# Servicios en Linux Ubuntu Server con clientes en Windows

UN ENFOQUE TEÓRICO PRÁCTICO

CÉSAR GRIJALVA / ÁLVARO CEVALLOS / GALO PUETATE




PUCE Ibarra
ediPUCE
PUBLICALO
y genera conocimiento que transforma

# Servicios en Linux Ubuntu Server con clientes en Windows

UN ENFOQUE TEÓRICO PRÁCTICO

**CÉSAR GRIJALVA**

**ÁLVARO CEVALLOS**

**GALO PUETATE**

edi
PUCE

Servicios en Linux Ubuntu Server con clientes en Windows
UN ENFOQUE TEÓRICO PRÁCTICO

Primera edición
© 2026 César Grijalva. Docente de la Pontificia Universidad Católica del Ecuador
© 2026 Álvaro Cevallos. Docente de la Pontificia Universidad Católica del Ecuador
© 2026 Galo Puetate. Docente de la Pontificia Universidad Católica del Ecuador

ediPUCE
laeditorial.puce.edu.ec
Quito, Av. 12 de Octubre y Roca
Apartado n.º 17-01-2184
Telf.: (593) (02) 2991 700 ext. 1711
Correo: publicaciones@puce.edu.ec

Producción editorial: Jossué Baquero
Gestión técnica: Macarena Orozco
Asistencia editorial: Danna Quintana
Diseño de portada: Oswaldo Portilla
Diagramación: Oswaldo Portilla PUCE Ibarra
Corrección de textos: ediPUCE

ISBN digital: 978-9978-77-810-4

Quito, julio de 2026




Esta obra está bajo una licencia de Creative Commons
Reconocimiento-No Comercial-CompartirIgual 4.0 Internacional

El manuscrito se sometió a revisión de pares ciegos lo que garantiza la confidencialidad de autores y árbitros, conforme a las normas de publicación de la Pontificia Universidad Católica del Ecuador

# DEDICATORIA

A mi querida hija,

nunca dejes de aprender y crecer, y siempre

recuerda que puedes lograr cualquier cosa que te propongas.

César

Para ti, hijo mío,
que el conocimiento te acompañe siempre
y que nunca pierdas la curiosidad por descubrir el mundo.

Álvaro

A mis hijos,

que la vida les regale caminos llenos de aprendizajes.

Galo

# PREFACIO

El objetivo de este libro es que el lector se instruya en la instalación de paquetes, la configuración de servicios y el manejo del sistema operativo Linux Ubuntu Server versión 24.04. Se trabaja también con ejemplos prácticos en los que se emplea *software* cliente en Windows 11, con la finalidad de que usted conozca y observe la variedad de interacciones que pueden existir entre Linux y Windows.

A lo largo del presente libro, se trabajará, en lo posible, desde una terminal Linux. Esto tiene varias razones: optimizar los recursos de *hardware,* ya que los entornos gráficos consumen memoria RAM, CPU y en un entorno de servidor cada bit de rendimiento puede ser crítico para la eficiencia del sistema, es decir, los servidores pueden dedicarse completamente a tareas importantes como alojar aplicaciones web, gestionar bases de datos, o manejar servicios de red, mejorando la capacidad de respuesta y la escalabilidad del servidor. Otra razón por considerar es la seguridad que ofrece Linux, pues, al no tener un entorno de escritorio, se reduce la superficie de ataque del servidor; menos *software* instalado significa menos vulnerabilidades potenciales que podrían ser explotadas. Los servidores suelen ser objetivos de ataques cibernéticos, por lo que mantener un sistema lo más liviano y seguro posible es una prioridad.

**El capítulo I** es una introducción a Linux. Inicialmente, hablaremos sobre la administración del *kernel* de Linux, un elemento importante de los sistemas modernos que consiste en una interfaz entre el *hardware* de un ordenador y sus procesos. Abordaremos también el tema de las distribuciones Linux, entre otros detalles como la instalación y compilación del *kernel* en Linux Ubuntu Server.

**El capítulo II** se trata de una explicación paso a paso del proceso de instalación de Linux Ubuntu Server. Este sistema operativo se instala en modo texto, no obstante, puede instalarse un entono gráfico como KDE o GNOME; asimismo, los administradores de sistemas recomiendan el manejo desde una terminal. Algo importante es que el usuario debe saber que existen algunas variantes de Ubuntu: Ubuntu Desktop, Ubuntu Pro, Ubuntu Core y Ubuntu Server, cada una con sus especificaciones y funcionalidades específicas.

**Capítulo III.** Aquí abordaremos la línea de comandos y, aunque puede parecer intimidante al principio, esta línea ofrece una flexibilidad y potencia inigualables para la administración del sistema operativo. Las tareas pueden ser automatizadas mediante *scripts*, lo que permite a los administradores ejecutar comandos repetitivos y complejos de manera eficiente y rápida.

**En el capítulo IV** repasaremos paquetes esenciales en Linux Ubuntu Server, es decir, este capítulo trata sobre la instalación y una breve explicación de aquellos paquetes que tienen mayor importancia o que se utilizan con mayor frecuencia. Los paquetes consisten en una colección de archivos que permiten la instalación de un determinado programa que servirá para gestionar de mejor manera nuestro sistema operativo.

**Capítulo V.** Se tratará sobre la instalación y utilidad del *software* cliente en Windows, que es del tipo *freeware,* es decir, podremos descargarlo e instalarlo gratuitamente sin las implicaciones que tiene el *software* con licencia. Por ejemplo, el objetivo de clientes como FileZilla Client o Bitvise SSH Client es conectarse remotamente a nuestro servidor Linux y desde MySQL Workbench Client a nuestra base de datos MySQL.

**Capítulo VI.** Aquí se explican las pautas más importantes para la instalación y configuración de MySQL en Linux, aspectos de seguridad como el ajuste de autenticación y privilegios de usuario, acceso remoto desde el cliente MySQL Workbench, creación de una base datos y algunos comandos básicos de MySQL.

**Capítulo VII.** Aquí se trata del proceso de instalación y configuración de Apache Web Server sobre Linux Ubuntu Server, las funciones principales de Apache, como habilitar y deshabilitar el servicio, sus principales características. Se explicará también cómo podemos crear y configurar *hosts* virtuales, interfaces virtuales, veremos la instalación de php y phpMyAdmin para la administración de bases de datos.

**Capítulo VIII.** Programación aplicada a la administración de sistemas Linux. En Linux, los lenguajes de programación como C y C++ se destacan por su eficiencia y control del *hardware*, siendo esenciales para el desarrollo de sistemas operativos y aplicaciones de alto rendimiento. Python, conocido por su sintaxis sencilla, es ideal para desarrollo rápido, automatización y análisis de datos. Shell Scripting permite automatizar tareas del sistema mediante comandos escritos en *scripts*, optimizando flujos de trabajo. La compilación y ejecución de programas en Linux sigue un enfoque práctico: lenguajes como C y C++ se compilan con herramientas como gcc o g++, mientras que Python ejecuta directamente *scripts* interpretados. Las *scripts* de programación en Linux para estos lenguajes ofrecen un entorno propicio para aprender desde la compilación básica hasta ver los resultados, aprovechando al máximo las capacidades del sistema operativo.

**Capítulo IX.** Aspectos de seguridad en Linux. La seguridad en Linux incluye múltiples herramientas y configuraciones diseñadas para proteger sistemas y redes. El sistema de inicialización y gestor de servicios systemd controla el arranque y la administración de servicios, ofreciendo un control detallado sobre procesos críticos del sistema. Nmap (Network Mapper) es una herramienta poderosa para el análisis de redes, utilizada para identificar *hosts* activos, puertos abiertos y servicios en ejecución, esencial en auditorías de seguridad. Linux también es una plataforma destacada para el *hacking* ético, proporcionando un entorno robusto para pruebas de penetración con herramientas como Metasploit o Kali Linux. Además, el *firewall* iptables permite configurar reglas avanzadas de filtrado de tráfico, reforzando la protección frente a amenazas externas. Estos elementos hacen de Linux una opción confiable para entornos donde la seguridad es prioritaria.

Esta obra tiene un enfoque teórico práctico, así que, en resumen, iniciaremos desde la introducción manejo de *kernel*, instalación de Linux Ubuntu Server https://ubuntu.com/download/server y paquetes específicos para realizar determinadas tareas, línea de comandos, instalación de *software* cliente para Windows, para luego pasar a los diferentes servicios en Linux y detallar la utilidad que estos tienen. La gestión de paquetes permitirá a los usuarios instalar, actualizar, eliminar y gestionar *software* de manera eficiente.

Ahora que hemos establecido el propósito y la estructura de este libro, es momento de sumergirnos en el mundo de Linux Ubuntu Server. Al descubrir este *software* libre y lo útil que es en el ámbito de administración de servicios, no solo ganaremos una perspectiva más profunda sobre sistemas, sino que también sentaremos las bases para los conceptos más avanzados que se desarrollarán en los capítulos siguientes como es la instalación y configuración de bases de datos, servidor web, configuración de otros servicios y su utilidad. Le invito a embarcarse en este viaje de descubrimiento y a mantener la mente abierta a las ideas y soluciones que iremos desgranando a lo largo de estas páginas.

# PRÓLOGO

Estimado lector:

La presente obra representa un esfuerzo sistemático y riguroso orientado a proporcionar una guía integral para la instalación, configuración y administración del sistema operativo Linux Ubuntu Server en su versión 24.04, así como para la interacción eficiente con software cliente en entornos Windows. Este libro surge de la necesidad de contar con un recurso actualizado y práctico que facilite la comprensión y aplicación de conceptos fundamentales en administración de sistemas, redes y servicios en plataformas Linux, enfocado especialmente en entornos educativos y profesionales.

El desarrollo de este texto se fundamenta en la experiencia acumulada a lo largo de una trayectoria profesional y académica que incluye la gestión de sistemas informáticos, administración de redes, docencia universitaria y proyectos de desarrollo de *software*. Asimismo, está respaldado por una sólida formación académica, con títulos de Ingeniería en Sistemas Computacionales, Maestría en Tecnologías para la Gestión y Práctica Docente, y Máster Universitario en Ingeniería de Software y Sistemas Informáticos, otorgados por instituciones reconocidas como la Universidad Técnica del Norte, la Pontificia Universidad Católica del Ecuador y la Universidad Internacional de La Rioja.

El contenido está diseñado para brindar al lector un enfoque teórico-práctico que permita comprender desde los fundamentos del *kernel* Linux hasta la configuración avanzada de servicios como servidores web, bases de datos y seguridad informática, incluyendo además el uso de lenguajes de programación y automatización en entornos Linux. Este libro está dirigido a estudiantes, docentes, administradores de sistemas y profesionales del área de tecnologías de la información que buscan fortalecer sus competencias técnicas y teóricas en el manejo de sistemas operativos libres y su integración en ambientes heterogéneos.

Finalmente, este prólogo no puede estar completo sin expresar una profunda gratitud: a Dios, a las instituciones académicas que han sido parte fundamental de nuestra formación, a nuestras familias por su apoyo constante, y a los lectores, por confiar en este material como una herramienta para su desarrollo profesional y académico. Confiamos en que esta obra será un aporte significativo para el aprendizaje y aplicación práctica de tecnologías Linux en el área actual de la informática.

Los autores

# CONTENIDO


PREFACIO ..... i
PRÓLOGO ..... iv
CAPÍTULO I INGENIERÍA DEL KERNEL ..... 5
Contenido del capítulo I ..... 5
¿Qué es el kernel? ..... 5
El rol del Kernel en entornos virtualizados y contenedores (Docker/LXC) ..... 8
Instalación y Configuración de LXD y Docker ..... 12
Ejercicios Prácticos ..... 17
Modos de operación ..... 18
Seguridad y Anillos de Ejecución ..... 21
Ejercicio ¿cómo interceptar la comunicación entre el anillo 3 y el anillo 0 ..... 24
1. Instalación de **strace** ..... 24
2. **El Experimento:** ¿Qué hace el comando ls en realidad? ..... 24
3. **Nivel Pro:** Espiando un contenedor Docker en vivo ..... 24
Actualización del kernel en Ubuntu ..... 25
Configuración y compilación del kernel ..... 31
Gestión de parches de seguridad y técnicas de Live Patching. ..... 34
Distribuciones Linux ..... 37
CAPÍTULO II INSTALACIÓN DE LINUX UBUNTU SERVER ..... 41
Contenido del capítulo II ..... 41
Linux Ubuntu Server ..... 41
Comparativa entre Ubuntu Server y otras distros ..... 42
Qué hay de novedades ..... 45
Requisitos mínimos del sistema ..... 46
Creación del USB de arranque ..... 46
Proceso de instalación ..... 50
Errores comunes durante la instalación ..... 64
CAPÍTULO III LÍNEA DE COMANDOS LINUX ..... 66
Contenido del capítulo III ..... 66
Estructura y sistema de archivos ..... 66
Directorios y archivos importantes ..... 73
Comandos esenciales ..... 75
Ejercicios propuestos ..... 90
CAPITULO IV PAQUETES ESENCIALES EN LINUX ..... 92
Contenido del capítulo IV ..... 92
Introducción a los paquetes ..... 92
Herramientas de red net-tools ..... 92
Herramienta de red iproute2 ..... 94
Editor Midnight Commander ..... 95
Editor Nano ..... 96

Herramienta de monitoreo y análisis de red nmap ..... 97
Paquete dnsutils y sus herramientas. ..... 98
CAPÍTULO V SOFTWARE CLIENTE EN WINDOWS ..... 101
Contenido del capítulo V ..... 101
Introducción al software cliente en Windows ..... 101
Filezilla Client ..... 102
Free Port Scanner ..... 103
Bitvise SSH Client ..... 105
MySQL Workbench 8.0 CE ..... 106
Arquitectura Cliente Servidor ..... 108
Componentes importantes: ..... 108
Seguridad en Conexiones Remotas ..... 109
1. Cifrado de Extremo a Extremo ..... 109
2. Autenticación por Llaves Públicas (SSH Keys) ..... 109
3. El concepto de "Puertos" y Firewalls ..... 109
CAPITULO VI INSTALACIÓN Y CONFIGURACIÓN DE MYSQL ..... 110
Contenido del capítulo VI ..... 110
Características de MySQL ..... 111
Instalar y Administrar el proceso de MySQL ..... 111
Bases de datos por defecto ..... 114
Ajuste de autenticación y privilegios de usuario ..... 114
Tomar precauciones de seguridad ..... 116
Permitiendo conexiones remotas ..... 117
Comandos esenciales de MySQL ..... 123
CAPÍTULO VII INSTALACIÓN Y CONFIGURACIÓN DE APACHE WEB SERVER ..... 128
Contenidos del Capítulo VII ..... 128
Introducción ..... 128
Funciones Principales de Apache Web Server ..... 129
Instalar y administrar el proceso de Apache ..... 130
Comprobando el servidor web ..... 132
Principales características y usos de los números de puerto ..... 137
Configurar hosts virtuales ..... 138
Características y Usos de las Interfaces Virtuales ..... 142
Instalación de php y phpMyAdmin ..... 148
Nociones básicas de seguridad web ..... 153
..... 154
¿Qué información puede encontrar en el Visor? ..... 155
Utilidad para un Administrador de Servidores ..... 155
¿Cómo acceder a él? ..... 155
I. Opción Profesional: Let's Encrypt (Certbot) ..... 156
II. Opción de Laboratorio: Certificado Autofirmado ..... 156
III. Verificación de la Configuración ..... 157
CAPÍTULO VIII Programación aplicada a la Administración de Sistemas Linux ..... 158
Contenidos del Capítulo VIII ..... 158

Introducción ..... 158
Instalación y configuración del entorno de desarrollo ..... 158
Fundamentos de programación en Linux ..... 159
Scripting con Python para la gestión de Linux ..... 163
Scripting en C y C++ para la gestión de Linux ..... 170
Conclusiones ..... 176
CAPÍTULO IX Aspectos de Seguridad en Linux. ..... 177
Contenidos del Capítulo IX ..... 177
Introducción a la Seguridad ..... 177
Gestión de Usuarios y Permisos ..... 178
Seguridad en Servicios de Red ..... 179
Registro y Monitoreo de Actividad ..... 183
Actualizaciones y Gestión de Paquetes ..... 186
Copias de Seguridad y Recuperación ..... 190
Buenas Prácticas y Recomendaciones Finales ..... 195
Bibliografía ..... 196

# CAPÍTULO I INGENIERÍA DEL *KERNEL*

## Contenido del capítulo I

- ¿Qué es el *kernel*?
  - El rol del *kernel* en entornos virtualizados y contenedores (Docker/LXC).
  - Instalación y configuración de LXD y Docker
  - Ejercicios prácticos
- Modos de operación
  - Seguridad y anillos de ejecución
  - Ejercicio ¿cómo interceptar la comunicación entre el anillo 3 y el anillo 0
- Actualización del *kernel*
  - Gestión de parches de seguridad
- Configuración y compilación del *kernel*
  - Técnicas de Live Patching
- Distribuciones Linux

## ¿Qué es el *kernel*?

**Objetivo de aprendizaje.** El principal objetivo de aprendizaje en torno al núcleo o *kernel* es identificar su función crítica en el aislamiento de procesos, la cual constituye la base fundamental de tecnologías modernas como Docker y LXC. En lugar de limitarse a la gestión de *hardware*, el núcleo actual debe entenderse como el motor de orquestación lógica que permite segmentar un sistema en múltiples entornos independientes. Esta comprensión es vital porque el futuro del desarrollo de *software* ha dejado de residir en servidores físicos o máquinas virtuales pesadas para centrarse en contenedores eficientes que comparten el mismo *kernel*, maximizando así el rendimiento y la densidad de servicios.

Dominar la arquitectura del núcleo permite al profesional comprender cómo se garantiza la seguridad y la asignación de recursos en la nube mediante primitivas de aislamiento. Al interiorizar que una aplicación moderna es, en esencia, un proceso aislado por el *kernel*, el estudiante adquiere la capacidad de diseñar infraestructuras escalables, inmutables y portables. De este modo, el estudio del *kernel* se transforma en una herramienta estratégica para navegar la transición hacia un ecosistema donde la infraestructura se define por código y la eficiencia operativa depende totalmente de la robustez del núcleo que sostiene los contenedores.

El *kernel* es el núcleo del sistema operativo Linux que actúa como un puente entre el *hardware* del ordenador y el *software* y que siempre está residente en la memoria RAM. El *kernel* de Linux se encarga de gestionar y coordinar las actividades del sistema, tales como:

**Gestión de procesos:** Controla la creación, ejecución y finalización de los procesos, así como la asignación de tiempo de CPU a cada uno.

**Gestión de la memoria:** Administra la memoria del sistema, asignando y liberando espacio de memoria según sea necesario para los programas que se están ejecutando.

**Gestión de dispositivos:** Proporciona una interfaz para interactuar con el *hardware* del sistema, como discos duros, redes, periféricos, etc.

**Control de seguridad:** Implementa mecanismos de seguridad, como la gestión de permisos y el aislamiento de procesos, para proteger el sistema contra accesos no autorizados.

**Gestión de sistemas de archivos:** Permite el acceso a los archivos y directorios almacenados en los discos, gestionando la lectura y escritura de datos en los sistemas de archivos soportados.

> El *kernel* de Linux es modular y se puede personalizar según las necesidades específicas del usuario o del dispositivo. Es de código abierto, lo que significa que puede ser modificado y distribuido libremente por la comunidad. Existe un sitio web oficial https://www.kernel.org/ donde se desarrolla y distribuye el *kernel* de Linux, un repositorio central donde se alojan las versiones oficiales del *kernel*, incluyendo versiones estables, versiones en desarrollo, y otros recursos importantes para desarrolladores y usuarios avanzados (Linux Kernel Organization, 2024).

Para conocer qué versión del *kernel* maneja nuestro sistema operativo Linux, pruebe el siguiente comando y observe los resultados.

```
root@svrlinux:~$ uname -r ┘ Enter
```

```
root@svrlinux:~$ uname -msr ┘ Enter
```

Este comando, por ejemplo, y de acuerdo con lo que usted tenga instalado, desplegará la versión 6.8.0-51-generic en el primer caso y Linux 6.8.0-51-generic x86_64, respectivamente. En el segundo caso, se nos brinda una mayor información, como el nombre Linux y la arquitectura del sistema operativo de 64 bits.

**Linux 6.8.0:** Indica que está utilizando la versión 6.8.0 del *kernel* de Linux. Este número sigue un esquema de versionado que incluye la versión principal (6), la versión secundaria (8) y un número de parche o revisión (0). Cada una de estas versiones puede incluir nuevas características, correcciones de errores y mejoras de seguridad.

**51-generic:** Este sufijo adicional "51-generic" generalmente se refiere al número de compilación del *kernel* y al tipo de *kernel* que se está utilizando. "generic" indica que este *kernel* está destinado a ser generalizado para una amplia gama de *hardware*, en contraposición a versiones específicas como "*lowlatency*" (para baja latencia) o "*realtime*" (para sistemas en tiempo real).

X86_64: Este término describe la arquitectura de la CPU para la cual está compilado el *kernel*. "x86_64" indica que el *kernel* está diseñado para sistemas con arquitecturas de 64 bits, comúnmente utilizadas en la mayoría de las computadoras modernas.

Es importante tomar en cuenta que en algunas ocasiones nos encontraremos con otra nomenclatura, que consignamos a continuación, para así seleccionar el *kernel* más adecuado a nuestras necesidades.

**Mainline: 6.11-rc3**: "*Mainline*" se refiere a la versión de desarrollo más reciente del *kernel* de Linux. "6.11-rc3" indica que es la tercera versión candidata ("*gateway candidate*") para la versión 6.11 del *kernel*. Estas versiones están en fase de prueba y desarrollo y son utilizadas por desarrolladores y usuarios avanzados para probar las nuevas funcionalidades antes de que se liberen como versiones estables.

**Stable: 6.10.4**: "*Stable*" indica la versión actual del *kernel* que se considera lista para producción. "6.10.4" es una versión estable que ha pasado por varias pruebas y es adecuada para la mayoría de los usuarios. Estas versiones reciben actualizaciones menores para corregir errores y mejorar la seguridad.

**Longterm**: **6.6.4**: "*Longterm*" se refiere a una versión del *kernel* que recibe soporte y actualizaciones a largo plazo. "6.6.4" es la versión de mantenimiento más reciente para la serie 6.6 del *kernel*. Estas versiones son ideales para entornos de producción donde se requiere estabilidad a largo plazo, ya que reciben parches de seguridad y correcciones de errores durante un periodo prolongado.

Podemos encontrar versiones para la serie 6.x del *kernel* de Linux en la siguiente dirección URL ( https://mirrors.edge.kernel.org/pub/linux/kernel/v6.x/ ). La última versión estable al momento es la 6.13.3, que corresponde al archivo linux-6.13.3.tar.xz. En este capítulo se verá cómo instalar y preparar un *kernel* para nuestro sistema Linux Ubuntu partiendo de los recursos disponibles en el sitio web oficial o realizando la actualización directamente desde la terminal y así preparar nuestro sistema. También existen los **parches del** *kernel*, que no son más que archivos que contienen cambios específicos y actualizaciones que se aplican a una versión ya existente del kernel para corregir errores, añadir nuevas funcionalidades, mejorar el rendimiento, mejorar la seguridad o aplicar otras modificaciones. Cuando se libera una nueva versión del *kernel*, los desarrolladores pueden optar por liberar un parche en lugar de una versión completa del código fuente del *kernel*. Esto permite a los usuarios que ya tienen una versión anterior aplicar solo las diferencias (el parche) sin tener que descargar todo el código fuente nuevamente.

El *kernel* Linux de la versión 6.5 incluye mejoras en seguridad, estabilidad y soporte para nuevo *hardware*, compatibilidad con Wi-Fi 7, MIDI 2.0 en ALSA, con ACPI para arquitecturas RISC-V, Landlock para UML (User-Mode Linux) y mejoras para sistemas Zen de AMD. También HWE (Hardware Enablement) asegura que sistemas más antiguos o versiones LTS (Long-Term Support) de Ubuntu puedan aprovechar el soporte para dispositivos más recientes, como nuevos procesadores, tarjetas gráficas, y periféricos, sin necesidad de

actualizar completamente el S.O. En la actualidad, cuando instalamos Linux, generalmente la mayoría del *hardware* es reconocida por defecto, pero hay veces en que determinado *hardware* no es reconocido o alguna de sus funcionalidades está deshabilitada.

Cuando se compila o configura un *kernel* de Linux, este tiene la capacidad de seleccionar qué controladores y características estarán disponibles. Esto incluye soporte para dispositivos de *hardware* que pueden haber estado inhabilitados debido a la falta de un controlador o una funcionalidad específica. Por ejemplo, en una tarjeta de red 10/100/1000 Mbps, si quisieras habilitar el QoS (Quality of Service) para gestionar y priorizar el tráfico de red,y para que los datos críticos reciban un nivel adecuado de servicio, o para que una tarjeta TBS 6909X sea capaz de recibir canales de TV digital, hay una serie de características que desearíamos habilitar. Es aquí donde entra la configuración del *kernel* y luego la compilación, y para esto se utilizan una serie de comandos desde la terminal, los mismos que veremos más adelante en este mismo capítulo.

## El rol del *kernel* en entornos virtualizados y contenedores (Docker/LXC)

Históricamente, el *kernel* de Linux fue diseñado para ser el "director de orquesta" de una única máquina física. Sin embargo, en la era del *Cloud Computing*, la eficiencia es la prioridad. Ya no instalamos un sistema operativo completo para cada aplicación; en su lugar, utilizamos el *kernel* para crear "parcelas" aisladas. Aquí es donde el núcleo se convierte en el pilar fundamental de tecnologías como **Docker** y **LXC** (*Linux Containers*).

"Docker es una herramienta que promete encapsular fácilmente el proceso de creación de un artefacto distribuible para cualquier aplicación, implementarlo a escala en cualquier entorno y optimizar el flujo de trabajo y la capacidad de respuesta de las organizaciones de software ágiles" (Kane, 2018).Es decir, Docker se enfoca en contenedores ligeros para aplicaciones únicas y microservicios, mientras que LXC funciona como un "contenedor de sistema", emulando un sistema operativo completo o máquina virtual funcional.

Veamos ahora cinco casos en los cuales podamos evidenciar las diferencias entre estas tecnologías y también dónde y cómo podríamos utilizarlas.

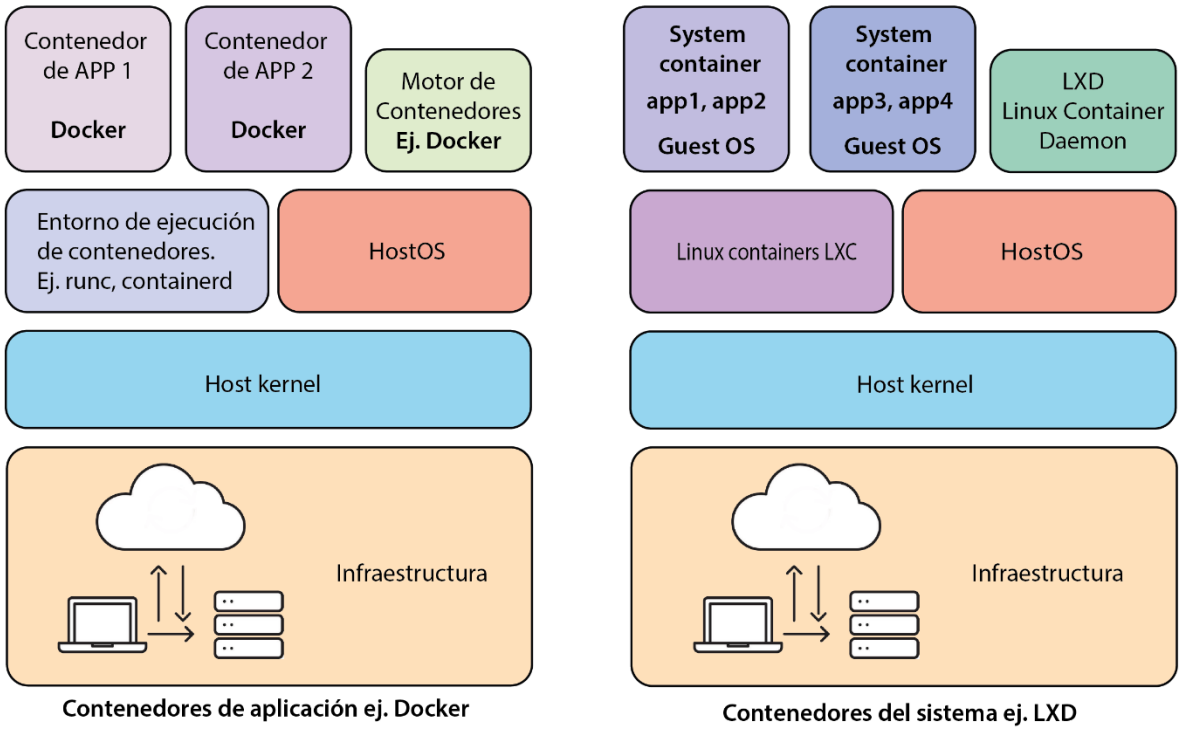



Figura 1. Comparación de la arquitectura LXC vs Docker

**1. En la arquitectura:** contenedores vs. virtualización tradicional

A diferencia de la virtualización clásica, que utiliza un hipervisor para emular *hardware* completo (lo que consume muchos recursos), los contenedores comparten el mismo *kernel* del host. Esta arquitectura permite ejecutar cientos de aplicaciones de forma independiente con un consumo mínimo de recursos, facilitando el escalado masivo en la nube.

Para lograr este aislamiento, el kernel utiliza dos mecanismos internos esenciales que todo administrador debe conocer:

**Namespaces**: Aíslan la visibilidad del sistema (procesos, red y archivos), haciendo que cada contenedor "crea" que es el único dueño del sistema.

**Cgroups (***Control Groups***):** Limitan el uso de *hardware* (CPU, RAM, E/S), evitando que un solo contenedor agote los recursos del servidor físico.

**LXC vs. Docker:** ¿Cuál elegir?

Aunque ambos utilizan las mismas funciones del kernel (*cgroups* y *namespaces*), su enfoque es distinto:

**LXC (***Linux Containers***):** Se centra en la virtualización de sistemas operativos. Funciona como un entorno de sistema completo donde se pueden instalar múltiples aplicaciones (web, correo, bases de datos) simultáneamente. Es ideal para infraestructuras persistentes y alojamiento privado, actuando como una "máquina virtual ligera".

**Docker:** Está optimizado para la contenerización de aplicaciones individuales. Cada contenedor suele ejecutar un único proceso o servicio. Es la herramienta estándar para el desarrollo ágil y despliegues rápidos (DevOps), gracias a su enfoque específico, ligero y altamente portable.

**2. En la persistencia de datos:** contenedores con estado (LXC) vs. sin estado (Docker)

La diferencia en el manejo de datos define el uso de cada tecnología:

**LXC (Con estado /** ***Stateful*****):** Estos contenedores funcionan como entidades completas con su propio sistema de archivos persistente. Cualquier cambio o actualización de datos se mantiene de forma nativa tras un reinicio, lo que los hace ideales para gestionar múltiples aplicaciones y servicios que requieren almacenamiento local permanente.

**Docker (Sin estado /** ***Stateless*****):** Diseñado para ser efímero; por defecto, los cambios realizados en un contenedor no persisten tras su reinicio. Aunque es posible conservar datos mediante el uso de volúmenes externos en el *host*, su verdadera potencia reside en el ciclo de vida de la aplicación: permite el seguimiento de versiones, comparación de cambios (*diffs*), confirmaciones (*commits*) y reversiones (*rollbacks*) rápidas.

Si se busca una solución única para administrar datos persistentes de diversos servicios, LXC es la opción robusta. Si se necesita despliegues ágiles con control de versiones y alta escalabilidad, Docker es la herramienta estándar.

**3. Arquitectura de servicios:** propósito y conectividad

La elección entre LXC y Docker depende de la complejidad y el aislamiento que requiera la arquitectura:

- **LXC (Sistemas multipropósito):** Permite ejecutar múltiples servicios (WordPress, MySQL, Apache) dentro de un mismo contenedor de forma sencilla. Es la opción más eficiente para pequeñas y medianas empresas que buscan consolidar su infraestructura en un entorno único, sin la complejidad de orquestar múltiples unidades.
- **Docker (Servicios de propósito único):** Se basa en el principio de un proceso por contenedor. Aunque configurar Docker para soportar aplicaciones múltiples puede resultar complejo, su ventaja radica en la inmutabilidad y el aislamiento total: es posible reiniciar el contenedor de la base de datos sin afectar al servidor web. Es la herramienta estándar para desarrolladores que necesitan portabilidad absoluta entre diferentes plataformas.

**Conectividad y acceso:** Es importante notar que, por diseño, los contenedores Docker están aislados de la red externa. Para ofrecer servicios al público, es necesario realizar un mapeo explícito de puertos (*Port Forwarding*), mientras que LXC facilita una integración más directa con la red del *host*, comportándose como un nodo adicional en la infraestructura.

**4. Independencia de la plataforma:** LXC constituye la tecnología base para segmentar un sistema en contenedores independientes, mientras que LXD actúa como el demonio de gestión que facilita la administración unificada y sencilla de dichas instancias. A diferencia de LXC puro, LXD ofrece un control avanzado sobre almacenamiento, redes y registros, operando mediante un sistema basado en imágenes que permite ejecutar cualquier carga de trabajo, incluso sistemas tradicionales. En términos funcionales, LXD se comporta de manera similar a hipervisores como VMware o KVM, con la ventaja estratégica de eliminar la sobrecarga de virtualización y reducir drásticamente el consumo de recursos de *hardware*. La diferencia en la arquitectura de ambas tecnologías impacta directamente en su movilidad.

**LXC (Nativo de Linux):** Al ser una tecnología diseñada específicamente para el núcleo de Linux, los contenedores LXC dependen estrechamente de las capacidades del *host* y no son fácilmente portables hacia otros sistemas operativos. Su uso es ideal para consolidar infraestructuras robustas de servicios comerciales dentro del ecosistema Linux.

**Docker (Multi-plataforma):** Gracias a Docker Engine, esta tecnología permite que las aplicaciones se ejecuten de forma consistente en casi cualquier entorno, incluyendo Linux, Windows y macOS. Esta capa de abstracción convierte a Docker en la solución estándar para el desarrollo y despliegue de aplicaciones que requieren ser probadas e implementadas en diversas plataformas y versiones de sistemas operativos.

**5. En la seguridad y aislamiento: el riesgo de contaminación cruzada**

Un vector común para la propagación de *malware* es la contaminación cruzada entre servicios. En este ámbito, la arquitectura de cada tecnología presenta desafíos distintos:

**LXC y la densidad de servicios:** Al alojar múltiples aplicaciones dentro de un mismo entorno, LXC presenta una superficie de riesgo compartida. Un compromiso de seguridad en una aplicación (por ejemplo, una vulnerabilidad en un sitio web) podría permitir que el *malware* realice un movimiento lateral hacia otros servicios del mismo contenedor, provocando caídas sistémicas. Sin embargo, este riesgo se puede mitigar significativamente mediante la implementación estricta de perfiles de seguridad como AppArmor o SELinux, que restringen las capacidades de cada proceso.

**Docker y el aislamiento por proceso:** Por el contrario, Docker minimiza este riesgo al ejecutar cada aplicación en su propio entorno aislado. Gracias al uso combinado de Namespaces (para el aislamiento de visibilidad) y SELinux (para el control de acceso obligatorio), se crea una barrera robusta que previene la contaminación entre sitios, asegurando que un incidente en una instancia no afecte al resto de la infraestructura.

## Instalación y configuración de LXD y Docker

Antes de pasar a la práctica en el mundo de los contenedores y el kernel, necesitamos montar un laboratorio de experimentación. Primeramente, hay que instalar una máquina virtual en Windows 11, que puede ser VirtualBox o VMware, y luego instalar Ubuntu Server 24.04. Se debe asignar al menos dos núcleos de CPU, 4GB de RAM y 20GB de disco duro.

Debemos asegurarnos de que el sistema esté al día con los comandos sudo apt update && sudo apt upgrade y que además tenga instaladas las herramientas de red necesarias (podemos revisar el capítulo 2 como guía para la instalación y actualización de Linux). También es necesario tener instalado en Windows un cliente SSH como BitVise client SSH o PuTTY para gestionar el servidor de forma profesional.

### Herramientas de red necesarias

Podemos tener instaladas previamente utilidades de monitoreo, redes y seguridad.

- Monitoreo: htop, iotop (para ver el uso de disco) y glances.
- Redes: net-tools, bridge-utils (importante para entender cómo LXC crea redes).
- Seguridad: apparmor-utils y selinux-utils.

### Instalación de herramientas de monitoreo

El comando para instalar la suite de herramientas de monitoreo es el siguiente:

```
root@svrlinux:~$ sudo apt update && sudo apt install -y htop iotop glances sysstat neofetch ┘ Enter
```

### Explicación de los comandos

Es importante detallar la utilizad de cada uno:

**A. htop (Monitoreo de procesos y CPU)**

Es la versión evolucionada de top. Permite ver visualmente cómo el *kernel* reparte la carga entre los núcleos del procesador.

Observa cómo sube la carga cuando lanza 50 contenedores Docker al mismo tiempo.

**B. iotop (Monitoreo de disco/escritura)**

Implica cómo el *kernel* gestiona la entrada/salida (E/S). Esta herramienta presenta qué proceso o contenedor está "estresando" el disco duro.

```
root@svrlinux:~$ sudo iotop
```

Ideal para demostrar la diferencia de escritura entre un contenedor LXC (que escribe mucho en su sistema de archivos persistente) y uno de Docker.

**C. glances (El "todo en uno")**

Es una herramienta de monitoreo avanzada que muestra CPU, Memoria, Red y Disco en una sola pantalla. Lo mejor es que tiene una sección específica para Docker.

```
root@svrlinux:~$ glances
```

Si se presiona la tecla D dentro de *glances*, se mostrará el consumo individual de cada contenedor.

**D. sysstat (iostat, mpstat, sar)**

No es un programa visual, sino un conjunto de utilidades de bajo nivel que recolectan estadísticas del *kernel*.

```
root@svrlinux:~$ iostat -xz 1 (para ver la latencia del disco en tiempo real).
```

**E. neofetch (Información del sistema)**

Muestra la versión exacta del *kernel* que estás usando, el tiempo de encendido (*uptime*) y la arquitectura.

```
root@svrlinux:~$ sudo neofetch
```

Para ver el "corazón" del kernel funcionando, podemos utilizar el siguiente comando que no requiere instalación (viene en el sistema de archivos /proc):

```
root@svrlinux:~$ watch -n 1 cat /proc/interrupts
```

Este comando muestra cuántas veces por segundo el *hardware* le "habla" al *kernel*. Es fascinante ver cómo cambian los números al mover el ratón o al haber tráfico de red.

**Instalación de herramientas de red**

Ejecute este comando en la terminal:

```
root@svrlinux:~$ sudo apt update && sudo apt install -y net-tools bridge-utils
root@svrlinux:~$ sudo apt install -y debootstrap build-essential
```

Con debootstrap, podrá crear sus propios entornos aislados o "mini sistemas operativos" dentro de carpetas.

Con build-essential, su servidor tiene la capacidad de compilar el *kernel* y transformar el código fuente (texto) en un programa binario ejecutable.

A continuación, se describe cómo utilizar cada una de estas herramientas para entender la red de LXC y Docker:

**A. net-tools (para el clásico *ifconfig*)**

Aunque el comando moderno es ip addr, net-tools sigue siendo fundamental para *scripts* y para una visualización rápida de las interfaces.

```
root@svrlinux:~$ sudo ifconfig
```

Utilidad en el kernel: Permite ver las interfaces virtuales que crea el Kernel. Al instalar LXD, verá una interfaz llamada lxdbr0.

**Qué observar:** busque interfaces que empiecen por veth (Virtual Ethernet). Cada vez que lance un contenedor, el kernel crea un "cable virtual" donde un extremo está dentro del contenedor y el otro en su *host*.

**B. bridge-utils (Gestión de puentes/*bridges*)**

Esta es la herramienta más importante para entender LXC. Un "*bridge*" funciona como un switch virtual dentro de su *kernel*.

```
root@svrlinux:~$ sudo brctl show
```

Utilidad en el Kernel: Muestra cómo el *kernel* está conectando las interfaces físicas a las interfaces virtuales de los contenedores.

**Qué observar:** se verá el bridge lxdbr0. En la columna "interfaces", aparecerán los IDs de las interfaces de cada contenedor que esté conectado a esa red.

**Instalación de herramientas de seguridad**

Para fortalecer la seguridad de su laboratorio y entender cómo el *kernel* restringe lo que un contenedor puede o no hacer, necesita instalar las herramientas de gestión de AppArmor y SELinux. En Ubuntu 24.04, AppArmor es el sistema de seguridad por defecto, mientras que SELinux es una alternativa (común en Red Hat/CentOS) que también puede explorar.

1. Instalación de las herramientas

Ejecute el siguiente comando desde su terminal para instalar ambos paquetes:

```
root@svrlinux:~$ sudo apt update && sudo apt install -y apparmor-utils selinux-utils
```

2. ¿Para qué sirven estas herramientas en el *kernel*?

En su libro, puede explicar que estas utilidades implementan el Control de Acceso Obligatorio (MAC). A diferencia de los permisos estándar (donde usted, como dueño de un archivo, decide quién lo lee), aquí el *kernel* tiene la última palabra basándose en políticas.

A. AppArmor (el estándar en Ubuntu)

Es el que protege los contenedores LXC y Docker por defecto.

```
root@svrlinux:~$ sudo aa-status
```

**Utilidad:** Muestra cuántos perfiles de seguridad están activos y qué procesos (como los de Docker) están bajo "confinamiento". Si un proceso intenta acceder a un archivo fuera de su lista permitida, el *kernel* lo bloquea inmediatamente.

Observará que existen perfiles específicos como docker-default o perfiles para lxc. Esto significa que el *kernel* tiene una "lista negra" y "lista blanca" de lo que esos contenedores pueden hacer.

B. SELinux (Security-Enhanced Linux)

Aunque no es el nativo de Ubuntu, sus utilidades permiten consultar contextos de seguridad.

Comando clave: getenforce

**Utilidad:** Verifica si el sistema de seguridad está en modo *Enforcing* (bloqueando), *Permissive* (solo avisando) o *Disabled*.

Ver el contexto de seguridad de un proceso: ejecuta un contenedor y luego use el siguiente comando para ver cómo el Kernel lo identifica.

```
root@svrlinux:~$ sudo ps -eZ | grep docker
```

La columna de la izquierda le mostrará la etiqueta de seguridad. Es el "DNI" que el *kernel* revisa antes de permitir que el proceso toque cualquier recurso del *hardware*.

Es importante mencionar que no se recomienda tener AppArmor y SELinux activos y bloqueando al mismo tiempo, ya que pueden entrar en conflicto. En Ubuntu, lo normal es usar las utilidades de SELinux solo para análisis o si decide desactivar AppArmor por completo (lo cual no se recomienda en producción).

**Configuración de LXD (Contenedores de sistema)**

LXD viene preinstalado en Ubuntu vía Snap, pero necesita ser inicializado para configurar la red y el almacenamiento.

Paso 1: Inicialización. Ejecute el siguiente comando y, para fines de práctica, puede presionar Enter a todas las opciones por defecto (esto configurará un almacenamiento tipo zfs o dir y una red puente lxdbr0).

```
root@svrlinux:~$ sudo lxd init
```

Paso 2: Lanzamiento de su primer contenedor LXC Vamos a crear un contenedor con Ubuntu 24.04 llamado "servidor-web":

```
root@svrlinux:~$ lxc launch images:ubuntu/24.04 servidor-web
```

Paso 3: Comandos de gestión básicos

- Ver estado: lxc list
- Entrar a la terminal del contenedor: lxc shell servidor-web
- Detenerlo: lxc stop servidor-web

**Configuración de Docker (Contenedores de aplicación)**

Para Ubuntu 24.04, lo más profesional es usar el repositorio oficial de Docker para asegurar estabilidad y actualizaciones de seguridad.

Paso 1: Instalar dependencias y llave GPG

```
root@svrlinux:~$ sudo apt install -y ca-certificates curl gnupg

root@svrlinux:~$ sudo install -m 0755 -d /etc/apt/keyrings

root@svrlinux:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

root@svrlinux:~$ sudo chmod a+r /etc/apt/keyrings/docker.gpg
```

Paso 2: Agregar el repositorio oficial

```
root@svrlinux:~$  echo \

  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \

  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \

root@svrlinux:~$ sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

root@svrlinux:~$ sudo apt update
```

Paso 3: Instalar el motor de Docker

```
root@svrlinux:~$ sudo apt install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

Paso 4: Gestionar Docker sin 'sudo' (ppcional pero recomendado) Para que no sea necesario escribir sudo en cada comando de Docker, se puede hacer lo siguiente:

```
root@svrlinux:~$ sudo usermod -aG docker $USER

root@svrlinux:~$ newgrp Docker
```

**Verificación de la coexistencia**

Ahora tiene ambas tecnologías corriendo sobre el mismo kernel. Puede comprobarlo viendo cómo se han creado interfaces de red distintas para cada uno:

```
root@svrlinux:~$ ip addr show | grep -E 'lxdbr0|docker0'
```

## Ejercicios prácticos

En LXC: Entre al contenedor (lxc shell servidor-web), cree un archivo en /root/hola.txt y salga del contenedor. Reinicie el contenedor y verifique que el archivo sigue ahí (Persistencia/*Stateful*).

En Docker: Corra un contenedor temporal: docker run -it --name prueba-docker ubuntu:24.04 bash. Crear un archivo en /root/hola.txt y salir con exit. Borrar el contenedor con docker rm pruebe-docker, vuelva a correrlo y verá que el archivo ha desaparecido (Inmutabilidad/*Stateless*).

**Ejercicio práctico de observación (El "camino de los datos")**

**Probar el siguiente flujo:**

1. Antes de encender el contenedor: ejecute brctl show. Verá el puente lxdbr0 vacío.

2. Lance un contenedor LXC:

lxc launch images:ubuntu/24.04 prueba-red

3.Después de encenderlo: ejecute de nuevo brctl show.

- Verá que ahora existe una interfaz (ej: vethxxxx) asociada a lxdbr0.
- Explicación técnica: el *kernel* Linux acaba de crear un "Switch Virtual" y ha "enchufado" el cable de red del contenedor en ese puerto.

4.Ver la tabla de rutas:

route -n

Esto muestra al *kernel* hacia dónde enviar los paquetes que salen de los contenedores para que lleguen a Internet.

## Modos de operación

**Objetivo de aprendizaje.** El objetivo central de este apartado es diferenciar con precisión los niveles de privilegio del procesador para comprender cómo el *hardware* se protege de fallos o accesos no autorizados por parte de aplicaciones externas. Esta distinción entre el modo usuario y el modo *kernel* representa la piedra angular de la ciberseguridad contemporánea, ya que establece una frontera infranqueable que impide que un proceso comprometido afecte la integridad total del sistema. Al profundizar en el funcionamiento del modo *kernel*, el profesional no solo entiende la arquitectura del sistema operativo, sino que adquiere la capacidad crítica de identificar, mitigar y prevenir ataques de escalada de privilegios, donde un actor malintencionado intenta saltar estas barreras de protección para tomar el control absoluto del *hardware* y los datos.

Debe tener claro que "la mayoría de las computadoras tienen dos modos de operación: modo *kernel* y modo de usuario. El sistema operativo, la pieza de *software* más importante, se ejecuta en modo *kernel* (también llamado modo supervisor) para al menos algunas de sus funcionalidades. En este modo, se tiene acceso completo a todo el *hardware* y se puede ejecutar cualquier instrucción que la máquina sea capaz de ejecutar. El resto del *software* se ejecuta en modo de usuario, en el que solo está disponible un subconjunto de las instrucciones de la máquina. En particular, aquellas instrucciones que afectan el control de la máquina determinan los límites de seguridad o realizan I/O (Entrada/Salida) están prohibidas para los programas en modo de usuario" (Tanenbaum & Bos, 2023, pág. 2).

No se debe confundir el modo supervisor con el modo superusuario. El **modo** *kernel* **(modo supervisor)** es un término que describe cómo el sistema operativo interactúa con el *hardware* en un nivel profundo, mientras que el **modo superusuario** se refiere a los permisos de un usuario para realizar tareas administrativas en el sistema operativo. Conocer la diferencia entre modo *kernel* y modo de usuario es particularmente importante para mantener la seguridad, la estabilidad y el control de los recursos en un sistema operativo moderno.

Por otro lado, para fines prácticos, a lo largo de este documento se omitirá el comando sudo y utilizaremos directamente el prompt del Shell #, que es cuando se ejecuta como usuario root o superusuario. El prompt del Shell # es una indicación visual que nos muestra que el usuario actual tiene privilegios administrativos completos sobre el sistema. En Linux Ubuntu siempre se inicia la sesión como un usuario normal, para cambiar al estado de administrador ingresamos el comando "sudo su" seguido de la clave que se registró al momento de instalar el sistema operativo (ver capítulo 2).

```
root@svrlinux:~$ sudo su ┘ Enter
```

Cuando se observa, el símbolo **#** indica que se está utilizando una sesión de terminal o consola con privilegios elevados. El usuario que está ejecutando comandos en este *prompt* tiene acceso completo al sistema y puede realizar cambios críticos, como instalar *software*, modificar configuraciones del sistema y administrar usuarios.

A diferencia de **#,** el símbolo de dólar $ representa un *prompt* regular en Linux y generalmente indica que se está utilizando una sesión de terminal como un usuario normal. Este usuario tiene permisos limitados y

no puede realizar acciones que afecten al sistema en su totalidad, como la instalación de *software* global o la modificación de archivos de configuración críticos.

Una vez estamos como usuario administrador, podemos volver al usuario de estado normal. Esta es una acción recomendable sobre todo si somos administradores del sistema por motivos de seguridad, ya que, si algún intruso quisiera instalar una aplicación como usuario normal, el sistema le presentará un mensaje de permiso denegado. El comando simplemente es como se indica a continuación.

```
root@svrlinux:~# exit ┘ Enter

exit

root@svrlinux:~$
```

Si volvemos a ingresar el comando exit una segunda vez, volveremos al inicio de sesión y, como política de seguridad, sobre todo en entornos de producción, es mejor estandarizar esta práctica.

```
root@svrlinux:~$ exit ┘ Enter


Ubuntu 24.04 LTS svrlinux tty1

svrlinux login:
```

Internamente los servicios del sistema seguirán ejecutándose. Los servicios en Linux generalmente se ejecutan como demonios (*daemons*), lo que significa que son procesos que se ejecutan en segundo plano independientemente de la sesión del usuario. Estos servicios se administran mediante el sistema de inicio, como **systemd** en muchas distribuciones modernas de Linux.

En general, los demonios (*daemons*) en Linux son procesos que se ejecutan en segundo plano para proporcionar servicios al usuario o al sistema. Su propósito es realizar actividades automáticas y mantener servicios importantes en ejecución, ejemplos comunes de demonios en Linux tenemos: los **demonios de red**, como sshd para manejar conexiones seguras y httpd para manejar servicios web que manejan solicitudes HTTP; **demonios de servicios del sistema**, como cron, syslogd o journald encargados de gestionar registros logs del sistema; demonios de impresión y demonios del *kernel*. Los demonios suelen estar controlados por el sistema de inicialización, como *systemd*, *init* o *upstart*, dependiendo de la distribución Linux. En el caso de Ubuntu, por systemd, aunque, si, por ejemplo, en alguna distribución se ejecuta en la terminal el comando init 6, se reiniciará el sistema operativo o si se ejecuta init 0, se apagará el sistema. Los demonios del *kernel* son creados y administrados por el propio núcleo, con sus respectivas funciones.

Cuando se inicia sesión, solo se detendrán los procesos específicos de esa sesión, no los servicios del sistema que están configurados para ejecutarse en segundo plano. Por lo tanto, los servicios críticos del sistema como servidores web, bases de datos, servidores de correo, etc., continuarán funcionando normalmente a pesar de que se desconecte o cierre la sesión.

En general, un sistema informático puede dividirse aproximadamente en cuatro componentes:

- *Hardware*
- Sistema operativo
- Programas de aplicación
- Usuario

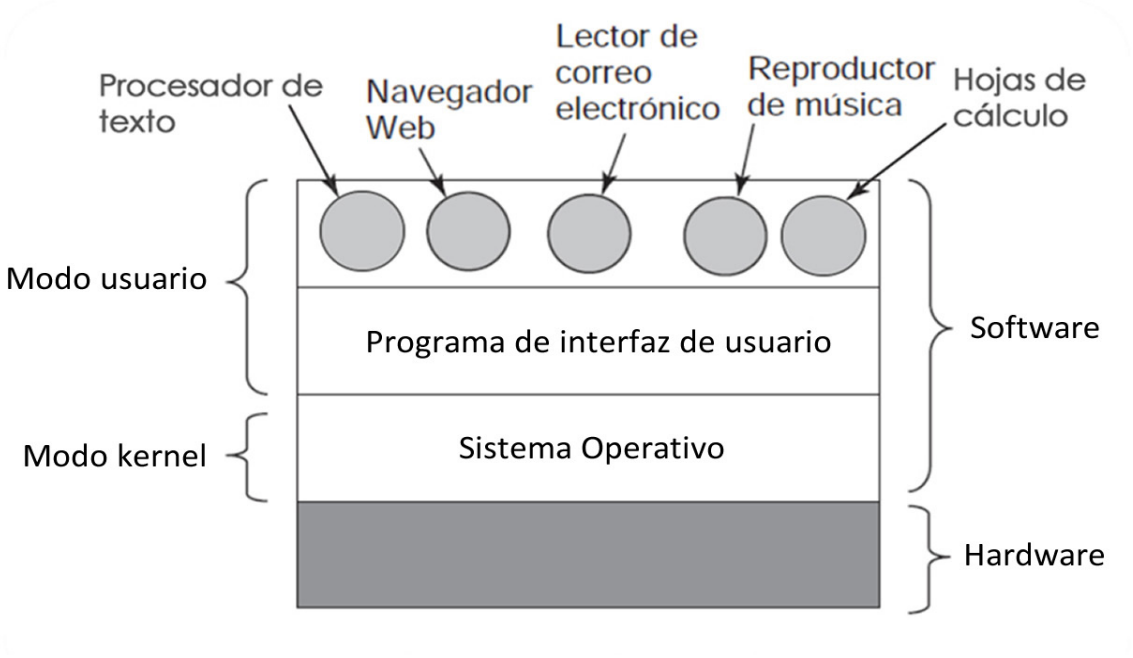



Figura 2. Ubicación del sistema operativo.

Fuente: Tanenbaum, A. S., & Bos, H. (2023). *Modern operating systems* (5th ed., p. 45).

Esta figura nos indica donde encaja el sistema operativo dentro de los cuatro elementos principales que trabajan en conjunto para realizar diversas tareas. El *hardware* representa la base física que incluye componentes como la CPU, memoria, y dispositivos de almacenamiento. El sistema operativo actúa como un intermediario entre el *hardware* y el usuario, gestionando los recursos y permitiendo que los programas de aplicación se ejecuten de manera eficiente.

Los programas de aplicación son el *software* que los usuarios utilizan para realizar tareas específicas, como procesadores de texto, hojas de cálculo, navegadores web o un reproductor de música. Finalmente, el usuario es la persona que interactúa con el sistema para llevar a cabo diversas actividades, beneficiándose de la combinación de los otros tres componentes. Como se mencionaba anteriormente, el *kernel* es la parte central de cualquier sistema operativo Linux y se encarga de manejar las interacciones entre el *hardware* del sistema y el *software*, gestionando recursos como la memoria, los procesos y la comunicación entre dispositivos.

Por lo tanto, una terminal de Linux se ejecutaría en **modo usuario**. Entonces, cuando utilice una terminal, estará interactuando con el sistema operativo a través de programas y comandos que se ejecutan en este modo, lo que significa que no se podrá realizar operaciones que interfieran directamente con el núcleo del sistema o acceder directamente al *hardware*. Por otro lado, el modo *kernel* al ser utilizado por el núcleo del sistema operativo (*kernel*) permitirá un control completo sobre el hardware del sistema. Es el nivel de privilegio más alto, donde se realizan tareas críticas como la gestión de la memoria, procesos y dispositivos.

A continuación, presentamos un uso básico del comando **systemctl**. Está es una herramienta que interactúa con **systemd** para controlar y gestionar servicios del sistema.

Para ver el estado de un servicio:

```
root@svrlinux:~# systemctl status nombre_del_servicio
```

Para iniciar un servicio:

```
root@svrlinux:~# systemctl start nombre_del_servicio
```

Para detener un servicio:

```
root@svrlinux:~# systemctl stop nombre_del_servicio
```

Para reiniciar un servicio:

root@svrlinux:~# systemctl restart nombre_del_servicio

Para habilitar un servicio y que inicie al arrancar:

root@svrlinux:~# systemctl enable nombre_del_servicio

Para habilitar un servicio y que inicie al arrancar:

root@svrlinux:~# systemctl disable nombre_del_servicio

## Seguridad y anillos de ejecución

Para entender la seguridad en la arquitectura de computadoras, imagine que el sistema operativo es un castillo con diferentes niveles de seguridad. Los anillos de ejecución (o Protection Rings) son niveles de privilegio que el *hardware* (la CPU) impone para proteger al sistema de errores o ataques.

Seguridad y anillos de ejecución (privilegios de la CPU). La CPU no permite que cualquier programa haga lo que quiera. Para mantener el orden, divide los privilegios en niveles, siendo el Anillo 0 y el Anillo 3 los más importantes en sistemas como Linux:

**1. Anillo 0 (Modo *kernel* / El núcleo del castillo)**

Es el nivel con máximo privilegio. Aquí es donde vive el *kernel* de Linux.

**Capacidades:** Tiene acceso total y directo al *hardware* (memoria RAM, disco duro, CPU, dispositivos).

**Riesgo:** Si un código falla en el Anillo 0, todo el sistema se colapsa (aparece el famoso *Kernel Panic*).

**Seguridad:** Solo el código más confiable del sistema operativo corre aquí.

**2. Anillo 3 (Modo Usuario / Los alrededores del castillo)**

Es el nivel con mínimo privilegio. Aquí es donde corren sus aplicaciones: el navegador, editor de texto, Docker y los procesos dentro de sus contenedores.

**Capacidades:** Las aplicaciones están "encerradas". No pueden tocar el hardware directamente.

**Seguridad:** Si su navegador falla en el Anillo 3, solo se cierra la aplicación; el resto del sistema sigue intacto.

**¿Cómo se comunican? (Llamadas al sistema)**

Si una aplicación en el Anillo 3 necesita algo del *hardware* (por ejemplo, guardar un archivo en el disco), no puede hacerlo solo. Debe tocar la puerta del Anillo 0 mediante una Syscall (Llamada al sistema). El *kernel* revisa si la aplicación tiene permiso y, si es así, realiza la tarea en su nombre.

**Relación con contenedores y seguridad:**

Docker y LXC: Ambos corren procesos en el Anillo 3. El aislamiento que logran (Namespaces y Cgroups) es una capa lógica adicional, pero físicamente siguen limitados por el *hardware* para no interferir con el Anillo 0.

Virtualización clásica (KVM/VMware): Utiliza un nivel especial (a veces llamado Anillo -1) donde el hipervisor controla a varios "*kernel* invitados", añadiendo una capa de seguridad física extra.

**Diagrama de flujo: el camino de una Syscall**

```
graph TD
    subgraph "ANILLO 3: Modo Usuario (Restringido)"
        A[Aplicación en Contenedor<br/>Ej: Servidor Web Nginx] -->|1. Necesita escribir en Disco/Red| B(Genera una SYSCALL)
    end

    subgraph "BARRERA DE SEGURIDAD (Kernel Check)"
        B --> C{¿Permitido por el Kernel?}
        C -->|No| D[Acceso Denegado<br/>Segmentation Fault / Permission Denied]
        C -->|Sí| E[Namespaces: ¿Es su recurso?]
        E --> F[Cgroups: ¿Tiene cuota disponible?]
        F --> G[AppArmor/SELinux: ¿Tiene permiso?]
    end

    subgraph "ANILLO 0: Modo Kernel (Privilegiado)"
        G -->|Cambio de Contexto| H[Manejador de Syscalls del Kernel]
        H --> I[Driver del Dispositivo<br/>Código que habla con el Hardware]
    end

    subgraph "HARDWARE (Nivel Físico)"
        I --> J[Disco SSD / Tarjeta de Red / RAM]
    end

    J -->|2. Respuesta enviada de vuelta| I
    I -->|Retorno de Datos| A
```

Explicación de los componentes:

1. Anillo 3 (Espacio de Usuario): Es donde "viven" tus contenedores. Es un entorno seguro porque, si la aplicación falla, no puede afectar a la memoria de otras aplicaciones ni al *kernel*.
2. La Syscall (la frontera): Es la única forma legal que tiene una aplicación de pedirle algo al *hardware*. Es como un formulario que se entrega en una ventanilla.
3. Filtros de contenedores (Namespaces/Cgroups): Esta es la magia de LXC y Docker. El *kernel* actúa como un recepcionista que verifica: "¿Este contenedor tiene permiso para ver esta carpeta?" (Namespaces) o "¿Ya gastó mucha CPU?" (Cgroups).
4. Anillo 0 (Espacio de Kernel): Una vez que el *kernel* confía en la petición, "se pone los guantes" y manipula directamente los chips y cables del *hardware*.
5. *Hardware*: El destino final donde se guardan o envían los datos.

"Un contenedor es como un prisionero en una celda de cristal (Anillo 3). Puede ver el mundo, pero para tocar cualquier cosa afuera (*hardware*), debe pedírselo al guardia (*kernel* en Anillo 0) a través de un pequeño hueco en la pared (Syscall). El guardia decide si le hace el favor o no basándose en las reglas (seguridad)".

## Ejercicio: ¿cómo interceptar la comunicación entre el anillo 3 y el anillo 0?

Vamos a usar *strace* para "interceptar" la comunicación entre el **Anillo 3** y el **Anillo 0**. Esta es una de las herramientas más potentes para un administrador de sistemas; es como poner un estetoscopio en el corazón del sistema operativo.

1. Instalación de *strace*

Primero, asegurarse de tener la herramienta instalada en tu Ubuntu:

```
root@svrlinux:~# apt update && sudo apt install -y strace
```

2. **El experimento:** ¿qué hace el comando ls en realidad?

Cuando se ejecuta un simple ls, parece algo instantáneo, pero el *kernel* tiene que trabajar mucho para mostrar esos archivos. Vamos a capturar esas llamadas:

Ejecute este comando:

```
root@svrlinux:~# strace ls
```

**Lo que verá en pantalla (análisis del flujo):** Una catarata de texto. Cada línea es una **Syscall**. Fíjese en estas tres, que son vitales:

- execve(...): Es el *kernel* creando el proceso para el comando ls.
- openat(...): Es el proceso pidiendo permiso al **Anillo 0** para abrir la carpeta actual.
- write(1, "archivo1.txt"...): Es el kernel enviando los nombres de los archivos a tu pantalla (el *hardware* de salida).

**3. Nivel pro: espiando un contenedor Docker en vivo**

Ahora vamos a lo que realmente importa: observe cómo un contenedor habla con el *kernel* del *host*.

1. **Lanzar un contenedor de prueba que se quede "durmiendo":**
   ```
   root@svrlinux:~# docker run -d --name espia-kernel ubuntu:24.04 sleep 1000
   ```
2. **Buscar el ID del proceso (PID) en el *host*:** Como los contenedores comparten el kernel, el proceso *sleep* tiene un número de identificación en tu sistema principal:
   ```
   root@svrlinux:~# ps aux | grep "sleep 1000"
   ```
   *(Anotar el número que aparece en la segunda columna, por ejemplo: 1234).*
3. **¡Capturar las Syscalls!** Ahora enganche *strace* a ese proceso que vive "dentro" del contenedor:
   ```
   root@svrlinux:~# sudo strace -p 1234
   ```

**¿Qué observará?** Verá que el proceso está en una llamada titulada *restart_syscall* o *nanosleep*. El proceso está literalmente suspendido y el *kernel* (Anillo 0) está contando el tiempo por él. Si detiene el contenedor desde otra terminal, verá cómo aparecen llamadas de "cierre" instantáneamente.

## Actualización del *kernel* en Ubuntu

**Objetivo de aprendizaje.** El objetivo fundamental de esta subsección es capacitar al administrador en la ejecución de actualizaciones críticas de seguridad mediante técnicas de **Live Patching**, garantizando la continuidad operativa absoluta y el cumplimiento de la filosofía de cero tiempos de inactividad. En un entorno global que exige disponibilidad 24/7, el dominio de la gestión de parches en caliente permite mitigar vulnerabilidades críticas en el *kernel* sin recurrir a reinicios costosos que interrumpen la cadena de suministro digital. Comprender esta tecnología no solo representa una ventaja táctica en ciberseguridad, sino que se

traduce en un valor estratégico para la organización al ahorrar miles de dólares en pérdidas operativas y asegurar que los servicios críticos permanezcan resilientes ante amenazas emergentes y sin sacrificar la estabilidad del sistema.

En el capítulo 2 veremos el proceso de instalación de Linux Ubuntu Server en su última versión.

Luego de instalar el sistema operativo Linux Ubuntu, el primer comando por ejecutar es apt update, apt significa (*advanced package tool*) herramienta avanzada de paquetes.

```
root@svrlinux:~# apt update ┘ Enter
```

Este comando permite actualizar la lista de paquetes disponibles y sus versiones, es decir, simplemente actualiza los repositorios que definimos en el archivo /etc/apt/sources.list. Se actualizan los listados de paquetes disponibles en las fuentes definidas en el archivo sources.list, pero no se instalan paquetes. Es importante sobre todo si ha pasado algún tiempo de la instalación de Linux ejecutar estos dos comandos "update" y/o "upgrade", "full-upgrade" antes de instalar algún paquete.

La diferencia entre *upgrade* y *full-upgrade* (actualización completa) es que *full-upgrade* actualiza aplicaciones, herramientas y utilidades, incluyendo el *kernel* del sistema operativo, elimina paquetes antiguos si es necesario para la actualización, también resuelve conflictos en las actualizaciones de dependencias.

Mientras que *upgrade* actualiza los paquetes disponibles, pero no todos, descarga y actualiza paquetes, excluyendo componentes importantes como el *kernel* de Linux y evita actualizar paquetes que se requieran eliminar debido a cambios en las dependencias.

El siguiente comando se utiliza para actualizar todos los paquetes instalados a las versiones más recientes disponibles en los repositorios configurados, siempre y cuando no requieran la instalación de nuevos paquetes ni la eliminación de otros ya existentes.

```
root@svrlinux:~# apt full-upgrade ┘ Enter
```

Al ejecutar este último comando se indica en pantalla lo siguiente:

```
root@svrlinux:/home/cesarg# apt full-upgrade
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Calculando la actualización... Hecho
Se instalarán los siguientes paquetes NUEVOS:
  linux-headers-6.8.0-40 linux-headers-6.8.0-40-generic linux-image-6.8.0-40-generic linux-modules-6.8.0-40-generic linux-modules-extra-6.8.0-40-generic
  linux-tools-6.8.0-40 linux-tools-6.8.0-40-generic
Se actualizarán los siguientes paquetes:
  apparmor apport apport-core-dump-handler curl dracut-install krb5-locales landscape-common libapparmor1 libcurl3t64-gnutls libcurl4t64 libgssapi-krb5-2
  libk5crypto3 libkrb5-3 libkrb5support0 libnss-systemd libopeniscsiusr libpam-systemd libsystemd-shared libsystemd0 libudev1 linux-generic
  linux-headers-generic linux-image-generic linux-libc-dev linux-tools-common needrestart open-iscsi python3-apport python3-problem-report systemd systemd-dev
  systemd-resolved systemd-sysv systemd-timesyncd thin-provisioning-tools udev xkb-data
37 actualizados, 7 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
13 standard LTS security updates
Se necesita descargar 16,1 MB/204 MB de archivos.
Se utilizarán 299 MB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] _
```

Figura 3. Actualización del *kernel* desde el terminal con el comando *apt update* y seguidamente *apt full-upgrade*

En la figura se puede apreciar que se instalará la versión del *kernel* 6.8.40-generic y otros paquetes adicionales. Luego de la instalación procederemos a reiniciar el sistema para que la actualización se vea reflejada. Podemos reiniciar el sistema con cualquiera de los siguientes comandos y confirmar la nueva versión con uname -msr

```
root@svrlinux:~# init 6 ┘ Enter

root@svrlinux:~# shutdown -r now ┘ Enter

root@svrlinux:~# init 0 ┘ Enter
```

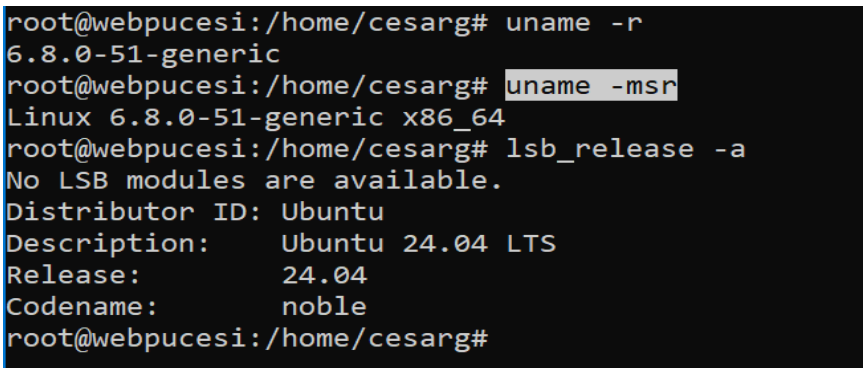



Figura 4. *Versión actualizada del kernel 6.8.0-51-generic*

También el comando lsb_release -a nos presenta la versión de Linux utilizada. Igualmente, se pueden comprobar más detalles, como el nombre, la versión, errores (bugs) reportados, etc. del sistema operativo con el siguiente comando.

```
root@svrlinux:~# cat /etc/os-release ↵ Enter
```

El comando **cat** lo que hace es presentar en la terminal el contenido del archivo al que se hace referencia, en este caso el archivo "os-release" que se encuentra dentro del directorio /usr/lib/os-release.

Dentro del directorio /etc/ lo que se encuentra específicamente es un enlace al archivo como @os-release, puede verificar si desde la terminal ejecuta el comando ls -l dentro del directorio /etc/.

```
root@svrlinux:/etc# ls -l | more ↵ Enter
```

Generalmente, dentro del directorio /etc/ encontraremos archivos de configuración de diferentes servicios, como por ejemplo de una base de datos, un servidor web, de una red, un servicio ftp, etc., que pueden estar dentro de otros subdirectorios, pero siempre bajo /etc/. Cabe recordar que en Linux las configuraciones se manejan a través de archivos. Se puede observar el símbolo "|" seguido de "more", este símbolo se llama "pipe o tubería" y procede de la palabra inglesa "*pipeline*" cuya traducción es tubería: lo que hace es unir dos procesos o serie de etapas en los que una salida de un proceso se convierte en la entrada de la siguiente. Por ejemplo, la salida del comando **ls** (listar archivos) se convierte en la entrada del comando "more", que es detener la pantalla en caso de que sea muy extensa. Otro ejemplo puede ser el siguiente:

```
root@svrlinux:/etc# ls -l | grep .conf ↵ Enter
```

donde la salida del comando "ls" se convierte en la entrada del comando "grep" para buscar archivos con extensión .conf

En Ubuntu, el directorio /usr/src/es donde generalmente se encuentra el código fuente del *kernel* y módulos relacionados, esto es importante para que los desarrolladores puedan compilar su propio *kernel* o activar ciertos módulos. Para poder descargar el código fuente del *kernel* que se desea configurar y compilar, puede realizarse desde el sitio oficial kernel.org o usando el comando apt para obtener el código fuente:

```
root@svrlinux:~# apt-get source linux-source ↵ Enter
```

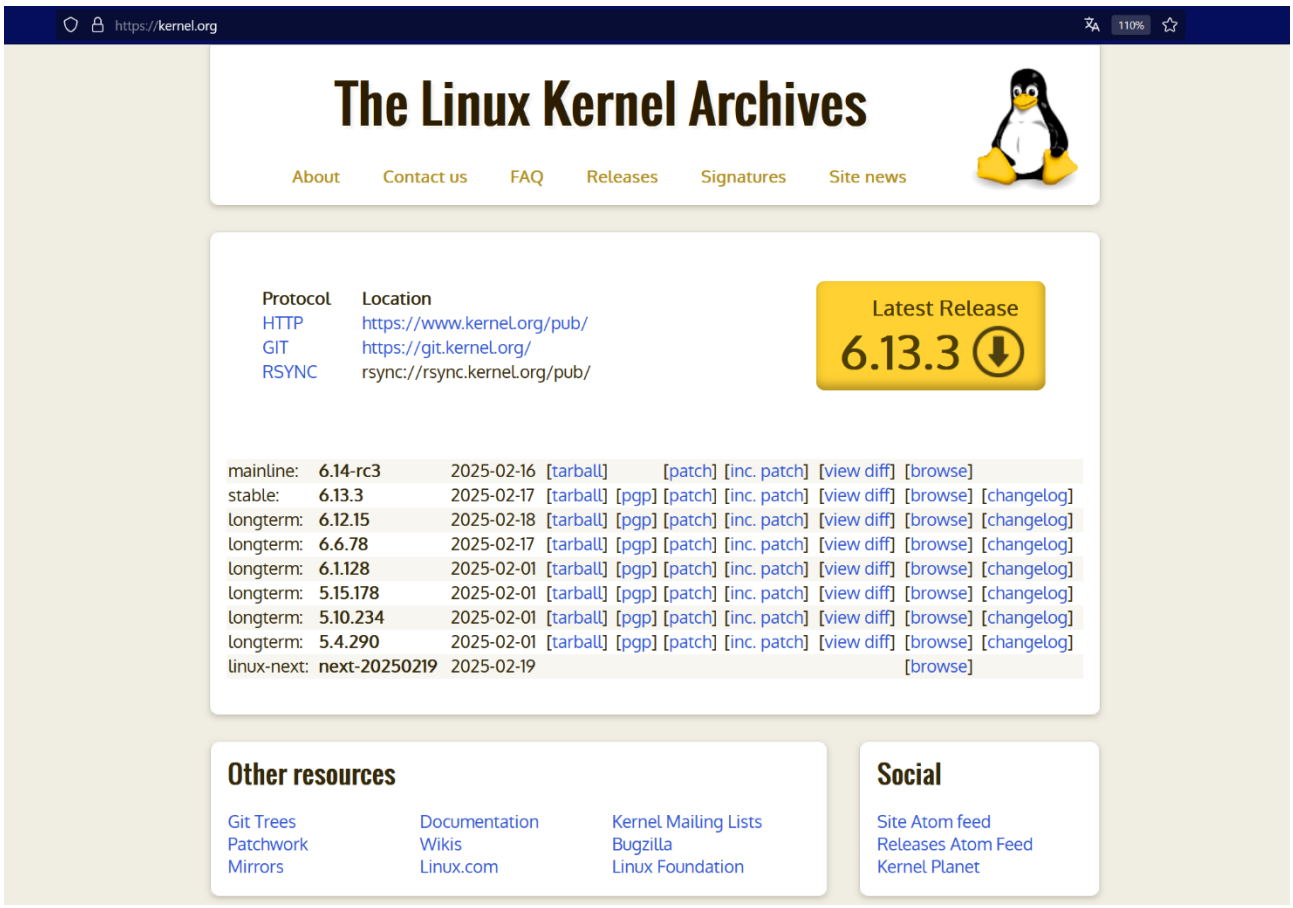



Figura 5. Sitio web oficial de Linux kernel. Fuente: https://kernel.org

El archivo para descargar es linux-6.13.3.tar.xz. Una vez en la ubicación del archivo desde la terminal utiliza el comando "xz -d" seguido del nombre del archivo y su extensión ".xz". Como se indica a continuación:

```
root@svrlinux:/usr/src/# xz -d linux-6.13.3.tar.xz ↵ Enter
```

Este comando descomprimirá el archivo quedando como linux-6.13.3.tar, ahora lo que faltaría es desempaquetar el archivo .tar, para esto ahora ejecutaremos el comando tar como se indica a continuación:

```
root@svrlinux:/usr/src/# tar -xvf linux-6.13.tar ↵ Enter
```

Al ejecutar este comando simplemente nos quedará el directorio con el contenido del *kernel*. Pero antes debemos verificar que tenemos el suficiente espacio en el disco, para esto podemos hacer uso del siguiente comando:

Comando para verificar el espacio disponible

```
root@svrlinux:/usr/src/# df -k ↵ Enter
```

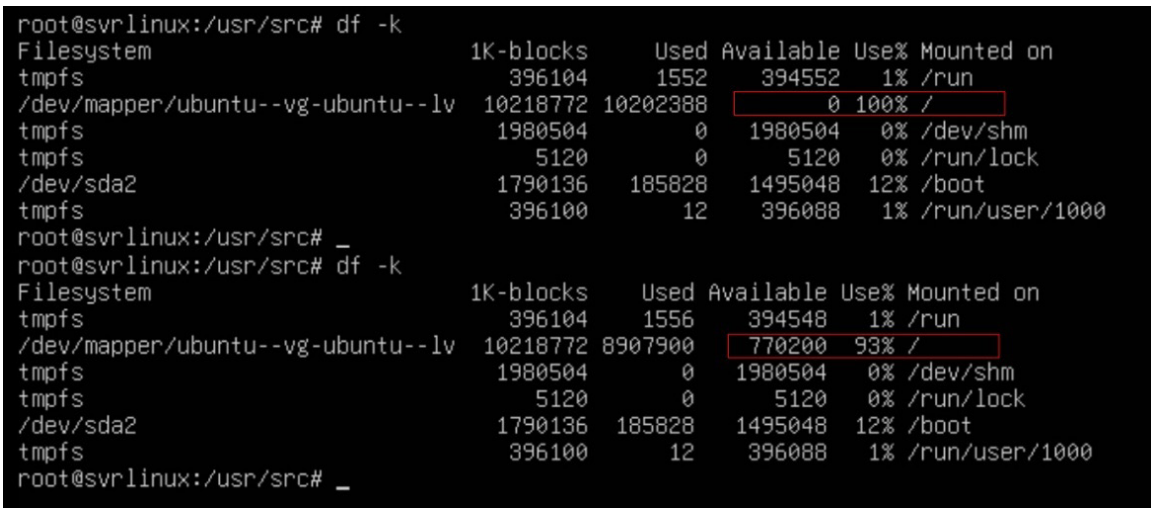



Figura 6. Despliegue de información acerca del espacio disponible en disco mediante el comando df -k

En caso de no tener suficiente espacio, antes de instalar el nuevo *kernel* es necesario eliminar archivos innecesarios, por ejemplo, cuando se van actualizando los kernels, asimismo van quedando anteriores instalaciones dentro del directorio /usr/src/. Entonces, como se indica en la pantalla siguiente, dejaremos solamente la última actualización, es decir /linux-headers-6.13.3/ y /linux-headers-6.13.3-generic/ sin embargo, para poder eliminar correctamente los residuos es mejor utilizar el comando siguiente:

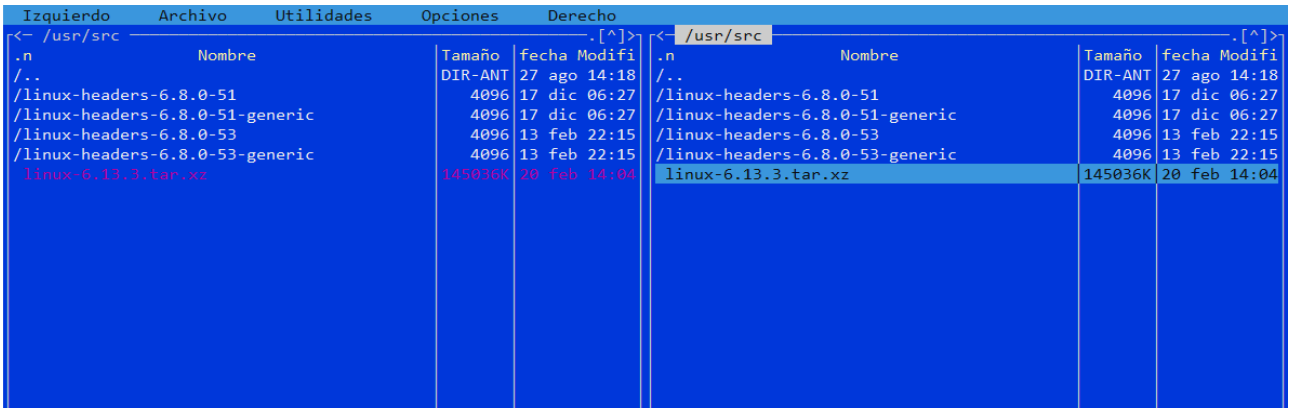



Figura 7. En este directorio /usr/src/ se almacenan versiones anteriores del *kernel* Linux.

root@svrlinux:/usr/src/# apt-get remove –purge linux-headers-6.8.0.51

root@svrlinux:/usr/src/# apt-get remove –purge linux-headers-6.8.0.51-generic

Para liberar espacio podríamos eliminar directamente archivos temporales con el comando sudo rm -rf /tmp/*, también podemos buscar archivos grandes que pueden ser eliminados o movidos a otra ubicación mediante el comando sudo find / -type f -size +100M, igualmente podríamos reasignar espacio o extender el volumen lógico para el sistema de archivos raíz "/". Puede además descomprimirse el archivo dentro del directorio *home* y luego copiamos o movemos el directorio del *kernel* dentro del directorio /usr/src/.

Tras completar este proceso, obtendremos el directorio /usr/src/linux-6.13.3/. Al examinar los contenidos del directorio /linux-headers-6.8.0-51-generic/, notaremos que son similares, pero con la presencia de un archivo oculto adicional llamado .config, el cual puede visualizarse usando el editor mc o el comando ls -la. Este archivo es el config base del *kernel* actual, que incluye todas las opciones de configuración utilizadas para compilar el *kernel* correspondiente a esa versión específica. Aunque es posible recompilar este *kernel*, modificando directamente el archivo .config o utilizando la herramienta make menuconfig para ajustar

las opciones antes de iniciar la compilación, en este caso realizaremos la compilación con el nuevo *kernel*. El archivo .config es importante para la configuración y compilación del *kernel* de Linux, ya que permite una personalización detallada del mismo.

Previamente hay que asegurarse de tener instaladas las herramientas necesarias para compilar el *kernel*, puede instalarlas con el siguiente comando:

```
#apt-get install build-essential libncurses-dev bison flex libssl-dev libelf-dev
```

Continuando con el proceso de instalación del nuevo *kernel* a partir del archivo descargado del sitio web oficial kernel.org, procederemos con la copia y luego con la compilación de este. Desde la terminal ingresamos al directorio /usr/src/linux-6.13.3/

```
root@svrlinux:/usr/src/# cd linux-6.13.3
root@svrlinux:/usr/src/linux-6.13.3#
```

En seguida copiaremos el archivo de configuración del *kernel* actual al directorio /usr/src/linux-6.13.3/ con el nombre .config, esto debido a que utilizaremos la misma configuración del *kernel* actual como base, para luego añadir o eliminar parámetros.

```
root@svrlinux:/usr/src/linux-6.13.3# cp -v /boot/config-$(uname -r) .config
```

O también podemos utilizar el siguiente comando si conocemos la versión del *kernel* actual. Lo que se está haciendo es la copia del archivo config-6.13.3, ,que se encuentra en el directorio /boot/config-6.13.3-generic hacia el directorio /usr/src/linux-6.13.3/ como nombre .config, y a través del comando cp.

```
root@svrlinux:~$ uname -r ┘ Enter
6.13.3-generic

root@svrlinux:/usr/src/linux-6.13.3# cp -v /boot/config-6.13.3-generic .config
```

El directorio /usr/src/ es comúnmente utilizado para almacenar archivos de desarrollo y compilación de *kernels* personalizados. Lo que puede encontrar es lo siguiente:

**Código fuente del kernel:** Si ha instalado los paquetes del código fuente del *kernel* (linux-source), el código fuente se almacenará en este directorio. Puede encontrar subdirectorios como linux-source-<versión>.

**Módulos adicionales:** Puede contener los módulos adicionales o los *headers* del *kernel* necesarios para compilar módulos de terceros. Los headers del kernel están en subdirectorios con nombres como linux-headers-<versión>.

**Paquetes de desarrollo:** También puede encontrar otros paquetes relacionados con el desarrollo de *software*, especialmente aquellos que requieren la construcción de módulos o interfaces específicas para *hardware* o *software*.

En Linux se tiene básicamente cinco herramientas para configurar su código fuente, pero particularmente utilizaremos make menuconfig ya que presenta una interfaz de usuario en modo texto y basada en menús

que nos permitirá trabajar desde una terminal, entonces, el usuario podrá elegir las funciones de Linux que luego se compilarán. Estas herramientas se listan a continuación:

**make menuconfig:** Interfaz de usuario basada en menús que permite seleccionar características y opciones de Linux para compilar.

**make xconfig:** Requiere Qt y proporciona una interfaz gráfica para la configuración.

**make gconfig:** Utiliza GTK+ y ofrece una interfaz gráfica similar a make xconfig.

**make nconfig:** Similar a make menuconfig, pero con algunas diferencias en la interfaz.

**make config/make oldconfig:** Una utilidad basada en preguntas y respuestas para la configuración de Linux.

## Configuración y compilación del *kernel*

El objetivo de la configuración y compilación del *kernel* es capacitar al profesional para diseñar un núcleo a medida, optimizado exclusivamente para las necesidades de su infraestructura. Al seleccionar únicamente los módulos esenciales, se reduce la superficie de ataque y se mejora el rendimiento del sistema, eliminando procesos innecesarios para que tecnologías como Docker y LXC aprovechen al máximo el *hardware*. Esta competencia otorga un control total sobre el sistema operativo, permitiendo transformar un núcleo genérico en una plataforma de alta eficiencia y seguridad.

A continuación, ejecutaremos el comando make menuconfig. Ya dentro del directorio /linux-6.13.3/, se presentará una ventana basada en *new curses* (una biblioteca de programación utilizada para manipular interfaces de usuario en modo texto en terminales) que nos permitirá navegar las categorías de *drivers* como veremos a continuación. Si no está instalado, se pedirá ejecutar apt install make y make-guile. También puede ser necesario instalar gcc y herramientas de desarrollo.

```
root@svrlinux:/# apt update ↵ Enter
root@svrlinux:/# apt install build-essential libncurses-dev bison flex libssl-dev ↵ Enter
```

build-essential: Incluye GCC, make y otras herramientas requeridas.

libncurses-dev: Necesaria para la interfaz de menuconfig.

bison y flex: Herramientas para el análisis sintáctico, requeridas para compilar el kernel.

libssl-dev: Ayuda con funciones relacionadas con SSL.

Si está correctamente instalado, podremos comprobarlo con el comando gcc ––version.

A continuación, ejecutamos el comando make menuconfig ya para presentar la pantalla de configuración del nuevo *kernel*.

```
root@svrlinux:/usr/src/linux-6.13.3# make menuconfig ↵ Enter
```

GCC es GNU Compiler Collection, una herramienta que nos permitirá compilar el código fuente de programas escritos en lenguajes como C y C++. Esto lo profundizaremos más en el capítulo de Guía de programación C y C++ en Linux.

Ahora el administrador podrá ajustar el núcleo del sistema para optimizarlo de acuerdo con sus necesidades específicas de *software* y de *hardware*. Con esta herramienta interactiva, en modo texto, el administrador podrá seleccionar o deseleccionar los módulos necesarios antes de proceder con la compilación del *kernel*.

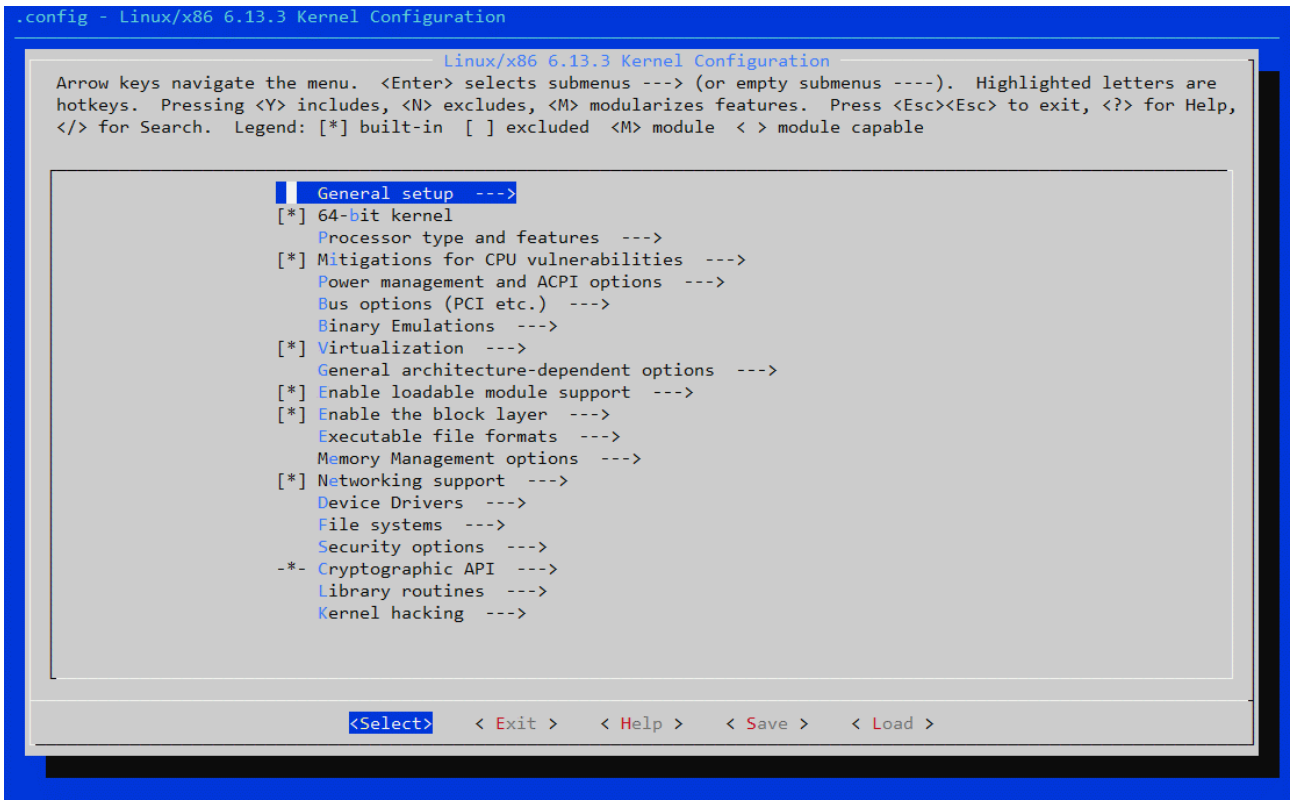



Figura 8. Configuración del *kernel* con make menuconfig en Linux Ubuntu Server.

Uno de los motivos por los que se necesita realizar una configuración y compilación del *kernel* es porque necesitamos optimizar nuestro hardware, por ejemplo, en el caso de tener un *kernel* más pequeño y eficiente, hay que eliminar controladores o módulos innecesarios o que no se utilizan para nuestro *hardware*.

También se realiza la configuración y compilación del *kernel* cuando se necesita compatibilidad con *hardware* o *software* específico, por ejemplo, un controlador para una tarjeta capturadora de señales satelitales como la TBS 6904SE DVB-S2. Otro motivo es la seguridad, como por ejemplo, activando o desactivando características de seguridad, redes o virtualización; se puede activar QoS (Quality of Service), una característica necesaria para gestionar tráfico de red y priorizar paquetes con base en ciertas reglas específicas.

Las opciones de la parte inferior las seleccionamos con la tecla tab y para movernos por las diferentes opciones, utilizamos las teclas de dirección, mientras que para marcar o desmarcar las diferentes opciones, se puede utilizar la barra espaciadora del teclado. Aquellas opciones indicadas con asterisco [*] representan módulos incorporados en el *kernel* propiamente dicho (es decir no son modularizables). Por otra parte, [M] presenta módulos que pueden crearse por fuera del *kernel*, los cuales pueden habilitarse o deshabilitarse bajo demanda.

En la opción de <Load> cargamos el módulo que previamente habíamos copiado o simplemente verificamos que sea el archivo .conf. Luego debemos compilar el *kernel,* en este caso, con los módulos que hayamos marcado o inclusive desmarcado, para luego preparar su inicio, realizar pruebas y aceptación del nuevo *kernel*. A continuación, procedemos con la compilación, por ejemplo, como se mencionaba anteriormente, si queremos activar el soporte para (QoS) and/or fair queueing, es decir Quality of Service (Calidad de Servicio), un mecanismo de red que ayuda a controlar y priorizar el tráfico para que el tráfico más crítico se pueda enviar primero en la red. Esta característica ayudaría a garantizar el rendimiento para el tráfico de red crítico, lo que es especialmente útil en redes donde los recursos de ancho de banda son limitados o donde ciertas aplicaciones requieren prioridad (como voz sobre IP, videoconferencias, etc.); y también activar el soporte para *fair queueing* (encolado justo), una técnica utilizada para responder a que todos los flujos de tráfico en una red reciban una cantidad equitativa de ancho de banda. Esto evitará que un solo flujo de datos monopolice la conexión, distribuyendo los recursos de red de manera más equilibrada (EtherWAN, 2022).

Entonces, el *kernel* tendrá soporte para estas técnicas permitiendo implementar políticas de QoS y mecanismos de "encolado justo" en nuestro sistema operativo Linux, por ejemplo, si se deseara dar más prioridad al tráfico https más que al tráfico mysql.

En este contexto, existen herramientas gráficas en Linux que permiten visualizar el tráfico de red como es Etherape, una aplicación que puede utilizarse para identificar la fuente del tráfico pesado que está originando una congestión en la red y así, con estos mecanismos de control de tráfico, se pueden resolver estos inconvenientes. La explicación de estas técnicas a profundidad va más allá del contenido de este libro, sin embargo, podemos darnos cuenta de que saber gestionar el *kernel* es importante. Muchos se preguntarán si es necesario actualizar el *kernel* cada vez que una nueva versión aparece y la respuesta es que depende del caso; la explicación a esto es la siguiente: cuando un nuevo *kernel* aparece puede ser por las siguientes razones:

- Han aparecido nuevas características del *hardware*.
- Se han corregido fallas en la programación.
- Fallos de seguridad se han rectificado.
- Un nuevo *hardware* es soportado.

## Gestión de parches de seguridad y técnicas de Live Patching

Si las características nuevas no se van a utilizar es evidente que no es necesario actualizar dicho *kernel*, cada usuario debe decidir cuándo es hora de una actualización.

No olvidemos también el **parche (***patch***)** del *kernel*, pues se trata de un archivo que, al descargarlo, es del tipo patch-6.13.3.xz, lo que implicaque representa un conjunto de actualizaciones que se aplican al código fuente del *kernel* de Linux para corregir errores, mejorar el rendimiento, añadir nuevas características o cerrar vulnerabilidades de seguridad. En el sitio web oficial de kernel.org podemos encontrar estos parches de acuerdo con la versión de *kernel*. A continuación, realizaríamos lo siguiente en caso de querer aplicar el parche:

Cabe mencionar que existen tres maneras diferentes de aplicar los parches de seguridad al *kernel* de Linux, los cuales son:

1. Desde la línea de comandos o terminal.
2. A través de kexec (*kernel execute*), una característica del *kernel* de Linux que permite cargar un nuevo *kernel* sin pasar por el proceso completo de reinicio de la máquina.
3. Mediante herramientas de *live kernel patching* sin reinicio: Canonical Livepatch, Red Hat's Kpatch, Oracle Ksplice Uptrack, KernelCare Enterprise y SUSE Kgraft (SLE Live Patching).

Sin embargo, únicamente aplicaremos los parches desde la línea de comandos. Así, para aplicar el parche al código fuente del *kernel* y después de descomprimir el archivo, dentro del directorio /usr/src/linux-6.13.3/, se realizaría lo siguiente:

```
cd /usr/src/linux-6.13.3/

root@svrlinux:/usr/src/linux-6.13.3# patch -p1 < patch-6.13.3
```

Aquí, -p1 es una opción que indica cómo tratar las rutas en el archivo de parche. Esta opción depende de cómo se creó el parche. En muchos casos, -p1 es la opción correcta, pero si no es así, podría probar con -p0 o ajustarse según la estructura del parche.

```
root@svrlinux:/usr/src/linux-6.13.3# patch -s -p0 < patch-6.13.3
```

**-s:** Ejecuta *patch* en modo silencioso, lo que significa que no se mostrarán mensajes de éxito en la salida (solo se mostrarán mensajes de error si los hay).

**-p0:** Indica a *patch* que no debe ignorar ningún nivel de directorios en las rutas de los archivos en el parche. Esto es útil cuando las rutas en el parche coinciden exactamente con las rutas de los archivos en el sistema.

**< /ruta/a/patch-6.13.3:** Indica que el contenido del archivo de parche patch-6.13.3 se debe leer y aplicar al código fuente del *kernel*.

Luego de ejecutar el comando anterior pueden presentarse algunos de los siguientes posibles mensajes que se indican asimismo con sus soluciones:

**"***Reversed (or previously applied) patch detected! Assume -R? [n]***":** Esto indica que el parche que está intentando aplicar ya ha sido aplicado o que se está intentando aplicar en sentido inverso. Puede elegir "n" para no aplicar el parche o "y" si desea revertirlo (esto deshará los cambios si el parche ya había sido aplicado).

**"***Hunk* **#** *failed at line* **#.":** Significa que el parche no coincide exactamente con el código fuente. Esto puede deberse a diferencias entre la versión del archivo que el parche esperaba y la versión actual. En este caso, podría ser necesaria una modificación de forma manual de los archivos o conseguir un parche que se adapte mejor a su versión actual del *kernel*.

**"***Apply anyway? [n]***":** Si aparece la pregunta de si desea aplicar el parche de todas formas, es porque hay algo en el parche que no coincide completamente con los archivos actuales. Responder "y" hará que la máquina intente aplicar el parche, aunque haya discrepancias.

Si aparece el mensaje "*File to patch*:" es porque el comando *patch* no puede encontrar automáticamente el archivo que necesita ser parcheado. Esto puede suceder por varias razones, como que el archivo esté en una ubicación diferente a la esperada o que el nombre del archivo haya cambiado.

Una vez aplicado el parche, se deberá recompilar el *kernel* para que los cambios surtan efecto:

Para esto, ejecutamos el comando make menuconfig y debemos asegurarnos de seleccionar las opciones necesarias, seguidamente el comando make -j$(nproc), make modules_install y make install. Después de la instalación, reiniciamos el sistema para arrancar con el *kernel* actualizado. Este proceso aplicará el parche al *kernel* y, después de reiniciarse, su sistema debería estar ejecutando el *kernel* con las actualizaciones aplicadas.

Cabe mencionar que los parches del *kernel* están diseñados normalmente para versiones específicas del mismo o un rango de versiones cercanas. Si el parche no es compatible, es probable que no funcione y el comando *patch* podría fallar o aplicarse incorrectamente y provocar errores de compilación en tiempo de ejecución. Por ejemplo, para una versión cercana, un parche 6.13.2 aplicado a 6.13.3 es posible que funcione, sin embargo, al pasar de una versión 6.13 a una 6.15 es probable que el parche no sea compatible. Esto deberá confirmarlo en la documentación del parche, además existen comandos que pueden realizar una prueba simulada utilizando el comando siguiente:

```
root@svrlinux:/usr/src/linux-6.13.3# patch –dry-run -p1 < archivo-del-parche
```

Es mejor probar el parche en un entorno seguro, nunca directamente en un sistema en producción.

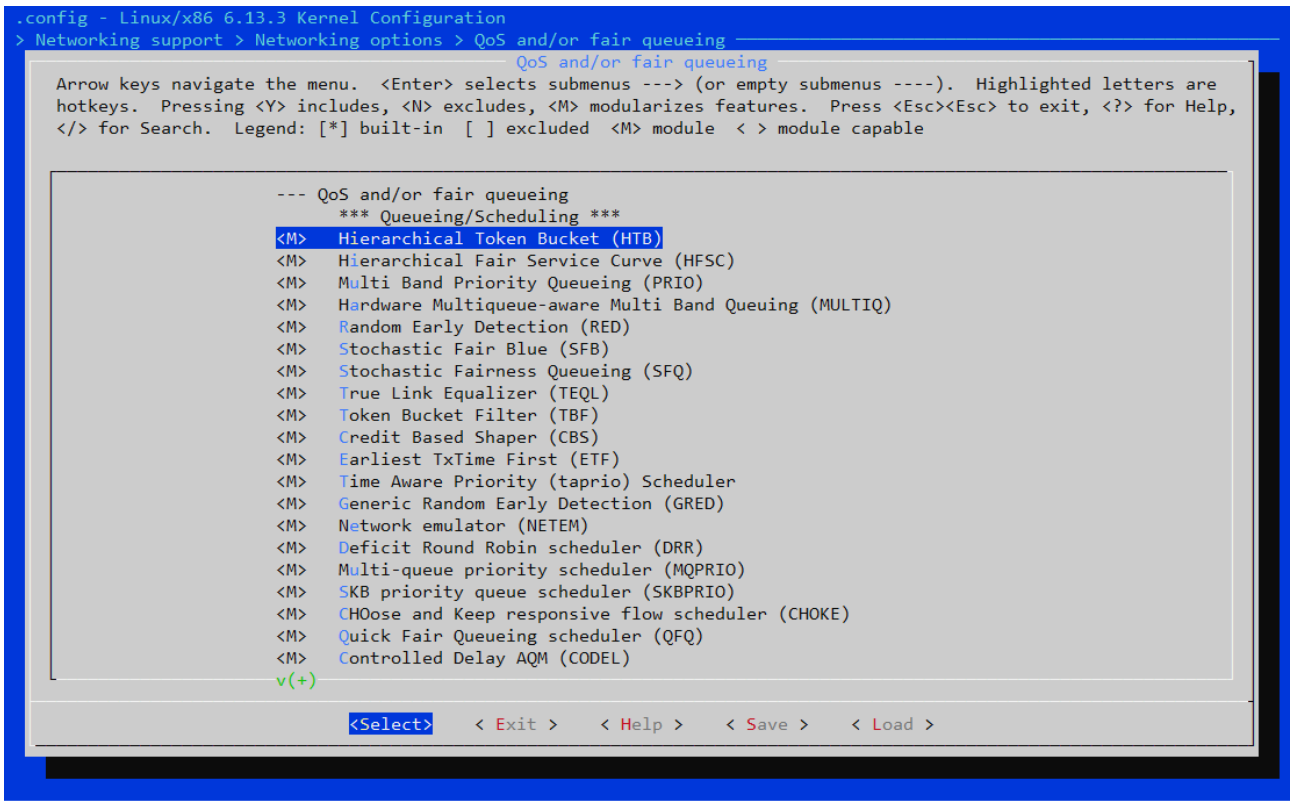



Figura 9. Selección de los módulos necesarios en la configuración del *kernel* con *make menuconfig*

El siguiente paso luego de seleccionar los módulos respectivos y guardar la configuración es limpiar los archivos generados durante compilaciones previas del *kernel*, para esto ejecutaremos el siguiente comando:

```
root@svrlinux:~$ make clean ┘Enter
```

*make clean* en la línea de comandos le permite deshacerse de sus archivos de objetos y ejecutables. A veces, el compilador vinculará o compilará archivos de forma incorrecta y la única forma de comenzar de nuevo es eliminar todos los archivos de objetos y ejecutables. A continuación, ejecutamos el comando:

```
root@svrlinux:~$ make -j$(nproc) ┘Enter
```

Si no hemos seleccionado previamente los módulos mediante *make menuconfig*, se presentarán algunas opciones para seleccionar los módulos requeridos, pero desde el modo texto o desde la terminal. El comando make -j$(nproc) ejecuta el proceso de compilación utilizando *make*, una herramienta común para construir *software* a partir de archivos fuente. Si su máquina tiene 8 núcleos de CPU, el comando será equivalente a make -j8, ejecutando hasta 8 tareas de compilación simultáneamente. Este comando es especialmente útil para compilar proyectos grandes (como el *kernel* de Linux, *software* complejo, etc.), ya que reduce significativamente el tiempo de compilación.

A partir de la ejecución de este comando, tomará unos minutos para que la compilación se realice totalmente.

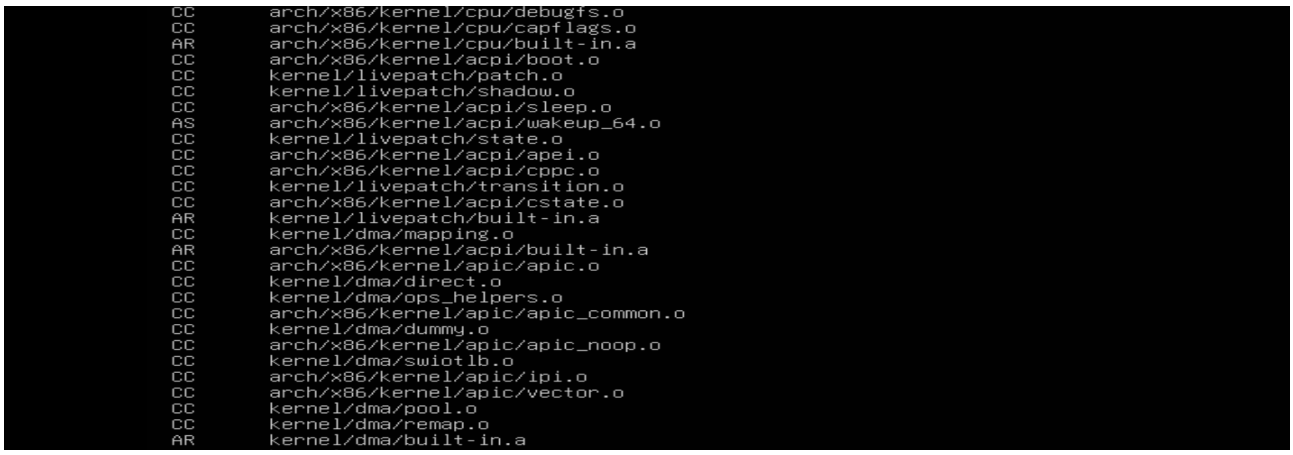



Figura 10. Pantalla de salida del proceso de compilación del nuevo *kernel*

Si el proceso anterior termina y no ha presentado mensajes de error, entonces la compilación se ha realizado con éxito. A continuación, ejecutamos los comandos:

```
root@svrlinux:~$ make modules_install ↵ Enter
root@svrlinux:~$ make install ↵ Enter
```

## Distribuciones Linux

El objetivo de este apartado es tener la capacidad de evaluar la durabilidad y el soporte técnico de una distribución para garantizar la continuidad del negocio por lo menos a mediano y largo plazo. Linux es un universo de posibilidades gracias a su naturaleza de código abierto, lo que permite a desarrolladores de todo el mundo ajustar y modificar el sistema para necesidades específicas: desde servidores y estaciones de trabajo hasta dispositivos embebidos (Smart TVs, routers o Raspberry Pi). Esta flexibilidad ha dado lugar a diversas distribuciones (distros), cada una optimizada para entornos técnicos distintos.

### 1. Ubuntu Server: elección profesional

Ubuntu Server es una variante de Linux basada en Debian, diseñada específicamente para ofrecer un entorno robusto y estable en la ejecución de servicios empresariales. Aunque existen versiones como Desktop (para escritorio), Pro o Core, nos enfocaremos en Ubuntu Server 24.04 LTS. La elección de esta versión se basa principalmente en lo siguiente:

**Popularidad:** garantiza una comunidad amplia y documentación abundante.

**Soporte LTS (***Long Term Support***):** el término "Soporte a largo Ppazo" indica que esta versión recibirá actualizaciones de seguridad y mantenimiento de estabilidad durante varios años, evitando la obsolescencia que sufren sistemas como CentOS o White Box Linux, que han perdido vigencia.

## 2. Panorama de distribuciones

El ecosistema Linux se divide principalmente en dos vertientes:

**Comerciales:** como Red Hat Enterprise Linux (RHEL) y SUSE Linux Enterprise, enfocadas en el soporte corporativo pago.

**Comunitarias / Código abierto:** como Debian, Fedora, Linux Mint o las nuevas alternativas para servidores: Rocky Linux y AlmaLinux.

## 3. Diferencias técnicas y comandos

Aunque todas comparten el mismo núcleo, las distribuciones difieren en sus herramientas y gestores de paquetes:

**Familia Debian/Ubuntu:** utilizan los comandos apt y dpkg.

**Familia Red Hat/Fedora:** utilizan dnf y rpm.

**Herramientas Exclusivas:** algunas distros incluyen utilidades propias, como yast en openSUSE, que facilita la configuración integral del sistema.

Sin embargo, gran parte del conocimiento es transversal. Comandos básicos de navegación como ls (para listar archivos) son comunes en casi todas las distros, incluyendo sistemas Unix y macOS. Esto permite que el aprendizaje en Ubuntu sirva como una base sólida para dominar cualquier otro entorno Linux.

## 4. Linux en el mundo embebido

La versatilidad de Linux llega hasta las placas Raspberry Pi. Su sistema oficial, Raspberry Pi OS (antes Raspbian), también está basado en Debian y es la herramienta estándar para la enseñanza de informática y el desarrollo de sistemas embebidos.

Elegir una distribución Linux adecuada para un servidor requiere analizar cuidadosamente varios factores, ya que cada distribución tiene características específicas que pueden adaptarse mejor a ciertos escenarios.

Hasta el momento, Linux Ubuntu Server, en nuestra experiencia, ha resultado funcional y relativamente fácil de manejar como servidor web. Sin embargo, a continuación, se presenta un enfoque detallado para que pueda tomar una decisión más acertada:

**Depende del tipo de aplicación o servicio**

- Debemos considerar el tipo de servicios que se vaya a ejecutar:
- Servidores web (Apache, Nginx, etc.): Ubuntu Server, Debian o Rocky Linux son opciones populares por su estabilidad y facilidad de uso.
- Bases de datos (MySQL, PostgreSQL): distribuciones como RHEL, Rocky Linux, y Ubuntu Server ofrecen excelentes herramientas de soporte para bases de datos empresariales.
- Virtualización y contenedores (Docker, KVM): Ubuntu Server, Red Hat Enterprise Linux (RHEL), y SUSE Linux Enterprise Server (SLES) tienen excelentes características de integración para contenedores y entornos virtualizados.

**Estabilidad y soporte empresarial**

Si nuestra prioridad es estabilidad a largo plazo y soporte técnico, podemos considerar las distribuciones siguientes:

- Red Hat Enterprise Linux: con soporte empresarial robusto, ideal para empresas.
- SuSE: especialmente fuerte en la nube y alta disponibilidad.
- Ubuntu Server: popular por su soporte a largo plazo (LTS) y comunidad activa.

Si busca algo más económico, pero con soporte basado en RHEL: Rocky Linux o Alma Linux: ambas son reemplazos comunitarios de CentOS con soporte empresarial opcional.

**Facilidad de uso y administración**

Para usuarios nuevos o administradores que prefieren simplicidad:

- Ubuntu Server: tiene un enfoque más accesible con abundante documentación.
- Debian: con una curva de aprendizaje más pronunciada, pero extremadamente confiable.

Para administradores experimentados:

- RHEL, SUSE y BSD pueden ser ideales debido a sus herramientas avanzadas.

**Costo**

- Distribuciones gratuitas: Debian, Rocky Linux, Alma Linux, y CentOS 7 (aunque CentOS ya no recibe actualizaciones activas) son opciones sin costo directo.
- Distribuciones comerciales: RHEL, SUSE Linux Enterprise Server: Requieren suscripción, pero incluyen soporte técnico de primer nivel.

**Soporte comunitario o técnico**

- Ubuntu Server: posee una gran comunidad activa y soporte comercial opcional (Canonical).
- RHEL: Está enfocado en empresas con soporte técnico premium.
- Debian: tiene una comunidad técnica fuerte, ideal para quienes buscan estabilidad y no necesitan soporte comercial.
- Rocky Linux y Alma Linux: comunidades que están creciendo rápidamente como alternativas a CentOS.

**Compatibilidad con *hardware***

Si su *hardware* es moderno y necesita soporte inmediato para los últimos controladores:

- Ubuntu Server y RHEL suelen ofrecer una mejor compatibilidad. Si utiliza un hardware especializado, como sistemas embebidos o arquitecturas no convencionales: SuSE o Debian son excelentes opciones debido a su flexibilidad.

**Seguridad y actualizaciones**

- Ubuntu Server: ofrece Canonical Livepatch para aplicar parches de seguridad sin reiniciar.
- RHEL: tiene Kpatch y soporte empresarial para actualizaciones críticas.
- SUSE: Live Patching para entornos sensibles.
- Debian: comunidad activa que lanza actualizaciones regulares.

**Requerimientos específicos**

- Alta disponibilidad y entornos críticos: SUSE o RHEL.
- Escenarios de aprendizaje o pruebas: Debian, Ubuntu Server, o CentOS.
- Cargas mixtas con Windows: Ubuntu Server y RHEL son ideales para integrar servicios con Active Directory o compartir recursos.

Si usted busca un sistema fácil de utilizar con una comunidad amplia y activa, Ubuntu Server es una excelente alternativa. Para ambientes empresariales que requieren soporte técnico de alta calidad, distribuciones como RHEL o SuSE son las más recomendadas. Si, por otro lado, necesita una solución gratuita con características empresariales, Rocky Linux o Alma Linux son opciones confiables, en el caso de Linux el ser gratuito no implica que sea un sistema corriente, muchas empresas de renombre utilizan sistemas operativos *open source.*

La elección de la distribución Linux más adecuada dependerá de los requerimientos específicos y del nivel de soporte que necesite para sus operaciones.

# CAPÍTULO II INSTALACIÓN DE LINUX UBUNTU SERVER

- Crea una imagen vertical en formato 2:3 de un mapa urbano estilo street-style de Loja, Ecuador, colocado sobre una mesa de madera oscura, con textura de papel cartográfico premium, líneas de calles, rutas y referencias urbanas visibles.
- Desde el mapa debe emerger una versión miniatura fotorrealista de Loja, como un diorama urbano tridimensional. La ciudad debe levantarse desde el mapa de manera natural, integrando arquitectura patrimonial, paisaje andino, vida urbana local y elementos culturales propios del lugar.
- Incluye de forma clara y reconocible: la Catedral de Loja, la Puerta de la Ciudad, la Iglesia de San Sebastián, montañas andinas, árboles urbanos, personas caminando por plazas, taxis locales y detalles culturales relacionados con la música lojana.
- Los monumentos, iglesias, plazas, montañas, calles, vegetación, transporte y personas en miniatura deben verse realistas, bien proporcionados y con detalles precisos. La imagen debe transmitir identidad territorial, memoria urbana y orgullo local.
- Composición editorial premium, cámara en ángulo cenital ligeramente inclinado, profundidad de campo cinematográfica, iluminación suave de estudio, sombras realistas, colores naturales, atmósfera cálida, alto nivel de detalle, textura realista, acabado de fotografía publicitaria, miniature city diorama, photorealistic, ultra detailed, 8k, sharp focus.Linux Ubuntu Server
    - Comparativa entre Ubuntu Server y otras distros
    - Qué hay de novedades
- Requisitos mínimos del sistema
- Creación del USB de arranque
- Proceso de instalación
    - Errores comunes durante la instalación

## Linux Ubuntu Server

El objetivo de aprendizaje de **Ubuntu Server** es dominar la administración de sistemas operativos en la medida de lo posible sin interfaz gráfica, priorizando la eficiencia en el uso de recursos y la gestión remota segura. Al prescindir de un entorno visual, el administrador aprende a interactuar directamente con el sistema a través de la línea de comandos (CLI), optimizando el *hardware* para servicios críticos como bases de datos, servidores web y orquestadores de contenedores. Este conocimiento permite configurar entornos de alta disponibilidad donde el minimalismo del *software* reduce drásticamente la superficie de ataque, convirtiéndose en la competencia clave para gestionar infraestructuras en la nube donde la escalabilidad y el rendimiento son la máxima prioridad.

Ubuntu Server es una edición de Ubuntu especialmente desarrollada para funcionar en entornos de servidor. A diferencia de las versiones de escritorio, Ubuntu Server no incluye una interfaz gráfica predeterminada, lo que permite optimizar recursos y realizar todas las operaciones mediante la consola, facilitando la administración remota y un rendimiento adecuado para aplicaciones de servidor. Su versatilidad ha impulsado su adopción para una variedad de servicios, como servidores web, bases de datos, servidores de archivos y más, tanto para entornos domésticos como empresariales.

Un servidor es un equipo que provee servicios específicos a otros dispositivos en una red. Puede tener distintas funciones —como almacenamiento de datos, alojamiento web o gestión de bases de datos— y no siempre requiere un *hardware* de gran capacidad; inclusive es posible configurar un servidor en un equipo antiguo para centralizar y compartir archivos de manera local o incluso en acceso remoto.

Ubuntu Server se administra con comandos, similar a otros sistemas Linux, y cuenta con la ventaja de herramientas propias como 'sudo' para la gestión de permisos. Aunque está diseñado para funcionar sin entorno gráfico, este puede añadirse si se requiere. Más adelante veremos la manera de realizar este proceso, aunque normalmente se recomienda trabajar en modo consola para maximizar el control y rendimiento del servidor.

## Comparativa entre Ubuntu Server y otras distros

Es importante entender que Ubuntu Server no está solo en el ecosistema. Su dominio se basa en un equilibrio entre facilidad de uso y estabilidad, pero otras distribuciones ofrecen ventajas específicas según el caso de uso.

Ubuntu Server frente a la competencia

A diferencia de Ubuntu, que busca ser el "estándar equilibrado", otras distribuciones se especializan en nichos extremos: desde el minimalismo absoluto hasta la rigidez corporativa.

*Tabla 1. Comparativa con Ubuntu Server y otras distribuciones Linux*

| Distribución | Filosofía principal | Comparación con Ubuntu Server |
|---|---|---|
| **Debian Stable** | Estabilidad "de roca". | Es el padre de Ubuntu. Es más ligero y conservador; sus paquetes son más antiguos, pero están más probados. Ideal si no necesitas el último *kernel*. |
| **RHEL / Rocky Linux** | Estabilidad corporativa | Usa el gestor dnf y SELinux por defecto. Es el estándar en entornos financieros y gubernamentales donde se requiere soporte comercial a largo plazo. |
| **Alpine Linux** | Minimalismo extremo | Mucho más pequeño (aprox. 5MB). Usa una librería de C distinta (musl). Es la reina absoluta para crear **imágenes de Docker** ultra ligeras. |
| **Arch Linux** | *Rolling Release* | Siempre tiene el último *kernel* disponible. Excelente para laboratorios y aprendizaje profundo, pero arriesgado para servidores de producción por su alta frecuencia de cambios. |
| **Kali Linux** | Auditoría y *pentesting* | También basada en Debian, pero optimizada para ofensiva. Incluye cientos de herramientas de *hacking* preinstaladas y un *kernel* modificado para inyección de paquetes. No se recomienda como servidor de producción por su alta superficie de exposición. |

Desde una perspectiva técnica, Ubuntu Server destaca por su excepcional equilibrio entre estabilidad y modernidad, gracias a su ciclo de lanzamientos LTS (Long Term Support) y su núcleo HWE (Hardware Enablement). A diferencia de otras distribuciones, Ubuntu integra de forma nativa y pulida **de** las últimas capacidades del *kernel* de Linux necesarias para el aislamiento de procesos, como parches de seguridad críticos, optimizaciones en el manejo de redes virtuales y soporte avanzado para sistemas de archivos como ZFS. Esto **favorece a** que el administrador cuente con un motor robusto y actualizado, capaz de exprimir el rendimiento del *hardware* moderno sin sacrificar la fiabilidad exigida en entornos de producción.

Además, su posición como el estándar de facto en la infraestructura de la nube facilita la automatización y la interoperabilidad con herramientas de orquestación como Kubernetes y Docker. Al elegir Ubuntu, se accede

a uno de los repositorios de *software* más vastos y mejor mantenidos del mundo, junto con soluciones exclusivas como Canonical Livepatch, que permite mitigar vulnerabilidades sin interrumpir los servicios. En definitiva, su arquitectura está diseñada para reducir la fricción operativa, permitiendo que el equipo de sistemas se enfoque en la gestión de aplicaciones y microservicios sobre una base que es, simultáneamente, segura por defecto y altamente escalable.

**¿Por qué Ubuntu Server?**

La selección de una distribución de Linux nunca debe ser arbitraria, sino que depende estrictamente de la utilidad final que se le dará al sistema, el tipo de servicios que se desplegarán, el nivel de soporte técnico requerido y los estándares de seguridad de la organización. En el mercado existen alternativas de grado empresarial sumamente robustas: Fedora, para quienes buscan innovación constante; Rocky Linux, Alma Linux, Oracle Linux, para quienes requieren la estabilidad del ecosistema de Red Hat (RHEL); u *open* SUSE, con su potente herramienta de configuración YaST. Cada una de estas opciones es conveniente y válida según el escenario operativo. "No existe la mejor herramienta, sino la más adecuada para el trabajo" (BARDOT, 2022).

Sin embargo, para este texto hemos decidido enfocarnos en Ubuntu Server 24.04 LTS por razones de peso que benefician tanto al estudiante como al profesional. Ubuntu ofrece la combinación más equilibrada entre una curva de aprendizaje accesible y un soporte masivo en la industria. Es la distribución con mayor documentación disponible, la más utilizada en nubes públicas como AWS y Azure, y garantiza un ciclo de vida de hasta doce años con parches de seguridad. Al dominar Ubuntu, el administrador adquiere una base sólida y versátil que facilita la transición hacia cualquier otra distribución, asegurando que sus habilidades sean transferibles y altamente demandadas en el ecosistema global de contenedores y microservicios.

Aunque Ubuntu Server y Ubuntu Desktop nacen del mismo código fuente y comparten el mismo *kernel*, son productos diseñados para filosofías de trabajo opuestas: la productividad humana frente a la eficiencia de máquinas. Ubuntu Desktop hace referencia a la estación de trabajo ya que está diseñada para la interacción directa. Su objetivo es que un usuario pueda encender el equipo y empezar a trabajar de inmediato en tareas de ofimática, desarrollo o diseño.

**Interfaz gráfica (GUI):** Incluye GNOME por defecto, lo que consume recursos significativos de CPU y RAM solo para renderizar las ventanas y efectos visuales.

**Periféricos y latencia:** El Kernel está configurado para responder rápidamente a eventos del ratón y teclado, y trae controladores preinstalados para casi cualquier impresora, cámara o tarjeta WiFi.

**Software de usuario:** Viene cargado con navegadores, suites de oficina (LibreOffice) y reproductores multimedia.

Por otro lado, Ubuntu Server es minimalista por diseño, es el motor de la infraestructura. Se asume que nadie se sentará físicamente frente a la máquina; se administrará de forma remota (vía SSH).

**Consumo de recursos:** Al no tener interfaz gráfica, un servidor puede funcionar con apenas 512 MB de RAM, dejando toda la potencia disponible para los contenedores Docker o las bases de datos.

**Instalación modular:** Solo instala lo estrictamente necesario. Esto no solo mejora el rendimiento, sino que reduce drásticamente la superficie de ataque: si no hay un navegador instalado, un *hacker* no puede explotar una vulnerabilidad en él.

**Kernel optimizado:** El *kernel* de la versión Server suele estar ajustado para priorizar el rendimiento de red y el flujo de datos (*throughput*) sobre la latencia de la interfaz de usuario.

Linux en sí mismo es modular, es decir, por ejemplo, si un alumno instala Ubuntu Server y luego necesita una interfaz, puede ejecutar sudo apt install ubuntu-desktop. Sin embargo, esto no es recomendable en servidores reales porque introduce "ruido" y procesos innecesarios que rompen el principio de minimalismo en cuanto a la gestión de un servidor. Regla de oro para el administrador: "En producción, si no lo necesitas para que el servicio funcione, no lo instales". Por ejemplo, pagar por una interfaz gráfica en la nube es un desperdicio de dinero y ancho de banda.

Cuando hablamos de un *host* de producción, el sistema operativo deja de ser una herramienta de trabajo personal y se convierte en el "cimiento" sobre el cual descansan sus servicios. En un *host* de producción, el sistema operativo debe ser invisible. Si tiene que entrar en él con frecuencia para arreglar algo, no es un buen *host*. Su única misión es proveer el kernel y los recursos para que Docker y LXC hagan su trabajo.

Tabla 2. Diferencia entre Ubuntu Server y Ubuntu Desktop

| Característica | Ubuntu Server | Ubuntu Desktop |
|---|---|---|
| **Interfaz de usuario** | Únicamente **CLI** (línea de comandos). | **GUI** (interfaz gráfica, usualmente GNOME). |
| **Uso de RAM** | Mínimo (aprox. 500 MB - 1 GB en reposo). | Mayor (aprox. 2 GB - 3 GB por la carga gráfica). |
| **Kernel** | Optimizado para cargas de trabajo y red. | Optimizado para periféricos y latencia de usuario. |
| **Paquetes por defecto** | Solo servicios esenciales y SSH. | Suite de oficina, navegadores y reproductores. |
| **Instalador** | Basado en texto (**Subiquity**). | Gráfico e intuitivo (**Ubiquity/Flutter**). |
| **Gestión de red** | Usa **Netplan** (configuración por archivos). | Usa **NetworkManager** (configuración visual). |

## Qué hay de novedades

La última versión LTS, al momento de redactar este libro de Ubuntu Server, es la 24.04.1 LTS, lo que significa soporte a largo plazo, con cinco años de actualizaciones de seguridad y mantenimiento gratuitas, y ampliables a diez años con Ubuntu Pro.

Por otro lado, existe una suscripción para Ubuntu Pro 24.04 LTS que incluye funcionalidades adicionales premium destinado a usuarios empresariales como por ejemplo: mantenimiento de seguridad ampliado hasta el año 2034, Kernel Livepatch para aplicar paches del *kernel* en tiempo de ejecución sin necesidad de reiniciar de inmediato, guía de seguridad de Ubuntu, controladores de Windows certificados para invitados KVM, gestión de sistemas Ubuntu con Landscape, acceso a la base de conocimientos, *kernel* en tiempo real, lógicamente con un costo y soporte adicional.

Según el sitio web oficial de Ubuntu www.ubuntu.com , la última versión de Ubuntu Server presenta algunas de las siguientes novedades:

- Kernel de Linux 6.8 con características de *kernel* de baja latencia habilitadas de forma predeterminada
- Punteros de marco habilitados de forma predeterminada para la mayoría de los paquetes en arquitecturas x86
- Rust 1.75, .NET 8 y OpenJDK 21 con certificación TCK, además de otras actualizaciones de la cadena de herramientas
- 64-bit time_t por defecto en armhf para abordar el problema del año 2038
- Espacios de nombres de usuario no privilegiados y reforzados por AppArmor, pero restringidos de forma predeterminada

## Requisitos mínimos del sistema

- Procesador de 1 GHz o superior
- 1 GB de memoria RAM o superior
- Al menos 5 GB de espacio libre en el disco duro
- Disponer de un puerto USB o una unidad de DVD para el instalador

## Creación del USB de arranque

Primeramente, descargamos la distribución del sitio web oficial de Linux Ubuntu Server generalmente es un archivo con extensión .iso https://ubuntu.com/download/server. La última versión LTS de Ubuntu Server al momento de la elaboración de este libro es la 24.04.1 LTS. LTS significa 'soporte a largo plazo', lo que significa cinco años de actualizaciones gratuitas de seguridad y mantenimiento, extendidas a diez años con Pro Ubuntu.

- Descargamos la imagen ISO
- Creamos en una unidad *flash* USB el arranque del sistema con **RUFUS** o similar
- Arrancamos desde la unidad *flash* USB, esto puede configurarse desde el BIOS

Una vez descargado el archivo "ubuntu-24.04.1-live-server-amd64.iso" procedemos a la creación del USB de arranque. Para esto, usted debe disponer de una memoria *flash* y podemos utilizar Rufus para este propósito. **Rufus** es una aplicación portable, libre y de código abierto para Microsoft Windows que se puede usar para formatear y crear unidades *flash* USB de arranque o Live USB. El sitio web oficial es https://rufus.ie/es/

Ubuntu proporciona habitualmente firmas SHA256 para sus archivos .iso. De esta manera, podemos compararlos contra los archivos .iso que hayamos descargado con la finalidad de comprobar si dicho archivo de instalación no tiene errores o no fue alterado. Primeramente, debemos ir al sitio oficial que es en el siguiente enlace https://releases.ubuntu.com presentará un listado de carpetas de instaladores de Linux, entonces ubicamos el directorio de la versión correspondiente y localizamos el archivo SHA256SUMS.

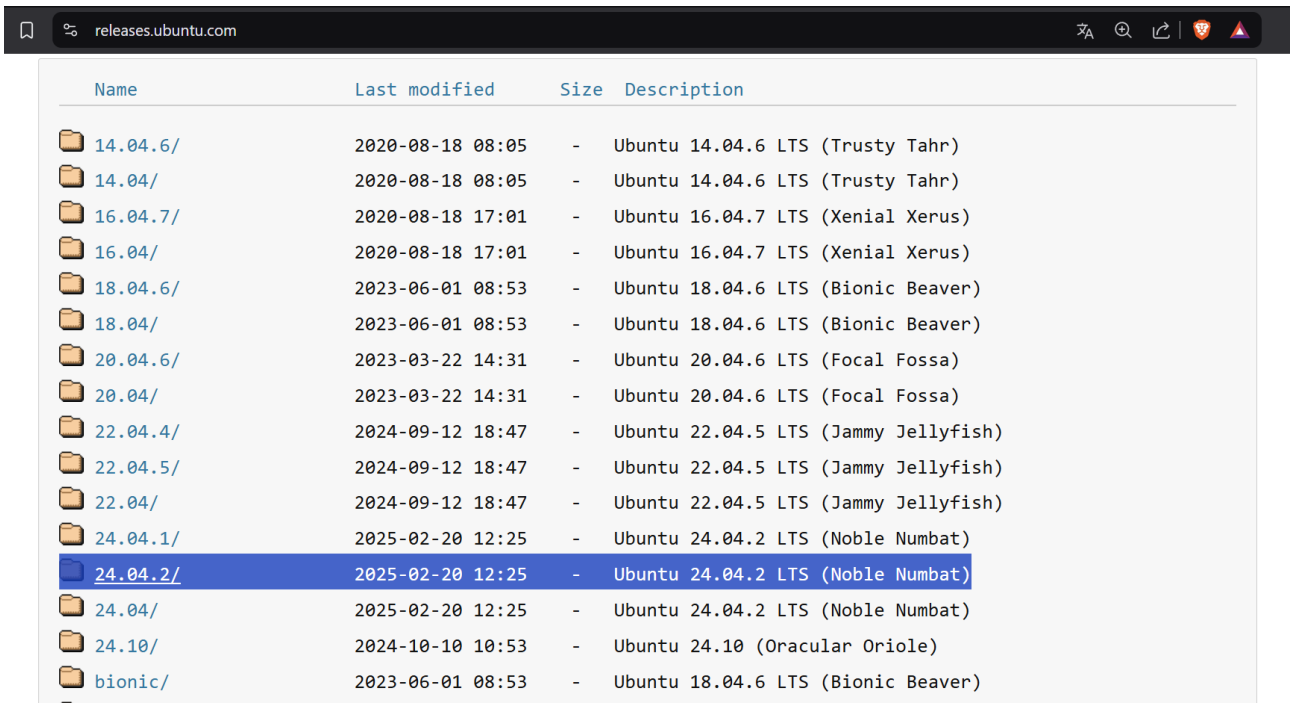



Figura 11. Versiones de Ubuntu disponibles, dentro localicemos el archivo SHA256SUMS, para Ubuntu 24.04.2

El contenido del archivo SHA256SUMS dentro de la siguiente dirección URL es:

https://releases.ubuntu.com/24.04.2/SHA256SUMS

d7fe3d6a0419667d2f8eff12796996328daa2d4f90cd9f87aa9371b362f987bf *ubuntu-24.04.2-desktop-amd64.iso

d6dab0c3a657988501b4bd76f1297c053df710e06e0c3aece60dead24f270b4d *ubuntu-24.04.2-live-server-amd64.iso

5d1eea52103166f1c460dc012ed325c6eb31d2ce16ef6a00ffdfda8e99e12f43 *ubuntu-24.04.2-wsl-amd64.wsl

Desde Windows 11, aplicamos el siguiente comando para obtener la suma SHA256 de nuestro archivo .iso

C:\Users\cesar\CertUtil -hashfile D:\ubuntu-24.04.2-live-server-amd64.iso SHA256 ↵ Enter

SHA256 hash de D:\ubuntu-24.04.2-live-server-amd64.iso:

d6dab0c3a657988501b4bd76f1297c053df710e06e0c3aece60dead24f270b4d

CertUtil: -hashfile comando completado correctamente.

El código de la salida del comando debe coincidir con el código de archivo. En Linux el comando es sha256sum nombre-del-archivo.iso

Luego de haber comprobado el archivo .iso, insertamos una memoria *flash* sin información, ya que los datos serán eliminados, damos clic en el botón seleccionar y escogemos el archivo .ISO, automáticamente el programa configurará las opciones como se indica a continuación. Daremos clic en el botón EMPEZAR y el proceso tomará algunos minutos.

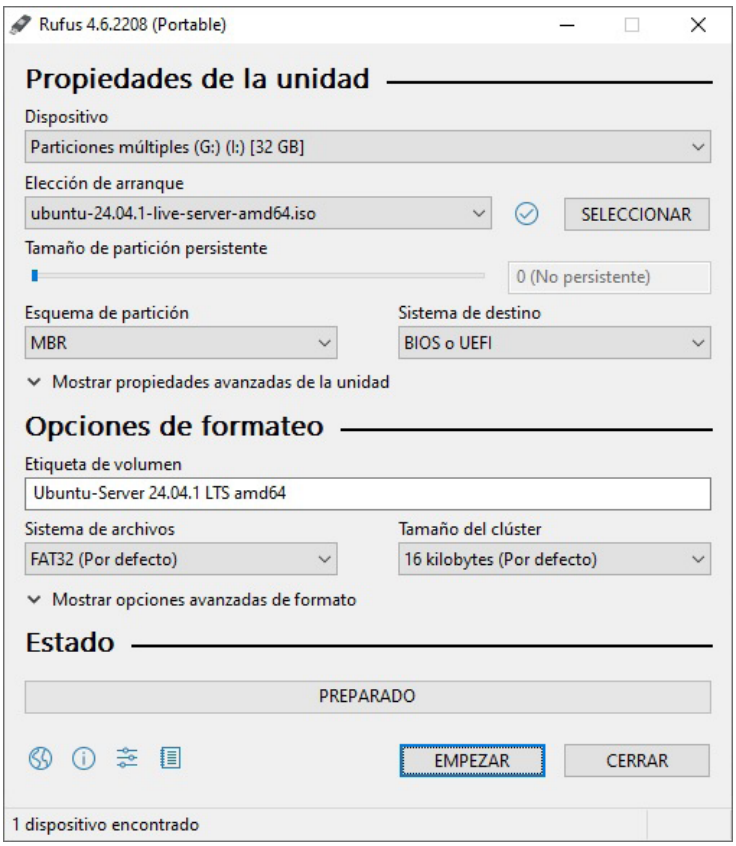



Figura 12. Programa Rufus 4.6 para creación de USB de arranque

Se presentan algunas opciones como: tamaño de partición persistente, esquema de partición, sistema de destino, sistema de archivos, tamaño del clúster que pueden cambiarse según si se necesita; normalmente, el USB de arranque funcionará con las opciones seleccionadas por defecto. Pero podríamos considerar lo que a continuación se describe y dependiendo claramente de nuestra necesidad:

El esquema de partición puede ser MBR o GPT. MBR (*Master boot record*) es el esquema de partición tradicional y ha sido utilizado desde los años ochenta. La información de la partición y el código de arranque se almacenan en los primeros 512 bytes del disco. MBR solo puede gestionar discos de hasta 2 TB debido a su estructura de 32 bits.

MBR permite un máximo de cuatro particiones primarias. Para tener más, se necesita una partición extendida que puede contener múltiples particiones lógicas. Entonces es ideal para discos menores de 2 TB y si se necesita compatibilidad con sistemas antiguos.

Por su parte, GPT (GUID *Partition Table*) es un estándar más moderno y es parte del sistema UEFI (*Unified Extensible Firmware Interface*). Utiliza identificadores únicos para cada partición y está diseñado para superar las desventajas de MBR. GPT permite hasta 128 particiones sin necesidad de particiones extendidas, almacena múltiples copias de la tabla de particiones en diferentes ubicaciones del disco, por lo que es más seguro frente a errores. GPT es compatible con sistemas que usan el *firmware* UEFI en lugar del BIOS tradicional, por cuanto es ideal para discos grandes y sistemas modernos que utilizan UEFI.

El BIOS (*Basic Input/Output System*),en cambio, es un sistema de arranque tradicional que ha existido desde los primeros días de la informática. Es responsable de realizar el POST (*Power-On Self-Test*) para verificar los componentes básicos y luego carga el sistema operativo desde el dispositivo de arranque, que generalmente es una unidad de disco duro. El arranque puede configurarse en el BIOS para que sea desde un CD, DVD o una memoria USB.

Haciendo un pequeño paréntesis, el sistema de archivos puede ser FAT32 o NTFS. FAT32 (*File Allocation Table 32*) es un sistema de archivos más antiguo y sencillo, introducido en Windows 95 para reemplazar a sus predecesores FAT16 y FAT12. Funciona en una variedad de sistemas operativos, incluidos Windows, macOS, Linux y otros, lo que lo convierte en una opción ideal para dispositivos USB y discos externos que se usarán entre sistemas. FAT32 solo admite archivos de hasta 4 GB y particiones de hasta 2 TB, lo que puede ser una limitación significativa en dispositivos de almacenamiento modernos. FAT32 es útil para unidades USB o discos externos que necesiten ser accesibles en múltiples sistemas operativos y no tengan que almacenar archivos de gran tamaño.

Por su parte, NTFS (*New Technology File System*) es un sistema de archivos más avanzado, desarrollado por Microsoft y utilizado en versiones modernas de Windows. Admite permisos de archivos, compresión de datos, cifrado, recuperación ante fallos y soporte para archivos de mayor tamaño. Aunque Windows y algunas versiones de Linux soportan NTFS de forma completa, macOS por ejemplo solo lo monta como de solo lectura, a menos que se use software adicional. NTFS es más eficiente para manejar archivos grandes y proporciona un sistema más robusto y seguro, adecuado para almacenamiento en computadoras y servidores con Windows. NTFS es preferido en discos internos de computadoras con Windows y en cualquier situación donde se necesiten características de seguridad y recuperación avanzadas. Hasta aquí han sido descritas estas dos tecnologías. Le animo a que pueda profundizar en los diferentes tipos de sistemas de archivos que existen, las diferencias y la utilidad que pueden tener.

Una vez realizado el proceso de creación del USB de arranque, es necesario saber que también lo podemos utilizar como unidad *live,* es decir, que puede ejecutarse directamente sin necesidad de instalarse en el disco duro de un ordenador. Estas unidades permiten ejecutar Linux directamente desde el USB o DVD, ofreciendo una experiencia de prueba o un entorno de emergencia sin realizar cambios permanentes en el sistema de almacenamiento de la máquina. De esta manera, se pueden realizar pruebas del sistema operativo sin necesidad de instalar el mismo en el disco duro del PC. También puede cumplir la función en caso de que se requiera recuperar información de algún sistema defectuoso.

Por ejemplo, tenemos un sistema Windows con algún problema de arranque, entonces, lo que haríamos es insertar la USB de arranque Linux en este PC con Windows e iniciaríamos desde la memoria USB sin proceder a aún con la instalación de Linux. Solamente ejecutamos en ese momento el modo *live*, pues, al hacer esto, la información del disco con Windows también se reflejará en las unidades del sistema de *live* Linux, permitiéndonos realizar un respaldo de la información.

## Proceso de instalación

La instalación de Ubuntu Server es un proceso sencillo que sigue una serie de pasos bien definidos, con un asistente basado en modo texto. El medio de instalación puede ser desde un disco DVD o desde una memoria USB que es lo más usual hoy en día.

Ya con la memoria USB de arranque insertamos en el ordenador donde vayamos a instalar Linux Ubuntu Server. En el BIOS también debemos asegurarnos que el arranque se inicie desde la memoria USB. Al iniciar, se presentarán los siguientes pasos de instalación:

Para una mejor apreciación de las opciones de instalación se ha cambiado el color de la pantalla a azul, ya que generalmente es de fondo negro y texto blanco. Para la instalación escogemos la primera opción "Try or Install Ubuntu Server", la segunda opción "Test memory" permite verificar el estado de la memoria RAM del sistema. Esta prueba, basada en el *software* Memtest86+, se incluye en el menú de inicio del instalador de Ubuntu, como se puede apreciar en la pantalla o en el cargador GRUB, y es especialmente útil para diagnosticar problemas relacionados con fallos en la memoria del *hardware*. Es decir, podemos someter a una prueba de stress a la memoria RAM del ordenador, prueba que durará alrededor de 1 hora por cada 8 GB de memoria y, de esta manera, podremos verificar que nuestro servidor sea apto para realizar una correcta instalación y libre de posibles errores en memoria.

GNU GRUB (Grand Unified Bootloader), versión 2.12, es la herramienta de arranque o gestor de arranque que ha sido ampliamente utilizado en sistemas operativos basados en Linux y Unix. Su función principal es permitir que los usuarios seleccionen y carguen sistemas operativos instalados en su ordenador, así como

proporciona opciones avanzadas de configuración para el arranque. GRUB es especialmente conocido por su capacidad para gestionar configuraciones de arranque complejas en sistemas con múltiples particiones o sistemas operativos.

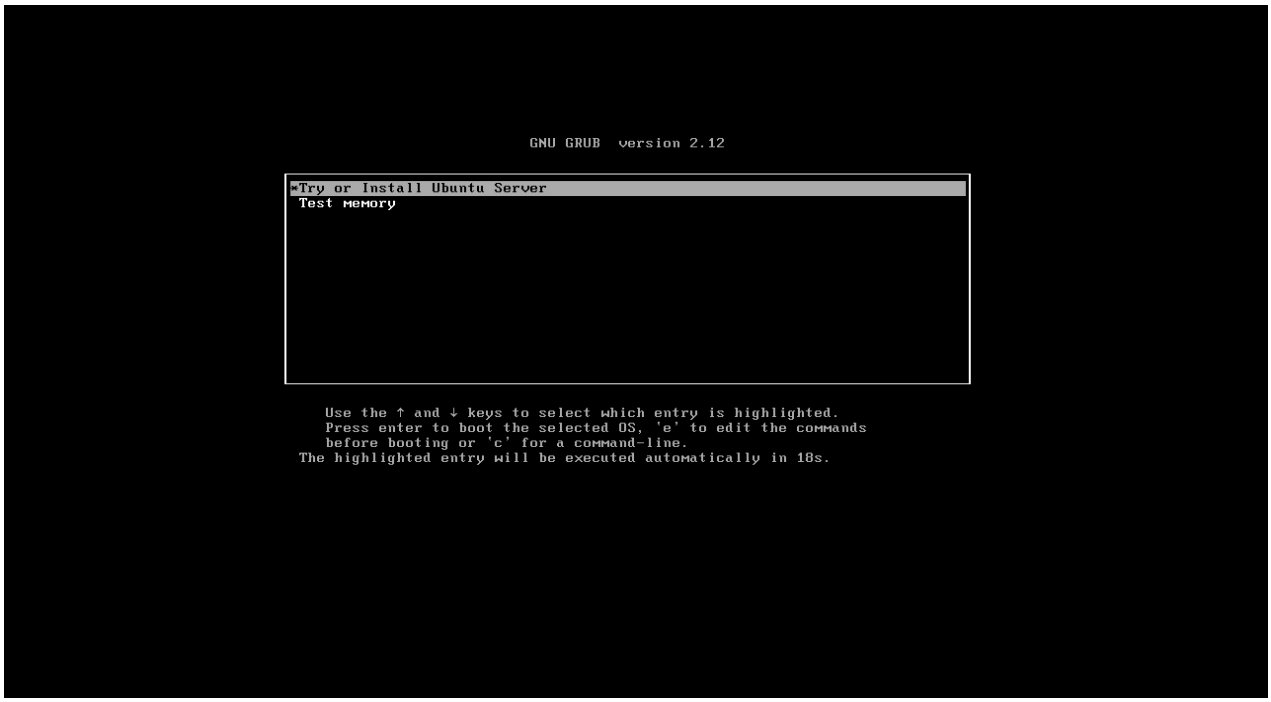



Figura 13. Pantalla de inicio de instalación de Linux Ubuntu Server

GRUB admite una variedad de sistemas operativos (Linux, Windows, macOS, etc.) y arquitecturas (x86, ARM, PowerPC, entre otras). Particularmente, la configuración se realiza de manera dinámica, es decir, esta versión, GRUB, carga su configuración desde un archivo (generalmente ubicado en /boot/grub/grub.cfg), lo que facilita la personalización de menús y opciones. Puede leer sistemas de archivos como FAT, NTFS, ext4, Btrfs, ZFS, entre otros, permitiendo acceder y cargar el *kernel* desde varias particiones; es por esta característica que podríamos instalar Linux y Windows y ejecutar cualquiera de los dos sistemas desde el menú de opciones.

El arranque es seguro (*Secure Boot*) es decir es compatible con configuraciones modernas que requieren firmas digitales para el proceso de arranque, lo que mejora la seguridad en sistemas UEFI.

Si tenemos instalado Linux podemos verificar la versión de GRUB con el comando:

```
root@svrlinux:~# grub-install --version ┘ Enter
grub-install (GRUB) 2.12-1ubuntu7.1
```

No es muy común actualizar la versión de GRUB, sobre todo si el sistema está funcionando correctamente y es lo razonablemente actual. Entonces, no sería necesario una actualización, bastaría con la versión que viene por defecto con la distribución de Linux; además, si la actualización la realizamos incorrectamente podríamos comprometer el arranque del sistema. Pero si usted ha actualizado el *hardware* o *firmware* (como UEFI) y desea corregir vulnerabilidades, arreglar errores o utilizar un kernel moderno con configuraciones avanzadas, posiblemente sea necesario actualizar GRUB.

Al seleccionar la instalación de Ubuntu Server, se inicia y se cargan los *scripts* de instalación de Linux, como puede apreciarse en la siguiente pantalla.

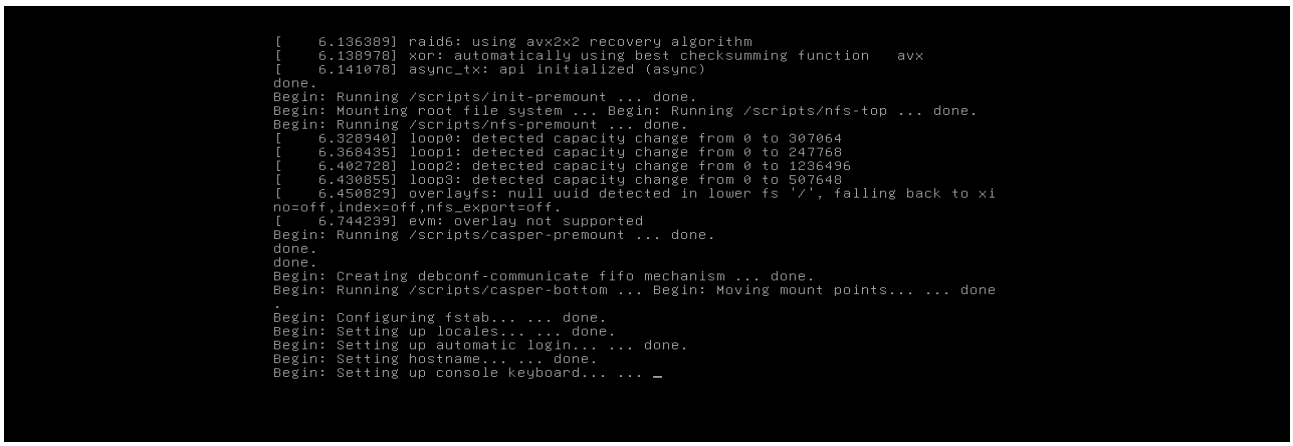



Figura 14. El instalador comienza a ejecutarse.

A continuación, seleccionamos el Idioma para nuestro sistema operativo.

En seguida se presenta una ventana para ver si la actualización del instalador está disponible, entonces, usted puede elegir si desea actualizar al instalador nuevo o continuar sin actualizar. Aquí puedo recomendar que saltemos este paso y simplemente continuemos sin actualizar, ya que deberá tener una buena conexión a Internet y también este proceso puede tomar un tiempo. Lo mejor es saltar este paso y luego, cuando ya esté instalado el sistema, se procede a actualizar el mismo.

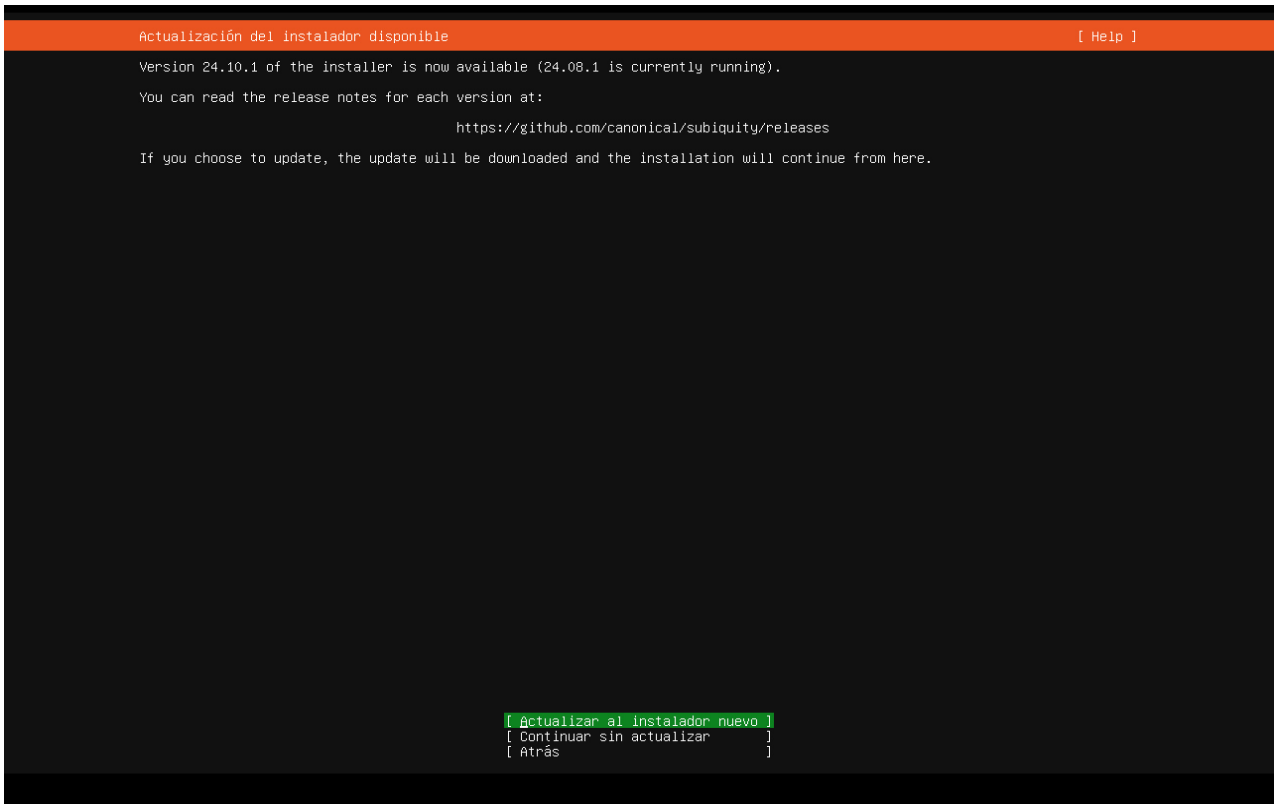



Figura 15. Pantalla de bienvenida y selección del idioma

Luego, es importante seleccionar el lenguaje de nuestro teclado; generalmente está en español, pero si el teclado no contiene la letra ñ, seguramente el teclado está en inglés.

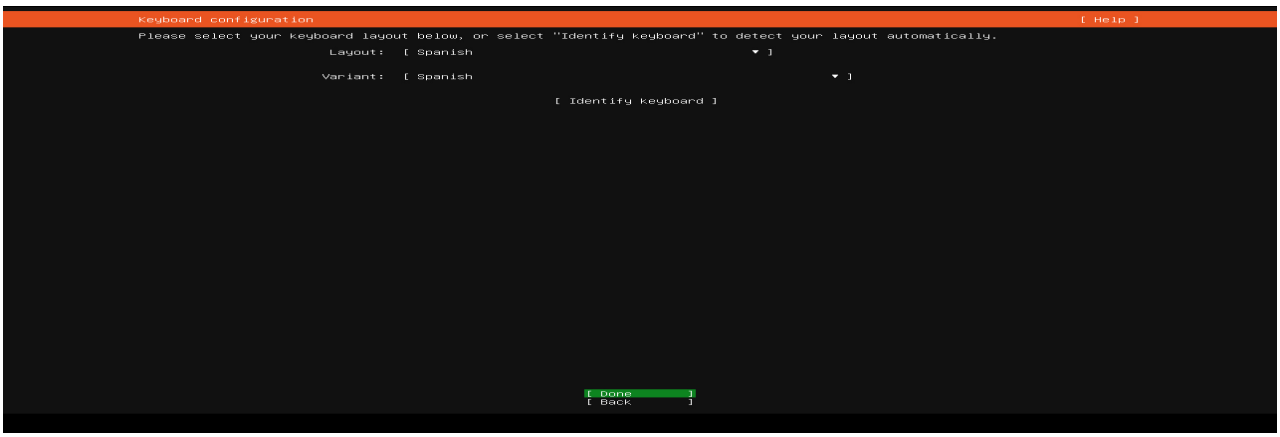



Figura 16. Pantalla donde se indica si la actualización del instalador está disponible

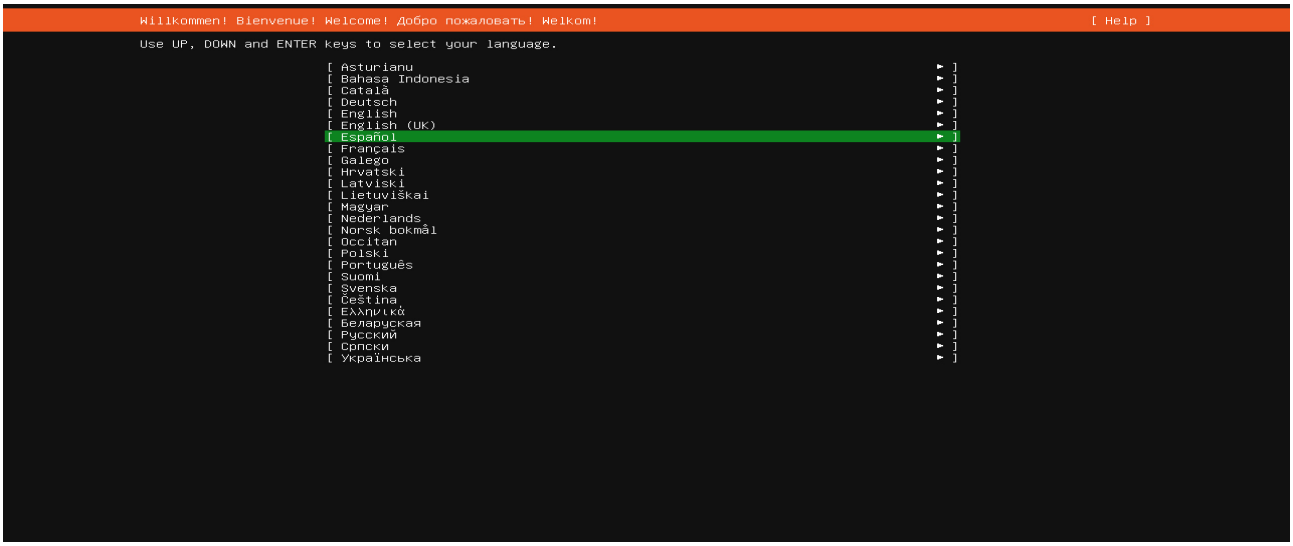



Figura 17. Se solicita seleccionar el idioma del teclado.

La siguiente ventana es para escoger el tipo de instalación: Ubuntu Server o Ubuntu Server (*minimized*), además, presenta una opción adicional para buscar controladores de terceros, mismos que están sujetos a los términos de licencia especificados en su documentación; estos podrían ser, antes que código abierto, propietarios, como es el caso de controladores gráficos de NVIDIA o ciertos *drivers* de tarjetas de red de Broadcom.

Ubuntu Server (*minimized*) es una versión que minimiza los paquetes preinstalados para reducir el tamaño del sistema operativo, mejorar el rendimiento y reducir la superficie de ataque desde el punto de vista de la seguridad.

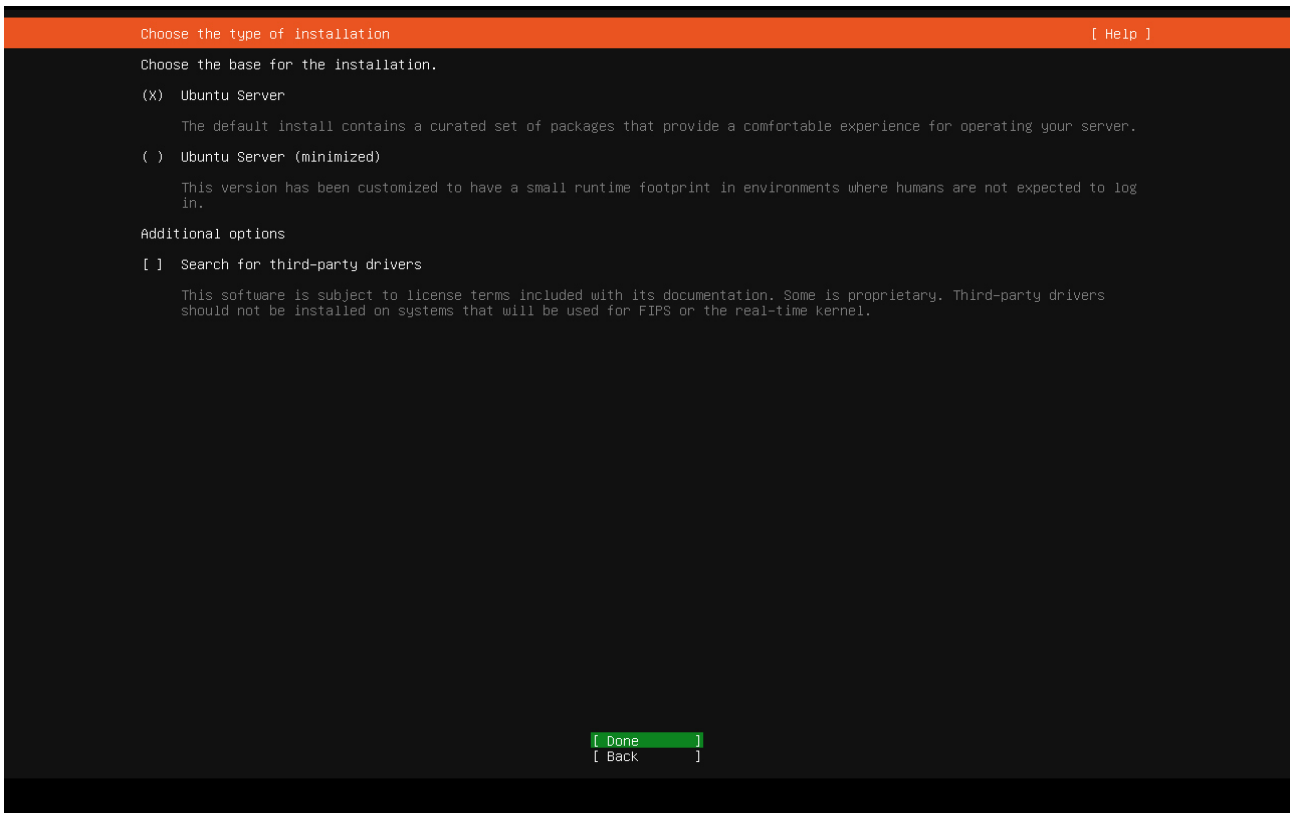



Figura 18. Seleccionamos Ubuntu Server, marcando con una X.

La instalación predeterminada de (X) Ubuntu Server incluye un conjunto seleccionado de paquetes que brindan una experiencia cómoda para operar su servidor. El texto traducido al español de la segunda opción minimizada dice: "Esta versión ha sido personalizada para tener una pequeña huella de ejecución en entornos donde no se espera que los humanos inicien sesión". Es decir, como ya lo había manifestado anteriormente, esta versión está diseñada para entornos donde se requiere un sistema operativo ligero y eficiente.

Además, se advierte que estos controladores no son recomendables para sistemas que utilicen configuraciones sensibles como el FIPS (Federal Information Processing Standards) o núcleos en tiempo real (*real-time kernel*), debido a posibles incompatibilidades o problemas de seguridad.

A continuación, configure al menos una interfaz que este servidor pueda usar para comunicarse con otras máquinas y que, preferentemente ofrezca acceso suficiente para las actualizaciones.

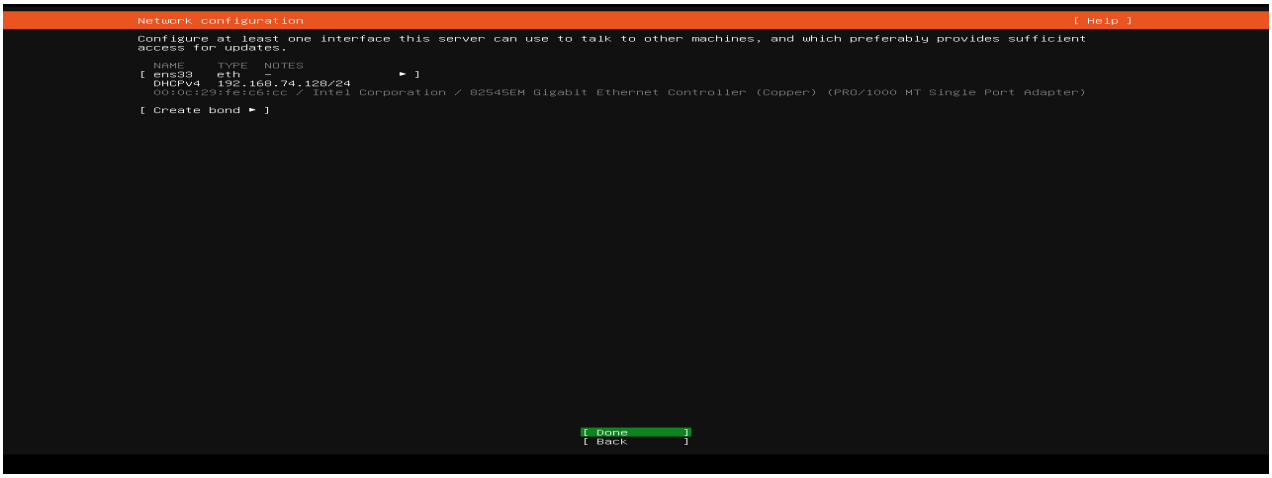



Figura 19. Pantalla de configuración de la dirección IP estática o dinámica mediante DHCP.

La configuración de la red puede realizarse a través de DHCP si es que existe un servidor DHCP mismo que asignara una dirección IP dinámicamente o también puede realizarse manualmente, seleccionando o accediendo a la interfaz ens33 eth0 e ingresando los siguientes datos, **por ejemplo**:

- **Subnet**: 192.168.1.0/24 (define la red a la que pertenece el servidor, donde /24 indica la máscara de red: 255.255.255.0).
- **Address**: 192.168.1.10 (la dirección IP estática que tendrá su servidor).
- **Gateway**: 192.168.1.1 (el *router* o dispositivo que proporciona acceso a otras redes, como Internet).
- **Name servers**: 8.8.8.8 (servidores DNS que resuelven nombres de dominio).

En todo caso, si usted no es el administrador de la red, puede consultar con el mismo para que le asigne una dirección IP y detalles del DNS. Caso contrario, generalmente se suele reservar entre 10 y 20 o más direcciones IP dependiendo de su organización, que puede ser de uso exclusivo para equipos de capa3 y direcciones IP en servidores. La práctica de reservar direcciones IP dentro de un rango específico tiene como objetivo garantizar que ciertos dispositivos o servicios críticos en una red, como servidores o equipos de capa 3 (*routers*, *switches* administrables, *firewalls*, etc.), tengan direcciones estáticas que no entren en conflicto con las asignadas dinámicamente por DHCP. Esta estrategia es común en configuraciones de redes empresariales y organizacionales.

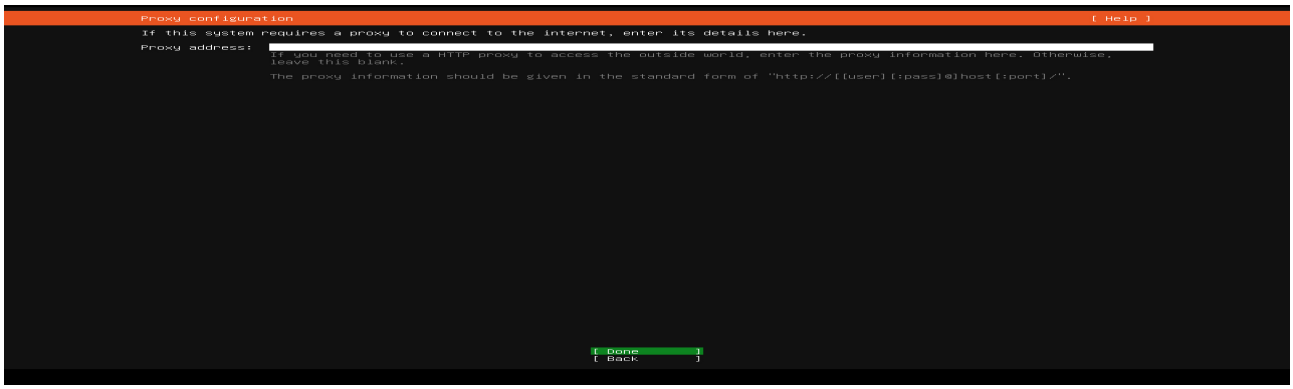



Figura 20. Si el sistema requiere un proxy para conectarse a internet, ingrese sus detalles aquí.

Reservar direcciones IP es una buena práctica, esto ayudará a evitar conflictos y problemas de conexión, ya que podremos asignar IPs estáticas tanto a servidores como a dispositivos de red. Esto previene que dichas direcciones sean reasignadas automáticamente por DHCP y se pueda perder conectividad. Además, simplifica la administración al facilitar la identificación rápida de dispositivos dentro de la red. También garantiza la compatibilidad con servicios críticos, como servidores web o bases de datos, que requieren direcciones fijas para su correcto funcionamiento.

En la siguiente pantalla se especifican los repositorios en Linux Ubuntu. Estos son servidores que almacenan *software*, actualizaciones y parches necesarios para el sistema operativo. Si usted conoce alguna dirección web o específicamente un *mirror* de paquetes de Ubuntu o simplemente un repositorio de este mismo, puede ingresar la dirección en este campo "Mirror address:"; esto se refiere a que un servidor aloja copias de los archivos de los repositorios del sistema operativo Ubuntu. Si no conoce la dirección es mejor dejarla por defecto, sin embargo, luego de la instalación se puede también cambiar esta dirección, por ejemplo, dentro del archivo /etc/apt/sources.list localice el texto http://us.archive.ubuntu.com/ubuntu y reemplace esto con http://archive.ubuntu.com/ubuntu, luego reinicie el sistema; es recomendable, antes de realizar el cambio, respaldar dicho archivo. El archivo /etc/apt/sources.list contiene las ubicaciones de los repositorios. Es aquí donde puedes agregar, eliminar o modificar las fuentes desde las cuales se descarga el *software*.

Aquí tiene espejos oficiales de archivos para Ubuntu, que proporcionan repositorios y archivos de todo el *software* de la distribución. Hay como 610 espejos registrados para Ubuntu. https://launchpad.net/ubuntu/+archivemirrors

También existen comandos relacionados a este propósito: #sudo apt update, que actualiza la información sobre los paquetes disponibles en los repositorios, #sudo apt upgrade, que actualiza los paquetes instalados a las últimas versiones disponibles y #sudo add-apt-repository, que agrega repositorios adicionales como PPAs (Personal Package Archives), útiles para instalar *software* no incluido en los repositorios oficiales.

Un *mirror* es un servidor ubicado en algún lugar del mundo que contiene una copia del repositorio oficial de Ubuntu. Para Ecuador, la dirección de espejo predeterminada es: http://ec.archive.ubuntu.com/ubuntu, pero podríamos cambiarla a un repositorio que cumpla con nuestras necesidades y que de pronto sea más accesible, con paquetes más actualizados o que no estén bloqueados, que tengan mayor velocidad de descarga, desde alguna otra ubicación por ejemplo desde la dirección principal: http://archive.ubuntu.com/ubuntu, o de algún otro país http://es.archive.ubuntu.com/ubuntu de España, o quizá algún *mirror* local sería una opción excelente para acelerar las descargas desde alguna Universidad, por ejemplo: http://mirror.espoch.edu.ec/ubuntu. Por lo general, nos mantenemos con la dirección que venga por defecto.

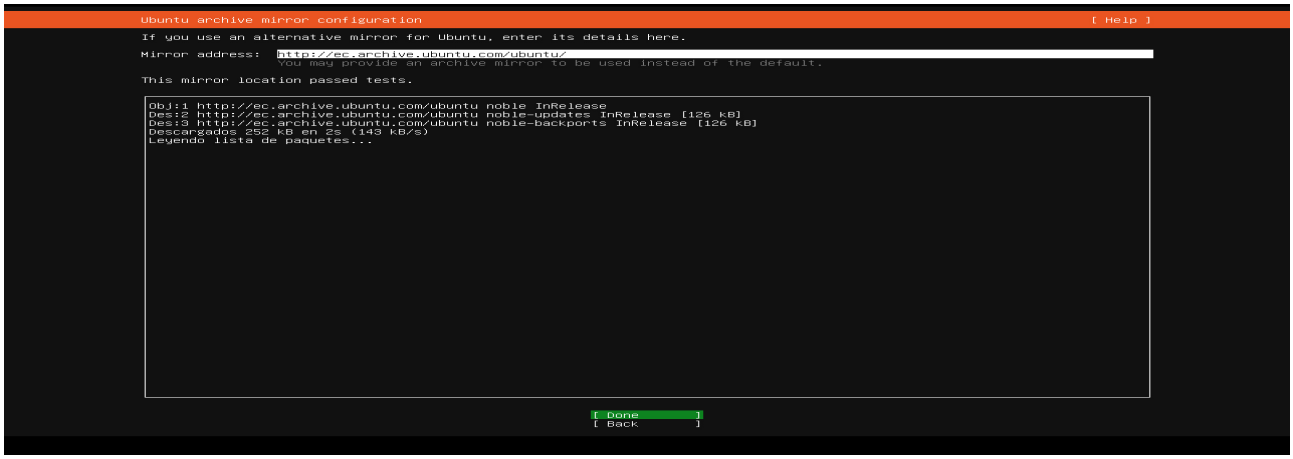



Figura 21. Configuración del espejo de archivos de Ubuntu

A continuación, tenemos la configuración del disco local: se debe prever cuánto espacio en disco es necesario para el correcto funcionamiento de nuestras aplicaciones y servicios. El nombre asignado al primer disco detectado es /dev/sda en el sistema por el *kernel* de Linux. En los sistemas basados en Linux, los discos y particiones se representan como archivos especiales en el directorio /dev (dispositivos). La nomenclatura sd indica que es un dispositivo SCSI o compatible, que incluye discos SATA y NVMe en muchos casos. Si hay más discos, se nombrarán secuencialmente (sdb, sdc, etc.). Por su parte local, *disk* describe el tipo de disco. En este caso, se indica que es un disco local, es decir, un disco físico directamente conectado al equipo, no una unidad remota o en red. El valor 20.000G indica el tamaño total del disco en gigabytes (GB). Aquí el valor es 20 GB, lo que equivale a aproximadamente 20000 MB (megabytes). Podemos utilizar el disco entero, marcando con una X "Use an entire disk", caso contrario, se podrán generar particiones en disco a parte de la unidad "/" raíz, podríamos tener una partición en "/home" para los usuarios, o una partición en "/var/www/" para los sitios web.

Existe un comando que luego de la instalación se podría utilizar y que permite conocer sobre la partición y volúmenes del disco duro, el comando es el siguiente:

```
root@svrlinux:/# df -hT ┘ Enter
```

En la siguiente dirección web oficial de ORACLE podemos consultar más detalles sobre este comando https://docs.oracle.com/cd/E26921_01/html/E25809/spmonitor-6.html. El espacio en el disco utilizable que informa df refleja sólo el 90 % de la capacidad total, ya que las estadísticas de informe permiten dejar un margen del 10 % sobre el total de espacio disponible. En general, este margen permanece vacío para un mejor rendimiento y, lógicamente, en la documentación oficial de Linux Ubuntu https://manpages.ubuntu.com/manpages/noble/es/man1/df.1.html df muestra el nivel de uso del sistema de archivos que contiene a cada archivo que se especifica como argumento.

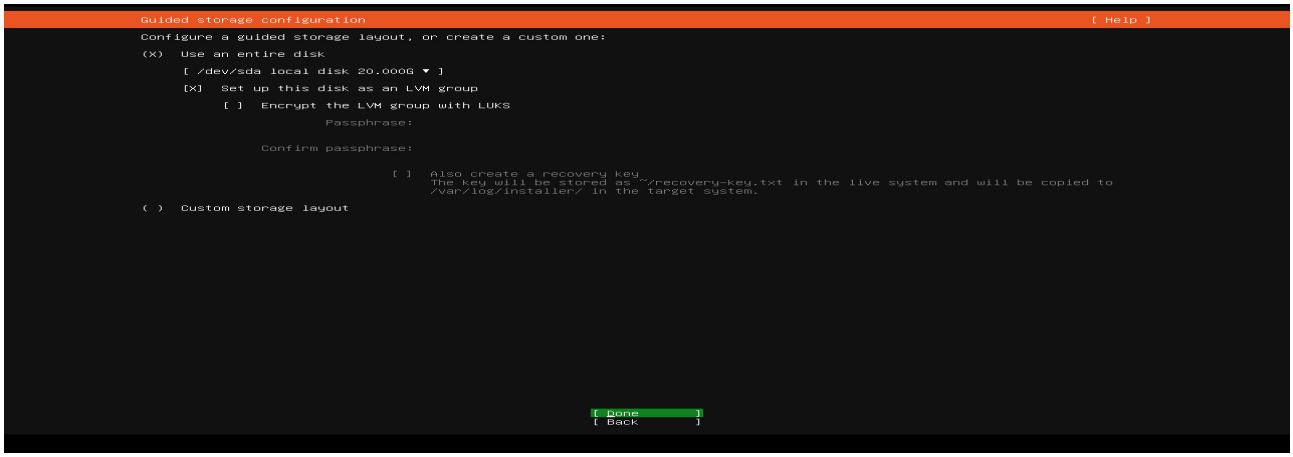



Figura 22. Configuración guiada de almacenamiento

Si no se proporciona ningún nombre de archivo, se muestra el nivel de uso del espacio en todos los sistemas de archivos montados. Por defecto, el espacio en disco se muestra en bloques de 1K, a menos que se establezca la variable de entorno POSIXLY_CORRECT, en cuyo caso se mostrará en bloques de 512 bytes.

Luego tenemos una opción que dice grupo LVM (Logical Volume Manager), la misma que especifica que es una tecnología de gestión de volúmenes lógicos en Linux y que permite una administración flexible de los discos duros y sistemas de almacenamiento. A diferencia de los esquemas tradicionales de particionamiento (como MBR o GPT), LVM permite crear volúmenes lógicos que pueden extenderse, reducirse y reorganizarse de manera más dinámica. Una de las ventajas de LVM es la flexibilidad, ya que se puede agregar más discos al grupo de volúmenes sin tener que reorganizar las particiones existentes.

El redimensionamiento de los volúmenes lógicos puede crecer o reducirse según sea necesario, sin tener que reformatear el disco, y la gestión de almacenamiento es más eficiente debido a que LVM permite crear instantáneas (*snapshots*) de los volúmenes, lo que es útil para copias de seguridad. En el capítulo de línea de comandos Linux ampliaré un poco más el uso de esta tecnología.

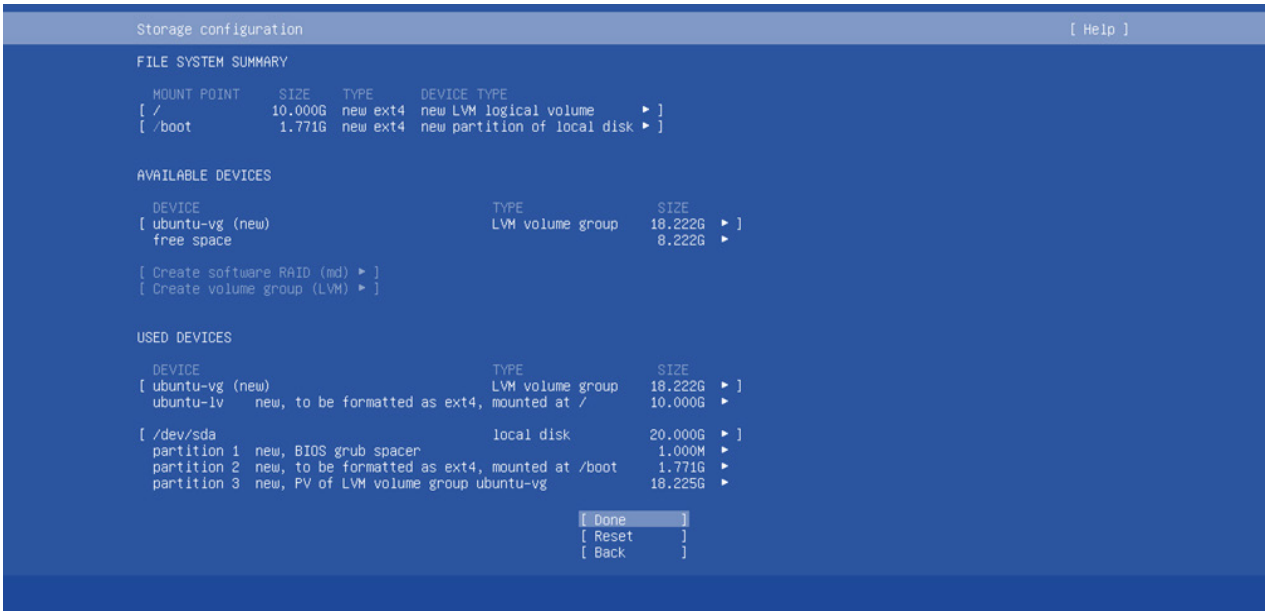



Figura 23. Configuración del espacio en disco

En la pantalla anterior se presenta nuevamente un resumen de los dispositivos, el tipo y el tamaño, se recalca que el disco local tiene 20.000G y 3 particiones por defecto: partición 1 para el GRUB BIOS de 1.000M, la partición 2 para formatearse como ext4, y montado en /boot con 1.771G; finalmente, la partición 3 hace referencia al PV del grupo de volúmenes LVM ubuntu-vg, es decir aquí, "PV" significa "*Physical Volume*" o "volumen físico", que es el componente físico del sistema de almacenamiento LVM. "ubuntu-vg" se refiere al "grupo de volúmenes" (*Volume Group*) llamado "ubuntu-vg", en el cual se almacenan los volúmenes lógicos. Ext4 es el Fourth Extended Filesystem, por sus siglas en inglés, un sistema de archivos ampliamente utilizado en sistemas operativos Linux. Es la evolución de sus predecesores ext2 y ext3 y fue diseñado para mejorar el rendimiento, la fiabilidad y la capacidad de manejo de grandes volúmenes de datos.

El *BIOS grub spacer* se refiere al espacio reservado para GRUB en discos con BIOS (no UEFI). En caso de disponer de más unidades de disco en nuestro ordenador, cada uno aparecerá como un dispositivo separado, como: /dev/sda para el primer disco, /dev/sdb será el segundo disco, /dev/sdc el tercer disco, y así sucesivamente; el instalador de Ubuntu nos permitirá seleccionar particiones existentes o crear nuevas. Para dispositivos NVMe (un tipo de SSD rápido) aparecerán con nombres como: /dev/nvme0n1 para el primer disco, /dev/nvme0n1p1 será la primera partición de ese disco; queda para el lector investigar cómo saldría la nomenclatura por ejemplo para discos RAID, unidades o medios externos, virtualización o almacenamiento iSCSI.

En la siguiente pantalla simplemente seleccionamos "Continuar" para iniciar el proceso de instalación, se perderán los datos de los discos que seleccione para formatear. Y usted no podrá volver a esta pantalla ni a una anterior una vez que haya comenzado la instalación.

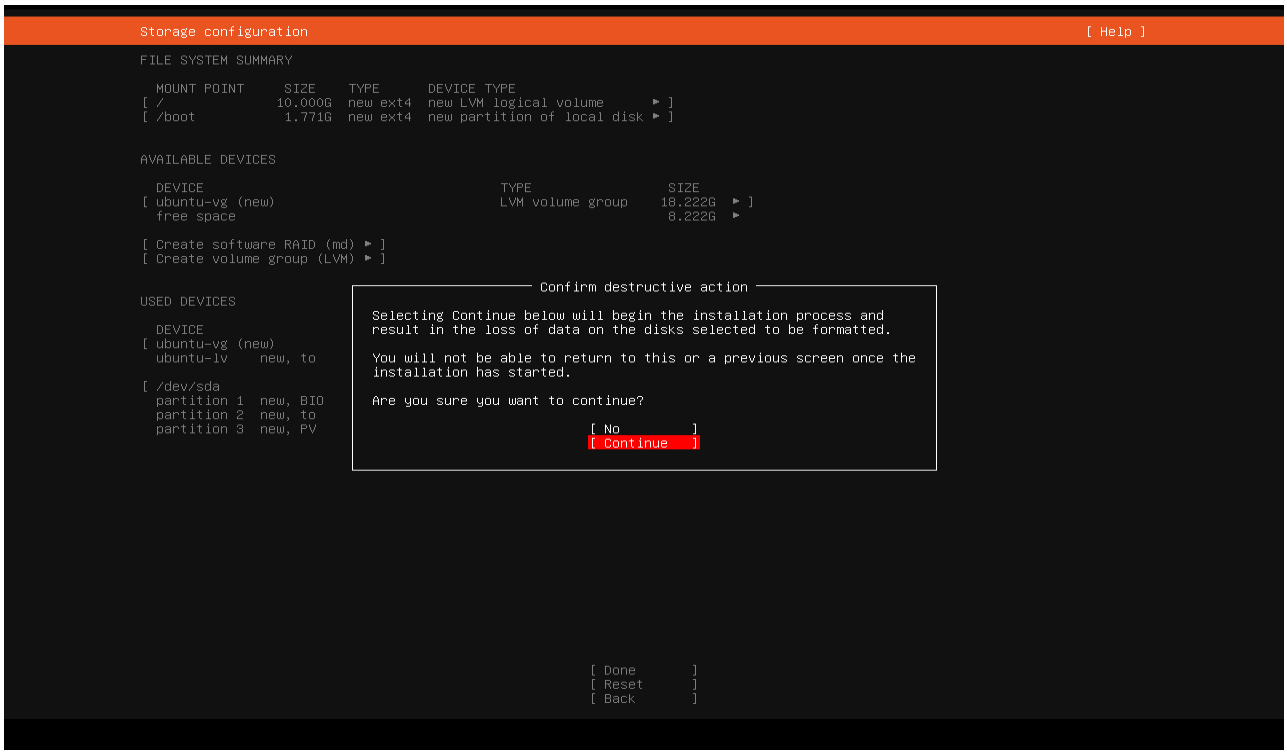



Figura 24. Confirmación de la estructura del disco sólido, el disco será formateado.

No menos importante es la configuración del perfil de usuario: podemos ingresar el nombre del usuario, seguido del nombre del servidor, un nombre de usuario (*username*) y una contraseña. Al arrancar el sistema operativo Ubuntu, se le solicitará el usuario y la contraseña. En caso de olvidar la contraseña, existen maneras de poder recuperar la misma, así, la más común es utilizar un Live USB de Ubuntu, montar la partición raíz con el comando mount /dev/sda1 /mnt, acceder al sistema instalado con el comando chroot /mnt y seguidamente cambiar la contraseña con *passwd* usuario y reiniciar el sistema.

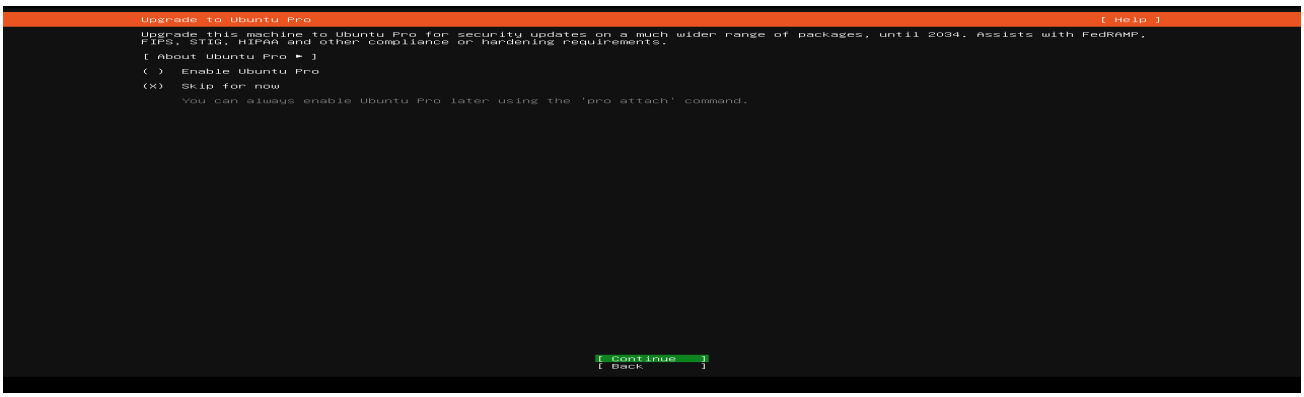



Figura 25. Pantalla de configuración del perfil de usuario y nombre del servidor

La pantalla siguiente nos indica que puede actualizar esta máquina a Ubuntu Pro para obtener actualizaciones de seguridad en una gama mucho más pro y amplia de paquetes, hasta 2034. Ayuda con FedRAMP, FIPS, STIG, HIPAA y otros requisitos de cumplimiento o fortalecimiento. Si no desea ninguno de estos servicios, simplemente saltarse este paso.

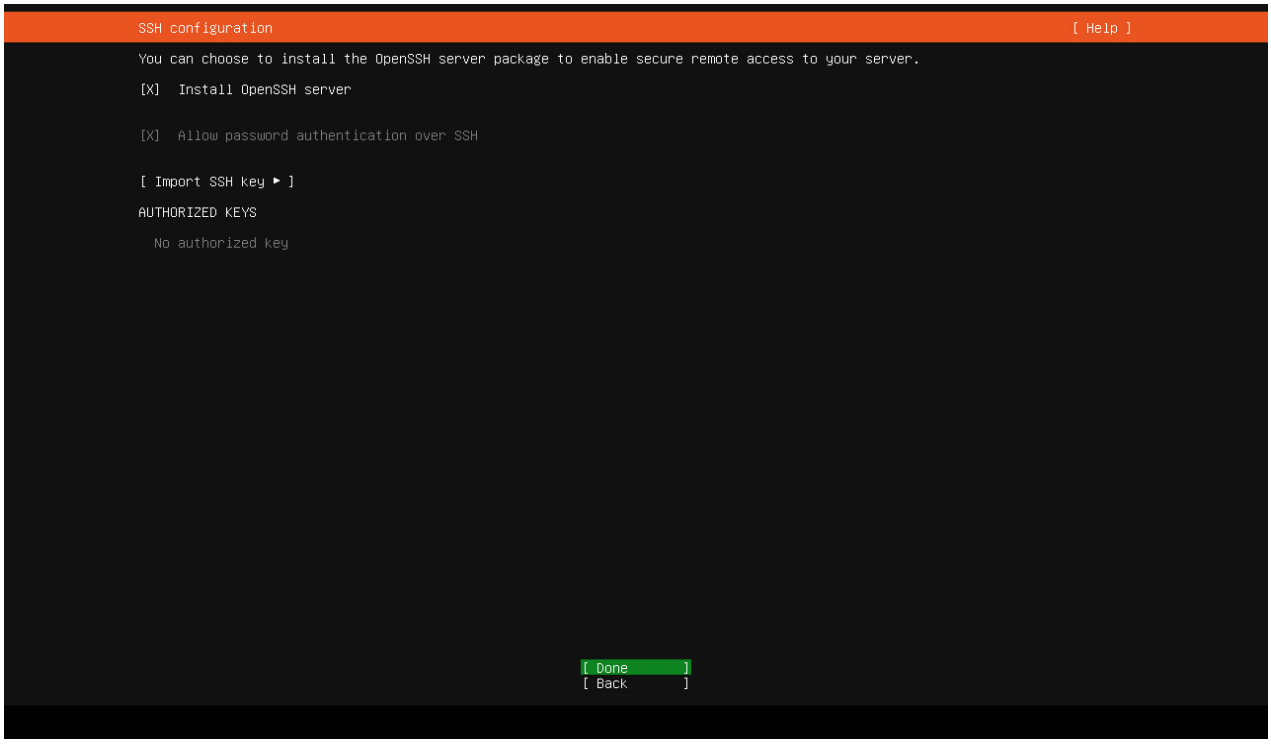



Figura 26. Actualizar a Ubuntu Pro.

En la pantalla siguiente de la instalación de Linux podemos instalar directamente OpenSSH server, marcando con una X, lo que nos permitirá tener acceso remoto vía SSH a través del puerto 22. Se recomienda activar esta opción, ya que es muy utilizada para fines de gestión remota segura del sistema operativo.




Figura 27. Configuración de SSH e instalación de OpenSSH *server*

A continuación, aparece una serie de servicios que podemos instalar, marcando con un (*) los que necesitemos para nuestro servidor.

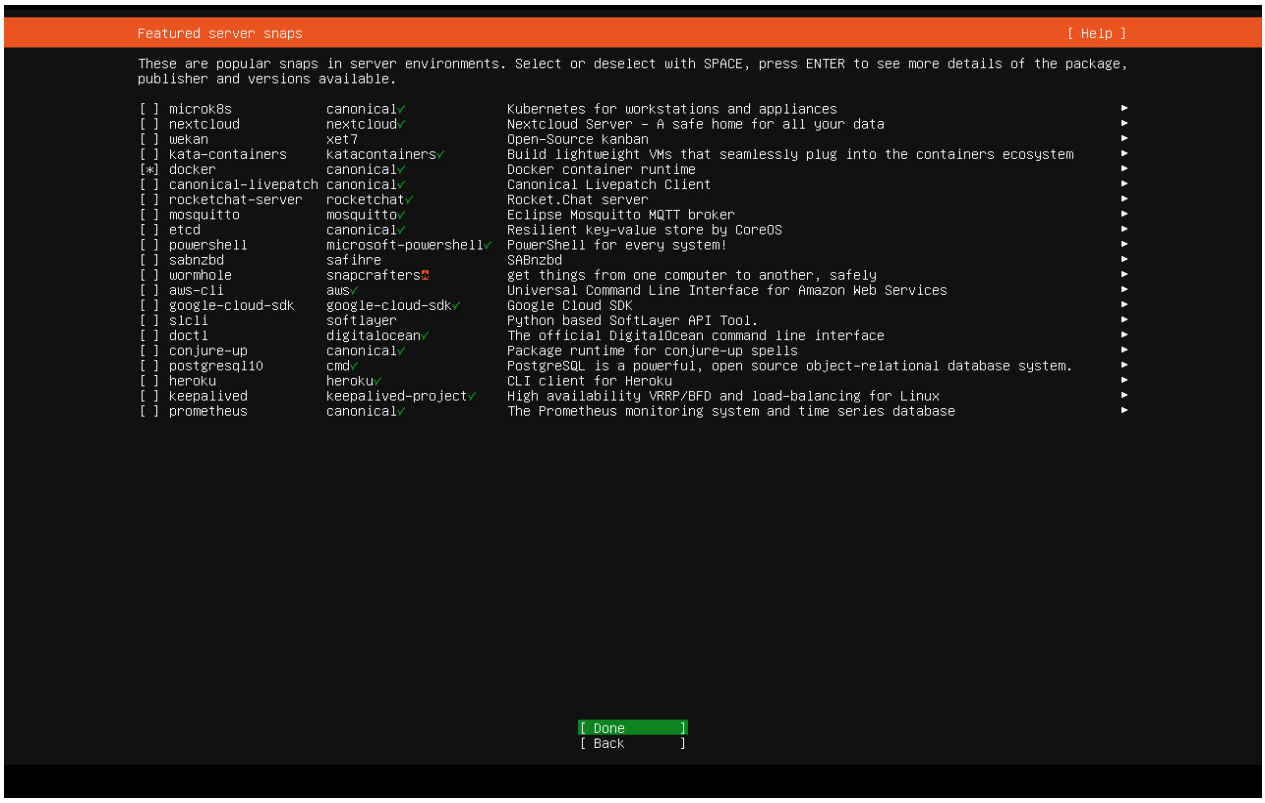



Figura 28. Diferentes aplicaciones que podemos instalar o no en Linux Ubuntu Server

Por ejemplo, si seleccionamos *'docker',* estaremos utilizando una plataforma abierta para desarrollar, enviar y ejecutar aplicaciones. Docker le permite separar sus aplicaciones de su infraestructura para que pueda entregar *software* rápidamente. Con Docker, puede administrar su infraestructura de la misma manera que administra sus aplicaciones. Al aprovechar las metodologías de Docker para enviar, probar e implementar código, puede reducir significativamente la demora entre la escritura del código y su ejecución en producción. Docker ofrece la posibilidad de empaquetar y ejecutar una aplicación en un entorno poco aislado llamado contenedor. El aislamiento y la seguridad le permiten ejecutar muchos contenedores simultáneamente en un *host* determinado; para más información, puede acudir al sitio web oficial de Docker (Inc., 2024)

En este punto, particularmente recomiendo no seleccionar ninguno de estos servicios, ya que primero tomará tiempo en la instalación, es decir, a más aplicaciones, más tiempo, dependiendo de la velocidad de su internet, además de que luego pueden instalarse desde una terminal, Luego, no es recomendable seleccionar al azar

los servicios a menos que conozcamos estrictamente lo que vayamos a utilizar, esto nos evitará dolores de cabeza al darnos cuenta de que hemos instalado paquetes innecesarios que pueden ralentizar nuestro sistema operativo o abrir interfaces que pueden vulnerar el sistema.

El proceso de instalación tomará unos minutos dependiendo de la velocidad de internet y de su ordenador, como ya lo mencioné, y es recomendable que, si no desea esperar mucho tiempo, no seleccione estas aplicaciones e instalarlas luego de instalar el sistema operativo. Lo primero que debe realizar luego de una instalación es actualizar los repositorios del sistema y actualizar Ubuntu con los comandos update y upgrade, respectivamente, como se analizó anteriormente.

Recordemos que Linux Ubuntu Server se instala por defecto en modo texto, es decir, usted verá únicamente una terminal, pero si lo desea puede también instalar un modo gráfico como KDE o GNOME, realizando los siguientes pasos, donde, personalmente, recomiendo GNOME como entorno gráfico. Los comandos necesarios son los siguientes:

Actualizamos el servidor

root@svrlinux:/# apt update ↵ Enter

Actualizamos los paquetes disponibles

root@svrlinux:/# apt upgrade -y ↵ Enter

Instalamos el gestor de paquetes tasksel

root@svrlinux:/# apt install tasksel ↵ Enter

Podemos reiniciar el Sistema

root@svrlinux:/# reboot ↵ Enter

Ejecutamos tasksel

root@svrlinux:/# sudo tasksel

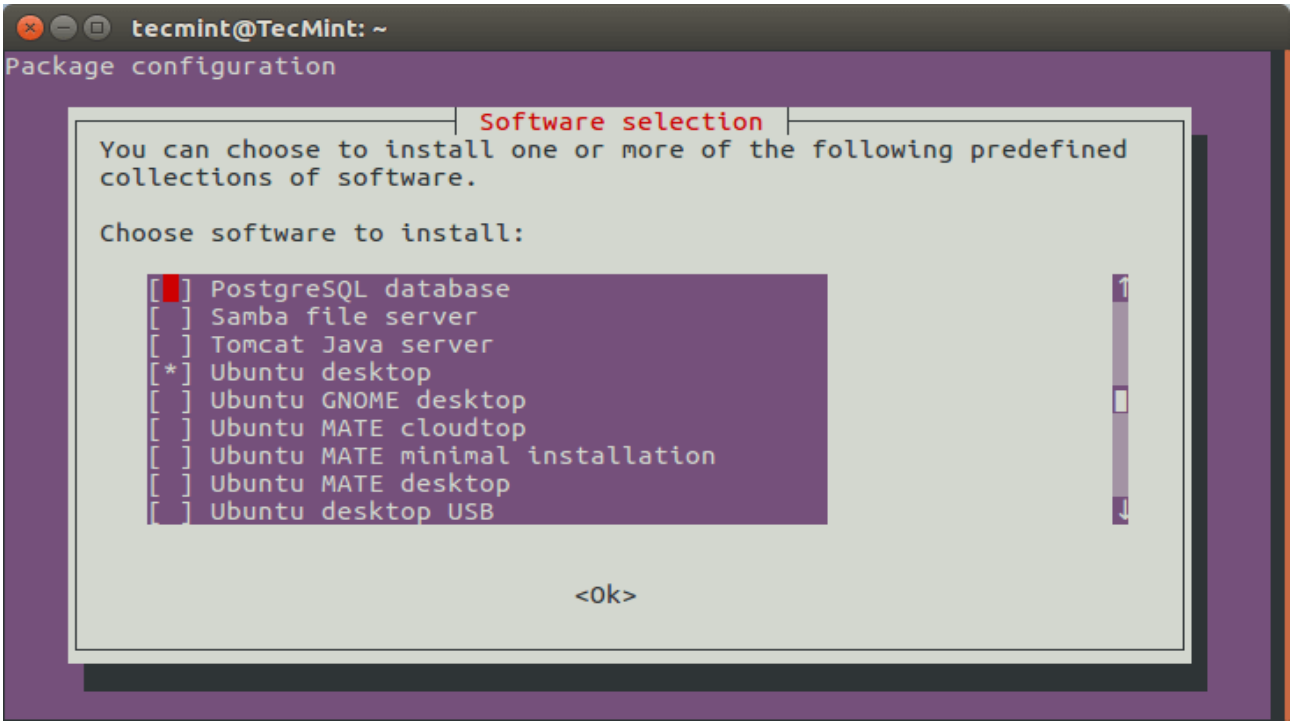



Figura 29. Ventana de la aplicación de tasksel, para seleccionar el entorno gráfico u otras aplicaciones

Seleccionamos Ubuntu Desktop para instalar el entorno completo de GNOME, pulsamos enter y Tasksel instalará todos los paquetes requeridos de una sola vez. Una vez completada la instalación reiniciamos el sistema. Otra forma de instalar directamente desde la línea de comandos es como se indica a continuación:

```
Podemos instalar GNOME completo

root@svrlinux:/# tasksel install ubuntu-desktop ┘ Enter
```

La ventaja de un entorno gráfico es que pueden disponerse aplicaciones que en entornos de solo texto no existen, por ejemplo, EtherApe es una herramienta de monitoreo de tráfico de red/sniffer de paquetes, de código abierto.

## Errores comunes durante la instalación

Incluso con una herramienta tan apropiada para la instalación de Ubuntu Server como Subiquity, existen puntos críticos donde la instalación puede fallar o quedar mal configurada para un entorno de producción. Subiquity ofrece una interfaz basada en texto (TUI) y automatización a través de *autoinstall* usando *cloud-init*. Desarrollado por Canonical, es el estándar para la instalación rápida de servidores, configuración de "primer arranque" y el *backend* del instalador de escritorio de Ubuntu.

A continuación, se describen los errores más comunes divididos por categorías.

**1. Errores de particionado y almacenamiento**

No dejar espacio para el volumen de Docker/LXC: Un error clásico es asignar todo el disco a la raíz (/) sin considerar que Docker, por defecto, guarda todas sus imágenes y volúmenes en /var/lib/docker. Si ese directorio se llena, el sistema operativo colapsará.

Solución: usar LVM (Logical Volume Management). Esto permite redimensionar particiones en caliente en caso de quedarse sin espacio en el futuro.

Olvidar la partición EFI: En sistemas modernos (UEFI), si no crea una partición de tipo ESP (EFI System Partition), el sistema simplemente no arrancará después de la instalación.

**2. Errores de red (El "fantasma" de la IP dinámica)**

Instalar con DHCP en un servidor: Si deja que el instalador tome una IP automática, el servidor funcionará hoy, pero, tras un reinicio, la IP podría cambiar. Esto romperá todas las conexiones de sus contenedores y el acceso SSH.

Solución: Configurar una IP Estática desde el proceso de instalación o mediante Netplan inmediatamente después de la instalación.

Falta de resolución DNS: A veces la red funciona (puede hacer ping a una dirección IP), pero no podrá descargar paquetes porque no se configuró los servidores DNS correctamente (como por ejemplo 8.8.8.8).

**3. Errores de seguridad y acceso**

No activar SSH durante la instalación: Ubuntu Server es normalmente para una administración remota. Si se olvida marcar la casilla "Install OpenSSH Server", se tendrá que conectar un monitor y teclado físico al servidor solo para instalarlo después.

Nombre de usuario conflictivo: Intentar crear un usuario llamado *root* o *admin*. Ubuntu reserva ciertos nombres o tiene políticas de seguridad que sugieren usar un usuario personal y luego escalar con sudo.

**4. Errores de *hardware* y *kernel***

Virtualización desactivada en BIOS/UEFI: Este es el error "invisible". Si va a usar Docker o LXC y la tecnología de virtualización (Intel VT-x o AMD-V) está desactivada en la BIOS, el *kernel* no podrá gestionar correctamente el aislamiento de procesos de bajo nivel.

Síntoma: El instalador termina bien, pero al intentar ejecutar contenedores se recibe errores de permisos o de recursos no disponibles.

**5. Corrupción del medio de instalación**

Errores durante la instalación de Ubuntu: Uno de los errores más frustrantes ocurre cuando la conexión a internet es inestable durante la descarga de la ISO de Ubuntu. Un solo bit que se pierda o se altere en un archivo de varios gigabytes puede hacer que el instalador se cierre inesperadamente, devuelva errores de "archivo no encontrado" o se detenga al intentar instalar el *kernel*. Incluso si el sistema llegara a instalarse, el sistema puede volverse inestable.

Solución técnica: Verificación MD5/SHA256. Antes de grabar la ISO en una memoria USB, es una práctica obligatoria verificar su integridad. Esto se hace comparando el "Hash" (la huella digital) del archivo descargado con el valor oficial que proporciona Canonical.

# CAPÍTULO III LÍNEA DE COMANDOS LINUX

## Contenido del capítulo III

- Estructura y sistema de archivos
- Directorios y archivos importantes
- Comandos esenciales
- Ejercicios propuestos

## Estructura y sistema de archivos

Los sistemas operativos basados en Unix, como por ejemplo GNU Linux, FreeBSD inclusive macOS (que también es basado en Unix y al igual que Android derivado de Linux y iOS derivado de Mac OS X), son sistemas que organizan sus datos en archivos y directorios formando una estructura de árbol llamada el *filesystem,* cuyo directorio raíz se llama 'root' y está siempre denotado por el símbolo "/".

Linux es un sistema de explotación enteramente orientado a archivos. Todo (o casi todo) se representa por un archivo, tanto los datos como los periféricos o incluso los medios de comunicación (*sockets*, *named pipes*, etc.). El sistema de archivos es el corazón de todo sistema Linux/UNIX" (BARDOT, 2022). A continuación, vamos a emplear el comando "ls -l" ubicándonos desde el directorio raíz para presentar precisamente un listado de directorios del sistema operativo Linux Ubuntu Server.

```
root@svrlinux:/# ls -l ┘Enter
```

Ubuntu, al igual que la mayoría de las distribuciones Linux, generalmente utiliza sistemas de archivos como ext4, Linux sigue el estándar de jerarquía estándar del sistema de archivos (FileSystem Hierarchy Estándar FHS) de los sistemas tipo UNIX, y es la cuarta generación del sistema extensible con una capacidad de manejar grandes volúmenes y archivos grandes; tiene además un control eficiente de errores y prevención de fragmentación del disco. Básicamente, está organizado en directorios con funciones específicas como /home para los archivos de usuario, /etc, /bin, /lib para binarios esenciales.

Por otro lado, Windows utiliza un diseño derivado de MS-DOS, utiliza un enfoque basado en múltiples unidades. Cada disco o partición es identificado por una letra o unidad como C:\, D:\, etc., las cuales son independientes entre sí. Está básicamente organizado en C:\Windows que contiene el núcleo del sistema operativo, C:\Program Files y C:\Program Files (x86), donde residen directorios de los programas y C:\Users, donde se almacenan otros usuarios, documentos y otros datos.

```
root@svreditorial:/# ls -l
total 4194388
lrwxrwxrwx   1 root root          7 abr 22  2024 bin -> usr/bin
drwxr-xr-x   2 root root       4096 feb 26  2024 bin.usr-is-merged
drwxr-xr-x   4 root root       4096 feb 14 06:37 boot
dr-xr-xr-x   2 root root       4096 ago 27 15:39 cdrom
drwxr-xr-x  20 root root       4100 ene 20 21:56 dev
drwxr-xr-x 111 root root       4096 feb 20 16:09 etc
drwxr-xr-x   3 root root       4096 dic  9 14:42 home
lrwxrwxrwx   1 root root          7 abr 22  2024 lib -> usr/lib
lrwxrwxrwx   1 root root          9 abr 22  2024 lib64 -> usr/lib64
drwxr-xr-x   2 root root       4096 feb 26  2024 lib.usr-is-merged
drwx------   2 root root      16384 dic  9 14:23 lost+found
drwxr-xr-x   2 root root       4096 ago 27 14:18 media
drwxr-xr-x   2 root root       4096 ago 27 14:18 mnt
drwxr-xr-x   2 root root       4096 ago 27 14:18 opt
dr-xr-xr-x 290 root root          0 ene 20 21:56 proc
drwx------   7 root root       4096 feb 14 17:36 root
drwxr-xr-x  33 root root       1040 feb 24 15:19 run
lrwxrwxrwx   1 root root          8 abr 22  2024 sbin -> usr/sbin
drwxr-xr-x   2 root root       4096 abr  3  2024 sbin.usr-is-merged
drwxr-xr-x   2 root root       4096 dic  9 14:42 snap
drwxr-xr-x   2 root root       4096 ago 27 14:18 srv
-rw-------   1 root root 4294967296 dic  9 14:36 swap.img
dr-xr-xr-x  13 root root          0 ene 20 21:58 sys
drwxrwxrwt  11 root root       4096 feb 24 15:09 tmp
drwxr-xr-x  12 root root       4096 ago 27 14:18 usr
drwxr-xr-x  14 root root       4096 ene 20 15:14 var
root@svreditorial:/#
```

Figura 30. Lista de directorios del sistema de archivos en Linux Ubuntu Server versión 24.04

El sistema de archivos de Linux es jerárquico. Describe una arborescencia de directorios y subdirectorios, partiendo de un elemento base llamado raíz o root directory. La jerarquía se llama FHS (Filesystem Hierarchy Standard) e incluye los siguientes directorios" (BARDOT, 2022).

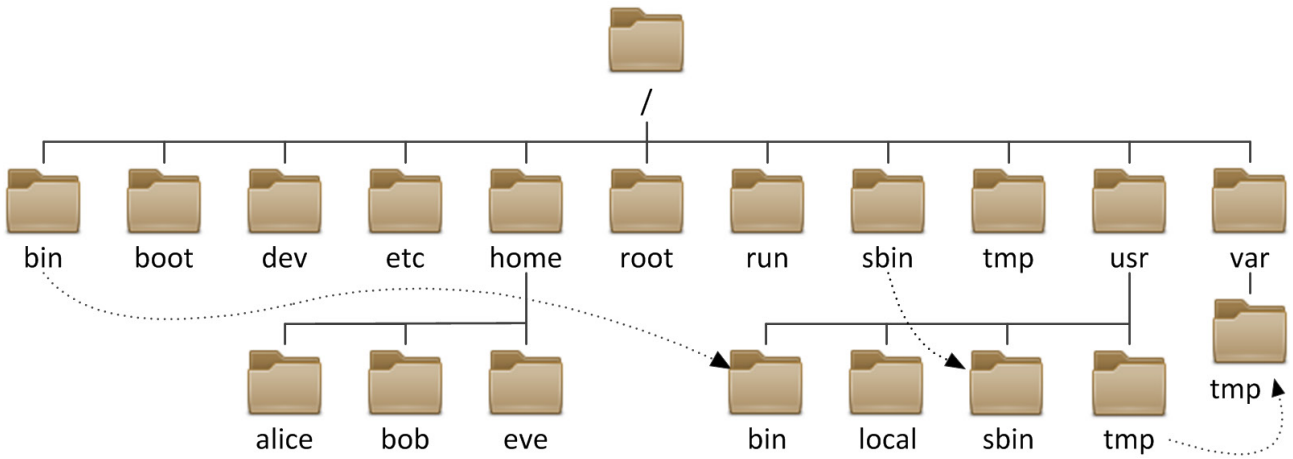



Figura 31. Estructura general de directorio de Linux (Linux Filesystem Hierarchy). Tomado de Linux Ubuntu Desktop

"Los sistemas de archivos son una estructura que utiliza el sistema operativo para gestionar la información de los diferentes dispositivos de almacenamiento. Cada sistema operativo suele tener una serie de sistemas de archivos propios" (Alegre Ramos, 2023).

Generalmente, estamos acostumbrados a la jerarquía del sistema de archivos NTFS (New Technology File System), propio de Microsoft Windows y Windows Server que nos permite organizar los datos en discos duros y otras unidades de almacenamiento. Como mencionaba anteriormente, lo común es tener la unidad

C:\ como partición principal y demás unidades D:\, E:\, etc. como secundarias, que se las suele definir al momento de la instalación de Windows. Cada una puede contener subcarpetas y archivos distintos, por ejemplo, dentro de la carpeta C:\Windows\System32, una de las carpetas más importantes del sistema puede contener lo siguiente:

- Archivos ejecutables: archivos con extensión .exe que ejecutan programas y servicios esenciales de Windows, como el administrador de tareas (taskmgr.exe) o el símbolo del sistema (cmd.exe).
- Controladores de *hardware*: archivos con extensión .sys que permiten que el sistema operativo se comunique con dispositivos de *hardware* como impresoras, tarjetas de red o gráficos.
- Bibliotecas DLL: archivos .dll (Dynamic Link Libraries) que contienen funciones y recursos compartidos que usan varios programas. Estas bibliotecas permiten que diferentes aplicaciones utilicen el mismo código sin duplicarlo, ahorrando espacio y recursos.
- Utilidades del sistema: incluye herramientas para gestionar el sistema operativo, como la configuración del sistema (msconfig.exe) o el editor de registro de Windows (regedit.exe).
- Archivos de configuración: archivos relacionados con la configuración del sistema y las políticas de seguridad

Dentro de la carpeta Archivos de programa(x86) y Archivos de programa generalmente se instalan programas de 32 y de 64 bits respectivamente. En la carpeta Windows tenemos todos los archivos necesarios para que el sistema operativo funcione. El sistema NTFS permite trabajar con archivos superiores a 4 GB o crear particiones de más de 2 TB. También existen mejoras respecto a la velocidad a la hora de escribir archivos pequeños y de resolver problemas de fragmentación del disco, muy evidentes en sistemas FAT32. En Windows, al ser un sistema propietario, no podemos gestionar o realizar alguna modificación de mejora en archivos del sistema, y esto limita la capacidad de personalizar o adaptar el sistema a necesidades específicas.

El acceso restringido a estos archivos implica que las mejoras o ajustes deben hacerse dentro de las limitaciones permitidas por Microsoft. Esto reduce el control que los usuarios avanzados o desarrolladores pueden tener sobre el rendimiento, seguridad y funcionamiento general del sistema, en comparación con sistemas de código abierto como Linux, donde los usuarios tienen acceso completo al código fuente y pueden modificarlo según sus necesidades.

El sistema de archivos NTFS es totalmente compatible con versiones recientes de Windows, por lo que no presenta ningún inconveniente en este entorno. Sin embargo, su compatibilidad se limita principalmente cuando necesitas compartir información con dispositivos que usan otros sistemas operativos. Por ejemplo, en macOS (Apple), los discos formateados en NTFS se pueden leer, pero no se permite escribir en ellos, a menos que se utilicen herramientas de terceros. Esto puede dificultar el intercambio de archivos entre dispositivos.

Con Linux y Android, la compatibilidad nativa con NTFS es aún más limitada. Para utilizar un disco o *pendrive* formateado en NTFS, es necesario instalar *software* adicional que permita leer y escribir en este sistema de archivos. Por estas razones, si planeas compartir información entre dispositivos con diferentes sistemas operativos, es recomendable utilizar un sistema de archivos más compatible, como exFAT o FAT32, que tiene una mayor interoperabilidad entre Windows, macOS, y Linux.

En Linux en cambio el sistema de archivos por defecto es el ext4 (Fourth Extended Filesystem) para instalaciones de Ubuntu y otras muchas distribuciones de Linux, ya que su *kernel* soporta numerosos formatos, pero los más utilizados son los de la familia Ext, XFS, Btrfs, F2FS, JFS y ReiserFS. Cada uno de estos tiene sus propias características para ser utilizadas en distintos escenarios. A continuación se describe brevemente cada uno de estos sistemas de archivos:

### Ext4 (Fourth Extended Filesystem)

"En ext4, se cambió el método de asignación de datos de bloques individuales a *extents*. La idea detrás de esto es escribir la mayor parte de los datos del archivo en un área continua y luego anotar sólo la dirección de su primer bloque y la cantidad de bloques en la secuencia. Hasta cuatro *extents* pueden almacenarse directamente en el inodo, mientras que el resto se organizan como un árbol B+" (Laboratories, s.f.)

**Características:**

- Es el sistema de archivos predeterminado en muchas distribuciones de Linux, basado en versiones anteriores de Ext.
- Soporta volúmenes de gran tamaño (hasta 1 exabyte) y archivos de hasta 16 terabytes.
- Incluye mejoras en la asignación de espacio, verificación de la integridad de archivos, y un mejor manejo de grandes volúmenes de datos.
- Compatibilidad con versiones anteriores como Ext3.

**Escenario:** Ideal para la mayoría de los usuarios de Linux que buscan un sistema de archivos robusto, eficiente y ampliamente soportado, como en servidores o sistemas de escritorio.

### XFS

"En general, este tipo de sistema de archivos es altamente escalable y funciona muy bien en el *hardware* de servidor. En 2001, llegó al núcleo de Linux y ahora se soporta en la mayoría de las distribuciones de Linux, algunas de las cuales, como, por ejemplo, Red Hat Enterprise Linux, incluso lo usan de forma predeterminada" (Laboratories, s.f.).

**Características:**

- Sistema de archivos de alto rendimiento diseñado para manejar grandes volúmenes de datos y archivos.
- Soporta tamaños de archivos de hasta 8 exabytes.
- Posee una asignación basada en *extents* (porciones grandes de disco) y un excelente manejo de múltiples procesos de E/S simultáneos.
- No permite redimensionar particiones fácilmente.

**Escenario:** Utilizado frecuentemente en servidores de alto rendimiento y entornos donde se requiere manejar grandes bases de datos o cargas de trabajo intensivas en almacenamiento.

### Btrfs (B-Tree File System)

"Btrfs (del inglés B-tree File System, Sistema de archivos del árbol B) es uno de los formatos de nueva generación más populares para Linux y se está haciendo un gran esfuerzo para hacerlo estable. Fue desarrollado por Oracle y se soporta por el núcleo principal de Linux desde 2009. Fedora y SUSE ya lo emplean de forma predeterminada" (Laboratories, s.f.).

**Características:**

- Se enfoca en ofrecer integridad de datos mediante sumas de verificación y *snapshots* (instantáneas).
- Permite la compresión de datos, la asignación dinámica de espacio y la fácil administración de volúmenes.
- Soporte para redimensionar volúmenes y agregar discos al sistema de archivos sin desmontar.

**Escenario:** Adecuado para sistemas que requieren características avanzadas como la creación de *snapshots* del sistema de archivos, es decir, copias en el tiempo sin duplicar toda la información y manteniendo una administración flexible de almacenamiento, como en servidores de virtualización o sistemas de respaldo. Compatible con sistemas como Ubuntu, es útil para *backups* o sistemas que necesiten restauraciones rápidas como sistemas virtualizados. Mantiene otras características ventajas y desventajas, que van más allá del contenido de este libro.

## F2FS (*Flash-Friendly File System*)

"F2FS (del inglés *Flash-Friendly File System*, Sistema de archivos compatible con memoria *flash*) es otro formato que fue introducido por Samsung Electronics en 2012. Ha sido diseñado específicamente para los dispositivos de almacenamiento basados en la memoria *flash* NAND y, por lo tanto, se utiliza más ampliamente en los teléfonos inteligentes modernos y medios de almacenamiento portátiles (extraíbles)" (Laboratories, s.f.)

**Características:**

- Diseñado específicamente para dispositivos de almacenamiento basados en memoria flash, como SSDs y tarjetas de memoria.
- Optimiza las operaciones de escritura para reducir el desgaste de la memoria flash y mejorar su vida útil.
- Tiene una estructura de asignación basada en segmentos para evitar la fragmentación.

**Escenario:** Se usa principalmente en dispositivos móviles, SSDs y sistemas embebidos que requieren un sistema de archivos optimizado para almacenamiento *flash*.

## JFS (Journaled File System)

"JFS (del inglés Journaled File System, Sistema de archivos de registro) fue creado por IBM en 1990. La versión original, a veces denominada JFS1, se implementó en el sistema operativo AIX de la empresa. Más tarde, se lanzó JFS2 que se llegó a Linux después de hacerse de código abierto" (Laboratories, s.f.).

**Características:**

- Desarrollado por IBM, es conocido por su baja utilización de CPU y su capacidad para manejar grandes volúmenes de datos con eficiencia.
- Es un sistema de archivos de *journaling* (registro), lo que lo hace adecuado para sistemas donde la integridad de los datos es importante.
- Tiene buen rendimiento en sistemas con uso intensivo de escritura.

**Escenario:** Se suele utilizar en servidores empresariales que requieren estabilidad y un bajo consumo de recursos. Si ocurre un fallo como un apagón, los datos pueden quedar corruptos porque las operaciones de escritura no se completaron correctamente. Los sistemas *journaling* como JFS mantienen un registro de transacciones, y solo cuando el *journal* confirma que la operación fue segura, los datos se guardan realmente en el sistema de archivos, garantiza una mayor seguridad e integridad de los datos en casos de fallos.

## ReiserFS

"ReiserFS es un formato alternativo de Linux que está optimizado para almacenar una gran cantidad de archivos pequeños. Fue diseñado por Namesys en 2001 y trajo una serie de características nuevas que se consideraron muy innovadoras en el momento de su presentación. No obstante, finalmente, su mantenimiento se entregó a voluntarios debido a ciertos problemas técnicos" (Laboratories, s.f.)

**Características:**

- Destacado por su manejo eficiente de pequeños archivos y su buena capacidad de búsqueda.
- Aunque ha sido reemplazado en popularidad por otros sistemas como Ext4 y Btrfs, sigue siendo notable por su rendimiento con ciertos tipos de cargas de trabajo.
- No está tan activamente mantenido como otros sistemas de archivos.

**Escenario:** Se usaba en servidores Linux antiguos y en sistemas donde se necesitaba un manejo eficiente de archivos pequeños.

*Swap*

A continuación, tenemos un recurso muy utilizado en Linux y es denominado *swap* yaunque no es un sistema de archivos en sí mismo, está relacionado con el almacenamiento y la gestión de memoria en Linux. *Swap* se refiere más a una característica del sistema operativo que utiliza una parte específica del disco, para extender la memoria virtual.

**Swap:** "Es un tipo especial que se utiliza para la memoria de intercambio, necesaria para no colapsar la memoria RAM con procesos que consuman mucha memoria. El sistema operativo usa esta partición como memoria virtual para descargar los datos de la RAM y así mejorar el rendimiento del sistema. En las últimas versiones de los sistemas operativos Linux se ha sustituido por un fichero llamado swapfile que cumple la misma función. Tiene la ventaja de no tener un espacio reservado previamente, pero como inconveniente tiene que, si hay más de un sistema operativo Linux instalado, cada uno de ellos tendrá su propio fichero swapfile, con lo cual no podrán compartir el espacio. De todas formas, es posible configurar el sistema para usar un sistema u otro" (Alegre Ramos, 2023).

Si el sistema tiene poca RAM disponible y está ejecutando muchas aplicaciones, puede usar *swap* para evitar que las aplicaciones se cierren o que el sistema se cuelgue. Sin embargo, es ideal que el sistema utilice *swap* lo menos posible, ya que la velocidad de acceso al disco es mucho menor que la de la RAM. El tamaño del *swap* depende de la cantidad de RAM del servidor y del uso planificado, como se recomienda en la siguiente tabla:

Tabla 3. Asignación de la *swap* recomendada según el tamaño en memoria RAM

| RAM instalada en el servidor | Tamaño de *swap* recomendado |
|---|---|
| Menos de 4GB | Igual o 2 x RAM |
| Entre 4-8 GB | Igual a la RAM |
| Más de 8 GB | 4GB o menos como precaución |

Para servidores con 16GB o más de RAM, el *swap* se convierte en un recurso secundario o de respaldo, y dependerá del uso que se le vaya a dar, si el uso de es intensivo, con 2-4 GB de swap será suficiente. En aplicaciones más exigentes, configurar con 4-8 GB y si se necesita hibernación, el tamaño del *swap* puede ser de 16 GB.

## Directorios y archivos importantes

Los sistemas operativos basados en Unix, como por ejemplo GNU Linux, FreeBSD inclusive macOS, que también se basó en Unix y al igual que Android derivado de Linux y iOS derivado de Mac OS X, y por supuesto Linux Ubuntu Server, son sistemas que organizan sus datos bajo un directorio de nivel superior llamado directorio raíz o *root*, denotado por el símbolo "/". Sin embargo, los subdirectorios pueden variar de nombre según la distribución.

Los sistemas operativos basados en Linux comparten una estructura de directorios ajustada al estándar Filesystem Hierarchy Standard FHS de Unix, por ejemplo: /bin, /etc, /usr, /var, /home. MacOS mantiene una estructura similar porque también está basado en Unix, pero introduce directorios personalizados como /Applications, /System, o /Library, que no están presentes en Linux. Por su lado, Android, aunque usa un núcleo Linux, difiere un poco más en cuanto a subdirectorios que utiliza, por ejemplo: /system, /data, /sdcard, diseñados para dispositivos móviles. A continuación, se presenta una tabla con una descripción general del contenido de cada uno los subdirectorios más comunes en Linux Ubuntu.

Esto es importante que usted lo conozca o memorice porque le facilitará la administración del sistema, podrá identificar rápidamente la fuente de un error, ayudando a solucionar problemas, puede mejorar la seguridad, detectando archivos anómalos; también le ayudará a identificar que datos respaldar e inclusive le ayudará al desarrollo de *software,* ubicando los directorios correctos.

Tabla 4. Descripción general del contenido que representa cada directorio de Linux.

Fuente (BARDOT, 2022)

| Rutas | Descripción |
|---|---|
| / | Directorio de nivel superior o directorio raíz, base del sistema de archivos |
| /bin | En este directorio se encuentran archivos binarios ejecutables. |
| /boot | Archivos que se requieren para iniciar |
| /cdrom | Directorio de montaje predefinido para los CD y DVD |
| /dev | Archivos especiales que representan los periféricos o componentes de hardware |
| /etc | Contiene archivos de configuración del sistema que rigen el comportamiento de varios programas y servicios. |
| /home | En este directorio se encuentran carpetas personales de los usuarios. |
| /lib | Bibliotecas del sistema que son críticas para la funcionalidad del sistema y la aplicación |
| /lib64 | Igual que el anterior, pero en 64 bits |
| /lost+found | Directorio que contiene los archivos huérfanos o dañados que se encuentran después de una verificación del sistema de archivos. Su presencia indica que el directorio actual es la raíz de un sistema de archivos. |
| /media | Directorio de montaje de periféricos temporales (llave USB, CD, disco externo, etc.) |
| /mnt | Directorio de montaje más bien usado para las pruebas del administrador o montajes manuales |
| /opt | Aplicaciones o bibliotecas adicionales que deberán colocarse en una estructura de árbol específica |
| /proc | Sistema de archivos virtual que proporciona una interfaz hacia las funciones y parámetros del núcleo |
| /root | Carpeta del administrador |
| /sbin | Archivos binarios ejecutables dedicados al administrador |
| /srv | Datos de los servicios ofrecidos por el sistema |
| /sys | Sistema de archivos virtual, similar a /proc, pero dedicado principalmente a los periféricos del tipo de bloques |
| /tmp | Archivos temporales |
| /usr | Archivos binarios, bibliotecas, fuentes, documentación, etc., que el usuario puede utilizar |
| /var | Archivos y carpetas de contenidos |

Una vez presentada esta descripción general de directorios y expuesto el propósito que cada una de ellas tiene, puedo decir que, desde el punto de vista de un administrador de sistemas, particularmente es muy habitual emplear los siguientes directorios /home, /etc, /tmp, /usr, /var, pero claro, todo depende del uso que se le dé al sistema operativo. Si voy a utilizar frecuentemente un dispositivo USB, es porque de seguro estaré accediendo con frencuencia al directorio /media. Como ya lo mencioné, lo mejor es que el lector memorice al menos la descripción general del contenido que representa cada directorio de Linux.

Algunos subdirectorios y archivos se crean automáticamente en el momento de realizar una instalación, por ejemplo, si instalo el servidor web de apache se creará el subdirectorio /var/www/html/ y más archivos de configuración dentro del directorio /etc como por ejemplo /etc/apache2/apache2.conf. Si nuevamente por ejemplo instalo una base de datos como Mysql, se crearán de similar manera archivos y subdirectorios dentro del directorio /etc como por ejemplo /etc/mysql/mysql.conf.d/mysqld.cnf. Si observamos en la descripción el directorio /etc, este contiene archivos de configuración del sistema: en este ejemplo del servidor web de Apache, de la base de datos MySQL y entre otras diversas aplicaciones que pretenda instalar, es la manera que el sistema operativo organiza las aplicaciones. Recuerde que en Linux casi todo se representa por un archivo que está dentro de un directorio o un subdirectorio.

## Comandos esenciales

Muchos técnicos profesionales prefieren administrar los sistemas operativos desde una terminal Linux llamado también consola o *shell*, una interfaz de línea de comandos que interpreta y ejecuta las órdenes introducidas por el usuario. La razón es porque desde una terminal Linux se ofrecen varias ventajas en cuanto a seguridad, acceso remoto, gestión de recursos, mantenimiento y actualización, instalación y configuración, supervisión y monitoreo, etc. Linux Ubuntu Server utiliza por defecto el Bash (*Bourne Again Shell*) como su *shell* principal muy utilizado en muchas distribuciones Linux debido a su versatilidad, características avanzadas y compatibilidad con *scripts* de *shell* escritos para el antiguo Bourne Shell (**sh**).

Muchos de los comandos de sistemas basados en Unix, en cuanto al nombre del comando y al resultado que estos presentan cuando se ejecutan, son particularmente similares o iguales. Si por ejemplo tiene experiencia utilizando otras distribuciones Linux, podrá comprobar que, desde una terminal, si ejecutamos el comando "ls", este presentará un listado de archivos y directorios; sin embargo, otros comandos pueden resultar totalmente diferentes. Basado en mi experiencia particular, trataré de utilizar los comandos mayormente empleados en Linux Ubuntu Server en el ámbito de la Administración de Sistemas, desde lo más básico a lo más cotidiano, pero si desea profundizar más, le sugiero conseguir un manual de comandos Linux.

### Comando ls

Para listar archivos y directorios del sistema operativo ejecutamos ls

root@svrlinux:/home/cesarg# ls ↵ Enter

El comando ls tiene algunos parámetros como, por ejemplo -l, -la, -a, -R, -lh, y dependiendo de la necesidad, utilizaremos el parámetro que más nos convenga. La salida se puede apreciar en la siguiente figura.

-l: Lista archivos y directorios con un detalle por ejemplo el total 1461932, usuario root, mes, hora, etc.

-la: Lista archivos y directorios incluidos los ocultos, pero si un detalle amplio.

-a: Muestra todos los archivos, incluidos los ocultos, ejemplo. bashrc

-R: Lista todos los archivos de los subdirectorios, en el ejemplo de folder1, folder2 y folder3.

-lh: convierte los tamaños a formatos legibles, como MB, GB y TB.

Si usted quisiera almacenar la salida que le da el comando ls, simplemente agregue el parámetro > u operador de redirección de salida, como se indica a continuación:

root@svrlinux:/home/cesarg# ls -l > archivo.txt ↵ Enter

Redirige la salida detallada del comando ls -l al archivo archivo.txt. Si el archivo no existía, se crea; si ya existía, se sobrescribe con la nueva información, como resultado, el listado de archivos y directorios no se mostrará en la terminal, sino que se almacenará en archivo.txt.

Existe otro operador llamado '*pipeline*' (tubería). Su propósito es pasar la salida de un comando, por ejemplo, "ls -l" directamente como entrada para otro comando como por ejemplo "*more*", por eso el siguiente comando mostrará una salida paginada, esto se utiliza cuando el listado no alcanza en la pantalla.

```
root@svrlinux:/home/cesarg# ls -l | more ↵ Enter
```

El siguiente comando lista directorios y organiza la salida, mostrando solamente directorios con extensión ".txt" en el directorio actual, la salida de ls -l se pasa al comando "grep", que filtra líneas que contienen ".txt".

```
root@svrlinux:/home/cesarg# ls -l | grep ".txt" ↵ Enter
```

Para contar cuántos archivos y carpetas hay en el directorio actual, ingresamos el siguiente comando:

```
root@svrlinux:/home/cesarg# ls -l | wc -l ↵ Enter
```

En la siguiente figura se muestra el comando ls con diferentes parámetros y sus respectivas salidas:

```
root@svreditorial:/home/soporte# ls
certificados.tar.gz  doi.zip  omp-3.3.0-20  omp-3.3.0-20.tar.gz  ssl  wordpress-6.7.1.zip  wp-config.php
root@svreditorial:/home/soporte# ls -l
total 80080
-rw-rw-r--  1 soporte soporte   134511 ene 20 22:02 certificados.tar.gz
-rw-rw-r--  1 soporte soporte    45331 feb 10 20:11 doi.zip
drwxrwxr-x 20 soporte soporte     4096 nov 29 20:08 omp-3.3.0-20
-rw-rw-r--  1 soporte soporte 53216184 feb 13 15:01 omp-3.3.0-20.tar.gz
drwxr-xr-x  4 root    root        4096 may  8  2022 ssl
-rw-rw-r--  1 soporte soporte 28585184 ene 20 16:42 wordpress-6.7.1.zip
-rwxrwxrwx  1 root    root        3234 ene 20 16:46 wp-config.php
root@svreditorial:/home/soporte# ls -la
total 80112
drwxr-x---  6 soporte soporte     4096 feb 24 19:25 .
drwxr-xr-x  3 root    root        4096 dic  9 14:42 ..
-rw-------  1 soporte soporte       95 dic 17 20:46 .bash_history
-rw-r--r--  1 soporte soporte      220 mar 31  2024 .bash_logout
-rw-r--r--  1 soporte soporte     3771 mar 31  2024 .bashrc
drwx------  2 soporte soporte     4096 dic  9 14:56 .cache
-rw-rw-r--  1 soporte soporte   134511 ene 20 22:02 certificados.tar.gz
-rw-rw-r--  1 soporte soporte    45331 feb 10 20:11 doi.zip
drwxrwxr-x 20 soporte soporte     4096 nov 29 20:08 omp-3.3.0-20
-rw-rw-r--  1 soporte soporte 53216184 feb 13 15:01 omp-3.3.0-20.tar.gz
-rw-r--r--  1 soporte soporte      807 mar 31  2024 .profile
drwx------  2 soporte soporte     4096 dic  9 14:42 .ssh
drwxr-xr-x  4 root    root        4096 may  8  2022 ssl
-rw-r--r--  1 soporte soporte        0 dic  9 14:58 .sudo_as_admin_successful
-rw-rw-r--  1 soporte soporte 28585184 ene 20 16:42 wordpress-6.7.1.zip
-rwxrwxrwx  1 root    root        3234 ene 20 16:46 wp-config.php
root@svreditorial:/home/soporte# ls -a
.   .bash_history  .bashrc  certificados.tar.gz  omp-3.3.0-20         .profile  ssl                       wordpress-6.7.1.zip
..  .bash_logout   .cache   doi.zip              omp-3.3.0-20.tar.gz  .ssh      .sudo_as_admin_successful  wp-config.php
root@svreditorial:/home/soporte# ls -R
```

Figura 32. Salida del comando ls con sus diferentes parámetros

Lista de archivos y directorios con más detalle, pero el total lo expresa en K, M, G o T, como en el siguiente ejemplo 1,4G

```
root@svrlinux:/home/cesarg# ls -lh ┘ Enter
total 1,4G
-rw-------  1 root   root    390  jul 10  15:20  50-cloud-init.yaml
drwxr-xr-x  2 root   root   4,0K  oct 5   16:05  folder1
drwxr-xr-x  2 root   root   4,0K  oct 5   16:06  folder2
drwxr-xr-x  2 root   root   4,0K  oct 5   16:07  folder3
-rw-rw-r--  1 cesarg cesarg  1,4G  sep 5   16:07  linux-6.10.10.8.tar
```

Cuando utilice el comando ls para listar archivos y directorios, los diferentes colores indican atributos específicos de los archivos y directorios. Esto es parte de la configuración predeterminada del sistema para ayudar a identificar rápidamente el tipo y estado de cada archivo. También se puede observar el usuario al que pertenece dicho archivo o directorio, fecha y hora de creación.

**Rojo:** Los archivos que aparecen en rojo suelen indicar archivos comprimidos o archivos que tienen permisos especiales, como los archivos ejecutables. También puede significar que los archivos tienen ciertos permisos restringidos o problemas de acceso.

**Azul:** Los directorios se muestran en azul para diferenciarlos de los archivos normales, facilitando su identificación dentro de una lista.

Los colores son manejados por la variable de entorno LS_COLORS, que define cómo el sistema pinta los diferentes tipos de archivos. Podemos personalizar estos colores modificando esta variable o los archivos de configuración del sistema.

En los detalles o salida del comando **ls** podemos observar algunos campos, el primer carácter indica el tipo de archivo: **d**: especifica que es un directorio, **-**:el guion al inicio especifica que se trata de un archivo regular, en el ejemplo 50-cloud-init.yaml es un archivo, mientras que folder1 y folde2 son directorios-, **l**: representa un enlace simbólico; **b**: dispositivo de bloque (como un disco duro) y **c**: dispositivo de caracteres (como un puerto USB serial).

El **2** representa el número de enlaces duros que apuntan al archivo o directorio, **root** es el propietario o usuario que creó el archivo o directorio, **group** es el grupo propietario del archivo o directorio, y también podría pertenecer al grupo **cesarg**.

```
-rw-------  1 root   root    390  jul 10  15:20  50-cloud-init.yaml
drwxr-xr-x  2 root   group  4096  oct 5   16:05  folder1
drwxr-xr-x  2 root   cesarg  4096  oct 5   16:05  folder2
```

En este caso, solo *root* (y los usuarios con permisos de *root*) puede modificar este archivo. Otros usuarios solo pueden leerlo si tienen permisos de lectura, pero no pueden modificarlo ni ejecutarlo.

4096: representa el tamaño del archivo en bytes. Para directorios, indica el espacio usado para almacenar los nombres de los archivos dentro del directorio. Oct 5 16:05: Es la fecha y hora de la última modificación del archivo o directorio.

Analicemos el siguiente ejemplo. Si tenemos un archivo llamado miarchivo.txt, el único que tiene permisos para modificar o eliminar el archivo son el usuario *root* y el usuario cesarg. En el segundo caso solo el usuario *root* tiene permiso para eliminar o modificar el archivo llamado files.docx y, en el tercer caso, tanto el usuario *root* como el usuario cesarg tienen permisos sobre el directorio midirectory.

```
-rwxr-xr-x  2  cesarg  cesarg  4096  oct  5  16:05  miarchivo.txt
-rwxr-xr-x  2  root    root    4096  oct  5  18:30  files.docx
drwxr-xr-x  2  root    cesarg  4096  oct  5  12:00  midirectory
```

A continuación, se tiene un enlace simbólico "**l**" que puede apuntar a un archivo o directorio ubicado en una ruta específica dentro del sistema de archivos. Este enlace actúa como un atajo o referencia, redirigiendo al usuario a las aplicaciones o al destino especificado, algo así como un acceso directo en Windows.

Si observamos la letra "**c**", esta se refiere a que ttyUSB0 representa un puerto serie virtual asignado a un dispositivo USB que se encuentra conectado al sistema, como un adaptador USB a serial. Finalmente, "**b**" hace referencia a un dispositivo de bloque como un disco duro.

```
lrwxrwxrwx  1  user  group  12 Nov 28 12:02  enlace -> /ruta/gateway
crw-rw----  1  root  group  10, 232 Nov 28 12:03 ttyUSB0
brw-rw----  1  root  group  8, 0 Nov 28 12:04  sda
```

Además, tenemos 9 caracteres rwx rwx rwx, que son los permisos de archivo en tres bloques:

- El primer bloque rwx son los permisos del propietario del archivo:
  - r (read) – Permiso de lectura. Permite leer archivos o listar directorios.
  - w (write) – Permiso de escritura. Permite modificar o borrar un archivo. En el caso de un directorio, podríamos alterar su contenido borrando o modificando cualquier archivo que contenga, también creando nuevos archivos o subdirectorios.
  - x (execute) – Permiso de ejecución. Permite ejecutar archivos o entrar en directorios. Por ejemplo, ejecutar el archivo como un programa o script.
  - s – setuid, setgid o sticky, puede ser minúscula (ejecutable) o mayúscula (no ejecutable).
- El segundo bloque rwx es para el grupo asociado al archivo.
- El tercer bloque rwx son los permisos para otros usuarios.

Recuerda que los permisos son tanto para archivos como para directorios. Para leer el contenido de un archivo, si tiene la opción r activada, podríamos utilizar el comando *cat* o *less*. Para crear un archivo, podemos utilizar *touch,* y para crear un directorio, el comando mkdir.

Como usuario *root,* a estos archivos o directorios podemos aplicar rwx, al usuario propietario u:user, a los usuarios que pertenecen al grupo asignado al archivo o directorio g:group, y a todos los demás usuarios que no son propietario ni están en el grupo, o:others. El usuario *root* es el usuario con máximos privilegios en Linux, es quien tiene el control de todo el sistema, pudiendo modificar configuraciones, acceder a cualquier archivo, instalar o desinstalar *software.*

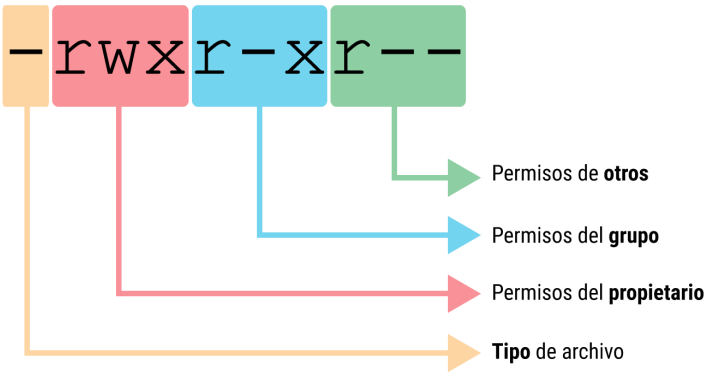



Figura 33. Los bits de permisos.

Fuente: (Prored., s.f.)

### Ejemplos de permisos

Estos son ejemplos de esquema de permisos en ambas notaciones:

Tabla 3. (Prored., s.f.)

| Notación Simbólica | Notación octal | Permiso | Notación decimal |
|---|---|---|---|
| ---------- | 000 | ninguno | 0 |
| ---x-x-x | 111 | solo x | 1 |
| --w-w-w- | 222 | solo w | 2 |
| --wx-wx-wx | 333 | w +x | 3 |
| -r-r-r-- | 444 | solo r | 4 |
| -r-xr-xr-x | 555 | r + x | 5 |
| -rw-rw-rw- | 666 | r + w | 6 |
| -rwxrwxrwx | 777 | rwx | 7 |
| -rwx------ | 700 | | |
| -rwxrwx--- | 770 | | |

Si quisiera otorgar rwx al usuario, r-x al grupo y r-- a otros, el equivalente numérico sería el 750, como se indica a continuación.

```
root@svrlinux:/#chmod 750 script.sh ┘Enter
```

Por ejemplo, si tenemos un *script* llamado script.sh y queremos que otros puedan ejecutarlo, pero no editarlo, podemos ejecutar.

```
root@svrlinux:/#chmod 755 script.sh ┘Enter
```

Esto significa:

Usuario: rwx (leer, escribir, ejecutar).

Grupo: r-x (leer, ejecutar) escribir no puede

Otros: r-x (leer, ejecutar) escribir no puede

Lo mismo aplicaría para un directorio en el cual sea accesible para todos, pero solo el dueño pueda modificar el contenido.

```
root@svrlinux:/#chmod 755 midirectorio ┘ Enter
```

Esto significa:

Usuario: rwx (leer, escribir, ejecutar).

Grupo: r-x (leer, ejecutar) escribir no puede

Otros: r-x (leer, ejecutar) escribir no puede

Otorgar permisos y quitar permisos con + y −, respectivamente

```
chmod u+x archivo  # Añadir permiso de ejecución al usuario
chmod g+w archivo  # Añadir permiso de escritura al grupo
chmod o+r archivo  # Hacer que otros puedan leer el archivo

chmod g-w archivo  # Quitar permiso de escritura al grupo
chmod o-x archivo  # Quitar permiso de ejecución a otros
```

Como hemos visto, tenemos tres comandos principales. Para cambiar permisos (leer, escribir y ejecutar) utilizamos el comando chmod**,** para cambiar el propietario y opcionalmente el grupo el comando chown**,** y para únicamente cambiar el grupo propietario del archivo el comando chgrp. Los permisos pueden cambiarse por algunas razones, garantizar la seguridad, privacidad y colaboración eficiente con otros usuarios.

Esto permite proteger datos sensibles contra accesos no autorizados, habilitas o restringes la ejecución de *scripts* o programas, evitas ediciones accidentales en archivos críticos, cumples con políticas de seguridad y buenas prácticas. A continuación, vamos a ver más ejemplos con estos comandos.

## Comando chmod

Es un comando muy utilizado que permite cambiar los permisos de lectura, escritura y ejecución de un archivo o de un directorio. Por ejemplo:

```
root@svrlinux:/#chmod 777 folder ┘ Enter
```

Este número define los permisos que se asignarán al archivo o directorio, en este caso, al directorio folder. Los permisos en Linux se dividen en tres categorías:

- **Usuario (Owner):** el propietario del archivo.
- **Grupo (Group):** los usuarios que pertenecen al mismo grupo que el archivo.
- **Otros (Others):** todos los demás usuarios del sistema.

Cada número en 777 corresponde a una combinación de permisos para cada categoría: 7 equivale a “lectura (r)”, “escritura (w)”, y “ejecución (x)” habilitadas. Se calcula como la suma de:

Lectura: 4

Escritura: 2

Ejecución: 1

**Por lo tanto:**

root@svrlinux:/#chmod 777 folder ↵ Enter

Primer 7: Da permisos completos (lectura, escritura, y ejecución) al usuario propietario, directorio folder.

Segundo 7: Da permisos completos al grupo.

Tercer 7: Da permisos completos a otros usuarios.

El comando chmod es especialmente útil cuando por ejemplo se publican sitios web en un servidor Apache, ya que permite gestionar los permisos de lectura, escritura y/o ejecución tanto de archivos como directorios del sitio web. Sobre todo, en escenarios donde la seguridad y la accesibilidad son muy importantes, avalando un adecuado control de acceso al publicar sitios web en nuestros servidores web.

Los archivos publicados en un servidor web Apache (HTML, CSS, JavaScript, etc.) necesitan permisos de lectura para que el servidor pueda servirlos correctamente a los usuarios. Por ejemplo, los archivos deben tener permisos como 644 (lectura para todos, escritura solo para el propietario).

Además, los directorios (como public_html) deben tener permisos como 755 para permitir que Apache pueda entrar en ellos y acceder a los archivos. Archivos de configuración sensibles como .env o config.php no deben tener permisos que permitan el acceso público como 777. Esto prevendría fugas de información sensible.

Si trabajamos con scripts (como PHP o Python) que necesitan ejecutarse en el servidor, estos deben tener permisos de ejecución (chmod + x).

En proyectos con múltiples archivos y directorios, se puede utilizar el modificador recursivo (**-R**) para aplicar permisos a todo el contenido de una carpeta y no tener que estar de archivo en archivo cambiando permisos. Esto se lo puede realizar de la siguiente manera:

root@svrlinux:/# chmod -**R** 755 /var/www/misitio ↵ Enter

root@svrlinux:/# chown -**R** www-data:www-data /var/www/misitio ↵ Enter

**Consideraciones de seguridad:** Evitar 777 siempre que sea posible. El permiso 777 otorga lectura, escritura y ejecución a todos los usuarios, lo cual puede abrir puertas a ataques, como modificaciones maliciosas o eliminación de datos.

Es preferible configurar los permisos mínimos necesarios, generalmente 644 para archivos y 755 para directorios.

**Propietario y grupo:** Es recomendable que los archivos web pertenezcan al usuario y grupo que utiliza Apache, típicamente **www-data** en Ubuntu de la siguiente manera:

root@svrlinux:/# chown -R www-data:www-data /var/www/misitio ↵ Enter

### Comando *chown*

Se utiliza para cambiar el propietario y/o el grupo de un archivo, directorio o enlace simbólico al usuario o grupo especificado. Esto resulta útil para administrar permisos de acceso en el sistema. La estructura básica de la sintaxis es la siguiente:

**chown [opciones] usuario[:grupo] archivo**

- usuario: El nombre del usuario que se convertirá en el propietario del archivo o directorio.
- grupo (opcional): El grupo al cual se asignará el archivo (se especifica después de un : o .).
- archivo: Especifica el archivo o directorio que será modificado.

Cambia el propietario de un archivo.

root@svrlinux:/#chown usuario archivo.txt

Cambiar propietario y grupo:

root@svrlinux:/#chown usuario:grupo archivo.txt

## Comando chgrp

Este comando se utiliza para cambiar el grupo propietario de un archivo o directorio. Es útil en sistemas multiusuario para gestionar permisos y accesos mediante la asignación de archivos a diferentes grupos. La estructura del comando es:

**chgrp [opciones] grupo archivo**

- **grupo:** Es el nombre o el identificador (GID) del grupo al que se asignará el archivo o directorio.
- **archivo:** Es el archivo o directorio cuyo grupo propietario será modificado.

**Opciones comunes:**

- **-R (Recursivo):** Cambia el grupo de todos los archivos y subdirectorios dentro de un directorio.
- **-v (Verbose):** Muestra los archivos y directorios cuyo grupo se ha cambiado.

root@svrlinux:/#chgrp -R grupo /ruta/del/directorio

root@svrlinux:/#chgrp -v grupo archivo

## Comando cd

Es un comando que permite recorrer directorios y archivos de Linux. Desde luego, para desplazarse al directorio raíz o tener un control completo primero debe autenticarse como usuario *root* con el comando *sudo su*.

Se puede utilizar el comando **cd** seguido de un espacio y de .. para regresar, como se muestra a continuación:

root@svrlinux:/home/cesarg#cd .. ↵ Enter

root@svrlinux:/home#cd .. ↵ Enter

root@svrlinux:/

También se puede digitar directamente el directorio al que se requiere acceder.

root@svrlinux:/#cd /home/cesarg ↵ Enter

root@svrlinux:/home/cesarg#

Puede utilizar el comando pwd para conocer la ubicación actual, por ejemplo:

root@svrlinux:/home/cesarg#pwd ↵ Enter

/home/cesarg/

## Comando mkdir y rmdir

El comando **mkdir** permite crear uno o varios directorios, por ejemplo, el siguiente comando crea simultáneamente dos directorios: carpeta y folder.

root@svrlinux:/#mkdir carpeta folder ┘Enter

El comando **rmdir** se utiliza principalmente para borrar directorios, se puede utilizar únicamente para borrar directorios vacíos.

root@svrlinux:/#rmdir directorio ┘Enter

El comando **rm** se utiliza para remover archivos o directorios con todo su contenido dentro, siempre y cuando se utilice el parámetro (-r). Nunca ejecutar el comando **rm -r** con rutas absolutas.

root@svrlinux:/#rm -r documentos ┘Enter

elimina el directorio documentos y todo su contenido.

root@svrlinux:/#rm -rf /var/www/html/editorial/cache/*

eliminar de forma recursiva y forzada todos los archivos dentro del directorio cache.

## Comando touch

Este comando permite crear uno o varios archivos vacíos o puede actualizar la fecha de modificación de un archivo existente, por ejemplo, el siguiente comando crea un archivo llamado texto.

root@svrlinux:/#touch texto ┘Enter

Se puede verificar que el archivo se ha creado listando con el comando ls.

root@svrlinux:/#ls ┘Enter

-rw-r--r-- 1 root root 0 feb 25 15:36 texto

Pero también se puede utilizar un echo para crear un archivo y escribir contenido en él, como se indica a continuación: root@svrlinux:/#echo "Hola mundo" > archivo.txt

Esto crea un archivo.txt con el texto "Hola mundo", pero si el archivo ya existe, se sobrescribirá. Para añadir contenido al final del archivo, se utiliza doble mayor que >> como se indica a continuación: root@svrlinux:/#echo "Bienvenidos" >> archivo.txt

## Comando cat

Otro comando que puede crear archivos es cat, aunque se utiliza principalmente para concatenar y mostrar archivos. Por ejemplo, para crear un archivo llamado mi_archivo.txt ejecutamos:

root@svrlinux:/#cat > mi_archivo.txt ┘Enter

Luego se puede seguir escribiendo texto "Esto es una prueba" que se añadirá al archivo mi_archivo.txt, pero para salir del *prompt* presionamos las teclas CTRL+D. Si queremos añadir más texto usamos el mismo comando, pero con doble signo >>. Para ver o mostrar el contenido del archivo en la terminal, ejecutamos

simplemente cat y el nombre del archivo.

```
root@svrlinux:/#cat mi_archivo.txt ┘ Enter
Esto es una prueba
```

Tenga cuidado de utilizar el comando cat con ">" en archivos de configuración de Linux porque podría alterar los mismos y causar un error de funcionamiento del sistema, sobre todo si trabaja como *root*, entonces, mejor utilizar un editor de texto como vi, mc o nano.

Para aprender estos comandos y estar lo más familiarizado posible con el sistema, el lector debe practicarlos constantemente en una terminal y servidor de pruebas. Linux Ubuntu maneja una gran cantidad de comandos y empezar por los más utilizados es un paso importante para la gestión de este sistema, inclusive de otras distribuciones.

### Comando *less*

Es una herramienta de visualización de archivos en Linux, se utiliza principalmente para leer archivos grandes, ya que permite desplazarse hacia adelante y hacia atrás por el contenido de un archivo, lo cual es una ventaja sobre el comando *cat*, que muestra el contenido desde el principio hasta el final sin permitir una navegación.

```
root@svrlinux:/#less nombre_archivo.txt ┘ Enter
```

## Desplazarse:

- Flecha arriba/Abajo o *Page Up*/*Page Down*: navegar por el contenido.
- Space: avanzar una página.
- b: retroceder una página.

## Buscar texto:

- / seguido del término: para buscar hacia adelante.
- ? seguido del término: para buscar hacia atrás.
- n: para ir a la siguiente ocurrencia de la búsqueda.

## Salir de *less*:

- Presionar **q** para salir y volver a la línea de comandos.

### Comando *tail*

Este comando se utiliza para mostrar las últimas líneas de un archivo. Por defecto, *tail* muestra las diez últimas líneas de un archivo, pero podemos personalizarlo. Maneja el parámetro -f, el cual quiere decir "proseguir", es decir, no solo muestra las diez últimas líneas, sino que continúa mostrando nuevas líneas del archivo en tiempo real. Por ejemplo, en Linux, un archivo .log es un archivo de registro que se utiliza para almacenar información de eventos que ocurren dentro del sistema operativo, aplicación o servicio y estos archivos generalmente van creciendo en texto de información. Ejemplos de estos archivos serían error.log, acess.log, etc.

root@svrlinux:/# tail -f /var/log/apache2/error.log

tail: Muestra las últimas líneas de un archivo.

-f: Permite que tail siga el archivo en tiempo real, mostrando nuevas líneas a medida que se añaden al final del archivo.

/var/log/apache2/error.log: Especifica el archivo de registro de errores de Apache que se está monitoreando.

error.log: registra errores de una aplicación, como los que pueden ocurrir en un servidor web como Apache.

access.log: documenta quién accedió a un recurso en un servidor web.

**Comando tar**

Es una herramienta muy importante en Linux utilizada para empaquetar archivos y, en combinación con otras herramientas, como gzip para comprimir o descomprimir archivos y tener archivos .tar.gz

root@svrlinux:/# tar -cvf paquete.tar dir/ ↵ Enter

Este comando empaqueta en un archivo con extensión .tar, en este caso se crea un archivo **"paquete.tar"** con todo el contenido del directorio **dir/**

-c: Esta opción indica que queremos crear un nuevo archivo tar.

-v: Significa verbose. Con esta opción, tar mostrará una lista de los archivos que se están añadiendo al archivo tar a medida que se procesan.

-f: Esta opción indica que el siguiente argumento es el nombre del archivo de salida, en este caso, paquete.tar.

-x: Indica el modo de extracción, es decir para extraer el contenido.

root@svrlinux:/# tar -xvf paquete.tar ↵ Enter

Este comando indica que vamos a extraer el contenido del archivo paquete.tar

El siguiente comando en una sola línea descomprime y extrae el contenido del archivo omp-3.4.0-8.tar.gz

root@svrlinux:/# tar -xf omp-3.4.0-8.tar.gz ↵ Enter

Uso de gzip, primero empaquetamos el archivo y luego comprimimos.

root@svrlinux:/# gzip -9 paquete.tar ↵ Enter

La salida de este comando será un archivo con extensión .tar.gz es decir **paquete.tar.gz** el número -9 especifica el nivel de compresión que se desea aplicar, los niveles de compresión pueden variar de -1 (compresión más rápida, menor tasa de reducción) a -9 (compresión más lenta, máxima reducción de tamaño). Al utilizar -9, se le está diciendo a gzip que utilice el máximo esfuerzo para comprimir el archivo, lo que dará como resultado un archivo comprimido más pequeño, aunque el proceso de compresión tardará más tiempo.

root@svrlinux:/# gunzip paquete.tar.gz ↵ Enter

Este comando descomprime el archivo paquete.tar.gz y quedaría como paquete.tar

## Comando df

Es una herramienta muy útil para mostrar información sobre el uso de espacio en disco de los sistemas de archivos montados. Podemos tener algunos parámetros como -T, que indica el tipo de sistema de archivos que podría ser ext4, xfs, ntfs, etc. Y el parámetro -h, que significa legible por humanos (*human-readable*) y hace que se muestren en un formato más fácil de entender como K (kilobytes), M (megabytes), G (gigabytes), etc. A continuación, observemos una salida típica de este comando.

```
root@svrlinux:/# df -Th ┘ Enter
Filesystem                    Type   Size  Used Avail Use% Mounted on
tmpfs                         tmpfs  1,6G  1,3M  1,6G   1% /run
/dev/mapper/ubuntu--vg-ubuntu--lv ext4    98G   12G   82G  13% /
/dev/sda2                      ext4   2,0G  182M  1,7G  10% /boot
tmpfs                         tmpfs  1,6G   12K  1,6G   1% /run/user/1000
root@svrlinux:/#
```

## El comando cp y mv

Se utiliza para copiar el contenido de un directorio o archivos a una ubicación especifica. Por ejemplo: cp archivo-origen ruta-destino. Al copiar con el comando **cp,** el archivo copiado no se modifica y si en el destino existe un archivo con el mismo nombre, se sobrescribe, a no ser que le digamos lo contrario; también maneja algunos parámetros que se indican a continuación:

-r: Copia directorios de forma recursiva. Esta opción es necesaria si está copiando un directorio y todo su contenido.

-i: Pregunta antes de sobrescribir archivos ya existentes en el destino.

-v: Muestra detalles de lo que está haciendo el comando, impresión en modo "verbose".

-u: Solo copia si el archivo de origen es más reciente que el archivo de destino o si este último no existe.

```
root@svrlinux:/#cp [parámetros] origen destino
```

Supongamos que tenemos un archivo.txt y queremos hacer una copia de él en el mismo directorio, así, el comando sería:

```
root@svrlinux:/#cp archivo.txt archivo-copia.txt ┘ Enter
```

Si deseamos copiar un directorio completo llamado directorio/ a otro directorio llamado nuevo-directorio/, debe utilizar el parámetro -r para que la copia sea recursiva, así, el comando sería:

```
root@svrlinux:/#cp -r directorio nuevo-directorio ┘ Enter
```

Si desea que el comando pregunte antes de sobrescribir un archivo, puede usar el parámetro -i y para ver lo que el comando está haciendo el parámetro -v.

```
root@svrlinux:/#cp -i archivo.txt archivo-copia.txt ┘ Enter
root@svrlinux:/#cp -v archivo.txt archivo-copia.txt ┘ Enter
```

El comando mv se utiliza para mover o renombrar archivos y directorios. Ejemplo: para mover un archivo a un directorio llamado documentos.

root@svrlinux:/#mv -i archivo.txt documentos/ ↵ Enter

Si queremos renombrar un archivo.txt como un nuevo archivo informe.txt, ingresaríamos el comando de la siguiente manera:

root@svrlinux:/#mv archivo.txt informe.txt ↵ Enter

### Instalar paquetes apt *(advanced package tool)*

El comando típico para instalar un paquete sería sudo apt install, nombre_del_paquete. Es necesario el prefijo sudo, ya que los comandos de instalación necesitan permisos de administrador.

Instalando el programa mc, que es un editor de archivos y directorios.

root@svrlinux:/$ sudo **apt** install mc ↵ Enter

So cambiamos el prompt a #, no necesitaríamos anteponer sudo, por ejemplo:

root@svrlinux:/$ sudo su ↵ Enter

nos pide la clave de administrador y si es correcta se cambia el prompt a #, que indica que estamos trabajando con privilegios de administrador.

Ahora, para instalar un paquete ya no es necesario utilizar el prefijo sudo, por ejemplo, para instalar curl, que es una herramienta de línea de comandos que permite realizar solicitudes y transferir datos a través de internet utilizando una variedad de protocolos como HTTP, HTTPS, FTP, etc.

root@svrlinux:/#**apt** install curl ↵ Enter

Una vez instalado, para ver la versión de curl, podemos ejecutar:

root@svrlinux:/#curl ––version ↵ Enter

Ahora, para ver el contenido de una URL de algún sitio, podemos ejecutar:

root@svrlinux:/#curl https://google.com ↵ Enter

esto desplegará el contenido del sitio de Google, como se puede observar:

<HTML><HEAD><meta http-equiv=”content-type” content=”text/html;charset=utf-8”>

<TITLE>301 Moved</TITLE></HEAD><BODY>

<H1>301 Moved</H1>

The document has moved

<A HREF=”https://www.google.com/”>here</A>.

</BODY></HTML>

Este comando guarda el contenido de la URL en un archivo llamado salida.html

root@svrlinux:/#curl https://google.com -o salida.html ↵ Enter

Veamos otro ejemplo. Para instalar un paquete llamado nmap, ejecutamos el siguiente comando:

```
root@svrlinux:/#apt install nmap ┘Enter
```

Nmap es una herramienta muy poderosa utilizada principalmente para el escaneo de puertos de servicios de red y seguridad informática. Cuando ya esté instalado el paquete, puede utilizar el comando

```
root@svrlinux:/#nmap localhost ┘Enter
La salida de este comando es la siguiente, e indica los números de puerto abiertos.
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-02-25 15:25 UTC
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000040s latency).
Not shown: 997 closed tcp ports (reset)
PORT    STATE SERVICE
22/tcp  open  ssh
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 0.08 seconds
```

### El comando pwd

El comando pwd (*print working directory*) en Linux y otros sistemas Unix se usa para mostrar la ruta absoluta del directorio en el que se encuentre actualmente.

Por ejemplo, si está dentro de una carpeta llamada copia, que a su vez está dentro de ejercicio1, al ejecutar pwd, el sistema mostrará algo como:

```
root@svrlinux:/#pwd ┘Enter
/home/usuario/ejercicio1/copia
```

Es útil para orientarse dentro de la estructura de directorios y confirmar en qué carpeta se está trabajando antes de ejecutar comandos como ls, cd, o mv.

## Ejercicios propuestos

Esta sección tiene como objetivo consolidar el dominio de la terminal y la comprensión de la jerarquía de directorios en Ubuntu Server. Se recomienda realizarlo dentro de un ambiente controlado, es decir, desde una instalación nueva, desde un sistema que no esté en producción para que pueda ir verificando con confianza los comandos ejecutados.

Nivel 1: Navegación y estructura de archivos

1. **Exploración del sistema:** Debe situarse en su directorio personal (~). Desde allí, debe desplazarse al directorio donde se guardan los archivos de configuración del sistema (/etc) y listar su contenido. ¿Qué comando se utilizó para ver los archivos sin salir de su carpeta personal?

2. **Identificación de rutas:** Debescribir la **ruta absoluta** del archivo de configuración de red de Ubuntu

(ubicado en /etc/netplan). Luego, intente llegar a ese mismo directorio usando una **ruta relativa** desde la raíz (/).

3. **El árbol de archivos:** Debe crear la siguiente estructura de directorios en su carpeta personal con un solo comando: Laboratorio/Proyecto_Docker/config Laboratorio/Proyecto_Docker/data Laboratorio/Proyecto_Docker/logs

Nivel 2: Manipulación de archivos y directorios

4. **Creación y edición:** Dentro de la carpeta config creada anteriormente, debe crear un archivo vacío llamado servidor.conf. Agregar la frase *"Configuración inicial del servidor",* usando el comando echo y redireccionar la salida al archivo.

5. **Copia y movimiento:** Copiar el archivo servidor.conf a la carpeta logs y cambiar el nombre a error. log. Luego, mover el archivo original servidor.conf a la carpeta data.

6. **Visualización de contenido:** Mostrar las primeras cinco líneas del archivo /etc/passwd. Consulta: ¿para qué sirve este archivo ubicado dentro de uno de los directorios más importantes del sistema?

Nivel 3: Administración y búsqueda (avanzado)

7. **Búsqueda de archivos:** Utilice el comando find para localizar todos los archivos que terminen en .conf dentro del directorio /etc.

8. **Filtrado de datos (*Pipes*):** Ejecutar un comando que liste todos los archivos del directorio /bin y que, mediante el uso de un *pipe* (|), filtre solo aquellos que contengan la palabra "net".

9. **Limpieza del entorno:** Eliminar todo el directorio "Laboratorio" y su contenido de forma recursiva con un solo comando. **Precaución:** Asegúrese de estar en la ruta correcta antes de ejecutarlo.

**Reto final: el diagnóstico del administrador**

Imagine que un servicio está fallando. Se pide investigar los últimos eventos del sistema:

- Use el comando *tail* con el parámetro -f para monitorizar en tiempo real el archivo /var/log/syslog.
- ¿Qué diferencia se observa entre usar *cat*, *more* y *less* para leer un archivo de log muy extenso?

# CAPITULO IV PAQUETES ESENCIALES EN LINUX

## Contenido del capítulo IV

- Introducción a los paquetes
- Herramientas de red net-tools
- Herramientas de red iproute2
- Editor Midnight commander
- Editor Nano
- Herramienta de monitoreo y análisis de red nmap
- Paquete dnsutils y sus herramientas.

## Introducción a los paquetes

En Linux Ubuntu Server (https://ubuntu.com/download/server), los paquetes son archivos que contienen todos los elementos necesarios para instalar un programa. Estos elementos incluyen el *software*, así como archivos de configuración, dependencias y *scripts* para la instalación y desinstalación. Trataremos de clasificar los mismos para ir instalándolos de manera secuencial y explicando brevemente cada uno de ellos. En el siguiente capítulo, se instalarán también algunas aplicaciones de *software* gratuitas para Windows, cuya utilidad se explicará más adelante y den función de la interacción entre Windows y Linux Ubuntu Server.

## Herramientas de red net-tools

Net-tools es un paquete clásico que proporciona una colección de herramientas de red que son esenciales para la configuración y administración de redes.

En muchas distribuciones modernas de Linux, **net-tools** no viene preinstalado debido a la preferencia por **iproute2**, sin embargo, a continuación, procedemos con la instalación de net-tools y explicaremos algunas de sus herramientas.

```
root@svrlinux:~# apt install net-tools ┘ Enter
```

**ifconfig:** Configura y muestra interfaces de red, es muy utilizado para asignar direcciones IP, habilitar o deshabilitar interfaces y más funcionalidades. Ejemplo

```
root@svrlinux:~# ifconfig ┘ Enter
root@svrlinux:~# ifconfig eth0 192.168.1.100 ┘ Enter
```

**netstat:** Este comando presenta conexiones de red, tablas de enrutamiento, estadísticas de interfaz entre otra información. Ejemplo:

```
root@svrlinux:~# netstat -an ┘ Enter
root@svrlinux:~# netstat -r ┘ Enter
root@svrlinux:~# netstat -tuln ┘ Enter
```

**arp:** Este comando muestra y manipula la caché ARP (Address Resolution Protocol), que mapea direcciones IP a direcciones MAC. Ejemplo

root@svrlinux:~# arp -a ┘Enter

root@svrlinux:~# arp -d 192.168.1.1 ┘Enter

**hostname:** Presenta o establece el nombre de *host* del sistema. Este nombre puede cambiarse editando el archivo *hostname* que se encuentra en la ruta **/etc/hostname** y cambiando al nuevo nombre, luego reiniciamos el sistema con el comando "init 6" para que los cambios surtan efecto.

root@svrlinux:~# hostname ┘Enter

root@svrlinux:~# hostname newhostname ┘Enter

**mii-tool:** Diagnostica problemas en la interfaz de red, especialmente relacionado con el estado del enlace. Ejemplo

root@svrlinux:~# mii-tool ┘Enter

**route:** Manipula la tabla de enrutamiento IP, es utilizado para agregar, eliminar o mostrar rutas.

root@svrlinux:~# route add default gw 192.168.1.1 ┘Enter

root@svrlinux:~# route add -net 192.168.2.0/24 gw 192.168.1.1 ┘Enter

Es recomendable practicar estos comandos en la terminal para ir familiarizándose con ellos, ya que son muy empleados en la administración de sistemas Linux. A continuación, un ejemplo de la salida típica del comando route -n:

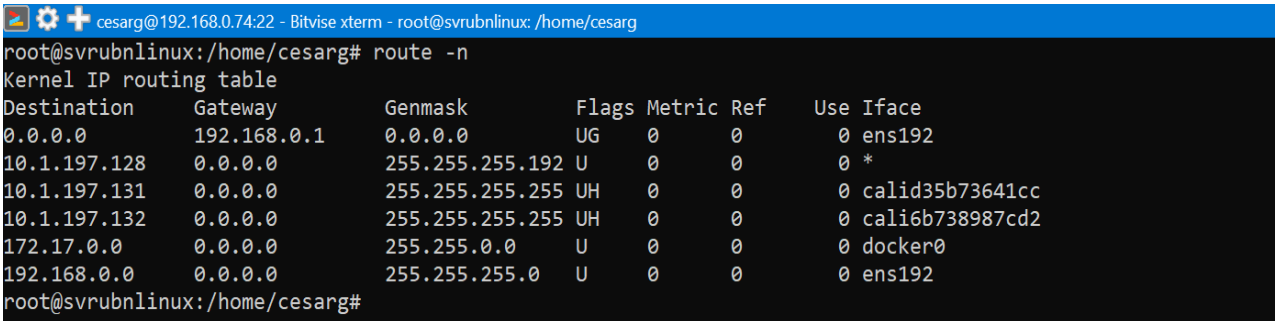



Figura 34. Vista de la tabla de rutas, conexión desde Bitvise xterm

Ahora explicaremos el significado de cada columna en la salida:

Destination: La dirección de destino para el enrutamiento.

Gateway: La dirección IP de la puerta de enlace o *router* a través del cual se enruta el tráfico.

Genmask: La máscara de red correspondiente a la ruta.

Flags: Indicadores que describen la ruta, como:

U: La ruta está activa (Up).

G: La ruta es a una puerta de enlace (Gateway).

H: La ruta es a un host.

Metric: El coste de la ruta. Un valor menor indica una ruta preferida.

Ref: El número de referencias a la ruta.

Use: El recuento de usos de la ruta.

Iface: La interfaz de red utilizada para la ruta.

## Herramienta de red iproute2

Iproute2 en sistemas operativos basados en Linux es una colección de utilidades de *software* de red para administrar y supervisar configuraciones de red y ofrece funcionalidades avanzadas para configurar interfaces de red, rutas, túneles, políticas de enrutamiento y más características. Algunas de las herramientas más comunes incluidas en iproute2 son ip, tc (*Traffic Control*), ss (*Socket Statistics*), arp (*Address Resolution Protocol*), entre otras. Estas herramientas son esenciales para poder administrar y diagnosticar redes en sistemas Linux modernos y que, de alguna manera, relevan a herramientas más tradicionales como ifconfig, route y netstat.

A continuación, se presenta algunos ejemplos de estos comandos:

Este comando muestra las interfaces de red:

root@svrlinux:~# ip addr ↵ Enter

Solo muestra interfaces en ejecución

root@svrlinux:~# ip link show up ↵ Enter

Permite agregar una dirección IP a una interfaz

root@svrlinux:~# ip addr add 192.168.1.100/24 dev eth0 ↵ Enter

Este comando presenta la tabla de enrutamiento

root@svrlinux:~# ip route ↵ Enter

Solo muestra conexiones TCP/IP Ipv6

root@svrlinux:~# ip -6 a ↵ Enter

Permite mostrar los puertos de escucha:

root@svrlinux:~# ss -l ↵ Enter

## Editor Midnight Commander

A continuación, instalaremos una herramienta imprescindible para la gestión de archivos y directorios que permite borrar, editar, copiar archivos, etc., y puede ser ejecutada desde una terminal Linux. Existen otros programas similares como el editor "nano" y el editor "vi", pero se ha preferido instalar Midnight Commander debido a su facilidad de manejo y otras funcionalidades que posee frente a otros paquetes similares. Midnight Commander o simplemente "mc" puede instalarse en casi todas las distribuciones Linux incluyendo Linux Ubuntu Server.

Es un administrador de archivos visual, con licencia pública general GNU y, por lo tanto, califica como *software* libre. La aplicación se ejecuta en modo texto de pantalla completa con muchas funciones que le permite copiar, mover y eliminar archivos y árboles de directorios completos, buscar archivos y ejecutar comandos en el *subshell*. Se incluyen visor y editor internos. Se basa en interfaces de texto versátiles, como Ncurses o S-Lang, lo que le permite funcionar en una consola normal, dentro de una terminal X Window, a través de conexiones SSH y todo tipo de *shells* remotos (https://midnight-commander.org). Para instalar, ingresamos en la terminal la siguiente instrucción:

root@svrlinux:~# apt install mc ↵ Enter

Para lazar la aplicación desde la terminal ingresamos simplemente *"mc"*.

root@svrlinux:~# mc ↵ Enter

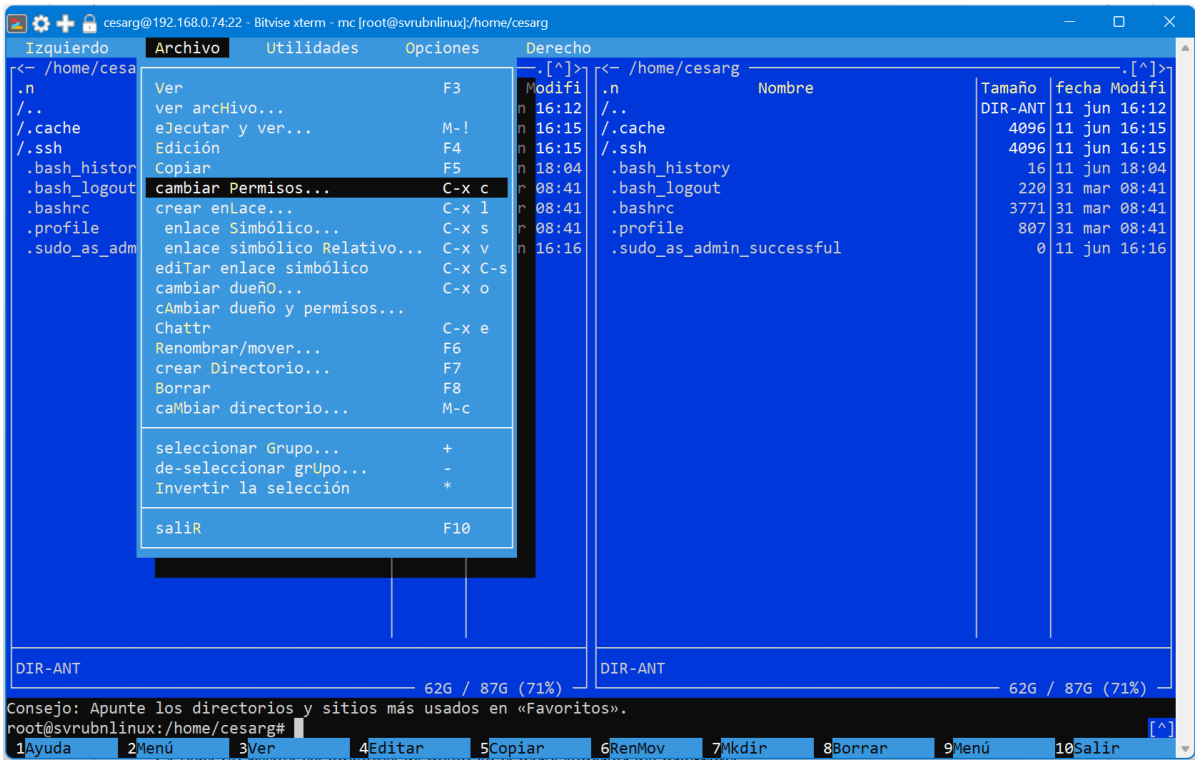



Figura 35. Software Midnight Commander, permite la gestión de archivos y directorios.

En la parte inferior tenemos opciones que se ejecutan al presionar la tecla de función con el número correspondiente. Por ejemplo, la tecla F1 indica la ayuda; F3, modo de solo lectura de un archivo seleccionado; F4, modo edición de un archivo previamente seleccionado; la tecla F7, para crear un directorio; F8, eliminar el archivo seleccionado; F10, salir del programa y volver a la terminal. Si nos fijamos en la parte superior y usando las teclas de dirección, se puede seleccionar el símbolo de *slash* seguido de dos puntos **/..**

esto es para regresar al directorio anterior. Por otro lado, recordemos que para movilizarnos a diferentes directorios desde la terminal usamos el comando cd ..

La tecla F9 resalta las opciones de menú en la parte superior del programa; en esta parte se puede presionar la tecla ESC para salir del menú de opciones. Se puede apreciar que el programa maneja dos paneles, izquierdo y derecho, por cuanto se puede seleccionar el panel presionando la tecla TAB.

Cuando se presente un mensaje de selección de editores, escoja la opción 3 o mcedit, que corresponde al editor "mc".

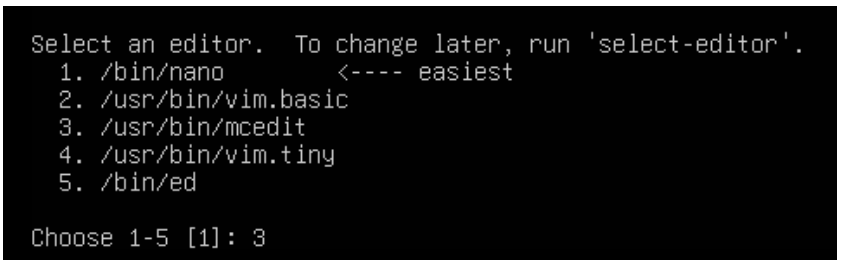



Figura 36. Ventana de mensaje para seleccionar un editor por default

## Editor Nano

Uno de los editores más utilizados en Linux es Nano, un editor de texto ligero y fácil de usar que se ejecuta en la línea de comandos, es decir, desde una terminal de sistemas Unix y/o Linux. Está diseñado para ser intuitivo y accesible, especialmente para usuarios que no están familiarizados con otros editores de texto más complejos como Vim o Emacs. Además, viene instalado por defecto en la mayoría de las distribuciones.

Los comandos básicos en nano generalmente están precedidos por la tecla Ctrl, por ejemplo: Ctrl + X para salir del editor, Ctrl + O para escribir el archivo, Ctrl + W para buscar texto, etc. La desventaja es que comparado con editores más avanzados como vim, una mejora del clásico editor de Unix vi o el editor de mc, Nano tiene menos funcionalidades y características. Pero si no quieres complicaciones en cuanto a tareas de edición de texto y rápidas, Nano es una muy buena opción.

Lo podemos ejecutar directamente desde una terminal ingresando el comando nano, seguido del nombre de archivo que queremos editar.

root@svrlinux:~# nano nombre_archivo ↵ Enter




Figura 37. Pantalla principal del editor nano 7.2 en Linux Ubuntu

## Herramienta de monitoreo y análisis de red nmap

Uno de los paquetes indispensables para la administración de servicios es nmap, una de las mejores herramientas de escaneo de puertos y descubrimiento de *hosts* que existe actualmente. Nmap (*Network Mapper*) nos permitirá obtener información sobre nuestro propio servidor o equipos de nuestra red, y es capaz de verificar qué puertos están abiertos y también qué *hosts* están levantados. La forma simple para escanear los puertos de nuestro propio servidor es ejecutar el comando nmap localhost. Previamente, procedemos con la instalación de nmap.

root@svrlinux:~# apt install nmap ↵ Enter

root@svrlinux:~# nmap localhost ↵ Enter

Como se indica en la siguiente captura de pantalla, se puede visualizar el estado de los puertos si se encuentran abiertos y a qué servicio pertenecen. Localhost (127.0.0.1) hace referencia al mismo ordenador, pero en lugar de localhost también puede utilizarse una dirección IP o nombre de dominio que pertenezca a un ordenador dentro de la red LAN o de Internet, por ejemplo:

root@svrlinux:~# nmap 192.168.0.10 ↵ Enter

root@svrlinux:~# nmap google.com ↵ Enter

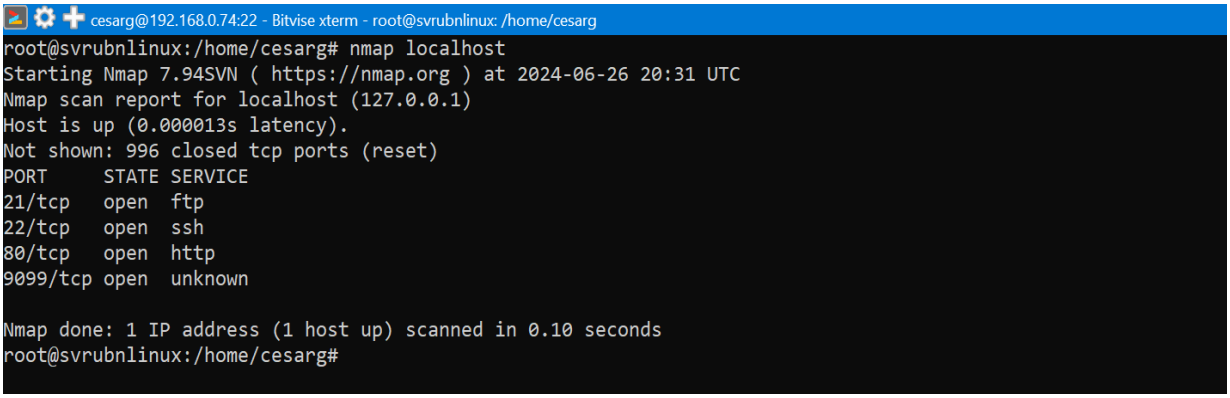



Figura 38. Salida del comando nmap localhost para indicar el estado abierto o cerrado de los servicios

Nmap es muy popular entre administradores de redes y profesionales de la seguridad informática debido a su versatilidad y potencia, es una herramienta de código abierto y gratuita para la exploración de redes y auditoría de seguridad. A continuación, se destacan algunos ejemplos:

Permite escanear una red completa:

root@svrlinux:~# nmap 192.168.1.0/24 ↵ Enter

Permite escanear puertos específicos por ejemplo escanear el puerto 80 en la IP 192.168.1.1:

root@svrlinux:~# nmap -p 80 192.168.1.1 ↵ Enter

Permite utilizar scripts NSE para detección de vulnerabilidades:

root@svrlinux:~# nmap ––script=<nombre_del_script> <gateway_ip> ↵ Enter

Por ejemplo, para ejecutar el script de detección de vulnerabilidades HTTP

root@svrlinux:~# nmap ––script=http-vuln* <gateway_ip> ┘ Enter

## Paquete dnsutils y sus herramientas

El paquete dnsutils incluye varias herramientas útiles para la gestión y consulta de servicios DNS. Estas herramientas son esenciales para la administración de sistemas y redes, especialmente en contextos donde la resolución de nombres es necesaria. A continuación, se presentan las herramientas incluidas en dnsutils:

**nslookup:** Herramienta de línea de comandos utilizada para consultar servidores DNS y obtener información sobre registros DNS.

**dig (***Domain Information Groper***):** Proporciona consultas de DNS más avanzadas. Es preferido por muchos administradores de sistemas debido a su capacidad para mostrar información detallada sobre los registros DNS.

**host:** Es una herramienta simple para realizar consultas DNS, pero ideal para consultas rápidas y básicas. Se utiliza para convertir nombres de dominio a direcciones IP y viceversa.

Para instalar el paquete, ingresamos los siguientes comandos:

root@svrlinux:~#apt update ┘ Enter

root@svrlinux:~#apt install dsnutils ┘ Enter

Por ejemplo, si quiere conocer la dirección IP de "google.com" o cualquier otro servidor, podemos utilizar nslookup google.com

root@svrlinux:~#nslookup google.com ┘ Enter

Server: 127.0.0.53

Address: 127.0.0.53#53

Non-authoritative answer:

Name: google.com

Address: 192.178.50.78

Name: google.com

Address: 2607:f8b0:4008:80a::200e

O podemos realizar una consulta inversa, es decir, de IP a dominio y obtener el nombre de dominio asociado a dicha dirección IP, y la ejecutaríamos de la siguiente manera:

root@svrlinux:~#nslookup 8.8.8.8 ┘ Enter

8.8.8.8.in-addr.arpa name = dns.google.

Authoritative answers can be found from:

Para realizar consultas DNS más avanzadas y obtener información detallada sobre los registros DNS podemos utilizar el comando dig, por ejemplo:

```
root@svrlinux:~#dig google.com ┘ Enter


; <<>> DiG 9.18.28-0ubuntu0.24.04.1-Ubuntu <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 8511
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             280     IN      A       142.250.217.174
;; Query time: 2 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Wed Feb 26 15:53:00 UTC 2025
;; MSG SIZE  rcvd: 55
```

La salida de dig incluye varias secciones, como la sección de pregunta (*QUESTION*), que muestra la consulta realizada en este caso de google; la sección de respuesta (*ANSWER*), que contiene los registros DNS encontrados como las direcciones IP asociadas a google.com; y la sección de autoridad (*AUTHORITY*), que muestra los servidores DNS autoritativos para el dominio.

También es interesante cómo dig puede ser utilizado para consultar otros tipos de registros DNS, como registros MX (para servidores de correo), registros TXT (para información de texto), o registros CNAME (para alias de dominio). Esto se hace añadiendo el tipo de registro al comando, por ejemplo, dig google.com MX.

Además, dig permite especificar el servidor DNS que se utilizará para la consulta, lo cual es útil para probar diferentes configuraciones o resolver problemas de DNS. Por ejemplo, puede usar dig @8.8.8.8 google.com para realizar la consulta usando el servidor DNS de Google.

Ahora, pruebe a ejecutar el siguiente comando, revisa e interpreta la salida.

```
root@svrlinux:~#dig gmail.com MX ┘ Enter
```

Cuando ejecutamos el comando *host* google.com se realiza una consulta DNS y obtener las direcciones IP asociadas con el dominio google.com. Por defecto, *host* consulta el registro A, que contiene las direcciones IPv4. También puede mostrar otros tipos de registros DNS, como registros MX (para servidores de correo) o registros TXT (para información de texto).

```
root@svrlinux:/home/cesarg# host google.com
google.com has address 192.178.50.78
google.com has IPv6 address 2607:f8b0:4008:80a::200e
```

```
google.com mail is handled by 10 smtp.google.com.
root@svrlinux:/home/cesarg#

root@svrlinux:/home/cesarg# host gmail.com
gmail.com has address 142.250.217.229
gmail.com has IPv6 address 2607:f8b0:4008:814::2005
gmail.com mail is handled by 20 alt2.gmail-smtp-in.l.google.com.
gmail.com mail is handled by 5 gmail-smtp-in.l.google.com.
gmail.com mail is handled by 40 alt4.gmail-smtp-in.l.google.com.
gmail.com mail is handled by 10 alt1.gmail-smtp-in.l.google.com.
gmail.com mail is handled by 30 alt3.gmail-smtp-in.l.google.com.
root@svrlinux:/home/cesarg#
```

# CAPÍTULO V *SOFTWARE* CLIENTE EN WINDOWS

## Contenido del capítulo V

- Introducción al *software* cliente en Windows
- *Filezilla Client*
- *Free Port Scanner*
- *Bitvise SSH Client*
- MySQL Workbench 8.0 CE
- Arquitectura cliente servidor

## Introducción al *software* cliente en Windows

El propósito de este capítulo básicamente es explicar la instalación y una breve descripción del *software* cliente en Windows. En los siguientes capítulos se expondrá la interacción que se puede tener entre este *software* cliente contra algunos de los servicios en Linux Ubuntu Server.

El *software* que se utiliza en Windows tiene un tipo de licencia *freeware,* es decir, se puede descargar de su sitio web oficial e instalarlo sin el inconveniente de la licencia y las limitaciones que otras licencias tienen respecto al *software* como por ejemplo la duración, la funcionalidad, etc. pero también utilizaremos *software* para Windows con licencia GNU General Public License (GPL) como es el caso de *Filezilla Client.*

Con base en la experiencia, puedo decir que las aplicaciones cliente en Windows tienen una mejor estilización y prestación frente a aplicaciones similares cliente en Linux en entornos gráficos como GNU o KDE, pero, curiosamente, cuando hablamos de servicios en Linux, si se los instala, configura y maneja correctamente, estos superan con creces a los mismos servicios que operan en sistemas Windows en cuanto a seguridad y funcionalidad. Por ejemplo, precisamente por este motivo hemos titulado a este libro "Servicios en Linux Ubuntu Server con clientes en Windows", para aprovechar estas mejores características.

También debe saber que, en general, las aplicaciones de pago o con licencia en Windows suelen ser mejores frente a aplicaciones cliente Windows con licencias gratuitas o *freeware*, aunque esto puede parecer subjetivo pues depende también de la utilidad que se le quiera dar al programa.

En Windows, existen varios tipos de *software* cliente que se utilizan para interactuar con servidores, gestionar bases de datos, desarrollar aplicaciones y más. Por ejemplo, existen clientes de correo electrónico como Mozilla Thuderbird o Microsoft Outlook; clientes FTP (*File Transfer Protocol*) como FileZilla y WinSCP; clientes SSH (*Secure Shell*) como por ejemplo PuTTY o Bitvise SSH Client; clientes de bases de datos como MySQL Workbench y SQL Server Management Studio (SSMS), clientes de navegación web como por ejemplo Firefox Developer Edition o Microsoft Edge; clientes de almacenamiento en la nube como Google Drive y Dropbox; clientes VPN (*Virtual Private Network*) como por ejemplo OpenVPN; clientes de control remoto como AnyDesk y TeamViewer; y clientes para administración de servidores web como, por ejemplo, phpMyAdmin o HeidiSQL. En este capítulo se instalará *software* cliente en Windows que, en la práctica, sobre todo en administración de sistemas, puede resultar de mucha utilidad.

## Filezilla Client

Filezilla es una aplicación para conexiones FTP (*File Transfer Protocol*) *opensource* que dispone tanto de un cliente como de un servidor; existen versiones para Linux y para Windows. Además, soporta los protocolos FTP, SFTP y FTP sobre SSL/TLS. Para más información, usted puede consultar el tutorial en español en el sitio web oficial ( https://wiki.filezilla-project.org/FileZilla_Client_Tutorial_(es) ). Es decir, FTP es un protocolo estándar utilizado para transferir archivos entre un cliente y un servidor en una red, como Internet. Existen una gran variedad de clientes FTP, desde aplicaciones gráficas con interfaces amigables, hasta herramientas de línea de comandos que son más adecuadas para usuarios avanzados. Es muy común su utilización para administrar o gestionar sitios web o también en entornos empresariales donde se necesita compartir grandes volúmenes de datos desde diferentes sitios.

La última versión estable de Filezilla Client es la 3.67.0, que puede obtenerse también del sitio web oficial de manera gratuita para Windows. Usted puede descargar el programa directamente desde el siguiente enlace:

https://download.filezilla-project.org/client/FileZilla_3.67.0_win64_sponsored2-setup.exe

El proceso de instalación simplemente consiste en ejecutar el archivo .exe descargado previamente, y seguir las instrucciones, presionamos el botón "*I Agree*", seleccionamos la opción que dice: "*Anyone who uses this computer (all users)*" es decir cualquier persona que utilice este equipo (todos los usuarios). Dejamos seleccionados los componentes que vienen por defecto, seleccionamos un folder de destino, por ejemplo: C:\Program Files\ FileZilla FTP Client\ y damos clic en el botón de Instalar. Una vez instalado y ejecutado el programa, se obtendrá la siguiente ventana. Debe saber que también existen otros clientes como gFTP, WinSCP, Cyberduck, pero hemos seleccionado Filezilla por ser uno de los más utilizados.

La idea más adelante es instalar y configurar el servidor FTP en Linux, particularmente vsftpd y a tráves del cliente Filezilla en Windows realizar la conexión al servidor, esto nos permitirá realizar transferencias de archivos y directorios en Linux hacia Windows o viceversa.

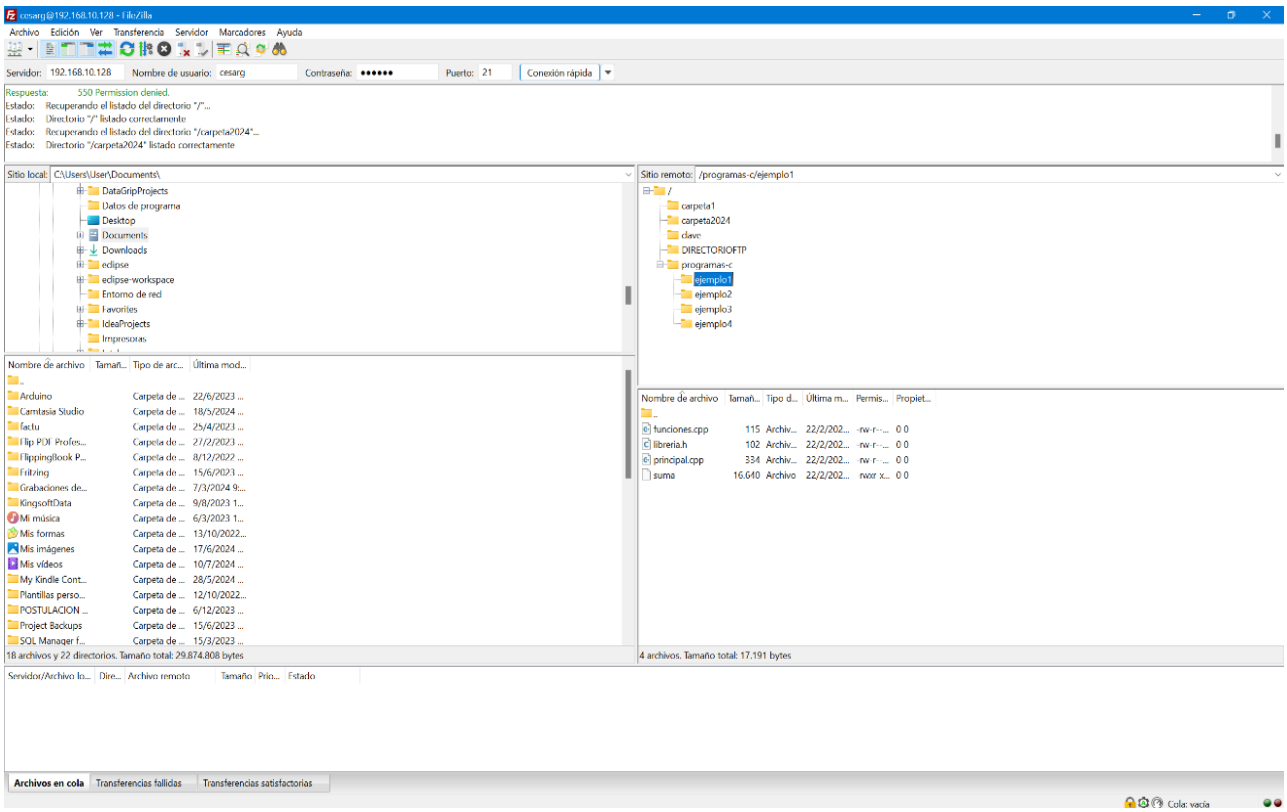



Figura 39. Ventana principal del Cliente de Filezilla, un *software* para conectarse a un servidor FTP

## Free Port Scanner

Es un programa gratuito para Windows que puede escanear puertos en máquinas locales o remotas de manera eficiente y realizar escaneos en rangos de puertos predefinidos; de esta manera, se puede verificar la disponibilidad de los mismos, el sitio web oficial es http://www.nsauditor.com . Cabe mencionar que existen disponibles en Internet otras herramientas que cumplen el mismo o similar propósito, así por ejemplo **nmap** para Linux, Windows y Mac OS ( https://nmap.org ), y **Angry IP Scanner** para Linux, Windows y Mac OS (https://angryip.org).

Free Port Scanner utiliza paquetes TCP para determinar: *hosts* disponibles y puertos abiertos, servicios asociados con el puerto y otras características importantes. La herramienta está diseñada con una interfaz fácil de utilizar y su licencia es *freeware*, lo que quiere decir que es un *software* que se ofrece sin costo, pero que mantiene ciertas restricciones impuestas por el autor, diferenciándose así del *software* libre que promueve la libre modificación y distribución del código fuente.

Free Port Scanner puede descargarse directamente de su sitio web oficial desde el siguiente enlace:

http://www.nsauditor.com/freeware/downloads/FreePortScanner.exe

El proceso de instalación en Windows es relativamente fácil: solo ejecutamos el archivo descargado y al dar clic en el mismo se presentará la ventana que se indica en la figura 7. Como puede apreciarse, tiene de forma básica dos campos, uno para ingresar la dirección IP del ordenador que quiero escanear, independientemente del sistema operativo, y, en el otro campo, los números de puertos TCP que quiero verificar separados por una coma. Para ejecutar el proceso de escaneo presionamos el botón de Scan, en la misma ventana más abajo puede observarse el estado del puerto *Open (abierto)* o *Close (cerrado)* con un nombre y una descripción más detallada.

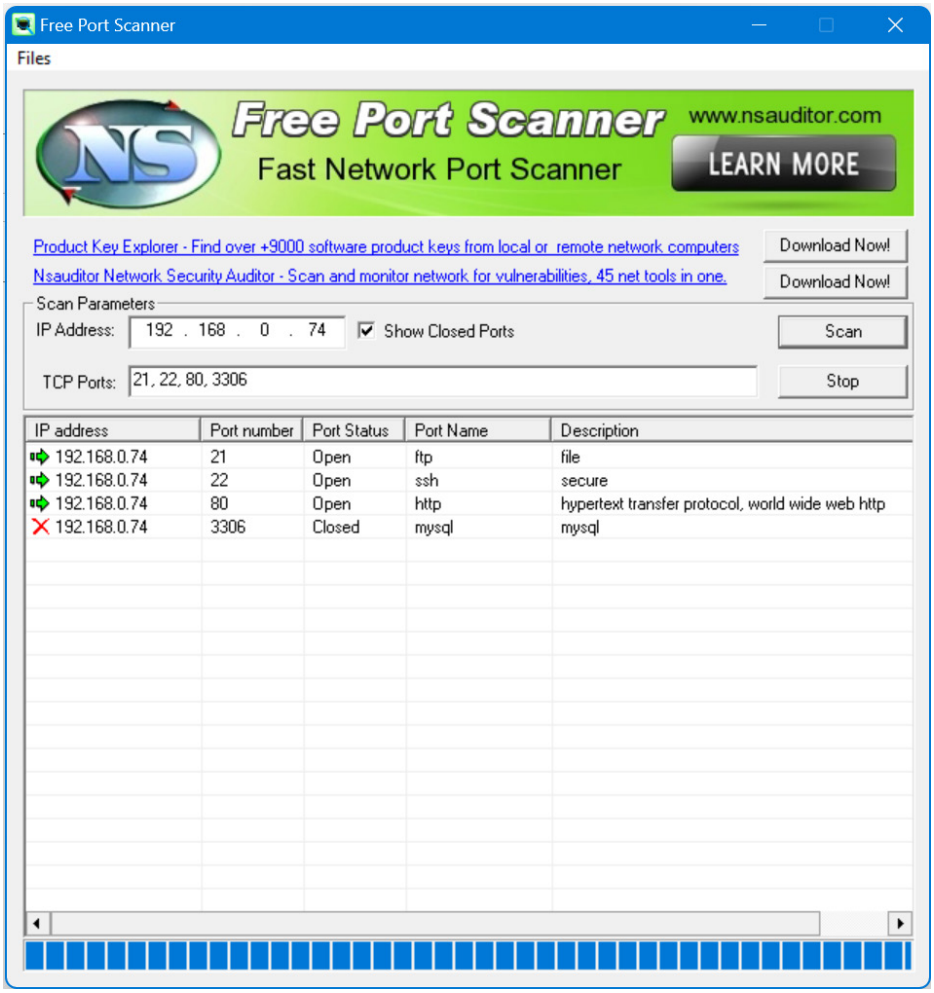



Figura 40. Ventana principal de freeportscanner, un programa para escanear puertos

En el ejemplo pueden verse los puertos 21, 22, 80, y 3306 correspondientes a ftp, ssh, http y mysql respectivamente. Esto indica que el servidor u ordenador con dirección IP 192.168.0.74 tiene disponibles los servicios precisamente con los puertos que se encuentran abiertos 21, 22 y 80 y no disponible el servicio de base de datos mysql puerto 3306. Más adelante ejemplificaremos el uso de esta potente herramienta, pero en general nos permitirá conocer si tenemos o no acceso a los servicios que estén ejecutándose en el Servidor de Linux Ubuntu.

## Bitvise SSH Client

El siguiente programa por instalar es **Bitvise SSH Client** versión 9.33. Este es un potente cliente de reenvío de puertos SSH2 con muchas características, que incluyen: tunelización dinámica a través de un proxy integrado que admite los protocolos de tunelización de proxy SOCKS4, SOCKS4A, SOCKS5 y HTTP CONNECT https://www.bitvise.com)

Este *software* puede descargarse directamente desde el sitio web oficial, haciendo clic en el siguiente link: ( https://www.bitvise.com/ssh-client-download )

Bitvise SSH Client se puede utilizar de forma libre, pero no existe una licencia gratuita para el Servidor SSH Bitvise. Cabe recalcar que una licencia adquirida para Bitvise SSH Server no implica una licencia para otro *software* ofrecido en bitvise.com.

Conocido este particular, en este apartado no vamos a instalar el servidor SSH Bitvise en Windows una debido a su licencia de pago y la otra porque el servidor SSH lo instalaremos en Linux Ubuntu Server mediante el paquete de openssh-server. Este proceso lo explicaremos en un capítulo más adelante.

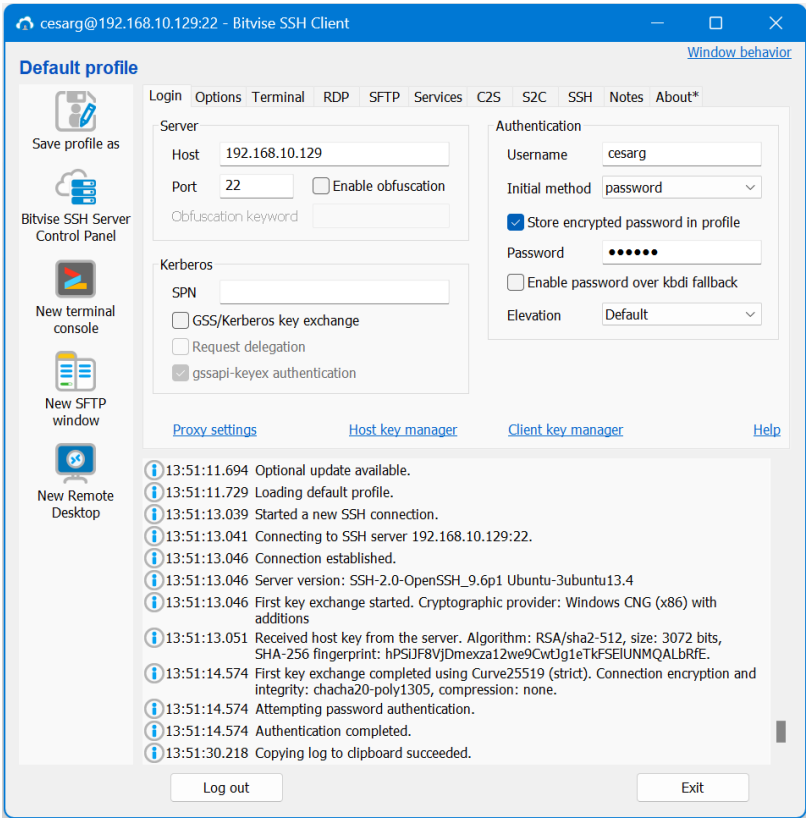



Figura 41. Bitvise SSH Client, un cliente para conectarse a un servidor SSH

Para establecer una conexión con el servidor, como puede apreciarse en la ventana de Bitvise SSH Client, se ingresa la dirección IP del servidor remoto en el campo que dice "Host", que en el ejemplo es 192.168.10.129. A continuación, se ingresa el número de puerto 22, enseguida el nombre de usuario en el campo "Username", y finalmente debemos seleccionar *password* en el campo "*Initial method*". Más abajo se indica una serie de mensajes que informan sobre el proceso de establecimiento de la conexión SSH seguro contra el servidor. A continuación, se detalla el flujo de eventos:

- El cliente SSH recibe la clave pública del servidor.
- La clave se autentica utilizando SHA-256.
- El primer intercambio de claves se realiza con el algoritmo Curve25519.
- La conexión se cifra y asegura utilizando el algoritmo chacha20-poly1305.
- Se intenta la autenticación por contraseña.
- La autenticación se completa con éxito.

Una vez iniciada la sesión se habilitarán algunas opciones, como iconos en la parte izquierda de la ventana. Básicamente nos interesan dos de ellas, las cuales son:

New terminal console y New SFTP window

La primera es para ingresar a un modo terminal. En nuestro caso, básicamente funciona como una terminal de Linux, donde se pueden ejecutar los diferentes comandos Linux. La segunda opción se ingresa a una ventana tipo explorador de archivos con dos paneles: el izquierdo es del sistema de archivos local (Windows) y el derecho es del sistema de archivos remoto (Linux).

## MySQL Workbench 8.0 CE

MySQL Workbench 8.0 CE para Windows también es compatible con macOS y Linux, y la instalación es bastante sencilla con asistentes de instalación guiada. MySQL Workbench 8.0 CE es una herramienta cliente visual de diseño para modelado, desarrollo, administración y mantenimiento de bases de datos MySQL. Esta aplicación proporciona una interfaz gráfica para trabajar con bases de datos MySQL, lo que facilita a los desarrolladores y administradores de bases de datos realizar tareas complejas con mayor eficiencia.

Permite modelar diagramas de entidad-relación para bases de datos MySQL. Puede emplearse para diseñar un esquema de base de datos nueva, documentar una ya existente o realizar una migración compleja. La aplicación puede presentar una representación visual de las tablas, vistas, procedimientos almacenados y claves foráneas de la base de datos. Además, es capaz de sincronizar el modelo en desarrollo con la base de datos real. MySQL Workbench también puede generar *scripts* necesarios para crear una base de datos que se haya dibujado en el esquema. Permite realizar migraciones con versiones anteriores de MySQL y, además, el programa cuenta con soporte para migración de Microsoft SQL Server, Microsoft Access, Sybase ASE o PostreSQL. MySQL Workbench 8.0 CE. También puede descargarse libremente desde el sitio web oficial ( https://dev.mysql.com/downloads/workbench/ )

### Características principales

- Modelado de datos
- Desarrollo SQL
- Administración de servidores
- Migración de datos
- Visualización y análisis
- Generación de reportes

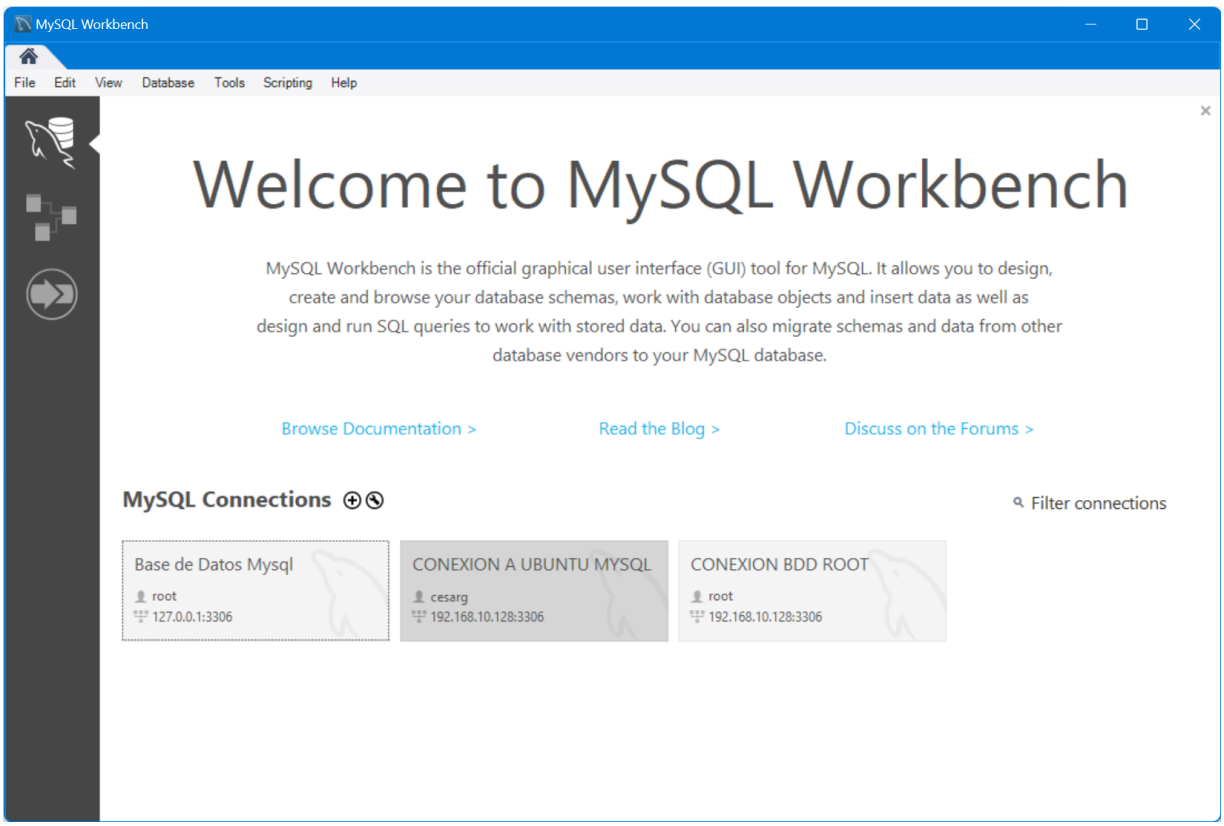



Figura 42. MySQL Workbench 8.0 CE es una herramienta para la gestión de bases de datos MySQL.

MySQL Workbench es uno de los primeros productos de la familia MySQL en ofrecer dos versiones distintas: una de código abierto y una versión comercial. La "Community Edition" es un producto completamente

funcional sin limitaciones. Como base para todas las demás ediciones, se beneficiará de todos los desarrollos futuros. La versión comercial, "Standard Edition", amplía la Community Edition con varios módulos y complementos adicionales.

## Arquitectura cliente-servidor

La arquitectura cliente-servidor es el modelo de diseño de *software* en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados **servidores**, y los demandantes, llamados **clientes**. En el contexto de este capítulo, nuestro servidor es Linux **Ubuntu Server** y los clientes son las herramientas de **Windows** que repasamos anteriormente.

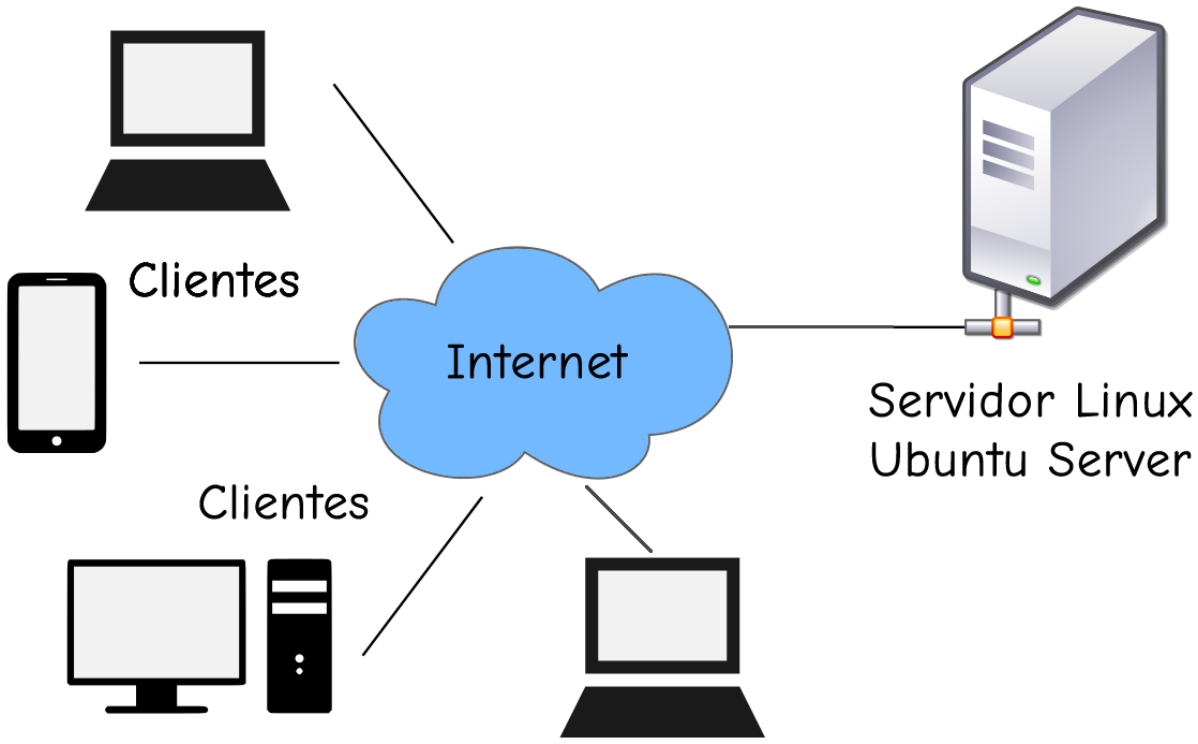



Figura 43. Arquitectura Cliente-Servidor

Componentes importantes:

1. **El cliente (Windows):** Es el proceso que inicia la comunicación. Solicita datos o acciones a través de una interfaz (como Bitvise o MySQL Workbench).
2. **El servidor (Ubuntu):** Es el proceso que espera pasivamente las solicitudes. Procesa la petición, valida los permisos y devuelve una respuesta.
3. **La red (protocolo):** El lenguaje común que ambos hablan (SSH, FTP, MySQL/TCP).

**Nota técnica:** En este modelo, el servidor es "centralizado" y puede atender a múltiples clientes simultáneamente, lo que permite que varios administradores trabajen sobre el mismo servidor Ubuntu desde diferentes estaciones Windows.

Seguridad en conexiones remotas

Administrar un servidor de forma remota involucra abrir "puertas" (puertos) al mundo. Si estas puertas no están protegidas, el servidor será vulnerable a ataques de fuerza bruta o interceptación de datos.

1. Cifrado de extremo a extremo

Toda conexión entre un cliente Windows y un servidor Linux debe estar cifrada, para esto normalmente se utilizan los siguientes protocolos:

- **Protocolos seguros:** SSH (puerto 22) y SFTP (puerto 22) cifran tanto el usuario/contraseña como los datos transmitidos.
- **Protocolos inseguros:** Evitan el uso de Telnet o FTP clásico, ya que envían la información en texto plano, permitiendo que un atacante en la misma red "lea" tus credenciales.

2. Autenticación por llaves públicas (SSH Keys)

En lugar de usar contraseñas tradicionales, la seguridad profesional utiliza un par de llaves:

- **Llave privada:** Se queda en su cliente Windows (Bitvise).
- **Llave pública:** Se instala en el servidor Ubuntu. Esto elimina el riesgo de ataques de diccionario, ya que sin el archivo físico de la llave privada, es imposible entrar al sistema.

3. El concepto de "puertos" y *firewalls*

La seguridad requiere cerrar todo lo que no se usa.

- **Free Port Scanner:** Esta herramienta nos permite auditar nuestro propio servidor para asegurar que solo los puertos necesarios (como el 22 para SSH o el 3306 para MySQL) estén abiertos.
- **Principio de menor privilegio:** Si solamente se administra el servidor, se puede configurar el *firewall* del servidor (**UFW**) para que solo acepte conexiones desde la dirección IP específica de su red Windows.

# CAPITULO VI INSTALACIÓN Y CONFIGURACIÓN DE MYSQL

## Contenido del capítulo VI

- Introducción a MySQL
- Características de MySQL
- Instalar y administrar el proceso de MySQL
- Bases de datos por defecto
- Ajuste de autenticación y privilegios de usuario
- Tomar precauciones de seguridad
- Permitiendo conexiones remotas
- Comandos esenciales de MySQL

## Introducción a MySQL

MySQL es un sistema de gestión de bases de datos relacional (RDBMS) de código abierto. Fue desarrollado originalmente por MySQL AB y es actualmente propiedad de Oracle Corporation. MySQL se utiliza para almacenar y gestionar grandes volúmenes de datos, y es conocido por su velocidad, confiabilidad y facilidad de uso. Cuando nos referimos al término SQL y MySQL debemos diferenciar dos cosas: la primera es que SQL se refiere a un lenguaje universal de consulta de bases de datos, mientras que MySQL se refiere a un sistema de gestión de bases de datos específico que utiliza SQL para interactuar con la base de datos. Percibir esta diferencia es básico para aquellos que trabajan con bases de datos y desean utilizar MySQL como su sistema preferido. SQL es un acrónimo en inglés para *Structured Query Language*. Un lenguaje de consulta estructurado es una forma de lenguaje de programación que permite gestionar y manejar datos de una base de datos tiene la capacidad de realizar *cálculos avanzados* como el *álgebra* y es utilizado por la mayoría de las empresas para almacenan datos en una base de datos. Por ejemplo, una consulta sencilla sería:

```
mysql> USE MIBASE;
mysql> SELECT *from table;
```

Donde "MIBASE" es el nombre de la base de datos y "*table*" es el nombre de la tabla, para el ejemplo la instrucción "*SELECT*" desplegará un listado del contenido de la tabla. La base de datos MySQL se utiliza en un sinnúmero de aplicaciones como por ejemplo para sitios web como los creados mediante gestores de contenido (CMS), así tenemos Wordpress y Joomla, entre otros; para sistemas de gestión de aprendizaje (LMS) usados para crear, alojar y compartir e-learning como es el caso de Moodle, Dokeos entre otros y que básicamente son sitios web que utilizan bases de datos como es MySQL. También se utiliza en sistemas de gestión de recursos empresariales (ERP) y sistemas de gestión de relaciones con el cliente (CRM), para análisis de datos permitiendo consultas complejas y análisis en tiempo real.

Mysql puede ejecutarse bajo cualquier plataforma incluyendo Linux, Windows y Mac. De hecho, es compatible con más plataformas que SQL Server una base de datos de Microsoft. Otra facilidad que tiene MySQL es que permite conexiones remotas, así podemos utilizar el cliente **MySQL Workbench** en Windows y conectarnos a un servidor Mysql en Linux. Existen varias opciones para gestionar Mysql entre las cuales puedo destacar a **phpMyAdmin,** que es una aplicación web desarrollada con php y permite administrar la base de datos, por ejemplo, crear tablas: en cada tabla puede crear y modificar campos, lanzar sentencias SQL, etc. Otra forma de gestionar es mediante el uso de un **OBDC,** del inglés *Open DataBase Connectivity*. El objetivo de ODBC es

hacer posible el acceder a cualquier dato desde cualquier aplicación, sin importar qué sistema de gestión de bases de datos almacene los datos.

## Características de MySQL

- Libre: MySQL es un *software* de código abierto y multiplataforma.
- Escalabilidad: Es escalable debido a que puede adaptarse a diferentes aplicaciones.
- Seguro: En cuanto a seguridad, proporciona opciones de autenticación y cifrado de datos para garantizar la seguridad.
- Compatible: MySQL es compatible con numerosos lenguajes de programación y herramientas de desarrollo.
- Rendimiento: Ofrece una recuperación de datos rápida y eficiente, ideal para aplicaciones de alto rendimiento.
- Flexible: Puede ser personalizado y configurado para adaptarse a las necesidades específicas de una aplicación.

## Instalar y administrar el proceso de MySQL

Para instalar mysql server simplemente ejecutamos en la terminal el siguiente comando:

```
root@svrlinux:~# apt install mysql-server ┘ Enter
```

Enseguida se puede verificar que el servicio MySQL está en ejecución o funcionando correctamente. Al ejecutar este comando, el sistema mostrará información sobre si el servicio MySQL está activo, inactivo, detenido o si ha encontrado algún error con el siguiente comando:

```
root@svrlinux:~# service mysql status ┘ Enter
```

Entonces se desplegará la siguiente información

```
root@svrlinux:/home/cesarg# service mysql status
• mysql.service - MySQL Community Server
     Loaded: loaded (/usr/lib/systemd/system/mysql.service; enabled; preset: enabled)
     Active: active (running) since Wed 2024-07-10 17:19:41 UTC; 2min 15s ago
    Process: 10199 ExecStartPre=/usr/share/mysql/mysql-systemd-start pre (code=exited, status=0/SUCCESS)
   Main PID: 10208 (mysqld)
     Status: "Server is operational"
      Tasks: 37 (limit: 4556)
     Memory: 365.5M (peak: 379.8M)
        CPU: 994ms
     CGroup: /system.slice/mysql.service
             └─10208 /usr/sbin/mysqld

jul 10 17:19:40 svrlinux systemd[1]: Starting mysql.service - MySQL Community Server...
jul 10 17:19:41 svrlinux systemd[1]: Started mysql.service - MySQL Community Server.
root@svrlinux:/home/cesarg# _
```

Figura 44. Estado activo (en ejecución) del servicio de MySQL

Concretamente, esto nos indica que el servicio está en ejecución y está funcionando de forma correcta. En sistemas más recientes, basados en **systemd**, como las distribuciones modernas de Ubuntu, se puede usar **systemctl** en lugar de service y obtendremos el mismo resultado, por cuanto de aquí en adelante ocuparemos este comando. Por ejemplo:

```
root@svrlinux:~# systemctl status mysql ┘ Enter
```

Systemd es un sistema de inicio y gestor de servicios para Linux, diseñado para proporcionar una gestión más eficiente y completa de los servicios del sistema. Es una parte fundamental de muchas distribuciones Linux modernas, reemplazando a los sistemas de inicio más antiguos como System V, init y Upstart. Systemd se ha convertido en el estándar de facto en muchas distribuciones Linux debido a sus características avanzadas y la mejora en el rendimiento y la gestión del sistema. La instrucción de manera general es como se indica a continuación:

Principales comandos de gestión del servicio de mysql

root@linux:~# systemctl [start|stop|restart|reload|status] [*nombre del servicio*]

Iniciar el servicio de mysql:

root@svrlinux:~# systemctl start mysql ↵ Enter

Detener el servicio de mysql:

root@svrlinux:~# systemctl stop mysql ↵ Enter

Reiniciar el servicio de mysql:

root@svrlinux:~# systemctl restart mysql ↵ Enter

Recargar la configuración del servicio de mysql sin detenerlo:

root@svrlinux:~# systemctl reload mysql ↵ Enter

Mostrar el estado del servicio de mysql:

root@svrlinux:~# systemctl status mysql ↵ Enter

Otros ejemplos de comandos de gestión del servicio de mysql

root@linux:~# systemctl [enable|disable|is-active|is-enabled] [*nombre del servicio*]

Habilita un servicio para que se inicie automáticamente al arrancar el sistema:

root@svrlinux:~# systemctl enable mysql ↵ Enter

Deshabilita un servicio para que no se inicie automáticamente al arrancar el sistema:

root@svrlinux:~# systemctl disable mysql ↵ Enter

Verifica si un servicio está activo:

root@svrlinux:~# systemctl is-active mysql ↵ Enter

Verifica si un servicio está habilitado para iniciarse al arrancar:

root@svrlinux:~# systemctl is-enabled mysql ↵ Enter

Conocer estos comandos previamente nos permitirá realizar un seguimiento a nuestro servicio en cuanto vayamos a realizar las configuraciones del servidor de MySQL y no tener inconvenientes. También podemos acceder a una documentación oficial de MySQL en (https://dev.mysql.com/doc/) donde encontraremos una manual de referencia, todos los productos de MySQL Enterprise, un manual de referencia de MySQL Workbench. También encontraremos información sobre MySQL InnoDB Cluster, una tecnología que permite el *clustering* de bases de datos en memoria en un ambiente de no compartición y acceso a conectores y APIs de MySQL, que son los controladores y bibliotecas que se utilizan para conectar aplicaciones en diferentes lenguajes de programación a los servidores de bases de datos MySQL.

Tal como viene mysql-server luego de la instalación, el mismo no posee una configuración segura, podemos comprobarlo porque tenemos acceso libre a la base de datos sin necesidad de una contraseña, solamente con ingresar el siguiente comando, donde -u es un parámetro que especifica un *usuario* y *root* es el nombre de usuario administrador.

```
root@svrlinux:~# mysql -u root ↵ Enter
```

La salida para este comando es la que se muestra en la siguiente ventana. Se puede apreciar que la versión de MySQL server es la 8.0.37 y, más abajo, las bases de datos que se instalan por defecto.

También se podría ingresar desde una terminal directamente a la base de datos utilizando simplemente el comando:

```
root@svrlinux:~# mysql ↵ Enter
```

```
root@svrlinux:/home/cesarg# mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.37-0ubuntu0.24.04.1 (Ubuntu)

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0,00 sec)

mysql> _
```

Figura 45. Entrada a la base de datos Mysql con el usuario *root*, presentación de las bases de datos por defecto

## Bases de datos por defecto

Puede observarse que MySQL trae instalado por defecto cuatro bases de datos, las cuales se describe brevemente a continuación:

1. ***Information_schema:*** Es un esquema que contiene información sobre todas las bases de datos que el servidor MySQL maneja. Es como un catálogo de metadatos.
2. **Mysql:** Este esquema contiene las tablas que el servidor MySQL usa para gestionar cuentas de usuario, privilegios, y la configuración general del servidor.
3. **Performance_schema:** Es un esquema diseñado para recolectar información de desempeño y monitoreo del servidor MySQL.
4. **Sys:** Es un esquema que proporciona una vista amigable y simplificada de las métricas y estadísticas presentes en *performance_schema*.

## Ajuste de autenticación y privilegios de usuario

Puede usted notar que el usuario *root* no está configurado para autenticarse con una contraseña al conectarse al atew de MySQL, sino que se ingresa directamente a la cuenta. Para los sistemas Ubuntu que estén usando MySQL 8.0.37 o superior, el usuario *root* de MySQL está configurado de forma predeterminada para autenticarse usando el **plugin auth_socket** en vez de una contraseña. En muchos casos, esto permite que la seguridad y usabilidad sea mayor pero también puede complicar las cosas cuando deba permitir que un programa externo tal como phpMyAdmin tenga acceso al usuario. Usted puede verificarlo con el siguiente comando, primero seleccionamos la base de datos mysql:

```
mysql> use mysql ┘ Enter
mysql> select user, host, plugin from mysql.user; ┘ Enter
```

Deberá cambiar el método de autenticación de **auth_socket** a **mysql_native_password** para usar una contraseña y poder conectarse a MySQL como *root* mediante una clave. Observe en la siguiente imagen la consulta respectiva.

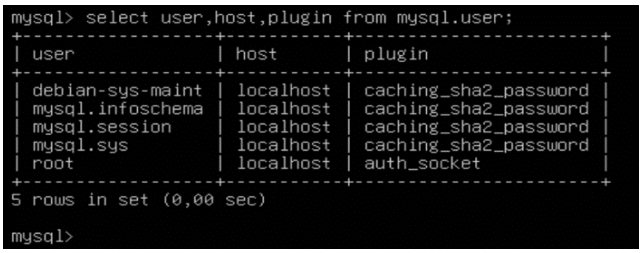



Figura 46. Consulta a la tabla mysql.user, auth_socket

Mencionado lo anterior, a continuación ejecutamos el siguiente comando de MySQL.

```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY '[contraseña]';
```

- **ALTER USER**: Este comando se utiliza para modificar las propiedades de un usuario existente en MySQL.
- **'root'@'localhost'**: Especifica el usuario *'root'* que se conecta desde *'localhost'*. Si quisiera modificar un usuario diferente o un usuario que se conecta desde una dirección IP específica, estos valores cambiarían.
- **IDENTIFIED WITH mysql_native_password**: Esto indica que el usuario debe autenticarse utilizando el método mysql_native_password. Este método es uno de los mecanismos de autenticación de MySQL.
- **BY '[contraseña]'**: Aquí debe reemplazar [contraseña] con la nueva contraseña que desea asignar al usuario.

Ejemplo para el uso de ALTER

```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY '!123';
mysql> select user, host, plugin from mysql.user; ┘ Enter
```

Nos damos cuenta de que el campo se ha alterado y, en lugar del método de autenticación de **auth_socket,** ahora es **mysql_native_password**

```
mysql> select user,host,plugin from mysql.user;
+------------------+-----------+-----------------------+
| user             | host      | plugin                |
+------------------+-----------+-----------------------+
| debian-sys-maint | localhost | caching_sha2_password |
| mysql.infoschema | localhost | caching_sha2_password |
| mysql.session    | localhost | caching_sha2_password |
| mysql.sys        | localhost | caching_sha2_password |
| root             | localhost | mysql_native_password |
+------------------+-----------+-----------------------+
5 rows in set (0,00 sec)
```

Figura 47. Consulta a la tabla mysql.user, mysql_native_password

Si salimos de mysql y volvemos a ingresar, podemos percatarnos de que ya no ingresa directamente a mysql y obtendremos el mensaje de: Error 1045 (28000): Access denied for user 'root'@'localhost' (using password: NO)

mysql> exit ↵ Enter

mysql> use mysql ↵ Enter

Ahora, para ingresar, se debe pasar como parámetro el nombre de usuario -u y la contraseña -p como se indica a continuación:

root@svrlinux:~# mysql -u root -p ↵ Enter

Enter password: ********

Se puede visualizar también en la columna otro método de autenticación como es el **caching_sha2_password**. El caching_sha2_password y sha256_password son *plugins* de autenticación que proporcionan un cifrado de contraseña más seguro que el mysql_native_password plugin, el **caching_sha2_password** proporciona mejor rendimiento que sha256_password.

Debido a estas características superiores de seguridad y rendimiento de **caching_sha2_password**, es a partir de MySQL 8.0 el complemento de autenticación preferido, y también es el complemento predeterminado de autenticación en lugar de mysql_native_password. Sin embargo, hay que tomar en cuenta posibles problemas de compatibilidad con clientes o conectores que pueden no soportar este método de autenticación.

Para más información puede consultar en la fuente del sitio web oficial en: https://dev.mysql.com/doc/refman/8.0/en/upgrading-from-previous-series.html#upgrade-caching-sha2-password

## Tomar precauciones de seguridad

Consiste primeramente en llamar a un script de seguridad para realizar algunos ajustes importantes. El comando para este proceso es el siguiente:

root@svrlinux:~# mysql_secure_installation ↵ Enter

A continuación, el programa le guiará por los siguientes pasos. Por ejemplo, puede instalar el Validate Password Plugin, que comprueba la seguridad de las contraseñas. Si un usuario introduce una contraseña demasiado débil, se le pide que la cambie por otra más segura. El nivel de seguridad que quiere exigir lo puede definir al introducir 0 (para débil), 1 (para medio) o 2 (para alto). En el siguiente paso le pedirá que introduzca una contraseña para el usuario *root*. De nuevo, la contraseña se introduce de forma oculta. A continuación, debe repetir la contraseña que ha introducido anteriormente. Si ha activado el Validate Password Plugin, también comprobará la seguridad de la contraseña que ha introducido. También le puede preguntar sobre el acceso remoto desde el usuario root podemos responder que "si"

Puede introducir Y para responder "sí" a las preguntas que se le hagan en los pasos posteriores: por ejemplo, para eliminar el usuario de prueba anónimo que se generó al principio o para eliminar la base de datos de

prueba. En caso de no querer eliminar a ninguno, puede introducir N para responder "no" a las preguntas que se le planteen.

## Permitiendo conexiones remotas

Con el comando #nmap localhost podemos ver que el puerto 3306 de mysql se encuentra abierto, sin embargo, es posible que la conexión remota a mysql no está habilitada, y solo se permita a 127.0.0.1. Para cambiar esto se tiene que editar el archivo de configuración de mysql, este archivo se encuentra en la siguiente ruta en Linux Ubuntu /etc/mysql/mysql.conf.d/mysqld.cnf

Aproximadamente en la línea número 31, cambiamos la dirección en el archivo de configuración de MySQL y reiniciamos.

```
Bind-address = 0.0.0.0

root@svrlinux:~# systemctl restart mysql ┘ Enter
```

Una vez reiniciado el servidor mysql debemos considerar lo siguiente:

**Accesibilidad:** El *bind-address* especifica la dirección IP en la que un servicio debe "escuchar" las conexiones entrantes. Esto es importante porque determina desde qué interfaces de red el servicio aceptará conexiones. Por ejemplo, si configura bind-address en 127.0.0.1, el servicio solo aceptará conexiones desde la misma máquina (*localhost*). Si lo configura en 0.0.0.0, el servicio escuchará en todas las interfaces de red disponibles, permitiendo conexiones desde cualquier dirección IP.

**Seguridad:** Hacer que MySQL esté accesible desde todas las interfaces de red puede implicar riesgos de seguridad. Permitirá conexiones remotas desde cualquier dirección IP, lo cual puede ser peligroso si no se configura adecuadamente la seguridad de MySQL (como el uso de contraseñas fuertes, la configuración de *firewall* adecuada, y permisos de usuario bien gestionados). Aquí el administrador de la base de datos debe considerar si el servicio debe ser accedido localmente o desde que dirección IP remota específica debe permitirse. Otros servicios como Redis o PostgreSQL también permiten configurar bind-address para controlar el acceso.

MySQL incluye varias funcionalidades integradas para proteger los datos, como la autenticación de usuarios, el control de acceso basado en roles, el cifrado de conexiones mediante SSL/TLS, y la capacidad de enmascarar datos sensibles.

En la pantalla siguiente vemos que todavía no se permite la conexión remota desde el cliente MySQL Workbench que está instalado en Windows 11.

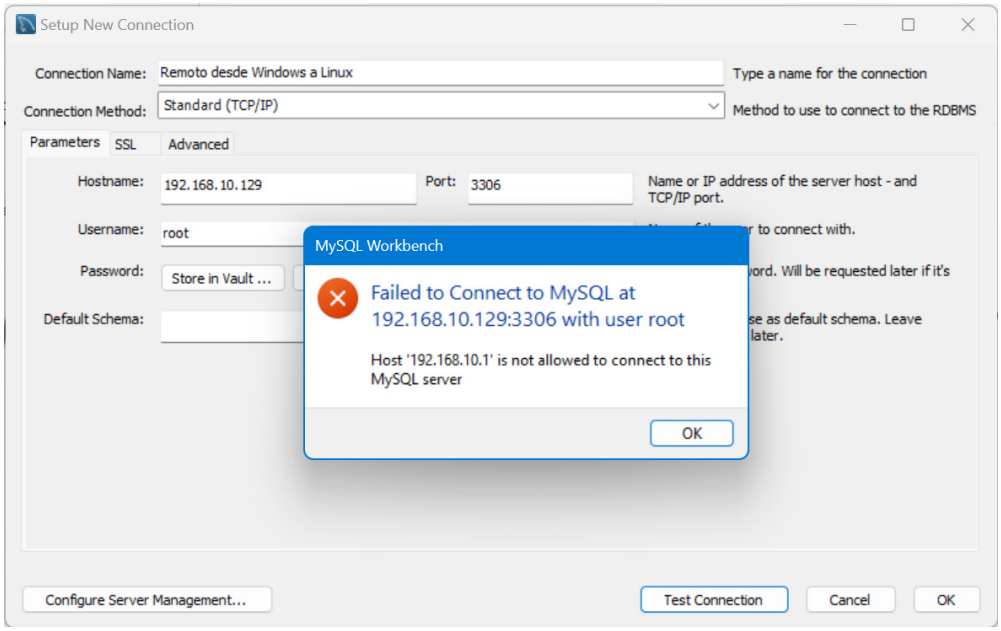



Figura 48. Conexión fallida desde MySQL Workbench al servidor MySQL en Linux.

Esto se debe a que la base de datos no permite la conexión a la dirección IP de Linux del ejemplo 192.168.10.129 con el usuario *root*. El mensaje dice también que el *host* 192.168.10.1 no tiene permitido conectarse a este servidor MySQL.

```
mysql> use mysql
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select user, host, plugin from mysql.user;
+------------------+-----------+-----------------------+
| user             | host      | plugin                |
+------------------+-----------+-----------------------+
| debian-sys-maint | localhost | caching_sha2_password |
| mysql.infoschema | localhost | caching_sha2_password |
| mysql.session    | localhost | caching_sha2_password |
| mysql.sys        | localhost | caching_sha2_password |
| root             | localhost | mysql_native_password |
+------------------+-----------+-----------------------+
5 rows in set (0,00 sec)
```

Figura 49. Consulta mediante *select* a la tabla mysql.user.

Si nos fijamos en el campo *host* vemos que la conexión del usuario *root* está permitida únicamente para *localhost*, entonces se debe permitir para la dirección IP 192.168.10.1 específica en este caso, o cualquier ip (%), por lo tanto, debemos escribir el siguiente comando:

**mysql**>CREATE USER 'root'@'%' IDENTIFIED WITH caching_sha2_password BY 'clave'

**mysql**>GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' WITH GRANT OPTION;

Ahora desde el cliente de MySQL Workbech se permite ingresar la clave del usuario *root* y acceder con normalidad a la base de datos. Si volvemos a realizar la consulta mysql > select user, host, plugin from mysql. user nos daremos cuenta de que se ha creado una nueva fila con el usuario *root* y en el campo *host* el símbolo %, que significa acceso desde cualquier dirección IP excepto *localhost*; esto quiere decir que por ejemplo si quisiéramos ingresar desde la terminal Linux a mysql con la clave de este usuario, *root* no nos lo permitirá, a menos que haya ingresado la misma clave.

Es como si tuviéramos dos usuarios *root:* uno para acceder desde *localhost,* es decir, desde la propia máquina con la primera clave y el otro *root,* con acceso desde cualquier dirección excepto *localhost.*

En la siguiente figura se muestra esta particularidad, ya depende mucho del administrador cómo desea y desde qué usuarios administrar las bases de datos, tanto si es desde *localhost* o remotamente desde cualquier IP, o inclusive desde alguna dirección IP en particular. Lo más usual es crear un usuario diferente a *root* y que no tenga todos los privilegios administrativos, solamente los necesarios para acceder a su respectiva base de datos. A continuación, observamos una consulta SQL con *select* a los campos *user*, *host* y *plugin* de la tabla mysql. user, esta forma parte del sistema de mysql, la cual se encarga de almacenar información sobre los usuarios, sus permisos y otras configuraciones relacionadas con la seguridad.

```
mysql> select user, host, plugin from mysql.user;
+------------------+-----------+-----------------------+
| user             | host      | plugin                |
+------------------+-----------+-----------------------+
| root             | %         | mysql_native_password |
| debian-sys-maint | localhost | caching_sha2_password |
| mysql.infoschema | localhost | caching_sha2_password |
| mysql.session    | localhost | caching_sha2_password |
| mysql.sys        | localhost | caching_sha2_password |
| root             | localhost | mysql_native_password |
+------------------+-----------+-----------------------+
6 rows in set (0,00 sec)

mysql> quit
Bye
root@svrlinux:/home/cesarg# mysql -u root -p
Enter password:
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: YES)
root@svrlinux:/home/cesarg# _
```

Figura 50. Acceso denegado para el usuario *root* desde *localhost*, la clave es incorrecta.

En este sentido, es importante tomar en cuenta la clave que hemos asignado al usuario *root* con *host* % o al usuario *root* con *host localhost*. A continuación, tenemos el cliente de MySQL Workbench para acceder remotamente al *host* con IP: 192.168.10.129 a través del puerto 3306 y usuario *root*.

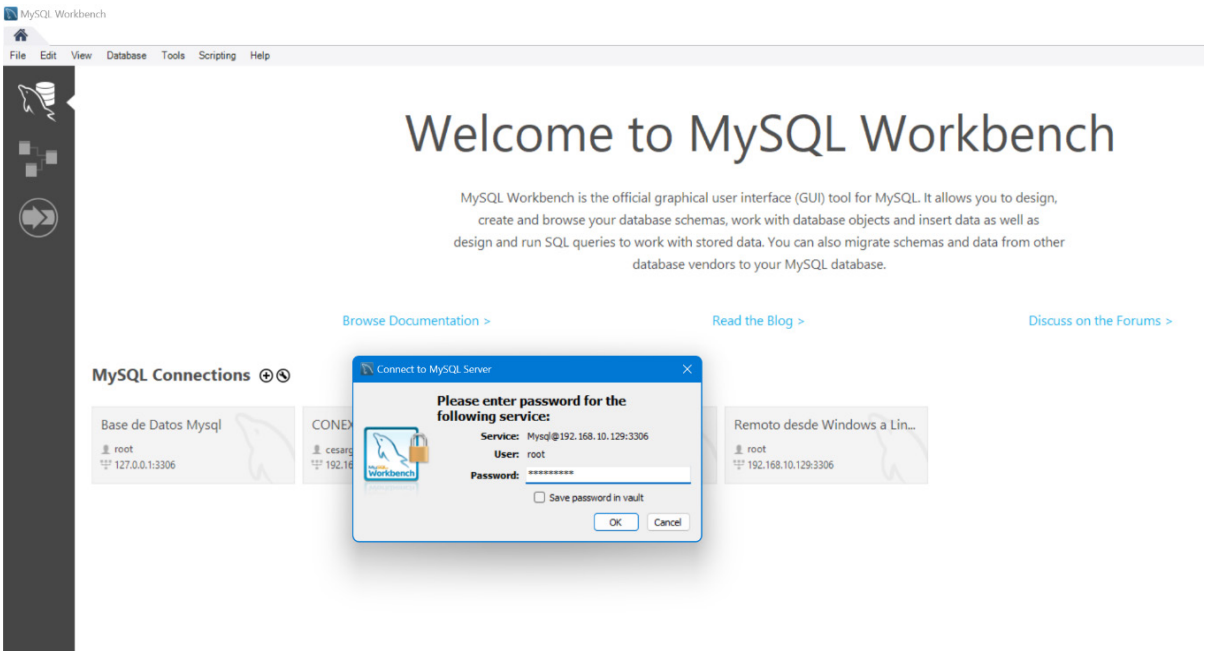



Figura 51. Acceso remoto a la base de datos desde el cliente MySQL Workbench

En un entorno de desarrollo o pruebas donde no quiera imponer restricciones estrictas sobre los usuarios administradores, puede ser de mucha utilidad, sin embargo, en un entorno de producción, se debería considerar

cuidadosamente las implicaciones de seguridad y rendimiento antes de permitir un acceso ilimitado. Permitir que el usuario *root* se conecte remotamente y sin restricciones puede representar un riesgo de seguridad significativo. Es recomendable restringir el acceso lo máximo posible y utilizar usuarios con privilegios limitados para tareas específicas. Además, asegúrese de que las conexiones se realicen a través de conexiones seguras y autenticadas correctamente.

Una vez ingresados al sistema se nos presentarán todas las bases de datos del cual el usuario tiene privilegios y sus respectivas tablas. En esta herramienta visual de diseño, podemos seleccionar la base de datos y realizar consultas SQL a sus tablas. En el ejemplo de la figura siguiente seleccionamos la base de datos MIBASE y realizamos una consulta con SELECT a la tabla empleado.

Por otro lado, sepa que Mysql Workbench incluye herramientas para migrar bases de datos desde otros sistemas, como Microsoft SQL Server, PostgreSQL, o incluso desde archivos de texto. Esto es útil cuando se necesita mover datos a MySQL desde otra plataforma. En cuanto a la ejecución de consultas SQL, Mysql Workbench ofrece un editor SQL con resaltado de sintaxis, autocompletado y herramientas para depurar y optimizar consultas. Más adelante analizaremos algunos comandos esenciales de MySQL, y cómo crear la base de datos del ejemplo.

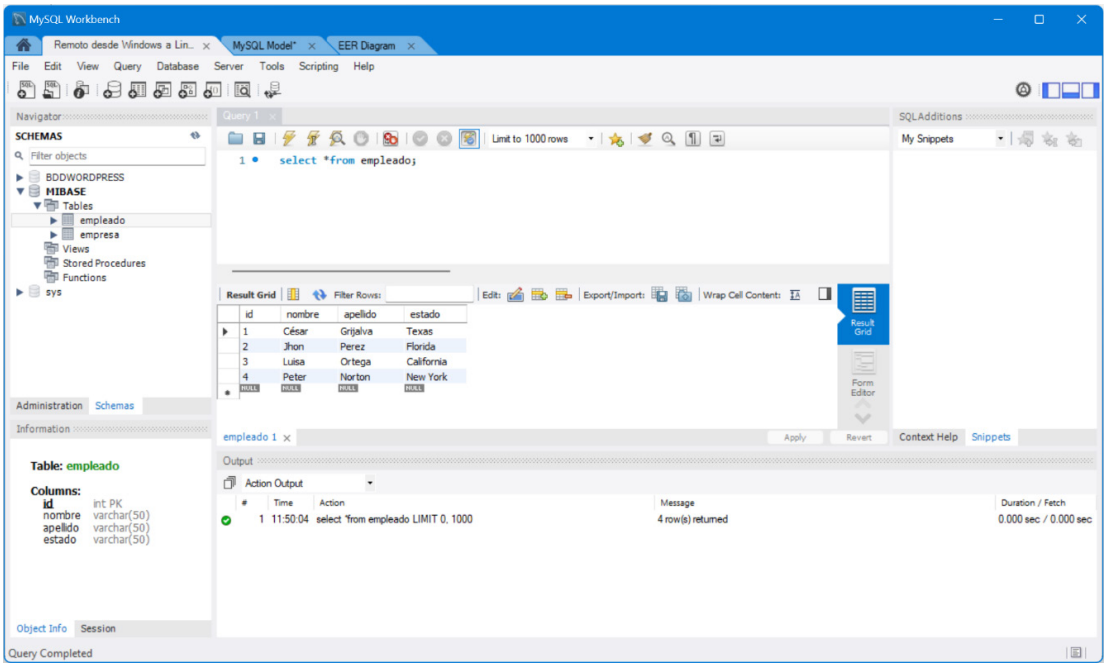



Figura 52. Interfaz gráfica de Mysql Workbech, presentando las tablas y las bases de datos

El cliente de MySQL Workbench permite también generar diagramas utilizando las tablas de la base de datos. Para esto debemos seleccionar en el menú de opciones Database y luego Reverse Engineer, lo que nos presentará una ventana de conexión donde debemos llenar todos los parámetros, dirección IP, puerto, usuario y por supuesto la clave del usuario., Seleccionamos la base de datos del ejemplo MIBASE y enseguida se presentarán las ventanas MySQL Model y EER Diagram, como se indica en la figura.

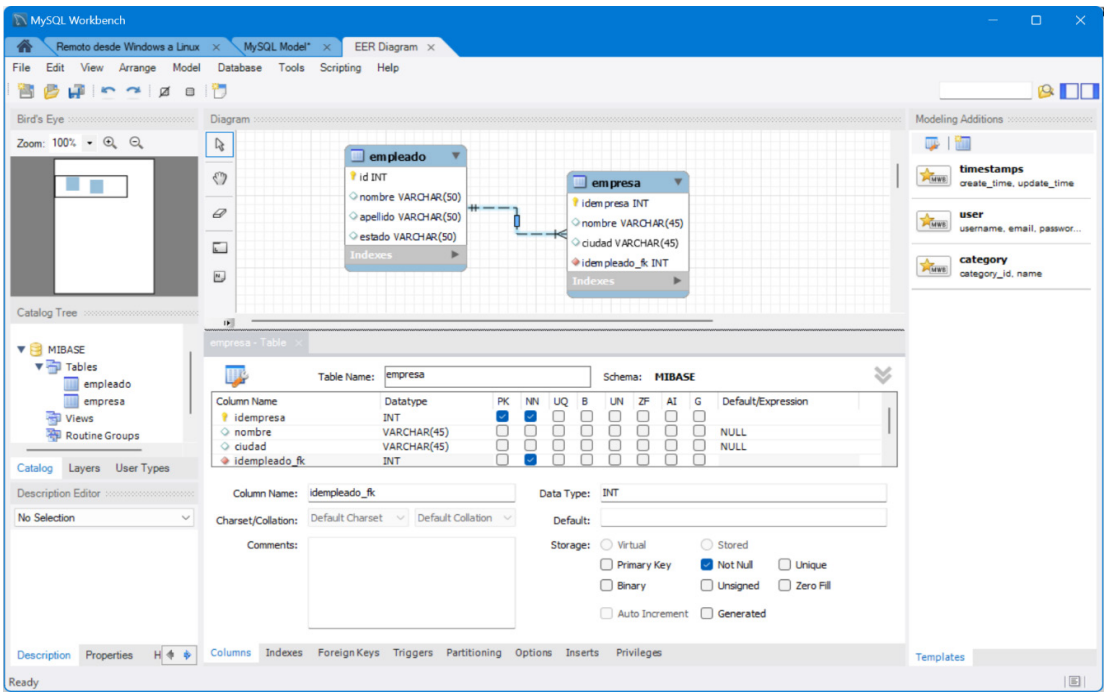



Figura 53. Presentación de las tablas en la interfaz gráfica para los diagramas EER

En la pestaña MySQL Model se presentan algunas opciones relacionadas con las bases de datos, como crear tablas, crear vistas, manejar usuarios, roles, etc. Los diagramas de Entidad Relación EER, cuyo significado es *Enhanced entity-relationship* o Diagrama Entidad-Relación Mejorado, es la versión avanzada del modelo ER básico y nos ayudará a crear y mantener la base de datos con mayor detalle. Mediante esta herramienta podemos desarrollar diagramas desde muy básicos a muy complejos.

Volviendo a Linux, proponemos: si por ejemplo ingresamos a la terminal y a la base de datos MySQL, creamos una serie de tablas, ingresamos datos y volvemos al cliente de MySQL Workbench, será necesario refrescar la información nueva: para esto seleccionamos la pestaña "Remoto desde Windows a Linux", que es como denominé la conexión. Seleccionamos nuevamente MIBASE y clic derecho opción "*Refresh All*".

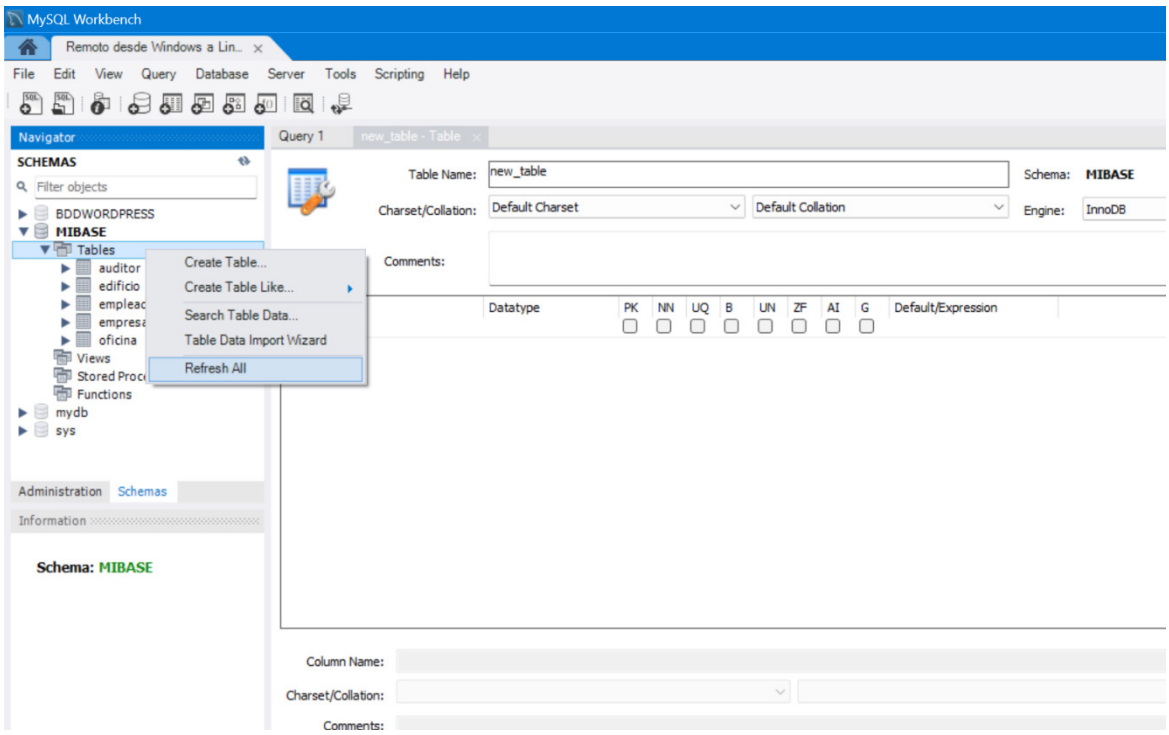



Figura 54. Refrescando la base de datos, aparecerán las tablas si las he creado desde Linux mysql.

MySQL Workbench, por tanto, es una herramienta muy versátil que facilita la gestión de bases de datos MySQL, especialmente útil tanto para desarrolladores como para administradores de bases de datos.

**Ventajas de usar MySQL Workbench**

- Interfaz amigable: facilita el trabajo con bases de datos gracias a su interfaz gráfica intuitiva.
- Herramientas integradas: combina modelado, desarrollo y administración en una sola aplicación.
- Eficiencia: acelera tareas comunes como la creación de esquemas, la escritura de consultas y la gestión de usuarios.
- Productividad: mejora la productividad al proporcionar herramientas avanzadas como el autocompletado de SQL y el historial de consultas.

Es ideal para aquellos que prefieren una interfaz gráfica sobre la línea de comandos, sin embargo, necesita de una instalación en una máquina local y el usuario puede requerir tiempo para familiarizarse con todas sus funcionalidades; además, debe contar con una conexión constante a la base de datos para ser totalmente funcional.

Para profundizar más sobre el manejo de esta herramienta, existe una documentación oficial en inglés en (https://dev.mysql.com/doc/workbench/en/)

## Comandos esenciales de MySQL

Los lenguajes de consulta, como SQL (*Structured Query Language*), permiten a los usuarios recuperar y manipular datos de una base de datos de manera precisa y rápida. SQL es un lenguaje estándar utilizado para gestionar y manipular bases de datos relacionales. Fue desarrollado por IBM en la década de 1970 y se ha convertido en el lenguaje más utilizado para interactuar con bases de datos relacionales.

Utilizar una terminal Linux para administrar bases de datos MySQL ofrece varias ventajas. Es una opción ligera y rápida, consumiendo menos recursos del sistema en comparación con herramientas gráficas. La terminal proporciona un acceso potente y flexible, permitiendo la ejecución de *scripts* y la automatización de tareas. Además, está disponible en cualquier sistema con acceso SSH al servidor, eliminando la necesidad de instalaciones adicionales. Los usuarios tienen un control total sobre el entorno y los comandos ejecutados, lo que es ideal para administradores de sistemas y usuarios avanzados que buscan una gestión detallada y precisa de sus bases de datos.

Los comandos esencialmente *básicos en toda base de datos son: crear, eliminar y alterar, es decir,* podemos crear una base de datos, eliminar una base de datos, crear una tabla, alterar una tabla y eliminar una tabla. A continuación, se presenta una breve descripción de estos comandos.

- CREATE DATABASE: crea una nueva base de datos vacía.
- DROP DATABASE: elimina completamente una base de datos existente.
- CREATE TABLE: crea una nueva tabla, es donde realmente se almacena la información.
- ALTER TABLE: modifica una tabla ya existente.
- DROP TABLE: elimina por completo una tabla existente.

Por ejemplo, podemos crear una base de datos llamada MIBASE, para esto podemos ejecutar desde la terminal Linux la siguiente instrucción:

Este comando crea una base de datos llamada MIBASE:

```
mysql> CREATE database MIBASE ↵ Enter
```

El siguiente comando permite crear una tabla llamada **empleado** y ejecutamos la siguiente instrucción:

```
mysql> CREATE TABLE empleado ( id_empleado int NOT NULL, nombre varchar(30) NOT NULL, apellido varchar(30), estado varchar(30), PRIMARY KEY (id_empleado));
```

Luego de crear la base de datos, si queremos ejecutar alguna otra instrucción o al momento de reingresar a la línea de comandos de mysql, es necesario seleccionar la base de datos con el comando **“use”**, para luego poder manipular los datos.

```
mysql>use MIBASE;
```

Este comando elimina una base de datos llamada MIBASE:

```
mysql>DROP database MIBASE ↵ Enter
```

Si utilizamos un editor gráfico se puede organizar el *script,* como se indica a continuación:

```
CREATE TABLE empleado (
  id_empleado int NOT NULL,
  nombre varchar(30) NOT NULL,
  apellido varchar(30),
  estado varchar(30),
  PRIMARY KEY (id_empleado)
);
```

A continuación, veamos algunos clientes que también pueden ayudarnos en la gestión de bases de datos. El primero es phpMyAdmin es una de las herramientas más populares para gestionar bases de dato MySQL en línea. Adminer es también otra alternativa minimalista y rápida para gestionar bases de datos MySQL y otros sistemas como PostgreSQL, SQLite y Oracle.

HeidiSQL, aunque es un cliente de escritorio, también puede utilizarse para gestionar bases de datos como MaríaDB, MySQL, Microsoft SQL Server, PostgreSQL, SQLite, Interbase/Firebird, es de software libre y código abierto.

TOAD es otra aplicación informática de desarrollo SQL y gestión de base de datos, considerada una herramienta útil para DBAs (administradores de base de datos). Actualmente está disponible para las siguientes bases de datos: Oracle Database, Microsoft SQL Server, IBM DB2, y MySQL.

Otro editor en línea que en general es muy útil a la hora de crear *scripts*, ese puede encontrar en https://sqliteonline.com.

*Sí* es verdad que SQLite es diferente a MySQL ya que este último tiene un conjunto más amplio y específico de tipos de datos como INT, VARCHAR, TEXT, DATE, DATETIME, BLOB, entre otros, pero puede aportarnos en caso de necesitar un *script básico*. Por ejemplo, vemos también la base de datos MariaDB que es altamente compatible con MySQL, ya que MariaDB se diseñó inicialmente como un reemplazo directo de MySQL. Sin embargo, a lo largo del tiempo, MariaDB ha introducido características adicionales y optimizaciones que no están presentes en MySQL, pero de igual manera al momento de crear *scripts* sencillos puede ser de ayuda el uso de esta herramienta.

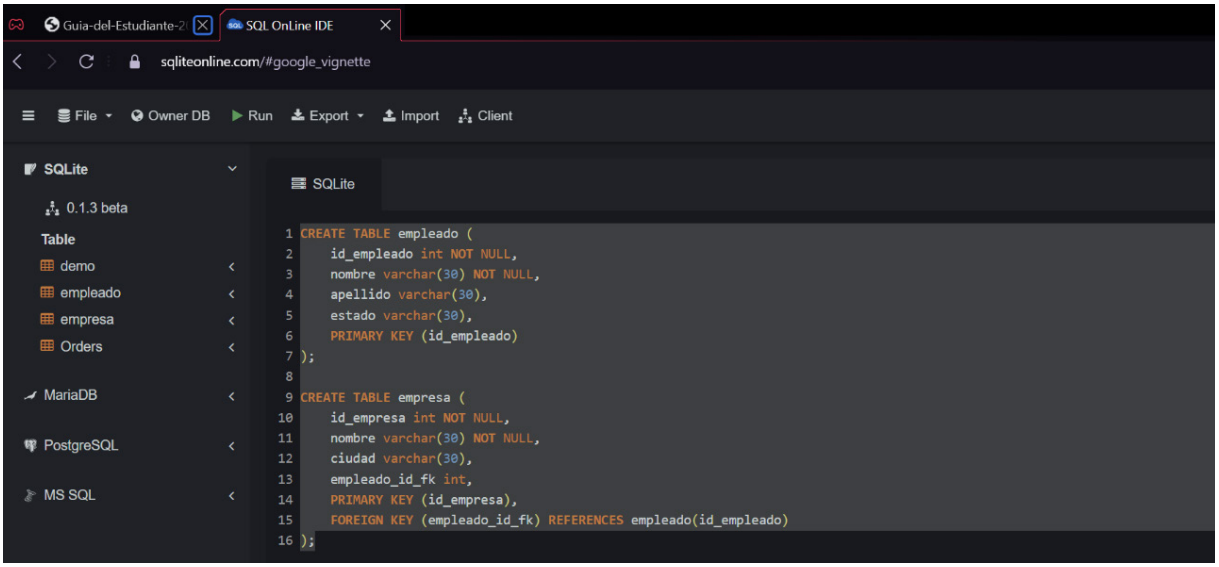



Figura 55. Ejecución del *script* para crear la tabla empleado, desde sqliteonline.com

A continuación, se presenta una breve descripción de estas líneas de código SQL.

- CREATE TABLE empleado: indica el comienzo de la creación de una nueva tabla llamada empleado.
- id_empleado int NOT NULL: aquí se define una columna llamada id_empleado de tipo entero (int) que no puede contener valores nulos (NOT NULL). Esta columna servirá como identificador único para cada empleado.
- nombre varchar(30) NOT NULL: se define una columna llamada nombre de tipo varchar con una longitud máxima de 30 caracteres. Esta columna tampoco puede contener valores nulos (NOT NULL), lo que significa que debe tener un valor para cada registro.
- apellido varchar(30): esta línea define una columna llamada apellido de tipo varchar con una longitud máxima de 30 caracteres. A diferencia de las columnas anteriores, esta puede contener valores nulos, lo que significa que es opcional proporcionar un apellido para un registro.
- estado varchar(30): se define una columna llamada estado de tipo varchar con una longitud máxima

de 30 caracteres. Al igual que la columna apellido, esta también puede contener valores nulos.

- PRIMARY KEY (id_empleado): esta línea define que la columna id_empleado será la clave primaria de la tabla. Una clave primaria es un campo (o combinación de campos) que identifica de manera única cada registro en la tabla. La restricción de clave primaria también impone que los valores en esta columna deben ser únicos y no pueden ser nulos.

A continuación, observamos la creación de otra tabla llamada **empresa,** que estará relacionada precisamente con la tabla **empleado** en esta versión, **empleado_id_fk** es la clave foránea en la tabla empresa que hace referencia a la columna **id** en la tabla **empleado**.

```
CREATE TABLE empresa (
  id_empresa int NOT NULL,
  nombre varchar(30) NOT NULL,
  ciudad varchar(30),
  empleado_id_fk int,
  PRIMARY KEY (id_empresa),
  FOREIGN KEY (empleado_id_fk) REFERENCES empleado(id_empleado)
);
```

Entre los comandos empleados para la manipulación de los datos tenemos SELECT, INSERT, UPDATE, DELETE, etc, que detallamos brevemente a continuación.

### Manipulando los datos

SELECT se utiliza cuando quiere leer o seleccionar sus datos.

INSERT se utiliza cuando quiere insertar nuevos datos.

UPDATE se utiliza cuando quiere actualizar datos existentes.

DELETE se utiliza cuando quiere eliminar datos existentes.

REPLACE se utiliza cuando quiere añadir o cambiar (o reemplazar) datos nuevos o ya existentes.

TRUNCATE se utiliza cuando quiere vaciar todos los datos de la plantilla.

### Desventajas de utilizar MySQL desde una terminal Linux

A pesar de sus ventajas, el uso de una terminal Linux también tiene algunas desventajas. Requiere un conocimiento avanzado de la sintaxis de MySQL y comandos de Linux, lo cual puede ser intimidante para usuarios no técnicos y presenta una curva de aprendizaje significativa. La falta de visualización gráfica puede dificultar la interpretación de datos y el diseño de esquemas complejos. Además, la gestión de usuarios, permisos y monitoreo de rendimiento debe realizarse de manera manual, sin las facilidades intuitivas que ofrecen las herramientas gráficas. Finalmente, no incluye herramientas integradas para *backup* y recuperación visual, lo que puede complicar estas tareas para los usuarios menos experimentados.

```
# Script SQL para la creación de la tabla empleado y empresa


CREATE TABLE IF NOT EXISTS `MIBASE`.`empleado` (
  `id` INT NOT NULL,
  `nombre` VARCHAR(50) NULL DEFAULT NULL,
  `apellido` VARCHAR(50) NULL DEFAULT NULL,
  `estado` VARCHAR(50) NULL DEFAULT NULL,
```

```
  PRIMARY KEY (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci

CREATE TABLE IF NOT EXISTS `MIBASE`.`empresa` (
  `idempresa` INT NOT NULL,
  `nombre` VARCHAR(45) NULL DEFAULT NULL,
  `ciudad` VARCHAR(45) NULL DEFAULT NULL,
  `idempleado_fk` INT NOT NULL,
  PRIMARY KEY (`idempresa`),
  INDEX `fk_empresa_empleado_idx` (`idempleado_fk` ASC) VISIBLE,
  CONSTRAINT `fk_empresa_empleado`
    FOREIGN KEY (`idempleado_fk`)
    REFERENCES `MIBASE`.`empleado` (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci
```

# CAPÍTULO VII INSTALACIÓN Y CONFIGURACIÓN DE APACHE WEB SERVER

## Contenidos del capítulo VII

- Introducción
- Funciones principales de Apache Web Server
- Instalar y administrar el proceso de Apache
- Comprobando el servidor web
- Principales características y usos de los números de puerto
- Configurar *hosts* virtuales
- Características y usos de las interfaces virtuales
- Instalación de php y phpMyAdmin
- Nociones básicas de seguridad web (HTTPS, certificados, permisos)

## Introducción

Apache Web Server ( https://httpd.apache.org ) es un servidor web de código abierto para plataformas Linux, Windows, MacOS, entre otras, que permite a los usuarios alojar y servir contenido web, principalmente para páginas php, aunque también puede soportar otros lenguajes de programación. El servidor Apache es desarrollado y mantenido por una comunidad de usuarios bajo la supervisión de la Apache Software Foundation dentro del proyecto HTTP Server (httpd). La licencia Apache (Apache License o Apache Software License para versiones anteriores a 2.0) es una licencia de *software* libre permisiva creada por la Apache Software Foundation (ASF).

En el mundo Linux también existen otros servidores web como **Nginx,** conocido por su capacidad para manejar muchas conexiones concurrentes con un uso bajo de recursos, muy popular para servir contenido estático y como *proxy* inverso para aplicaciones web. También existe **Apache Tomcat,** un servidor web y contenedor de *servlets* que ejecuta aplicaciones Java y es ampliamente utilizado para aplicaciones basadas en Java EE (Enterprise Edition). **Lighttpd** es otro servidor web ligero y eficiente diseñado para ser rápido y consumir pocos recursos; es adecuado para servidores que manejan muchas conexiones simultáneas y es popular en entornos con recursos limitados. **Cherokee** un servidor web de alto rendimiento y fácil de configurar, ofrece una interfaz de administración web que facilita su configuración y mantenimiento. En realidad, hay muchos más servidores web con diferentes características y funcionalidades, por mencionar algunos ejemplos,pero se ha elegido **Apache web Server** por ser uno de los servidores web más populares del mundo debido a su flexibilidad, robustez y capacidad para ser ampliado mediante módulos adicionales que aportan mucha de la funcionalidad que podría considerarse básica para un servidor web.

## Funciones principales de Apache Web Server

**Servir contenido web:**

Apache recibe las solicitudes de los navegadores web de los usuarios y responde enviando el contenido web solicitado, como páginas HTML, imágenes, videos y otros recursos.

**Alojamiento de múltiples stios web:**

Apache permite alojar múltiples sitios web en un solo servidor, utilizando Virtual Hosts. Esto significa que se pueden servir diferentes sitios web desde una única máquina física, cada uno con su propio nombre de dominio.

**Gestión de aplicaciones web:**

Apache puede ejecutar aplicaciones web dinámicas desarrolladas en diferentes lenguajes de programación como PHP, Python, Perl y Ruby. Utiliza módulos específicos (como mod_php, mod_python, etc.) para procesar el código del lado del servidor y generar contenido dinámico.

**Seguridad:**

Apache proporciona múltiples capas de seguridad, como autenticación básica y avanzada, encriptación mediante SSL/TLS, y módulos de seguridad como mod_security. Esto ayuda a proteger el servidor y los datos transmitidos.

**Reescritura y redirección de URL:**

Con el módulo mod_rewrite, Apache permite reescribir URL entrantes en tiempo real, lo que es útil para SEO, manejo de URL amigables y redirección de tráfico.

**Balanceo de carga y *proxy*:**

Apache puede funcionar como un *proxy* inverso, distribuyendo las solicitudes entrantes a diferentes servidores *backend*. Esto es útil para balanceo de carga y para proteger servidores *backend*.

**Registro y monitoreo:**

Apache tiene capacidades integradas de registro (*logging*) que permiten a los administradores monitorear el tráfico, errores, y otros eventos en el servidor. Los archivos de registro (atewa.log y error.log) son fundamentales para la administración y resolución de problemas.

**Personalización y extensibilidad:**

Apache es altamente configurable y extensible mediante módulos. Se pueden agregar o quitar módulos según sea necesario para personalizar el comportamiento del servidor.

Algunos de los módulos se mencionan a continuación:

**mod_ssl:** comunicaciones eguras *vía TLS*

**Mod_auth_ldap:** permite autentificar usuarios contra un servidor LDAP.

**Mod_proxy_ajp:** conector para enlazar con el servidor Jakarta Tomcat de páginas dinámicas en Java (servlets y JSP).

El servidor de base puede ser extendido con la inclusión de módulos externos entre los cuales se encuentran:

**mod_cband:** control de tráfico y limitador de ancho de banda

**Mod_php:** *páginas dinámicas en PHP*

**Mod_python:** *páginas dinámicas en Python*

**Mod_security:** filtrado a nivel de aplicación, para seguridad

**Compresión de contenido:**

Apache puede comprimir el contenido antes de enviarlo al cliente utilizando módulos como mod_deflate, lo que reduce el tiempo de carga y el ancho de banda utilizado.

**Soporte multiplataforma:**

Apache funciona en una amplia variedad de sistemas operativos, incluyendo Unix, Linux, Windows, y macOS, lo que lo hace muy versátil y adaptable a diferentes entornos.

## Instalar y administrar el proceso de Apache

Para instalar Apache en Linux Ubuntu Server simplemente ejecutamos en la terminal el siguiente comando, saldrá un mensaje de instalación de paquetes nuevos y, si desea continuar, le ponemos que sí o ingresamos la letra S:

```
root@svrlinux:~# apt install apache2 ↵ Enter
```

Enseguida podemos verificar que el servicio de Apache está ejecutándose correctamente. Al ejecutar el comando a continuación, el sistema mostrará información sobre si el servicio de Apache está activo, inactivo, detenido o si ha encontrado algún error:

```
root@svrlinux:~# systemctl status apache2 ↵ Enter
```

Entonces, se desplegará la siguiente información

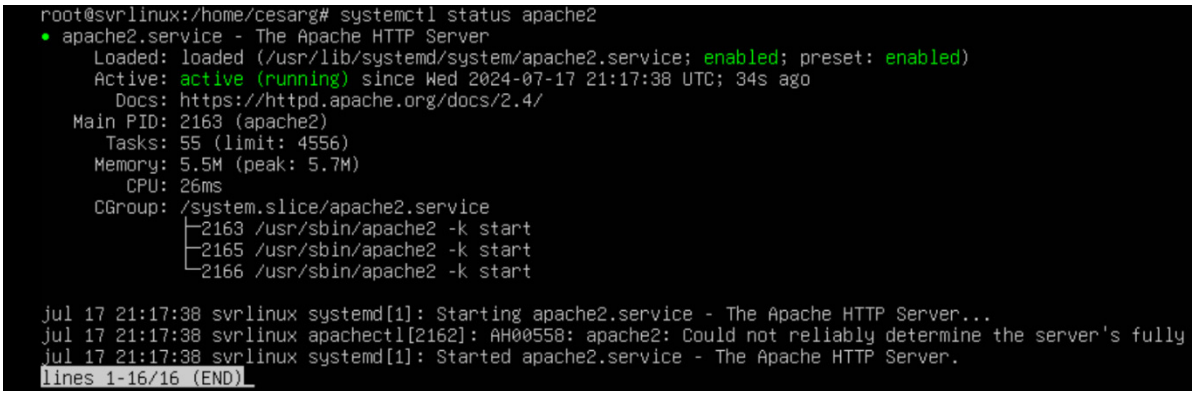



Figura 56. Estado activo (en ejecución) del servicio de Apache

### Ejemplos prácticos

Iniciar el servicio de Apache:

```
root@svrlinux:~# systemctl start apache2 ↵ Enter
```

Detener el servicio de Apache:

```
root@svrlinux:~# systemctl stop apache2 ↵ Enter
```

Reiniciar el servicio de Apache:

```
root@svrlinux:~# systemctl restart apache2 ↵ Enter
```

Recargar la configuración del servicio de Apache sin detenerlo:

root@svrlinux:~# systemctl reload apache2 ↵ Enter

Mostrar el estado del servicio de Apache:

root@svrlinux:~# systemctl status apache2 ↵ Enter

Deshabilita el servicio para que no se inicie automáticamente al arrancar el sistema.:

root@svrlinux:~# systemctl disable apache2 ↵ Enter

Para volver a habilitar el servicio de modo que se cargue en el inicio, escriba lo siguiente:

root@svrlinux:~# systemctl enable apache2 ↵ Enter

## Comprobando el servidor web

Como se había indicado anteriormente, apenas se ha realizado la instalación, el servidor debería estar ejecutándose, simplemente verificamos con el comando siguiente:

root@svrlinux:~# systemctl status apache2 ↵ Enter

Sin embargo, la manera más fácil de comprobarlo es a través de una llamada a la página web de Apache. En este caso, primero debemos conocer la dirección IP del servidor, esto lo podemos obtener con el comando ifconfig desde la terminal de Linux.

```
root@svrlinux:/home/cesarg# ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.10.129  netmask 255.255.255.0  broadcast 192.168.10.255
        inet6 fe80::20c:29ff:fe65:1c4c  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:65:1c:4c  txqueuelen 1000  (Ethernet)
        RX packets 2866  bytes 3450151 (3.4 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1668  bytes 105453 (105.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 266  bytes 20067 (20.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 266  bytes 20067 (20.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figura 57. Obtención de la dirección IP con el comando ifconfig

Ahora, desde un navegador web en Windows, deberemos ingresar la dirección IP en la URL como se indica a continuación: http://192.168.10.129

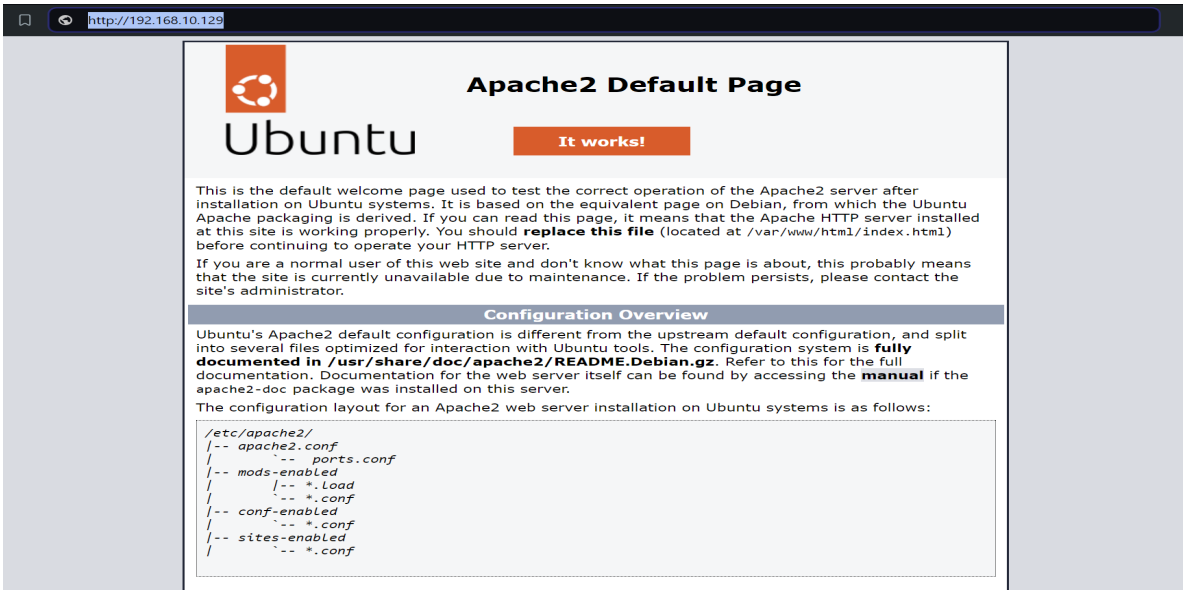



Figura 58. Al acceder al servidor web, se presenta la página por defecto de Apache2.

Si todo va correctamente, debe desplegarse la página inicial de Apache2 como se indica en la figura anterior. Recordemos que Apache Web Server es el servidor de sitios web que está instalado en Linux en modo texto y los clientes vendrían a representarse por cualquier navegador web como Firefox, Google Chrome, Safari, Microsoft Edge, etc. En este caso, los clientes pueden estar ejecutándose en Windows o inclusive Android.

Para ejemplificar más a detalle lo que se está explicando, a continuación, se presenta un diagrama de conexión de la red LAN cliente-servidor.

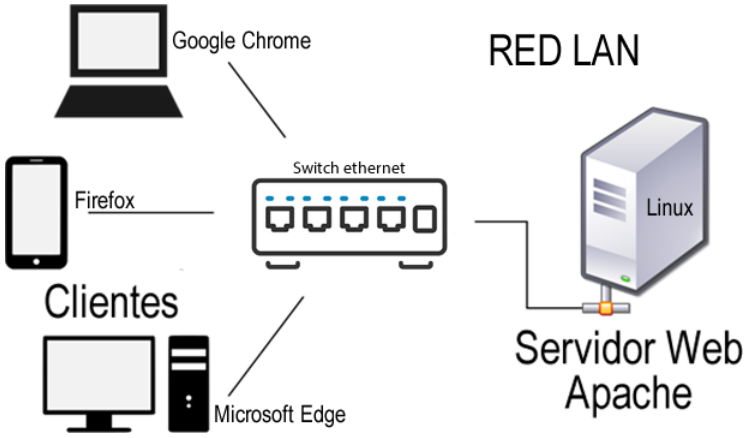



Figura 59. Esquema de red cliente servidor en una red LAN

Lógicamente, para poder acceder al servidor web los clientes deben tener conectividad hacia el servidor. Esto es muy fácil de comprobar, simplemente ejecutando el comando ping seguido de la dirección IP del servidor o del nombre de dominio, si tuviésemos configurado un DNS. El propósito principal de los nombres de dominio en Internet y del sistema de nombres de dominio es traducir las direcciones IP de cada activo en la red, a términos memorizables y fáciles de encontrar, por ejemplo, cuando ejecutamos ping google.com al servidor web de Google, obtendremos respuesta en realidad de la dirección IP 192.178.50.46. El DNS (*Domain Name System*) es un protocolo de Internet que tiene como función la resolución de nombres de dominio, traduciendo estos mismos en IPs. En este apartado no configuraremos el DNS. El comando ping en cambio es una utilidad de diagnóstico en redes de ordenadores que nos permite comprobar la conectividad y latencia del anfitrión local con otro equipo remoto de una red a nivel IP.

A continuación, se presenta la ventana del símbolo del sistema de Windows desde la cual se realiza una prueba de conectividad contra el servidor de Google y seguidamente una prueba de conectividad contra nuestro servidor web en Linux Ubuntu. En ambos casos puede observarse una respuesta desde sus respectivas direcciones IP.

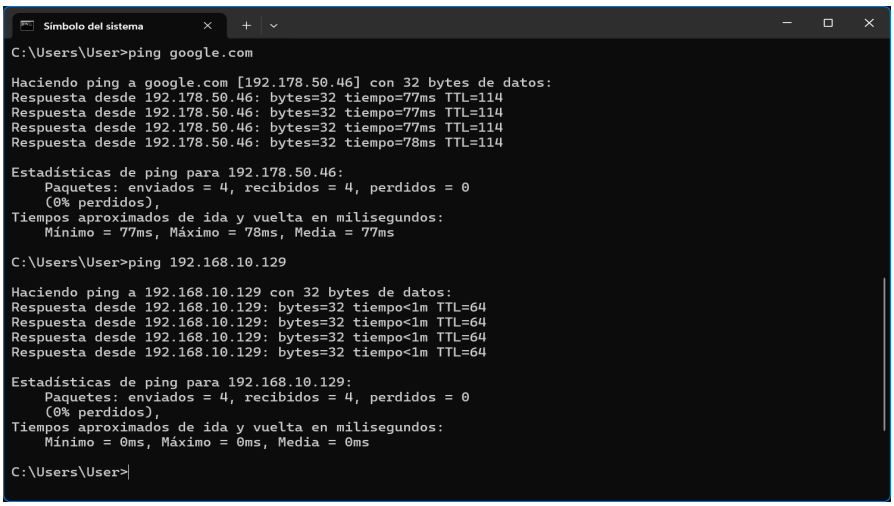



Figura 60. Respuesta de los servidores web a través del comando ping desde Windows 11

En el ejemplo anterior, el servidor Linux tiene dirección IP: 192.168.10.129 y el cliente en Windows tiene dirección IP: 192.168.10.1, estas son direcciones IP privadas dentro de la red local o LAN, que está representado en el esquema de red. Mientras que la dirección IP: 192.178.50.46 de Google es una dirección IP pública, es decir, accesible desde Internet. El diagrama en este caso sería como en la siguiente figura:

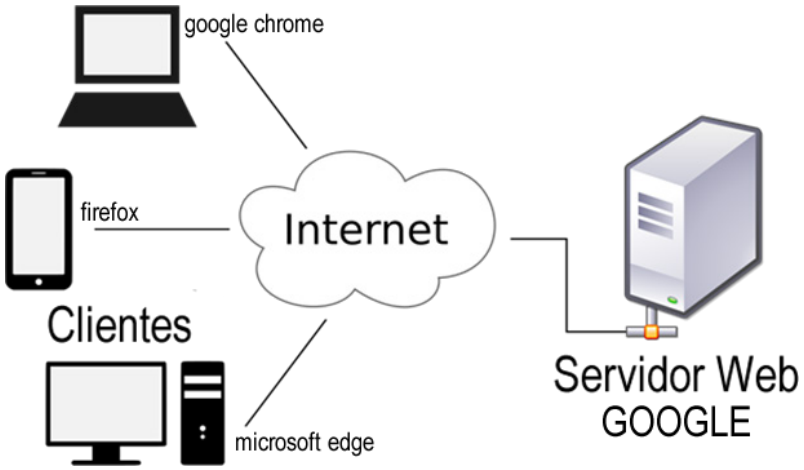



Figura 61. Esquema de red cliente servidor en la red Internet

En lo que se ha venido mencionando, si en la URL del navegador ingresamos http://192.178.50.46, en realidad estamos accediendo al servidor web de Google, es decir, al buscador que ampliamente utilizamos teniendo una simple conexión a Internet.

Muchos de los servidores web, no necesariamente Apache, tienen por defecto abierto el número de puerto 80. El puerto 80 se utiliza para el tráfico HTTP o se refiere al servicio web. Cuando un cliente especifica el puerto 80 en el puerto de destino, el servidor que recibe el mensaje sabe que se está solicitando un servicio web. El puerto 80 es por defecto para especificar el protocolo de transferencia de hipertexto (HTTP), es decir, es el protocolo utilizado para transmitir y recibir páginas web en Internet. Cuando se accede a un sitio web mediante un navegador, la comunicación entre el navegador y el servidor web suele producirse a través del puerto 80, esto puede comprobarse si en la dirección URL ingresamos por ejemplo: http://192.178.50.46:80 o casi en cualquier dirección puedes al final agregar “dos puntos seguido del número de puerto 80” y observará el mismo resultado. Por ejemplo, puede ingresar la dirección http://www.facebook.com:80.

Actualmente, la mayoría de sitios web utilizan el puerto 443, que es para especificar el protocolo de transferencia de hipertexto seguro (HTTPS); podemos comprobarlo ingresando en la URL https://www.google.com:443. Esto quiere decir que el navegador y el servidor establecen una conexión segura y cifrada antes de transferir datos. HTTPS utiliza el protocolo SSL/TLS para encriptar las comunicaciones, de modo que los atacantes no puedan robar los datos. SSL/TLS también revalida que el servidor de un sitio web es quien dice ser evitando las suplantaciones.

Por seguridad y para evitar ataques a nuestros sitios web, es deseable únicamente habilitar el puerto 443 y deshabilitar el puerto 80, comentando la directiva #Listen 80 que se encuentra en el archivo ports.conf, con esto estaremos deshabilitando el protocolo HTTP y ya no podremos ingresar con http://192.178.50.46 sino únicamente con HTTPS https://192.178.50.46.

En la configuración de Apache podemos cambiar el número de puerto por defecto, por ejemplo, si quisiéramos tener otro servidor web dentro de nuestra misma red LAN, podríamos optar por manejar el puerto 8080. En Linux Ubuntu Server esta configuración se realiza en el archivo de configuración de Apache que se encuentra dentro del directorio /etc/apache2/ports.conf; aproximadamente en la línea 5 del archivo, se encuentra la directiva Listen 80, simplemente cambiamos el número por 8080.

Seguidamente se encuentra un bloque de código <IfModule ssl_module> que contiene directivas que se aplicarán solo si el módulo ssl_module (que proporciona soporte SSL/TLS en Apache) está cargado. La directiva Listen 443 indica a Apache que escuche en el puerto 443, que es el puerto estándar para HTTPS, y a continuación reiniciamos el servicio de Apache para que los cambios surtan efecto.

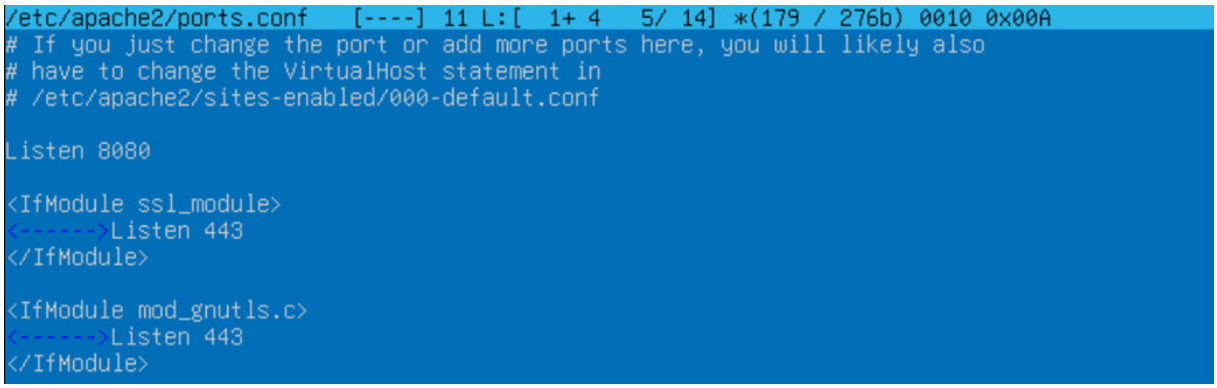



Figura 62. Cambio del número de puerto de 80 a 8080 en el archivo de configuración ports.conf

Reiniciamos el servicio de Apache para que los cambios surtan efecto:

```
root@svrlinux:~# systemctl restart apache2 ↵ Enter
```

Ahora podemos comprobar el cambio ingresando a la URL del navegador http://192.168.10.129:8080 o http://tu-dominio.com:8080 seguido de dos puntos y el número de puerto 8080. Debemos obtener la misma página de salida inicial de Apache2. Otra forma de comprobar que el número de puerto ejecutándose es el 8080 es a través del comando nmap.

```
root@svrlinux:~# nmap localhost
```

Si ejecutamos este comando se puede observar el puerto 8080/tcp como abierto y el nombre del servicio como http-proxy, en realidad podemos cambiar al número de puerto que quisiéramos siempre y cuando no exista un conflicto con otros servicios que estén ocupando ese número de puerto. Sin embargo, no debemos abusar de estos números, ya que podría ocasionar un funcionamiento no adecuado, por cuanto existe una organización llamada IANA (Internet Assigned Numbers Authority) que se encarga de la administración de los recursos clave de Internet. Sus principales responsabilidades incluyen: asignación de números de protocolo de Internet (IP), asignación de números de puerto que se utilizan para identificar servicios específicos en los dispositivos de red, gestión de espacios de nombres de DNS, etc.

La IANA es actualmente operada por la ICANN (Internet Corporation for Assigned Names and Numbers), que es una organización sin fines de lucro, responsable de la coordinación global del sistema de identificadores únicos de Internet. La IANA asegura que los identificadores únicos de Internet (como direcciones IP, números de puerto y nombres de dominio) sean consistentes y universales en todo el mundo, permitiendo así la interoperabilidad de los sistemas y aplicaciones de red.

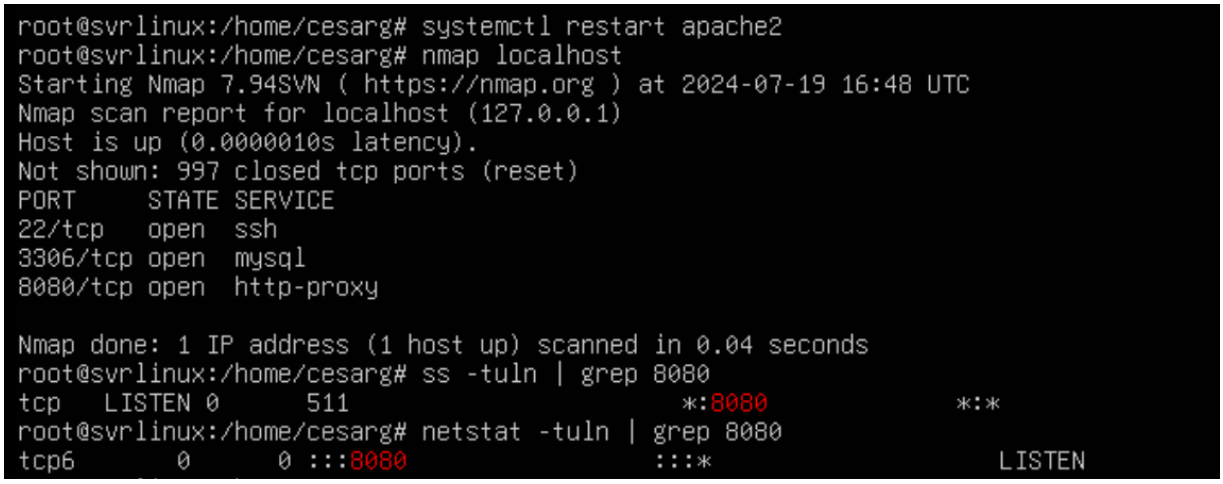



Figura 63. Comprobación del estado del servicio y puerto en escucha

También podemos verificar el funcionamiento del servicio web de Apache con los siguientes comandos. El comando ss -tuln | grep 8080 se utiliza para mostrar información sobre las conexiones de red y puertos en uso, específicamente filtrando para mostrar solo las líneas que contienen "8080". El comando netstat -tuln | grep 8080 es similar al comando ss -tuln | grep 8080, que se utiliza para mostrar información sobre las conexiones de red y puertos en uso, específicamente filtrando para mostrar solo las líneas que contienen "8080". Aunque netstat es una herramienta más antigua, sigue siendo útil y funcional en muchos sistemas.

root@svrlinux:~# ss -tuln | grep 8080

root@svrlinux:~# netstat -tuln | grep 8080

## Principales características y usos de los números de puerto

### Identificación de servicios:

Los números de puerto permiten identificar qué servicio o aplicación debe manejar un paquete de datos entrante. Por ejemplo, el puerto 80 es el puerto predeterminado para HTTP (navegación web) y el puerto 443 es el puerto predeterminado para HTTPS (navegación web segura).

### División en rango de puertos:

Puertos bien conocidos (0-1023): reservados para servicios y aplicaciones más comunes y estandarizados, como HTTP (80), HTTPS (443), FTP (21), y SSH (22).

Puertos registrados (1024-49151): asignados a aplicaciones y servicios menos comunes que han sido registrados por la IANA.

Puertos dinámicos o privados (49152-65535): usados temporalmente por aplicaciones y servicios de manera dinámica.

### Funcionamiento en protocolo TCP/UDP:

TCP (*Transmission Control Protocol*): utiliza puertos para establecer conexiones confiables y orientadas a la conexión. Ejemplos: HTTP (80), HTTPS (443), SMTP (25), POP3 (110), IMAP (143).

UDP (*User Datagram Protocol*): utiliza puertos para conexiones rápidas y no orientadas a la conexión. Ejemplos: DNS (53), TFTP (69).

**Configuración de números de puerto**

La configuración de números de puerto se realiza en los archivos de configuración de las aplicaciones o servicios específicos. Como lo mencionamos anteriormente, en un servidor Apache se puede cambiar el puerto predeterminado (80) a otro puerto (como 8081) editando el archivo de configuración (httpd.conf o apache2.conf) y ajustando la directiva *Listen*:

**Seguridad y números de puerto**

*Firewalls*: utilizan números de puerto para permitir o bloquear el tráfico hacia servicios específicos.

Puertos cerrados y abiertos: Un puerto abierto puede ser una puerta de entrada para ataques si no se gestiona adecuadamente. Por lo tanto, es una práctica de seguridad común cerrar puertos innecesarios.

## Configurar *hosts* virtuales

Los *hosts* virtuales son una tecnología utilizada en servidores web para alojar múltiples sitios web en una sola máquina física o utilizando una única dirección IP, lógicamente creando varias interfaces de red virtuales o alias IP. Esta técnica permite que un único servidor web maneje varias webs de manera eficiente, asignando recursos y configuraciones específicas para cada una de ellas.

**Tipos de *host* virtuales**

*Host* **virtual basado en nombre:**

Utiliza el nombre del dominio para distinguir entre diferentes sitios web en el mismo servidor.

Ejemplo: www.ejemplo1.com y www.ejemplo2.com pueden estar en el mismo servidor, pero se distinguen por su nombre de dominio.

*Host* **virtual basado en IP:**

Utiliza diferentes direcciones IP para distintos sitios web en el mismo servidor.

*Ejemplo*: Un sitio web puede estar en la dirección IP 192.168.1.1 y el otro en la dirección IP 192.168.1.2. En este caso, tendríamos que configurar dos interfaces de red virtuales, en Linux. Las interfaces virtuales suelen nombrarse añadiendo un sufijo a la interfaz física, como eth0:1, eth0:2, etc.

*Host* **virtual basado en puerto:**

Utiliza diferentes puertos para servir distintos sitios web.

Ejemplo: un sitio web puede estar en www.ejemplo.com:80 y otro en www.ejemplo.com:8080.

Antes de realizar la configuración de los *hosts* **virtuales** primero debemos conocer qué interfaces de red tenemos disponibles en Linux, entonces, puede utilizar el comando #ifconfig o el comando #ip a, esto desplegará los nombres de las interfaces y la dirección IP de la máquina. Si la tarjeta de red ethernet es reconocida como enp3s0, normalmente este tipo de interfaz hace referencia a eth0, eth1, eth2, etc. Si la interfaz de red inalámbrica es reconocida como wlp1s0, regularmente se refiere a wlan0, wlan1, etc. Si tenemos instalado Linux Ubuntu mediante un *software* de virtualización como Vmware, la interfaz de red puede estar como ens33, **"ens"** indica que es una interfaz de ethernet **"s33"** muestra que es una interfaz específica asignada por el sistema basado en un número de serie o *slot*.

Si ejecutamos el comando ifconfig, podemos ver las interfaces **"ens33"** y **"lo"** es la interfaz de *loopback* (o bucle invertido). Esta interfaz es una parte esencial de la red local del sistema y se utiliza principalmente para el autodiagnóstico y la comunicación interna del sistema operativo. La interfaz utiliza la dirección IP 127.0.0.1, que es la dirección de *loopback* estándar en Ipv4. Esta dirección es también conocida como "*localhost*". Para Ipv6, la dirección de *loopback* es ::1.

Si quiere probar un servidor web localmente sin una conexión de red externa, puede configurar el servidor para que escuche en 127.0.0.1. Luego, puede acceder al servidor utilizando http://127.0.0.1 o http://localhost.

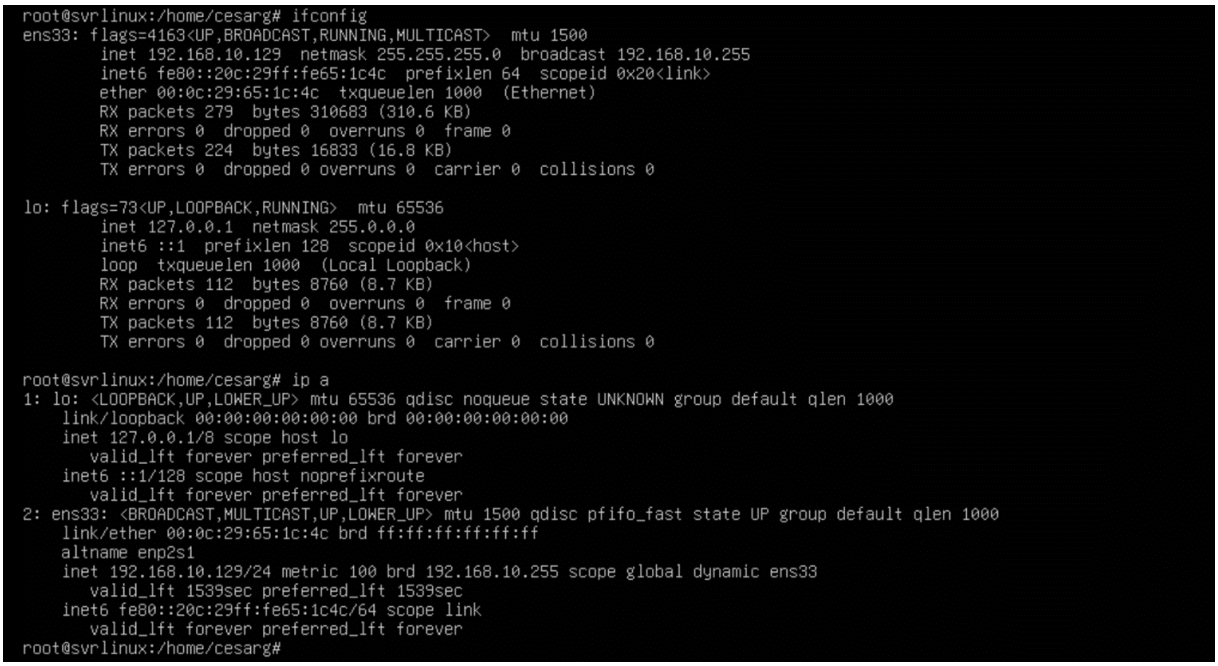



Figura 64. Salida del comando ifconfig, y del comando ip a

El archivo 50-cloud-init.yaml es un archivo de configuración utilizado por cloud-init en sistemas operativos basados en Linux, como Ubuntu Server. Cloud-init es un conjunto de herramientas de inicialización en la nube que se utiliza para personalizar y configurar instancias de máquinas virtuales en nubes públicas y privadas durante su primer arranque. Cambiar la configuración de red manualmente implica editar el archivo de configuración **50-cloud-init.yaml,** que debe estar dentro del siguiente directorio /etc/netplan/. Desde la versión 17.10, Ubuntu utiliza la herramienta **netplan** para administrar la configuración de conectividad. Esta utilidad se basa en reglas de sinopsis YAML. Todas las configuraciones de la red se almacenan en archivos *.yaml. Si no se encuentran archivos en el directorio, tal vez pueda reconstruir  la configuración predeterminada del sistema con el comando:

```
root@svrlinux:~# netplan generate
```

Una vez localizado el archivo, para el ejemplo 50-cloud-init.yaml, lo podemos visualizar mediante el editor mc.

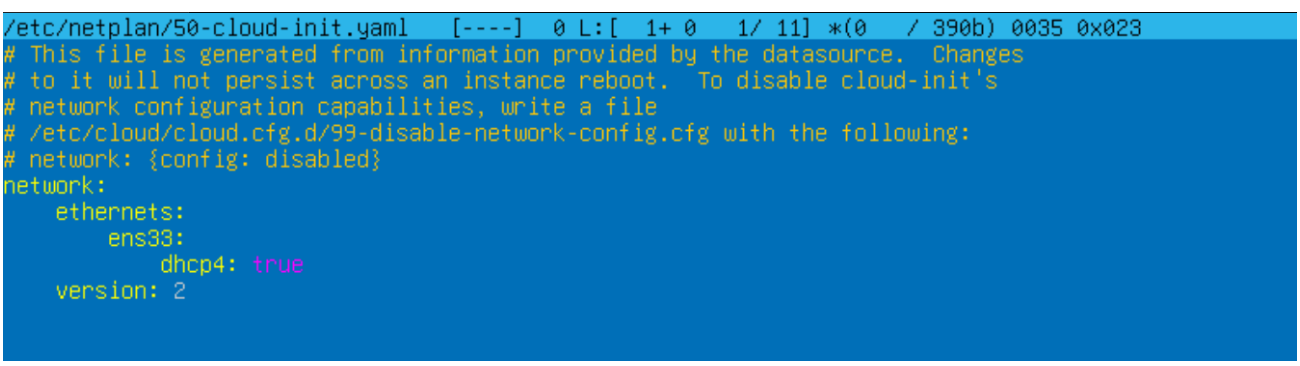



Figura 65. Archivo 50-cloud-init.yaml, configuración por defecto de la dirección IP con dhcp4

A continuación, editamos el archivo **50-cloud-init.yaml** con el editor **nano**. Tenga en cuenta las tabulaciones de cada línea y además evite faltas en el tipeado del texto, caso contrario, le saldrá un error al momento de aplicar la configuración con el comando:

```
root@svrlinux:~#netplan apply ↵ Enter
```

```
 GNU nano 7.2                                                  50-cloud-init.yaml
# This file is generated from information provided by the datasource.  Changes
# to it will not persist across an instance reboot.  To disable cloud-init's
# network configuration capabilities, write a file
# /etc/cloud/cloud.cfg.d/99-disable-network-config.cfg with the following:
# network: {config: disabled}
network:
    version: 2
    ethernets:
        ens33:
            dhcp4: false
            addresses:
              - 192.168.10.129/24
            routes:
            - to: default
              via: 192.168.10.2
            nameservers:
              addresses:
                - 8.8.8.8
                - 8.8.4.4
```

Figura 66. Edición del archivo 50-cloud-init.yaml y asignación manual de la dirección IP

```
root@svrlinux:~#netplan apply try ↵ Enter
```

Este comando probará la nueva configuración sin aplicarla permanentemente. Si la configuración es correcta, tendrá la opción de aplicarla; de lo contrario, se revertirá a la configuración anterior.

Para el ejemplo, debemos configurar una interfaz virtual. Las interfaces virtuales son interfaces de red lógicas que se crean a partir de una interfaz de red física existente en un sistema operativo. Estas interfaces permiten la asignación de múltiples direcciones IP a una sola interfaz física, lo que puede ser útil, como por ejemplo, para la creación de servidores virtuales, para el aislamiento de tráfico de red o para la implementación de redes privadas virtuales (VPN).

Una VPN permite crear una conexión segura a través de una red menos segura, como Internet. Esto se logra encriptando los datos que se envían y se reciben, lo que previene que terceros, como *hackers* o incluso proveedores de servicios de Internet, puedan acceder a esa información.

## Características y usos de las interfaces virtuales

### Multiplicidad de IP en una sola interfaz física:

Alias IP: Permiten asignar varias direcciones IP a una única tarjeta de red física. Esto es útil en entornos donde se necesitan múltiples direcciones IP para diferentes servicios o aplicaciones sin necesidad de *hardware* adicional.

Ejemplo en Linux: En Linux, las interfaces virtuales suelen nombrarse añadiendo un sufijo a la interfaz física, como eth0:1, eth0:2, etc. Esto se configura usando comandos como ifconfig o ip.

**Aislamiento de tráfico de red:**

Seguridad y segmentación: Las interfaces virtuales se pueden utilizar para aislar diferentes tipos de tráfico en un servidor. Por ejemplo, se puede separar el tráfico de administración del tráfico de usuario.

VLAN: Las VLAN (Redes de área local virtuales) utilizan interfaces virtuales para segmentar redes en subredes lógicas, mejorando la seguridad y la eficiencia.

**Configuración de servidores virtuales:**

*Hosts* virtuales en servidores web: Al utilizar múltiples interfaces virtuales, un servidor web como Apache puede alojar varios sitios web diferentes en la misma máquina física, cada uno con su propia dirección IP.

Ejemplo de configuración: utilizando múltiples interfaces virtuales para diferentes servidores virtuales en Apache.

**Redes privadas virtuales (VPN):**

Tunelización de tráfico: Las interfaces virtuales se utilizan en VPN para crear túneles seguros entre diferentes redes, permitiendo el acceso remoto seguro a recursos de red.

**Contenedores y máquinas virtuales:**

Docker y Kubernetes: Las interfaces virtuales se usan ampliamente en tecnologías de contenedores y orquestadores de contenedores para gestionar la red entre contenedores y nodos.

Hypervisores: Los hipervisores como Vmware o KVM utilizan interfaces virtuales para conectar máquinas virtuales a redes físicas y virtuales.

Por otro lado, con el comando #route -n mostramos la tabla de enrutamiento de la red del sistema. La tabla de enrutamiento contiene información sobre las rutas que los paquetes de red deben tomar para llegar a sus destinos. La opción -n especifica que se deben mostrar las direcciones IP en formato numérico, en lugar de resolver los nombres de *host*. Para ver el estado de la configuración DNS, simplemente ejecutamos el siguiente comando:

root@svrlinux:~#resolvectl status ↵ Enter

```
root@svrlinux:/home/cesarg# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.10.2    0.0.0.0         UG    100    0        0 ens33
192.168.10.0    0.0.0.0         255.255.255.0   U     100    0        0 ens33
192.168.10.2    0.0.0.0         255.255.255.255 UH    100    0        0 ens33
root@svrlinux:/home/cesarg#
root@svrlinux:/home/cesarg#
root@svrlinux:/home/cesarg# resolvectl status
Global
         Protocols: -LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported
  resolv.conf mode: stub

Link 2 (ens33)
    Current Scopes: DNS
         Protocols: +DefaultRoute -LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported
Current DNS Server: 192.168.10.2
       DNS Servers: 192.168.10.2
        DNS Domain: localdomain
root@svrlinux:/home/cesarg#
root@svrlinux:/home/cesarg#
```

Figura 67. Tabla de enrutamiento de la red con el comando route -n y presentación del estado de la configuración DNS con el comando #resolvectl *status*

### Descripción de las columnas

Destination: la red de destino o la dirección de destino de los paquetes.

Gateway: ka dirección IP del gateway (puerta de enlace) a través de la cual se deben enviar los paquetes para llegar a la red de destino.

Genmask: la máscara de red que define el rango de direcciones IP en la red de destino.

Flags: las banderas que indican el estado de la ruta. Algunas de las banderas comunes son:

- U: La ruta está activa.
- G: La ruta utiliza un ateway.
- H: La ruta apunta a un *host* específico.
- D: La ruta ha sido generada dinámicamente.
- M: La ruta ha sido modificada por redireccionamiento ICMP.

Metric: el costo de la ruta. Se utiliza para determinar la mejor ruta cuando hay múltiples rutas hacia el mismo destino.

Ref: el número de referencias a esta ruta (no siempre se usa en todos los sistemas).

Use: el número de veces que esta ruta ha sido utilizada.

Iface: la interfaz de red a través de la cual se envían los paquetes para esta ruta.

```
# This file is generated from information provided by the datasource.  Changes
# to it will not persist across an instance reboot.  To disable cloud-init's
# network configuration capabilities, write a file
# /etc/cloud/cloud.cfg.d/99-disable-network-config.cfg with the following:
# network: {config: disabled}
network:
    version: 2
    ethernets:
        ens33:
            dhcp4: false
            addresses:
              - 192.168.10.129/24
              - 192.168.10.130/24
            routes:
            - to: default
              via: 192.168.10.2
            nameservers:
              addresses:
                - 8.8.8.8
                - 8.8.4.4
```

Figura 68. Configuración de una interfaz virtual, al caso agregamos la IP 192.168.10.130

Nuevamente, editamos el archivo **50-cloud-init.yaml** con el editor nano Igual que como lo hicimos anteriormente, tengamos en cuenta las tabulaciones de cada línea y evitemos las faltas en el tipeado del texto. Solo agregamos una nueva dirección IP, para el ejemplo, será la IP que continua de la .129, es decir, 192.168.10.130/24

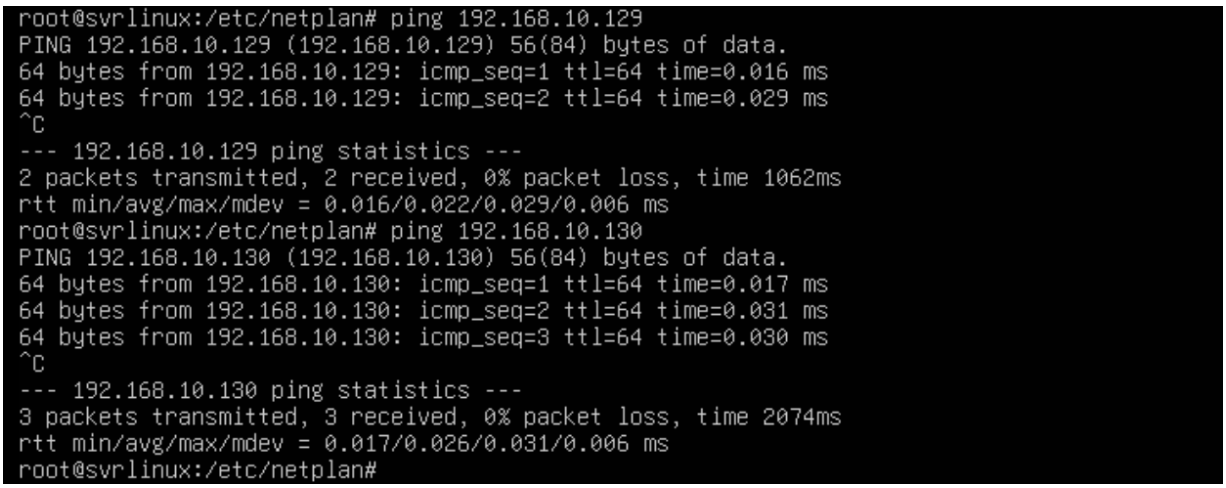



Figura 69. Al hacer ping a las dos direcciones, observamos que existe respuesta.

Una vez comprobada la conectividad con las dos interfaces, es decir, con las dos direcciones IP 192.168.10.129 y 192.168.10.130 respectivamente. Ahora pasaremos a la configuración de los *hosts* virtuales basados en direcciones IP, esto lo realizamos en el archivo de configuración de Apache, dentro del directorio /etc/apache2/ en el cual se encuentran dos carpetas en particular /etc/apache2/sites-available/ y /etc/apache2/sites-enabled/.

En el directorio /sites-available/ se encuentran todos los archivos de configuración de los *hosts* virtuales, cada *host* se corresponde con un archivo, mientras que en el directorio /sites-enabled/ se crearán enlaces simbólicos a los archivos de la carpeta sites-available para los *hosts* que queramos activar en cada momento.

Con la instalación de Apache, se creará por defecto el archivo /etc/apache2/sites-available/**000-default.conf** y su correspondiente enlace simbólico en la carpeta /sites-enabled/ que se encarga de configurar el *host* virtual por defecto al que accedemos tras instalar Apache.

**Cómo crear el *host* virtual**

Normalmente, los sitios webs en Apache se alojan dentro del directorio /var/www/html/, este directorio se creó automáticamente al momento de instalar Apache. Para este ejemplo crearemos dos directorios dentro del directorio /html/ para dos sitios web diferentes. Estando dentro del directorio /html/ ejecutamos el comando mkdir:

```
root@svrlinux:/var/www/html# mkdir misitio
root@svrlinux:/var/www/html# mkdir miwebsite
```

Luego, ingresamos al directorio /etc/apache2/sites-available/ y ejecutamos el comando cp, esto nos permite crear una copia del archivo 000-default.conf en el nuevo archivo misitio.conf y miwebsite.conf respectivamente. Es importante que el archivo al final tenga la extensión .conf

```
root@svrlinux:/etc/apache2/sites-available# cp 000-default.conf misitio.conf
root@svrlinux:/etc/apache2/sites-available# cp 000-default.conf miwebsite.conf
```

```
/etc/apache2/sites-available/misitio.conf    [----]  0 L:[  1+ 0   1/ 30] *(0   /1286b) 0060 0x03C
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to
        # include a line for only one particular virtual host. For example the
        # following line enables the CGI configuration for this host only
        # after it has been globally disabled with "a2disconf".
        #Include conf-available/serve-cgi-bin.conf
</VirtualHost>
```

Figura 70. Contenido del archivo original de configuración para el *host* virtual.

Al editar los archivos nos encontramos con el siguiente contenido por defecto. Vea la siguiente figura.

Los archivos misitio.conf y miwebsite.conf debemos modificarlos, pues recordemos que queremos configurar dos *hosts* virtuales; si quisiéramos un solo *host* virtual, bastaría únicamente con misitio.conf.

Si nos fijamos en el contenido, existe texto que está comentado y no tiene relevancia en la configuración; esto se lo realiza mediante el símbolo de #, por cuanto podemos eliminar estas líneas de texto.

A continuación, el archivo misitio.conf quedaría de la siguiente manera, donde utilizaremos el puerto 80:

```
<VirtualHost *:80>
ServerAdmin webmaster@localhost
DocumentRoot /var/www/html/misitio

ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined

<Directory /var/www/html/misitio>
  Options Indexes FollowSymLinks MultiViews
  AllowOverride All
  Order allow,deny
  allow from all
</Directory>

</VirtualHost>
```

Y el archivo miwebsite.conf quedaría como sigue, utilizando el puerto 8080:

```
<VirtualHost *:8080>
ServerAdmin webmaster@localhost
DocumentRoot /var/www/html/miwebsite

ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined

<Directory /var/www/html/miwebsite>
  Options Indexes FollowSymLinks MultiViews
  AllowOverride All
  Order allow,deny
  allow from all
</Directory>

</VirtualHost>
```

Por defecto, Apache escucha el puerto 80, pero para conseguir que Apache también pueda escuchar el puerto 8080, debemos editar el archivo ports.conf que se encuentra dentro del directorio /etc/apache2/ports.conf y únicamente agregamos la directiva Listen 8080.

```
/etc/apache2/ports.conf    [----]  9 L:[  1+ 4   5/ 15] *(177 / 286b) 0010 0x00A
# If you just change the port or add more ports here, you will likely also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf

Listen 80
Listen 8080

<IfModule ssl_module>
<------>Listen 443
</IfModule>

<IfModule mod_gnutls.c>
<------>Listen 443
</IfModule>
```

Figura 71. Se puede apreciar en la configuración dos puertos que estarán en escucha el 80 y el 8080.

A continuación, habilitamos el nuevo *virtualhost* mediante la creación de un enlace simbólico esto es mediante el comando a2ensite. Al ejecutar este comando se creará automáticamente un enlace simbólico en el directorio /etc/apache2/sites-enable/@misitio.conf y /etc/apache2/sites-enable/@miwebsite.conf respectivamente.

```
root@svrlinux:/etc/apache2/sites-available# a2ensite misitio.conf
root@svrlinux:/etc/apache2/sites-available# a2ensite miwebsite.conf
```

Reiniciamos el servicio de Apache para que los cambios surtan efecto:

```
root@svrlinux:~# systemctl restart apache2 ↵ Enter
```

Si existiera algún error, por lo general es por alguna equivocación en el tipeado del texto, estos errores se pueden visualizar con el comando:

```
root@svrlinux:~# systemctl status apache2 ↵ Enter
```

Tanto en mi *sitio.conf* como en *miwebsite.conf* se tiene la directiva <VirtualHost *:80> y <VirtualHost *:8080>. Si nos fijamos en el símbolo de asterisco esto significa que Apache debería escuchar en todas las direcciones IP configuradas en el servidor para el puerto especificado. En este caso, *:80 significa que Apache escuchará en el puerto 80 en todas las interfaces de red disponibles en el servidor y también en el 8080.

Si en lugar de * especificamos una dirección IP, Apache solo atenderá las solicitudes dirigidas a esa dirección IP en el puerto indicado. Anteriormente, se había configurado dos direcciones, la IP 192.168.10.129 y la IP 192.168.10.130, es decir, ahora, para habilitar dos *hosts* virtuales, simplemente cambiaremos en lugar de asterisco por la dirección IP, quedando como sigue: <VirtualHost 192.168.10.129:80> y <VirtualHost 192.168.10.130:8080>. Recordemos también deshabilitar la configuración por defecto en el archivo /etc/apache2/sites-available/000-default.conf caso contrario se seguirá manteniendo para toda dirección ya que este archivo contiene la configuración con *, por cuanto podemos aplicar el comando siguiente:

```
root@svrlinux:~# a2dissite 000-default.conf ↵ Enter
```

```
root@svrlinux:/etc/apache2/sites-available# ls -l
total 20
-rw-r--r-- 1 root root 1286 mar 18 12:35 000-default.conf
-rw-r--r-- 1 root root 4573 mar 18 12:35 default-ssl.conf
-rw-r--r-- 1 root root  342 jul 23 18:40 misitio.conf
-rw-r--r-- 1 root root  347 jul 23 18:40 miwebsite.conf
root@svrlinux:/etc/apache2/sites-available# a2dissite 000-default.conf
Site 000-default disabled.
To activate the new configuration, you need to run:
  systemctl reload apache2
root@svrlinux:/etc/apache2/sites-available# systemctl reload apache2
```

Figura 72. Deshabilitando el sitio web por defecto, a través del comando a2dissite

Para comprobar esta configuración, por ejemplo, en la URL del navegador web de Firefox en Windows, podemos ingresar http://192.168.10.130:80/miwebsite/ y http://192.168.10.129:8080/misitio/ respectivamente, deberá desplegarse entonces el sitio web correspondiente, según las páginas .html ubicadas dentro del directorio, por ejemplo: /var/www/html/miwebsite/index.html y /var/www/html/misitio/inicio.html. De esta manera, tendremos configurado y disponibles dos sitios web en dos direcciones IP distintas, utilizando el mismo servidor Apache.

## Instalación de php y phpMyAdmin

PHP significa "PHP: *Hypertext Preprocessor*". Es un lenguaje de programación de código abierto que se utiliza principalmente para el desarrollo web del lado del servidor, es decir, es un lenguaje que se ejecuta en el servidor y genera contenido dinámico, lo que permite crear sitios web interactivos.

PHP es compatible con múltiples sistemas operativos, lo que significa, por ejemplo, que puede ejecutarse en Windows, Linux y macOS, esto demuestra una considerable apertura en cuanto a entorno de desarrollo.

PHP también tiene una comunidad muy activa, lo cual promueve la creación de bibliotecas y *frameworks* como Laravel, Symfony y CodeIgniter. Además, el uso de CMS (*Content Management System*) o Sistema de Gestión de Contenidos preparados generalmente para implementar sitios web. Uno de los más populares es Wordpress, que permite crear blogs, webs empresariales, tiendas *online*, etc. desarrollado en PHP y con soporte a bases de datos como MySQL.

Precisamente, PHP se destaca por su facilidad de uso, amplia documentación y su capacidad para integrarse con diversas bases de datos como MySQL y PostgreSQL y servidores web como Apache o Nginx.

Es recomendable utilizar la versión más actual y estable al momento de redactar este libro es la versión 8.3.6 en Linux Ubuntu, sin embargo, a veces pueden existir aplicaciones que funcionen con versiones anteriores de php como la versión 7.x y no sean compatibles con la versión actual, pero debe saber que se pueden instalar múltiples versiones en el mismo servidor, aunque no lo recomiendo; es mejor tener una única versión instalada.

Luego de instalar Apache 2.x podemos a continuación instalar PHP. Primero, nos aseguramosde que nuestro sistema esté actualizado.

```
root@svrlinux:~#apt update && apt upgrade -y ↵ Enter
```

para instalar la versión más reciente de PHP, ingresamos el siguiente comando.

```
root@svrlinux:~#apt install php ↵ Enter
```

para instalar un módulo php para Apache, esto permite que apache pueda interpretar archivos php, ingresamos el siguiente comando.

```
root@svrlinux:~#apt install libapache2-mod-php -y ↵ Enter
```

En este punto, podemos verificar la versión instalada con el comando siguiente:

```
root@svrlinux:~# php -v
PHP 8.3.6 (cli) (built: Dec  2 2024 12:36:18) (NTS)
Copyright (c) The PHP Group
Zend Engine v4.3.6, Copyright (c) Zend Technologies
    with Zend OPcache v8.3.6, Copyright (c), by Zend Technologies
root@svrlinux:~#
```

Si deseáramos instalar una versión específica de php, como PHP8.1, ingresamos el siguiente comando, a veces es común en la misma línea de instalación enviar varios parámetros en lugar de solamente apt install php8.1.

```
root@svrlinux:~#apt install php8.1 libapache2-mod-php8.1 -y
```

A continuación, instalaremos módulos adicionales, esto es necesario para que todo el ecosistema funcione correctamente, por ejemplo, podemos instalar php-xml, php-curl, php-mbstring, php-mysql, etc.

php-xml es una función para manipular XML en PHP, es necesaria para leer, escribir y procesar archivos XML. Muchas aplicaciones CMS como OMP, OJS y WordPress dependen de esta extensión.

php-mbstring permite manejar cadenas de texto con caracteres especiales o codificaciones como UTF-8.

php-curl permite realizar solicitudes HTTP a otros servidores desde PHP. Se usa para integrar API externas, descargar archivos y enviar datos a servicios web.

Ejemplo de código fuente en PHP utilizando curl.

```
$ch = curl_init("https://api.example.com/data"); curl_setopt($ch, CURLOPT_RETURNTRANSFER, true); $response = curl_exec($ch); curl_close($ch); echo $response;
```

php-mysql es una extensión de PHP que permite conectar aplicaciones PHP con bases de datos MySQL o MariaDB. Es necesario para sistemas como WordPress, OJS, OMP y otras plataformas basadas en PHP.

Ejemplo típico de código fuente php para la conexión a una base de datos

```
<?php

$conexion = new mysqli("localhost", "usuario", "contraseña", "basedatos");
if ($conexion->connect_error) {
    die("Error de conexión: " . $conexion->connect_error);
}
$resultado = $conexion->query("SELECT * FROM usuarios");
while ($fila = $resultado->fetch_assoc()) {
    echo "Usuario: " . $fila["nombre"] . "<br>";
}
$conexion->close();

?>
```

Instalación de módulos adicionales, según nuestros requerimientos:

```
root@svrlinux:~# apt install php-mysql -y
root@svrlinux:~# apt install php-xml php-curl php-mbstring -y
```

Por ejemplo, si no instala php e ingresas al sitio web desde un navegador, como se indica en la siguiente imagen, únicamente observará el código fuente como texto. Si no instala los módulos adicionales, no tendrá conectividad con su sitio web hecho en WordPress u otro CMS.

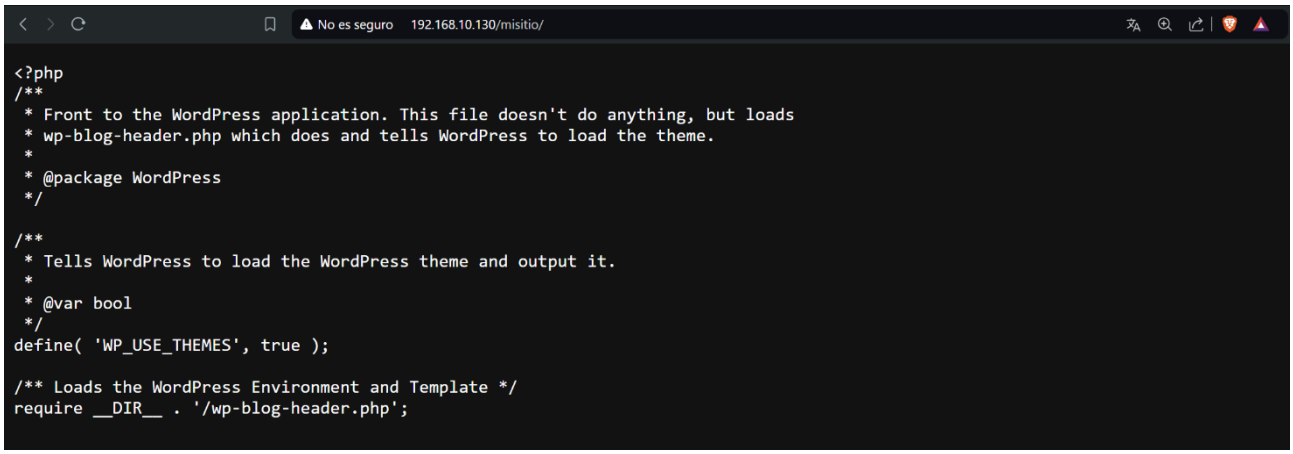



Figura 73. Accediendo a un sitio web, sin php instalado

Podemos comprobar la versión de php y, si el mismo está trabajando correctamente con Apache, podemos agregar un archivo con extensión .php dentro del directorio de publicación en Linux que suele ser en /var/www/html/; a este archivo puede llamarlo info.php, dentro del mismo debe tener el siguiente código:

```
<?php
 phpinfo();
?>
```

Abra un navegador web e ingrese https://su_dominio_o_direccion_IP/info.php, esto debería presentar una página web con toda la información sobre la configuración de PHP en su servidor, como se muestra en la siguiente figura.

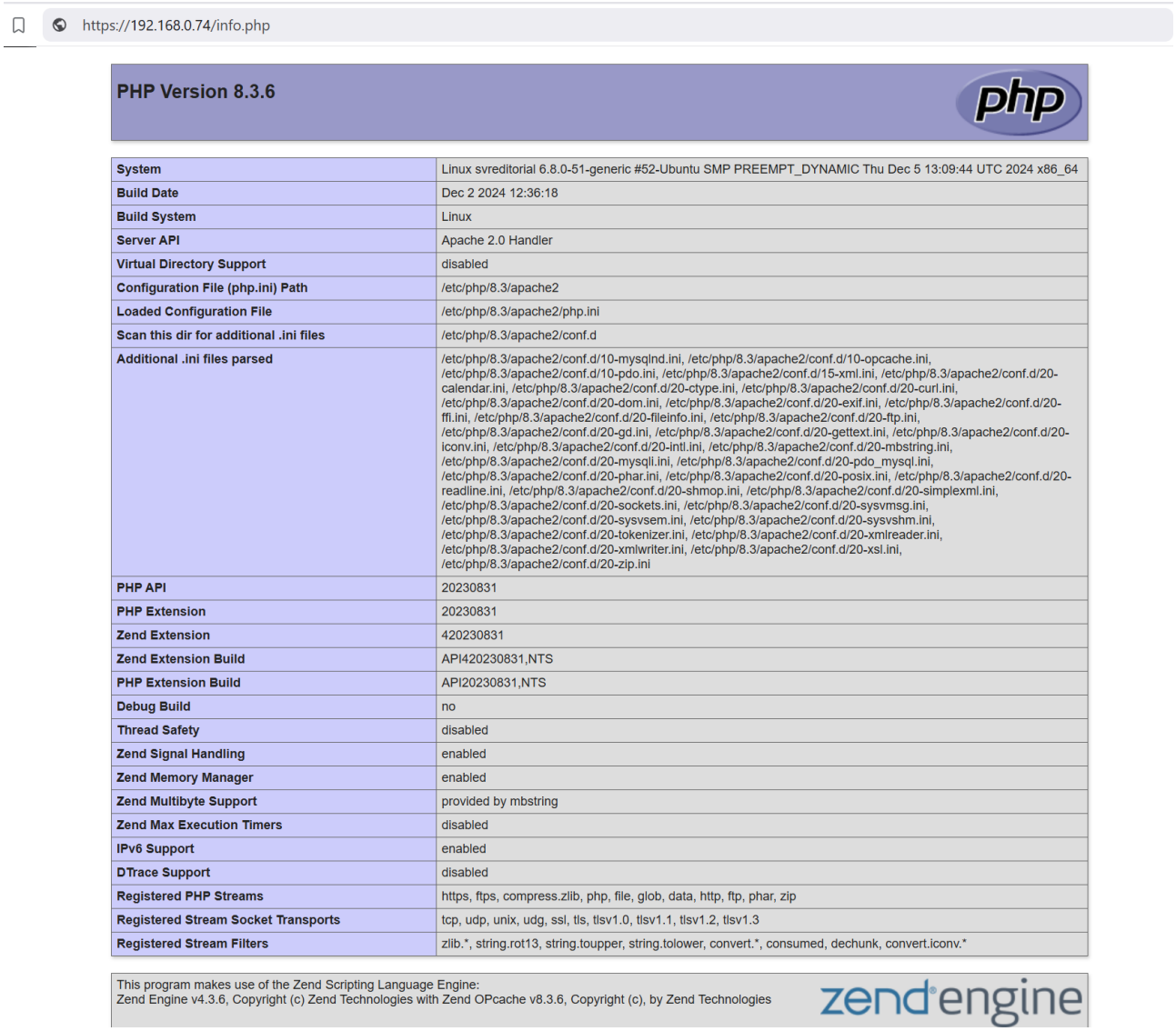



Figura 74. Pantalla informativa de PHP

Para poder instalar PhpMyAdmin en Linux Ubuntu, es necesario tener ya configurado el servidor web. En nuestro caso, tenemos instalado Apache y debemos disponer de php con Mysql. Para la instalación, ingresamos a la terminal el siguiente comando:

```
root@svrlinux:~# apt install phpmyadmin -y
```

phpMyAdmin necesita que ciertas extensiones de PHP estén activas, para esto habilitamos con el siguiente comando y reiniciamos el servidor de Apache para que los cambios surtan efecto.

```
root@svrlinux:~# phpenmod mbstring
root@svrlinux:~# systemctl restart apache2
```

Durante la instalación de phpMyAdmin seleccionamos Apache2 como el servidor web, mediante las teclas de dirección y la barra espaciadora. Cuando en la instalación se pregunte si desea configurar la base de datos para phpMyAdmin con dbconfig-common, seleccionamos 'Sí'. Seguidamente, ingresamos la clave para el usuario de phpmyadmin. Luego verificamos el acceso, abrimos un navegador web e ingresamos en la URL http://direccion-ip o dominio /phpmyadmin, como se indica en la siguiente ventana:




Figura 75. Pantalla inicial de phpMyAdmin

Verificamos el acceso a phpMyAdmin e iniciamos la sesión con las credenciales de MySQL, con usuario *root* y la clave configurada durante la instalación de MySQL. Por motivos de seguridad está configurado por defecto para utilizarse localmente. En caso de necesitar acceder remotamente, editamos el archivo de configuración de Apache.

```
root@svrlinux:~# nano /etc/apache2/conf-available/phpmyadmin.conf
```

Buscamos y comentamos la línea que restringen el acceso, esto es lo siguiente:

```
# Require local

# Allow from all
```

Guardamos y reiniciamos Apache n caso de necesitar permisos más elevados para el usuario phpMyAdmin, pero no es recomendable para servidores que están en producción, entonces, podemos realizar lo siguiente:

```
root@svrlinux:~# mysql -u root -p


mysql>GRANT ALL PRIVILEGES ON *.* TO 'phpmyadmin'@'localhost' IDENTIFIED BY 'su-clave';

mysql>FLUSH PRIVILEGES;
```

Esto sería todo y deberíamos poder ingresar a la pantalla de administración de las bases de datos desde phpMyAdmin.

## Nociones básicas de seguridad web

Una vez que Apache está funcionando, el siguiente paso crítico es "blindar" la comunicación y el acceso a los datos. La seguridad web se apoya en tres pilares fundamentales: cifrado, identidad y control de acceso.

### 1. HTTPS y el cifrado SSL/TLS

El protocolo HTTPS (Puerto 443) es la versión segura de HTTP. Su función principal es cifrar la información que viaja entre el navegador del cliente y el servidor Apache.

SSL/TLS: Son los protocolos criptográficos que realizan el cifrado. SSL es el término antiguo, mientras que TLS (*Transport Layer Security*) es el estándar moderno y seguro.

¿Por qué es importante? Sin HTTPS, cualquier atacante en la misma red (como en un Wi-Fi público) podría leer contraseñas de phpMyAdmin o datos de formularios en texto plano.

Tabla 5. Resumen de puertos de seguridad web

| Puerto | Protocolo | Estado | Recomendación |
|---|---|---|---|
| 80 | HTTP | Inseguro | Usar solo para redireccionar al 443 |
| 443 | HTTPS | **Seguro** | Estándar obligatorio para producción |
| 8080 | HTTP Alt | Inseguro | Comúnmente usado para pruebas o *proxies* |

### 2. Certificados digitales (*Let's Encrypt*)

Para que HTTPS funcione, el servidor necesita un Certificado SSL. Este documento digital valida la identidad del sitio.

Autoridades de certificación (CA): Son entidades que emiten estos certificados.

Let's Encrypt: Es la revolución en seguridad web actual. Es una CA gratuita, automática y abierta que permite instalar certificados válidos en segundos usando una herramienta llamada Certbot.

El **visor de certificados** es una herramienta de interfaz gráfica (presente en navegadores web y sistemas operativos) que permite a un usuario o administrador inspeccionar los detalles técnicos de un certificado digital.

Es, esencialmente, el "documento de identidad" de un sitio web o un servidor. Cuando se accede a un sitio por HTTPS, el visor le permite verificar si la conexión es realmente segura y quién garantiza esa seguridad.




Figura 76. Ejemplo de visor de certificados desde un navegador web

¿Qué información puede encontrar en el visor?

Al abrirlo (generalmente haciendo clic en el candado de la barra del navegador), se pueden observar los siguientes datos críticos:

- **Sujeto (emitido para):** Indica a qué dominio pertenece el certificado (ej. www.pucesi.edu.ec). Es vital para evitar el *phishing*.
- **Entidad emisora (emitido por):** Quién es la autoridad de certificación (CA) que lo validó (ej. Let's Encrypt, DigiCert, GlobalSign).
- **Periodo de validez:** La fecha de emisión y de expiración. Un certificado caducado provoca que el navegador bloquee el acceso al sitio.
- **Huellas digitales (*Thumbprints*):** Códigos únicos (*hashes*) que aseguran que el certificado no ha sido alterado.
- **Jerarquía de certificados:** Muestra la "Cadena de confianza", desde el certificado del sitio hasta el certificado raíz de la entidad emisora.

Utilidad para un administrador de servidores

En el contexto de **Ubuntu Server y Apache**, el visor de certificados es una herramienta de diagnóstico principal para:

1. **Verificar la instalación de SSL:** Tras configurar Apache con SSL, se usa el visor desde Windows para confirmar que el servidor está entregando el certificado correcto.
2. **Detectar certificados "*Self-Signed*" (Autofirmados):** Si se instaló un certificado creado por uno mismo, el visor se lo indicará. El navegador mostrará una alerta porque ninguna autoridad externa ha validado su identidad.
3. **Comprobar algoritmos de cifrado:** Le permite ver si el certificado usa algoritmos modernos (como **SHA-256**) o si está usando tecnología obsoleta y vulnerable.

¿Cómo acceder a él?

- **En navegadores (Chrome/Edge):** Clic en el icono del candado a la izquierda de la URL > "La conexión es segura" > "El certificado es válido".
- **En Windows:** Se puede usar el comando certlm.msc para ver los certificados instalados en el sistema operativo cliente.
- **En Linux (Línea de comandos):** Si no tiene interfaz gráfica, los administradores usan openssl para inspeccionar certificados:

```
root@svrlinux:~# openssl x509 -in certificado.crt -text -noout
```

Para configurar un certificado en Linux (específicamente en **Ubuntu Server** con **Apache**) existen dos caminos principales: el uso de certificados gratuitos y automatizados con **Let's Encrypt** (estándar de la industria) o la creación de un certificado **Autofirmado** (para pruebas locales).

I. Opción profesional: Let's Encrypt (Certbot)

Esta es la forma recomendada para servidores con un dominio real (ej. www.nuestrositio.com). Es gratuito, reconocido por todos los navegadores y se renueva solo.

- **Paso 1: Instalar Certbot**

```
root@svrlinux:~# sudo apt update
root@svrlinux:~# sudo apt install certbot python3-certbot-apache
```

- **Paso 2: Obtener e instalar el certificado.** Certbot leerá su configuración de Apache y se configurará el SSL automáticamente.

```
root@svrlinux:~# sudo certbot --apache
```

- **Paso 3: Verificación.** El comando anterior modificarás sus archivos en /etc/apache2/sites-available/ y creará una versión con terminación -le-ssl.conf.

II. Opción de laboratorio: Certificado autofirmado

Si no tiene un dominio y solo está probando en una red local, puede generar su propia "identidad". El navegador mostrará una advertencia de seguridad, pero el tráfico irá cifrado.

- **Paso 1: Generar la llave y el certificado** Este comando crea una llave privada y un certificado válidos por 365 días:

```
root@svrlinux:~# sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/apache-selfsigned.key -out /etc/ssl/certs/apache-selfsigned.crt
```

*Nota: Al ejecutarlo, le pedirá datos como país, ciudad y nombre. En "Common Name", ingrese la dirección IP de su servidor.*

- **Paso 2: Configurar Apache para usar los archivos.** Debe editar el archivo de host virtual de SSL:

root@svrlinux:~# sudo nano /etc/apache2/sites-available/default-ssl.conf

Busque las siguientes líneas y apunte sus nuevos archivos:

Apache

SSLCertificateFile /etc/ssl/certs/apache-selfsigned.crt

SSLCertificateKeyFile /etc/ssl/private/apache-selfsigned.key

- **Paso 3: Activar los módulos y el sitio**

root@svrlinux:~#sudo a2enmod ssl

root@svrlinux:~#sudo a2ensite default-ssl

root@svrlinux:~#sudo systemctl restart apache2

III. Verificación de la configuración

Para asegurar que todo quedó bien configurado, el administrador debe verificar tres puntos:

1. **Estado del puerto:** Asegúrese de que el puerto 443 esté escuchando: sudo ss -tuln | grep 443.
2. **Firewall:** Abrir el tráfico HTTPS: sudo ufw allow 'Apache Full'.
3. **Logs de error:** Si Apache no reinicia, revisar el log: sudo tail -f /var/log/apache2/error.log.

**3. Gestión de permisos en el servidor web**

Un error común de seguridad es dar demasiados permisos a los archivos web. En Ubuntu, Apache corre bajo el usuario www-data.

Propiedad de archivos: Los archivos en /var/www/html deben pertenecer a www-data.

**Permisos recomendados:** * Directorios: 755 (El dueño lee/escribe/ejecuta, otros solo leen/ejecutan).

**Archivos:** 644 (El dueño lee/escribe, otros solo leen).

**Riesgo:** Nunca debe otorgarse permisos 777 (acceso total a todos), ya que un atacante podría subir un script malicioso y tomar control total del servidor.

**4. Seguridad específica para phpMyAdmin**

Al ser una herramienta de gestión de bases de datos, es un objetivo frecuente de ataques.

**Doble autenticación:** Es recomendable proteger el directorio de phpMyAdmin con un archivo .htaccess y .htpasswd, exigiendo una contraseña adicional antes de llegar a la pantalla de login de MySQL.

**Cambio de alias:** Cambiar la ruta de acceso de /phpmyadmin a algo secreto (ej. /gestion_db_2024) reduce los ataques de bots automáticos.

# CAPÍTULO VIII
# PROGRAMACIÓN APLICADA A LA ADMINISTRACIÓN DE SISTEMAS LINUX

## Contenidos del Capítulo VIII

- Introducción
- Instalación y configuración del entorno de desarrollo
- Fundamentos de programación en Linux
- *Scripting* con Python para la gestión de Linux
- *Scripting* en C y C++ para la gestión de Linux
- Conclusiones

## Introducción

En este capítulo, estableceremos las bases técnicas para la ejecución de código en entornos Linux. El objetivo no es formar programadores expertos, sino dotar al administrador de sistemas de las competencias necesarias para gestionar y optimizar servicios mediante el desarrollo de herramientas personalizadas. Exploraremos por qué Linux se ha consolidado como el ecosistema predilecto para el desarrollo y cómo la terminal actúa como nuestro intérprete principal para interactuar directamente con los procesos del núcleo.

A través del uso estratégico de lenguajes como Python, C y C++, aprenderemos a aprovechar la potencia de la línea de comandos para la automatización de tareas críticas. Nos enfocaremos en pequeñas aplicaciones prácticas para la administración en Ubuntu Server, tales como el monitoreo proactivo de servicios, la implementación de sistemas de autorrecuperación (reinicio automático ante fallos) y la configuración programática de procesos. De esta manera, el código se convierte en un aliado indispensable para garantizar la alta disponibilidad y la eficiencia operativa del servidor.

## Instalación y configuración del entorno de desarrollo

Para instalar Python, C y C++ en Linux Ubuntu desde la terminal, primeramente, abrimos una terminal y actualizamos el mismo con el comando *update*.

```
root@svrlinux:~# apt update ↵ Enter
```

En seguida se puede verificar si el programa Python está instalado con el comando python3 --version

```
root@svrlinux:~# python3 –version ↵ Enter
```

Si no está instalado, puede utilizar el comando apt install python3 para instalar la versión más reciente de Python 3.

```
root@svrlinux:~# apt install python3 ↵ Enter
```

Ahora, para instalar el compilador de C, primero verificamos si está instalado con el comando gcc --version.

```
root@svrlinux:~# gcc –version ↵ Enter
```

Si no está instalado, puede utilizar el comando apt install build-essential, que incluye gcc junto con otras herramientas necesarias para compilar programas en C.

```
root@svrlinux:~# apt install build-essential ↵ Enter
```

Finalmente, para instalar el compilador de C++, primero comprobamos si g++ (el compilador de GNU para C++) está instalado con g++ --version.

```
root@svrlinux:~# apt g++ --version ↵ Enter
```

Si no está instalado, se puede instalar como parte de *build-essential*, o directamente con sudo apt install g++.

```
root@svrlinux:~# apt install g++ ↵ Enter
```

A continuación, podemos verificar que todo esté instalado correctamente comprobando las versiones de Python, gcc y g++ de nuevo.

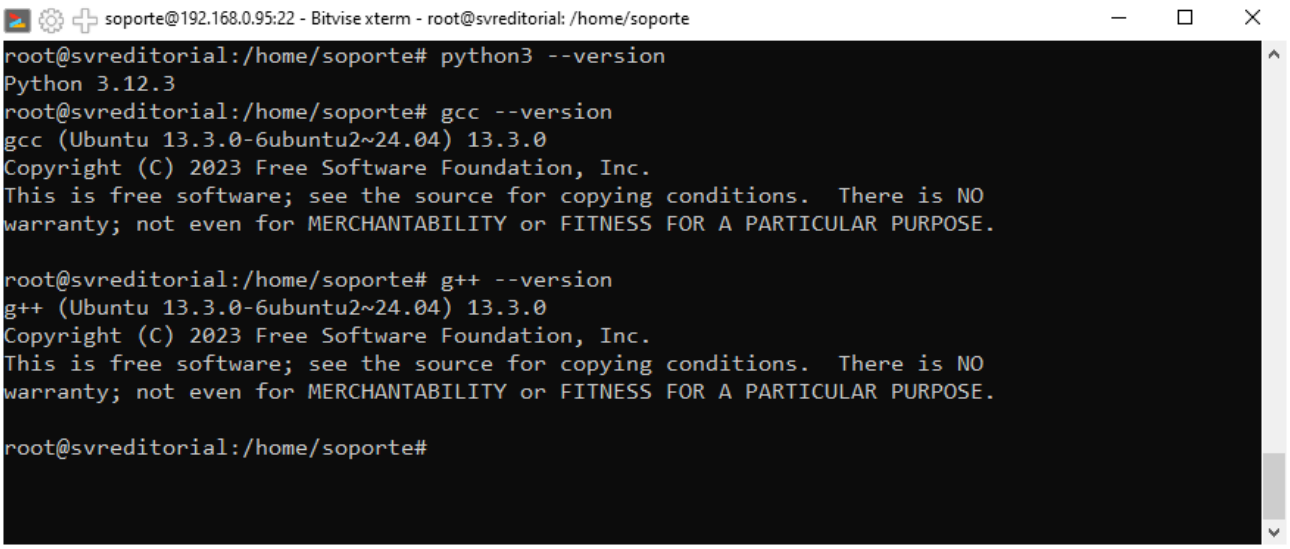



Figura 77. Versiones de Python, C y C++ visto desde la terminal

## Fundamentos de programación en Linux

En el capítulo 3 se abordó el uso de la terminal y comandos básicos de Linux. Para programar en Linux es primero necesario conocer algunos de los comandos más utilizados e importantes del sistema operativo, como por ejemplo para listar archivos, cambiar de directorio, mostrar la ruta actual, crear una carpeta, eliminar un archivo, también como manejar procesos y ejecutar programas, por ejemplo para ver los procesos en ejecución utilizamos el comando ps, para terminar un proceso el comando kill, para cambiar permisos de archivos el comando chmod.

Los lenguajes más comunes de programación para Linux suelen ser: Python, Bash Scripting, C y C++, Perl y Ruby. Python es muy utilizado para crear *scripts* y automatización, Bash Scripting es ideal para automatizar tareas en la terminal (.sh), C y C++ son lenguajes de bajo nivel usados también en sistemas Linux y finalmente Perl y Ruby utilizados para desarrollar *scripts* y para administrar sistemas. En este apartado, nos enfocaremos únicamente en *scripts* Python y también en C y C++, orientado al manejo del sistema operativo Linux Ubuntu.

A continuación, vamos a elaborar un pequeño *script* en Python para Linux que muestra la dirección IP del sistema:

```
#!/usr/bin/env python3

import os

print("Tu dirección IP es: ")

os.system("hostname -I | awk '{print $1}' ")
```

Guardamos el archivo como ver_ip.py y le damos permisos de ejecución con chmod +x ver_ip, y en seguida ejecutamos el *script* desde el terminal.

```
root@svrlinux:~# ./ver_ip.py ↵ Enter
```

Si por acaso obtiene un error que dice "sh: 1: awk{print $1}: not found" ocurre porque hay un problema de sintaxis en el comando, como puede observar, falta un espacio después de awk. Si el problema persiste puede utilizar cut -d' ' -f1 de la siguiente manera:

```
#!/usr/bin/env python3
import os

print("Tu dirección IP es: ")
os.system("hostname -I | cut -d' ' -f1")
```

hostname -I obtiene todas las direcciones IP del sistema mientras que awk o cut solo muestra la primera IP.

En C++ para obtener la dirección IP en Linux, creamos el siguiente código.

```
#include <iostream>
#include <cstdlib> //para system()
int main(){
 std::cout << "Tu dirección IP es: \n";
 system("hostname -I | cut -d' ' -f1");
return 0;
}
```

Las dos primeras líneas son librerías que sí o sí debemos invocar con **#include <iostream>**, a continuación, está la función principal **main** entre las llaves estaría toda la codificación y finalmente se retorna 0 indicando que el programa terminó correctamente. Guardamos el archivo como **ip_check.cpp** igualmente le damos permisos de ejecución con **chmod +x**, y entonces ejecutamos, compilamos el *script* desde el terminal, tal cual se indica a continuación; si todo ha salido perfecto no saldrá ningún mensaje.

```
root@svrlinux:~# g++ ip_check.cpp -o ip_check ↵ Enter
root@svrlinux:~#
```

Al ejecutar esta línea de comando en el listado de los directorios veremos un archivo que se ha generado llamado **ip_check**, el cual es un archivo binario, para ejecutar el archivo simplemente ingresamos lo siguiente:

```
root@svrlinux:~# ./ip_check ↵ Enter
Tu dirección IP es:
192.168.0.10
root@svrlinux:~#
```

El código en lenguaje C para obtener la dirección IP es el siguiente:

```
#include <stdio.h>
#include <stdlib.h> //para system()

int main(){

 printf("Tu dirección IP es: \n");
 int status=system("hostname -I | cut -d' ' -f1");
```

```
 if(status !=0)
  {
    printf("Error al obtener la dirección IP.\n")
  }
 return 0;
}
```

Podemos observar que para el lenguaje C tenemos las librerías stdlib.h y stdio.h es una librería para la función printf(), existen otras funciones como scanf(), getchar(), fgets(), fopen(), etc. Guardamos el archivo como **ipcheck.c** igualmente le damos permisos de ejecución con **chmod + x**, y seguidamente ejecutamos t compilamos el *script* desde el terminal, tal cual se indica a continuación; si todo ha salido perfecto no saldrá ningún mensaje.

```
root@svrlinux:~# gcc ipcheck.c -o ipcheck ↵ Enter
root@svrlinux:~# ./ipcheck ↵ Enter
```

El objetivo fundamental de este contenido es enseñar a resolver problemas, mejorar, automatizar y gestionar procesos del sistema operativo gracias al uso de pequeños programas que pueden ser de utilidad, mediante la creación de algoritmos.

Un algoritmo es un conjunto de instrucciones programadas para resolver una tarea específica. Cuando el procesador es una computadora, el algoritmo se ha de expresar en un formato que se denomina programa, ya que el pseudocódigo o diagrama de flujo no son comprensibles por la computadora, aunque pueda entenderlos cualquier programador. Un programa se escribe en un lenguaje de programación y las operaciones que conducen a expresar un algoritmo en forma de programa se llaman programación. Así pues, los lenguajes utilizados para escribir los programas de computadoras son los lenguajes de programación y los programadores son los escritores y diseñadores de programas. El proceso de traducir un algoritmo en pseudocódigo a un lenguaje de programación se denomina codificación y el algoritmo escrito en un lenguaje de programación se denomina código fuente (Aguilar, 2008, pág. 36).

C y C++ son lenguajes compilados, mientras que Python es un lenguaje interpretado, por lo que, en los ejemplos anteriores, en el caso de Python, directamente se ejecutaba desde el terminal el *script* que contenía el código y enseguida presentaba el resultado, mientras que C y C++, primero se compilaba, se generaba un archivo binario o código máquina, y seguidamente se ejecutaba ese archivo binario desde el terminal, para obtener el resultado.

Tabla 6. Características entre lenguajes compilados e interpretados

| Característica | C / C++ (Compilados) | Python (Interpretado) |
|---|---|---|
| **Ejecución** | Se compilan a código máquina antes de ejecutarse. | Se ejecuta línea por línea mediante un intérprete. |
| **Velocidad** | Muy rápido (ejecuta código directamente en la CPU) | Más lento (traduce el código en tiempo de ejecución) |
| **Portabilidad** | Depende del compilador (se necesita recompilar para cada sistema) | Portátil (puede ejecutarse en cualquier sistema con el intérprete instalado) |
| **Errores** | Detecta errores en la fase de compilación. | Detecta errores en tiempo de ejecución. |
| **Ejemplo de ejecución** | g++ programa.cpp -o programa && ./programa | python programa.py |

Los programas escritos en lenguaje de alto nivel deben ser traducidos por un compilador antes de que se puedan ejecutar en una máquina específica. En la mayoría de los lenguajes de programación se requiere un compilador para cada máquina en la que se desea ejecutar programas escritos en un lenguaje específico (Aguilar, 2008, pág. 43).

Según Joyanes Aguilar (2008), aunque el proceso de diseñar programas es, esencialmente, un proceso creativo, se pueden considerar una serie de fases o pasos comunes que generalmente deben seguir todos los programadores. Estas fases son las siguientes: análisis del problema, diseño del algoritmo, codificación (implementación), ejecución, verificación y depuración, mantenimiento y documentación. A continuación, veamos brevemente cada una de estas fases:

**Análisis**. El problema se analiza teniendo presente la especificación de los requisitos dados por el cliente de la empresa o por la persona que encarga el programa.

**Diseño**. Una vez analizado el problema, se diseña una solución que conducirá a un algoritmo que resuelva el problema.

**Codificación** (implementación). La solución se escribe en la sintaxis del lenguaje de alto nivel (por ejemplo, Python, C o C++) y se obtiene un programa fuente que se interpreta, o se compila a continuación.

**Ejecución, verificación y depuración.** El programa se ejecuta, se comprueba rigurosamente y se eliminan todos los errores (denominados "*bugs*", en inglés) que puedan aparecer.

**Mantenimiento.** El programa se actualiza y modifica, cada vez que sea necesario, de modo que se cumplan todas las necesidades de cambio de sus usuarios.

**Documentación.** Escritura de las diferentes fases del ciclo de vida del *software*, esencialmente el análisis, diseño y codificación, unidos a manuales de usuario y de referencia, así como normas para el mantenimiento.

Trabajaremos únicamente desde un terminal, creando *scripts* que realicen alguna tarea específica en cuanto a la administración del sistema operativo. El terminal es una interfaz de línea de comandos que nos permitirá interactuar con el sistema operativo mediante comandos de texto, del cual podremos principalmente ejecutar nuestros propios programas y *scripts*.

## Scripting con Python para la gestión de Linux

En esta sección, aplicaremos Python para interactuar directamente con el sistema operativo. Utilizaremos librerías nativas como os y sys para automatizar la exploración de directorios y el procesamiento de archivos de diagnóstico.

A continuación, un *script* en Python que imprime el mensaje de "Hola Mundo". Creamos un archivo llamado **mensaje.py** e ingresamos el siguiente código:

```
#!/usr/bin/env python3


print("Hola mundo")
```

Guardamos, y ejecutamos como:

```
root@svrlinux:~# ./mensaje.py ↵ Enter
```

Es importante recordar que estos *scripts* pueden también escribirse desde un IDE como **Visual Studio Code** o **PyCharm Community** y luego copiarse a Linux vía SSH.

**Ejercicio 1.** Crear un *script* en Python llamado **listar.py** que liste archivos y directorios del sistema operativo.

```
#!/usr/bin/env python3
import os
```

```
    directorio = "/var/www/html"
try:
    archivos = os.listdir(directorio)
    print(f"Contenido de {directorio}:")
    for archivo in archivos:
        print(archivo)
except PermissionError:
    print("Permiso denegado")
except FileNotFoundError:
    print("El directorio no existe")
except Exception as e:
    print(f"error: {e}")
```

La instrucción **import os** importa el módulo, que permite interactuar con el sistema operativo, sea de Linux, Windows y MacOS, lo que significa que es multiplataforma manipulando archivos o directorios, ejecutando comandos del sistema u obteniendo información de este.

Guardamos el archivo y le damos permisos de ejecución con **chmod + x**. Si acaso se nos presenta un mensaje de error con la palabra **indentation**, significa que se debe verificar en el código los espacios en blanco o tabulaciones al inicio de una línea de código para organizar la estructura del programa, ya que Python define de esta manera obligatoriamente los bloques de código. Seguidamente ejecutamos como:

```
root@svrlinux:~# ./listar.py ┘ Enter
```

Hay algunas diferencias que debe tener en cuenta en cuanto a la forma en que funcionan ciertos comandos o rutas en estos sistemas operativos. A continuación, crearemos algunos ejemplos de uso de **os en Python,** el módulo os tiene las siguientes funciones:

- Manipulación de archivos y directorios (crear, eliminar, mover)
- Ejecución de comandos del sistema
- Información sobre el sistema (variables de entorno, usuario, etc.)

### Para listar archivos en un directorio

```
import os


archivos = os.listdir("/var/www/html")
print(archivos)
```

### Para obtener el directorio actual

```
import os

print(os.getcwd())  # Muestra la ruta actual
```

### Para verificar si un archivo existe

```
import os
if os.path.exists("archivo.txt"):
    print("El archivo existe")
else:
    print("No existe")
```

**Para ejecutar un comando del sistema**

```
import os

os.system("ls -l")  # En Linux/Mac

os.system("dir")  # En Windows
```

**Manejo de rutas de archivos**

Se debe tomar en cuenta que Windows utiliza barras invertidas (\) para las rutas de archivos, mientras que Linux y MacOS utilizan barras (/).

El módulo os abstrae estas diferencias. Por ejemplo, al usar funciones como **os.path.join**, **os** se encarga de usar la barra adecuada para el sistema operativo:

```
import os

ruta = os.path.join("carpeta", "archivo.txt")

print(ruta)  # En Windows: "carpeta\\archivo.txt", en Linux/MacOS: "carpeta/archivo.txt"
```

**os.system:** En Windows, algunos comandos pueden no existir, por lo que se pueden requerir alternativas.

**os.path:** Las rutas de archivos y directorios pueden ser diferentes, por eso es recomendable usar funciones como **os.path.join** y **os.path.sep** para manejar las rutas de forma segura.

Las variables de entorno, como **PATH**, pueden diferir entre los sistemas, pero **os** permite acceder a ellas de manera uniforme con **os.environ**.

**Ejercicio 2.** Crear un *script* en Python llamado **leer_archivo.py** que pueda leer el contenido de un archivo.

```
#!/usr/bin/env python3


def leer_archivo(ruta_archivo):
    try:
        with open(ruta_archivo, 'r', encoding='utf-8') as archivo:
            contenido = archivo.read()
            print("Contenido del archivo:")
            print(contenido)
    except FileNotFoundError:
        print(f"Error: El archivo '{ruta_archivo}' no se encontró.")
    except Exception as e:
        print(f"Ocurrió un error: {e}")

# Uso del script
if __name__ == "__main__":
    ruta = input("Ingrese la ruta del archivo a leer: ")
    leer_archivo(ruta)
```

Explicación del código anterior. Para el ejemplo en Windows puede crear un archivo llamado **archivo.txt** con el contenido que usted desee en la siguiente ruta **C:\Users\User\Documents\archivo.txt,** si lo va a ejecutar desde Linux puede ser dentro del usuario de home, por ejemplo: **\home\usuario\archivo.txt**

**def leer_archivo(ruta_archivo):** Define una función llamada **leer_archivo** que recibe un parámetro **ruta_archivo**, que es la ruta del archivo que se va a leer.

**try:** Intenta ejecutar el bloque de código dentro del **try**. Si ocurre un error, el código lo manejará en los bloques **except**.

**with open(ruta_archivo, 'r', encoding='utf-8') as archivo**: Usa **open(ruta_archivo, 'r', encoding='utf-8')** para abrir el archivo en modo lectura ('r') con codificación UTF-8 (para evitar problemas con caracteres es-

peciales). Y **with** garantiza que el archivo se cierre automáticamente después de usarlo. La variable archivo almacena el archivo abierto.

**contenido = archivo.read()**

**print("Contenido del archivo:")**

**print(contenido)**

Usa **archivo.read()** para leer todo el contenido del archivo y lo almacena en contenido. Muestra el mensaje "Contenido del archivo:" seguido del contenido leído.

**except FileNotFoundError:**

**print(f"Error: El archivo '{ruta_archivo}' no se encontró.")**

Si el archivo no existe, se captura la excepción **FileNotFoundError** y muestra un mensaje de error.

**except Exception as e:**

**print(f"Ocurrió un error: {e}")**

Si ocurre cualquier otro error (por ejemplo, problemas de permisos), se captura la excepción y muestra un mensaje con el detalle del error (e).

**if __name__ == "__main__":** Verifica si el *script* se está ejecutando directamente (y no importado como módulo). Si es así, ejecuta el bloque de código siguiente.

**ruta = input("Ingrese la ruta del archivo a leer: ")**. Solicita al usuario que ingrese la ruta del archivo. Guarda la entrada en la variable ruta.

**leer_archivo(ruta).** Llama a la función **leer_archivo**, pasando la ruta ingresada por el usuario.

Seguidamente ejecutamos como:

```
root@svrlinux:~# ./leer_archivo.py ↵ Enter
```

**Ejercicio 3.** Crear un programa que lea las últimas tres líneas del archivo access.log, mismo que es un archivo de registro (log) que almacena información sobre todas las solicitudes realizadas a un servidor web, como Apache. Por lo general, se encuentra en las siguientes ubicaciones dependiendo del servidor web: Apache: /var/log/apache2/access.log. Cada línea en access.log representa una solicitud HTTP, e incluye: Dirección IP del cliente (quién hizo la solicitud). Fecha y hora de la solicitud.

Método HTTP usado (GET, POST, etc.). URL solicitada. Código de estado HTTP (por ejemplo, 200 OK, 404 Not Found. Tamaño de la respuesta en bytes. Agente de usuario (navegador o bot que hizo la solicitud).

Ejemplo de línea en el archivo access.log

```
192.168.1.1 - - [21/Mar/2025:14:30:02 +0000] "GET /index.html HTTP/1.1" 200 1024 "https://google.com" "Mozilla/5.0"
```

192.168.1.1 → IP del visitante.
[21/Mar/2025:14:30:02 +0000] → Fecha y hora de la solicitud.
"GET /index.html HTTP/1.1" → Método y página solicitada.
200 → Código de estado HTTP (éxito).
1024 → Tamaño de la respuesta en bytes.
"Mozilla/5.0" → Navegador del usuario.

Esto es muy útil para realizar monitoreo del tráfico web, detección de ataques o accesos sospechosos y para optimización del rendimiento del sitio. Aunque podría simplemente ejecutar el comando tail -f /var/log/apache2/access.log, para ver las últimas líneas del archivo en tiempo real. El *script* en Python nos permitiría analizar a un nivel más avanzado como poder integrarlo en reportes, alertas o bases de datos, analizar URLs y extraer información útil.

```
#!/usr/bin/env python3


from collections import Counter

def leer_archivo(ruta_archivo):
    try:
        with open(ruta_archivo, 'r', encoding='utf-8') as archivo:
            lineas = archivo.readlines()
            print(f"Se han leído {len(lineas)} líneas del archivo.")
            return lineas
    except FileNotFoundError:
        print(f"Error: El archivo '{ruta_archivo}' no se encontró.")
        return []
    except Exception as e:
        print(f"Ocurrió un error: {e}")
        return []

def analizar_access_log(ruta_archivo):
    lineas = leer_archivo(ruta_archivo)
    if not lineas:
        return
    ips = []
    codigos_estado = []
    for linea in lineas:
        partes = linea.split()
        if len(partes) > 8:
            ips.append(partes[0])  # IP del cliente
            codigos_estado.append(partes[8])  # Código de estado HTTP

    print("\nTop 5 IPs que más han accedido:")
    for ip, count in Counter(ips).most_common(5):
        print(f"{ip}: {count} veces")

    print("\nCódigos de estado más comunes:")
    for codigo, count in Counter(codigos_estado).most_common():
        print(f"{codigo}: {count} veces")


if __name__ == "__main__":
    ruta = input("Ingrese la ruta del archivo access.log: ")
    analizar_access_log(ruta)
```

Una vez ejecutado el *script,* la salida es la siguiente:

```
Ingrese la ruta del archivo access.log: /var/log/apache2/access.log

Se han leído 155 líneas del archivo.

Top 5 IPs que más han accedido:

112.6.214.244: 45 veces

78.153.140.203: 40 veces

104.234.115.55: 29 veces

34.22.213.78: 4 veces

152.32.160.252: 4 veces
```

Códigos de estado más comunes:

400: 78 veces

404: 72 veces

200: 4 veces

"-": 1 veces

Como puede apreciarse, el *script* está hecho para leer y analizar un archivo access.log del servidor web Apache. Vamos a analizar cada línea de código.

```
#!/usr/bin/env python3
```

Esta línea le dice al sistema operativo que debe ejecutar el archivo usando el intérprete de Python 3. A continuación tenemos la línea siguiente:

```
from collections import Counter
```

**Counter** es una clase que permite contar elementos repetidos en una lista fácilmente.

Seguidamente definimos una **función leer_archivo.**

```
def leer_archivo(ruta_archivo):
```

Esta función intenta leer el archivo de logs cuya ruta le pasas como argumento.

```
try:
    with open(ruta_archivo, 'r', encoding='utf-8') as archivo:
        lineas = archivo.readlines()
        print(f"Se han leído {len(lineas)} líneas del archivo.")
        return líneas
```

Utilizamos with open(...) para abrir el archivo en modo lectura ('r') y con codificación utf-8,se procede a leer todas las líneas y devolverlas como lista.

```
except FileNotFoundError:
        print(f"Error: El archivo '{ruta_archivo}' no se encontró.")
```

Si el archivo no existe, presenta un mensaje de error.

```
except Exception as e:
        print(f"Ocurrió un error: {e}")
```

Si ocurre otro tipo de error, lo presenta en pantalla como "Ocurrió un error".

A continuación, se define una **función analizar_access_log**

```
def analizar_access_log(ruta_archivo):
```

Esta función es la que analiza el contenido del archivo de logs.

```
lineas = leer_archivo(ruta_archivo)
    if not lineas:
        return
```

Llama a la función anterior. Si no hay líneas, termina. Luego se crean listas vacías y, a medida que el programa analiza cada línea del archivo access.log, va agregando elementos a esas listas con .append().

```
ips = []
    codigos_estado = []
    for linea in lineas:
        partes = linea.split()
        if len(partes) > 8:
            ips.append(partes[0])  # IP del cliente
            codigos_estado.append(partes[8])  # Código de estado HTTP
```

Divide cada línea en partes por espacios. Toma:

partes[0]: la dirección IP del cliente.

partes[8]: el código de estado HTTP (ej. 200, 404, 500...).

Luego se reportan estadísticas con **counter**

```
print("\nTop 5 IPs que más han accedido:")
    for ip, count in Counter(ips).most_common(5):
        print(f"{ip}: {count} veces")
```

Cuenta cuántas veces aparece cada IP y muestra las 5 IPs más frecuentes.

```
print("\nCódigos de estado más comunes:")
    for codigo, count in Counter(codigos_estado).most_common():
        print(f"{codigo}: {count} veces")
```

Finalmente, cuenta los códigos de estado HTTP y los muestra ordenados por frecuencia.

```
if __name__ == "__main__":
    ruta = input("Ingrese la ruta del archivo access.log: ")
    analizar_access_log(ruta)
```

Este bloque solo se ejecuta si el archivo se ejecuta directamente (no si es importado). Se pide al usuario la ruta del archivo y llama a la función principal **analizar_access_log.**

## *Scripting* en C y C++ para la gestión de Linux

La programación en C y C++ en sistemas Linux es una práctica ampliamente utilizada en entornos de desarrollo de *software* debido a la potencia, eficiencia y control que ofrecen estos lenguajes de programación de bajo nivel. Linux, al ser un sistema operativo de código abierto y basado en Unix, proporciona una plataforma ideal para programar en C y C++, ya que el propio núcleo (*kernel*) de Linux está desarrollado principalmente en C.

C y C++ permiten a los desarrolladores escribir programas que interactúan directamente con el *hardware* y el sistema operativo, lo cual es básico para aplicaciones que requieren alto rendimiento, como sistemas embebidos, controladores, herramientas del sistema, juegos y sistemas operativos. Además, C++ añade características de programación orientada a objetos, facilitando la modularidad y reutilización del código.

En Linux, el entorno de desarrollo suele incluir herramientas como el compilador GCC (GNU Compiler Collection), editores de texto como Vim, Nano o Visual Studio Code, y utilidades de depuración como gdb. El uso del terminal de comandos es necesario para compilar, ejecutar, depurar y gestionar proyectos, ya que brinda al programador un control completo sobre el proceso de desarrollo. Si bien es posible utilizar cualquier editor de texto, en mi caso particular prefiero trabajar con Midnight Commander (**mc**) y ejecutar todas las acciones directamente desde la terminal. Es importante recordar que este enfoque está orientado al uso de Linux Ubuntu Server, el cual lo estamos operando sin un entorno gráfico, lo que resalta aún más la importancia del dominio de la línea de comandos. A continuación, escribamos el siguiente ejercicio en C.

**Ejercicio 1.** Escribir un código fuente en C que imprima un mensaje en pantalla y realice una operación simple de suma:

```
#include <stdio.h>  // Biblioteca estándar de entrada/salida

int main() {
    int a, b, suma;
    // Solicitar datos al usuario
    printf("Ingrese el primer número: ");
    scanf("%d", &a);
```

```
    printf("Ingrese el segundo número: ");
    scanf("%d", &b);
    // Sumar los números
    suma = a + b;
    // Mostrar el resultado
    printf("La suma de %d + %d es: %d\n", a, b, suma);

    return 0;
}
```

Guarda el archivo como **suma.c** y desde el terminal ejecuta la siguiente instrucción:

```
root@svrlinux:~# gcc suma.c -o suma ┘ Enter
root@svrlinux:~#./suma
```

**Ejercicio 2.** Escribir un código fuente en C++ que muestre el nombre del usuario actual.

```
#include <iostream>
#include <unistd.h>
#include <pwd.h>

int main() {
    uid_t uid = getuid();  // Obtener el ID de usuario
    passwd* pw = getpwuid(uid);  // Obtener info del usuario

    if (pw) {
        std::cout << "Usuario actual: " << pw->pw_name << std::endl;
    } else {
        std::cerr << "No se pudo obtener el nombre de usuario." << std::endl;
    }

    return 0;
}
```

En el código anterior se utilizan las funciones del sistema (*getuid* y getpwuid), las mismas que son para obtener el nombre del usuario actual en Linux. Es una tarea muy común para *scripts* o herramientas de administración.

Para compilarlo primero guardamos el archivo como **usuario.cpp** y desde el terminal ejecuta la siguiente instrucción:

```
root@svrlinux:~# g++ usuario.cpp -o usuario ┘ Enter
root@svrlinux:~#./usuario
```

**Ejercicio 3.** Escribir un código fuente en C++ para obtener el *hostname* y la dirección IP local.

```
#include <iostream> // Para entrada/salida estándar (std::cout, std::cerr)
#include <unistd.h> // Para gethostname()
#include <netdb.h> // Para gethostbyname()
#include <arpa/inet.h> // Para inet_ntoa(), struct in_addr

int main() {
    char hostname[256];

    if (gethostname(hostname, sizeof(hostname)) == 0) {
        std::cout << "Hostname: " << hostname << std::endl;

        struct hostent* he = gethostbyname(hostname);
        if (he && he->h_addr_list[0]) {
            struct in_addr* addr = (struct in_addr*)he->h_addr_list[0];
            std::cout << "IP local: " << inet_ntoa(*addr) << std::endl;
        } else {
            std::cerr << "No se pudo resolver la IP local." << std::endl;
        }
    } else {
        std::cerr << "No se pudo obtener el hostname." << std::endl;
    }

    return 0;
}
```

A continuación, una breve explicación del código:

Primero, se declara un arreglo de caracteres para almacenar el nombre del *host* (*hostname*), esto es con `char hostname[256];` el cual tiene espacio para 256 caracteres, suficiente para la mayoría de sistemas.

Llamamos a la función *gethostname*() para obtener el nombre del equipo (*host*). Si la llamada tiene éxito (devolvemos 0), y continuamos. if (gethostname(hostname, sizeof(hostname)) == 0)

Luego mostramos el nombre del equipo en pantalla con std::cout << "Hostname: " << hostname << std::endl;

La función *gethostbyname*() busca la información del *host* en la red o en /etc/hosts. Retornamos un puntero a una estructura *hostent* que contiene información como la IP del *host*, como se indica en struct hostent* he = gethostbyname(hostname);

Se verifica que la búsqueda fue exitosa y que hay al menos una IP asociada, justamente en la línea que contiene la condición if (he && he->h_addr_list[0])

Convertimos la dirección binaria de red (h_addr_list[0]) a una estructura in_addr, con la siguiente instrucción struct in_addr* addr = (struct in_addr*)he->h_addr_list[0];

La función inet_ntoa() convierte la IP de formato binario a una cadena legible como 192.168.1.10. Muestra la IP local obtenida, con el código std::cout << "IP local: " << inet_ntoa(*addr) << std::endl;

Si no se puede convertir el *hostname* a IP, se muestra un mensaje de error, con la instrucción *else* (caso contrario). else { std::cerr << "No se pudo resolver la IP local." << std::endl; } y si *gethostname*() falla, también se informa al usuario. else { std::cerr << "No se pudo obtener el hostname." << std::endl; }

Finalmente guardamos el archivo como **hostname_ip.cpp** y desde el terminal ejecutamos la siguiente instrucción:

root@svrlinux:~# g++ hostname_ip.cpp -o hostname_ip ↵ **Enter**

root@svrlinux:~#./hostname_ip

**Ejercicio 4.** Escribir un código fuente en lenguaje C de seguridad, que verifica si el programa se está ejecutando como *root* (súperusuario).

```
#include <stdio.h>
#include <unistd.h>

int main() {
   if (geteuid() == 0)
      printf("Ejecución como root (superusuario).\n");
   else
      printf("No se está ejecutando como root.\n");

   return 0;
}
```

La librería #include <unistd.h> nos permite acceder a funciones POSIX como *geteuid*(). La función *geteuid*() nos devuelve el ID efectivo del usuario que ejecuta el programa. El número 0 significa que se está ejecutando como *root*. Y finalmente la función *printf*() nos imprime el resultado en la terminal. Guardamos el archivo como **verificar_root.c** y desde el terminal ejecutamos la siguiente instrucción:

```
root@svrlinux:~# gcc verificar_root.c -o verificar_root ┘ Enter
root@svrlinux:~#./verificar_root
```

**Ejercicio 5.** Escribir un código en lenguaje C que verifique los permisos de lectura, escritura y ejecución de un archivo para el usuario actual en Linux, utilizando funciones del sistema.

```
#include <stdio.h>
#include <unistd.h>

int main() {
   const char *ruta = "archivo.txt";
   // Verificar permiso de lectura
   if (access(ruta, R_OK) == 0)
      printf("Tienes permiso de LECTURA sobre '%s'\n", ruta);
   else
      printf("NO tienes permiso de LECTURA sobre '%s'\n", ruta);
   // Verificar permiso de escritura
   if (access(ruta, W_OK) == 0)
      printf("Tienes permiso de ESCRITURA sobre '%s'\n", ruta);
   else
      printf("NO tienes permiso de ESCRITURA sobre '%s'\n", ruta);
   // Verificar permiso de ejecución
   if (access(ruta, X_OK) == 0)
      printf("Tienes permiso de EJECUCIÓN sobre '%s'\n", ruta);
   else
      printf("NO tienes permiso de EJECUCIÓN sobre '%s'\n", ruta);
   return 0;
}
```

#include <unistd.h> es una directiva del preprocesador en el lenguaje C, y su función es incluir la biblioteca de cabecera unistd.h, que proporciona acceso a la API del sistema operativo tipo Unix, como Linux o macOS. Es básico para tareas como verificación de permisos, manipular procesos, acceder a archivos o controlar usuarios.

Guardamos el archivo como **verificar_permisos.c** y desde el terminal ejecutamos la siguiente instrucción:

```
root@svrlinux:~# gcc verificar_permisos.c -o verificar_permisos ┘ Enter
root@svrlinux:~#./verificar_permisos
```

**Ejercicio 6:** Verificador de Integridad de Servicios con C++

El objetivo de este programa es interactuar con systemd (el gestor de servicios de Ubuntu) para verificar si un servicio específico (como Apache) está funcionando correctamente.

```
#include <iostream>
#include <cstdlib> // Librería necesaria para usar system()

int main() {
    std::cout << "--- Verificador de Integridad de Servicios ---" << std::endl;

    // Definimos el servicio que queremos consultar
    std::string servicio = "apache2";

    // Construimos el comando: systemctl is-active --quiet nombre_servicio
    // --quiet se usa para que el comando no imprima nada en pantalla
    std::string comando = "systemctl is-active --quiet " + servicio;

    // Ejecutamos el comando y capturamos el valor de retorno
    int salida = std::system(comando.c_str());

    // En Linux, un código de salida 0 significa "éxito" o "activo"
    if (salida == 0) {
        std::cout << "ESTADO: El servicio [" << servicio << "] esta funcionando correctamente." << std::endl;
    } else {
        std::cout << "ALERTA: El servicio [" << servicio << "] NO esta activo o no existe." << std::endl;
        std::cout << "Sugerencia: Ejecute 'sudo systemctl start " << servicio << "'" << std::endl;
    }

    return 0;
}
```

Para que este código se convierta en una herramienta útil en Ubuntu Server, debemos compilarlo usando g++ (parte del paquete *build-essential* que instalamos previamente):

**Compilar el código:**

g++ check_service.cpp -o check_service

Esto generará un archivo binario ejecutable llamado check_service.

**Dar permisos de ejecución (si es necesario):**

chmod +x check_service

Ejecutar la herramienta:

./check_service

## Conclusiones

**Python: Flexibilidad y rapidez para tareas de alto nivel**

Python es ideal para automatizar tareas administrativas, como gestión de archivos, monitoreo de servicios, *parsing* de *logs*, y conexiones a API.

Su sintaxis clara y su amplio ecosistema de librerías (como os, subprocess, psutil) hacen que sea una herramienta poderosa para *scripts* y prototipos rápidos.

Sin embargo, no es recomendable para tareas que requieren bajo nivel o control total del sistema, como controladores, módulos del *kernel*, o manipulación directa de *hardware*.

**Ventajas:** rapidez de desarrollo y legibilidad del código.

**C: Control total, esencial para tareas de bajo nivel**

C es el lenguaje base del núcleo Linux y muchas de sus herramientas. Ideal para desarrollar utilidades del sistema, servicios en segundo plano (*daemons*), y tareas críticas de administración como control de dispositivos, uso de memoria o redes.

Ofrece máximo rendimiento y control del *hardware*, pero a costa de mayor complejidad y riesgo (errores de punteros, gestión manual de memoria, etc.). Requiere un conocimiento más profundo del sistema y herramientas como gcc, make, y gdb.

**Ventajas:** eficiencia y acceso directo al sistema operativo.

**C++: Potencia orientada a objetos con control del sistema**

C++ combina el poder de bajo nivel de C con características de alto nivel (POO, plantillas, excepciones), lo que lo hace útil para herramientas administrativas complejas, servicios de red, e interfaces CLI más estructuradas.

Aunque no es tan simple como Python ni tan cercano al *hardware* como C, es útil cuando se requiere organización de código, reusabilidad y rendimiento al mismo tiempo. Es ideal para aplicaciones administrativas que crecen en complejidad y requieren modularidad.

**Ventajas:** equilibrio entre control del sistema y diseño de *software* robusto.

# CAPÍTULO IX Aspectos de seguridad en Linux.

## Contenidos del Capítulo IX

- Introducción a la seguridad
- Gestión de usuarios y permisos
- Seguridad en servicios de red
- Registro y monitoreo de actividad
- Actualizaciones y gestión de paquetes
- Copias de seguridad y recuperación
- Buenas prácticas y recomendaciones finales

## Introducción a la seguridad

Desde hace mucho tiempo, los sistemas operativos como Windows han sido sujetos de ataques informáticos como *malwares* principalmente de virus, troyanos, *spyware*, etc. ingeniería social, ataques de fuerza bruta, ataques a aplicaciones web, etc. Los sistemas operativos más comúnmente afectados por ataques son Windows en mayor medida al ser el más usado en el mundo, especialmente en entornos corporativos y de usuario final, seguidos por Android, muy extendido en *smartphones* y *tablets*, especialmente con fabricantes que no lo actualizan. A continuación, Linux es también ampliamente utilizado en servidores, *cloud* y dispositivos IoT, aunque es seguro por diseño, la mala configuración es la principal fuente de fallos, y en menor medida sistemas macOS al ser un sistema cerrado, pero objetivo atractivo por el valor de sus usuarios. Esto es una tendencia con cierta lógica, pero esto puede cambiar con el avance tecnológico. Las aplicaciones también son susceptibles de ataques sobre todo si tienen un amplio uso.

Existen también ciberataques dirigidos a otros sistemas operativos y dispositivos, como los de la "Internet de las Cosas" (IoT), que también pueden ser vulnerables.

Los ciberataques pueden acarrear consecuencias significativas, como la pérdida de información, el robo de datos confidenciales, afectaciones a la imagen pública y perjuicios económicos. Por ello, resulta fundamental que tanto personas como entidades adopten estrategias de ciberseguridad para resguardarse frente a estas amenazas.

"Los posibles ataques son una amenaza constante a nuestros sistemas y pueden comprometer su funcionamiento, así como los datos que manejamos; ante todo lo cual, siempre tenemos que definir una cierta política de requerimientos de seguridad sobre nuestros sistemas y datos. Las amenazas que podemos sufrir podrían afectar a los aspectos siguientes: confidencialidad, integridad, accesibilidad, autenticación y control de acceso" (Jorba Esteve & Suppi Boldrito, 2004, pág. 340)

Ningún sistema operativo es 100 % seguro, pero, particularmente en sistemas Linux para una seguridad adecuada basta con una correcta configuración en cuanto a servicios, a nivel de archivos y directorios, actualización preventiva de paquetes y seguimiento continuo por parte del administrador es suficiente para un normal funcionamiento.

En los siguientes contenidos iremos desglosando diferentes lineamientos de seguridad aplicadas en Linux Ubuntu Server como gestión de usuarios, actualización de paquetes, herramientas de seguridad, etc.

## Gestión de usuarios y permisos

La gestión de usuarios y permisos en Linux es muy importante para mantener la seguridad, además de mantener una organización y control de acceso adecuados. Esto permite quien puede entrar al sistema (usuarios) y qué pueden hacer cada uno de ellos, por ejemplo, un usuario puede únicamente acceder a su carpeta personal y otro usuario puede tener permisos para instalar *software* o modificar configuraciones.

Limitar los permisos reduce el riesgo de eliminar accidentalmente archivos importantes, modificación no autorizada del sistema y ejecutar programas maliciosos.

### Creación y administración de usuarios y grupos

Para agregar un nuevo usuario con su directorio personal, podemos utilizar el comando siguiente:

root@svrlinux:~# adduser nombre_usuario ↵ **Enter**

Este comando crea el usuario y solicita información, incluida la contraseña.

### Administración de usuarios

Para cambiar la contraseña de un usuario, utilizamos el siguiente comando:

root@svrlinux:~# passwd nombre_usuario ↵ **Enter**

Para eliminar un usuario, ingresamos el siguiente comando:

root@svrlinux:~# deluser nombre_usuario ↵ **Enter**

### Creación y administración de grupos

Para crear un grupo:

root@svrlinux:~#addgroup nombre_grupo ↵ **Enter**

Para agregar un usuario a un grupo:

root@svrlinux:~#usermod -aG nombre_grupo nombre_usuario ↵ **Enter**

Para eliminar un usuario de un grupo:

root@svrlinux:~#deluser nombre_usuario nombre_grupo ↵ **Enter**

### Gestión de permisos

Para cambiar el propietario de un archivo o carpeta:

root@svrlinux:~#chown usuario:grupo archivo_o_directorio ↵ **Enter**

Para modificar los permisos:

root@svrlinux:~#chmod 755 archivo_o_directorio ↵ **Enter**

## Seguridad en servicios de red

La seguridad en los servicios de red nos permite proteger la información, los sistemas y los usuarios que se comunican a través de una red, ya sea local (LAN) o global como internet, esto nos permite proteger la información y los servicios de red como correo electrónico, sitios web, servidores de archivos, bases de datos, etc.

Existen muchos ataques en red como: *phishing*, *malware*, ataques de denegación de servicio DDoS, instrucciones de *hackers*, la seguridad en red ayuda a detectar, prevenir y responder ante estas amenazas. Un sistema sin protección puede ser atacado, saturado y dejando a los usuarios sin acceso a los servicios esenciales, como plataformas educativas, servicios bancarios, sistemas de salud, etc.

Muchas instituciones educativas, gubernamentales, financieras deben cumplir con leyes de protección de datos. La seguridad en red ayuda a cumplir estas normativas y evitar sanciones legales. Ejemplos de servicios de red que deben ser seguros: servidores web (HTTP/HTTPS), servidores de correo (SMTP, IMAP, POP), accesos remotos, bases de datos accesibles por red. En el capítulo IV explicamos un poco sobre nmap, una herramienta muy potente para el escaneo de puertos, el escaneo de puertos es sustancial para detectar posibles vulnerabilidades.

"Una de las principales medidas de seguridad en sistemas Linux es la configuración adecuada del cortafuegos. Este actúa como una barrera entre la red interna y externa, filtrando el tráfico de red entrante y saliente según las reglas predefinidas. Mediante el uso de herramientas como *iptables* o *firewalld*, los administradores pueden especificar qué tipos de tráfico están permitidos y qué puertos están abiertos para la comunicación" (Belmar, 2025, pág. 91).

En Linux Ubuntu Iptables suele venir preinstalado, iptables es una herramienta en Linux, diseñada para gestionar y filtrar el tráfico de red a través de reglas establecidas.

Para instalar iptables, podemos ingresar el siguiente comando:

```
root@svrlinux:~#sudo apt update
root@svrlinux:~#sudo apt install iptables ┘ Enter
```

Para verificar que se instaló correctamente:

```
root@svrlinux:~#sudo iptables -L ┘ Enter
```

Ubuntu 20.04+ usa nftables como *backend* por defecto, aunque aún puedes usar comandos iptables gracias a la compatibilidad (iptables-nft):

```
root@svrlinux:~#sudo update-alternatives --display iptables ┘ Enter
```

Si ves que apunta a /usr/sbin/iptables-nft, entonces está usando el *backend* moderno, lo cual está correcto.

Ahora veamos algunas reglas básicas de iptables:

Para bloquear una dirección IP, ingresamos el siguiente comando:

root@svrlinux:~#sudo iptables -A INPUT -s 192.168.1.100 -j DROP ┘ **Enter**

Permitir tráfico SSH, utilizamos el siguiente comando:

root@svrlinux:~#sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT

Para guardar las reglas y que se mantengan tras reiniciar, ya que Ubuntu no guarda las reglas automáticamente, se debe instalar lo siguiente:

root@svrlinux:~#sudo apt install iptables-persistent ┘ **Enter**

Luego, guarde las reglas actuales con el comando:

root@svrlinux:~#sudo netfilter-persistent save

Linux Ubuntu también ofrece **ufw** (Uncomplicated Firewall) como alternativa a iptables, con una sintaxis más sencilla. Por ejemplo:

```
sudo ufw allow 22/tcp
sudo ufw deny 80
sudo ufw enable
```

Igualmente, podemos crear un *script* para ejecutar las reglas que vayamos estableciendo, como se indica en el código a continuación, primeramente, creamos un archivo que puede llamarse iptables_reglas.sh y dentro ubicamos las siguientes instrucciones:

```
#!/bin/bash

# Limpiar reglas existentes
iptables -F
iptables -X
iptables -Z

# Política por defecto: denegar todo
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT
```

```
# Permitir tráfico en loopback (localhost)
iptables -A INPUT -i lo -j ACCEPT

# Permitir conexiones ya establecidas
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

# Permitir SSH
iptables -A INPUT -p tcp --dport 22 -j ACCEPT

# Permitir HTTP y HTTPS
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -p tcp --dport 443 -j ACCEPT

# Bloquear una IP específica (ejemplo)
# iptables -A INPUT -s 192.168.1.100 -j DROP

# Registrar paquetes descartados (opcional)
iptables -A INPUT -j LOG --log-prefix "IPTABLES DROPPED: " --log-level 4
```

A continuación, podemos dar permisos de ejecución con el comando chmod +x iptables_rules.sh, y ejecutamos con privilegios de súperusuario o *root* de la siguiente manera:

```
root@svrlinux:~#sudo ./iptables_reglas.sh ┘ Enter
```

Ahora podemos utilizar la herramienta **nmap** para verificar el estado de los puertos, abierto o cerrado.

Escaneo básico para detectar servicios en ejecución:

```
root@svrlinux:~#nmap -sS dirección_ip
```

Para escanear todas las máquinas en una red:

```
root@svrlinux:~#nmap -sn 192.168.1.0/24
```

"Para enfrentar la seguridad de Linux de manera efectiva, es fundamental adoptar un enfoque proactivo y seguir algunas recomendaciones generales que son comunes a cualquier sistema operativo: Mantener el sistema actualizado, utilizar contraseñas seguras, limitar los privilegios de usuario, configurar un cortafuegos, utilizar herramientas de monitoreo y registro, proteger los servicios de red, realizar copias de seguridad periódicas. Siguiendo estas recomendaciones generales iniciales, ya estamos en condiciones de profundizar en medidas de seguridad más avanzadas, por ejemplo, la gestión de puertos" (Belmar, 2025, pág. 93).

"Los puertos son puntos de conexión específicos que permiten la comunicación entre dispositivos a través de una red. Cada puerto está asociado a un número único, conocido como número de puerto, que identifica de manera exclusiva un servicio o aplicación determinada en un dispositivo" (Belmar, 2025, pág. 94).

Según Belmar (2025), los puertos están clasificados en puertos bien conocidos, puertos registrados y puertos dinámicos. A los puertos bien conocidos también se los conoce como puertos reservados y van del 0 al 1023. Están asignados por la Internet Assigned Numbers Authority (IANA) a servicios específicos, como por ejemplo el 80 para HTTP, el 443 para HTTPS y el 22 para SSH.

Seguidamente, tenemos los puertos registrados que van desde 1024 a 49151, están asignados a servicios y aplicaciones específicas, algunos ejemplos incluyen el puerto 3306 para MySQL y el 5432 para PostgreSQL.

Finalmente, los puertos dinámicos o privados que van desde el 49152 al 65535, estos puertos están disponibles para ser utilizados por aplicaciones y servicios de manera dinámica. Se emplean, principalmente, para establecer conexiones salientes y temporales, como por ejemplo el puerto 49160 para servicio de protocolo SNMP, del número 49162 al 49166 considerados puertos de *backdoor* utilizados por algunos troyanos y el puerto 49158 para el servicio Oracle Secure Backup.

Comprender los puertos y sus números asignados es esencial para gestionar adecuadamente la seguridad y el acceso en sistemas Linux. Esto abarca configurar el *firewall* para regular el tráfico que entra y sale de la red, además de detectar y solucionar inconvenientes vinculados a la conectividad y los servicios en ejecución (Belmar, 2025, pp. 95-97)

**Tipos de puertos: TCP vs. UDP**

Según Belmar (2025), en el contexto de la comunicación de red, los puertos TCP (Protocolo de Control de Transmisión) y UDP (Protocolo de Datagrama de Usuario) son dos protocolos fundamentales que operan en la capa de transporte del modelo OSI.

Cada uno tiene características específicas que los hacen adecuados para diferentes aplicaciones y escenarios de red. El Protocolo de Control de Transmisión (TCP) es un protocolo orientado a la conexión, confiable, que se utiliza para la transmisión de datos en redes. Los puertos TCP son adecuados para aplicaciones que requieren una transmisión confiable de datos, como la transferencia de archivos, el correo electrónico y la navegación web.

El Protocolo de Datagrama de Usuario (UDP) es un protocolo sin conexión y no confiable que se utiliza para la transmisión de datos en redes. Los puertos UDP son apropiados para aplicaciones donde la velocidad y la eficiencia son más importantes que la confiabilidad, como juegos en línea, *streaming* de medios y comunicaciones en tiempo real. En resumen, los puertos son ideales para aplicaciones que requieren una transmisión confiable de datos, mientras que los UDP son más indicados para aplicaciones que priorizan la velocidad y la eficiencia sobre la confiabilidad. (Belmar, 2025, pp. 98-100).

## Registro y monitoreo de actividad

El registro y monitoreo de actividad es una función esencial dentro de la administración de cualquier sistema operativo, y adquiere una importancia aún mayor en entornos basados en Linux, donde la estabilidad, la seguridad y el rendimiento son factores importantes. Esta función se encarga de capturar, almacenar y analizar en tiempo real o de forma diferida una amplia variedad de eventos del sistema que ocurren constantemente, tales como accesos de usuarios, ejecución de comandos, conexiones de red, modificaciones de archivos sensibles, entre otros.

Por otro lado, el registro (o *logging*) consiste en la documentación automática de cada evento relevante que sucede en el sistema. Estos eventos son almacenados en archivos de log, los cuales pueden encontrarse, por ejemplo, en el directorio /var/log/. Cada uno de estos archivos contiene información detallada que permite reconstruir la cronología de sucesos, facilitando la auditoría y la identificación de posibles errores o incidentes de seguridad.

En todo caso, el monitoreo implica la vigilancia activa o periódica de estos registros y del estado actual del sistema. Esta tarea permite detectar comportamientos inusuales, intentos de acceso no autorizados, fallas de *hardware* o *software*, uso anormal de recursos, y otros indicadores que podrían representar amenazas o comprometer la integridad del sistema.

El monitoreo también puede realizarse manualmente mediante el análisis de archivos de log o de forma automatizada con herramientas especializadas que permiten recibir alertas, generar informes, o incluso tomar acciones correctivas de manera inmediata.

**¿Qué es el monitoreo de actividad?**

El monitoreo de actividad es el proceso de observar, medir y analizar el comportamiento de un sistema, usuario o proceso en tiempo real. Su objetivo es detectar patrones, identificar anomalías y optimizar el rendimiento en diversos contextos, como la seguridad informática, la gestión de redes, el seguimiento de proyectos o el análisis de productividad. Es el proceso de revisar, analizar y responder a los datos registrados para:

- Detectar intrusiones o ataques en tiempo real.
- Ver el uso de recursos del sistema (RAM, CPU, disco).
- Identificar procesos o servicios anómalos.
- Controlar qué usuarios están activos y qué están haciendo.
- Supervisar el tráfico y el estado de los servicios.

Existen herramientas especializadas para este propósito, como *software* de monitoreo de redes, sistemas de auditoría de seguridad y plataformas de análisis de datos. A continuación, veamos algunos ejemplos.

**Herramientas comunes:**

- top, htop: monitoreo de procesos
- journalctl, logwatch: revisar *logs* del sistema
- auditd: auditoría detallada
- fail2ban: bloqueo automático ante múltiples intentos fallidos de acceso

**¿Qué es el registro de actividad?**

El registro de actividad es un mecanismo que permite documentar y supervisar las acciones realizadas en un sistema, plataforma o entorno digital. Su propósito es proporcionar un historial detallado de eventos, facilitando la auditoría, el monitoreo de seguridad y el cumplimiento de normativas. Es decir, es un proceso mediante el cual el sistema guarda información y lo almacena generalmente en archivos .log bajo un directorio log, esto es para lo siguiente:

- Accesos de usuarios (inicios de sesión, cambios de permisos)
- Ejecución de comandos o procesos
- Conexiones de red entrantes/salientes
- Fallos de autenticación
- Cambios en archivos sensibles
- Errores del sistema o servicios

**¿Dónde se guardan?**

Puedes comprobarlo y revisar que los registros de actividad se almacenan en diferentes ubicaciones dependiendo del sistema operativo y la aplicación que los genera. A continuación, veamos algunos ejemplos:

**Windows:** Los registros de actividad se guardan en el visor de eventos, accesible con eventvwr.msc. También pueden almacenarse en archivos de registro (.log) en C:\Windows\System32\winevt\Logs\.

**Linux:** Se encuentran en /var/log/, donde archivos como auth.log, syslog y dmesg registran eventos del sistema, /var/log/auth.log (autenticación), /var/log/syslog o /var/log/messages (eventos generales).

**Aplicaciones web y servidores:** Los registros suelen guardarse en directorios específicos dentro del servidor, como /var/log/apache2/access.log para Apache o /var/log/nginx/ para Nginx.

**Bases de datos:** Sistemas como MySQL y PostgreSQL almacenan logs en archivos dentro de sus directorios de instalación.

**Monitorear los puertos abiertos es esencial para la seguridad**

Mantener un control sobre los puertos abiertos ayuda a prevenir accesos no autorizados y posibles vulnerabilidades en un sistema. Puede comprobarlo desde un terminal de Linux. Algunas herramientas muy utilizadas para este propósito incluyen:

netstat: Muestra conexiones y puertos abiertos.

```
root@svrlinux:~#netstat -tulnp ┘ Enter
```

ss: Más rápido que netstat para listar sockets activos.

```
root@svrlinux:~#ss -tulnp ┘ Enter
```

lsof: Muestra qué procesos están usando puertos específicos.

```
root@svrlinux:~#lsof  -i :80 ┘ Enter
```

Estas herramientas ayudan a identificar posibles accesos no autorizados y gestionar el tráfico en un servidor. Principalmente, en el ámbito de la seguridad, auditoría, diagnóstico y cumplimiento de normas.

Seguridad: para detectar accesos no autorizados, prevenir ataques.

Auditoría: tener evidencia de lo que ocurrió en el sistema.

Diagnóstico: para identificar la causa de fallos del sistema o de aplicaciones.

Cumplimiento: en entornos corporativos, cumplir con normas como ISO 27001, PCI-DSS, etc.

## Actualizaciones y gestión de paquetes

Ubuntu utiliza el gestor de paquetes APT (*Advanced Package Tool*) para instalar, actualizar y eliminar *software* de manera eficiente. Vamos a explorar los conceptos importantes y los comandos esenciales. En Ubuntu (y en general en sistemas Linux), la gestión de paquetes es el mecanismo que permite:

- Instalar, actualizar, eliminar y mantener *software*.
- Administrar automáticamente dependencias (otros paquetes que necesita un programa para funcionar).
- Mantener el sistema organizado y consistente.
- Ubuntu usa el sistema de paquetes deb (Debian) y herramientas como apt, dpkg, y snap.

A continuación, veamos el proceso recomendado para actualizar el sistema de Ubuntu Server. Antes de instalar paquetes, es importante iniciar primero con la actualización de los repositorios y de los paquetes existentes; esto lo podemos realizar con el siguiente comando:

```
root@svrlinux:~#sudo apt update
```

Seguidamente, hay que actualizar todos los paquetes instalados, esto es instalar las versiones más recientes de los paquetes sin modificar dependencias. Ingresamos el siguiente comando:

```
root@svrlinux:~#sudo apt upgrade -y
```

También podemos hacer actualizaciones con cambios en dependencias. Este comando actualiza el sistema y permite cambios en dependencias y eliminaciones necesarias.

```
root@svrlinux:~#sudo apt full-upgrade -y
```

Luego, es fundamentaleliminar paquetes no necesarios. Después de una actualización, puedes eliminar archivos y paquetes obsoletos con el siguiente comando. Esto optimiza el espacio y mejora el rendimiento del sistema.

```
root@svrlinux:~#sudo apt autoremove -y
root@svrlinux:~#sudo apt autoclean
```

Las actualizaciones permiten:

- Corregir errores
- Parchear vulnerabilidades
- Actualizar funcionalidades
- Mejorar rendimiento

Ubuntu Server clasifica las actualizaciones en:

- Seguridad (security.ubuntu.com)
- Actualizaciones generales (archive.ubuntu.com)

También Linux Ubuntu permite configurar actualizaciones de seguridad automáticas. Para activarlas, simplemente ejecutamos el siguiente comando:

```
root@svrlinux:~#sudo apt install unattended-upgrades
root@svrlinux:~#sudo dpkg-reconfigure unattended-upgrades
```

Este último comando sirve para reconfigurar el paquete unattended-upgrades, que gestiona actualizaciones automáticas en Ubuntu Server. Este proceso permite definir si el sistema instalará actualizaciones de seguridad sin intervención del usuario. También puedes editar el archivo /etc/apt/apt.conf.d/50unattended-upgrades para afinar qué se actualiza automáticamente.

APT permite instalar *software* fácilmente desde los repositorios oficiales. Para instalar un paquete normalmente se utiliza sudo apt install nombre_paquete -y, por ejemplo para instalar nginx para un servidor web:

```
root@svrlinux:~#sudo apt install nginx -y
```

Podemos ver la información del paquete con el comando:

```
root@svrlinux:~#apt show nombre_paquete
```

Por ejemplo,

```
root@svrlinux:~#apt show apache2
```

También podemos buscar paquetes disponibles con:

```
root@svrlinux:~#apt search palabra_clave
```

Por ejemplo,

```
root@svrlinux:~#apt search firewall
```

Finalmente, hay queeliminar un paquete y hacerlo junto con sus archivos de configuración respectivamente:

```
root@svrlinux:~#sudo apt remove nombre_paquete -y
root@svrlinux:~#sudo apt purge nombre_paquete -y
```

Ubuntu se basa en repositorios, que son servidores que contienen miles de paquetes. Los repositorios están definidos en:

```
/etc/apt/sources.list
```

Repositorios importantes:

- *main*: *software* libre y oficial de Ubuntu
- *universe*: *software* libre mantenido por la comunidad
- *restricted*: *drivers* propietarios
- *multiverse*: *software* no libre

Muestra los paquetes que se pueden actualizar

```
root@svrlinux:~#apt list --upgradable
```

Muestra de dónde proviene un paquete

```
root@svrlinux:~#apt-cache policy paquete
```

Muestra información detallada sobre un paquete

```
root@svrlinux:~#apt show paquete
```

A continuación, creemos un *script* de mantenimiento automático para Ubuntu Server que puede programar para ejecutarse periódicamente. Este *script*:

- Actualiza los repositorios (apt update)
- Aplica actualizaciones disponibles (apt upgrade)
- Realiza limpieza (*autoremove*, *autoclean*)
- Genera un log con la fecha
- Envía notificación básica al administrador (opcional)

Primero creamos un archivo llamado mantenimiento.sh, y a continuación lo hacemos ejecutable con el comando chmod.

```
root@svrlinux:~#sudo nano /usr/local/bin/mantenimiento.sh
root@svrlinux:~#sudo chmod +x /usr/local/bin/mantenimiento.sh
```

Luego podemos programarlo en cron (una vez por semana, por ejemplo: domingos a las 3 AM):

```
root@svrlinux:~#sudo crontab -e
```

y agregamos la siguiente línea.

```
0 3 * * 0 /usr/local/bin/mantenimiento.sh
```

A continuación, el código fuente

```
#!/bin/bash

# === CONFIGURACIÓN ===
LOG_DIR="/var/log/mantenimiento"
LOG_FILE="$LOG_DIR/mantenimiento-$(date +%F).log"
ADMIN_EMAIL="tucorreo@ejemplo.com"

# Crear directorio de logs si no existe
mkdir -p "$LOG_DIR"

# Iniciar log
echo "=== Mantenimiento del sistema: $(date) ===" > "$LOG_FILE"

# Actualizar repositorios
echo -e "\n Actualizando lista de paquetes..." | tee -a "$LOG_FILE"
apt update >> "$LOG_FILE" 2>&1

# Aplicar actualizaciones disponibles
echo -e "\n Actualizando paquetes instalados..." | tee -a "$LOG_FILE"
apt upgrade -y >> "$LOG_FILE" 2>&1

# Limpieza de paquetes obsoletos
echo -e "\n  Limpiando paquetes no necesarios..." | tee -a "$LOG_FILE"
apt autoremove -y >> "$LOG_FILE" 2>&1
apt autoclean -y >> "$LOG_FILE" 2>&1
```

```
# Finalización
echo -e "\n Mantenimiento finalizado a las: $(date)" | tee -a "$LOG_FILE"

# (Opcional) Enviar log por correo si tienes configurado `mail`
# mail -s "Mantenimiento de $(hostname)" "$ADMIN_EMAIL" < "$LOG_FILE"
```

Finalmente, usted puede también integrar notificaciones por correo usando mailutils. Siempre el usuario que ejecuta el cron debe tener permisos sudo o ejecutar como *root*. Se pueden programar diferentes tareas como por ejemplo chequeos de espacio en disco, uso de CPU, estado de servicios, etc.

## Copias de seguridad y recuperación

Las copias de seguridad (*backups*) y su recuperación (*recovery*) son pilares esenciales para la resiliencia, continuidad operativa y seguridad de los datos de un sistema. Son medidas preventivas para evitar la pérdida total o parcial de información por:

- Fallos del *hardware* (discos duros, servidores)
- Errores humanos (borrado accidental)
- Ciberataques (*ransomware*)
- Desastres naturales (incendios, inundaciones)

Los objetivos principales se centran en minimizar la pérdida de datos, garantizando mecanismos eficaces para su protección y recuperación rápida en caso de fallos. Además, se busca asegurar la seguridad de la información crítica y las configuraciones esenciales del sistema, evitando posibles vulnerabilidades. Por último, es fundamental cumplir con las normativas y políticas de protección de datos vigentes, asegurando que todas las acciones y procedimientos estén alineados con los estándares de seguridad establecidos.

### Clasificación de copias de seguridad

**1. Por tipo de respaldo:**

| Tipo | Descripción |
|---|---|
| **Completa** | Copia de todos los archivos y datos. |
| **Incremental** | Solo copia archivos que han cambiado desde el último respaldo (completo o incremental). |
| **Diferencial** | Copia archivos cambiados desde el último respaldo completo. |

**2. Por ubicación:**

Local: en un disco interno, externo o red LAN.

Remota: en otro servidor (por SSH, FTP, rsync) o en la nube.

Híbrida: combinación de ambas.

A continuación, veamos algunas herramientas para este propósito. El siguiente comando rsync sincroniza directorios locales/remotos. Ideal para copias programadas o automatizadas.

```
root@svrlinux:~#rsync -avz /home/usuario/ /mnt/respaldo/
```

También tenemos el comando tar (archivos comprimidos), para respaldar configuraciones o carpetas completas.

```
root@svrlinux:~#tar -czvf backup-home-$(date +%F).tar.gz /home/usuario
```

Existen también otras herramientas como Borg, Duplicity y Restic que son herramientas avanzadas de respaldo diseñadas para garantizar la seguridad y eficiencia en la protección de datos. Todas ofrecen compresión para reducir el tamaño de los archivos almacenados, cifrado para proteger la información contra accesos no autorizados, versionado que permite mantener múltiples estados de un respaldo y recuperación precisa, y la posibilidad de realizar copias remotas, asegurando la integridad de los datos en ubicaciones externas. Estas características las convierten en soluciones robustas para la gestión de respaldos en entornos personales y empresariales.

Asimismo, podemos respaldar en la nube (*backups* externos) mediante Google Drive, Dropbox, Amazon s3, etc.

Para garantizar la seguridad y recuperación de datos en caso de fallos, es fundamental respaldar archivos de usuario ubicados en directorios como /home/ y /srv/, así como configuraciones críticas en /etc/, incluyendo archivos esenciales para servicios como nginx, Apache y ssh. También es importante realizar respaldos programados de bases de datos utilizando herramientas como mysqldump o pg_dump, asegurando la integridad de la información almacenada. Además, es recomendable guardar *scripts* personalizados y datos clave del sistema que puedan ser necesarios para restaurar funcionalidades específicas. Finalmente, si se utilizan contenedores o máquinas virtuales, es esencial incluirlos en la estrategia de respaldo para mantener la operatividad de los entornos virtualizados. Estos aspectos garantizan una recuperación eficiente y minimizan la pérdida de información crítica.

Tabla 7. Elementos que deben respaldarse y algunas herramientas

| **Categoría** | **Elementos a respaldar** | **Ejemplo de herramientas** |
|---|---|---|
| **Archivos de usuario** | /home/, /srv/ | rsync, Borg, Restic |
| **Configuraciones críticas** | /etc/, archivos de servicios (nginx, Apache, ssh) | tar, Ansible |
| **Bases de datos** | Dumps programados (mysqldump, pg_dump) | cron, PostgreSQL, MySQL |
| **Scripts y datos del sistema** | Scripts personalizados y archivos clave | Git, rsync |
| **Contenedores o máquinas virtuales** | Imágenes y volúmenes de contenedores | Docker Backup, Virtual Machine Snapshots |

La recuperación de datos es el proceso de restaurar información que ha sido perdida, dañada, eliminada accidentalmente o que se ha vuelto inaccesible en un dispositivo de almacenamiento. Esto puede ocurrir por diversas razones, como fallos de *hardware*, errores humanos, ataques de *malware* o corrupción del sistema de archivos.

Existen diferentes métodos para recuperar datos, dependiendo del tipo de pérdida:

- *Software* de recuperación: herramientas especializadas pueden restaurar archivos eliminados o dañados si aún existen en el almacenamiento.
- Recuperación desde copias de seguridad: si se ha realizado un respaldo previo, los datos pueden ser restaurados desde una copia externa.
- Técnicas avanzadas en laboratorio: en casos de fallos físicos graves, expertos pueden reparar discos duros o recuperar información directamente de los componentes dañados

La recuperación es el proceso inverso al *backup*: restaurar los archivos o el sistema completo desde las copias.

Ejemplos:

Para descomprimir y restaurar un .tar.gz

```
root@svrlinux:~#tar -xzvf backup.tar.gz -C /
```

Podemos restaurar con rsync:

```
root@svrlinux:~#rsync -av /mnt/respaldo/ /home/usuario/
```

y también podemos restaurar una base de datos con el comando siguiente:

```
root@svrlinux:~#mysql -u root -p basededatos < respaldo.sql
```

A continuación, veamos un ejemplo práctico desde terminal en Ubuntu Server para respaldar y restaurar una base de datos MySQL/MariaDB o PostgreSQL. **Es muy recomendable** cuando se tiene una base de datos "grande" realizar este procedimiento desde un terminal y no desde un sistema web como phpmyadmin, ya que la información se puede perder.

Primero para respaldar y crear el *backup*

```
mysqldump -u root -p nombre_basedatos > respaldo_basedatos.sql
```

- -u root: usuario de la base de datos
- -p: solicitará la contraseña
- nombre_basedatos: nombre de la base de datos que vas a respaldar
- respaldo_basedatos.sql: archivo de salida que contendrá todo el SQL

Luego para comprimir el respaldo

```
mysqldump -u root -p nombre_basedatos | gzip > respaldo_basedatos.sql.gz
```

Segundo para restaurar la base de datos de respaldo

Nos aseguramos de que la base de datos exista y luego restauramos desde el archivo .sql

```
mysql -u root -p -e "CREATE DATABASE nombre_basedatos;"
```

```
mysql -u root -p nombre_basedatos < respaldo_basedatos.sql
```

Si el respaldo está comprimido (.sql.gz), descomprime y pasa el contenido:

```
gunzip < respaldo_basedatos.sql.gz | mysql -u root -p nombre_basedatos
```

Ahora un ejemplo para Postgresql, para respaldar ejecutamos lo siguiente:

```
pg_dump -U postgres nombre_basedatos > respaldo_pg.sql
```

Primero creamos la base de datos si no existe

```
createdb -U postgres nombre_basedatos
```

y finalmente

```
psql -U postgres nombre_basedatos < respaldo_pg.sql
```

A continuación, creemos un *script* en Bash para hacer copias de seguridad automáticas de una base de datos MySQL/MariaDB. El respaldo incluirá la fecha actual en el nombre del archivo, estará comprimido y se guardará en un directorio específico. Primero, cree un archivo backup_mysql.sh, dele permisos de ejecución con el comando chmod +x /usr/local/bin/backup_mysql.sh, seguidamente puede automatizar con cron.

Código para realizar copias de seguridad de una base de datos.

```
#!/bin/bash
# Configuración
USER="root"
PASSWORD="tu_contraseña"
DATABASE="nombre_basedatos"
BACKUP_DIR="/var/backups/mysql"
FECHA=$(date +%F)
NOMBRE_ARCHIVO="${DATABASE}_${FECHA}.sql.gz"
# Crear directorio si no existe
mkdir -p "$BACKUP_DIR"
# Realizar el respaldo comprimido
mysqldump -u $USER -p$PASSWORD $DATABASE | gzip > "$BACKUP_DIR/$NOMBRE_ARCHIVO"
# Permisos de seguridad (opcional)
chmod 600 "$BACKUP_DIR/$NOMBRE_ARCHIVO"
# Mensaje de éxito
echo "Backup de la base de datos '$DATABASE' realizado con éxito: $BACKUP_DIR/$NOMBRE_ARCHIVO"
```

Para hacer el *backup* diariamente a las 2:00 am:

```
root@svrlinux:~#crontab -e
```

Y añade esta línea al final:

```
0 2 * * * /usr/local/bin/backup_mysql.sh
```

Luego puedes ejecutar el *script* en cualquier momento para probarlo:

```
root@svrlinux:~#/usr/local/bin/backup_mysql.sh
```

Queda como tarea al lector crear *script* que **elimine respaldos antiguos** automáticamente después de ciertos días.

## Buenas prácticas y recomendaciones finales

Para garantizar un sistema seguro, estable y eficiente, es fundamental aplicar buenas prácticas de mantenimiento y administración. Mantener actualizaciones frecuentes y aplicar parches de seguridad de manera oportuna ayuda a proteger contra vulnerabilidades. Asimismo, es esencial realizar respaldos periódicos siguiendo la regla 3-2-1, asegurando la disponibilidad de los datos en caso de fallos. La automatización y el monitoreo constante permiten detectar y solucionar problemas antes de que afecten el rendimiento. Además, es necesario evaluar cambios en entornos de pruebas antes de su implementación en producción, pues esto minimiza riesgos. La seguridad debe ser una prioridad, incluyendo el uso de cifrado en datos sensibles y el control de accesos. Implementar estos hábitos fortalece la integridad y confiabilidad del sistema a largo plazo.

## 1.Actualizaciones y seguridad

- Realiza actualizaciones frecuentes, al menos una vez por semana.
- Aplica parches de seguridad tan pronto como estén disponibles para proteger el sistema.

Antes de ejecutar apt upgrade, revisa los cambios con:

```
root@svrlinux:~#apt list –upgradable
```

- Usa herramientas como *screen* o tmux al actualizar remotamente para evitar interrupciones en la conexión.
- Prueba las actualizaciones en un entorno de *staging* antes de aplicarlas en producción, especialmente en servidores críticos.

## 2. Gestión de respaldos

- Realiza pruebas periódicas de restauración para verificar la integridad de los respaldos.
- Mantén varias versiones de *backups,* utilizando versionado para evitar pérdidas irreversibles.
- Aplica cifrado a copias remotas o datos sensibles para mejorar la seguridad.
- Automatiza y monitorea los respaldos para garantizar su correcta ejecución.
- Aplica la regla 3-2-1 para una estrategia de respaldo efectiva:
- 3 copias de los datos.
- 2 medios distintos (ejemplo: disco local + almacenamiento en la nube).
- 1 copia externa o remota para mayor protección.

# BIBLIOGRAFÍA


ALEGRE RAMOS, M. D. (2023). *Sistemas informáticos.* Madrid: Ediciones Paraninfo, S.A.

Aguilar, L. J. (2008). *Fundamentos de Programación.* Madrid: McGraw Hill.

BARDOT, Y. (2022). *Ubuntu Administración de un sistema Linux* (Segunda ed.). ENI.

Belmar, A. (2025). *Seguridad GNU/Linux: Curso práctico.* México: Ra-Ma.

EtherWAN. (Diciembre de 15 de 2022). *Implementing Quality of Service for Prioritizing Network Traffic*. Obtenido de https://www.etherwan.com/support/featured-articles/implementing-quality-service-prioritizing-network-traffic

Inc., D. (09 de 12 de 2024). *https://www.docker.com/*. Obtenido de https://www.docker.com/: https://docs.docker.com/get-started/docker-overview/

Jorba Esteve, J., & Suppi Boldrito, R. (2004). *Administración avanzada de GNU/Linux.* Barcelona.

Kane, S. P. (2018). *Docker: Up & Running.* . Sebastopol, CA: O'Reilly Media.

Laboratories, S. (s.f.). *Los sistemas de archivos en Linux*. Obtenido de UFS Explorer: https://www.ufsexplorer.com/es/articles/linux-file-systems/

Tanenbaum, A. S., & Bos, H. (2023). *Modern operating systems.* Hoboken, NJ.: PEARSON.

PUCE Ibarra

edi PUCE

PUBLÍCALO
y genera conocimiento que transforma

ISBN: 978-9978-77-810-4
9789978778104