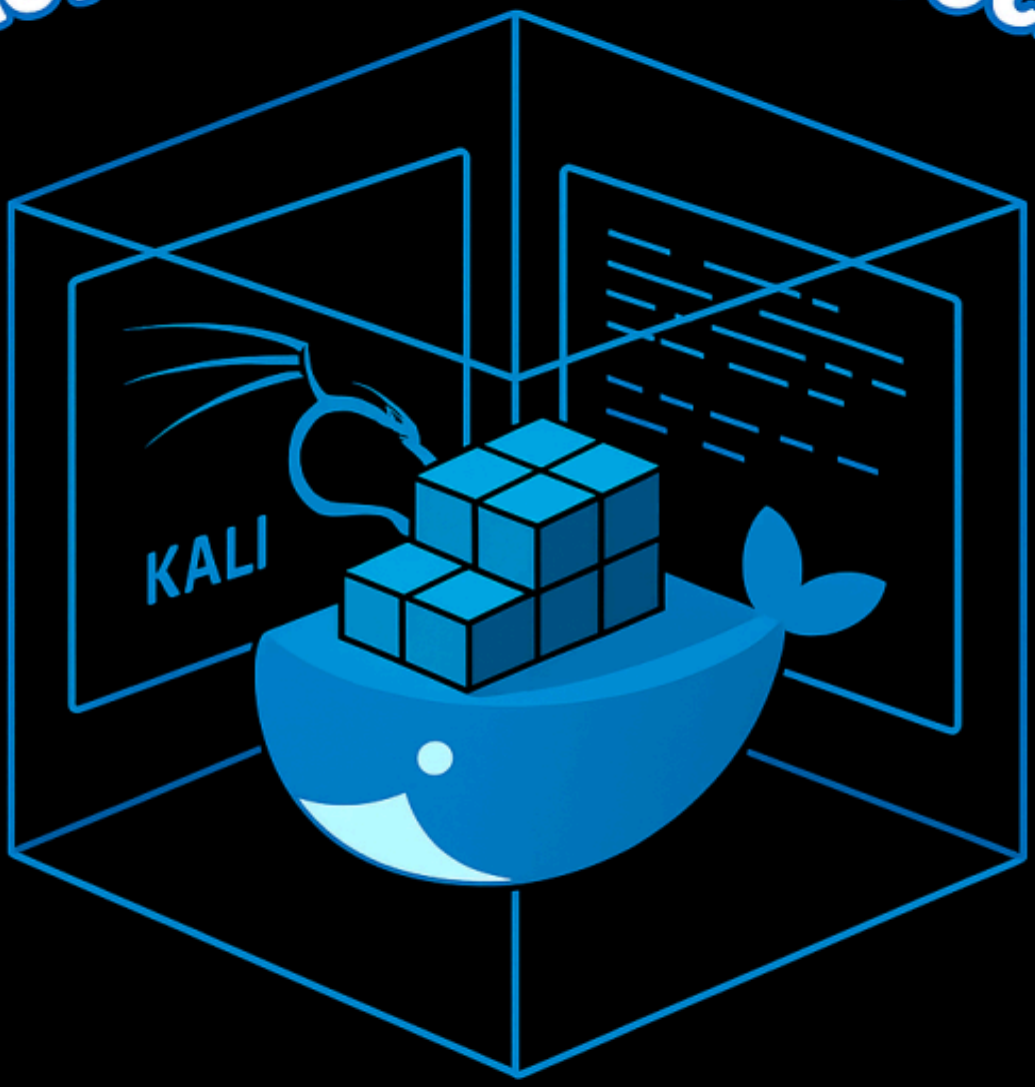
docker

PRÁCTICA 1: LEVANTAR DOCKER



CONTENEDORES DOCKER PARA PENTESTING

CONFIGURACIÓN DE LABORATORIO KALI LINUX EN WINDOWS

ILANA AMINOFF

ÍNDICE

INTRODUCCIÓN....3
DESCRIPCIÓN DE LA PRÁCTICA....4
OBJETIVOS....4
Objetivo General....4
Objetivos Específicos....4
MARCO TEÓRICO....5
Docker: Plataforma de Virtualización por Contenedores....5
Conceptos Fundamentales de Docker....5
Características y Ventajas de Docker....5
Usos y Aplicaciones de Docker....6
Kali Linux: Distribución Especializada en Ciberseguridad....6
– Características Técnicas de Kali Linux....7
– Herramientas y Categorías de Ciberseguridad....7
– Aplicaciones en Ciberseguridad Profesional....8
DESARROLLO DE LA PRÁCTICA....9
Laboratorio de Seguridad Informática con Tecnología de Contenedores....9
Fases del Desarrollo de la Práctica....9
Especificaciones del Sistema y Requisitos....11
Alcance y Entorno de la Práctica....14
Descripción del Entorno de Trabajo....14
– Plataforma de Contenedores:....15
– Entorno de Ciberseguridad:....15
– Objetivos Específicos del Entorno....15
FASE 0: Preparación del Entorno Docker en Windows....16
– Fase de Preparación : Instalaciones de Docker Desktop en Windows....16
– Fase de Descarga e Instalación de Docker Desktop....19
– Fase de Configuración Inicial de Docker Desktop....22
– Fase de Verificación esencial del entorno Docker....25
FASE 1: Implementación de Imagen Base....26
– Fundamento Teórico....26
– Procedimiento Detallado....26
FASE 2: Configuración de Red con Puertos Estratégicos....29
– Fundamento Teórico....29
– Procedimiento Detallado....29
FASE 3: Gestión Profesional del Contenedor....31
– Fundamento Teórico....31
– Procedimiento Detallado....31
FASE 4: Persistencia de Datos con Volúmenes Docker....34

– Fundamento Teórico....34
– Procedimiento Detallado....35
– Reflexión Final....43
Verificación de Persistencia desde el Sistema Host (Windows)....43
Sección Complementaria: Verificación Avanzada y Mantenimiento....45
1. Mapeo de Volúmenes Windows ↔ Kali....45
2. Análisis : Volúmenes Externos en Docker....45
3. Validación del Entorno Kali....45
3.1 Iniciar el Contenedor....46
3.2 Acceso al Contenedor....46
3.3 Actualización e Instalación de Herramientas Clave....46
3.4 Verificación de Versiones....47
3.5 Salir del Contenedor....48
4. Ejecución Temporal con Privilegios Elevados....48
4.1 Creación de Contenedor Temporal con Privilegios....49
4.2 Instalación de Herramientas y Verificación en el Contenedor con Privilegios 49
4.2.1 Prueba de escaneo con Nmap....50
4.3 Eliminación del Contenedor Temporal....51
5. Pruebas de Conectividad desde Windows....52
5.1 Verificación de Puertos Expuestos....52
5.2 Monitoreo de Recursos del Contenedor en Tiempo Real....53
6. Monitoreo y Administración del Contenedor....53
6.1 Comandos desde PowerShell....53
6.2 Interfaz Gráfica en Docker Desktop....54
6.3 Comandos adicionales de gestión del entorno....55
7. Limpieza del Entorno (Opcional)....57
7.1 Comandos en PowerShell:....57
7.2 Limpieza desde Docker Desktop (GUI):....58
CONCLUSIÓN....59
BIBLIOGRAFÍA....60

INTRODUCCIÓN

La virtualización mediante contenedores se ha convertido en una tecnología fundamental para la implementación de laboratorios de ciberseguridad modernos. Docker, como plataforma líder en contenedorización, ofrece un entorno aislado, reproducible y eficiente para el despliegue de herramientas de seguridad informática. Esta tecnología permite a los profesionales de la ciberseguridad crear entornos controlados donde pueden ejecutar análisis de vulnerabilidades, pruebas de penetración y análisis forense sin comprometer la infraestructura principal del sistema.

La importancia de Docker en el ámbito de la ciberseguridad radica en su capacidad para proporcionar aislamiento de procesos, gestión eficiente de recursos y portabilidad de aplicaciones. Los contenedores Docker permiten encapsular herramientas especializadas junto con sus dependencias, creando un entorno consistente que puede ser desplegado en cualquier sistema que ejecute Docker Engine. Esta característica resulta especialmente valiosa para profesionales que necesitan mantener múltiples versiones de herramientas de seguridad o trabajar con configuraciones específicas para diferentes tipos de pruebas.

En el contexto educativo, el uso de Docker para prácticas de ciberseguridad ofrece ventajas significativas tanto para instructores como para estudiantes. Los instructores pueden crear laboratorios estandarizados que se ejecutan de manera consistente en diferentes sistemas operativos, mientras que los estudiantes pueden acceder a herramientas profesionales sin necesidad de realizar instalaciones complejas o modificar sus sistemas principales. Esta aproximación pedagógica permite concentrarse en el aprendizaje de conceptos y técnicas de seguridad sin las distracciones técnicas asociadas con la configuración de entornos.

La presente práctica tiene como objetivo principal demostrar la implementación correcta de contenedores Docker aplicados específicamente al campo de la ciberseguridad. Se utilizará la distribución Kali Linux, reconocida mundialmente como la plataforma estándar para profesionales de la seguridad informática, que incluye más de 600 herramientas preinstaladas para pruebas de penetración, análisis forense y auditorías de seguridad. Esta elección no es casual, ya que Kali Linux representa el estándar de facto en la industria de la ciberseguridad para actividades de pentesting y análisis de seguridad.

A través de esta práctica, se desarrollarán competencias fundamentales en la gestión de contenedores, incluyendo la configuración de redes, gestión de volúmenes persistentes, mapeo de puertos para servicios de seguridad e implementación de mejores prácticas de seguridad en entornos contenedorizados. Estas habilidades son esenciales para cualquier profesional que desee trabajar en el campo de la ciberseguridad moderna, donde la contenedorización se ha convertido en una herramienta indispensable para la creación de laboratorios seguros y eficientes.

DESCRIPCIÓN DE LA PRÁCTICA

PRÁCTICA 1: LEVANTAR DOCKER

Para resolver este ejercicio práctico deberás seguir los siguientes pasos:

1. Levantar imagen Docker (a tu elección).
2. Levantar Docker abriendo los puertos.
3. Levantar Docker con nombre.
4. Levantar Docker con volumen.

- ❖ Se deberá entregar un documento pdf con los elementos mencionados acompañado de las capturas de imágenes requeridas.

OBJETIVOS

Objetivo General

Implementar y configurar contenedores Docker para crear un laboratorio de ciberseguridad funcional y seguro utilizando la distribución Kali Linux.

Objetivos Específicos

- Desplegar imágenes Docker especializadas en ciberseguridad
- Configurar mapeo de puertos para herramientas de seguridad
- Implementar nomenclatura profesional para gestión de contenedores
- Establecer persistencia de datos mediante volúmenes Docker
- Aplicar mejores prácticas de seguridad en entornos contenedorizados

---

MARCO TEÓRICO

Docker: Plataforma de Virtualización por Contenedores

Docker es una plataforma de código abierto diseñada para automatizar el despliegue, escalado y gestión de aplicaciones mediante el uso de contenedores de software. Desarrollado inicialmente por Solomon Hykes en 2013, Docker ha revolucionado la forma en que las aplicaciones se desarrollan, distribuyen y ejecutan en entornos de producción. A diferencia de la virtualización tradicional que requiere un hipervisor y sistemas operativos completos para cada máquina virtual, Docker utiliza virtualización a nivel de sistema operativo, compartiendo el kernel del host entre múltiples contenedores aislados.

Conceptos Fundamentales de Docker

◆ **Contenedores:** Son unidades de software ligeras y portátiles que encapsulan una aplicación junto con todas sus dependencias, librerías, configuraciones y archivos del sistema necesarios para su ejecución. Los contenedores garantizan que la aplicación funcione de manera consistente independientemente del entorno donde se despliegue, eliminando el problema clásico de "funciona en mi máquina". Cada contenedor es aislado del sistema host y de otros contenedores, proporcionando un entorno seguro y predecible.

◆ **Imágenes Docker:** Son plantillas de solo lectura que contienen el sistema de archivos y la configuración necesaria para crear contenedores. Las imágenes se construyen mediante capas incrementales, donde cada capa representa un cambio específico (instalación de software, configuración, etc.). Esta arquitectura en capas permite una reutilización eficiente del espacio de almacenamiento y facilita la distribución de imágenes a través de registros como Docker Hub.

◆ **Dockerfile:** Es un archivo de texto que contiene una serie de instrucciones para construir automáticamente una imagen Docker. Incluye comandos para especificar la imagen base, instalar software, copiar archivos, configurar variables de entorno y definir el comando de inicio del contenedor. Los Dockerfiles permiten la creación reproducible y versionada de entornos de aplicación.

Características y Ventajas de Docker

✔ **Portabilidad:** Los contenedores Docker pueden ejecutarse en cualquier sistema que soporte Docker Engine, independientemente del sistema operativo subyacente (Linux, Windows, macOS). Esta característica es fundamental para equipos de desarrollo que trabajan

en diferentes plataformas y para la migración entre entornos de desarrollo, pruebas y producción.

✔ **Eficiencia de Recursos:** Los contenedores comparten el kernel del sistema operativo host, requiriendo significativamente menos recursos que las máquinas virtuales tradicionales. Pueden iniciarse en segundos y consumir una fracción de la memoria y CPU comparado con la virtualización completa, permitiendo mayor densidad de aplicaciones por servidor físico.

✔ **Escalabilidad:** Docker facilita el escalado horizontal de aplicaciones mediante la creación rápida de múltiples instancias de contenedores. Herramientas como Docker Swarm y Kubernetes permiten la orquestación automática de contenedores, distribuyendo la carga de trabajo y gestionando la disponibilidad de servicios.

✔ **Aislamiento de Seguridad:** Cada contenedor opera en un espacio de nombres aislado con sus propios procesos, red y sistema de archivos. Docker utiliza tecnologías de seguridad del kernel Linux como cgroups, namespaces y capabilities para proporcionar aislamiento robusto entre contenedores y del sistema host.

Usos y Aplicaciones de Docker

◆ **Desarrollo y DevOps:** Docker estandariza entornos de desarrollo, eliminando inconsistencias entre diferentes máquinas de desarrolladores. Facilita la integración continua (CI) y el despliegue continuo (CD) mediante pipelines automatizados que construyen, prueban y despliegan aplicaciones de manera consistente.

◆ **Microservicios:** Docker es ideal para arquitecturas de microservicios, donde cada servicio se ejecuta en su propio contenedor. Esta aproximación facilita el desarrollo independiente, el escalado selectivo y el mantenimiento de componentes individuales de aplicaciones complejas.

◆ **Ambientes de Prueba:** Permite crear entornos de prueba idénticos a producción de manera rápida y económica. Los desarrolladores pueden probar aplicaciones en diferentes versiones de sistemas operativos y dependencias sin afectar su entorno de trabajo principal.

◆ **Computación en la Nube:** Los principales proveedores de servicios en la nube (AWS, Azure, Google Cloud) ofrecen servicios especializados para ejecutar contenedores Docker, facilitando el despliegue y escalado de aplicaciones en infraestructuras distribuidas.

Kali Linux: Distribución Especializada en Ciberseguridad

Kali Linux es una distribución de Linux basada en Debian, desarrollada y mantenida por Offensive Security Ltd. Lanzada oficialmente en 2013 como sucesor de BackTrack Linux,

Kali se ha establecido como el estándar de facto para profesionales de la ciberseguridad, investigadores de seguridad y entusiastas del hacking ético. Esta distribución ha sido meticulosamente diseñada para proporcionar un entorno completo y especializado para actividades relacionadas con pruebas de penetración, análisis forense digital, ingeniería inversa y auditorías de seguridad.

– Características Técnicas de Kali Linux

◆ **Arquitectura y Compatibilidad:** Kali Linux soporta múltiples arquitecturas de hardware incluyendo x86, x64, ARM y ARMEL, permitiendo su ejecución en una amplia variedad de dispositivos desde computadoras de escritorio hasta dispositivos embebidos como Raspberry Pi. Esta versatilidad es crucial para pruebas de seguridad en diferentes tipos de infraestructura tecnológica.

◆ **Rolling Release Model:** Kali utiliza un modelo de distribución continua (rolling release) basado en Debian Testing, lo que significa que las herramientas y el sistema se actualizan constantemente sin necesidad de reinstalaciones mayores. Este enfoque garantiza que los usuarios siempre tengan acceso a las versiones más recientes de herramientas de seguridad y parches de seguridad.

◆ **Repositorios Especializados:** La distribución mantiene repositorios específicos que contienen más de 600 herramientas de seguridad cuidadosamente seleccionadas, probadas y optimizadas para trabajar en conjunto. Estas herramientas se categorizan en áreas especializadas como reconocimiento, análisis de vulnerabilidades, explotación, post-explotación, forense digital e ingeniería inversa.

– Herramientas y Categorías de Ciberseguridad

✔ **Reconocimiento y Recopilación de Información:** Incluye herramientas como Nmap para descubrimiento de red, Maltego para análisis de relaciones, theHarvester para recopilación de información pública, y Recon-ng para automatización de reconocimiento. Estas herramientas son fundamentales para la fase inicial de cualquier evaluación de seguridad.

✔ **Análisis de Vulnerabilidades:** Contiene scanners especializados como OpenVAS, Nikto, SQLmap para inyecciones SQL, y Burp Suite Community para análisis de aplicaciones web. Estas herramientas permiten identificar debilidades de seguridad en sistemas, redes y aplicaciones.

✔ **Herramientas de Explotación:** Incluye frameworks como Metasploit para desarrollo y ejecución de exploits, Social Engineering Toolkit (SET) para ingeniería social, y BeEF para explotación de navegadores web. Estas herramientas permiten a los profesionales validar la criticidad de las vulnerabilidades identificadas.

✔ **Análisis Forense Digital:** Proporciona herramientas como Autopsy para análisis de sistemas de archivos, Volatility para análisis de memoria RAM, Wireshark para análisis de tráfico de red, y Sleuth Kit para investigaciones forenses. Estas herramientas son esenciales para investigaciones de incidentes de seguridad y análisis post-compromiso.

– Aplicaciones en Ciberseguridad Profesional

✔ **Pruebas de Penetración (Pentesting):** Kali Linux proporciona un entorno integrado para realizar evaluaciones completas de seguridad siguiendo metodologías estándar como OWASP, NIST y PTES. Los profesionales pueden ejecutar todas las fases de un pentest desde una única plataforma.

✔ **Análisis de Malware:** Incluye herramientas especializadas para análisis estático y dinámico de malware, permitiendo a los analistas de seguridad investigar muestras maliciosas en entornos controlados y seguros.

✔ **Investigación de Incidentes:** Proporciona capacidades forenses avanzadas para recopilar, preservar y analizar evidencia digital durante investigaciones de seguridad, cumpliendo con estándares legales y técnicos requeridos en entornos corporativos y legales.

✔ **Educación y Certificación:** Kali Linux es ampliamente utilizado en programas educativos de ciberseguridad y como plataforma de preparación para certificaciones profesionales como CEH (Certified Ethical Hacker), OSCP (Offensive Security Certified Professional) y CISSP.

DESARROLLO DE LA PRÁCTICA

Laboratorio de Seguridad Informática con Tecnología de Contenedores

Este laboratorio, basado en contenedores Docker, ha sido diseñado como una plataforma **versátil y profesional para la ciberseguridad**, permitiendo la ejecución tanto de **pruebas de penetración ofensivas** como de **análisis forense digital**, gracias a la integración de **volúmenes persistentes**, **herramientas especializadas** y una arquitectura modular **reutilizable y escalable**.

El desarrollo de esta práctica se estructura en **cuatro fases progresivas**, que evidencian la implementación completa de un entorno de laboratorio basado en contenedores Docker. Cada fase se apoya en los conocimientos y configuraciones de la anterior, construyendo un sistema cada vez más sofisticado y funcional para actividades de seguridad informática.

La metodología adoptada sigue un enfoque **hands-on**, combinando la **ejecución práctica de comandos** con el **análisis técnico** de los resultados obtenidos. Cada paso está documentado con **capturas de pantalla**, explicaciones detalladas de las configuraciones aplicadas y su relevancia en el contexto de la **ciberseguridad profesional**.

Fases del Desarrollo de la Práctica

✔ <u>FASE 0: Preparación del Entorno Docker en Windows</u>

◆ (Corresponde al requerimiento previo: instalación y configuración de Docker Desktop)
Antes de comenzar con el desarrollo de la práctica, es fundamental instalar y configurar correctamente Docker Desktop en Windows. Este proceso incluye:

- Habilitación de características de virtualización de Windows (WSL y Virtual Machine Platform)
- Instalación automatizada de Docker Desktop vía PowerShell
- Configuración inicial desde la interfaz gráfica (GUI)
- Validaciones críticas post-instalación (WSL, Hypervisor, Docker en ejecución)

Esta fase asegura que el entorno cumpla con todos los requisitos técnicos antes de levantar cualquier contenedor.

✔ FASE 1: Implementación de Imagen Base

◆ (Corresponde al requerimiento: "Levantar imagen Docker")
Esta fase aborda la descarga y verificación de la imagen oficial de Kali Linux, estableciendo los fundamentos del laboratorio. Introduce los conceptos de gestión de imágenes Docker y verificación de integridad, elementos cruciales para garantizar la autenticidad y seguridad de las herramientas utilizadas.

✔ FASE 2: Configuración de Red con Puertos Estratégicos

◆ (Corresponde al requerimiento: "Levantar Docker abriendo los puertos")
Se centra en el mapeo estratégico de puertos para herramientas específicas de ciberseguridad. Se configuran puertos dedicados para interfaces web de herramientas de pentesting, conexiones de reverse shell y servicios HTTPS seguros, creando una infraestructura de red controlada que simula escenarios reales de evaluación de seguridad.

✔ FASE 3: Gestión Profesional del Contenedor

◆ (Corresponde al requerimiento: "Levantar Docker con nombre")
Implementa nomenclatura empresarial y sistemas de metadatos para la organización eficiente de laboratorios complejos. Esta fase introduce conceptos de gestión de infraestructura a escala, fundamentales para entornos de producción donde múltiples equipos gestionan diversos proyectos de seguridad simultáneamente.

✔ FASE 4: Persistencia de Datos con Volúmenes Docker

◆ (Corresponde al requerimiento: "Levantar Docker con volumen")
Establece un sistema robusto de volúmenes persistentes que garantiza la preservación de reportes, evidencia forense y configuraciones personalizadas. Esta implementación refleja las mejores prácticas en gestión de información de seguridad, donde la trazabilidad y conservación de datos son requisitos críticos para cumplimiento regulatorio y continuidad operacional.

Cada fase incluye comandos específicos verificados, capturas de pantalla documentadas y análisis de resultados que permiten evaluar el éxito de la implementación. La progresión sistemática asegura que se desarrollen competencias tanto en aspectos técnicos de Docker como en aplicaciones especializadas de ciberseguridad, preparando el terreno para enfrentar desafíos reales en entornos profesionales de seguridad informática.

Especificaciones del Sistema y Requisitos

◆ Entorno de Desarrollo Objetivo

Esta práctica está diseñada para implementarse en un entorno Windows con las siguientes características específicas:

- **Sistema Operativo Base:** Windows 11 Pro (también compatible con Windows 10 Pro/Enterprise/Education versión 1903 o superior).
- **Plataforma de contenedores:** Docker Desktop para Windows con backend WSL2.
- **Distribución de Seguridad:** Kali Linux Rolling Release (imagen oficial de contenedor) Arquitectura: x64 (AMD64) con soporte para virtualización anidada.

◆ Requisitos Mínimos del Sistema

- ***Hardware:***
    - **Procesador:** Intel Core i5 4ta gen / AMD Ryzen 5 (con VT-x/AMD-V)
    - **Memoria RAM:** 8 GB (mínimo) | 16 GB (recomendado)
    - **Almacenamiento:** 25 GB libres (mínimo) | 40 GB (recomendado)
    - **Conectividad:** Conexión a Internet estable para descargas
    - **BIOS/UEFI:** Virtualización habilitada (Intel VT-x/AMD-V)
- ***Software:***
    - Windows 11 Pro/Enterprise o Windows 10 Pro/Enterprise (versión 1903+)
    - PowerShell 5.1 o superior (incluido en Windows)
    - Windows Terminal (recomendado, disponible en Microsoft Store)
    - Navegador web moderno (Chrome, Edge, Firefox)

◆ Características de Windows Requeridas

- Windows Subsystem for Linux (WSL2)
- Virtual Machine Platform
- Hyper-V (se configura automáticamente con Docker Desktop)
- Contenedores de Windows (opcional)

◆ Requisitos Recomendados para Rendimiento Óptimo

- ***Hardware Recomendado:***
    - **Procesador:** Intel Core i7 / AMD Ryzen 7 (8+ cores)
    - **Memoria RAM:** 32 GB (para múltiples contenedores simultáneos)
    - **Almacenamiento:** SSD NVMe con 100+ GB libres
    - **Conectividad:** Banda ancha 50+ Mbps
    - **Dispositivos:** Ratón con scroll wheel para Docker Desktop GUI

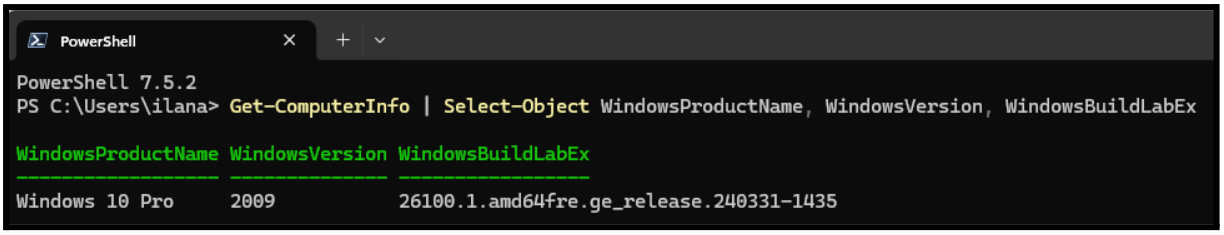
◆ Configuración de Red:

- **Firewall:** Windows Defender Firewall habilitado
- **Proxy:** Configuración de proxy corporativo (si aplica)
- **Puertos:** Acceso a puertos 8080, 4444, 8443 (para la práctica)
- **Antivirus:** Exclusiones para directorios Docker (si es necesario)

◆ Verificación Previa del Sistema

Antes de proceder con la instalación, ejecutar los siguientes comandos para verificar compatibilidad:

```
# Verificar versión de Windows
Get-ComputerInfo | Select-Object WindowsProductName,
WindowsVersion, WindowsBuildLabEx
```



Captura de pantalla : Verificación de la versión de Windows.

```
# Verificar memoria RAM total
Get-CimInstance -ClassName Win32_ComputerSystem | Select-Object
TotalPhysicalMemory
```

```
PS C:\Users\ilana> Get-CimInstance -ClassName Win32_ComputerSystem | Select-Object TotalPhysicalMemory

TotalPhysicalMemory
-------------------
        34282618880
```

Captura de pantalla : Verificación de la memoria RAM total.

```
# Verificar espacio en disco disponible
Get-CimInstance -ClassName Win32_LogicalDisk | Where-Object
{$_.DeviceID -eq "C:"} | Select-Object
@{Name="TotalSize(GB)";Expression={[math]::Round($_.Size/1GB,2)}},
@{Name="FreeSpace(GB)";Expression={[math]::Round($_.FreeSpace/1GB,
2)}}
```

```
PS C:\Users\ilana> Get-CimInstance -ClassName Win32_LogicalDisk | Where-Object {$_.DeviceID -eq "C:"} | Select-Object @{
Name="TotalSize(GB)";Expression={[math]::Round($_.Size/1GB,2)}}, @{Name="FreeSpace(GB)";Expression={[math]::Round($_.Fre
eSpace/1GB,2)}}

TotalSize(GB) FreeSpace(GB)
------------- -------------
       601,90         87,34
```

Captura de pantalla : Verificación del espacio en disco disponible.

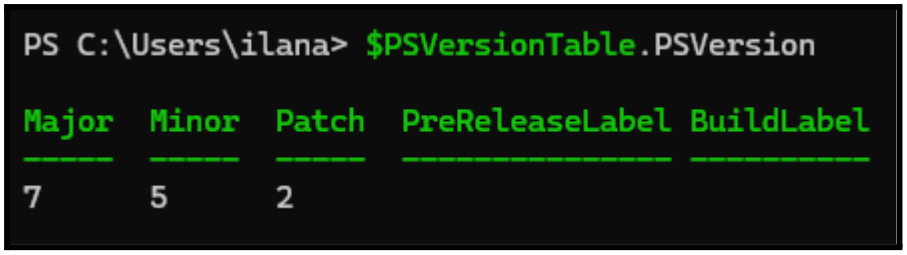
```
# Verificar soporte de virtualización
Get-ComputerInfo | Select-Object "HyperV*
```

```
PS C:\Users\ilana> Get-ComputerInfo | Select-Object "HyperV*"

HyperVisorPresent                                   : True
HyperVRequirementDataExecutionPreventionAvailable   :
HyperVRequirementSecondLevelAddressTranslation      :
HyperVRequirementVirtualizationFirmwareEnabled      :
HyperVRequirementVMMonitorModeExtensions            :
```

Captura de pantalla : Verificación del soporte de virtualización.

```
# Verificar versión de PowerShell
$PSVersionTable.PSVersion
```



Captura de pantalla : Verificación de la versión de PowerShell.

◆ Consideraciones Especiales para Entornos Corporativos

✔ Políticas de Seguridad:

- Verificar que las políticas corporativas permitan la instalación de Docker Desktop
- Confirmar permisos para habilitar características de Windows (WSL2, VM Platform)
- Revisar restricciones de firewall para puertos de contenedores

✔ Configuración de Proxy:

- Docker Desktop requiere configuración específica para entornos con proxy
- WSL2 puede requerir configuración adicional de proxy
- Considerar certificados corporativos para registros Docker privados

---

Alcance y Entorno de la Práctica

Descripción del Entorno de Trabajo

Esta práctica implementará un laboratorio completo de ciberseguridad utilizando la siguiente arquitectura tecnológica:

◆ Capa de Sistema Operativo Host:

Windows 11 Pro actuará como sistema operativo principal, proporcionando la plataforma base para todas las operaciones. Windows 11 ofrece mejoras significativas en seguridad, rendimiento y compatibilidad con tecnologías de virtualización modernas, incluyendo soporte nativo mejorado para WSL2 y contenedores.

◆ **Capa de Virtualización**

Windows Subsystem for Linux 2 (WSL2) funcionará como la capa de virtualización principal, proporcionando un kernel Linux real ejecutándose como una máquina virtual ligera dentro de Windows. Esta tecnología permite ejecutar contenedores Linux de manera nativa y eficiente, eliminando las limitaciones de rendimiento de versiones anteriores.

– Plataforma de Contenedores:

Docker Desktop para Windows servirá como la plataforma principal de gestión de contenedores, integrándose seamlessly con WSL2 para proporcionar una experiencia de contenedorización completa. Incluye Docker Engine, Docker CLI, Docker Compose y una interfaz gráfica intuitiva para gestión visual de contenedores, imágenes y redes.

– Entorno de Ciberseguridad:

Kali Linux Rolling Release se desplegará como contenedor Docker, proporcionando acceso a más de 600 herramientas especializadas de ciberseguridad. Esta implementación permite un entorno aislado y reproducible para actividades de pentesting, análisis forense y evaluación de seguridad.

– Objetivos Específicos del Entorno

- **Aislamiento y Seguridad:**

  El entorno contenedorizado garantiza aislamiento completo entre las herramientas de ciberseguridad y el sistema host Windows, permitiendo ejecutar software de pentesting sin riesgo para el sistema principal. Los contenedores proporcionan sandbox naturales que previenen modificaciones no deseadas del sistema host.

- **Portabilidad y Reproducibilidad:**

  La configuración basada en contenedores permite replicar exactamente el mismo entorno en diferentes sistemas Windows, facilitando la colaboración entre equipos y garantizando consistencia en resultados de pruebas de seguridad.

- **Escalabilidad y Gestión:**

  Docker Desktop permite gestionar múltiples contenedores simultáneamente,

facilitando la creación de laboratorios complejos con múltiples herramientas especializadas, targets de prueba y entornos de análisis.

- **Integración con Flujos de Trabajo Profesionales:**

  El entorno implementado refleja las mejores prácticas utilizadas en organizaciones profesionales de ciberseguridad, donde la contenedorización se ha convertido en estándar para despliegue de herramientas y gestión de laboratorios de seguridad.

FASE 0: Preparación del Entorno Docker en Windows

Antes de comenzar con el desarrollo de la práctica, es fundamental instalar y configurar correctamente Docker Desktop en Windows. Este proceso incluye la habilitación de características del sistema, instalación de WSL2, y configuración óptima de Docker para ciberseguridad.

– Fase de Preparación : Instalaciones de Docker Desktop en Windows

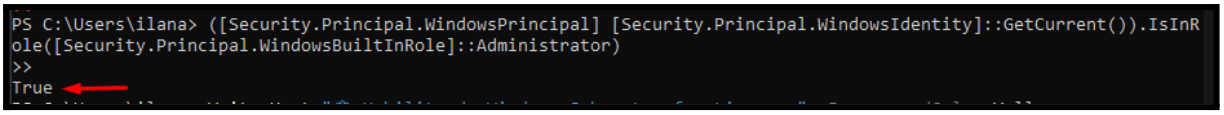
◆ Habilitación de Características de Windows

1. ***Abrir PowerShell como Administrador***

Antes de ejecutar cualquier comando, es imprescindible tener privilegios elevados. Para ello, se debe abrir **PowerShell** como **Administrador**.

2. ***Verificar que PowerShell se está ejecutando como Administrador:***

```
# Verificar si el proceso tiene permisos de administrador
if (-NOT ([Security.Principal.WindowsPrincipal]
[Security.Principal.WindowsIdentity]::GetCurrent()).IsInRole([Secu
rity.Principal.WindowsBuiltInRole]::Administrator)) {
    Write-Host "Este script requiere privilegios de administrador.
Reinicie PowerShell como administrador." -ForegroundColor Red
    exit
}
```



Captura de pantalla: *Verificación en PowerShell mostrando* ***True*** *si se ejecuta como administrador.*

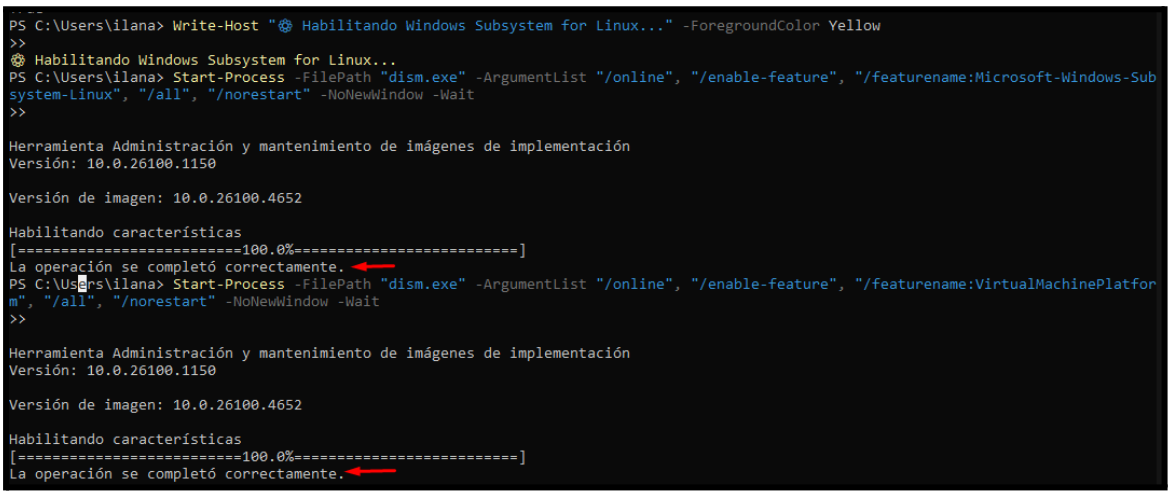
3. Habilitar características obligatorias para WSL y Docker

- **Habilitar Windows Subsystem for Linux (WSL):**

```
Write-Host "Habilitando Windows Subsystem for Linux (WSL)..."
-ForegroundColor Yellow
Start-Process -FilePath "dism.exe" -ArgumentList "/online",
"/enable-feature",
"/featurename:Microsoft-Windows-Subsystem-Linux", "/all",
"/norestart" -NoNewWindow -Wait
```

- **Habilitar Virtual Machine Platform (requisito para WSL 2 y Docker Desktop):**

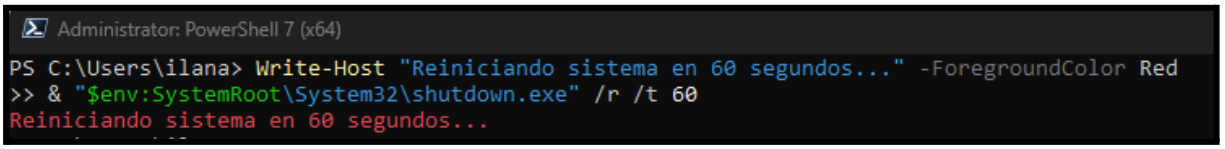
```
Write-Host "Habilitando Virtual Machine Platform..."
-ForegroundColor Yellow
Start-Process -FilePath "dism.exe" -ArgumentList "/online",
"/enable-feature", "/featurename:VirtualMachinePlatform", "/all",
"/norestart" -NoNewWindow -Wait
```



Captura de Pantalla: Mensajes de confirmación de activación de características (WSL + Virtual Machine Platform).

4. ***Reiniciar el sistema***

```
# Programar reinicio en 60 segundos
Write-Host "Reiniciando sistema en 60 segundos..."
-ForegroundColor Red
& "$env:SystemRoot\System32\shutdown.exe" /r /t 60
```



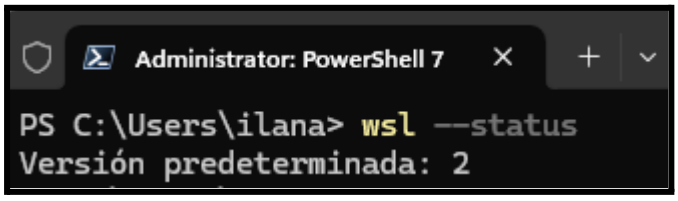
Captura de Pantalla : Mensaje de reinicio programado.

5. ***Verificación tras el reinicio***

Una vez que el sistema se ha reiniciado, es necesario confirmar que las características de virtualización y WSL se han aplicado correctamente.

- ***Verificar que WSL está instalado y configurado correctamente :*** Ejecutar en PowerShell (como Administrador):

```
powershell
wsl --status
```



Captura de pantalla: *Mostrar salida de* ***wsl --status*** *con "Versión predeterminada: 2".*

6. ***Nota Importante: Verificación de Requisitos Críticos***

⚠ **ANTES DE CONTINUAR**, es fundamental confirmar que se cumplan estos requisitos previos:

◆ **BIOS/UEFI:**

✔ **Virtualización habilitada** (Intel VT-x / AMD-V) en configuración de BIOS

✔ **VT-d habilitado** (si está disponible)

◆ **Características de Windows:**

✔ **Windows Subsystem for Linux** (habilitado)

✔ **Virtual Machine Platform** (habilitado)

✔ **Hyper-V** (se activa automáticamente con Docker Desktop)

```
#Verificación rápida:Confirmar que el hipervisor está presente
Get-ComputerInfo | Select-Object HypervisorPresent
# Resultado que nos debe mostrar:
HypervisorPresent : True
```

Si cualquiera de estos elementos no está correctamente configurado, Docker Desktop **no funcionará** o tendrá problemas de rendimiento. Con estos requisitos cumplidos, el sistema está **óptimamente preparado** para la contenedorización.

– Fase de Descarga e Instalación de Docker Desktop

Ejecutaremos en PowerShell como Administrador los siguientes comandos para descargar Docker Desktop.

1. ***Configurar variables de descarga***

```
# URL de descarga de Docker Desktop
$dockerUrl =
"https://desktop.docker.com/win/main/amd64/Docker%20Desktop%20Inst
aller.exe"
$dockerInstaller = "$env:TEMP\DockerDesktopInstaller.exe"
```

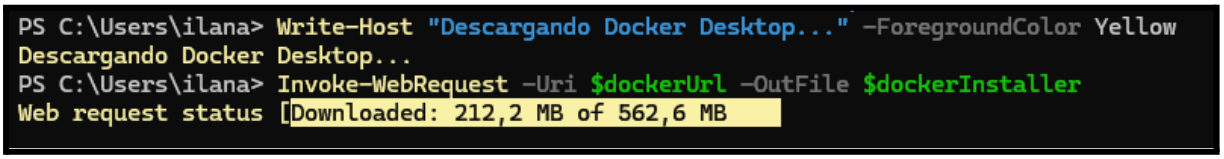


Captura de Pantalla : Configurando las variables de descarga en PowerShell

2. ***Descargar Docker Desktop***

```
Write-Host "Descargando Docker Desktop..." -ForegroundColor Yellow
Invoke-WebRequest -Uri $dockerUrl -OutFile $dockerInstaller
```

Nota: *Este proceso puede tardar entre 3-5 minutos dependiendo de la velocidad de conexión.*



Captura de Pantalla : Descargando Docker Desktop con progreso visible desde PowerShell

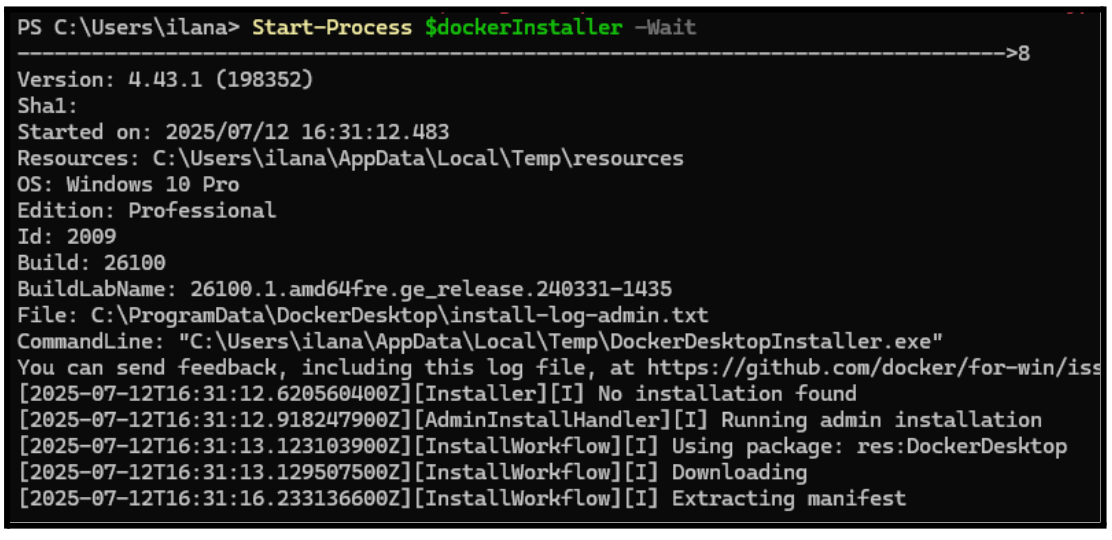
3. **Ejecutar instalador de Docker Desktop**

   ◆ **Configuración durante instalación:**

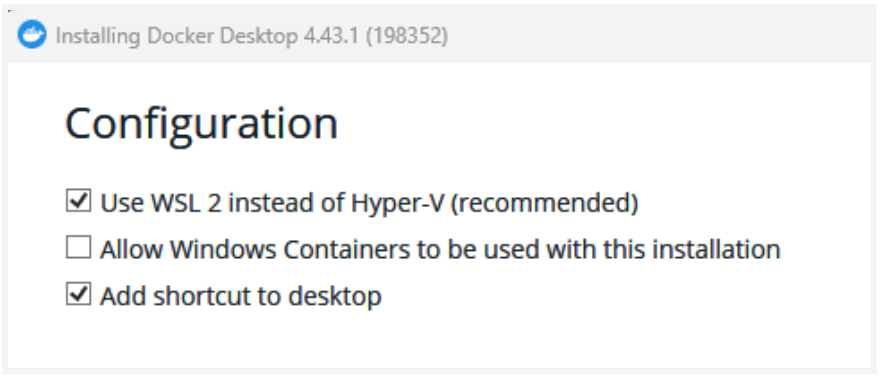
   ✔ Use WSL 2 instead of Hyper-V (CRÍTICO)

   ✔ Add shortcut to desktop

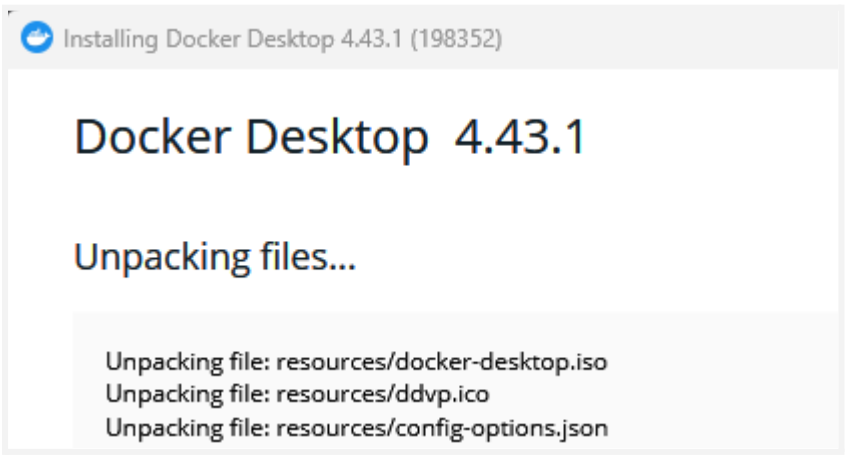
```
# Iniciar instalación interactiva
Start-Process $dockerInstaller -Wait
```



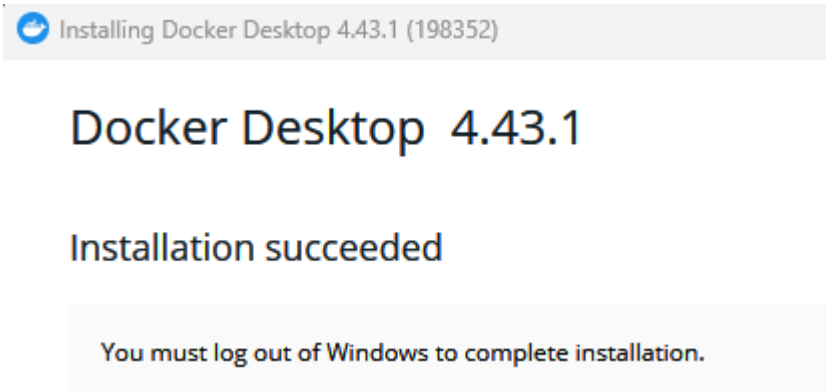
Captura de Pantalla : Instalando Docker Desktop con progreso visible desde PowerShell



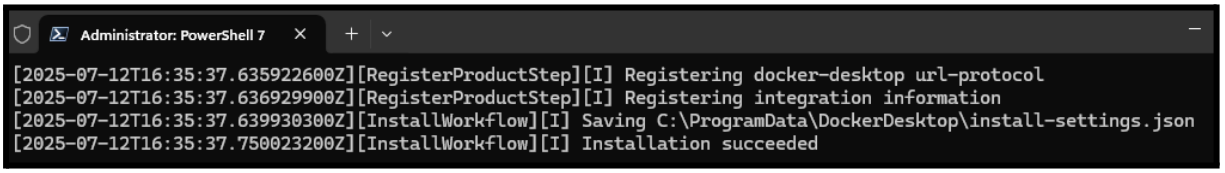
Captura de Pantalla : Configurando Instalación de Docker Desktop con progreso visible desde GUI



Captura de Pantalla : Instalando Docker Desktop con progreso visible desde GUI



Captura de Pantalla : Mensaje de Instalación de Docker Desktop finalizada desde GUI

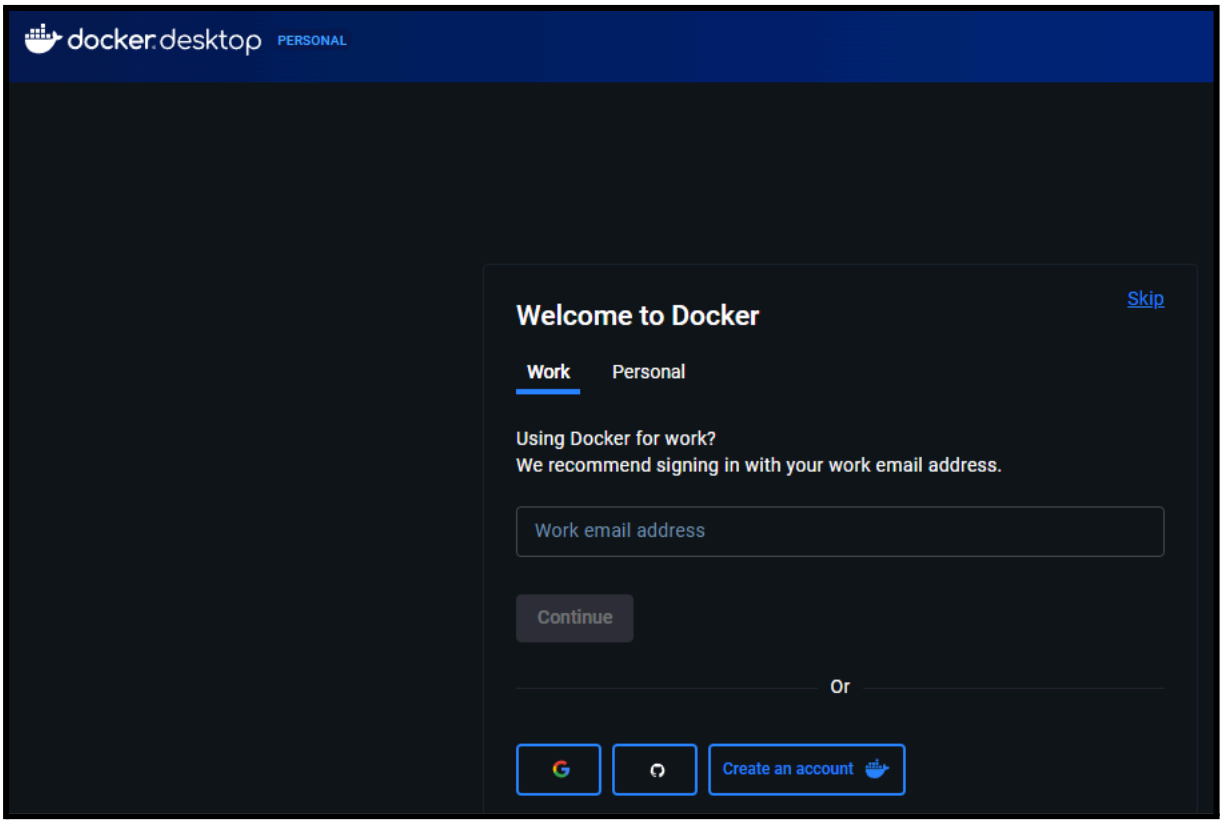


Captura de Pantalla : Mensaje de Instalación de Docker Desktop finalizada desde Powershell

– Fase de Configuración Inicial de Docker Desktop

1. ***Verificar que Docker Desktop está ejecutándose***

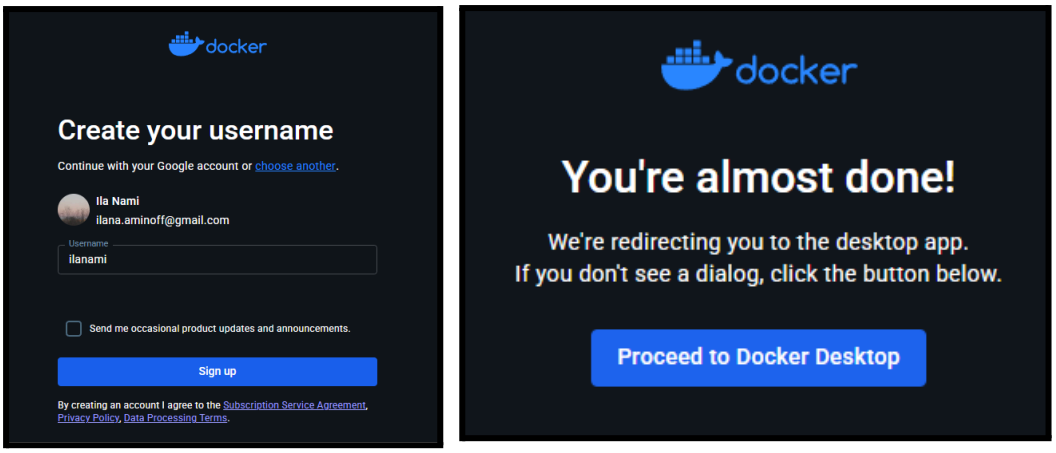
```
powershell
# Esperar a que Docker Desktop esté disponible y verificar proceso
Start-Sleep -Seconds 30
Get-Process "Docker Desktop" -ErrorAction SilentlyContinue
```



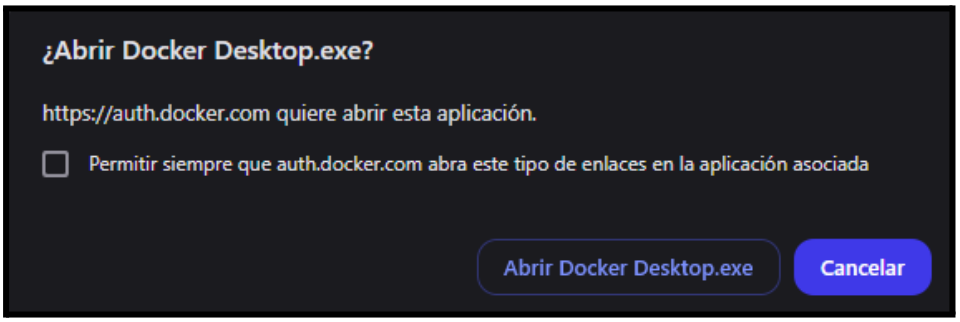
Captura de Pantalla : Mostrando Docker Desktop iniciándose

2. ***Configuración manual en Docker Desktop GUI:***

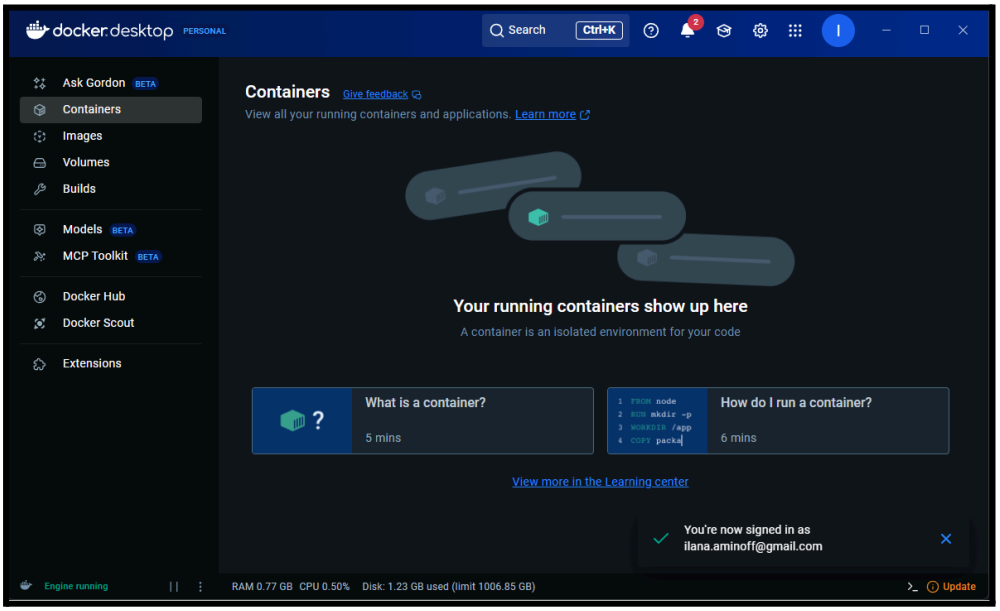
- ◆ Abrir Docker Desktop desde el escritorio
- ◆ Aceptar términos de servicio
- ◆ Saltar tutorial inicial o completarlo



Capturas de Pantalla : Configurando cuenta en docker con Gmail.



Capturas de Pantalla : Abriendo Docker Desktop después de configurar la cuenta de usuario con Gmail.

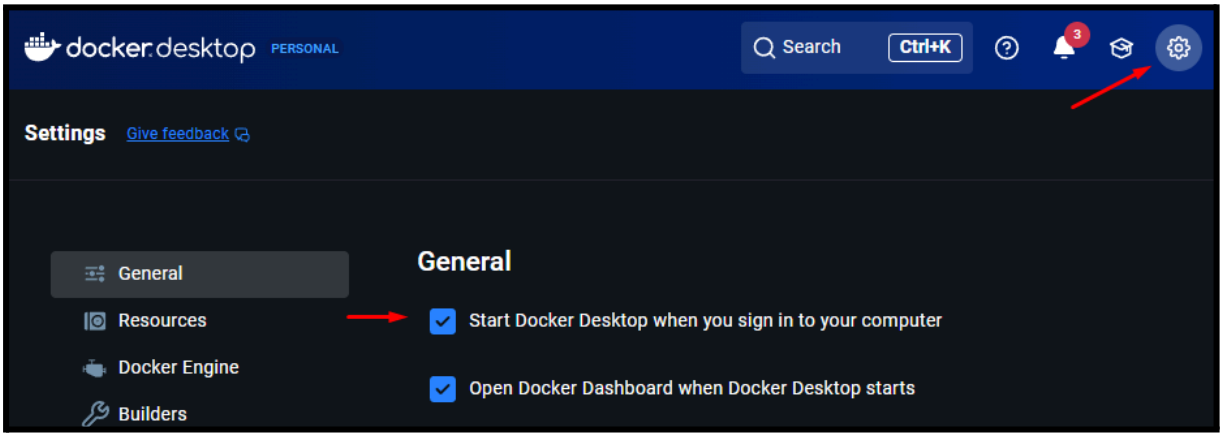


Captura de Pantalla : Docker Desktop ya ejecutándose, ventana de presentación.

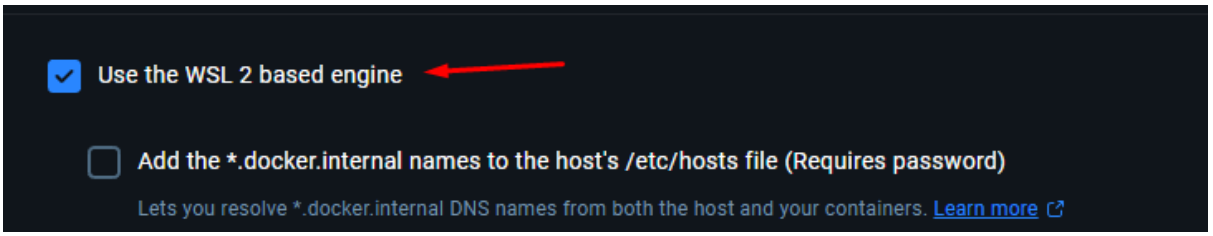
◆ **Settings → General:**

✔ Use the WSL 2 based engine

✔ Start Docker Desktop when you log in



Captura de Pantalla : Configurando opciones en la pestaña General de Docker Desktop.

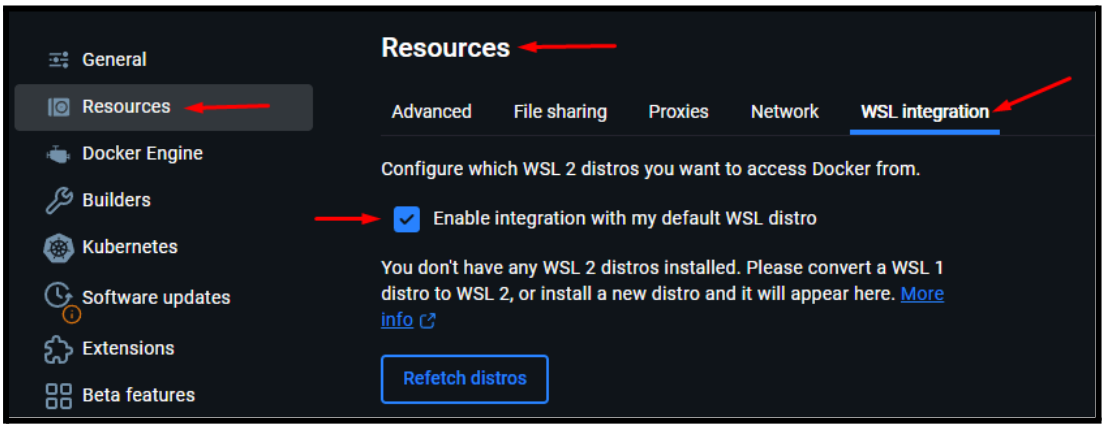


Captura de Pantalla : Habilitando la Configuración ***Use the WSL 2 based engine.***

◆ **Settings → Resources → WSL Integration:**

✔ Enable integration with my default WSL distro

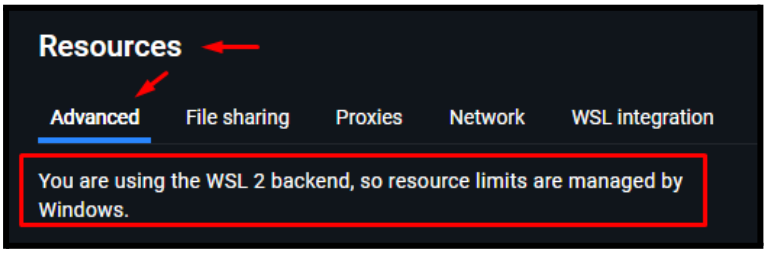
✔ Enable integration with additional distros (si están disponibles)



Captura de Pantalla : Habilitando la Configuración ***Enable integration with my default WSL distro***

◆ **Settings → Resources → Advanced:**

Con WSL2 Los recursos son gestionados automáticamente por Windows, lo podemos ver en el mensaje: ***"Resource limits are managed by Windows",*** *esto quiere decir que no requiere configuración manual de RAM/CPU/Disco*



Captura de Pantalla : Mensaje en opciones avanzadas de recursos donde se indica que Windows gestiona automáticamente los recursos del sistema.

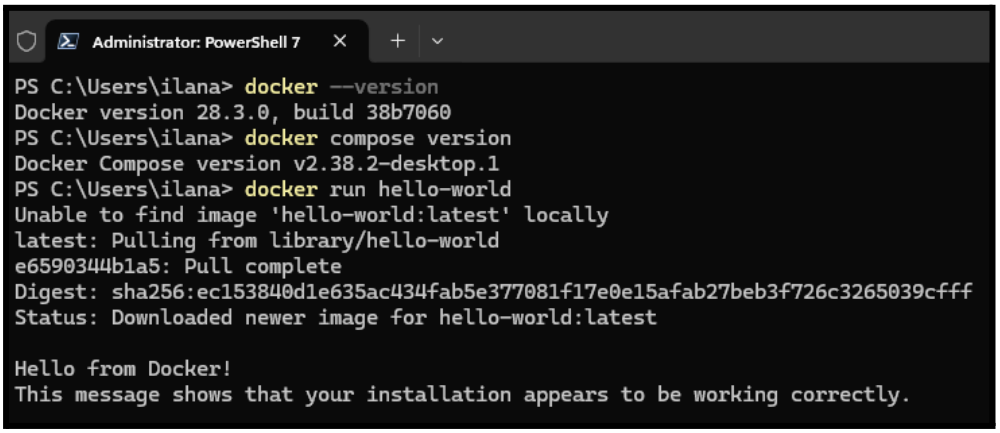
– Fase de Verificación esencial del entorno Docker

En esta fase se realizan unas verificaciones esenciales del entorno Docker, si se confirma que todo es correcto se puede avanzar a la implementación de la imagen de Kali Linux.

```
# Verificar versión de Docker
docker --version

# Verificar Docker Compose
docker compose version

# Test básico de funcionamiento
docker run hello-world
```



Captura de pantalla: Mostrando las verificaciones de Docker funcionando correctamente

◆ **Análisis de Resultados de la Instalación**

La verificación exitosa confirma que Docker Desktop con WSL2 está completamente operativo y listo para ejecutar contenedores de ciberseguridad. El test con **hello-world** garantiza que el motor Docker puede descargar, crear y ejecutar contenedores correctamente.

FASE 1: Implementación de Imagen Base

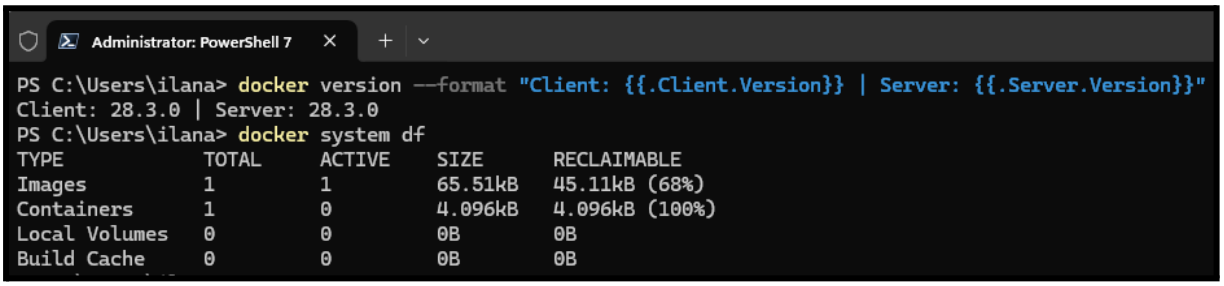
– Fundamento Teórico

Una vez completada la instalación y configuración de Docker Desktop en Windows, el primer paso del desarrollo práctico consiste en descargar e implementar la imagen oficial de Kali Linux. Esta imagen contiene una distribución completa de Kali Linux optimizada para contenedores, incluyendo el sistema base y los repositorios necesarios para instalar herramientas de ciberseguridad.

– Procedimiento Detallado

1. ***Verificar estado de Docker Desktop***

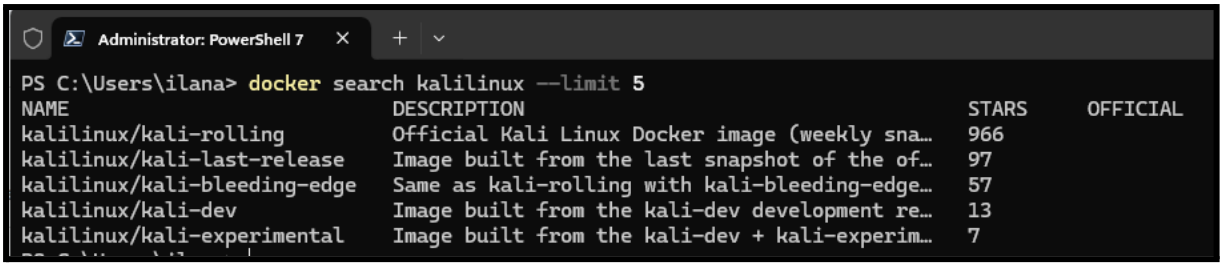
```
# Verificar conexión con Docker daemon
docker version --format "Client: {{.Client.Version}} | Server:
{{.Server.Version}}"
# Verificar recursos disponibles
docker system df
```



Captura de Pantalla de PowerShell mostrando Docker Desktop funcionando correctamente

2. ***Explorar repositorios Docker Hub para Kali Linux***

```
# Buscar imágenes oficiales de Kali Linux
docker search kalilinux --limit 5
```



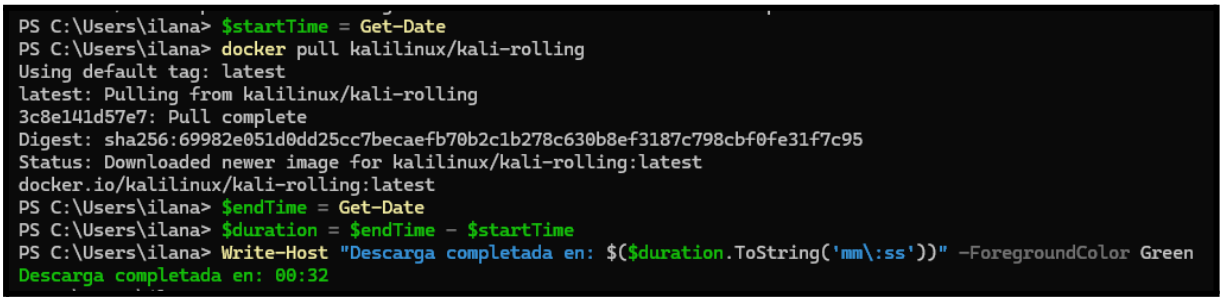
Captura de Pantalla de PowerShell mostrando la búsqueda de imágenes de Kali Linux en Docker Hub

3. *Descargar imagen oficial de Kali Linux*

```
# Registrar tiempo de inicio
$startTime = Get-Date

# Descargar imagen oficial
docker pull kalilinux/kali-rolling

# Calcular tiempo transcurrido
$endTime = Get-Date
$duration = $endTime - $startTime
Write-Host "Descarga completada en:
$($duration.ToString('mm\:ss'))" -ForegroundColor Green
```



Captura de Pantalla : Descarga de Kali Linux con el comando 'docker pull', ejecución y tiempo de descarga

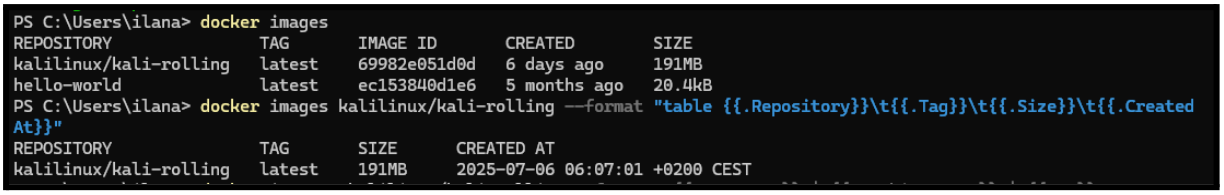
4. *Verificar imagen descargada y validar integridad*

```
# Listar todas las imágenes
docker images

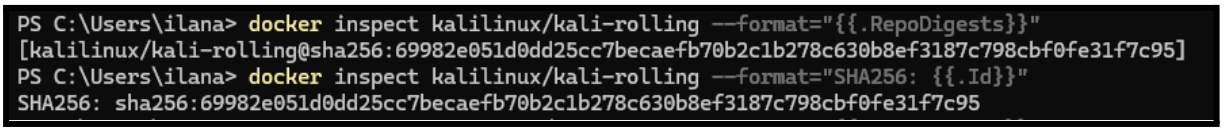
# Información específica de la imagen Kali
docker images kalilinux/kali-rolling --format "table
{{.Repository}}\t{{.Tag}}\t{{.Size}}\t{{.CreatedAt}}"
```

```
# Inspeccionar metadatos de la imagen
docker inspect kalilinux/kali-rolling --format="{{.RepoTags}} |
{{.Architecture}} | {{.Os}}"

# Verificación de integridad SHA256 (crítico para ciberseguridad)
docker inspect kalilinux/kali-rolling --format="{{.RepoDigests}}"
docker inspect kalilinux/kali-rolling --format="SHA256: {{.Id}}"
```



Captura de Pantalla de PowerShell mostrando información detallada de la imagen descargada.



Captura de Pantalla de PowerShell mostrando información de la verificación SHA256 de la imagen de Kali Linux

◆ Análisis de Resultados de la Descarga de Kali Linux en Docker

La descarga exitosa de la imagen oficial garantiza acceso a un entorno Kali Linux verificado y optimizado para contenedores. **La verificación SHA256 confirma la integridad y autenticidad de la imagen**, aspecto fundamental en ciberseguridad para evitar imágenes comprometidas o manipuladas. El tamaño comprimido de **191MB** demuestra la eficiencia de la imagen base optimizada para contenedores, que se expandirá cuando se instalen herramientas adicionales de ciberseguridad.

FASE 2: Configuración de Red con Puertos Estratégicos

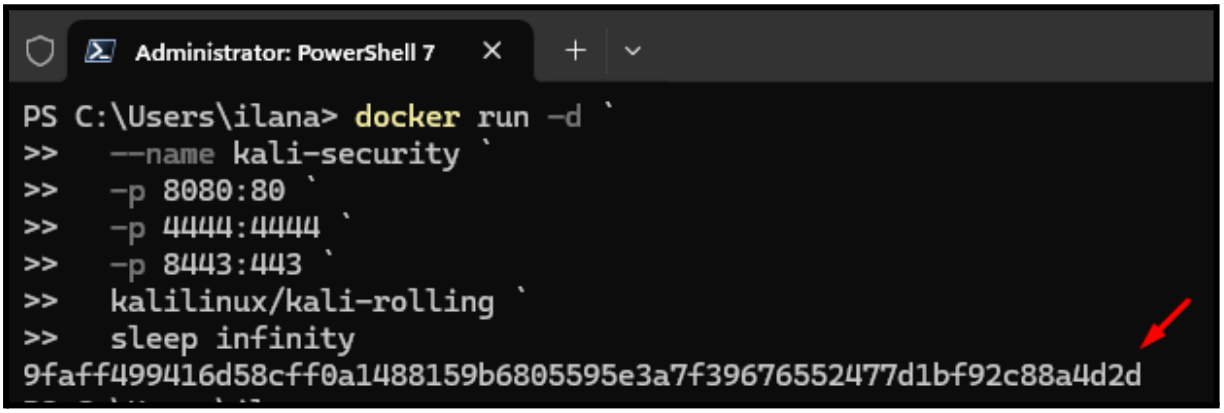
– Fundamento Teórico

El mapeo de puertos permite que los servicios ejecutándose dentro del contenedor sean accesibles desde el sistema host. Esto es fundamental para herramientas de ciberseguridad que requieren interfaces web o conexiones de red específicas.

– Procedimiento Detallado

1. ***Ejecutar contenedor con puertos mapeados***

```
# Ejecutar contenedor con mapeo de puertos específicos para
herramientas de seguridad
docker run -d `
  --name kali-security `
  -p 8080:80 `
  -p 4444:4444 `
  -p 8443:443 `
  kalilinux/kali-rolling `
  sleep infinity
```



Captura de Pantalla del comando docker run ejecutándose en PowerShell y creando un ID del Contenedor.

2. ***Verificar el contenedor en ejecución***

```
# Verificar que el contenedor está ejecutándose
docker ps

# Verificar los puertos mapeados
docker port kali-security
```



Captura de Pantalla mostrando el contenedor activo con 'docker ps' y puertos mapeados

◆ Configuración de Puertos Explicada

- **Puerto 8080 → 80**: Para interfaces web de herramientas (ej: Metasploit Web Interface)
- **Puerto 4444 → 4444**: Para conexiones Meterpreter y reverse shells
- **Puerto 8443 → 443**: Para servicios HTTPS seguros

◆ Análisis de Resultados

✔ Contenedor kali-security creado exitosamente

✔ Puertos 8080, 4444, 8443 mapeados correctamente

✔ Servicios accesibles desde localhost

✔ IPv4 e IPv6 configurados automáticamente

✔ Estado: Up 3 minutes - funcionando estable

◆ Observaciones técnicas:

- Tiempo de creación: ~3 minutos
- Mapeo automático IPv4/IPv6
- Docker Desktop maneja NAT transparentemente

La configuración exitosa de puertos permite que las herramientas de ciberseguridad dentro del contenedor sean accesibles desde el sistema host, manteniendo el aislamiento de seguridad.

FASE 3: Gestión Profesional del Contenedor

– Fundamento Teórico

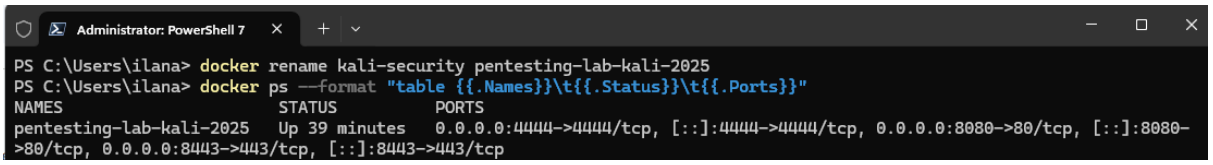
Asignar nombres descriptivos a los contenedores es una práctica fundamental para la gestión eficiente de infraestructuras Docker. Los nombres permiten identificar rápidamente el propósito y contenido de cada contenedor.

– Procedimiento Detallado

1. Evolucionar contenedor existente con nomenclatura profesional

```
# Renombrar contenedor existente con nomenclatura profesional
docker rename kali-security pentesting-lab-kali-2025

# Verificar el cambio de nombre exitoso
docker ps --format "table {{.Names}}\t{{.Status}}\t{{.Ports}}"
```



Captura de pantalla mostrando el contenedor renombrado exitosamente

⚠ Aclaración Importante

◆ *¿Por qué no creamos un contenedor nuevo?*

- En el anterior punto ya creamos un contenedor funcional con nombre `**kali-security**`.
- En esta sección se demuestra evolución continua de infraestructura (mejor práctica profesional).
- Evitamos recrear recursos innecesariamente.
- Mantenemos continuidad operacional.

◆ ***Flujo de Nomenclatura:***

- **`--name kali-security`** (nombre básico funcional)
- **`docker rename`** → **`pentesting-lab-kali-2025`** (nomenclatura profesional)

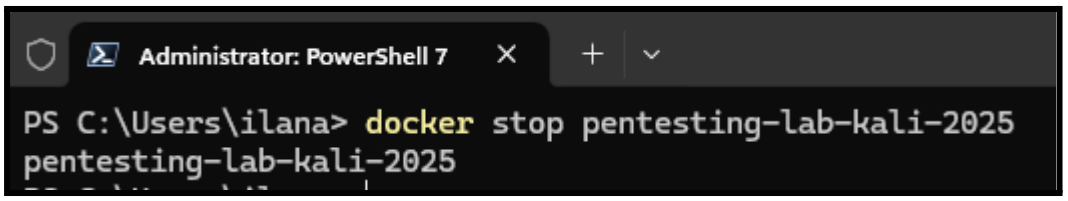
2. Agregar metadatos profesionales al contenedor

⚠ **Consideración Técnica Importante**

Limitación de Docker: *Los metadatos sólo pueden establecerse durante la creación del contenedor, no mediante modificación posterior. Para incorporar metadatos profesionales al contenedor existente, es necesario recrearlo con la configuración completa.*

◆ **Procedimiento de Recreación con Metadatos:**

```
# 1. Detener contenedor actual
docker stop pentesting-lab-kali-2025
```



Captura de pantalla mostrando que se ha detenido el contenedor.

```
# 2. Crear contenedor con metadatos profesionales
docker run -d `
  --name pentesting-lab-kali-2025-new `
  -p 8080:80 `
  -p 4444:4444 `
  -p 8443:443 `
  --label "project=cybersecurity" `
  --label "environment=lab" `
  --label "tools=kali-linux" `
  --label "purpose=penetration-testing" `
  --label "platform=windows-docker-desktop" `
  --label "created-by=ilana-aminoff" `
  kalilinux/kali-rolling `
  tail -f /dev/null
```

```
PS C:\Users\ilana> docker run -d `
>>   --name pentesting-lab-kali-2025-new `
>>   -p 8080:80 `
>>   -p 4444:4444 `
>>   -p 8443:443 `
>>   --label "project=cybersecurity" `
>>   --label "environment=lab" `
>>   --label "tools=kali-linux" `
>>   --label "purpose=penetration-testing" `
>>   --label "platform=windows-docker-desktop" `
>>   --label "created-by=ilana-aminoff" `
>>   kalilinux/kali-rolling `
>>   tail -f /dev/null
821ca7f135191ac4b3ce8ee0b9c076391bb5559a0ba6680127429cc0c35ff005
```

Captura de pantalla mostrando los comandos de actualización de metadatos.

```
# 3. Verificar funcionamiento correcto
docker ps --format "table {{.Names}}\t{{.Status}}\t{{.Ports}}"
```

```
PS C:\Users\ilana> docker ps --format "table {{.Names}}\t{{.Status}}\t{{.Ports}}"
NAMES                        STATUS         PORTS
pentesting-lab-kali-2025-new   Up 2 minutes   0.0.0.0:4444->4444/tcp, [::]:4444->4444/tcp, 0.0.0.0:8080->80/tcp, [::]:80
80->80/tcp, 0.0.0.0:8443->443/tcp, [::]:8443->443/tcp
```

Captura de pantalla mostrando el correcto funcionamiento del contenedor.

```
# 4. Eliminar contenedor anterior y renombrar
docker rm pentesting-lab-kali-2025
docker rename pentesting-lab-kali-2025-new
pentesting-lab-kali-2025
```

```
PS C:\Users\ilana> docker rm pentesting-lab-kali-2025
pentesting-lab-kali-2025
PS C:\Users\ilana> docker rename pentesting-lab-kali-2025-new pentesting-lab-kali-2025
```

Captura de pantalla mostrando la eliminación y renombrando el contenedor.

```
# 5. Inspeccionar metadatos agregados
docker inspect pentesting-lab-kali-2025 --format='{{range $k, $v
:= .Config.Labels}}{{printf "%s: %s`n" $k $v}}{{end}}'
```



Captura de pantalla mostrando el contenedor evolucionado con metadatos

◆ ***Convenciones de Nomenclatura Aplicadas***

- **Prefijo descriptivo**: "pentesting-lab" indica el propósito específico
- **Tecnología**: "kali" especifica la distribución utilizada
- **Año**: "2025" para control de versiones y seguimiento temporal
- **Evolución continua**: Sin interrumpir servicios activos

◆ ***Análisis de Resultados***

✔ Contenedor del PASO 2 evolucionado exitosamente

✔ Nomenclatura profesional aplicada sin pérdida de configuración

✔ Metadatos organizacionales agregados correctamente

✔ Continuidad operacional mantenida (puertos y servicios activos)

✔ Etiqueta adicional de plataforma para mejor organización

La evolución del contenedor existente demuestra flexibilidad de Docker para modificar contenedores en tiempo real, manteniendo continuidad operacional mientras se implementan mejores prácticas de nomenclatura y metadatos organizacionales.

FASE 4: Persistencia de Datos con Volúmenes Docker

– Fundamento Teórico

Los **volúmenes en Docker** permiten la persistencia de datos más allá del ciclo de vida de un contenedor. En contextos de **ciberseguridad**, esta característica es vital para:

- Mantener reportes de pruebas de penetración.

- Conservar evidencias forenses recolectadas.
- Guardar configuraciones personalizadas para reutilización.

A diferencia del sistema de archivos interno de un contenedor, que desaparece al eliminarlo, los volúmenes garantizan que los datos se mantengan accesibles en el sistema host.

– Procedimiento Detallado

1. ***Crear estructura de directorios en Windows***

```
# Crear ruta base para los volúmenes
$dockerVolumePath = "$env:USERPROFILE\docker-volumes\kali-lab"
```



Captura de pantalla mostrando la definición automática de la ruta base para volúmenes persistentes con $env:USERPROFILE en PowerShell.

```
# Crear subdirectorios para diferentes tipos de datos
New-Item -ItemType Directory -Path "$dockerVolumePath\tools"
-Force
New-Item -ItemType Directory -Path "$dockerVolumePath\reports"
-Force
New-Item -ItemType Directory -Path "$dockerVolumePath\evidence"
-Force
New-Item -ItemType Directory -Path "$dockerVolumePath\configs"
-Force
```

```
PS C:\Users\ilana> New-Item -ItemType Directory -Path "$dockerVolumePath\tools" -Force

    Directory: C:\Users\ilana\docker-volumes\kali-lab

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----          12/07/2025    23:17                tools

PS C:\Users\ilana> New-Item -ItemType Directory -Path "$dockerVolumePath\reports" -Force

    Directory: C:\Users\ilana\docker-volumes\kali-lab

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----          12/07/2025    23:17                reports

PS C:\Users\ilana> New-Item -ItemType Directory -Path "$dockerVolumePath\evidence" -Force

    Directory: C:\Users\ilana\docker-volumes\kali-lab

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----          12/07/2025    23:17                evidence

PS C:\Users\ilana> New-Item -ItemType Directory -Path "$dockerVolumePath\configs" -Force

    Directory: C:\Users\ilana\docker-volumes\kali-lab

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----          12/07/2025    23:17                configs
```

Captura de pantalla mostrando la creación de los subdirectorios tools, reports, evidence y configs dentro del volumen persistente kali-lab en PowerShell.

```
# Verificar la estructura creada
Get-ChildItem -Path $dockerVolumePath -Recurse | Format-Table Name, FullName
```

```
PS C:\Users\ilana> Get-ChildItem -Path $dockerVolumePath -Recurse | Format-Table Name, FullName

Name     FullName
----     --------
configs  C:\Users\ilana\docker-volumes\kali-lab\configs
evidence C:\Users\ilana\docker-volumes\kali-lab\evidence
reports  C:\Users\ilana\docker-volumes\kali-lab\reports
tools    C:\Users\ilana\docker-volumes\kali-lab\tools
```

Captura de pantalla mostrando la verificación de la estructura de directorios persistentes creada para el laboratorio Kali, utilizando Get-ChildItem en PowerShell.

2. Evolucionar el contenedor existente con volúmenes

En esta etapa, se procede a **reemplazar el contenedor inicial pentesting-lab-kali-2025** por una nueva instancia mejorada denominada **kali-forensic-lab**, que incorpora volúmenes

montados desde el sistema host. Esta acción **no responde a un fallo**, sino a una decisión arquitectónica fundamentada en buenas prácticas profesionales.

> ⚠ **Importante:** Docker **no permite añadir volúmenes a un contenedor ya creado**. La única forma de incorporar esta funcionalidad es generando un nuevo contenedor desde cero con los volúmenes definidos desde el comando docker run.

Esta evolución está motivada por los siguientes criterios técnicos y estratégicos:

✔ ***Criterios técnicos y de ciberseguridad***

- ◆ **Inmutabilidad del contenedor:** Docker promueve contenedores efímeros y reemplazables. Los cambios estructurales, como el mapeo de volúmenes, sólo se aplican en el momento de su creación.
- ◆ **Persistencia de datos:** En un laboratorio de pentesting o análisis forense, es crucial que los datos generados (reportes, evidencias, configuraciones) **no desaparezcan al eliminar el contenedor**.
- ◆ **Separación entre sistema y datos:** Al mantener los datos forenses fuera del sistema de archivos del contenedor, se evita su pérdida por errores humanos, reinicios o reconstrucción del entorno.
- ◆ **Gestión modular y trazabilidad:** Al versionar el entorno con un nuevo nombre **kali-forensic-lab**, se mejora la trazabilidad del laboratorio, se documenta su evolución, y se mantiene una estructura profesional alineada con principios de DevSecOps.

◆ ***Procedimientos paso a paso***

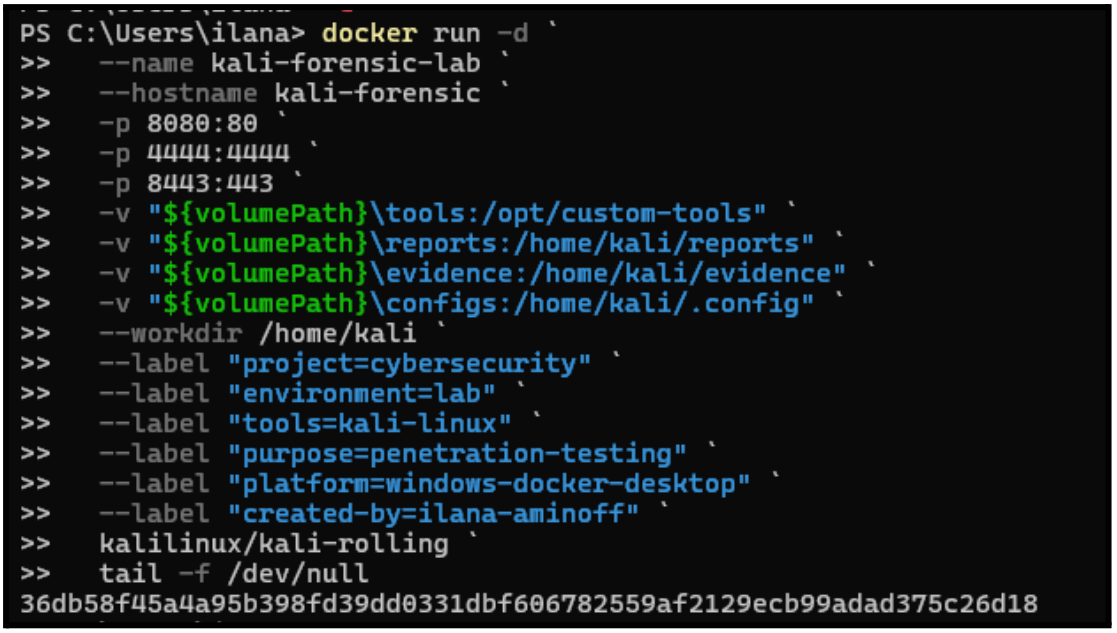
```
# Detener el contenedor anterior (si está corriendo)
docker stop pentesting-lab-kali-2025
```

```
# Definir la ruta base de volúmenes
$volumePath = "$env:USERPROFILE\docker-volumes\kali-lab"
```

```
PS C:\Users\ilana> docker stop pentesting-lab-kali-2025
pentesting-lab-kali-2025
PS C:\Users\ilana> $volumePath = "$env:USERPROFILE\docker-volumes\kali-lab"
```

Captura de pantalla mostrando la detención del contenedor original y definición de ruta de volúmenes para evolución

```
# Crear el nuevo contenedor con mapeo de volúmenes
docker run -d `
  --name kali-forensic-lab `
  --hostname kali-forensic `
  -p 8080:80 `
  -p 4444:4444 `
  -p 8443:443 `
  -v "${volumePath}\tools:/opt/custom-tools" `
  -v "${volumePath}\reports:/home/kali/reports" `
  -v "${volumePath}\evidence:/home/kali/evidence" `
  -v "${volumePath}\configs:/home/kali/.config" `
  --workdir /home/kali `
  --label "project=cybersecurity" `
  --label "environment=lab" `
  --label "tools=kali-linux" `
  --label "purpose=penetration-testing" `
  --label "platform=windows-docker-desktop" `
  --label "created-by=ilana-aminoff" `
  kalilinux/kali-rolling `
  tail -f /dev/null
```



Captura de pantalla mostrando la creación del contenedor kali-forensic-lab con volúmenes persistentes y etiquetas personalizadas

```
# Eliminar contenedor anterior
docker rm pentesting-lab-kali-2025
```

```
PS C:\Users\ilana> docker rm pentesting-lab-kali-2025
pentesting-lab-kali-2025
```

Captura de pantalla mostrando la eliminación controlada del contenedor pentesting-lab-kali-2025 tras migrar a una versión mejorada con volúmenes persistentes.

3. Verificar volúmenes montados

```
# Verificar que el contenedor está ejecutándose
docker ps
```

```
PS C:\Users\ilana> docker ps
CONTAINER ID   IMAGE                  COMMAND              CREATED         STATUS         PORTS

      NAMES
36db58f45a4a   kalilinux/kali-rolling   "tail -f /dev/null"   4 minutes ago   Up 4 minutes   0.0.0.0:4444->4
444/tcp, [::]:4444->4444/tcp, 0.0.0.0:8080->80/tcp, [::]:8080->80/tcp, 0.0.0.0:8443->443/tcp, [::]:8443->443
/tcp   kali-forensic-lab
```

Captura de pantalla mostrando el contenedor kali-forensic-lab en ejecución, con puertos expuestos y configuración activa mediante docker ps.

```
# Inspeccionar los volúmenes montados
docker inspect kali-forensic-lab --format='{{range
.Mounts}}{{.Source}} -> {{.Destination}}`n{{end}}'
```

```
PS C:\Users\ilana> docker inspect kali-forensic-lab --format='{{range .Mounts}}{{.Source}} -> {{.Destination
}}`n{{end}}'
C:\Users\ilana\docker-volumes\kali-lab\reports -> /home/kali/reports`nC:\Users\ilana\docker-volumes\kali-lab
\evidence -> /home/kali/evidence`nC:\Users\ilana\docker-volumes\kali-lab\configs -> /home/kali/.config`nC:\U
sers\ilana\docker-volumes\kali-lab\tools -> /opt/custom-tools`n
```

Captura de pantalla verificando la correcta asignación de volúmenes en el contenedor kali-forensic-lab, mediante el comando docker inspect, que muestra la correspondencia entre las rutas del host y los puntos de montaje dentro del contenedor.

4. Probar persistencia de datos desde el contenedor

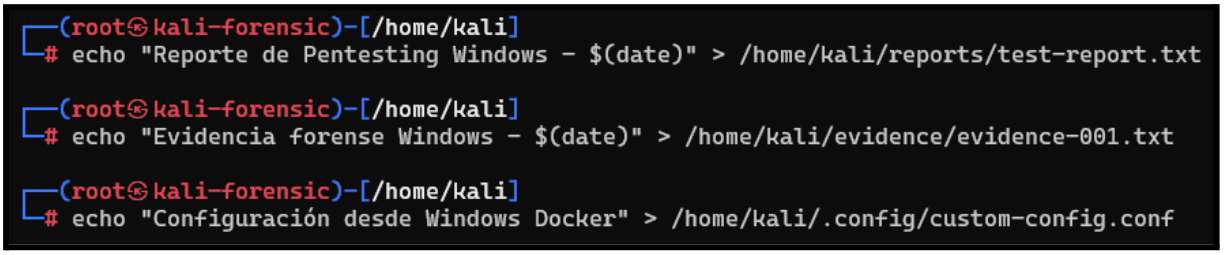
El objetivo de este punto es validar que los volúmenes montados permiten almacenar datos persistentes desde el contenedor kali-forensic-lab.

```
# Acceder al contenedor
docker exec -it kali-forensic-lab /bin/bash
```



Captura de pantalla mostrando el acceso interactivo al contenedor kali-forensic-lab mediante el comando docker exec -it, confirmando que se ha ingresado correctamente al entorno de trabajo de Kali Linux dentro del contenedor.

```
# Crear archivos de prueba
echo "Reporte de Pentesting Windows - $(date)" >
/home/kali/reports/test-report.txt
echo "Evidencia forense Windows - $(date)" >
/home/kali/evidence/evidence-001.txt
echo "Configuración desde Windows Docker" >
/home/kali/.config/custom-config.conf
```



✔ Verificar archivos creados y su contenido:

```
# Listar archivos en los volúmenes montados
ls -l /home/kali/reports
ls -l /home/kali/evidence
ls -l /home/kali/.config
```

```
┌──(root㉿kali-forensic)-[/home/kali]
└─# ls -l /home/kali/reports
total 0
-rw-r--r-- 1 root root 61 Jul 12 22:23 test-report.txt

┌──(root㉿kali-forensic)-[/home/kali]
└─# ls -l /home/kali/evidence
total 0
-rw-r--r-- 1 root root 57 Jul 12 22:24 evidence-001.txt

┌──(root㉿kali-forensic)-[/home/kali]
└─# ls -l /home/kali/.config
total 0
-rw-r--r-- 1 root root 36 Jul 12 22:24 custom-config.conf
```

Captura de Pantalla : Verificación de Archivos Persistentes en Volúmenes Docker con ***ls -l***

```
# Visualizar el contenido de los archivos creados
cat /home/kali/reports/test-report.txt
cat /home/kali/evidence/evidence-001.txt
cat /home/kali/.config/custom-config.conf
```

```
┌──(root㉿kali-forensic)-[/home/kali]
└─# cat /home/kali/reports/test-report.txt
Reporte de Pentesting Windows - Sat Jul 12 22:23:57 UTC 2025

┌──(root㉿kali-forensic)-[/home/kali]
└─# cat /home/kali/evidence/evidence-001.txt
Evidencia forense Windows - Sat Jul 12 22:24:07 UTC 2025

┌──(root㉿kali-forensic)-[/home/kali]
└─# cat /home/kali/.config/custom-config.conf
Configuración desde Windows Docker
```

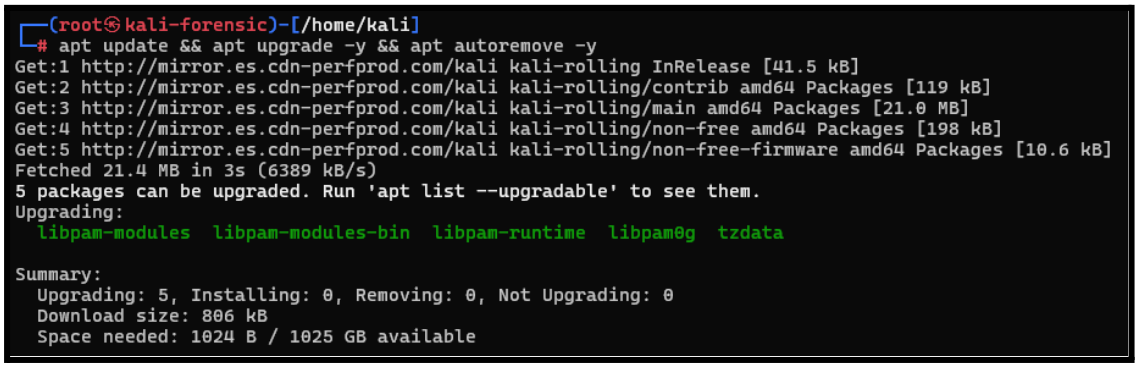
Captura de Pantalla : Lectura de Archivos en Volúmenes Docker con ***cat***

◆ ***Actualización del Sistema Kali dentro del Contenedor***

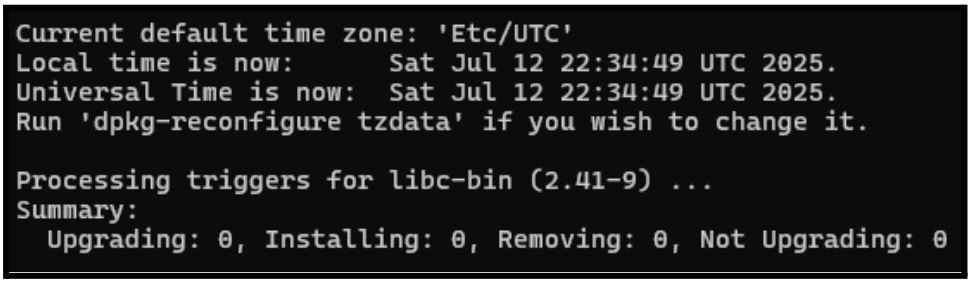
Con el fin de garantizar la estabilidad, seguridad y compatibilidad del entorno forense, se procede a actualizar el sistema base Kali desde dentro del contenedor:

```
apt update && apt upgrade -y && apt autoremove -y
```

Esta acción asegura que todas las herramientas y librerías cuenten con sus versiones más recientes, y que el sistema esté libre de paquetes obsoletos.



Captura de Pantalla : Actualización de Paquetes en Kali Linux

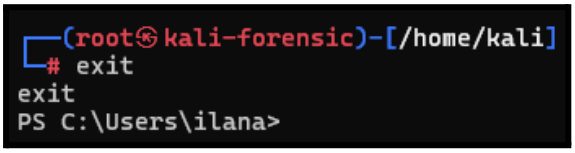


Captura de Pantalla : Finalización de Actualización y Confirmación de Zona Horaria

◆ *Finalización del Trabajo en Contenedor*

Una vez comprobada la persistencia de datos, actualizadas las herramientas base del sistema y verificado el correcto funcionamiento del entorno, se procede a salir de la sesión interactiva dentro del contenedor kali-forensic-lab.

```
# Salir del contenedor
exit
```



Captura de Pantalla : Salida del Contenedor Kali Forensic

– Reflexión Final

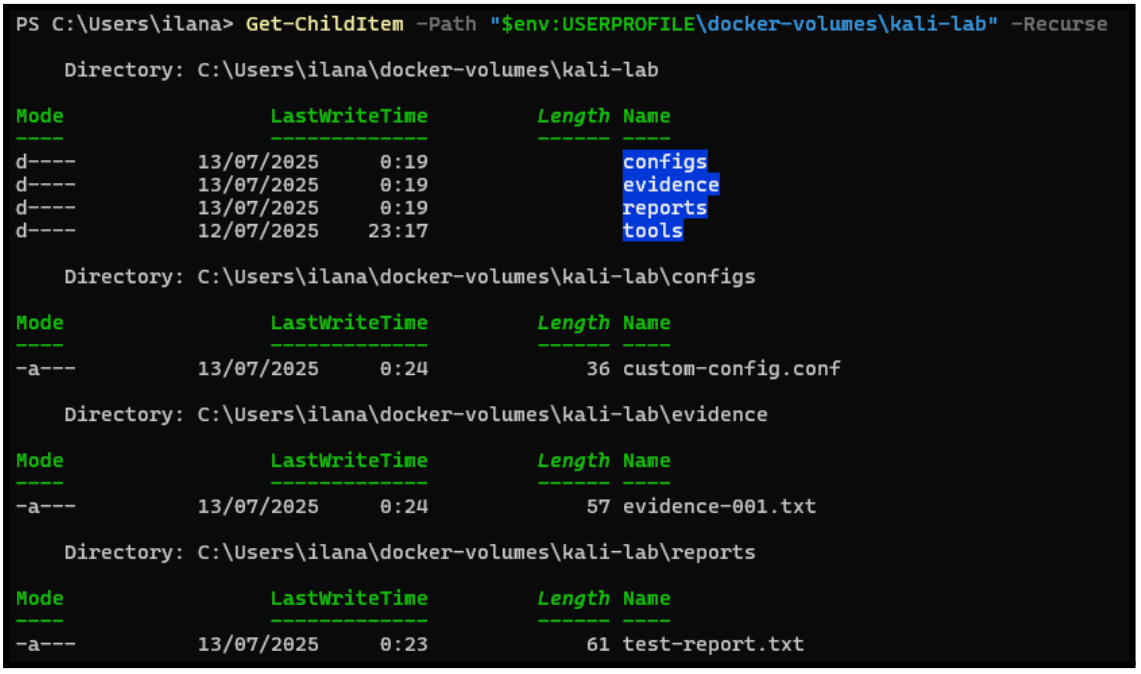
La implementación progresiva de un entorno Kali Linux dentro de contenedores Docker no solo facilita la portabilidad y modularidad de laboratorios de ciberseguridad, sino que también promueve buenas prácticas profesionales como la separación de datos, persistencia de evidencias y versión controlada del entorno. Este enfoque potencia el trabajo forense y ofensivo al garantizar trazabilidad, integridad y recuperación eficiente ante fallos o reconstrucciones. Adoptar esta arquitectura representa un paso clave hacia la automatización y estandarización de entornos de pruebas en equipos de seguridad modernos.

Verificación de Persistencia desde el Sistema Host (Windows)

Una vez completada la fase de volúmenes persistentes, se procede a verificar desde el sistema anfitrión que los archivos creados dentro del contenedor existen en el directorio mapeado en Windows:

◆ Ver archivos creados desde el host

```
# listar todos los archivos creados dentro del volumen
Get-ChildItem -Path "$env:USERPROFILE\docker-volumes\kali-lab" -Recurse
```

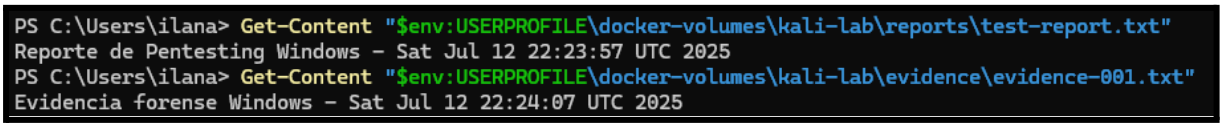


Captura de Pantalla : Verificación de Volúmenes desde PowerShell

◆ Leer contenido de archivos clave

```
# Mostrar contenido del archivo de reporte
Get-Content
"$env:USERPROFILE\docker-volumes\kali-lab\reports\test-report.txt"

# Mostrar contenido de la evidencia forense
Get-Content
"$env:USERPROFILE\docker-volumes\kali-lab\evidence\evidence-001.txt"
```



Captura de Pantalla : Lectura de Archivos Persistentes desde Windows

◆ Estos comandos validan:

✔ Que los archivos existen fuera del contenedor.
✔ Que su contenido fue correctamente persistido.
✔ Que los volúmenes están funcionando como se esperaba.

◆ Conclusión de la Verificación Final de la Persistencia desde el Sistema Host

Tras completar la evolución del contenedor con volúmenes montados, se confirma desde el entorno Windows que los archivos creados dentro del contenedor Kali (reportes, evidencias y configuraciones) están accesibles en el sistema host. Esta validación externa demuestra que la arquitectura de volúmenes ha sido correctamente implementada, asegurando la persistencia de datos críticos incluso tras detener o eliminar el contenedor.

Este procedimiento no forma parte de una fase estructural, sino que actúa como **verificación técnica transversal**, aportando evidencia concluyente sobre la integridad del laboratorio forense y su capacidad de conservación de información sensible.

Sección Complementaria: Verificación Avanzada y Mantenimiento

1. Mapeo de Volúmenes Windows ↔ Kali

El siguiente esquema detalla la correspondencia entre las carpetas del sistema anfitrión (Windows) y las rutas dentro del contenedor Kali, lo que permite mantener la persistencia de datos de forma estructurada y trazable:

Carpeta en Windows	*Ruta en el Contenedor*
C:\Users[Usuario]\docker-volumes\kali-lab\tools	/opt/custom-tools
C:\Users[Usuario]\docker-volumes\kali-lab\reports	/home/kali/reports
C:\Users[Usuario]\docker-volumes\kali-lab\evidence	/home/kali/evidence
C:\Users[Usuario]\docker-volumes\kali-lab\configs	/home/kali/.config

---

2. Análisis : Volúmenes Externos en Docker

La arquitectura de volúmenes externos ofrece ventajas significativas desde una perspectiva de ciberseguridad y continuidad operativa:

- Permite mantener los datos persistentes fuera del ciclo de vida del contenedor.
- Facilita la creación de entornos de pruebas reutilizables y modulares.
- Protege evidencias, configuraciones y reportes frente a reinicios o eliminaciones accidentales.
- Refleja buenas prácticas alineadas con estándares de laboratorios profesionales.

---

3. Validación del Entorno Kali

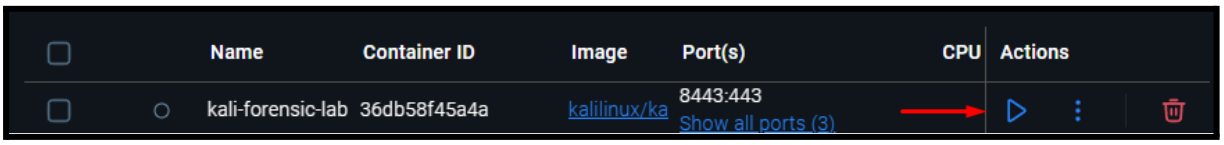
Una vez desplegado el entorno **kali-forensic-lab**, se procede a validar su funcionamiento mediante la instalación de herramientas fundamentales de ciberseguridad y la verificación de

sus versiones.

3.1 Iniciar el Contenedor

El contenedor puede iniciarse de dos formas:

- **Desde Docker Desktop**: Pulsando el botón **"Play"**.



Captura de Pantalla: Inicio del contenedor desde Docker Desktop.

- **Desde terminal PowerShell**:

```
docker start kali-forensic-lab
```

3.2 Acceso al Contenedor

```
docker exec -it kali-forensic-lab /bin/bash
```



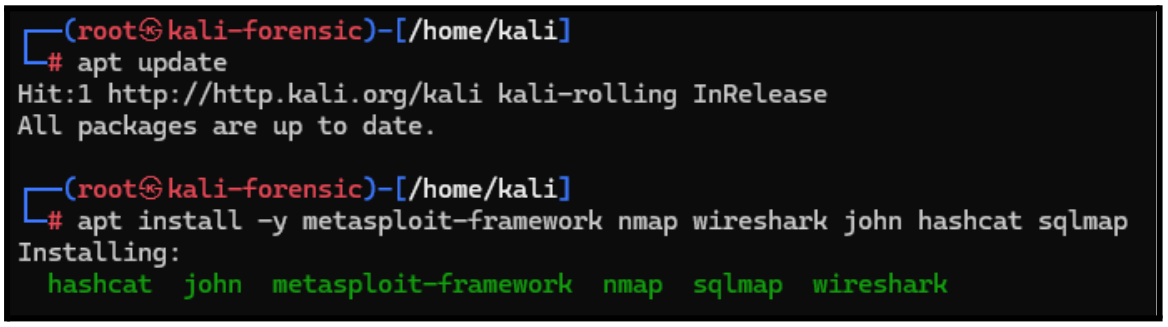
Captura de Pantalla: Iniciar y acceder al contenedor desde PowerShell.

3.3 Actualización e Instalación de Herramientas Clave

Dentro del contenedor, se actualiza el sistema y se instalan herramientas esenciales para pentesting y análisis forense:

```
apt update
apt install -y metasploit-framework nmap wireshark john hashcat
```

```
sqlmap
```

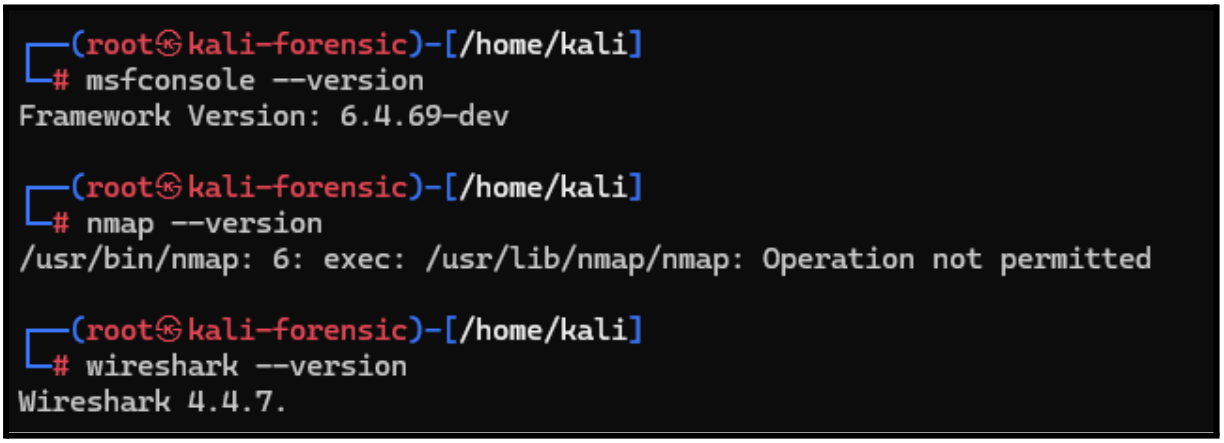


Captura de Pantalla: Proceso de actualización e instalación de herramientas en Kali Linux.

3.4 Verificación de Versiones

A continuación, se verifica que las herramientas se hayan instalado correctamente:

```
msfconsole --version
nmap --version   # Puede mostrar error si no se ejecuta con privilegios elevados
wireshark --version
john hashcat --version
sqlmap --version
```



Capturas de Pantalla: Comprobación de versiones instaladas de las herramientas Metasploit, Nmap y Wireshark.

```
┌──(root㉿kali-forensic)-[/home/kali]
└─# hashcat --version
v6.2.6

┌──(root㉿kali-forensic)-[/home/kali]
└─# sqlmap --version
1.9.6#stable

┌──(root㉿kali-forensic)-[/home/kali]
└─# john
John the Ripper 1.9.0-jumbo-1+bleeding-aec1328d6c 2021-11-02
Copyright (c) 1996-2021 by Solar Designer and others
```

Capturas de Pantalla: Comprobación de versiones instaladas de Hashcat, Sqlmap y John the Ripper.

3.5 Salir del Contenedor

```
exit
```

```
┌──(root㉿kali-forensic)-[/home/kali]
└─# exit
exit
```

Captura de Pantalla: Finalización de la sesión dentro del contenedor.

---

4. Ejecución Temporal con Privilegios Elevados

En determinados escenarios, puede ser necesario contar con un entorno Kali que disponga de privilegios extendidos para ejecutar herramientas como **Wireshark**, **Hashcat** o cualquier utilidad que requiera acceso a recursos del sistema no disponibles en un contenedor estándar.

Este contenedor **no sustituye** al entorno principal **kali-forensic-lab**, sino que se ejecuta de forma puntual con fines operativos o de validación técnica.

> ◆ **Nota previa:**
> Antes de iniciar el nuevo contenedor con privilegios (kali-root-lab), se debe detener el contenedor actual para evitar conflictos de acceso a volúmenes y puertos.

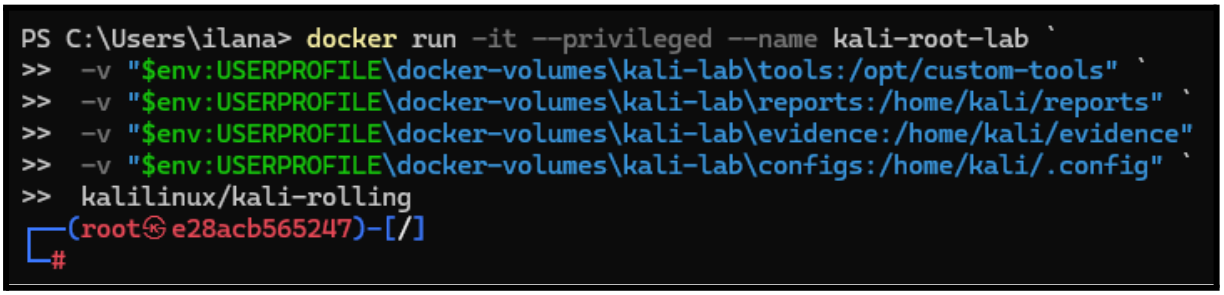
```
#Ejecutar:

docker stop kali-forensic-lab
```

4.1 Creación de Contenedor Temporal con Privilegios

Desde PowerShell, ejecutamos el siguiente comando para iniciar un contenedor temporal con acceso privilegiado y los volúmenes previamente creados:

```
docker run -it --privileged --name kali-root-lab `
 -v "$env:USERPROFILE\docker-volumes\kali-lab\tools:/opt/custom-tools" `
 -v "$env:USERPROFILE\docker-volumes\kali-lab\reports:/home/kali/reports" `
 -v "$env:USERPROFILE\docker-volumes\kali-lab\evidence:/home/kali/evidence" `
 -v "$env:USERPROFILE\docker-volumes\kali-lab\configs:/home/kali/.config" `
 kalilinux/kali-rolling
```



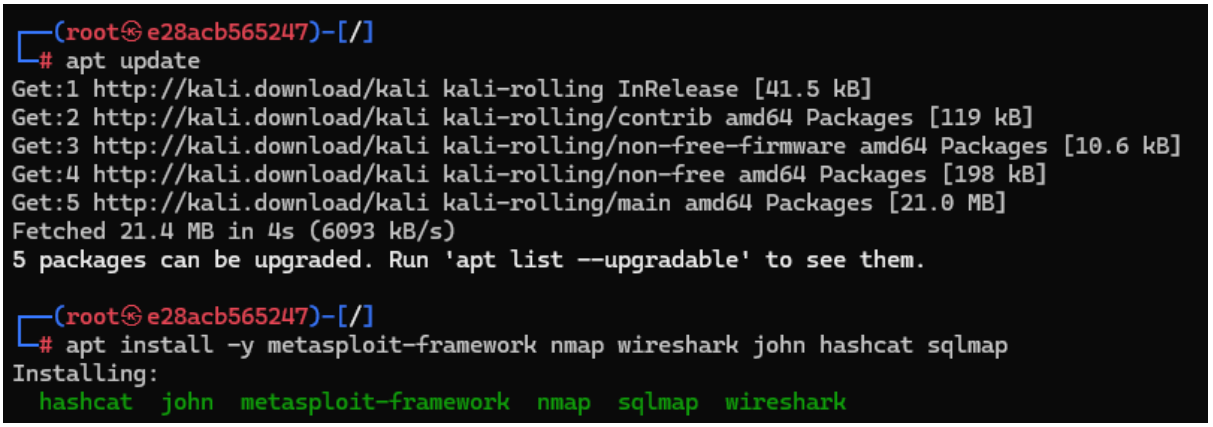
Captura de Pantalla : Ejecución de contenedor Kali con privilegios y volúmenes montados

4.2 Instalación de Herramientas y Verificación en el Contenedor con Privilegios

Una vez iniciado el contenedor temporal **kali-root-lab** con privilegios elevados, se procede a instalar manualmente las herramientas de ciberseguridad necesarias, ya que los contenedores no comparten instalaciones a menos que estén predefinidas en una imagen personalizada.

◆ **Comandos:**

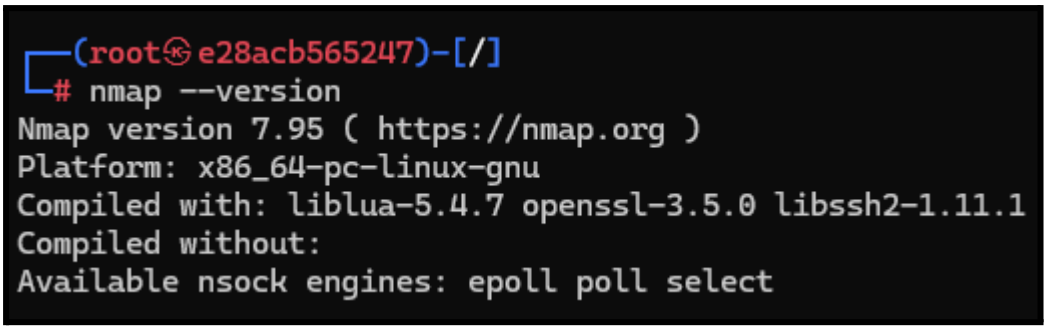
```
apt update
apt install -y metasploit-framework nmap wireshark john hashcat
sqlmap
```



*Captura de pantalla: Actualización e instalación de herramientas en **kali-root-lab***

Una vez completada la instalación, se realiza la comprobación de la versión de nmap que es una de las herramientas que en el contenedor anterior nos daba error por no tener privilegios al querer visualizar su versión.

```
nmap --version
```



*Captura de pantalla: Versión de la herramienta nmap en **kali-root-lab***

Una vez dentro del contenedor, se pueden ejecutar herramientas que requieran acceso extendido al sistema. Este entorno es útil para realizar capturas de pantalla técnicas o pruebas puntuales sin afectar el entorno persistente principal.

4.2.1 Prueba de escaneo con Nmap

Como validación práctica del entorno privilegiado, se realiza una prueba básica de escaneo con nmap contra la IP real del sistema anfitrión. Esto permite verificar tanto la conectividad como la funcionalidad de la herramienta instalada.

- **Dirección IP objetivo:**

```
192.168.1.139
```

Esta corresponde a la interfaz de red principal del sistema Windows donde se ejecuta Docker.

- **Comando ejecutado desde el contenedor:**

```
nmap -sV 192.168.1.139
```

La opción **-sV** permite detectar versiones de los servicios disponibles en los puertos abiertos del objetivo. Esta operación validó también la capacidad de ejecución de escaneos reales desde el contenedor con privilegios elevados.

```
┌──(root㉿e28acb565247)-[/]
└─# nmap -sV 192.168.1.139
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-13 14:43 UTC
Nmap scan report for 192.168.1.139
Host is up (0.0024s latency).
Not shown: 995 filtered tcp ports (no-response)
PORT    STATE  SERVICE       VERSION
53/tcp  closed domain
80/tcp  open   http          Microsoft IIS httpd 10.0
135/tcp open   msrpc         Microsoft Windows RPC
139/tcp open   netbios-ssn   Microsoft Windows netbios-ssn
445/tcp open   microsoft-ds?
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/
Nmap done: 1 IP address (1 host up) scanned in 24.44 seconds
```

Captura de pantalla: Resultado del escaneo desde Kali.

4.3 Eliminación del Contenedor Temporal

Al finalizar su uso, se recomienda eliminar el contenedor para mantener un entorno limpio:

```
docker rm -f kali-root-lab
```

```
PS C:\Users\ilana> docker rm -f kali-root-lab
```

Captura de pantalla: Comando de eliminación del contenedor temporal.

◆ ***Nota:*** *No se elimina el contenedor* ***kali-root-lab*** *para prácticas futuras.*

---

5. Pruebas de Conectividad desde Windows

Una vez desplegado el contenedor **kali-forensic-lab** con los puertos mapeados correctamente, se procede a verificar desde el sistema anfitrión (Windows) que los servicios están accesibles a través de los puertos expuestos.

5.1 Verificación de Puertos Expuestos

Desde PowerShell en Windows, se ejecutan los siguientes comandos para comprobar la conectividad hacia los puertos 8080, 4444 y 8443. Se debe tener iniciado el contenedor **kali-forensic-lab** para que los tests se ejecuten correctamente.

```
powershell
test-netconnection -ComputerName localhost -Port 8080
test-netconnection -ComputerName localhost -Port 4444
test-netconnection -ComputerName localhost -Port 8443
```

Cada comando devuelve información sobre el estado de conectividad, incluyendo la resolución DNS, el estado del puerto **(TcpTestSucceeded)**, y la latencia estimada.

```
PS C:\Users\ilana> test-netconnection -ComputerName localhost -Port 8080

ComputerName     : localhost
RemoteAddress    : ::1
RemotePort       : 8080
InterfaceAlias   : Loopback Pseudo-Interface 1
SourceAddress    : ::1
TcpTestSucceeded : True

PS C:\Users\ilana> test-netconnection -ComputerName localhost -Port 4444

ComputerName     : localhost
RemoteAddress    : ::1
RemotePort       : 4444
InterfaceAlias   : Loopback Pseudo-Interface 1
SourceAddress    : ::1
TcpTestSucceeded : True

PS C:\Users\ilana> test-netconnection -ComputerName localhost -Port 8443

ComputerName     : localhost
RemoteAddress    : ::1
RemotePort       : 8443
InterfaceAlias   : Loopback Pseudo-Interface 1
SourceAddress    : ::1
TcpTestSucceeded : True
```

Captura de Pantalla: Resultado de pruebas de conectividad a puertos mapeados desde Windows.

5.2 Monitoreo de Recursos del Contenedor en Tiempo Real

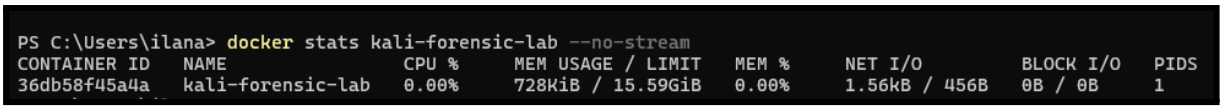
Una vez verificada la conectividad y el estado de los puertos, se procede a observar el comportamiento del contenedor **kali-forensic-lab** en cuanto a consumo de recursos. Esta operación es útil para:

- Verificar que el contenedor está activo y operando correctamente.
- Detectar posibles picos anómalos de consumo (CPU, RAM, red, disco) durante tareas forenses intensivas.
- Corroborar que los volúmenes y servicios se ejecutan dentro de parámetros aceptables.

Para ello se emplea el siguiente comando de PowerShell:

```
# Monitoreo del contenedor
docker stats kali-forensic-lab --no-stream
```

Este comando permite obtener una fotografía del estado actual del contenedor sin dejar el monitoreo activo indefinidamente (**--no-stream**).



Captura de Pantalla donde se observa el uso de CPU, memoria, red y disco del contenedor.

---

6. Monitoreo y Administración del Contenedor

Para asegurar la correcta operación del entorno forense en Docker, se implementan tareas de monitoreo y gestión tanto desde la línea de comandos como desde la interfaz gráfica de Docker Desktop. Estas acciones permiten detectar fallos, controlar recursos y mantener una administración eficiente del contenedor **kali-forensic-lab**.

6.1 Comandos desde PowerShell

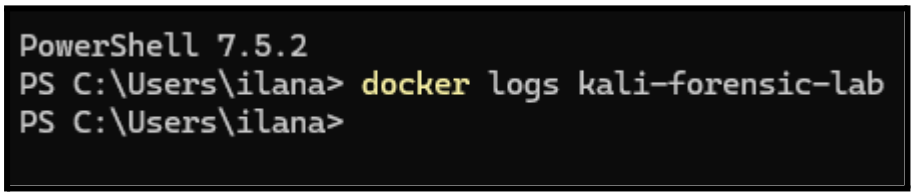
Se utilizan los siguientes comandos desde PowerShell para supervisar el contenedor y obtener información clave:

```
# Monitoreo de recursos en tiempo real
docker stats kali-forensic-lab
```



Captura de pantalla : Monitoreo de recursos del contenedor ***kali-forensic-lab*** *con* ***docker stats***

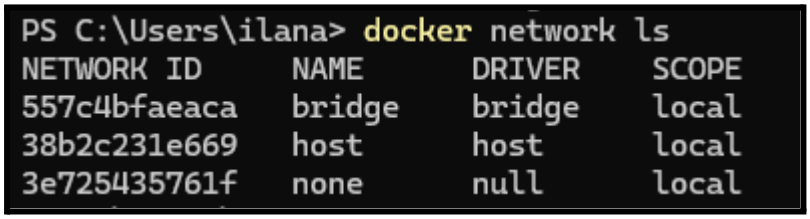
```
# Revisión de logs (salida estándar del contenedor)
docker logs kali-forensic-lab
```



Captura de pantalla : Revisión de logs del contenedor sin salida

◆ **Nota:** Este contenedor se ejecuta en modo interactivo sin servicios preinstalados que generen salida a consola, por lo tanto el comando **docker logs** no devuelve información en este caso.

```
# Información de redes Docker activas
docker network ls
```



Captura de pantalla : Listado de redes disponibles en Docker

Estos comandos permiten validar el estado operativo, diagnosticar errores en ejecución y revisar la integración del contenedor con la red Docker.

6.2 Interfaz Gráfica en Docker Desktop

Docker Desktop ofrece una vista amigable y detallada para la administración del contenedor. Los pasos para su revisión son los siguientes:

1. Abrir **Docker Desktop**.
2. Ir a la pestaña **Containers** y seleccionar kali-forensic-lab.
3. Se puede revisar:
    - Estadísticas de uso de **CPU, RAM, I/O**.
    - Registro de **logs** en tiempo real.
    - Volúmenes y directorios **montados**.



Captura de pantalla - Estadísticas de uso de recursos del contenedor kali-forensic-lab en Docker Desktop

6.3 Comandos adicionales de gestión del entorno

Para obtener información adicional sobre el estado global del sistema Docker y su almacenamiento, se pueden ejecutar los siguientes comandos desde PowerShell :

```
# Listado de todos los contenedores (activos e inactivos)
```

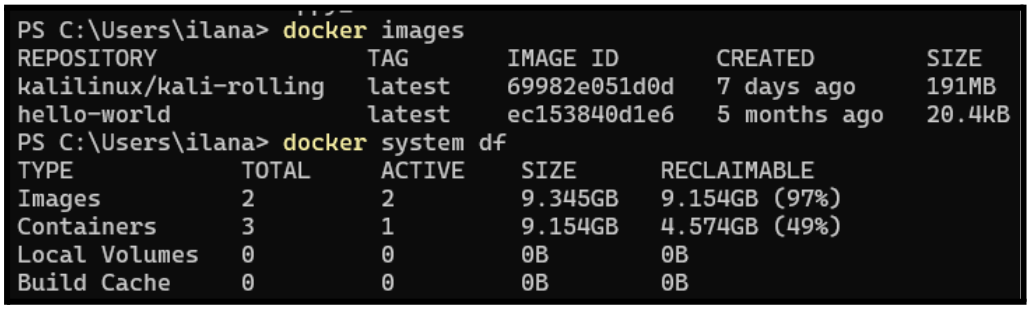
```
docker ps -a
```

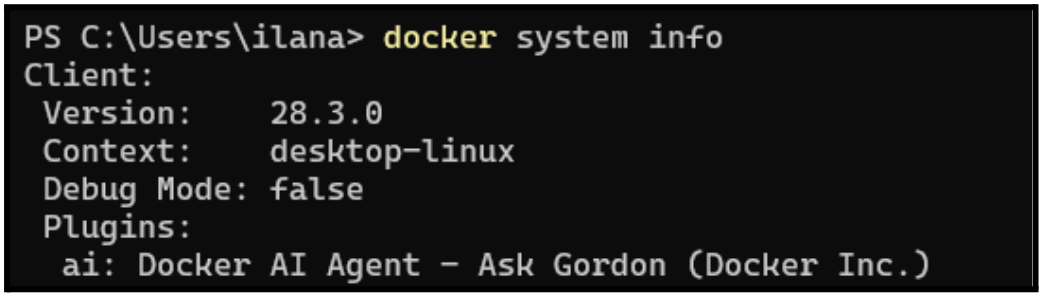
```
# Listado de imágenes descargadas
docker images
```

```
# Uso de espacio en volúmenes, imágenes y contenedores
docker system df
```

```
# Información completa del sistema Docker
docker system info
```







Capturas de Pantalla con salida combinada de docker ps -a, docker images, y docker system df.

Estos comandos son útiles para el mantenimiento preventivo, depuración de recursos y auditorías técnicas del entorno Docker

7. Limpieza del Entorno (Opcional)

Este bloque está diseñado para realizar una limpieza general del entorno Docker tras finalizar la práctica, recuperando espacio y eliminando recursos que ya no se utilicen. Es una medida útil cuando se realizan muchos experimentos o laboratorios, evitando acumulación innecesaria de contenedores, imágenes o volúmenes.

> ⚠ **Nota importante:** En este caso **no se eliminará el contenedor kali-forensic-lab**, ya que puede reutilizarse para futuras prácticas. Las instrucciones están incluidas únicamente con fines demostrativos.

7.1 Comandos en PowerShell:

```
# Detener todos los contenedores
# ⚠ No ejecutar si deseas conservar el contenedor activo
docker stop $(docker ps -aq)
```

```
# Eliminar contenedores detenidos
# ⚠ No ejecutar si se deseas conservar el contenedor de
laboratorio
docker rm $(docker ps -aq)
```

```
# Eliminar imágenes no utilizadas
docker image prune -a
```

```
# Eliminar volúmenes huérfanos
docker volume prune
```

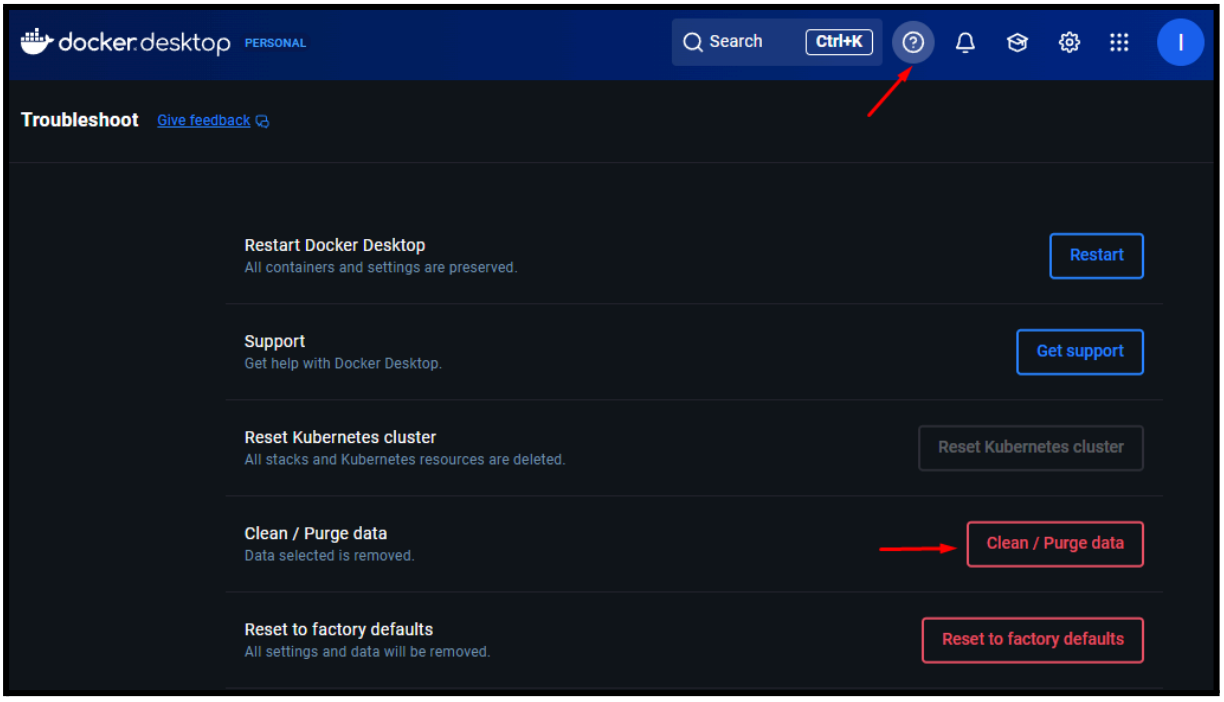
```
# Limpieza completa del sistema (incluye volúmenes)
# ⚠ Esta opción elimina TODO (contenedores, imágenes, volúmenes)
docker system prune -a --volumes
```

```
# Verificar espacio recuperado
docker system df
```

7.2 Limpieza desde Docker Desktop (GUI):

1. Abrir **Docker Desktop**.
2. Hacer clic en el **icono de interrogación (?)** en la esquina superior derecha.
3. Seleccionar **"Troubleshoot"**.
4. Hacer clic en **"Clean / Purge data"**.
5. Confirmar si se está seguro de eliminar datos innecesarios.

> ⚠ **Recomendación:** Realiza limpieza únicamente de imágenes o volúmenes huérfanos si no estás segura del impacto. Evita el comando prune -a --volumes sin revisar primero.



Captura de la sección "Troubleshoot > Clean / Purge data" en Docker Desktop

CONCLUSIÓN

La implementación exitosa de contenedores Docker en el ámbito de la ciberseguridad ha demostrado ser una metodología altamente efectiva para la creación de laboratorios de seguridad informática. A través de esta práctica, se ha evidenciado cómo la tecnología de contenedores proporciona un entorno controlado, aislado y reproducible que es fundamental para el desarrollo de competencias en seguridad informática. La utilización de Kali Linux como imagen base ha permitido acceder a un ecosistema completo de herramientas profesionales de pentesting y análisis de seguridad, configuradas de manera óptima para trabajo en producción.

La configuración de puertos específicos para herramientas de ciberseguridad ha revelado la importancia de la gestión de redes en entornos contenedorizados. El mapeo de puertos 8080, 4444 y 8443 no solo permite el acceso a interfaces web y servicios de red, sino que también mantiene la seguridad del sistema host mediante el aislamiento de procesos. Esta aproximación es especialmente relevante en escenarios de pentesting donde se requiere ejecutar herramientas potencialmente peligrosas sin comprometer la integridad del sistema principal.

La implementación de nomenclatura profesional y sistema de etiquetado ha demostrado ser crucial para la gestión eficiente de infraestructuras Docker complejas. El uso de nombres descriptivos como "pentesting-lab-kali-2025" junto con metadatos estructurados facilita la identificación, categorización y gestión de múltiples contenedores en entornos de producción. Esta práctica no solo mejora la organización del trabajo, sino que también establece las bases para la automatización y orquestación de laboratorios de seguridad a gran escala.

La configuración de volúmenes persistentes representa uno de los aspectos más críticos de la práctica, ya que garantiza la continuidad y trazabilidad del trabajo en ciberseguridad. La separación lógica de datos en directorios específicos para herramientas, reportes, evidencia y configuraciones refleja las mejores prácticas en gestión de información forense. Esta aproximación asegura que los datos sobrevivan a la eliminación y recreación de contenedores, y facilita el cumplimiento de requisitos regulatorios sobre preservación de evidencia digital.

Finalmente, esta práctica ha establecido una base sólida para el desarrollo de competencias avanzadas en ciberseguridad mediante tecnologías modernas de contenedores. Las habilidades adquiridas en gestión de Docker, redes aisladas, persistencia de datos y automatización de despliegues son directamente aplicables en entornos donde la eficiencia, seguridad y escalabilidad son esenciales. La experiencia práctica con herramientas como Kali Linux en entornos contenedorizados prepara a los estudiantes para enfrentar desafíos reales de la industria, donde la capacidad de gestionar laboratorios dinámicos es una competencia cada vez más valorada.

BIBLIOGRAFÍA

[1] Docker Inc., "Docker Hub – Repositorio oficial de imágenes de contenedores", *Docker*, 2024. [En línea]. Disponible en: https://hub.docker.com/

[2] Offensive Security, "Kali Linux Docker Images", *Kali.org*, 2024. [En línea]. Disponible en: https://www.kali.org/docs/containers/using-kali-docker-images/

[3] Docker Inc., "Persistencia de datos en Docker", *Docker Docs*, 2024. [En línea]. Disponible en: https://docs.docker.com/storage/volumes/

[4] OWASP Foundation, "OWASP Container Security Verification Standard", *OWASP*, 2024. [En línea]. Disponible en: https://owasp.org/www-project-container-security/

[5] Check Point Software, "Seguridad de contenedores Docker", *Check Point Cyber Hub*, 2024. [En línea]. Disponible en: https://www.checkpoint.com/es/cyber-hub/cloud-security/what-is-container-security/docker-container-security/

[6] Red Hat Inc., "Guía de seguridad para contenedores", *Red Hat Customer Portal*, 2024. [En línea]. Disponible en: https://access.redhat.com/documentation/es-es/red_hat_enterprise_linux/8/html/building_running_and_managing_containers/

[7] Apuntes de DevOps, "Docker Hub y gestión de imágenes", *Apuntes DevOps*, 2024. [En línea]. Disponible en: https://apuntes.de/docker-certificacion-dca/docker-hub/

[8] Una Mirada Entre Bytes, "Gestión de volúmenes en Docker", *Blog Técnico*, 2024. [En línea]. Disponible en: https://unamiradaentrebytes.wordpress.com/2024/06/12/gestion-de-volumenes-y-almacenamiento-persistente-en-docker/

[9] Hostinger, "Cómo crear un contenedor Docker", *Hostinger España*, 2024. [En línea]. Disponible en: https://www.hostinger.es/tutoriales/como-crear-contenedor-docker

[10] LabEx, "Persistencia de datos en volúmenes de Docker", *LabEx.io*, 2024. [En línea]. Disponible en: https://labex.io/es/tutorials/docker-how-to-persist-data-in-docker-volumes-493636