts like Profile-Guided Optimization (PGO) and some common tooling.

## 4.8.1. Performance

This area of the documentation is about system tuning and will list various tools that can either help to determine the state of the system or to tune it for a given workload.

**Note**

Disclaimer - To tune you need to know your system and workload

Almost all tunable parameters can be good for one and bad for another type of workload or environment. If the system could do it for you, it probably would be the default setting already.

Not even the goal of tuning is the same for everyone; do you want to improve latency, throughput, thermal or work-unit-per-power? But those default settings generally have

to aim for a good compromise on all of these aspects, no matter what you will do with your system.

Therefore you have to know your workload, your system and the important metrics of what you want to achieve. The more you know, the more you'll be able to improve your system it to suit your needs by:

- Identifying bottlenecks as unblocking them usually has the biggest impact for your workload
- Identifying where your needs are different from the generic assumptions to change related tunables
- Identifying architectural mismatches between the solution and your needs to allow adapting

To identify those aspects and to then apply static or dynamic tuning Ubuntu carries various tools, a few of them are outlined in more detail in the following sections.

- Profile-Guided Optimization *PGO*
- Obtain the hierarchical map of key computing elements using *hwloc and lstopo*
- Check and control CPU governors, power and frequency with *cpupower*
- Dynamic, adaptive system tuning *with TuneD*
- Energy Performance Preference *EPP*

### Profile-Guided Optimization

Profile-guided optimization (PGO) or feedback-driven optimization (FDO) are synonyms for the same concept. The idea is that a binary is compiled with built-in instrumentation which, when executed, will generate a profile. This profile can be used as input for the compiler in a subsequent build of the same binary, and it will serve as a guide for further optimizations.

It can be hard to do profiling of real world applications. Ideally, the profile should be generated by a representative workload of the program, but it's not always possible to simulate a representative workload. Moreover, the built-in instrumentation impacts the overall performance of the binary which introduces a performance penalty.

In order to address these problems, we can use tools like `perf` to "observe" what the binary is doing externally (sampling it, by monitoring events using Linux kernel's Performance Monitoring Unit – PMU), which makes the process more suitable to be used in production environments. This technique works better than the regular built-in instrumentation, but it still has a few drawbacks that we will expand later.

### Caveats

- The purpose of this guide is to provide some basic information about what PGO is and how it works. In order to do that, we will look at a simple example using OpenSSL (more specifically, the `openssl speed` command) and learn how to do basic profiling. We will not go into a deep dive on how to build the project, and it is assumed that the reader is comfortable with compilation, compiler flags and using the command line.

- Despite being a relatively popular technique, PGO is not always the best approach to optimize a program. The profiling data generated by the workload will be extremely tied to it, which means that the optimized program might actually have worse performance when other types of workloads are executed. There is not a one-size-fits-all solution for this problem, and sometimes the best approach might be to **not** use PGO after all.
- If you plan to follow along, we recommend setting up a test environment for this experiment. The ideal setup involves using a bare metal machine because it's the more direct way to collect the performance metrics. If you would like to use a virtual machine (created using QEMU/libvirt, LXD, Multipass, etc.), it will likely only work on Intel-based processors due to how Virtual Performance Monitoring Unit (vPMU) works.

### `perf` and AutoFDO

Using `perf` to monitor a process and obtain data about its runtime workload produces data files in a specific binary format that we will call `perfdata`. Unfortunately, GCC doesn't understand this file format; instead, it expects a profile file in a format called `gcov`. To convert a `perfdata` file into a `gcov` one, we need to use a software called `autofdo`. This software expects the binary being profiled to obey certain constraints:

- The binary **cannot** be stripped of its debug symbols. `autofdo` does not support separate debug information files (i.e., it can't work with Ubuntu's `.ddeb` packages), and virtually all Ubuntu packages run `strip` during their build in order to generate the `.ddeb` packages.
- The debug information file(s) **cannot** be processed by `dwz`. This tool's purpose is to compress the debug information generated when building a binary, and again, virtually all Ubuntu packages use it. For this reason, it is currently not possible to profile most Ubuntu packages without first rebuilding them to disable `dwz` from running.
- We must be mindful of the options we pass to `perf`, particularly when it comes to recording branch prediction events. The options will likely vary depending on whether you are using an Intel or *AMD* processor, for example.

On top of that, the current `autofdo` version in Ubuntu (`0.19-3build3`, at the time of this writing) is not recent enough to process the `perfdata` files we will generate. There is a PPA with a newer version of `autofdo` package for Ubuntu Noble. If you are running another version of Ubuntu and want to install a newer version of `autofdo`, you will need to build the software manually (please refer to the upstream repository for further instructions).

### A simple PGO scenario: `openssl speed`

PGO makes more sense when your software is CPU-bound, i.e., when it performs CPU intensive work and is not mostly waiting on I/O, for example. Even if your software spends time waiting on I/O, using PGO might still be helpful; its effects would be less noticeable, though.

OpenSSL has a built-in benchmark command called `openssl speed`, which tests the performance of its cryptographic algorithms. At first sight this seems excellent for PGO because there is practically no I/O involved, and we are only constrained by how fast the CPU can run. It is possible, however, to encounter cases where the built-in benchmark has already been highly optimized and could cause issues; we will discuss more about this problem later. In the end, the real benefit will come after you get comfortable with our example and apply similar methods against your own software stack.

### Running OpenSSL tests

In order to measure the performance impact of PGO, it is important to perform several tests using an OpenSSL binary without PGO and then with it. The reason for the high number of repeated tests is because we want to eliminate outliers in the final results. It's also important to make sure to disable as much background load as possible in the machine where the tests are being performed, since we don't want it to influence the results.

The first thing we have to do is to do several runs of `openssl speed` *before* enabling PGO, so that we have a baseline to compare with. As explained in the sections above, `autofdo` does not handle stripped binaries nor `dwz`-processed debug information, so we need to make sure the resulting binary obeys these restrictions. See the section below for more details on that.

After confirming that everything looks OK, we are ready to start the benchmark. Let's run the command:

```
$ openssl speed -seconds 60 -evp md5 sha512 rsa2048 aes-256-cbc
```

This will run benchmark tests for `md5`, `sha512`, `rsa2048` and `aes-256-cbc`. Each test will last 60 seconds, and they will involve calculating as many cryptographic hashes from `N`-byte chunks of data (with `N` varying from 16 to 16384) with each algorithm (with the exception of `rsa2048`, whose performance is measured in signatures/verifications per second). By the end, you should see a report like the following:

```
The 'numbers' are in 1000s of bytes per second processed.
type             16 bytes     64 bytes    256 bytes   1024 bytes   8192 bytes
16384 bytes
sha512           91713.74k   366632.96k   684349.30k  1060512.22k  1260359.00k
1273277.92k
aes-256-cbc    1266018.36k  1437870.33k  1442743.14k  1449933.84k  1453336.84k
1453484.99k
md5             140978.58k   367562.53k   714094.14k   955267.21k  1060969.81k
1068296.87k
                  sign    verify    sign/s verify/s
rsa 2048 bits 0.000173s 0.000012s   5776.8  86791.9
```

### Building OpenSSL for profiling

Before we are able to profile OpenSSL, we need sure that we compile the software in a way that the generated program meets the requirements. In our case, you can use the `file` command after building the binary to make sure that it has not been stripped:

```
$ file /usr/bin/openssl
/usr/bin/openssl: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV),
dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2,
BuildID[sha1]=42a5ac797b6981bd45a9ece3f67646edded56bdb, for GNU/Linux 3.2.0, with
debug_info, not stripped
```

Note how `file` reports the binary as being `with debug_info, not stripped`. This is the output you need to look for when inspecting your own binaries.

You can use the `eu-readelf` command (from the `elfutils` package) to determine whether any compression was performed with `dwz`:

```
$ eu-readelf -S /usr/bin/openssl
[...]
Section Headers:
[Nr] Name              Type         Addr             Off      Size     ES Flags
Lk Inf Al
[...]
[31] .debug_abbrev     PROGBITS     0000000000000000 001bcbf3 000110c4  0
 0   0  1
[...]
```

If the `Flags` field has `C` (as in `compressed`) in it, this means that `dwz` was used. You will then need to figure out how to disable its execution during the build; for this particular build, this was done by overriding the `dh_dwz` rule on `debian/rules`, setting it to an empty value, and executing the build again. The particularities of how to skip build steps depend on the build system being used, and as such this guide will not go into further details.

### Using `perf` to obtain profile data

With OpenSSL prepared and ready to be profiled, it's now time to use `perf` to obtain profiling data from our workload. First, let's make sure we can actually access profiling data in the system. As `root`, run:

```
# echo -1 > /proc/sys/kernel/perf_event_paranoid
```

This should allow all users to monitor events in the system.

Then, if you are on an Intel system, you can invoke `perf` using:

```
$ sudo perf record \
        -e br_inst_retired.near_taken \
        --branch-any \
        --all-cpus \
        --output openssl-nonpgo.perfdata \
        -- openssl speed -mr -seconds 60 -evp md5 sha512 rsa2048 aes-256-cbc
```

On an AMD system, use the following invocation:

```
$ sudo perf record \
        -e 'cpu/event=0xc4,umask=0x0,name=ex_ret_brn_tkn/' \
        --branch-any \
        --all-cpus \
        --output openssl-nonpgo.perfdata \
        -- openssl speed -mr -seconds 60 -evp md5 sha512 rsa2048 aes-256-cbc
```

After the command has finished running, you should see a file named `openssl-nonpgo.perfdata` in your current directory.

Note that the only thing that differs between the Intel and AMD variants is the PMU event to be monitored. Also, note how we are using `sudo` to invoke `perf record`. This is necessary in order to obtain full access to Linux kernel symbols and relocation information. It also means that the `openssl-nonpgo.perfdata` file ownership will need to be adjusted:

```
$ sudo chown ubuntu:ubuntu openssl-nonpgo.perfdata
```

Now, we need to convert the file to gcov.

### Converting perfdata to gcov with autofdo

With autofdo installed, you can convert the perfdata generated in the last step to gcov by doing:

```
$ create_gcov \
        --binary /usr/bin/openssl \
        --gcov openssl-nonpgo.gcov \
        --profile openssl-nonpgo.perfdata \
        --gcov_version 2 \
        -use_lbr false
```

create_gcov is verbose and will display several messages that may look like something is wrong. For example, the following output is actually from a successful run of the command:

```
[WARNING:[...]/perf_reader.cc:1322] Skipping 200 bytes of metadata: HEADER_CPU_
TOPOLOGY
[WARNING:[...]/perf_reader.cc:1069] Skipping unsupported event PERF_RECORD_ID_
INDEX
[WARNING:[...]/perf_reader.cc:1069] Skipping unsupported event PERF_RECORD_EVENT_
UPDATE
[WARNING:[...]/perf_reader.cc:1069] Skipping unsupported event PERF_RECORD_CPU_MAP
[WARNING:[...]/perf_reader.cc:1069] Skipping unsupported event UNKNOWN_EVENT_17
[WARNING:[...]/perf_reader.cc:1069] Skipping unsupported event UNKNOWN_EVENT_18
[...] many more lines [...]
```

Nevertheless, you should inspect its output and also make sure to check the $? shell variable to make sure that it has finished successfully. However, even if create_gcov finishes with exit 0 the generated file openssl-nonpgo.gcov might not be valid. It is also a good idea to use the dump_gcov tool on it and make sure that it actually contains valid information. For example:

```
$ dump_gcov openssl-nonpgo.gcov
cpu_get_tb_cpu_state total:320843603 head:8557544
  2: 8442523
  7: 8442518
  7.1: 8442518
  7.4: 9357851
  8: 9357851
  10: 9357854
  19: 0
  21: 0
[...] many more lines [...]
```

If the command does not print anything, there is a problem with your gcov file.

If everything went well, we can now use the gcov file and feed it back to GCC when recompiling OpenSSL.

### Rebuilding OpenSSL with PGO

We have everything we need to rebuild OpenSSL and make use of our profile data. The most important thing now is to set the `CFLAGS` environment variable properly so that GCC can find and use the profile we generated in the previous step.

How you perform this `CFLAGS` adjustment depends on how you built the software in the first place, so we won't cover this part here. The resulting `CFLAGS` variable should have the following GCC option, though:

```
-fauto-profile=/path/to/openssl-nonpgo.gcov
```

Make sure to adjust the path to the `openssl-nonpgo.gcov` file.

Also note that enabling PGO will make GCC automatically enable several optimization passes that are usually disabled, even when using `-O2`. This can lead to new warnings/errors in the code, which might require either the code to be adjusted or the use of extra compilation flags to suppress these warnings.

After the software is built, you can install this new version and run the benchmark tests again to compare results.

### Our results

At first sight it might appear that our results don't have much to show for themselves, but they are actually very interesting and allow us to discuss some more things about PGO.

First, it will not work for every scenario, and there might be cases when it is not justified to use it. For example, if you are dealing with a very generic program that doesn't have a single representative workload. PGO is great when applied to specialized cases, but may negatively impact the performance of programs that can be used in a variety of ways.

It is crucial to be rigorous when profiling the program you want to optimize. Here are a few tips that might help you to obtain more reliable results:

- Perform multiple runs of the same workload. This helps to eliminate possible outliers and decrease the contribution of external factors (like memory and cache times). In our example, we performed 5 consecutive runs without PGO and 5 consecutive runs with PGO. Performing multiple runs is useful for benchmarking the software as well as for profiling it.
- Try to profile your software in a regular, production-like setup. For example, don't try to artificially minimize your system's background load because that will produce results that will likely not reflect workloads found in the real world. On the other hand, be careful about non-representative background load that might interfere with the measurements.
- Consider whether you are dealing with code that has already been highly optimized. In fact, this is exactly the case with our example: OpenSSL is an ubiquitous cryptographic library that has received *a lot* of attention from several developers, and it is expected that cryptographic libraries are extremely optimized. This is the major factor that explains why we have seen minimal performance gains in our experiment: the library is already pretty performant as it is.
- We also have written a blog post detailing a PGO case study which rendered performance improvements in the order of 5% to 10%. It is a great example of how powerful

this optimization technique can be when dealing with specialized workloads.

### Alternative scenario: profiling the program using other approaches

We do not always have well behaved programs that can be profiled like `openssl speed`. Sometimes, the program might take a long time to finish running the workload, which ends up generating huge `perfdata` files that can prove very challenging for `autofdo` to process. For example, we profiled QEMU internally and, when executing a workload that took around 3 hours to complete, the `perfdata` generated had a size of several gigabytes.

In order to reduce the size of the `perfdata` that is collected, you might want to play with `perf record`'s `-F` option, which specifies the frequency at which the profiling will be done. Using `-F 99` (i.e., profile at 99 hertz) is indicated by some `perf` experts because it avoids accidentally sampling the program in lockstep with some other periodic activity.

Another useful trick is to profile in batches instead of running `perf record` for the entire duration of your program's workload. To do that, you should use the `-p` option to specify a PID to `perf record` while also specifying `sleep` as the program to be executed during the profiling. For example, this would be the command like we would use if we were to profile a program whose PID is `1234` for 2 minutes:

```
$ sudo perf record \
        -e br_inst_retired.near_taken \
        --branch-any \
        --all-cpus \
        -p 1234 \
        --output myprogram.perfdata
        -- sleep 2m
```

The idea is that you would run the command above several times while the program being profiled is still running (ideally with an interval between the invocations, and taking care not to overwrite the `perfdata`). After obtaining all the `perfdata` files, you would then convert them to `gcov` and use `profile_merger` to merge all `gcov` files into one.

### Conclusion

PGO is an interesting and complex topic that certainly catches the attention of experienced software developers looking to extract every bit of performance gain. It can certainly help optimize programs, especially those that perform specialized work. It is not a one-size-fits-all solution, though, and as with every complex technology, its use needs to be studied and justified, and its impact on the final application should be monitored and analyzed.

## hwloc

> System tuning tools are either about better understanding the system's performance, or applying such knowledge to improve it. See our common *system tuning thoughts* for the general reasons for that.

The default installed package `hwloc-nox` provides various tools to discover the topology of the chip internal structures as well as any associated devices like PCI cards, NVME devices and memory.

By default the tools provide command line interfaces, but it can also render those complex relations in a *GUI* environment. Depending on which one you want to use you can install them

with `apt install hwloc-nox` or `apt install hwloc` respectively.

It can be important with a sensitive workload to align the CPU cores it runs on, to be close to the nodes where the related devices are attached to the system. That will help to avoid crossing topology boundaries that add additional latencies. But for that, one needs tools to inspect the CPU and device topology on the system.

### hwloc-info - hierarchy and details

By default, `hwloc-info` provides a summary of the topology levels that are present on this system.

Command:

```
hwloc-info
```

Output (example on a laptop):

```
depth 0:           1 Machine (type #0)
 depth 1:          1 Package (type #1)
  depth 2:         1 L3Cache (type #6)
   depth 3:        4 L2Cache (type #5)
    depth 4:       4 L1dCache (type #4)
     depth 5:      4 L1iCache (type #9)
      depth 6:     4 Core (type #2)
       depth 7:    4 PU (type #3)
Special depth -3:  1 NUMANode (type #13)
Special depth -4:  3 Bridge (type #14)
Special depth -5:  4 PCIDev (type #15)
Special depth -6:  3 OSDev (type #16)
```

But the tool also allows us to get detailed information on any of these elements. Here is an example to get details on the first core.

Command:

```
hwloc-info -p core:0
```

Output:

```
Core L#0
 type = Core
 full type = Core
 logical index = 0
 os index = 0
 gp index = 2
 depth = 6
 sibling rank = 0
 children = 1
 memory children = 0
 i/o children = 0
 misc children = 0
 cpuset = 0x00000001
```

(continues on next page)

(continued from previous page)

```
complete cpuset = 0x00000001
allowed cpuset = 0x00000001
nodeset = 0x00000001
complete nodeset = 0x00000001
allowed nodeset = 0x00000001
symmetric subtree = 1
cpukind = 0
cpukind efficiency = 0
cpukind info FrequencyMaxMHz = 4000
cpukind info FrequencyBaseMHz = 1800
```

See the `hwloc-info(1)` manual page for a list of all the potential objects that can be queried.

### hwloc-ls - system topology in CLI

If executed without graphical capabilities `hwloc-ls` will provide a text representation of the CPU, its caches and how they relate to each other between cores, sockets and hardware in the system.

Command:

```
hwloc-ls
```

Output (example on a laptop):

```
Machine (31GB total)
  Package L#0
    NUMANode L#0 (P#0 31GB)
    L3 L#0 (8192KB)
      L2 L#0 (256KB) + L1d L#0 (32KB) + L1i L#0 (32KB) + Core L#0 + PU L#0 (P#0)
      L2 L#1 (256KB) + L1d L#1 (32KB) + L1i L#1 (32KB) + Core L#1 + PU L#1 (P#1)
      L2 L#2 (256KB) + L1d L#2 (32KB) + L1i L#2 (32KB) + Core L#2 + PU L#2 (P#2)
      L2 L#3 (256KB) + L1d L#3 (32KB) + L1i L#3 (32KB) + Core L#3 + PU L#3 (P#3)
  HostBridge
    PCI 00:02.0 (VGA)
    PCIBridge
      PCI 04:00.0 (Network)
        Net "wlp4s0"
    PCIBridge
      PCI 40:00.0 (NVMExp)
        Block(Disk) "nvme0n1"
    PCI 00:1f.6 (Ethernet)
      Net "enp0s31f6"
```

Output (example on a server - shortened for readbility):

```
Package L#0
    NUMANode L#0 (P#0 378GB)
    L3 L#0 (32MB)
      L2 L#0 (1024KB) + L1d L#0 (48KB) + L1i L#0 (32KB) + Core L#0
        PU L#0 (P#0)
```

(continues on next page)

(continued from previous page)

```
        PU L#1 (P#128)
...
      L2 L#63 (1024KB) + L1d L#63 (48KB) + L1i L#63 (32KB) + Core L#63
        PU L#126 (P#63)
        PU L#127 (P#191)
    HostBridge
      PCIBridge
        PCI 12:00.0 (NVMExp)
          Block(Disk) "nvme1n1"
    HostBridge
      PCIBridge
        PCI 51:00.0 (NVMExp)
          Block(Disk) "nvme0n1"
      PCIBridge
        PCI 52:00.0 (Ethernet)
          Net "enp82s0"
      PCIBridge
        PCIBridge
          PCI 54:00.0 (VGA)
  Package L#1
    NUMANode L#1 (P#1 378GB)
    L3 L#8 (32MB)
      L2 L#64 (1024KB) + L1d L#64 (48KB) + L1i L#64 (32KB) + Core L#64
        PU L#128 (P#64)
        PU L#129 (P#192)
...
      L2 L#127 (1024KB) + L1d L#127 (48KB) + L1i L#127 (32KB) + Core L#127
        PU L#254 (P#127)
        PU L#255 (P#255)
    HostBridge
      PCIBridge
        2 x { PCI f2:00.0-1 (SATA) }
  Misc(MemoryModule)
```

### hwloc-ls - system topology in GUI

Instead of `hwloc-nox` there also is `hwloc` with graphical capabilities which would render the same in a graphical representation.

Command:

```
hwloc-ls
```

Output (CLI shows navigation info, the real content is in the GUI):

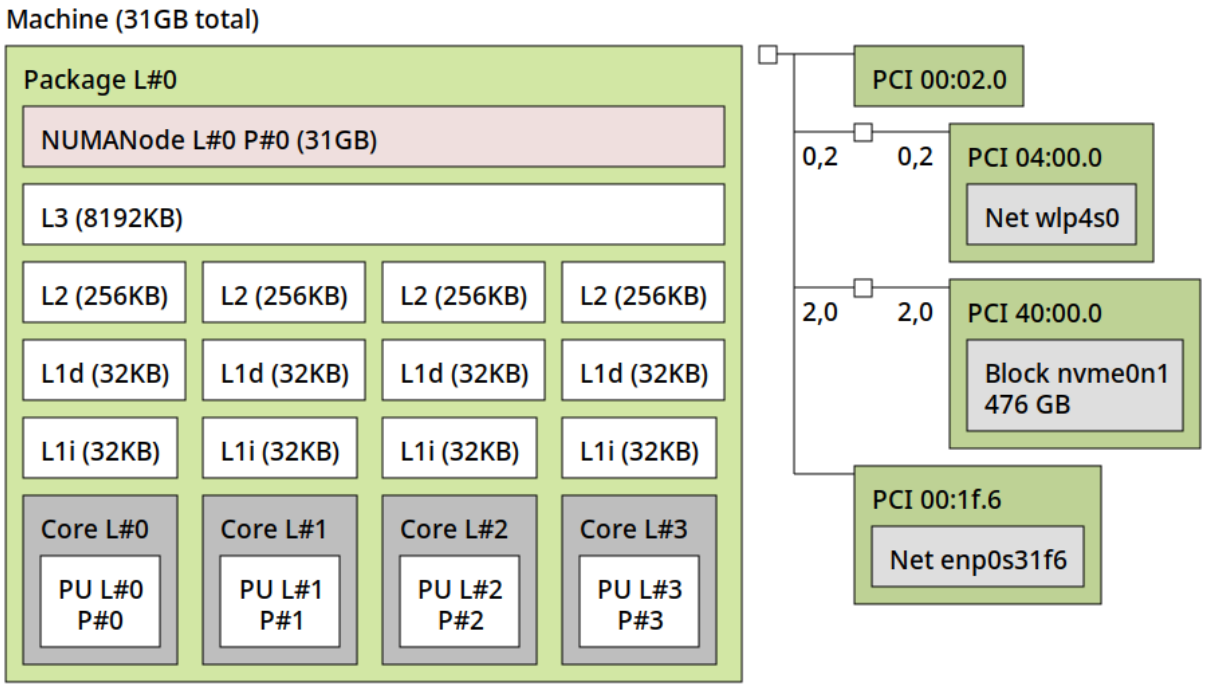
```
Keyboard shortcuts:
 Zooming, scrolling and closing:
  Zoom-in or out ..................... + -
  Reset scale to default ............. 1
  Try to fit scale to window ......... F
```

(continues on next page)

(continued from previous page)

```
 Resize window to the drawing ........ r
 Toggle auto-resizing of the window .. R
 Scroll vertically ................... Up Down PageUp PageDown
 Scroll horizontally ................. Left Right Ctrl+PageUp/Down
 Scroll to the top-left corner ....... Home
 Scroll to the bottom-right corner ... End
 Refresh the topology ................ F5
 Show this help ...................... h H ?
 Exit ................................ q Q Esc
Configuration tweaks:
 Toggle factorizing or collapsing .... f
 Switch display mode for indexes ..... i
 Toggle displaying of object text .... t
 Toggle displaying of obj attributes . a
 Toggle displaying of CPU kinds ...... k
 Toggle color for disallowed objects . d
 Toggle color for binding objects .... b
 Toggle displaying of legend lines ... l
 Export to file with current config .. E
```

Output (GUI):




## Further reading

- `lstopo(1)` manual page
- `hwloc(7)` manual page
- `hwloc-diff(1)` manual page

### CPU Governors and the cpupower tool

System tuning tools are either about better understanding the system's performance, or applying such knowledge to improve it. See our common *system tuning thoughts* for the general reasons for that.

### CPU governors

The kernel provides several CPU governors which can be configured, per core, to optimise for different needs.

| Governor | Design philosophy |
|---|---|
| ondemand | This sets the CPU frequency depending on the current system load. This behavior is usually a good balance between the more extreme options. |
| conservative | Similar to ondemand, but adapting CPU speed more gracefully rather than jumping to max speed the moment there is any load on the CPU. This behaviour is more suitable in a battery-powered environment. |
| performance | This sets the CPU statically to the highest frequency. This behaviour is best to optimise for speed and latency, but might waste power if being under-used. |
| powersave | Sets the CPU statically to the lowest frequency, essentially locking it to P2. This behavior is suitable to save power without compromises. |
| usersp | Allows a user-space program to control the CPU frequency. |

See the Linux CPUFreq Governors Documentation for a more extensive discussion and explanation of the available Linux CPU governors.

While these governors can be checked and changed directly in sysfs at `/sys/devices/system/cpu/cpu*/cpufreq/scaling_governor`, the command `cpupower` which comes with the package `linux-tools-common` makes this easier by providing a commandline interface and providing access to several related values.

### Monitor CPU frequency

Before changing anything, look at the current frequencies via `cpupower monitor`. Many systems have various potential monitors, and by default one sees all of them which can be quite confusing. Therefore start with looking at the available power monitors.

Command (list all available cpupower monitors available on the system):

```
sudo cpupower monitor -l
```

Output (An example from a common consumer laptop):

```
Monitor "Nehalem" (4 states) - Might overflow after 922000000 s
  C3        [C] -> Processor Core C3
```

(continues on next page)

(continued from previous page)

```
  C6        [C] -> Processor Core C6
  PC3        [P] -> Processor Package C3
  PC6        [P] -> Processor Package C6
Monitor "Mperf" (3 states) - Might overflow after 922000000 s
  C0        [T] -> Processor Core not idle
  Cx        [T] -> Processor Core in an idle state
  Freq       [T] -> Average Frequency (including boost) in MHz
Monitor "RAPL" (4 states) - Might overflow after 8640000 s
  pack       [M] ->
  dram       [M] ->
  core       [M] ->
  unco       [M] ->
Monitor "Idle_Stats" (9 states) - Might overflow after 4294967295 s
  POLL       [T] -> CPUIDLE CORE POLL IDLE
  C1        [T] -> MWAIT 0x00
  C1E        [T] -> MWAIT 0x01
  C3        [T] -> MWAIT 0x10
  C6        [T] -> MWAIT 0x20
  C7s        [T] -> MWAIT 0x33
  C8        [T] -> MWAIT 0x40
  C9        [T] -> MWAIT 0x50
  C10        [T] -> MWAIT 0x60
```

Here we can see that the machine has four available monitors shown in ".

- Nehalem - Hardware specific C states.
- Mperf - Average of frequencies and time in active (C0) or sleep (Cx) states.
- RAPL - Running Average Power Limit covering different system elements.
- Idle_Stats - Statistics of the `cpuidle` kernel subsystem (software based).

Those counters can represent different system units:

- [T] -> Thread
- [C] -> Core
- [P] -> Processor Package (Socket)
- [M] -> Machine/Platform wide counter

So if we want to know what frequency the CPU threads were in (Mperf) and what was consumed at the different system levels of package, dram, core and uncore (RAPL) averages over a minute (`-i <seconds>`) we would run:

Command:

```
sudo cpupower monitor -i 60 -m Mperf,RAPL
```

Output:

```
    | Mperf              || RAPL
 CPU| C0   | Cx   | Freq || pack    | dram    | core    | unco
```

(continues on next page)

(continued from previous page)

```
   0| 61,83| 38,17|  1850||616950936|145911797|375373063|71556823
   1| 62,03| 37,97|  1848||616950936|145911797|375373063|71556823
   2| 65,51| 34,49|  1852||616950936|145911797|375373063|71556823
   3| 62,04| 37,96|  1852||616950936|145911797|375373063|71556823
```

### Get details about the boundaries for the CPU frequency

There are more details influencing the CPU frequency, such as the driver used to control the hardware, the min and max frequencies, and potential boost states. These can be collected with `cpupower frequency-info`

Command:

```
cpupower frequency-info
```

Output:

```
analyzing CPU 3:
  driver: intel_pstate
  CPUs which run at the same hardware frequency: 3
  CPUs which need to have their frequency coordinated by software: 3
  maximum transition latency:  Cannot determine or is not supported.
  hardware limits: 400 MHz - 4.00 GHz
  available cpufreq governors: performance powersave
  current policy: frequency should be within 400 MHz and 4.00 GHz.
                  The governor "powersave" may decide which speed to use
                  within this range.
  current CPU frequency: Unable to call hardware
  current CPU frequency: 1.80 GHz (asserted by call to kernel)
  boost state support:
    Supported: yes
    Active: yes
```

By default this checks the CPU it is executed on. The argument `-c` can be set to either a number representing a core or `all` to get the info for all available CPUs.

### Get details about the idle states

Idle states represent situations when a CPU enters a state of suspension to save power. The tool `cpupower idle-info` reports about the available idle states, their description and attributes. These can be useful when debugging CPU performance if one is curious about the details of a given state after running `cpupower monitor` above.

Command:

```
cpupower idle-info
```

Output:

```
CPUidle driver: intel_idle
CPUidle governor: menu
```

(continues on next page)

(continued from previous page)

```
analyzing CPU 0:

Number of idle states: 9
Available idle states: POLL C1 C1E C3 C6 C7s C8 C9 C10
POLL:
Flags/Description: CPUIDLE CORE POLL IDLE
Latency: 0
Usage: 26053898
Duration: 695768311
C1:
Flags/Description: MWAIT 0x00
Latency: 2
Usage: 263751626
Duration: 21296361635
C1E:
Flags/Description: MWAIT 0x01
Latency: 10
Usage: 1071864698
Duration: 122465703132
C3:
Flags/Description: MWAIT 0x10
Latency: 70
Usage: 941753727
Duration: 117177626397
C6:
Flags/Description: MWAIT 0x20
Latency: 85
Usage: 2580936435
Duration: 1258804567087
C7s:
Flags/Description: MWAIT 0x33
Latency: 124
Usage: 2946723
Duration: 1783856599
C8:
Flags/Description: MWAIT 0x40
Latency: 200
Usage: 1580297534
Duration: 1234136981613
C9:
Flags/Description: MWAIT 0x50
Latency: 480
Usage: 2015405
Duration: 3198208930
C10:
Flags/Description: MWAIT 0x60
Latency: 890
Usage: 511786893
Duration: 1546264384800
```

After reading a bit (much more in the *Further reading* section) into C-states, P-states and Idle states we can also re-run `cpupower monitor` without filtering as now the further columns can be related to the above output.

Command:

```
sudo cpupower monitor
```

Output:

```
    | Nehalem                   || Mperf              || RAPL
     || Idle_Stats
 CPU| C3   | C6   | PC3  | PC6  || C0   | Cx   | Freq || pack | dram | core | unco
     || POLL | C1   | C1E  | C3   | C6   | C7s  | C8   | C9   | C10
   0|  2,99| 11,92|  0,00|  0,00|| 70,98| 29,02|
1991||13733058|2706597|7438396|3080986||  0,05|  1,84|  5,01|  3,87| 14,05|  0,06|
 3,81|  0,00|  0,04
   1|  3,58| 14,84|  0,00|  0,00|| 67,65| 32,35|
1991||13733058|2706597|7438396|3080986||  0,07|  1,87|  5,42|  4,46| 17,21|  0,36|
 2,73|  0,00|  0,00
   2|  3,99|  7,15|  0,00|  0,00|| 73,25| 26,75|
1990||13733058|2706597|7438396|3080986||  0,09|  1,95|  8,76|  5,20|  9,44|  0,01|
 1,12|  0,04|  0,00
   3|  3,86| 13,68|  0,00|  0,00|| 68,40| 31,60|
1990||13733058|2706597|7438396|3080986||  0,03|  2,52|  6,35|  4,92| 15,97|  0,00|
 1,52|  0,00|  0,00
```

### What should I do with all of this?

All this information is usually only *data* without any insight until you either:

- compare them with historical data (it is generally recommended to gather performance and power metrics regularly to be able to compare them to the healthy state in case of any debugging scenario), or
- compare them with your expectations and act on any mismatch

### Does it match what you expect?

One might have expectations about the behaviour of a system. Examples are:

- I'm not doing much – it should be idling most of the time
- I have a very busy workload, I expect it to run at highest frequency
- I do not expect my workload to allow the system to go into low power states

You can hold any of these assumptions against the output of `cpupower monitor` and verify that they are true. If they are not, use `cpupower frequency-info` to check if the current constraints match what you think. And use `cpupower frequency-set` (below) to set a different governor if needed.

### Control the CPU governors and CPU frequency

An administrator can execute the `cpupower` command to set the CPU governor.

Command (set the CPU governor to Performance mode on all CPUs):

```
cpupower frequency-set -g performance
```

Since all commands of `cpupower` can be for a sub-set of CPUs, one can use `-c` here as well if that matches what is needed for more complex scenarios.

Command (Set conservative on the first 8 cores in a system):

```
cpupower -c 0-7 frequency-set -g conservative
```

### Powertop

`powertop` supports the user in identifying reasons for unexpected high power consumption by listing reasons to wake up from low power states. The look and feel aligns with the well known `top`. `powertop` is not installed by default, before trying run `sudo apt install powertop`. This command needs elevated permissions, so run it with `sudo`.

```
sudo powertop
```

It has six tabs for the various areas of interest:

- Overview - frequency and reason for activity
- Idle stats - time spent in the different idle states
- Frequency stats - current frequency per core
- Device stats - activity of devices
- Tunables - a list of system tunables related to power (ratings are to save power, you might have some `Bad` for that being considered better for performance)
- WakeUp - device wake-up status

### Further reading

- Intel Frequency scaling
- Idle states
- C-states

### TuneD

> Any tool related to system tuning is either about better understanding the system or after doing so applying this knowledge. See our common *system tuning thoughts* for the general reasons for that.

The same is true for the TuneD profiles - they are only suggestions and starting points for a few named workload categories that allow you to react dynamically.

TuneD*1 is a service used to tune your system and optimise the performance under certain workloads. At the core of TuneD are **profiles**, which tune your system for different use cases. TuneD is distributed with a number of predefined profiles for use cases such as:

- High throughput
- Low latency
- Saving power

It is possible to modify the rules defined for each profile and customise how to tune a particular device. When you switch to another profile or deactivate TuneD, all changes made to the system settings by the previous profile revert back to their original state.

You can also configure TuneD to dynamically react to changes in device usage and adjust settings to improve the performance of active devices and reduce the power consumption of inactive devices.

### Static vs. dynamic tuning

TuneD can perform two types of tuning: **static** and **dynamic**.

- Static tuning mainly consists of applying predefined `sysctl` and `sysfs` settings and the one-shot activation of several configuration tools such as `ethtool`.
- In dynamic tuning, it watches how various system components are used throughout the uptime of your system. TuneD then adjusts the system settings dynamically based on that monitoring information. For example, the hard drive is used heavily during startup and login, but is barely used later when the user is mainly working with applications (e.g. web browsers or email clients). Similarly, the CPU and network devices are used differently at different times. TuneD monitors the activity of these components and reacts to the changes in their use.

By default, dynamic tuning is enabled. To disable it, edit the `/etc/tuned/tuned-main.conf` file and change the `dynamic_tuning` option to `0`. TuneD then periodically analyses system statistics and uses them to update your system tuning settings. To configure the time interval in seconds between these updates, use the `update_interval` option. After any change in this configuration file, the systemd service needs to be restarted.

### Profiles

TuneD works with profiles, which are configuration files grouping tuning plugins and their options. Upon installation of the `tuned` package, a profile will be applied by default depending on the detected environment. These are the default profiles for each type of environment:

| Environment | Default profile |
|---|---|
| Compute nodes | `throughput-performance` |
| Virtual Machines | `virtual-guest` |
| Containers | `default` |
| Other cases | `balanced` |

### Anatomy of a profile

Predefined tuned profiles provided by the package are located in the directory `/usr/lib/tuned/<profile-name>`, those added by the administrator should be placed in `/etc/tuned/<profile-name>`.

> In TuneD version 2.24 and thereby Ubuntu 25.04 Plucky (and later) the location of these files changed. An upgrade will migrate any custom profiles, however since most users are not yet on the new release, the rest of this page uses the old paths in the examples. Predefined profiles: `/usr/lib/tuned/<profile-name>` -> `/usr/lib/tuned/profiles/<profile-name>` User defined profiles: `/etc/tuned/<profile-name>` -> `/etc/tuned/profiles/<profile-name>`

In each of these `<profile-name>` directories a file `tuned.conf` defines that profile. That file has an INI structure which looks like this:

```
[main]
include=PROFILE # if inheriting from another profile
summary=A short summary
description=A short description

[plugin instance]
type=TYPE
replace=REPLACE
enabled=ENABLED
devices=DEVICES

[another plugin instance]
type=TYPE
replace=REPLACE
enabled=ENABLED
devices=DEVICES
... other plugin-specific options ...

...
```

Here is a brief explanation of these configuration parameters:

- `type`: This is the plugin type. A list of all types can be obtained with the command `tuned-adm list plugins`.
- `replace`: This can take the values `true` or `false`, and is used to control what happens when two plugins of the same type are trying to configure the same devices. If `true`, then the plugin defined last replaces all options.
- `enabled`: This also accepts the values `true` or `false`, and is used to control if a plugin should remain enabled or disabled when inheriting from another profile.
- `devices`: A comma separated list of device names (without the `/dev/` prefix) which represents the devices this plugin should act on. If not specified, all compatible devices found, now or in the future, will be used. This parameter also accepts simple globbing and negation rules, so that you can specify `nvme*` for all `/dev/nvme*` devices, or `!sda` to not include `/dev/sda`.
- plugin-specific options: These can be seen in the output of the `tuned-adm list plugins`

`-v` command, for each listed plugin.

See the `tuned.conf(5)` manpage for details on the syntax of this configuration file.

The plugin instance concept can be useful if you want to apply different tuning parameters to diferent devices. For example, you could have one plugin instance to take care of NVMe storage, and another one for spinning disks:

```
[fast_storage]
type=disk
devices=nvme0n*
... options for these devices

[slow_storage]
type=disk
devices=sda, sdb
... options for these devices
```

### Available profiles and plugins

The list of available profiles can be found using the following command:

```
tuned-adm list profiles
```

Which will result in a long list (output truncated for brevity):

```
Available profiles:
- accelerator-performance     - Throughput performance based tuning with disabled
higher latency STOP states
- atomic-guest                - Optimize virtual guests based on the Atomic
variant
- atomic-host                 - Optimize bare metal systems running the Atomic
variant
(...)
```

Here are some useful commands regarding profiles:

- `tuned-adm active`: Shows which profile is currently enabled.
- `tuned-adm recommend`: Shows the recommended profile for this system.
- `tuned-adm profile <name>`: Switch to the named profile, applying its settings.

The list of plugin types can be obtained with the `tuned-adm list plugins`, and to see plugin-specific options, run `tuned-adm list plugins -v`. Unfortunately at the moment the documentation of those options is only present in the source code of each plugin.

For example, the `cpu` plugin options are:

```
cpu
        load_threshold
        latency_low
        latency_high
        force_latency
        governor
```

(continues on next page)

(continued from previous page)

```
        sampling_down_factor
        energy_perf_bias
        min_perf_pct
        max_perf_pct
        no_turbo
        pm_qos_resume_latency_us
        energy_performance_preference
```

And their description can be found in `/usr/lib/python3/dist-packages/tuned/plugins/plugin_cpu.py`.

### Customising a profile

For some specific workloads, the predefined profiles might not be enough and you may want to customise your own profile. You may customise an existing profile, just overriding a few settings, or create an entirely new one.

Custom profiles live in `/etc/tuned/<profile-name>/tuned.conf` (Remember this location changed in 25.04 Plucky and later). There are 3 ways they can be created:

- Copy an existing profile from `/usr/lib/tuned/<profile-name>` to `/etc/tuned/<profile-name>`, and make changes to it in that location. A profile defined in `/etc/tuned` takes precedence over one from `/usr/lib/tuned` with the same name.
- Create an entirely new profile in `/etc/tuned/<new-profile-name>` from scratch.
- Create a new profile in `/etc/tuned/<new-profile-name>`, with a name that doesn't match an existing profile, and inherit from another profile. In this way you only have to specify the changes you want, and inherit the rest from the existing profile in `/usr/lib/tuned/<profile-name>`.

After that, the new profile will be visible by TuneD via the `tuned-adm list` command.

Here is a simple example of a customised profile named `mypostgresql` that is inheriting from the existing `/usr/lib/tuned/postgresql` profile. The child profile is defined in `/etc/tuned/mypostgresql/tuned.conf`:

```
[main]
include=postgresql

[cpu]
load_threshold=0.5
latency_low=10
latency_high=10000
```

The inheritance is specified via the `include` option in the `[main]` section.

After the `[main]` section come the plugins that we want to override, and their new settings. Settings not specified here will take the value defined in the parent profile, `postgresql` in this case. If you want to completely ignore whatever the `cpu` plugin defined in the parent profile, use the `replace=true` setting.

### Merging profiles

There are some profiles that are neither a parent profile, nor a child profile. They only specify a few plugins and settings, and no inheritance relationship. By themselves, they are not useful, but they can be merged with an existing profile on-the-fly.

Here is an example which applies the base profile `cpu-partitioning` and then overlays `intel-sst` on top:

```
sudo tuned-adm profile cpu-partitioning intel-sst
```

In a sense, it's like a dynamic inheritance: instead of having the `intel-sst` profile include `cpu-partitioning` in a hardcoded `include` statement, it can be used in this way and merge its settings to any other base profile on-the-fly, at runtime.

Another example of merging profiles is the combining of the `powersave` profile with another one:

```
sudo tuned-adm profile virtual-guest powersave
```

This would optimise the system for a virtual guest, and then apply power saving parameters on top.

> **Warning**
>
> Just because `tuned-adm` accepted to merge two profiles doesn't mean it makes sense. There is no checking done on the resulting merged parameters, and the second profile could completely revert what the first profile adjusted.

### An example profile: hpc-compute

Let's take look at the predefined `hpc-compute` profile in more detail as an example. You can find the configuration of this profile in `/usr/lib/tuned/hpc-compute/tuned.conf`:

```
#
# tuned configuration
#

[main]
summary=Optimize for HPC compute workloads
description=Configures virtual memory, CPU governors, and network settings for HPC
compute workloads.
include=latency-performance

[vm]
# Most HPC application can take advantage of hugepages. Force them to on.
transparent_hugepages=always

[disk]
# Increase the readahead value to support large, contiguous, files.
readahead=>4096
```

(continues on next page)

(continued from previous page)

```
[sysctl]
# Keep a reasonable amount of memory free to support large mem requests
vm.min_free_kbytes=135168

# Most HPC applications are NUMA aware. Enabling zone reclaim ensures
# memory is reclaimed and reallocated from local pages. Disabling
# automatic NUMA balancing prevents unwanted memory unmapping.
vm.zone_reclaim_mode=1
kernel.numa_balancing=0

# Busy polling helps reduce latency in the network receive path
# by allowing socket layer code to poll the receive queue of a
# network device, and disabling network interrupts.
# busy_read value greater than 0 enables busy polling. Recommended
# net.core.busy_read value is 50.
# busy_poll value greater than 0 enables polling globally.
# Recommended net.core.busy_poll value is 50
net.core.busy_read=50
net.core.busy_poll=50

# TCP fast open reduces network latency by enabling data exchange
# during the sender's initial TCP SYN. The value 3 enables fast open
# on client and server connections.
net.ipv4.tcp_fastopen=3
```

The `[main]` section contains some metadata about this profile, a summary and description, and whether it includes other profiles. In this case, another profile *is* included; the `latency-performance` profile.

The sections that follow `[main]` represent the configuration of tuning plugins.

- The first one is the `vm` plugin, which is used to always make use of huge pages (useful in this *HPC* scenario).
- The second plugin used is `disk`, which is used to set the `readahead` value to at least `4096`.
- Finally, the `sysctl` plugin is configured to set several variables in `sysfs` (the comments in the example explain the rationale behind each change).

*1: This is a universe package Ubuntu ships this software as part of its universe repository, which is maintained by volunteers from the Ubuntu community. Canonical also offers Ubuntu Pro – a free-for-personal-use subscription that provides a 10 year security maintenance commitment.

### Further reading

- TuneD website
- `tuned-adm(8)` manpage
- `TuneD profiles` manpage
- `TuneD daemon` manpage
- `TuneD configuration` manpage

### Energy Performance Preference

Energy Performance Preference (EPP) is a CPU power-management feature available on many modern processors (particularly Intel and AMD). It allows the operating system or user to influence the CPU's balance between performance and power consumption. Instead of forcing a strict maximum performance or minimum power mode, EPP provides a hint to the processor's hardware power control logic about the desired balance.

The actual effect depends on the CPU model and firmware, but in general, lower EPP values tell a processor to favor performance while higher values tell it to favor lower power consumption.

### Caveats

- EPP is only a *preference* or hint to the hardware. It does not guarantee that the CPU will behave in a precise or deterministic way.
- EPP value interpretations will often vary between different CPUs.
- Hardware support is required. On some processors, EPP may have no effect even if it appears to be configurable in the OS.
- Some systems may require an update to BIOS settings to allow OS control of power regulation.

### EPP values on Ubuntu

On Ubuntu, the EPP for each core on a CPU is stored in `/sys/devices/system/cpu/cpu#/cpufreq/energy_performance_preference`. To check the current status on cpu0, for example, you can run:

```
cat /sys/devices/system/cpu/cpu0/cpufreq/energy_performance_preference
```

This file will normally contain one of the following named profiles:

- `performance` - Prioritize responsiveness and raw performance.
- `balance_performance` - Lean toward performance, with lower power consumption.
- `balance_power` - Lean toward reduced power, with some additional performance.
- `power` - Prioritize a reduction in power consumption.
- `default` - Use the default mix of power and performance as recommended by Ubuntu.

The full list of options is provided in `/sys/devices/system/cpu/cpu#/cpufreq/energy_performance_available_preferences`. To see them, run:

```
cat /sys/devices/system/cpu/cpu0/cpufreq/energy_performance_available_preferences
```

which will display something like:

```
default performance balance_performance balance_power power
```

For more specificity, a raw numeric value from **0-255** can also be provided in the `energy_performance_preference` file. In general, a value closer to `0` will prioritize performance and a value closer to `255` will prioritize efficiency.

### Setting EPP values

EPP values are managed through each core's `energy_performance_preference` file, so the EPP values can be updated manually by overriding the contents of that file. For example, to set cpu0 to the `performance` profile, run the following:

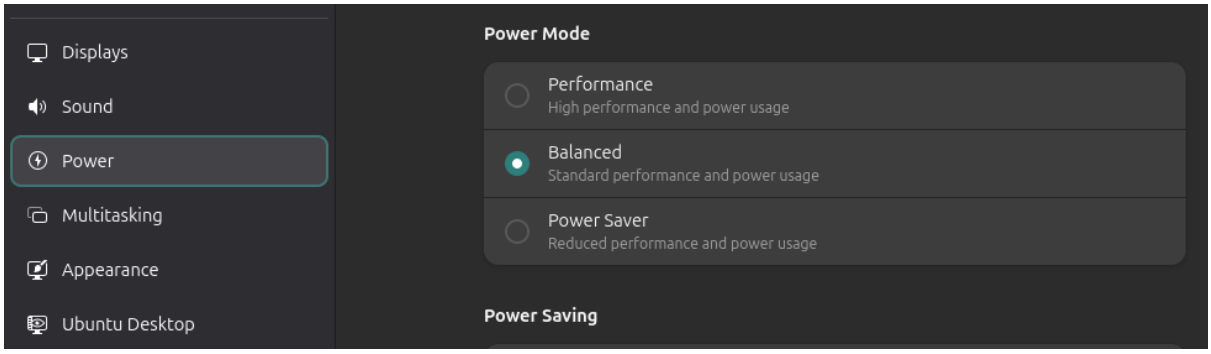
```
echo performance | sudo tee /sys/devices/system/cpu/cpu0/cpufreq/energy_
performance_preference
```

There are several additional tools that can manipulate EPP values on Ubuntu too.

*cpupower* will likely override the value to `performance` on all cores when swapping to the `performance` governer with:

```
sudo cpupower frequency-set -g performance
```

On Ubuntu Desktop, GNOME provides a power management menu in settings with options that may modify EPP values.




These settings will normally map to the following EPP profiles:

| Power Mode setting | EPP profile |
|---|---|
| Performance | performance |
| Balanced | balance_performance |
| Power Saver | power |

# 4.9. High Availability

*High Availability* is a method for clustering resources to ensure minimal downtime if a particular component fails.

In this section we provide an introduction to High Availability and explain some of the key concepts.

## 4.9.1. High Availability

High Availability is a way to ensure minimal downtime if and when system failures occur by using redundancy, failover and load balancing to keep services running.

### Introduction

- *Introduction to High Availability*

### Key concepts

- *Pacemaker resource agents*
- *Pacemaker fence agents*

### Introduction to High Availability

A definition of high availability clusters from Wikipedia:

### High Availability clusters

> **High-availability clusters** (also known as **HA clusters** , **failover clusters** or **Metroclusters Active/Active** ) are groups of computers that support server applications that can be reliably utilised with a minimum amount of down-time.They operate by using high availability software to harness redundant computers in groups or clusters that provide continued service when system components fail. Without clustering, if a server running a particular application crashes, the application will be unavailable until the crashed server is fixed. HA clustering remedies this situation by detecting hardware/software faults, and immediately restarting the application on another system without requiring administrative intervention, a process known as failover. As part of this process, clustering software may configure the node before starting the application on it. For example, appropriate file systems may need to be imported and mounted, network hardware may have to be configured, and some supporting applications may need to be running as well.
>
> HA clusters are often used for critical databases, file sharing on a network, business applications, and customer services such as electronic commerce websites.

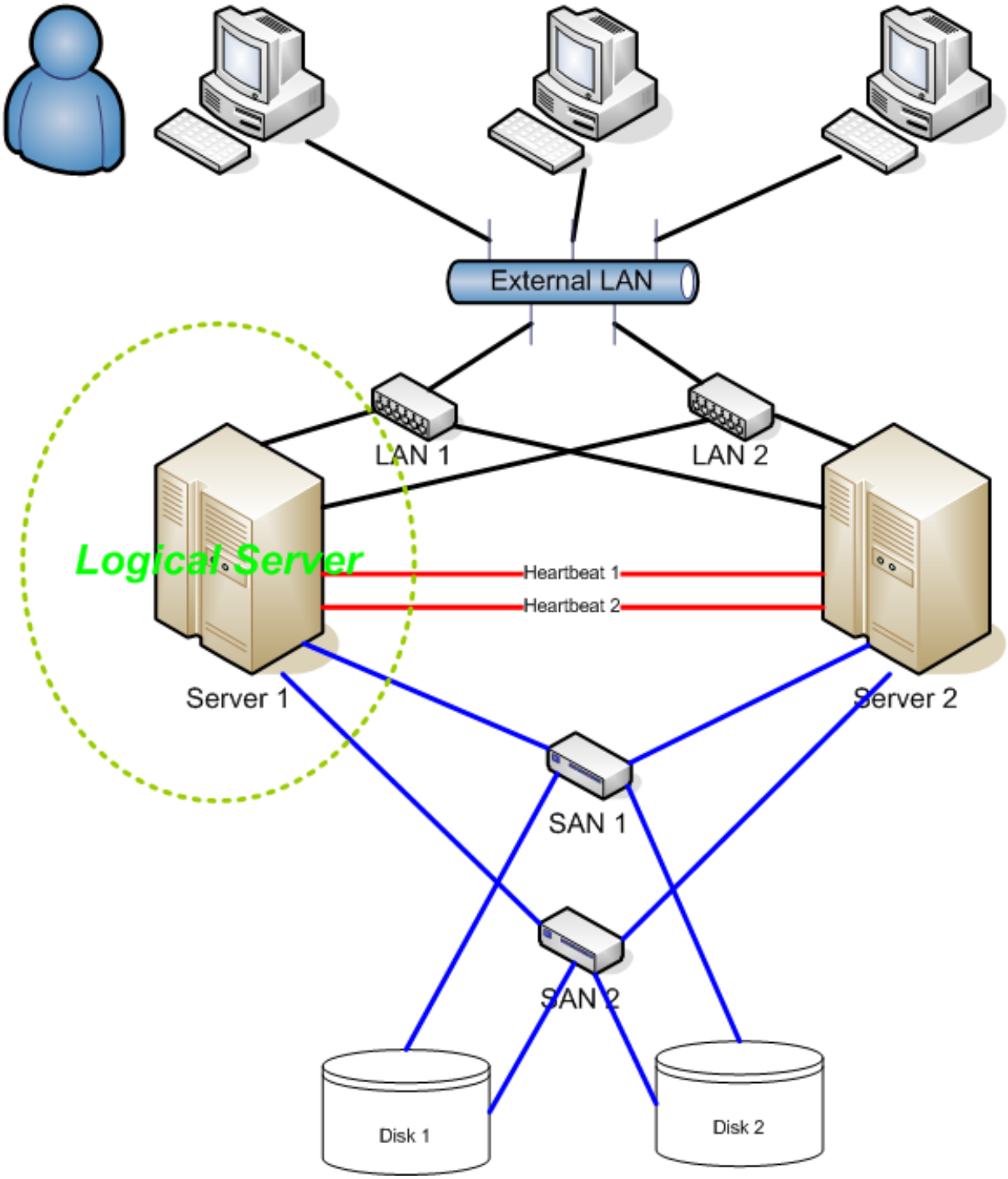
### High Availability cluster heartbeat

> HA cluster implementations attempt to build redundancy into a cluster to eliminate single points of failure, including multiple network connections and data storage which is redundantly connected via storage area networks.
>
> HA clusters usually use a heartbeat private network connection which is used to monitor the health and status of each node in the cluster. One subtle but serious condition all clustering software must be able to handle is split-brain, which occurs when all of the private links go down simultaneously, but the cluster nodes are still running. If that happens, each node in the cluster may mistakenly decide that every other node has gone down and attempt to start services that other nodes are still running. Having duplicate instances of services may cause data corruption on the shared storage.

### High Availability cluster quorum

HA clusters often also use quorum witness storage (local or cloud) to avoid this scenario. A witness device cannot be shared between two halves of a split cluster, so in the event that all cluster members cannot communicate with each other (e.g., failed heartbeat), if a member cannot access the witness, it cannot become active.

### Example of HA cluster quorum



### Fencing

Fencing protects your data from being corrupted, and prevents your application from becoming unavailable, due to unintended concurrent access by rogue nodes. If a node is unresponsive, it doesn't mean it has stopped accessing your data. The only way to be absolutely sure your data is safe is to use fencing, which ensures that the unresponsive node is truly offline before the data can be accessed by another node.

In cases where a clustered service cannot be stopped, a cluster can use fencing to force the whole node offline, making it safe to start the service elsewhere. The most popular example of fencing is cutting a host's power. Key benefits:

- An active counter-measure taken by a functioning host to isolate a misbehaving (usually dead) host from shared data.
- Fencing is the **most critical** part of a cluster using Storage Area Network (SAN) or other shared storage technology (*Ubuntu HA Clusters can only be supported if the fencing mechanism is configured*).
- Required by OCFS2, *GFS2*, cLVMd (before Ubuntu 20.04), lvmlockd (from 20.04 and beyond).

### Linux High Availability projects

There are many upstream high availability related projects that are included in Ubuntu Linux. This section will describe the most important ones. The following packages are present in the latest Ubuntu LTS release:

### Main Ubuntu HA packages

Packages in this list are supported just like any other package available in the **[main] repository**:

| Package | URL |
|---|---|
| `libqb` | Ubuntu \| Upstream |
| `kronosnet` | Ubuntu \| Upstream |
| `corosync` | Ubuntu \| Upstream |
| `pacemaker` | Ubuntu \| Upstream |
| `resource-agents` | Ubuntu \| Upstream |
| `fence-agents` | Ubuntu \| Upstream |
| `crmsh` | Ubuntu \| Upstream |
| `pcs*` | Ubuntu \| Upstream |
| `cluster-glue` | Ubuntu \| Upstream |
| `drbd-utils` | Ubuntu \| Upstream |
| `dlm` | Ubuntu \| Upstream |
| `gfs2-utils` | Ubuntu \| Upstream |
| `keepalived` | Ubuntu \| Upstream |

- `libqb` - Library which provides a set of high performance client-server reusable features. It offers high performance logging, tracing, IPC and poll. Its initial features were spun off the `Corosync` cluster communication suite to make them accessible for other projects.

- `Kronosnet` - `Kronosnet`, often referred to as `knet`, is a network abstraction layer designed for High Availability. `Corosync` uses `Kronosnet` to provide multiple networks for its interconnect (replacing the old Totem Redundant Ring Protocol) and adds support for some more features like interconnect network hot-plug.
- `Corosync` - or *Cluster Membership Layer*, provides reliable messaging, membership and quorum information about the cluster. Currently, Pacemaker supports `Corosync` as this layer.
- **Pacemaker** - or *Cluster Resource Manager*, provides the brain that processes and reacts to events that occur in the cluster. Events might be: nodes joining or leaving the cluster, resource events caused by failures, maintenance, or scheduled activities. To achieve the desired availability, Pacemaker may start and stop resources and fence nodes.
- **Resource Agents** - Scripts or operating system components that start, stop or monitor resources, given a set of resource parameters. These provide a uniform interface between pacemaker and the managed services.
- **Fence Agents** - Scripts that execute node fencing actions, given a target and fence device parameters.
- **crmsh** - Advanced command-line interface for High-Availability cluster management in *GNU*/Linux.
- **pcs** - Pacemaker command line interface and GUI. It permits users to easily view, modify and create pacemaker based clusters. `pcs` also provides `pcsd`, which operates as a GUI and remote server for `pcs`. Together `pcs` and `pcsd` form the recommended configuration tool for use with pacemaker. *NOTE: It was added to the [main] repository in Ubuntu Lunar Lobster (23.10).*
- **cluster-glue** - Reusable cluster components for Linux HA. This package contains node fencing plugins, an error reporting utility, and other reusable cluster components from the Linux HA project.
- **DRBD** - Distributed Replicated Block Device, **DRBD** is a distributed replicated storage system for the Linuxplatform. It is implemented as a kernel driver, several userspace management applications, and some shell scripts. DRBD is traditionally used in high availability (HA) clusters.
- **DLM** - A distributed lock manager (DLM) runs in every machine in a cluster, with an identical copy of a cluster-wide lock database. In this way DLM provides software applications which are distributed across a cluster on multiple machines with a means to synchronize their accesses to shared resources.
- **gfs2-utils** - Global File System 2 - *filesystem* tools. The Global File System allows a cluster of machines to concurrently access shared storage hardware like SANs or iSCSI and network block devices.
- **Keepalived** - Provides simple and robust facilities for load balancing and high availability to Linux systems and Linux-based infrastructures. The load balancing framework relies on the well-known and widely used Linux Virtual Server (IPVS) kernel module which provides Layer4 load balancing. It implements a set of checkers to dynamically and adaptively maintain and manage a load-balanced server pool according to their health, while high availability is achieved by the VRRP protocol.

### Ubuntu HA community packages

The HA packages in this list are supported just like any other package available in the **[universe] repository**.

| Package | URL |
|---|---|
| pcs* | Ubuntu \| Upstream |
| csync2 | Ubuntu \| Upstream |
| corosync-qdevice | Ubuntu \| Upstream |
| fence-virt | Ubuntu \| Upstream |
| sbd | Ubuntu \| Upstream |
| booth | Ubuntu \| Upstream |

- **Corosync-Qdevice** - Primarily used for even-node clusters and operates at the corosync (quorum) layer. Corosync-Qdevice is an independent arbiter for solving split-brain situations. (qdevice-net supports multiple algorithms).
- **SBD** - It is a fencing block device that can be particularly useful in environments where traditional fencing mechanisms are not possible. SBD integrates with Pacemaker, which serves as a watchdog device and shared storage, to arrange for nodes to reliably self-terminate when fencing is required.

> **Note**
>
> **pcs** was added to the [main] repository in Ubuntu Lunar Lobster (23.04).

### Ubuntu HA deprecated packages

Packages in this list are **only supported by the upstream community** . All bugs opened against these agents will be forwarded to upstream **if** it makes sense (the affected version is closer to upstream).

| Package | URL |
|---|---|
| ocfs2-tools | Ubuntu \| Upstream |

### Ubuntu HA related packages

Packages in this list aren't necessarily **HA** related packages, but they play a very important role in High Availability Clusters and are supported like any other package provided by the **[main]** repository.

| Package | URL |
|---|---|
| multipath-tools | Ubuntu \| Upstream |
| open-iscsi | Ubuntu \| Upstream |
| sg3-utils | Ubuntu \| Upstream |
| tgt OR targetcli-fb* | Ubuntu \| Upstream |
| lvm2 | Ubuntu \| Upstream |

- **LVM2** in a Shared-Storage Cluster Scenario

  **CLVM** - supported before **Ubuntu 20.04**

  A distributed lock manager (DLM) is used to broker concurrent *LVM* metadata accesses. Whenever a cluster node needs to modify the LVM metadata, it must secure permission from its local `clvmd` , which is in constant contact with other `clvmd` daemons in the cluster and can communicate a need to lock a particular set of objects. `lvmlockd(8)` - supported after **Ubuntu 20.04** As of 2017, a stable LVM component that is designed to replace `clvmd` by making the locking of LVM objects transparent to the rest of LVM, without relying on a distributed lock manager. The lvmlockd benefits over clvm are:

  - lvmlockd supports two cluster locking plugins: DLM and SANLOCK. SANLOCK plugin can supports up to ~2000 nodes that benefits LVM usage in big virtualization / storage cluster, while DLM plugin fits HA cluster.
  - lvmlockd has better design than clvmd. clvmd is command-line level based locking system, which means the whole LVM software will get hang if any LVM command gets dead-locking issue.
  - lvmlockd can work with lvmetad.

> **Note**
>
> `targetcli-fb (Linux LIO)` will likely replace `tgt` in future Ubuntu versions.

### Upstream documentation

The Server documentation is not intended to document every option for all the HA related software described in this page. More complete documentation can be found upstream at:

- ClusterLabs
  - Clusters From Scratch
  - Managing Pacemaker Clusters
  - Pacemaker Configuration Explained
  - Pacemaker Remote - Scaling HA Clusters
- Other
  - Ubuntu Bionic HA in Shared Disk Environments (Azure)

  A very special thanks and all the credit to ClusterLabs Project for their detailed documentation.

### Pacemaker resource agents

From the ClusterLabs definition:

> Resource agents are the abstraction that allows Pacemaker to manage services it knows nothing about. They contain the logic for what to do when the cluster wishes to start, stop or check the health of a service. This particular set of agents conform to the Open Cluster Framework (OCF) specification.

Currently, the `resource-agents` binary package has been split into two: `resource-agents-base` and `resource-agents-extra`. The `resource-agents-base` binary package contains a set of curated agents which the Ubuntu Server team continuously runs tests on to make sure everything is working as expected. All the other agents previously in the `resource-agents` binary package are now found in the `resource-agents-extra` package.

The `resource-agents-base` binary package contains the following agents in the latest Ubuntu release:

- `IPaddr2`
- `iscsi`
- `iSCSILogicalUnit`
- `iSCSITarget`
- `LVM-activate`
- `systemd`

All of these agents are in `main` and are fully supported.

All other agents are in `resource-agents-extra` and while most of them are supported by upstream, they are not curated by the Ubuntu Server team. The set of resource agents that are not maintained by upstream is listed in `/usr/share/doc/resource-agents-extra/DEPRECATED_AGENTS`, the use of those agents is discouraged.

For the resource agents provided by `resource-agents-base`, we will briefly describe how to use them.

> **Note**
>
> There are two well known tools used to manage fence agents, they are `crmsh` and `pcs`. Here we will present examples with both, since `crmsh` is the recommended and supported tool until Ubuntu 22.10 Kinetic Kudu, and `pcs` is the recommended and supported tool from Ubuntu 23.04 Lunar Lobster onwards. For more information on how to migrate from `crmsh` to `pcs` *refer to this migration guide*.

### IPaddr2

From its manpage:

> This Linux-specific resource manages IP alias IP addresses. It can add an IP alias, or remove one. In addition, it can implement Cluster Alias IP functionality if invoked as a clone resource.

One could configure a `IPaddr2` resource with the following command:

```
$ crm configure primitive $RESOURCE_NAME ocf:heartbeat:IPaddr2 \
            ip=$IP_ADDRESS \
            cidr_netmask=$NET_MASK \
            op monitor interval=30s
```

One can do the same using `pcs` via the following command:

```
$ pcs resource create $RESOURCE_NAME ocf:heartbeat:IPaddr2 \
            ip=$IP_ADDRESS \
            cidr_netmask=$NET_MASK \
            op monitor interval=30s
```

This is one way to set up `IPaddr2`, for more information refer `to its manpage`.

### iscsi

From its manpage:

> Manages a local iSCSI initiator and its connections to iSCSI targets.

Once the iSCSI target is ready to accept connections from the initiator(s), with all the appropriate permissions, the `iscsi` resource can be configured with the following command:

```
$ crm configure primitive $RESOURCE_NAME ocf:heartbeat:iscsi \
          target=$TARGET \
          portal=$PORTAL
```

One can do the same using `pcs` via the following command:

```
$ pcs resource create $RESOURCE_NAME ocf:heartbeat:iscsi \
          target=$TARGET \
          portal=$PORTAL
```

Where `$TARGET` is the iSCSI Qualified Name (IQN) of the iSCSI target and `$PORTAL` its address, which can be, for instance, formed by the IP address and port number used by the target daemon.

This is one way to set up `iscsi`, for more information refer `to its manpage`.

### iSCSILogicalUnit

From its manpage:

> Manages iSCSI Logical Unit. An iSCSI Logical unit is a subdivision of an SCSI Target, exported via a daemon that speaks the iSCSI protocol.

This agent is usually used alongside with `iSCSITarget` to manage the target itself and its Logical Units. The supported implementation of iSCSI targets is using `targetcli-fb`, due to that, make sure to use `lio-t` as the implementation type. Considering one has an iSCSI target in place, the `iSCSILogicalUnit` resource could be configured with the following command:

```
$ crm configure primitive $RESOURCE_NAME ocf:heartbeat:iSCSILogicalUnit \
          implementation=lio-t \
          target_iqn=$IQN_TARGET \
          path=$DEVICE \
          lun=$LUN
```

One can do the same using `pcs` via the following command:

```
$ pcs resource create $RESOURCE_NAME ocf:heartbeat:iSCSILogicalUnit \
          implementation=lio-t \
```

(continues on next page)

(continued from previous page)

```
          target_iqn=$IQN_TARGET \
          path=$DEVICE \
          lun=$LUN
```

Where implementation is set to `lio-t` as mentioned before, `$IQN_TARGET` is the iSCSI Qualified Name (IQN) that this Logical Unit belongs to, `$DEVICE` is the path to the exposed block device, and `$LUN` is the number representing the Logical Unit which will be exposed to initiators.

This is one way to set up `iSCSILogicalUnit`, for more information refer to its manpage.

### iSCSITarget

From its manpage:

> Manages iSCSI targets. An iSCSI target is a collection of SCSI Logical Units (LUs) exported via a daemon that speaks the iSCSI protocol.

This agent is usually used alongside with `iSCSILogicalUnit` to manage the target itself and its Logical Units. The supported implementation of iSCSI targets is using `targetcli-fb`, due to that, make sure to use `lio-t` as the implementation type. With `targetcli-fb` installed on the system, the `iSCSITarget` resource can be configured with the following command:

```
$ crm configure primitive $RESOURCE_NAME ocf:heartbeat:iSCSITarget \
          implementation=lio-t \
          iqn=$IQN_TARGET
```

One can do the same using `pcs` via the following command:

```
$ pcs resource create $RESOURCE_NAME ocf:heartbeat:iSCSITarget \
          implementation=lio-t \
          iqn=$IQN_TARGET
```

Where implementation is set to `lio-t` as mentioned before and `$IQN_TARGET` is the IQN of the target.

This is one way to set up `iSCSITarget`, for more information refer to its manpage.

### LVM-activate

From its manpage:

> This agent manages LVM activation/deactivation work for a given volume group.

If the LVM setup is ready to be activated and deactivated by this resource agent (make sure the `system_id_resource` is set to `uname` in `/etc/lvm/lvm.conf`), the `LVM-activate` resource can be configured with the following command:

```
$ crm configure primitive $RESOURCE_NAME ocf:heartbeat:LVM-activate \
              vgname=$VOLUME_GROUP \
              vg_access_mode=system_id
```

One can do the same using `pcs` via the following command:

```
$ pcs resource create $RESOURCE_NAME ocf:heartbeat:LVM-activate \
             vgname=$VOLUME_GROUP \
             vg_access_mode=system_id
```

This is one way to set up `LVM-activate`, for more information refer `to its manpage`.

### Systemd

There is also a way to manage systemd unit files via a resource agent. One need to have the systemd unit file in place (already loaded by systemd) and configure a resource using the following command:

```
$ crm configure primitive $RESOURCE_NAME systemd:$SERVICE_NAME
```

One can do the same using `pcs` via the following command:

```
$ pcs resource create $RESOURCE_NAME systemd:$SERVICE_NAME
```

The `$SERVICE_NAME` can be any service managed by a systemd unit file, and it needs to be available for the cluster nodes.

### Further reading

- ClusterLabs website
- The OCF resource-agent developer's guide

### Pacemaker fence agents

From the ClusterLabs definition:

> A **fence agent** (or **fencing agent**) is a **stonith**-class resource agent.
>
> The fence agent standard provides commands (such as `off` and `reboot`) that the cluster can use to fence nodes. As with other resource agent classes, this allows a layer of abstraction so that Pacemaker doesn't need any knowledge about specific fencing technologies — that knowledge is isolated in the agent.

Currently, the `fence-agents` binary package has been split into two: `fence-agents-base` and `fence-agents-extra`. The `fence-agents-base` binary package contains a set of curated agents which the Ubuntu Server team continuously runs tests on to make sure everything is working as expected. All the other agents previously in the `fence-agents` binary package are now moved to the `fence-agents-extra`.

The `fence-agents-base` binary package contains the following agents in the latest Ubuntu release:

- `fence_ipmilan`
  - `fence_idrac`
  - `fence_ilo3`
  - `fence_ilo4`
  - `fence_ilo5`

    - fence_imm
    - fence_ipmilanplus
- fence_mpath
- fence_sbd
- fence_scsi
- fence_virsh

All of these agents are in main and are fully supported. All other agents, in fence-agents-extra, are supported by upstream but are not curated by the Ubuntu Server team.

For the fence agents provided by fence-agents-base, we will briefly describe how to use them.

**Note**

There are two well known tools used to manage fence agents, they are crmsh and pcs. Here we present examples with both, since crmsh is the recommended and supported tool until Ubuntu 22.10 Kinetic Kudu, and pcs is the recommended and supported tool from Ubuntu 23.04 Lunar Lobster onwards. For more information on how to migrate from crmsh to pcs *refer to our migration guide*.

### fence_ipmilan

The content of this section is also applicable to the following fence agents: fence_idrac, fence_ilo3, fence_ilo4, fence_ilo5, fence_imm, and fence_ipmilanplus. All of them are symlinks to fence_ipmilan.

From its manpage:

> fence_ipmilan is an I/O Fencing agent which can be used with machines controlled by IPMI. This agent calls support software ipmitool. WARNING! This fence agent might report success before the node is powered off. You should use -m/method onoff if your fence device works correctly with that option.

In a system which supports IPMI and with ipmitool installed, a fence_ipmilan resource can be configured with the following command:

```
$ crm configure primitive $RESOURCE_NAME stonith:fence_ipmilan \
            ip=$IP \
            ipport=$PORT \
            username=$USER \
            password=$PASSWD \
            lanplus=1 \
            action=$ACTION
```

One can do the same using pcs via the following command:

```
$ pcs stonith create $RESOURCE_NAME fence_ipmilan \
            ip=$IP \
```

(continues on next page)

(continued from previous page)

```
            ipport=$PORT \
            username=$USER \
            password=$PASSWD \
            lanplus=1 \
            action=$ACTION
```

Where `$IP` is the IP address or *hostname* of fencing device, `$PORT` is the TCP/UDP port to use for connection, `$USER` is the login name and `$PASSWD` its password, and `$ACTION` is the fencing actions which by default is `reboot`.

This is one way to set up `fence_ipmilan`, for more information refer `to its manpage`.

### fence_mpath

From its manpage:

> `fence_mpath` is an I/O fencing agent that uses SCSI-3 persistent reservations to control access multipath devices. Underlying devices must support SCSI-3 persistent reservations (SPC-3 or greater) as well as the "preempt-and-abort" subcommand. The `fence_mpath` agent works by having a unique key for each node that has to be set in `/etc/multipath.conf`. Once registered, a single node will become the reservation holder by creating a "write exclusive, registrants only" reservation on the device(s). The result is that only registered nodes may write to the device(s). When a node failure occurs, the fence_mpath agent will remove the key belonging to the failed node from the device(s). The failed node will no longer be able to write to the device(s). A manual reboot is required.

One can configure a `fence_mpath` resource with the following command:

```
$ crm configure primitive $RESOURCE_NAME stonith:fence_mpath \
            pcmk_host_map="$NODE1:$NODE1_RES_KEY;$NODE2:$NODE2_RES_KEY;$NODE3:
$NODE3_RES_KEY" \
            pcmk_host_argument=key \
            pcmk_monitor_action=metadata \
            pcmk_reboot_action=off \
            devices=$MPATH_DEVICE \
            meta provides=unfencing
```

One can do the same using `pcs` via the following command:

```
$ pcs stonith create $RESOURCE_NAME fence_mpath \
            pcmk_host_map="$NODE1:$NODE1_RES_KEY;$NODE2:$NODE2_RES_KEY;$NODE3:
$NODE3_RES_KEY" \
            pcmk_host_argument=key \
            pcmk_monitor_action=metadata \
            pcmk_reboot_action=off \
            devices=$MPATH_DEVICE \
            meta provides=unfencing
```

The `$NODE1_RES_KEY` is the reservation key used by this node 1 (same for the others node with access to the multipath device), please make sure you have `reservation_key <key>` in the

`default` section inside `/etc/multipath.conf` and the multipathd service was reloaded after it.

This is one way to set up `fence_mpath`, for more information please check its manpage.

### fence_sbd

From its manpage:

> fence_sbd is I/O Fencing agent which can be used in environments where SBD can be used (shared storage).

With STONITH Block Device (SBD) configured on a system, the `fence_sbd` resource can be configured with the following command:

```
$ crm configure primitive $RESOURCE_NAME stonith:fence_sbd devices=$DEVICE
```

One can do the same using `pcs` via the following command:

```
$ pcs stonith create $RESOURCE_NAME fence_sbd devices=$DEVICE
```

This is one way to set up `fence_sbd`, for more information refer `to its manpage`.

### fence_scsi

From its manpage:

> `fence_scsi` is an I/O fencing agent that uses SCSI-3 persistent reservations to control access to shared storage devices. These devices must support SCSI-3 persistent reservations (SPC-3 or greater) as well as the “preempt-and-abort” subcommand. The `fence_scsi` agent works by having each node in the cluster register a unique key with the SCSI device(s). Reservation key is generated from “node id” (default) or from “node name hash” (RECOMMENDED) by adjusting “key_value” option. Using hash is recommended to prevent issues when removing nodes from cluster without full cluster restart. Once registered, a single node will become the reservation holder by creating a “write exclusive, registrants only” reservation on the device(s). The result is that only registered nodes may write to the device(s). When a node failure occurs, the `fence_scsi` agent will remove the key belonging to the failed node from the device(s). The failed node will no longer be able to write to the device(s). A manual reboot is required.

A `fence_scsi` resource can be configured with the following command:

```
$ crm configure primitive $RESOURCE_NAME stonith:fence_scsi \
            pcmk_host_list="$NODE1 $NODE2 $NODE3" \
            devices=$SCSI_DEVICE \
            meta provides=unfencing
```

One can do the same using `pcs` via the following command:

```
$ pcs stonith create $RESOURCE_NAME fence_scsi \
            pcmk_host_list="$NODE1 $NODE2 $NODE3" \
            devices=$SCSI_DEVICE \
            meta provides=unfencing
```

The pcmk_host_list parameter contains a list of cluster nodes that can access the managed SCSI device.

This is one way to set up fence_scsi, for more information refer to its manpage.

### fence_virsh

From its manpage:

> fence_virsh is an I/O Fencing agent which can be used with the virtual machines managed by libvirt. It logs via ssh to a dom0 and there run virsh command, which does all work. By default, virsh needs root account to do properly work. So you must allow ssh login in your sshd_config.

A fence_virsh resource can be configured with the following command:

```
$ crm configure primitive $RESOURCE_NAME stonith:fence_virsh \
            ip=$HOST_IP_ADDRESS \
            login=$HOST_USER \
            identity_file=$SSH_KEY \
            plug=$NODE \
            ssh=true \
            use_sudo=true
```

One can do the same using pcs via the following command:

```
$ pcs stonith create $RESOURCE_NAME fence_virsh \
            ip=$HOST_IP_ADDRESS \
            login=$HOST_USER \
            identity_file=$SSH_KEY \
            plug=$NODE \
            ssh=true \
            use_sudo=true
```

This is one way to set up fence_virsh, for more information refer to its manpage.

In order to avoid running the resource in the same node that should be fenced, we need to add a location restriction:

```
$ crm configure location fence-$NODE-location $RESOURCE_NAME -inf: $NODE
```

Using pcs:

```
$ pcs constraint location $RESOURCE_NAME avoids $NODE
```

### Further reading

- ClusterLabs website
- Fence agents API documentation

### See also

- How-to: *Set up a Distributed Replicated Block Device (DRBD)*
- Reference: *High availability*

# 4.10. Debugging

*These debugging pages* are for readers interested in packaging and Ubuntu development.

## 4.10.1. Debugging

These pages are for those interested in packaging and Ubuntu development.

- *About debuginfod*
- *Debug symbol packages*

### About debuginfod

`debuginfod` is a service for software developers. It helps with diagnosing issues in software and centralises the storage of debug symbols, source code, etc.

One of the main advantages to `debuginfod` is that debugging information can be retrieved on-demand for packages shipped with Ubuntu without the need to *manually install* the debug symbol packages.

Ubuntu maintains its own `debuginfod` service, which regularly indexes the debug symbols present in `ddebs` and other packages and serves this information over HTTPS.

Currently, the service only provides DWARF information. There are plans for it to also index and serve source-code in the future.

### Using the service

`debuginfod` is indexing `ddebs` packages from all supported Ubuntu releases. Once a release goes unsupported, we stop indexing `ddebs` from it and eventually stop serving debug symbols for its packages.

From Kinetic onwards, when you install GNU Debugger (GDB) your system will be automatically configured to use Ubuntu's `debuginfod` service. For previous Ubuntu releases, you can manually enable the service by setting the `DEBUGINFOD_URLS` environment variable in your shell. If you use Bash, you can do that by adding the following snippet to your `~/.bashrc`:

```
export DEBUGINFOD_URLS="https://debuginfod.ubuntu.com"
```

When you run *GDB*, and if you have the `DEBUGINFOD_URLS` variable in your environment, you will be asked whether you would like to use the service. If you want to make sure that GDB always uses `debuginfod`, you can put the following snippet inside your `~/.gdbinit` file:

```
set debuginfod enabled on
```

The debug symbol files will be downloaded on-the-fly during your debugging session, and will be saved locally inside the `$XDG_CACHE_HOME/.debuginfod_client/` directory. If `$XDG_CACHE_HOME` is empty, then `~/.cache/debuginfod_client` is used instead.

You can safely remove this directory or any files inside it; they will only be downloaded again if and when they are needed.

### Example session with GDB

If you have enabled use of `debuginfod` on your system, here is what happens when you invoke GDB to debug a binary from an Ubuntu package:

```
$ gdb -q program
Reading symbols from program...

This GDB supports auto-downloading debuginfo from the following URLs:
https://debuginfod.ubuntu.com
Enable debuginfod for this session? (y or [n])
```

When you answer `y` to the question above, GDB will download the debug symbols for `program`:

```
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Downloading 0.20 MB separate debug info for /home/ubuntu/program
Reading symbols from /home/ubuntu/.cache/debuginfod_client/
c0fbda15a807f880e9d0b2dcc635eeeb1f0f728e/debuginfo...
(gdb)
```

### Opting out of the service

If, for some reason, you prefer not to use the service, you can opt-out of it by unsetting the `DEBUGINFOD_URLS` environment variable. This can be done by putting the following snippet inside your shell's configuration file:

```
unset DEBUGINFOD_URLS
```

You can also disable GDB's willingness to use `debuginfod` by putting the following snippet inside your `~/.gdbinit`:

```
set debuginfod enabled off
```

### How does `debuginfod` find the debug symbols for the binary I am debugging?

`debuginfod` relies on a unique hash that identifies binaries and shared libraries called **Build-ID**. This 160-bit SHA-1 hash is generated by the compiler, and can be consulted using tools like `readelf`:

```
$ readelf -n /usr/bin/bash

Displaying notes found in: .note.gnu.property
  Owner                Data size        Description
  GNU                  0x00000020       NT_GNU_PROPERTY_TYPE_0
      Properties: x86 feature: IBT, SHSTK
        x86 ISA needed: x86-64-baseline

Displaying notes found in: .note.gnu.build-id
  Owner                Data size        Description
```

(continues on next page)

(continued from previous page)

```
  GNU                  0x00000014       NT_GNU_BUILD_ID (unique build ID
bitstring)
    Build ID: 3e770d2cd0302c6ff2a184e8d2bf4ec98cfcded4

Displaying notes found in: .note.ABI-tag
  Owner                Data size        Description
  GNU                  0x00000010       NT_GNU_ABI_TAG (ABI version tag)
    OS: Linux, ABI: 3.2.0
```

When you are debugging a program, GDB will send the program's Build-ID to the `debuginfod` server, which will check if it has the corresponding debug information for that binary/library. If it does, then it will send the debug symbols via HTTPS back to GDB.

### Can `ddebs` packages co-exist with `debuginfod`?

Yes. GDB will try to use local debug information if available. That means that if you have a `ddeb` package installed that provides the necessary debug symbols for the program being debugged (or if you have already downloaded that information from the `debuginfod` service earlier), then GDB will use it in favour of performing the download.

### Can I use `debuginfod` with my own binary that links against system libraries?

Yes! `debuginfod` will not be able to provide any debug symbols for your own program, but it will happily serve debug information for any system libraries that your program links against.

## Debug symbol packages

If you want to debug a crash – whether in a project you are developing yourself or from a third-party package – or if you frequently need the debug symbols for specific libraries, it might be helpful to install them permanently on your system if you can't use debuginfod.

This document describes how to set up the debugging symbol packages (`*-dbg.deb` and `*-dbgsym.ddeb`). You might need to do this when you are performing tasks like a Backtrace or using Valgrind.

## Debuginfod

If you are on Ubuntu Jammy (22.04) or later, you don't need to worry about installing debug symbol packages since the Ubuntu project maintains a *Debuginfod* server. GNU Debugger (GDB) and other debuginfo-consumer applications support Debuginfod (mostly) out of the box. For more information about it, please refer *to our Debuginfod guide*.

You will only need to follow the methods outlined in this section if you are on Ubuntu Focal (20.04) or earlier.

## Getting -dbgsym.ddeb packages

If you are debugging without debuginfod, your first step will be to enable the `ddebs.ubuntu.com` repository as described in this section, which will provide access to the `-dbgsym` packages.

In the rare cases where the `-dbgsym` package is not available, you might need to install the `-dbg` package instead. The subsequent section (Manual install of debug packages) provides

more information about this case.

### Import the signing key

Import the debug symbol archive signing key from the Ubuntu server. On Ubuntu 18.04 LTS and newer, run the following command:

```
sudo apt install ubuntu-dbgsym-keyring
```

### Create a ddebs.list file

Create an `/etc/apt/sources.list.d/ddebs.list` by running the following line at a terminal:

```
echo "Types: deb
URIs: http://ddebs.ubuntu.com/
Suites: $(lsb_release -cs) $(lsb_release -cs)-updates $(lsb_release -cs)-proposed
Components: main restricted universe multiverse
Signed-by: /usr/share/keyrings/ubuntu-dbgsym-keyring.gpg" | \
sudo tee -a /etc/apt/sources.list.d/ddebs.sources
```

You can also add these repositories in your software sources from the Ubuntu software center or from Synaptic (refer to this article, especially the section on adding other repositories). You will need to add lines like:

```
deb http://ddebs.ubuntu.com focal main restricted universe multiverse
```

> **Note**
>
> Make sure you replace “focal” with the Ubuntu release name you’re using.

### Update package list

Run the following to update your package list or click the Reload button if you used the Synaptic Package Manager:

```
sudo apt-get update
```

### Manual install of debug packages

To install the debug symbol package (`*-dbgsym.ddeb`) for a specific package, you can now invoke:

```
sudo apt-get install PACKAGE-dbgsym
```

For example, to install the debug symbols for `xserver-xorg-core`:

```
sudo apt-get install xserver-xorg-core-dbgsym
```

As mentioned in the section above, some packages will ship their debug symbols via `*-dbg.deb` packages instead. Using `glibc` as an example, you can install its debug symbols using:

```
sudo apt-get install libc6-dbg
```

This procedure will install the debug symbol package for a single package only. It is likely that the binary uses shared libraries in other packages, and their debug symbols may be needed in order to obtain a readable stack trace or perform other debugging tasks.

### The debian-goodies tool

You can use the `find-dbgsym-packages` command from the `debian-goodies` package to find debug symbols for a core file, running PID or binary path.

For a binary path it only finds debug symbols for the actual binary itself, and not any dynamically linked library dependencies or other libraries loaded at runtime. For that functionality to work you need to use either a core file or a running PID (which is the preferred method).

This tool will find both `-dbg` and `-dbgsym` style packages. However it only finds debug symbols for APT repositories that are currently enabled and updated, so you need to ensure that you enable at least the `ddebs.ubuntu.com` archive as described above. For a Launchpad PPA or the Ubuntu Cloud Archive you need to add another source line with the component changed from `main` to `main/debug`:

```
sudo apt install debian-goodies
find-dbgsym-packages [core_path|running_pid|binary_path]
```

### "debs" versus "ddebs"

It used to be the case that Debian/Ubuntu maintainers needed to manually create debug symbol packages as part of the packaging process, and included them in the same repository as their binary package. These debug symbol packages had the `-dbg.deb` suffix, so for example, both the `apache2-bin.deb` package and the `apache2-dbg.deb` package would be placed in the same repository. However, since the process is a manual one, not every maintainer did this, and the `.deb` package was not always kept up-to-date.

Modern package building tools automatically create the debug symbol packages when binary packages are compiled on Ubuntu servers, so the older (manual) process is no longer used. These automatically-created debug symbol packages have the `-dbgsym.ddeb` suffix. Unlike `-dbg.deb` packages, `-dbgsym.ddeb` packages are hosted in their own separate repository, since these packages are used relatively rarely.

You can choose to use either `-dbg.deb` or `-dbgsym.ddeb` packages, but for any given binary package only one debug symbol package can be used at once.

### See also

- External: The Ubuntu Packaging Guide

# 5. Contribute to this documentation

Contributing to documentation can be a fantastic way to get started as a contributor to open source projects, no matter your level of experience!

The Ubuntu Server documentation is a collaborative effort between Canonical and the Ubuntu community. All types of contributions are welcome, whether you are new to Ubuntu Server and want to highlight something you found confusing, or you're an expert and want to create "how-to" guides to help others.

We hope to make it as easy as possible to contribute. If you find any part of our process doesn't work well for you, please let us know!

## 5.1. The Open Documentation Academy

Ubuntu Server is a proud member of the Canonical Open Documentation Academy (CODA). If you are a newcomer to making open source contributions, or new to technical writing and want to boost your skills – or both! – we will be glad to help.

Check out the CODA repository for guidance and useful resources on getting started.

## 5.2. Prerequisites

There are some prerequisites to contributing to Ubuntu Server documentation.

- **Code of Conduct** You will need to read and agree to the Ubuntu Code of Conduct. By participating, you implicitly agree to abide by the Code of Conduct.
- **GitHub account** You need a GitHub account to create issues, comment, reply, or submit contributions.

  You don't need to know `git` before you start, and you definitely don't need to work on the command line if you don't want to. Many documentation tasks can be done using GitHub's web interface. On the command line, we use the standard "fork and pull" process.
- **Licensing** The first time you contribute to a Canonical project, you will need to sign the Canonical License agreement (CLA). If you have already signed it, e.g. when contributing to another Canonical project, you do not need to sign it again.

  This license protects your copyright over your contributions, including the right to use them elsewhere, but grants us (Canonical) permission to use them in our project. You can read more about the CLA before you sign the CLA.

## 5.3. The Ubuntu Server docs overview

This documentation is hosted in GitHub and rendered on Read the Docs. You need to create a GitHub account to participate, but you do not need a Read the Docs account.

## 5.4. Contribution types

There are many different ways to contribute to the Ubuntu Server documentation, including many options that don't require any coding knowledge. To find out more, check out the *Types of contributions* page.

## 5.5. General workflow

Most contributions are made on GitHub by working on your machine and submitting your changes in a pull request (PR) via the command line. This workflow is outlined in these broad steps:

- *Find an issue to work on*
- *Clone the documentation to your machine*
- Work on the issue (*get help* if you need it!)
- *Test your contribution*
- *Submit it for review*

The Ubuntu Server documentation is very large and comprehensive. If you can't find an issue you want to work on, feel free to look around the docs and see what improvements you think you can make.

For spelling and grammatical changes on a page, which are quick and easy to submit, feel free to create a PR. For more substantial changes or suggestions, we suggest creating an issue first, so that we can discuss and agree on an approach before you spend time working on it.

Make sure you check the issue list before submitting a PR - if you start working on a task that is listed and already assigned to someone else, we won't be able to accept your PR.

## 5.6. Thank you!

Lastly, we would like to thank you for spending your time to help make the Ubuntu Server documentation better. Every step in the right direction is a step worth taking, no matter how large or small.

Check out what *our contributors* have been working on!

### 5.6.1. Types of contributions

There are many different ways to contribute, no matter your level of prior knowledge. Our issue labels include the following options that you can use to filter the issues list according to the type of contribution you want to make.

#### Technical

These types of tasks might require you to have a good amount of prior experience with servers, networking, hardware, or related topics. This might include:

- Technical reviews
- Technical updates

It could also include creating content, if you want to submit a guide on how to accomplish a specific task, or an explanation of a topic.

Find technical issues to work on.

### Coding options

Some issues require specific knowledge or technical expertise, such as:

- Creating vale rules
- Improving GitHub actions and workflows
- Creating graphics from code using Mermaid
- Fixing incorrect code snippets in the documentation

Find coding issues to work on.

### Low-code options

If you are happy to work with Markdown (or MyST) formatting, you can edit our documentation directly to fix:

- Spelling and grammatical errors.
- Updating broken links.
- Unclear explanations or imprecise language.
- Testing tutorials and how-to guide steps.
- Fixing or creating graphics from code using Mermaid. If you are not familiar with the syntax, they have a live online generator tool that will create the code for a diagram you design.

Find low-code issues to work on.

### No-code options

If you spot something in the documentation that could be improved (or needs to be corrected), the best way to contribute is to report it!

### Creating issues

Since we use GitHub issues to track problems with the documentation, creating issues is a great way to contribute. You can use the “Give feedback” button at the top of any page in the documentation to report the issue you have found.

Take care to describe any issue in detail so that the reviewer can understand the problem clearly. A detailed issue might include a link to the affected part of the documentation, a description of the problem and – optionally – a suggested solution.

This can also be a method for suggesting improvements or giving any other feedback you might want to share!

### User-experience testing

We want the Ubuntu Server documentation to be as accessible as possible. If you have feedback about any aspect of your experience of using our documentation, this is incredibly valuable to us! If you have identified any areas we can improve upon please let us know by opening an issue.

Please check the issues list first to see if someone has already reported the same (or a similar) issue – if they have, then leave your thoughts as a reply on that issue. By keeping discussions together in one place, it makes it easier for us to understand the impact and gather opinions from affected readers.

### Copy editing

You can edit individual documentation pages directly in the GitHub web interface if you don't want to work on the command line. To do this, select the pencil icon next to the "Give feedback" button. For details about how to use the web editor, you can refer to their documentation which has several handy guides.

Find no-code issues to work on.

## 5.6.2. Find issues to work on

We use GitHub issues to track documentation tasks. Start by checking out the issues list to see if there are any tasks you'd like to work on. We use labels to help filter the tasks that need to be done and to show whether they're assigned to someone or not.

### Get an issue assigned

When you find a task you want to work on, leave a comment on the issue saying you'd like to be assigned the task, with an estimated date for when you hope to complete it. One of the documentation maintainers will respond and assign that task to you.

> **Note**
>
> There is no time limit for completing a task, but if you need more time, need help, or won't be able to complete the issue after all, make sure to comment on the issue to let us know. It's quite normal for plans to change, and handing an issue back won't prevent you from picking up more issues in the future!
>
> Issues that have no work being done on them, with no updates from the author, will be considered stale after one month and will be unassigned to allow others a chance to pick them up.

Each issue can be worked on by a single person, and each person can work on one issue at a time. You can see which issues are unassigned by selecting "Assigned to nobody" from the "Assignee" drop-down menu (or use this link as a shortcut).

### Issue labels

The Ubuntu Server team regularly triages issues, and we use the following issue labels to help manage them. If you are looking for certain types of tasks, you can filter the issues list by label. The labels are grouped together into the following categories.

Note that we don't use the **help wanted** or **good first issue** labels. All the issues we have can be considered as available for anyone who wants to work on them.

### Coding level

These labels correspond to the different types of contributions that we accept according to the details in the *Types of contributions* page.

- *code: coding*
- *code: low-code*
- *code: non-code*
- *code: technical*

### Content

These labels are used to mark out issues related to content. They will often also have a Diátaxis label to give more context.

- *content: edit/update*

  Edit or update existing content for consistency, accuracy, style, completeness, etc.

- *content: new*

  Add new documentation for a specific tool, feature, or function.

### For

- *For: Open Documentation Academy*

  This issue has enough context/detail to be picked up by members of the Canonical Open Documentation Academy.

- *For: Server ToDo*

  This issue has been identified as something the Ubuntu Server team needs to work on or fix.

### Review

This issue or Pull Request needs a specific type of review.

- *review: wording*

  Review the quality, clarity, and consistency of the wording.

- *review: technical*

  Review the technical accuracy, completeness, and up-to-dateness of the content.

### State labels

The set of *State:* issues labels are for the maintainers of the repository to flag possible problems with the listed issue or Pull Request.

- *State: duplicate*

  Used when an issue is closed.

- *State: Incomplete*

  Used when there isn't enough information in an issue to resolve the problem.

- *State: Invalid*

  Used when an issue is closed.

- *State: WIP*

  Used for Pull Requests that have been submitted but are not yet ready for review.

#### After you find an issue

After you have found an issue you want to work on, and have been assigned the issue, you will want to either use the GitHub web interface to create a quick pull request, or fetch the documentation to your own machine so you can *build the documentation locally* and work on your own local copy.

### 5.6.3. Build the documentation locally

To contribute to the Ubuntu Server documentation you will first need to create your own fork of the repository, then clone that fork to your machine. If you're not sure what this means or need some help getting started, check out the working with git guide from the Open Documentation Academy which will walk you through the process and explain this terminology. Just remember to change all instances of `open-documentation-academy` in the commands to `ubuntu-server-documentation`!

If you're already familiar with the process, then use the steps below to clone the repository and build the documentation. Don't forget to set up your own fork!

#### Get the docs

Before you can start working on an issue, first you will need to download the documentation. To do this, you can run:

```
git clone git@github.com:canonical/ubuntu-server-documentation.git
```

This will create a new folder on your machine called `ubuntu-server-documentation` that contains the contents of this repository.

You can then navigate to this folder using:

```
cd ubuntu-server-documentation
```

#### Install required software

To build the documentation, you will first need to install some necessary dependencies on your system with the following commands:

```
sudo apt update
sudo apt install make python3 python3-venv python3-pip
make install
```

### Create a new branch

Before making any changes, ensure the `main` branch on your machine is up-to-date with any recent changes made to the remote repository:

```
git pull
```

Now, create a branch and switch to it with the following:

```
git checkout -b my-new-branch
```

Remember to give your branch a more descriptive name than `my-new-branch`. In this way, even if you are working on multiple branches, you will know at a glance what each of them is for.

### Work on the docs

You're now ready to start working on the docs! You should run the following command before you start, to build a live preview:

```
make run
```

This will build **and serve** the documentation at http://127.0.0.1:8000/. It will watch the folder, and whenever you save changes to a file, this URL will give update the preview to show your changes (or warn you if something has gone horribly wrong!).

If you are building locally on an Ubuntu Cloud VM or a container, you may experience issues accessing the page from your host's browser. To resolve this, add the export variable to your shell by running the following command:

```
export SPHINX_HOST=0.0.0.0
```

Running the `make run` command should then work as expected.

> **Note**
>
> If you have problems getting the documentation to run on your machine, reach out to the team or leave a comment on your issue to get additional support.

### Writing guidance

Once your environment is set up and you have been able to get your local copy running without any build errors, you can check out our *guidance for writing* section to find out about our style guide and other important information.

### Submit your changes

Once you have made your changes and are happy with them, you can *find out how to submit them*.

## 5.6.4. Guidance for writing

If you get stuck at all – please don't hesitate to reach out for help!

### Files and structure

The documentation pages are all written in standard Markdown (`.md` file types) with MyST support for more advanced elements if you want them. All the documentation pages are in the `docs/` folder, then split into a subfolder for the Diátaxis section it belongs in.

The structural elements, such as landing pages, are written in reStructuredText (`.rst` file types). It shouldn't be necessary to change them, unless you are making substantial restructuring changes, or adding/deleting pages.

If you do need to change a landing page, they are found in the `subsections/` folder. Each Markdown page is only called once, by the landing page for that section.

> **Tip**
>
> For example, if a page appears in the How-to section about Virtualisation, the landing page you need will be `subsections/how-to-virtualisation.rst`.

### Versioning policy

Ubuntu Server is relatively stable from one LTS to the next, which means the documentation is not substantially re-written between releases. Due to this, we opt not to have separate branches for each Ubuntu release. Instead, we call out any changes or behaviour particular to a specific release.

### Versioning by admonition

Small, specific changes can be expressed in a "note" admonition block:

```
:::{note}
For Ubuntu 24.04 LTS (Noble) onwards, the recommended method for ...
:::
```

or

````
```{note}
For Ubuntu 24.04 LTS (Noble) onwards, the recommended method for ...
```
````

Any content not called out as belong to a specific release (or set of releases) is assumed to be valid for all supported releases.

### Versioning by tabs

For large changes, where whole sets of commands may be different between releases, we can split the instructions using tabs.

The MyST documentation has a helpful demonstration of the syntax and how tabs work.

It's good to include a "sync" keyword, so that if users are on a particular release, they only need to choose their release one time. Default to the release number, e.g. *24.04* or *21.10*.

In all cases, order the tabs and content from "most recent" to "oldest" release – this ensures that the most recent release is always shown in the left-most tab and provides a more consistent experience.

If you would like to see an example of this in the live documentation, *the QEMU page* uses three tabs in the section about using 1024 vCPUs.

### Style guide

Consistency of writing style in documentation is vital for a good user experience. In the Server Guide, we use the Canonical documentation style guide.

### Language

We use US English. It's a good idea to set your spellchecker to `en-US`. We use an automated spelling checker that sometimes throws errors about terms we would like it to ignore:

- If it complains about a file name or a command, enclose the word in backticks (`) to render it as inline code.
- If the word is a valid acronym or a well-known technical term (that should not be rendered as code), add it to the spelling exception list, `.custom_wordlist.txt` (terms should be added in alphabetical order).

Both methods are valid, depending on whether you want the term to be rendered as normal font, or as inline code (monospaced).

### Acronyms

Acronyms should always be capitalised.

They should always be expanded the first time they appear on a page, and then can be used as acronyms after that. E.g. YAML should be shown as Yet Another Markup Language (YAML), and then can be referred to as YAML for the rest of the page.

All acronyms should also be in our glossary – you can add the term to the glossary if it is not present.

### Links

The first time you refer to a package or other product, you should make it a link to either that product's website, or its documentation, or its manpage.

Links should be from reputable sources (such as official upstream docs). Try not to include blog posts as references if possible.

Try to use links sparingly in the page. If you have a lot of useful references you think the reader might be interested in, feel free to include a "Further reading" section at the end of the page.

### Writing style

Try to be concise and to-the-point in your writing.

It's OK to be a bit light-hearted and playful in your writing, but please keep it respectful, and don't use emoji (they don't render well in documentation).

It's also good practice not to assume that your reader will have the same knowledge as you. If you're covering a new topic (or something complicated) then try to briefly explain, or link to supporting explanations of, the things the typical reader may not know, but needs to.

### Markdown elements

### Sections and headings

Avoid skipping header levels in your document structure, i.e., a level 2 header (##) should be followed by a level 3 sub-header (###) not level 4.

```
# Heading level 1
## Heading level 2
### Heading level 3
#### Heading level 4
```

Always include some text between headers if you can. You can see this demonstrated between this section's heading and the one above it (Markdown elements). It looks quite odd without text to break the headers apart!

### Semantic markup

We encourage (but do not mandate) the use of semantic mark-up where possible. See Roles in the Sphinx documentation for an overview of inline semantic roles available by default. The most helpful ones for this project are:

```
Keyboard keys: {kbd}`Ctrl`, {kbd}`C`
```

Rendered as `Ctrl`, `C`

```
Manpages: {manpage}`package-name(section)` (e.g. {manpage}`dpkg(1)`)
```

Rendered as `dpkg(1)`

It is not necessary to provide the hardcoded URL to a manpage - they are generated when Sphinx rebuilds the documentation so that they are always up to date.

### Lists

For a numbered list, use `1.` in front of each item. The numbering will be automatically rendered, so it makes it easier for you to insert new items in the list without having to re-number them all:

```
1. This is the first item
1. This is the second
1. This is the third
```

Unless a list item includes punctuation, don't end it with a full stop. If one item in a list needs a full stop, add one to all the items in that list.

### Code blocks

Enclose a code block with three backticks:

````
```text
  ```yaml
  Some code block here
  ```
```
````

Use separate command input blocks from command output blocks. We do this because we have a "copy code" feature in the documentation, and it's more convenient for the reader to copy the code if it only contains the input.

Avoid using a command line prompt (e.g. $ or #) in an input block if possible, and precede the output block with some kind of text that explains what's happening. For example:

````
```bash
uname -r
```

Produces the following output:

```text
4.14.151
```
````

It can also be helpful to orient the reader with what they *should* be seeing if you can include examples (although this is optional).

Use a single backtick to mark inline commands and other string literals, like `paths/to/files`.

## 5.6.5. Submitting your work

You should submit your pull request (PR) to the **Ubuntu Server documentation repository** (repo), whether you claimed your issue via the ODA repo or the Ubuntu Server repo.

If you need help with any aspect of the process (forking the repo, committing, pushing, etc) then refer to the getting started with git guide on the ODA repo, which will guide you through those steps as you construct your changes.

### Testing your changes

Before pushing your changes or creating a pull request, you should first test the documentation to catch any spelling errors, broken links, or similar. This allows the reviewers to focus on the main changes you are proposing and makes the review process more efficient.

You can run:

```
make spelling
make linkcheck
```

To perform a full spelling and link check. You can also run `make` by itself to see a list of all the possible `make` targets.

### Check if you need redirects

If you rename, move or delete an existing file, a corresponding redirect must be created to ensure users don't run into 404 errors when clicking links in the published documentation.

### Internal redirects

To set up a redirect from one file path to another, add a line to the end of the redirects.txt` file in the root directory, in the following format:

```
redirect/path/from/ redirect/path/to/
```

Note that since we use *dirhtml* to build, the built documentation is in the format *path/to/-file/index.html* where *file* corresponds to the file name you are redirecting. This means that you only need a trailing slash at the end of the file name, without the file extension. See the Sphinx Rediraffe docs for more guidance, or reach out to us for help.

### External redirects

Rediraffe doesn't currently handle redirects from a page to an external website. To redirect outside of the Server documentation, you will need to set up a redirect in the *custom_conf.py* file in the root directory.

Under the Redirects section, you can add the source page and the target page as follows:

```
redirects = {
    "example/source": "https://exampletarget.org",
    "how-to/containers/lxc-containers": "https://linuxcontainers.org/lxc/
documentation/"
}
```

When you set up a redirect in this way, the path of the source file you're redirecting from should include everything *after* the base URL (https://documentation.ubuntu.com/server).

### Manual testing

If your contribution contains any code or process steps, it's a good idea to do a final run-through of your guide from start to finish in a clean environment, just to make sure everything works as you expected.

Particularly check the code snippets – does the output in your terminal match what you've presented in the guide?

### Submit a pull request

- Make sure all your proposed changes are committed:
  - `git status` will show your uncomitted changes
  - Select the ones you want to add to each commit using `git add <filename>`
  - Commit your selected changes using `git commit`.

> **Note**
>
> Try to group your changes "logically". For example, if you have one set of changes that modifies spelling in 10 files, and another set of changes that modifies formatting in 10 different files, you can group them into two commits (one for spelling, and one for formatting). You don't need a separate commit for every file.

- Push the changes to your fork: `git push <your username> <branch name>`
- Create a Pull Request against the Ubuntu Server documentation repository.
- Link your pull request to your issue, typically by adding `fixes #<issue-number>` to your description.
- Give your pull request a description and click on submit!

#### Read the Docs preview

You will be able to see a live preview of your documentation as it will appear on Read the Docs at the bottom of your pull request's page – where the checks appear, click on "Show all checks" and next to the "docs/readthedocs" line, click on "Details".

#### Reviews

After you have submitted your PR, one of the Ubuntu Server team maintainers will be in touch to review it. Depending on time zones, there may be a delay in your PR being reviewed. Please be patient!

One or more of the Ubuntu Server team maintainers will review the changes you have proposed, and they will either "Approve" the changes, or leave some feedback and suggested changes (with reasons). If you agree with the feedback, you can make the suggested changes, and the reviewer will approve the PR.

> **Note**
>
> The team has adopted the Conventional Comments approach with the intention of making feedback easier to parse.

If you disagree with any parts of the review, it's OK to discuss this with the reviewer – feedback is made in good faith, and is intended to help strengthen your contribution. This is a collaboration, after all! It's quite normal to have some back-and-forth on a PR, but it should be a respectful dialogue on all sides.

Once the discussion has concluded, and you have made any agreed changes, the PR will be approved and then merged. Congratulations (and thank you)! You are now an open source contributor!

### 5.6.6. Getting help

You might occasionally run into problems – this is normal and expected! We will do our best to help.

### Directly on GitHub issues

Each issue should contain a description of the task to be completed, and some suggestions for how you might want to tackle it if there are several options. You can always leave a question on an issue to ask for more information if there is something missing or some clarification if the issue is confusingly presented.

If you have not been able to work on the issue for a while, please leave a comment on it. This helps us know if you intend to complete the task, and to reassign the task to someone else if you know you will not be able to finish it.

If you want help from another contributor, that's also fine! Leave a comment describing the problem you're facing and what help you need to resolve it.

### Help with your submission

Your pull request will be assigned a reviewer from among the repository maintainers. You can contact them by leaving a comment on your own PR – you can also "ping" them in that comment to send them a notification by using `@<their username>`.

### Community forums

For more general questions about Ubuntu Server, you're welcome to create a post on the Ubuntu Server forum.

You can also use the Canonical Open Documentation Academy (CODA) forum, which is the hub for questions about documentation specifically. It includes our Getting started guide and links to our weekly Documentation office hours, alongside meeting notes, updates, external links and discussions.

### Synchronous chat

For more interactive chat, the documentation team can be found on Matrix.

### Calendar

Subscribe to the Documentation event calendar. Not only does this include the Documentation office hours, it will also include any other discussion or training events we organise.

## 5.6.7. Our contributors

Curious about what our contributors have been doing? As you can see from the sections below (organised alphabetically), there is a huge variety of fixes, content updates and improvements being made across the Ubuntu Server documentation. The most recently submitted PRs are at the top of each table.

This documentation is a testament to the valued efforts of our community members who have invested their time in its creation, maintenance and improvement.

### Content updates

| PR number | Contributed by | Contribution made |
|---|---|---|
| PR #202 | network-charles | Updated instructions for building docs locally |
| PR #200 | Zetway-Technolog | Clarified instructions in install-isc-dhcp-server.md |
| PR #192 | minaelee | Updated CODA links in our contributing guide |
| PR #190 | kenyon | Removed no-longer-needed reference to a recently closed bug |
| PR #181 | davidekete | Deprecated the backup-with-shell-scripts.md page |
| PR #130 | Sophie-Pages | Deprecated the lxc-containers.md page |
| PR #119 | Sophie-Pages | Added introduction to install-openldap.md page |
| PR #115 | minaelee | Updated and refreshed content on lxd-containers.md page |
| PR #113 | teward | Updated apparmor.md to explain how to totally enable/disable AppArmor |
| PR #54 | khbecker | Added details about kdump enablement in 24.10 to kernel-crash-dump.md |

### Copyediting and proofreading

| PR number | Contributed by | Contribution made |
|---|---|---|
| PR #184 | thorn-shadow99 | Improved the sssd.md page |
| PR #183 | thorn-shadow99 | Improved the security.md page |
| PR #167 | POS-ToakDEV | Improved the libvirt.md page |
| PR #155 | thorn-shadow99 | Improved the multipath.md page |
| PR #149 | thorn-shadow99 | Improved the high-availability.md page |
| PR #148 | thorn-shadow99 | Improved the ebpf.md page |
| PR #147 | bird-stare | Improved the introduction-to-backups.md, introduction-to-mail-services.md and introduction-to-virtualisation.md pages |
| PR #146 | thorn-shadow99 | Improved the crypto-libraries.md page |
| PR #141 | gregbell | Improved the qemu-microvm.md page |
| PR #132 | bird-stare | Substantial copyediting on the networking.md page |
| PR #131 | network-cl | Improved the ebpf.md page |
| PR #123 | davideket | Converted “note” blocks into MyST admonitions (explanation section) |
| PR #122 | YanisaHS | Improved the install-dns.md page |
| PR #120 | davideket | Converted “note” blocks into MyST admonitions (reference section) |
| PR #83 | daveedoo | Improved the serve-ntp-with-chrony.md page |
| PR #81 | peat-psuw | Improved the report-a-bug.md page |
| PR #67 | pnhuy | Improved the configur-nginx.md page |

### Error fixes

| PR number | Contributed by | Contribution made |
|---|---|---|
| PR #97 | yukihane | Corrected an error in `install-openvpn.md` |
| PR #66 | Pexers | Removed duplicate text block in `peer-to-site-on-router.md` |
| PR #52 | sbworth | Corrected an error in WireGuard VPN `on-an-internal-system.md` |
| PR #48 | nikolas | Corrected an error in `install-postfix.md` |

### Functionality and enhancements

| PR number | Contributed by | Contribution made |
|---|---|---|
| PR #193 | network-charles | Added hover-over referencing functionality |
| PR #116 | activus-d | Created an issue template to help readers provide better feedback |
| PR #112 | goodylili | Created a pull request template |

### Glossary

| PR number | Contributed by | Contribution made |
|---|---|---|
| PR #194 | network-charles | Contributed terms to the glossary |
| PR #187 | network-charles | Contributed terms to the glossary |
| PR #186 | network-charles | Contributed terms to the glossary |
| PR #185 | network-charles | Contributed terms to the glossary |
| PR #182 | network-charles | Contributed terms to the glossary |
| PR #174 | network-charles | Contributed terms to the glossary |
| PR #162 | network-charles | Contributed terms to the glossary |
| PR #161 | kaushikjayant | Contributed terms to the glossary |
| PR #160 | network-charles | Contributed terms to the glossary |
| PR #159 | network-charles | Contributed terms to the glossary |
| PR #158 | network-charles | Contributed terms to the glossary |
| PR #156 | network-charles | Contributed terms to the glossary |
| PR #152 | Ibom99 | Contributed “J” terms to the glossary |
| PR #134 | RomainDusi | Contributed terms to the glossary |
| PR #124 | activus-d | Added the glossary page and functionality |

### PDF improvements

| PR number | Contributed by | Contribution made |
|---|---|---|
| PR #10 | SecondSkoll | Added the ability to build the PDF locally |
| PR #9 | SecondSkoll | Fixed our PDF build issues |