Administración de Servidores Linux

Curso Profesional para Abacom

Diego Saavedra - Instructor Principal

Feb 2, 2026

Table of contents

1 Administración de Servidores Linux

1.1 Introducción

Este curso te prepara para instalar, operar y asegurar un servidor Linux con enfoque profesional: orden, seguridad y diagnóstico reproducible.

1.2 Qué vas a lograr

- Operar un servidor Ubuntu (paquetes, usuarios, permisos y servicios con systemd).
- Conectarte por SSH y diagnosticar problemas con logs, puertos y estado de servicios.
- Aplicar hardening básico: actualizaciones seguras, UFW y Fail2Ban.
- Desplegar servicios web con Nginx + SSL (Certbot).
- Ejecutar aplicaciones con Docker y Docker Compose.

1.3 Cómo usar este material

- Sigue las unidades en orden.
- Ejecuta los ejemplos y conserva evidencias (salidas, capturas y checklist).
- Completa los laboratorios (son la parte principal del curso).

1.4 Antes de empezar

- Preparación del entorno: Guía de configuración (SETUP)
- Diapositivas del curso y por unidad: Presentaciones (Reveal.js)

Selecciona la Unidad 1 en el menú lateral para comenzar.

Part I

Introducción al Curso

2 Acerca del Instructor

2.1 Diego Saavedra - Instructor Principal

Soy Diego Saavedra, especialista en desarrollo avanzado de software, investigación e infraestructura de sistemas. Mi experiencia abarca:

2.1.1 Experiencia Técnica

- **Backend:** Python (Django, Flask, FastAPI), Java, C#
- **DevOps & Infraestructura:** Linux Administration, Docker, Kubernetes, Cloud Services
- **Bases de Datos:** PostgreSQL, MySQL, MongoDB
- **Arquitectura:** Diseño de sistemas escalables, Clean Architecture, patrones de diseño
- **Liderazgo Ágil:** Metodologías Scrum, Kanban, gestión de equipos

2.1.2 Formación Académica

- **Doctorado en Ciencias de la Computación:** Enfocado en Inteligencia Artificial y Visión Artificial (detección de neurodivergencia)
- **Maestría en Ciencias de la Computación:** Deep Learning y Convolutional Neural Networks
- **Experiencia Académica:** Departamento de Ciencias de la Computación, Universidad de las Fuerzas Armadas ESPE

2.1.3 Experiencia en Educación

He sido docente de programación en institutos, universidades, bootcamps y corporativos, enseñando: - Desarrollo Web (JavaScript, Python, Java, C#) - Administración de Sistemas Linux - Arquitectura de Software - DevOps y Cloud - Inteligencia Artificial

2.2 Sobre Este Curso

En el **Curso de Administración de Servidores Linux para Abacom**, aprenderás:

- Instalación y configuración profesional de servidores Linux
- Administración avanzada de usuarios, permisos y seguridad
- Gestión de procesos, servicios y automatización
- Configuración de redes, firewall y SSH
- Despliegue de servicios empresariales (Apache, Nginx, DNS, FTP)
- Monitorización, logging y troubleshooting
- Mejores prácticas de la industria y seguridad

Este curso combina teoría sólida con laboratorios prácticos hands-on, preparándote para roles como Administrador de Sistemas, DevOps Engineer o especialista en Infraestructura.

2.2.1 Metodología

Mi enfoque educativo se basa en: - **Aprender Haciendo:** Laboratorios prácticos desde la primera sesión - **Desafíos Reales:** Casos de uso de empresas reales - **Mentoría Activa:** Apoyo personalizado y resolución de dudas - **Evaluación Continua:** Checklist de aceptación por laboratorio/práctica + feedback constructivo

¡Estoy emocionado de guiarte en el dominio de la administración de servidores Linux! Juntos construiremos las habilidades que Abacom necesita.

3 Licencia

Este material se entrega como contenido educativo para el curso **Administracion de Servidores Linux**.

3.1 Derechos de autor

El contenido (textos, material didactico, ejercicios y evaluaciones) esta protegido por derechos de autor.

3.2 Uso permitido

- Uso personal y educativo.
- Modificar ejemplos para aprendizaje.
- Citar fragmentos con atribucion.

3.3 Restricciones

- No redistribuir el contenido completo (o partes sustanciales) sin autorizacion.
- No vender el contenido ni usarlo con fines comerciales sin permiso.
- No presentar el contenido como propio.

3.4 Atribucion

Si compartes fragmentos o capturas del material, incluye atribucion al repositorio y al autor.

4 Guía de Configuración del Entorno

Listing 4.1 `QUARTO-TITLE-BLOCK`

Opciones recomendadas para seguir el curso

5 Guía de Configuración del Entorno

Tienes **4 opciones** para ejecutar Linux y seguir este curso. Elige la que mejor se adapte a tu máquina.

> **Consejo:** Si no sabes cuál elegir, ir a la sección Recomendaciones al final.

5.1 Opción 1: Windows + WSL2 Mejor para Desarrolladores en Windows

¿Qué es WSL2?

Windows Subsystem for Linux 2 te permite ejecutar un kernel Linux completo directamente en Windows. Es como tener "Linux como programa" en tu PC.

¿Por qué WSL2? - Más rápido que máquinas virtuales - Menos requisitos de almacenamiento - Integración perfecta con Windows - Excelente para Docker

Requisitos Mínimos:

```
Windows 10 Build 19041+ o Windows 11
Procesador: Cualquiera moderno (Intel VT-x o AMD-V)
RAM: 4 GB (8 GB recomendado para este curso)
Disco: 20 GB disponibles
```

5.1.1 Instalación (3 pasos simples)

5.1.1.1 Paso 1: Instalar WSL2

Abre **PowerShell como Administrador** (clic derecho → "Ejecutar como Administrador")

Después, **reinicia tu PC**.

Listing 5.1 POWERSHELL

```
# Instala WSL2 con Ubuntu 22.04
wsl --install

# Si tienes WSL1 antiguo:
wsl --install -d Ubuntu-22.04
```

5.1.1.2 Paso 2: Primera conexión

Después del reinicio, WSL te pedirá crear usuario y contraseña:

```
Enter new UNIX username: tu_usuario
New password: [escribe algo que recuerdes]
Retype new password: [confirma]
```

5.1.1.3 Paso 3: Verifica que funciona

Listing 5.2 POWERSHELL

```
# Abre PowerShell de nuevo y verifica
wsl --list --verbose

# Deberías ver algo como:
# NAME            STATE           VERSION
# Ubuntu-22.04    Running         2

# O simplemente entra a WSL
wsl
```

5.1.2 Usar WSL2 para el Curso

Opción A: PowerShell (Windows Terminal)

Listing 5.3 POWERSHELL

```
# Desde PowerShell o Windows Terminal
wsl

# Ahora estás en bash de Linux
whoami
pwd
ls
```

Opción B: Acceso directo desde menú

Presiona `Win` y busca "Ubuntu" → click → abre terminal Linux directo

Opción C: Windows Terminal (recomendado)

Listing 5.4 POWERSHELL

```
# Descarga gratis desde Microsoft Store
# O: https://www.microsoft.com/en-us/p/windows-terminal/

# Luego tienes pestaña "Ubuntu" en Windows Terminal
```

5.1.3 Primeras configuraciones útiles

Listing 5.5 BASH

```
# Actualizar paquetes (SIEMPRE haz esto primero)
sudo apt update
sudo apt upgrade -y

# Instalar herramientas de desarrollo
sudo apt install -y \
  build-essential \
  git \
  vim \
  nano \
  curl \
  wget \
  net-tools \
  htop \
  tree

# Verificar que todo está
git --version
gcc --version
python3 --version
```

5.1.4 Acceso a archivos Windows desde WSL2

Tu unidad `C:` está en `/mnt/c`:

5.1.5 Docker en WSL2

Listing 5.6 BASH

```bash
# Ver contenidos de Documentos
ls /mnt/c/Users/TuUsuario/Documents

# Crear carpeta de trabajo
mkdir -p /mnt/c/Users/TuUsuario/CursoLinux
cd /mnt/c/Users/TuUsuario/CursoLinux

# Crear archivos aquí = los ves en Windows
touch archivo.txt
# Lo ves en Explorador de archivos
```

Listing 5.7 BASH

```bash
# Instala Docker Desktop para Windows
# https://www.docker.com/products/docker-desktop

# En WSL2 luego puedes usar:
docker --version
docker run -it ubuntu:22.04 /bin/bash
```

5.1.6 Ventajas y Limitaciones

Ventajas: - Linux completo sin virtualización pesada - Perfecto para desarrollo en Windows - Excelente para Docker - No consume mucho disco

Limitaciones: - NO es ideal para aprender instalación del SO - Interfaz gráfica muy limitada - Es kernel Windows, no Linux puro

---

5.2 Opción 2: Linux Nativo La Mejor Opción para Aprender Serio

¿Qué significa?

Tu computadora ya tiene Linux instalado como sistema operativo (ya sea como único SO o dual-boot con Windows/macOS).

¿Por qué es lo mejor? - Acceso completo y real al sistema - Máximo rendimiento - Entorno profesional auténtico - Aprendes administración real, no simulada

5.2.1 Requisitos:

```
Máquina con Linux ya instalado (Ubuntu, Fedora, Debian, Arch, etc.)
RAM: 2 GB mínimo (4 GB recomendado)
Terminal de comandos accesible
```

5.2.2 Verificar tu sistema

Abre una terminal (Ctrl+Alt+T típicamente, o busca "Terminal" en menú)

Listing 5.8 BASH

```bash
# Ver distribución y versión completa
cat /etc/os-release

# Ejemplo de salida:
# NAME="Ubuntu"
# VERSION="22.04.1 LTS"
# ID=ubuntu
```

Listing 5.9 BASH

```bash
# Ver solo la versión
lsb_release -a

# Ver versión del kernel
uname -r
```

5.2.3 Mantener tu sistema actualizado

Para Ubuntu/Debian:

Listing 5.10 BASH

```bash
sudo apt update     # Actualiza lista de paquetes
sudo apt upgrade    # Instala actualizaciones
sudo apt autoremove # Limpia paquetes viejos
```

Para Fedora/RHEL:

Para Arch Linux:

Listing 5.11 BASH

```
sudo dnf upgrade
```

Listing 5.12 BASH

```
sudo pacman -Syu
```

5.2.4 Instalar herramientas necesarias para el curso

Ubuntu/Debian:

Fedora/RHEL:

Arch Linux:

5.2.5 Para Practicar Instalación de Linux

Si YA tienes Linux nativo, pero quieres practicar la instalación, tienes 3 opciones:

Opción A: Máquina Virtual (Recomendado)

Opción B: Partición Dual Boot

Opción C: Docker

5.2.6 Ventajas y Limitaciones

Ventajas: - Acceso completo y real al sistema - Mejor rendimiento posible - Ideal para administración de sistemas - Aprendizaje profesional auténtico

Limitaciones: - Requiere tener Linux ya instalado - Menor protección al experimentar (podrías afectar tu SO principal)

5.3 Opción 3: macOS + Máquinas Virtuales

¿Por qué máquinas virtuales en Mac? macOS es compatible con Unix, pero para aprender Linux "puro" necesitas una VM con Linux instalado.

5.3.1 Parte 1: macOS Nativo

5.3.1.1 Acceso a terminal:

Listing 5.13 BASH

```
sudo apt install -y \
  build-essential \
  git \
  vim \
  nano \
  curl \
  wget \
  net-tools \
  htop \
  tree \
  openssh-client \
  openssh-server
```

Listing 5.14 BASH

```
sudo dnf install -y \
  gcc \
  gcc-c++ \
  make \
  git \
  vim \
  nano \
  curl \
  wget \
  net-tools \
  htop \
  tree \
  openssh-clients \
  openssh-server
```

5.3.1.2 Instala Homebrew (gestor de paquetes para Mac)

5.3.2 Parte 2: Máquina virtual para Linux

Opciones según tu procesador:

5.3.2.1 Si tienes Mac Intel:

Opción A: VirtualBox (gratuito)

Opción B: Parallels Desktop (pagado, ~$100/año)

`Descarga desde: https://www.parallels.com/`

Listing 5.15 BASH

```
sudo pacman -S \
  base-devel \
  git \
  vim \
  nano \
  curl \
  wget \
  net-tools \
  htop \
  tree \
  openssh
```

Listing 5.16 BASH

```
# Instala VirtualBox
sudo apt install virtualbox

# O descarga desde: https://www.virtualbox.org/

# Luego:
# 1. Descarga ISO de Ubuntu
# 2. Crea máquina virtual vacía
# 3. Practica la instalación sin riesgo
```

5.3.2.2 Si tienes Mac Apple Silicon (M1/M2/M3):

Opción A: UTM (gratuito) Recomendado

Opción B: Parallels Desktop (pagado)

```
Mejor rendimiento en Apple Silicon
Descarga desde: https://www.parallels.com/
```

Opción C: VMware Fusion (pagado)

```
Descarga desde: https://www.vmware.com/products/fusion/
```

5.3.3 Crear máquina virtual en Mac:

5.3.3.1 Con UTM (Para Apple Silicon):

```
1. Descarga UTM desde https://mac.getutm.app/
2. Abre UTM
```

Listing 5.17 BASH

```
# Crea partición adicional en tu disco
# Instala otra distribución Linux en esa partición
# Elige cuál SO usar al arrancar

#  REQUIERE CUIDADO: puedes perder datos
# Haz backup ANTES de intentarlo
```

Listing 5.18 BASH

```
# Crea contenedores Linux
docker run -it ubuntu:22.04 /bin/bash

# Dentro del contenedor tienes un "Linux completo"
# Perfecto para aprender comandos sin virtualización
```

```
3. Click "+" → Create a new virtual machine
4. Selecciona: Emulate
5. Operating System: Linux
6. Architecture: ARM64 (importante para M1/M2/M3)
7. Descarga imagen Ubuntu 22.04 ARM64
8. Completa el wizard
9. Inicia VM
```

5.3.3.2 Con VirtualBox (Para Intel):

```
1. Instala VirtualBox
2. File → New
3. Name: Ubuntu 22.04
4. Type: Linux, Version: Ubuntu 64-bit
5. RAM: 4096 MB
6. Crea disco virtual: 50 GB
7. Configura ISO de Ubuntu
8. Inicia y sigue instalador
```

5.3.4 Ventajas:

macOS + Linux en la misma máquina
Protección completa
Aprender Linux puro y macOS
Excelente para desarrollo

Listing 5.19 BASH

```
# Abre Terminal.app o iTerm2
# O presiona Cmd + Espacio y escribe "Terminal"

# macOS es UNIX, así que muchos comandos Linux funcionan
uname -a
pwd
ls -la
```

Listing 5.20 BASH

```
# En Terminal:
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/instal

# Luego puedes instalar herramientas
brew install git vim curl
```

5.3.5 Limitaciones:

Requiere máquina potente (RAM, disco)
Algunas opciones son pagadas

5.4 Opción 4: Docker (Contenedores) Para aprender rápido

¿Qué es Docker? Docker crea "máquinas Linux ligeras" llamadas contenedores. Son como máquinas virtuales pero mucho más rápidas.

5.4.1 Instalación:

5.4.1.1 En Windows:

5.4.1.2 En macOS:

5.4.1.3 En Linux:

Listing 5.21 BASH

```
brew install virtualbox

# O descarga desde: https://www.virtualbox.org/
```

Listing 5.22 BASH

```
brew install utm

# O descarga desde: https://mac.getutm.app/
```

5.4.2 Primeros pasos con Docker:

5.4.2.1 Ejecuta un contenedor Ubuntu:

5.4.2.2 Crear tu propia imagen (Dockerfile):

5.4.2.3 Trabajar con archivos desde Docker:

5.4.3 Ventajas de Docker:

Ultraligero y rápido
Fácil de crear y destruir
Reproducible en cualquier máquina
Perfecto para aprender comandos
Ideal para probar servicios

5.4.4 Limitaciones:

No es una máquina “completa”
No puedes instalar bootloader/GRUB
Kernel compartido del host

5.5 Tabla de Comparación

Listing 5.23 POWERSHELL

```powershell
# Descarga Docker Desktop
# https://www.docker.com/products/docker-desktop

# Instala siguiendo el wizard
# Requiere WSL2 habilitado

# Verifica instalación
docker --version
```

Listing 5.24 BASH

```bash
# Opción 1: Homebrew
brew install docker

# Opción 2: Descarga Docker Desktop
# https://www.docker.com/products/docker-desktop

# Verifica
docker --version
```

Aspecto	WSL2	Linux Nativo	macOS + VM	Docker
Facilidad	Muy fácil	Fácil	Fácil	Muy fácil
Rendimiento	Excelente	Excelente	Bueno	Excelente
SO Completo	Sí	Sí	Sí	Parcial
Aprender instalación	No	Sí*	Sí	No
Aprender comandos	Sí	Sí	Sí	Sí
Precio	Gratis	Gratis	Gratis-$100	Gratis
Nivel de control	Medio	Máximo	Alto	Bajo
Ideal para	Dev	Admin	Principiantes	Aprendizaje rápido

*Con máquina virtual adicional

5.6 Recomendaciones

5.6.1 Si tienes Windows:

→ **WSL2** para desarrollo rápido
→ **VirtualBox** para aprender instalación
→ **Docker** para probar servicios

Listing 5.25 BASH

```
# Ubuntu/Debian
sudo apt install docker.io

# Fedora
sudo dnf install docker

# Arch
sudo pacman -S docker

# Inicia servicio
sudo systemctl start docker

# Verifica
docker --version
```

Listing 5.26 BASH

```
# Descarga e inicia
docker run -it ubuntu:22.04 /bin/bash

# Ahora estás dentro de Ubuntu
# Prueba comandos
apt update
apt install vim curl

# Sale con: exit
```

5.6.2 Si tienes Linux:

→ **Linux nativo** para aprender en serio
→ **Docker** para laboratorios rápidos
→ **Máquina virtual** para instalar otra distro

5.6.3 Si tienes Mac Intel:

→ **macOS nativo** para desarrollo
→ **VirtualBox** para Linux (gratuito)
→ **Parallels** para mejor rendimiento (pagado)

5.6.4 Si tienes Mac Apple Silicon (M1/M2/M3):

→ **UTM** para máquinas virtuales (gratuito)
→ **Parallels** para mejor rendimiento (pagado)

Listing 5.27 DOCKERFILE

```
# Crear archivo: Dockerfile
FROM ubuntu:22.04

# Actualiza paquetes
RUN apt-get update && apt-get install -y \
    build-essential \
    git \
    vim \
    nano \
    curl \
    net-tools \
    htop \
    tree

# Crea directorio de trabajo
WORKDIR /workspace

# Comando por defecto
CMD ["/bin/bash"]
```

Listing 5.28 BASH

```
# Construir imagen
docker build -t mi-ubuntu-curso .

# Ejecutar contenedor
docker run -it mi-ubuntu-curso

# Dentro del contenedor tienes todas las herramientas
```

→ **Docker** para contenedores rápidos

5.6.5 Para máximo aprendizaje (ideal):

→ **Linux nativo** (aprendes administración real)
→ + **Docker** (para servicios y laboratorios)
→ + **Máquina virtual** (para instalar otras distros)

5.7 Verifica tu entorno

Ejecuta estos comandos en tu entorno elegido:

Listing 5.29 BASH

```
# Montar carpeta local en contenedor
docker run -it -v /path/to/local:/workspace ubuntu:22.04 /bin/bash

# Ahora /workspace en el contenedor = tu carpeta local
# Los cambios se sincronizan automáticamente
```

5.8 Documentación Oficial

- **WSL2**: https://learn.microsoft.com/en-us/windows/wsl/
- **Docker**: https://docs.docker.com/
- **macOS Terminal**: https://support.apple.com/en-us/HT201236
- **Linux Command Line**: https://linux.die.net/man/
- **Ubuntu**: https://ubuntu.com/

5.9 ¿Tienes problemas?

Si tu entorno no funciona, sigue estos pasos:

1. **Verifica versiones:**
2. **Busca en Google** el error específico
3. **Consulta documentación oficial** de tu SO
4. **Contacta al instructor** con detalles específicos

¡Listo para comenzar el curso!

Selecciona tu entorno arriba y sigue la guía paso a paso.

Listing 5.30 BASH

```bash
# 1. Ver información del sistema
uname -a

# 2. Ver shell
echo $SHELL

# 3. Ver user
whoami

# 4. Ver PATH
echo $PATH

# 5. Instalar git si no lo tienes
# Windows+WSL2: apt install git
# macOS: brew install git
# Linux: apt install git (o dnf/pacman)

git --version

# 6. Clonar este repositorio
git clone https://github.com/tu-repo/curso-linux.git
cd curso-linux

# ¡Si todo funciona, estás listo para el curso!
```

Listing 5.31 BASH

```bash
uname -a          # Sistema
bash --version    # Shell
git --version     # Git
```

6 Glosario del Curso

Listing 6.1 `QUARTO-TITLE-BLOCK`

7 Glosario del Curso

7.1 Introduccion

Este glosario define los terminos clave que vas a ver a lo largo del curso (Linux, redes, seguridad, servicios, Docker y Bash). La idea es que puedas volver aqui cuando un concepto aparezca en una unidad, practica o laboratorio.

7.2 Como usar este glosario

- Si ves un comando o concepto nuevo, busca su nombre aqui.
- Si estas en un laboratorio, prioriza: seguridad, permisos, red, servicios, logs.
- Si estas en Bash, prioriza: variables, quoting, pipes, exit codes.

7.3 Sistema Operativo y Linux

Termino	Definicion practica
Sistema operativo (SO)	Software base que administra CPU, memoria, disco, red y procesos.
Kernel	Nucleo del SO; gestiona hardware y recursos. En Linux, el kernel es "Linux".
Userland	Herramientas y librerias que usas diariamente (coreutils, systemd, etc.).
Distribucion (distro)	Paquete completo: kernel + userland + instalador + repositorios (Ubuntu, Debian, etc.).
LTS	Long Term Support; soporte extendido con parches de seguridad por anos.
Terminal	Programa donde escribes comandos (Terminal.app, GNOME Terminal, Windows Terminal).
Shell	Interprete que lee tus comandos (bash, zsh).
CLI	Interfaz de linea de comandos (trabajo por texto).
Prompt	Texto que indica que la shell espera un comando (ej: `$`, `PS>`).

Termino	Definicion practica
Ruta (path)	Direccion de un archivo/directorio (ej: `/etc/ssh/sshd_config`).
Home	Directorio personal del usuario (ej: `/home/diego`).
Root (/)	Directorio raiz del sistema de archivos (no es el usuario root).
Usuario root	Cuenta con privilegios totales en Linux.
sudo	Ejecuta un comando con privilegios elevados (segun politica).
Paquete	Unidad instalable (binarios/configs) desde repositorios.
Repositorio	Fuente de paquetes (servidor/indice) para instalar/actualizar software.
apt	Gestor de paquetes en Debian/Ubuntu (instalar/actualizar/remover).
dpkg	Herramienta base de paquetes .deb (bajo apt).

7.4 Archivos, Permisos y Propiedad

Termino	Definicion practica
Permisos	Reglas `rwx` para usuario/grupo/otros.
Propietario (owner)	Usuario dueno del archivo/directorio.
Grupo	Conjunto de usuarios para permisos compartidos.
chmod	Cambia permisos (ej: `chmod 640 archivo`).
chown	Cambia propietario y/o grupo (ej: `chown root:root archivo`).
umask	Mascara de permisos por defecto al crear archivos/directorios.
Ejecutable	Archivo con permiso `x` (puede ejecutarse como programa/script).
PATH	Variable de entorno con rutas donde se buscan ejecutables.
Glob/wildcards	Patron de archivos (ej: `*.log`, `nginx*.conf`).
Symlink	Enlace simbolico a otro path (tipo “atajo”).
Mount (montaje)	Conectar un dispositivo/particion a un directorio (ej: montar `/dev/sdb1` en `/mnt`).
Filesystem	Estructura de almacenamiento (ext4, xfs).

7.5 Procesos, Servicios y systemd

Termino	Definicion practica
Proceso	Programa en ejecucion con PID.
PID	Identificador numerico de proceso.
Señal (signal)	Mensaje para un proceso (ej: terminar, recargar).
Daemon	Proceso en segundo plano (ej: sshd, nginx).
systemd	Sistema de init en muchas distros; gestiona servicios, logs y mas.
Unidad (unit)	Objeto gestionado por systemd (service, timer, socket).
systemctl	CLI para administrar unidades systemd.
Servicio (service)	Definicion de un daemon en systemd (start/stop/enable).
enable/disable	Configura si un servicio inicia al boot.
start/stop/restart	Control inmediato del servicio en la sesion actual.
reload	Recarga configuracion sin reiniciar (si el servicio lo soporta).
journal	Log central de systemd (journald).
journalctl	CLI para consultar logs del journal.

7.6 Logs y Diagnostico

Termino	Definicion practica
Log	Registro de eventos: errores, accesos, warnings, etc.
syslog	Estilo tradicional de logs (ej: `/var/log/syslog`).
stdout/stderr	Salida normal vs salida de error de un comando.
exit code	Codigo de salida: `0` exito; `>0` error (convencion).
Observabilidad	Capacidad de entender que pasa: logs, metricas, trazas.
Causa raiz	Motivo real del problema (no solo el sintoma).

7.7 Redes (lo minimo para operar)

Termino	Definicion practica
IP	Direccion de red (IPv4/IPv6).
CIDR	Notacion de red (ej: `192.168.1.0/24`).
Gateway	Ruta de salida por defecto (“puerta” hacia otras redes).
DNS	Servicio que resuelve nombres a IP (ej: `example.com` -> `93.184.216.34`).
Puerto	Numero logico que identifica un servicio (ej: 22 SSH, 80 HTTP).
TCP/UDP	Protocolos de transporte; TCP es orientado a conexion, UDP no.
Socket	Endpoint de comunicacion (IP:puerto).
Firewall	Filtra trafico entrante/saliente segun reglas.
UFW	Firewall “facil” en Ubuntu (frontend de iptables/nftables).
SSH	Protocolo para acceso remoto seguro (shell remota).
sshd	Servicio de SSH en el servidor.
Clave SSH	Par publico/privado para autenticacion sin password.
known_hosts	Archivo local que guarda huellas de hosts SSH conocidos.
Bastion/jump host	Servidor intermedio para acceder a redes privadas.

7.8 Seguridad Basica

Termino	Definicion practica
Principio de menor privilegio	Dar solo los permisos necesarios, por el tiempo necesario.
Hardening	Endurecer configuracion para reducir superficie de ataque.
Parche (patch)	Actualizacion para corregir bugs o vulnerabilidades.
CVE	Identificador publico de vulnerabilidades.
Fail2ban	Bloquea intentos repetidos (ej: fuerza bruta a SSH).

Termino	Definicion practica
MFA	Autenticacion multifactor (no siempre aplica a SSH basico).

7.9 Docker y Contenedores

Termino	Definicion practica
Contenedor	Proceso aislado que empaqueta app + dependencias, comparte kernel.
Imagen	Plantilla inmutable para crear contenedores.
Dockerfile	Receta para construir una imagen.
Docker Compose	Define y ejecuta stacks multi-servicio (archivo `compose.yml`).
Volumen	Persistencia de datos manejada por Docker.
Bind mount	Montaje directo de un path del host dentro del contenedor.
Registry	Repositorio de imagenes (Docker Hub, GHCR, etc.).
Tag	Etiqueta de version de imagen (ej: `ubuntu:22.04`).
Puerto publicado	Mapea un puerto del host a un puerto del contenedor.

7.10 Bash y Scripting

Termino	Definicion practica
Script	Archivo ejecutable con comandos automatizados.
Shebang	Primera linea `#!/bin/bash`; define con que interprete ejecutar.
Variable	Nombre que guarda un valor (ej: `NOMBRE="Diego"`).
Variable de entorno	Variable exportada a procesos hijos (ej: `export PATH=...`).
Quoting	Uso de comillas para controlar expansion y espacios (`"` vs `'`).

Termino	Definicion practica
Expansion	Reemplazos automaticos: variables, globs, command substitution.
Command substitution	Ejecuta comando y usa su salida: `$(date)`.
Pipe	Conecta stdout de un comando al stdin de otro: `cmd1 | cmd2`.
Redireccion	Envia stdout/stderr a archivos o a otros streams (`>`, `2>`, `&>`).
Subshell	Shell hija (ej: `(cd /tmp && ls)`), no afecta el shell padre.
Funcion	Bloque reutilizable de codigo (ej: `log() { ... }`).
Argumento	Parametro pasado al script (`$1`, `$2`, `$@`).
getopts	Parser de flags estilo `-h -f archivo` en Bash.
set -euo pipefail	Modo estricto (fallar temprano) para scripts mas seguros.
trap	Ejecuta limpieza al salir o al recibir señales.

7.11 Ejemplos Practicos Multi-SO

7.11.1 Linux

Listing 7.1 BASH

```
$ bash --version                                                    ①
GNU bash, version 5.2.15(1)-release (x86_64-pc-linux-gnu)
Copyright (C) 2022 Free Software Foundation, Inc.
```

① **bash –version** muestra la version instalada de Bash (util para compatibilidad)

7.11.2 macOS

Listing 7.2 BASH

```
$ bash --version                                                    ①
GNU bash, version 3.2.57(1)-release (x86_64-apple-darwin22)
Copyright (C) 2007 Free Software Foundation, Inc.
```

① **bash –version** muestra la version instalada; en macOS suele ser 3.2.x por licencia

7.11.3 Windows

Listing 7.3 POWERSHELL

```
PS> Get-Help Get-ChildItem                                                    ①
NAME
    Get-ChildItem
```

① **Get-Help Get-ChildItem** muestra ayuda integrada de PowerShell (equivalente a consultar manual)

7.12 Mejores Practicas

- Aprende a leer `man`/help antes de copiar comandos.
- Entiende `stdout/stderr` y exit codes; ahi esta el 80% del troubleshooting.
- Usa el glosario como checklist: permisos + red + servicio + logs.

7.13 Resumen

Este glosario es tu referencia rapida para hablar el mismo idioma que el sistema (y que tu equipo) cuando operas servidores Linux.

7.14 Referencias

- https://man7.org/linux/man-pages/
- https://www.gnu.org/software/bash/manual/
- https://revealjs.com/

Part II

Unidad 1: Introducción a Linux

8 Unidad 1: Introducción a Sistemas Operativos

Listing 8.1 QUARTO-TITLE-BLOCK

9 Introducción a Sistemas Operativos

9.1 Introducción

Un **Sistema Operativo** es el software fundamental que controla todos los recursos de una computadora y gestiona las interacciones entre el usuario, aplicaciones y hardware. En Abacom, entender profundamente qué es un SO y cómo funciona es crítico para administrar servidores, resolver problemas de infraestructura y tomar decisiones arquitectónicas correctas.

Tiempo de lectura: ~20 minutos
Nivel: Principiante
Requisitos previos: Conceptos básicos de computadoras

9.2 En este tema aprenderás

En esta unidad cubriremos:

1. **¿Qué es un Sistema Operativo?** - Definición, rol y responsabilidades
2. **Componentes Principales** - Kernel, shell, drivers y utilidades
3. **Funciones Esenciales** - Gestión de recursos, procesos y memoria
4. **Diferencias entre SOs** - Comparativa entre Windows, macOS y Linux
5. **Por qué existe Linux** - Historia, filosofía y ventajas

9.3 ¿Qué es un Sistema Operativo?

Definición clara:

Un Sistema Operativo (SO) es un programa especial que actúa como **intermediario entre el usuario y el hardware**. Su función principal es gestionar todos los recursos de la computadora (memoria, procesador, disco duro, dispositivos periféricos) y permitir que otros programas funcionen correctamente (*POSIX.1-2017* 2017).

¿Por qué importa?

- **Impacto directo:** Sin un SO, no podrías ejecutar ningún programa. El SO es lo que permite que todo funcione.
- **En Abacom:** Necesitamos entender cómo el SO gestiona servidores para optimizar rendimiento y resolver problemas.
- **Ventaja profesional:** Un administrador que entiende el SO puede diagnosticar 80% de problemas sin herramientas externas.

9.4 Kernel vs Shell

Definición clara:

Aunque a menudo se confunden, son dos cosas diferentes:

- **Kernel:** El núcleo del SO. Controla directamente el hardware y gestiona procesos, memoria y dispositivos.
- **Shell:** La "interfaz" entre el usuario y el kernel. Es lo que ves en terminal - interpreta tus comandos y le dice al kernel qué hacer.

Punto importante

El kernel trabaja en "modo privilegiado" (ring 0), lo que significa tiene acceso total al hardware. Los programas normales corren en "modo usuario" (ring 3) con permisos limitados. Esta separación es crítica para la seguridad.

9.5 Ejemplos Prácticos

9.5.1 Ejemplo 1: Verifica tu SO

9.5.1.1 Linux

Explicación:

- **uname -a**: Muestra nombre del kernel, versión y arquitectura
- En Linux: Kernel es "Linux 5.15.0" en arquitectura "x86_64" (64 bits)
- **Resultado**: Información técnica para administración de servidores

Listing 9.1 BASH

```
# Ver información del SO en Linux
$ uname -a
Linux servidor-abacom 5.15.0-89-generic #99-Ubuntu SMP Mon Oct 5 09:29:34 UTC 2024 x86_64

# Información adicional
$ cat /etc/os-release | grep PRETTY_NAME
PRETTY_NAME="Ubuntu 22.04.4 LTS"
```

Listing 9.2 BASH

```
# Ver información del SO en macOS
$ uname -a
Darwin macbook-pro 23.3.0 Darwin Kernel Version 23.3.0 arm64

# Versión del sistema macOS
$ sw_vers
ProductName:    macOS
ProductVersion: 14.2.1
BuildVersion:   23C71
```

9.5.1.2 macOS

Explicación:

- **uname -a**: Muestra "Darwin" (kernel de macOS) en arquitectura arm64 (Apple Silicon) o x86_64 (Intel)
- **sw_vers**: Muestra versión específica de macOS instalado
- **Resultado**: macOS es basado en Unix, compatible con comandos Linux

9.5.1.3 Windows

Listing 9.3 POWERSHELL

```
# Ver información del SO en Windows (PowerShell)
PS> systeminfo | findstr /B /C:"OS Name" /C:"OS Version" /C:"System Type"
OS Name:                   Microsoft Windows 11 Professional
OS Version:                10.0.22621 Build 22621
System Type:               x64-based PC

# Alternativa más detallada
PS> Get-ComputerInfo -Property osname, osversion, systemskufamily
```

Explicación:

- **systeminfo**: Muestra información detallada del sistema
- **findstr**: Filtra solo líneas específicas (similar a grep en Linux)
- **Resultado**: Windows usa arquitectura x64 (equivalente a x86_64 en Linux)

9.5.2 Ejemplo 2: Ver procesos del SO

9.5.2.1 Linux

Listing 9.4 BASH

```
# Listar procesos en ejecución en Linux
$ ps aux | head -10
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1  10960  5216 ?        Ss   10:23   0:01 /sbin/init
root         2  0.0  0.0      0     0 ?        S    10:23   0:00 [kthreadd]
systemd+   560  0.0  0.2  89456  3564 ?        Ss   10:23   0:00 /lib/systemd/systemd-res

# Ver solo procesos de tu usuario
$ ps ux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
diego     5432  0.0  0.5 234567 8912 pts/0    S+   14:35   0:00 bash
```

Cómo funciona:

- **ps aux**: Lista todos los procesos del sistema con detalles completos
- **PID**: Identificador único de proceso
- **%CPU / %MEM**: Porcentaje de CPU y memoria que usa
- **STAT**: Estado del proceso (Ss = running, S = sleeping)

9.5.2.2 macOS

Listing 9.5 BASH

```
# Listar procesos en macOS
$ ps aux | head -10
USER               PID  %CPU %MEM      VSZ    RSS   TT  STAT STARTED      TIME COMMAND
root                 1   0.0  0.1  4348784  16564   ??  Ss   Wed01PM   0:32.54 /sbin/laun
root                46   0.0  0.0  4484408   5232   ??  Ss   Wed01PM   0:16.32 /usr/sbin/
_windowserver      108   1.5  1.2  5672744 201456   ??  Ss   Wed01PM  15:24.32 /System/Li

# Ver procesos más legiblemente
$ top -l 1 -n 10
```

Cómo funciona:

- Similar a Linux pero con columnas ligeramente diferentes
- **STARTED**: Cuándo se inició el proceso
- **WindowServer**: Proceso especial de macOS para gráficos

9.5.2.3 Windows

Listing 9.6 POWERSHELL

```
# Ver procesos en Windows (PowerShell)
PS> Get-Process | Select-Object Name, Id, CPU, Memory | Sort-Object Memory -Descending |

Name                Id  CPU      Memory
----                --  ---      ------
svchost           1234   1.5   523264
explorer          5678   2.3   456789
chrome            9012   8.5   1234567
python            3456   0.1   234567

# Alternativa: usar tasklist
PS> tasklist /V | findstr "python"
```

Cómo funciona:

- **Get-Process**: Obtiene lista de procesos activos
- **Id**: Identificador único (equivalente a PID en Linux)
- **Memory**: Uso de memoria en bytes
- **tasklist**: Comando clásico de Windows para listar procesos

---

9.5.3 Ejemplo 3: Caso Real Abacom

9.5.3.1 Linux (Servidor de Producción)

Por qué lo usamos en Abacom:

- Monitorear memoria disponible para aplicaciones críticas
- Detectar si el servidor está bajo carga (load average)
- Uptime de 247 días = Excelente estabilidad
- Planificar upgrades de hardware si es necesario

Listing 9.7 BASH

```
# En un servidor Linux de Abacom, verificar recursos del sistema
$ free -h
               total        used        free      shared  buff/cache   available
Mem:            15.6G       3.2G       9.1G       256M       3.3G      12.0G
Swap:            4.0G       0.0B       4.0G

# Ver carga del servidor
$ uptime
 14:35:42 up 247 days, 3:42, 2 users, load average: 0.85, 0.92, 0.88

# Ver procesos principales de la aplicación
$ ps aux | grep "python\|node\|java"
```

Listing 9.8 BASH

```
# En una Mac de desarrollo de Abacom
$ memory_pressure
System memory pressure level: 2 (elevated)
Physical memory: 16GB

# Ver recursos del sistema
$ top -l 1 -stats cpu,mem -n 5

# Memoria disponible
$ vm_stat
```

9.5.3.2 macOS (Máquina de Desarrollo)

Por qué lo usamos en Abacom:

- Verificar que tenemos suficiente RAM para Docker/VirtualBox
- Monitorear si las herramientas de desarrollo están consumiendo demasiado
- Detectar memory leaks en aplicaciones

9.5.3.3 Windows (Máquina de Administrador)

Por qué lo usamos en Abacom:

- Monitorear servidores remotos desde Windows
- Verificar que tengas suficiente RAM en tu máquina
- Automatizar tareas de administración

Listing 9.9 POWERSHELL

```
# En Windows para administración remota de Abacom
PS> Get-ComputerInfo -Property TotalPhysicalMemory, FreePhysicalMemory

TotalPhysicalMemory : 17179869184 (16 GB)
FreePhysicalMemory  : 8589934592 (8 GB)

# Ver procesos por memoria
PS> Get-Process | Sort-Object Memory -Descending | Select-Object -First 5

# Información del sistema
PS> systeminfo | findstr /C:"Processor" /C:"System Type" /C:"Total Physical Memory"
```

9.6 Componentes Principales del Sistema Operativo

9.6.1 El Kernel - El Corazón

El kernel es la parte más importante. Gestiona:

- **Procesos:** Ejecuta programas y los alterna entre CPU
- **Memoria:** Asigna RAM a cada proceso
- **Devices (Dispositivos):** Controla disco, red, impresoras, etc.
- **Interrupts (Interrupciones):** Responde a eventos urgentes

Listing 9.10 BASH

```
# Ver información del kernel en Linux
$ uname -r                                                    ①
5.15.0-89-generic

# Ver módulos del kernel cargados
$ lsmod | head -10                                            ②
Module                  Size  Used by
overlay                81920  0
kvm_intel             323584  0
kvm                  1019904  1 kvm_intel
...
```

① **uname -r** muestra la versión exacta del kernel de tu sistema
② **lsmod** lista los módulos del kernel que están actualmente cargados en memoria

El kernel Linux se actualiza para seguridad y rendimiento, pero rara vez requiere reinicio.

9.6.2 Drivers - Los Traductores

Los drivers (controladores) permiten que el SO hable con el hardware específico:

- **Video Driver:** Comunica con tu tarjeta gráfica
- **Network Driver:** Comunica con tu tarjeta de red
- **Storage Driver:** Comunica con disco duro/SSD

Listing 9.11 BASH

```
# Listar dispositivos reconocidos
$ lspci | grep -E "(Network|VGA|Audio)"                                    ①
00:02.0 VGA compatible controller: Intel Corporation UHD Graphics 630
00:1f.6 Ethernet controller: Intel Corporation Ethernet Connection

# Ver drivers cargados
$ lsmod | grep -i eth                                                      ②
```

① **lspci** lista todos los dispositivos PCI conectados al sistema, filtramos solo Network, VGA y Audio
② **lsmod | grep -i eth** muestra solo los módulos de ethernet cargados en el kernel

Sin drivers correctos, el hardware no funciona.

9.6.3 Shell - La Interfaz

El shell es lo que **tú ves y usas**. Convierte tus comandos en órdenes que el kernel entiende:

Listing 9.12 BASH

```
# Usando shell bash
$ ls -la /home                                                             ①
total 28
drwxr-xr-x  3 root root 4096 Jan 29 14:22 .
drwxr-xr-x 23 root root 4096 Jan 29 14:22 ..
drwxr-xr-x  5 diego staff 160 Jan 29 14:22 diego

# El shell interpreta "ls -la /home" y le dice al kernel:                  ②
# "Lee el directorio /home y muestra detalles"
```

① **ls -la /home** lista todos los directorios en /home con detalles (permisos, dueño, fecha, etc.)
② El shell convierte este comando en instrucciones que el kernel entiende

Diferentes shells: **bash**, **zsh**, **sh**, **fish**.

9.7 Funciones Esenciales del Sistema Operativo

9.7.1 Gestión de Procesos

El SO decide **qué programa corre y cuándo**:

Listing 9.13 BASH

```
# Ver procesos en tiempo real
$ top -b -n 1 | head -20                                                    ①

top - 14:40:15 up 247 days, 3:47,  2 users,  load average: 0.92, 0.91, 0.89
Tasks: 156 total,   1 running, 155 sleeping,   0 stopped,   0 zombie      ②
%Cpu(s):  3.2 us,  2.1 sy,  0.0 ni, 94.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st ③
```

① **top -b -n 1** ejecuta top en modo batch (sin interfaz interactiva) una sola vez
② La línea "Tasks" muestra cuántos procesos están ejecutándose (156 total, 1 corriendo, 155 dormidos)
③ La línea "%Cpu(s)" muestra el porcentaje de CPU usado en modo usuario (us), sistema (sy), inactivo (id), etc.

Cómo funciona:

1. El SO maneja un "planificador" (scheduler) que decides cuál proceso ejecutar
2. Si tienes un CPU de 4 núcleos, puede ejecutar 4 cosas simultáneamente
3. En paralelo, el SO cambia entre procesos muy rápidamente (time slicing)
4. Resultado: Parece que 100 programas corren "al mismo tiempo"

9.7.2 Gestión de Memoria

El SO asigna RAM a cada proceso de forma segura:

Listing 9.14 BASH

```
# Ver uso de memoria
$ cat /proc/meminfo                                                         ①
MemTotal:       16409912 kB    # RAM Total en el sistema                    ②
MemFree:         9654832 kB    # RAM Disponible sin usar                    ③
MemAvailable:   12506144 kB    # RAM que puede usar si es necesario         ④
```

① **/proc/meminfo** es un archivo virtual que muestra información detallada de memoria del kernel
② MemTotal es la cantidad total de RAM disponible
③ MemFree es RAM completamente sin usar (no incluye caché)

④ MemAvailable es RAM que el kernel puede liberar rápidamente si la necesita otra aplicación …

Importancia:

- Cada proceso cree que tiene acceso a toda la memoria (memoria virtual)
- El SO maneja la traducción real a direcciones físicas
- Si se acaba RAM, usa disco (swap) - mucho más lento

9.7.3 Gestión de Dispositivos

El SO es el "policía" que controla acceso a hardware:

Listing 9.15 BASH

```
# Ver dispositivos de almacenamiento
$ lsblk                                                          ①
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT                       ②
sda      8:0    0  500G  0 disk                                  ③
 sda1   8:1    0    1G  0 part /boot                             ④
 sda2   8:2    0  499G  0 part /
sdb      8:16   0    2T  0 disk
 sdb1   8:17   0    2T  0 part /data
```

① **lsblk** (block list) muestra todos los dispositivos de almacenamiento en forma de árbol
② NAME: nombre del dispositivo, SIZE: tamaño, TYPE: tipo (disk=disco, part=partición), MOUNTPOINT: dónde está montado
③ **sda** es el primer disco duro (500 GB), identificado como 8:0 en el kernel
④ **sda1** y **sda2** son particiones (divisiones) del disco sda, la primera (1G) es /boot (donde arranca el SO)

El SO decide qué proceso puede acceder a qué dispositivo en qué momento.

9.7.4 Gestión de Archivos

El SO organiza datos en carpetas y archivos:

Listing 9.16 BASH

```
# Ver sistema de archivos
$ df -h                                                          ①
Filesystem      Size  Used Avail Use% Mounted on                 ②
/dev/sda2       500G  120G  380G  24% /                          ③
/dev/sdb1         2T  1.5T  500G  75% /data                      ④
tmpfs           7.8G     0  7.8G   0% /dev/shm
```

① **df -h** (disk free) muestra el espacio disponible en todos los sistemas de archivos montados (-h = human readable, en GB/TB)
② Filesystem: dispositivo, Size: tamaño total, Used: usado, Avail: disponible, Mounted on: dónde está montado
③ **/dev/sda2** es la partición principal (/) con 500G total, 120G usados, 380G disponibles (24% ocupado)
④ **/dev/sdb1** es una segunda partición montada como /data, casi llena (75% usada, solo 500G disponibles)

El SO traduce "abre archivo.txt" en "busca en sector X del disco Y".

9.8 Errores Comunes

9.8.1 Confundir "No hay memoria" con "No hay espacio en disco"

Síntoma que verás

```
Cannot allocate memory
# O en aplicaciones:
java.lang.OutOfMemoryError: Java heap space
```

Causa raíz: Se llenó la RAM, no el disco. Estas son dos cosas diferentes.
Solución comprobada:

Listing 9.17 BASH

```
#   Forma INCORRECTA (buscar espacio en disco)
$ df -h                                                    ①
Filesystem      Size  Used Avail
/               500G  100G  400G  ← Aún hay espacio!

#   Forma CORRECTA (revisar RAM)
$ free -h                                                  ②
               total    used    free
Mem:          15.6G   14.8G   0.8G  ← ¡RAM llena!
```

① df -h** muestra espacio en disco (500G disponibles), así que el problema NO es falta de disco
② **free -h** muestra RAM, y aquí está el problema: solo 0.8G de 15.6G disponibles, la RAM está casi llena

Por qué funciona la solución: El error es de RAM, no de disco. **df** ve espacio en disco, pero **free** muestra RAM agotada. Son dos sistemas diferentes del SO.

9.8.2 El sistema "se congela"

Warning

Síntoma: Todo se ralentiza o se detiene completamente
Causa: Generalmente swapping excesivo (usando disco en lugar de RAM) o CPU sobrecargada
Solución:

Listing 9.18 BASH

```
# Verificar carga de CPU
$ uptime                                                          ①
load average: 12.5, 11.2, 10.1  ← Si es > núcleos CPU, está sobrecargado ②

# Verificar swap (disco usado como memoria)
$ swapon -s                                                       ③
Filename  Size  Used Priority
/dev/sda1 4000000 3900000 -1  ← Si Used es alto, es el problema   ④
```

① **uptime** muestra cuánto tiempo lleva el sistema activo y el "load average" (promedio de procesos en cola)
② Si load average (12.5, 11.2, 10.1) es mayor que los núcleos CPU que tienes (ej: 4 núcleos), el CPU está sobrecargado
③ **swapon -s** muestra si el sistema está usando swap (disco como RAM, mucho más lento)
④ Si "Used" de swap es alto, significa que se llenó la RAM y ahora está usando disco, causando ralentizaciones severas

9.9 Mejores Prácticas

! Patrones Recomendados

HACER (Prácticas Probadas):

- Monitorear RAM disponible continuamente → previene problemas antes
- Mantener kernél actualizado → parches de seguridad
- Revisar procesos que consumen recursos → identificar culpables

NO HACER (Antipatrones):

- Ignorar alertas de memoria llena → causa crashes
- Ejecutar procesos como **root** innecesariamente → riesgo seguridad
- Llenar swap sin investigar → enmascara problema real

9.10 Tabla de Referencia Rápida

Comando	Propósito	Uso
`uname -a`	Ver información del SO y kernel	Diagnóstico del sistema
`ps aux`	Listar todos los procesos	Monitoreo de aplicaciones
`top` o `htop`	Monitor en tiempo real	Diagnóstico de rendimiento
`free -h`	Ver uso de memoria RAM	Verificar disponibilidad memoria
`df -h`	Ver espacio en disco	Monitorear particiones
`lsmod`	Listar módulos del kernel	Diagnóstico de drivers

9.11 Quiz: Verifica tu Comprensión

i Antes de continuar

Responde estas preguntas para verificar que comprendiste los conceptos clave. Si tienes dudas, vuelve a las secciones anteriores.

9.11.1 ¿Cuál es la diferencia entre Kernel y Shell?

Ver respuesta

Kernel es el software central que controla directamente el hardware. **Shell** es la interfaz que usa el usuario para comunicarse con el kernel.

Ejemplo diferenciador:

Listing 9.19 BASH

```
# El kernel hace esto:
# - Gestiona CPU, memoria, disco
# - Maneja interrupciones de hardware
# - Asigna recursos a procesos

# El shell hace esto:
$ ls -la /home
# ↑ Shell interpreta esto
# ↓ Le dice al kernel: "Lee directorio /home"
```

Por qué es importante la diferencia: El kernel es invisible - nunca lo usas directamente. El shell es lo que ves. Si entiendes esta separación, entiendes cómo funciona todo.

9.11.2 ¿Cuál es el comando para ver si tu RAM está llena?

Ver respuesta

Listing 9.20 BASH

```
$ free -h
```

Explicación técnica: free muestra memoria RAM. **-h** significa "human-readable" (formato legible). Este comando le dice al kernel: "dame un reporte de memoria".

Validación: Si ejecutas esto, deberías ver:

```
               total        used        free      shared  buff/cache   available
Mem:           15.6G       3.2G        9.1G       256M        3.3G        12.0G
```

Si **free** es muy pequeño o **used** es muy grande, la RAM está llena.

9.11.3 ¿En qué escenario revisarías `free -h` en lugar de `df -h`?

Ver respuesta

Usarías `free -h` cuando una **aplicación reporta error de memoria**, no cuando te falta espacio en disco.

Ejemplo práctico:

- Usar `free -h` cuando: Java reporta "OutOfMemoryError" o aplicación se crashes con "Cannot allocate memory"
  - No usar `free -h` cuando: El disco muestra "100% full" o error de "No space left on device"

  Caso Abacom: En nuestros servidores, cuando una base de datos se ralentiza, PRIMERO revisamos `free -h`. Si RAM está bien, LUEGO revisamos `df -h` y otros factores.

9.12 Práctica Guiada con el Instructor

9.12.1 Identificar tu Sistema Operativo

Objetivo: Ejecutar comandos básicos para entender qué SO tienes y cómo funciona.

Pasos:

1. Abrir una terminal (macOS/Linux) o PowerShell (Windows)
2. Ejecutar comandos para obtener información
3. Interpretar los resultados

Instrucciones:

Listing 9.21 BASH

```
# Paso 1: Ver información del SO
$ uname -a                                                    ①
# Deberías ver: información del kernel, versión, arquitectura

# Paso 2: Ver los primeros procesos del sistema
$ ps aux | head -10                                           ②
# Deberías ver: procesos del sistema operativo en ejecución

# Paso 3: Ver memoria disponible
$ free -h                                                     ③
# Deberías ver: cuánta RAM tienes y cuánta está disponible
```

① **uname -a** (all) muestra toda la información del sistema: nombre kernel, versión, máquina, procesador
② **ps aux** lista todos los procesos en ejecución, `| head -10` muestra solo los primeros 10
③ **free -h** (human readable) muestra RAM total, usada, libre en formato legible (GB)

Verificación:

- Ves el nombre del kernel (Linux, Darwin para macOS, etc.)
- Ves una lista de procesos numerados
- Ves números de memoria (Mem: XXG)

9.12.2 Monitorear Recursos en Tiempo Real

Objetivo: Observar cómo el SO gestiona recursos mientras ejecutas comandos.

Pasos:

1. Abrir **top** o **htop** para ver monitoreo
2. Notar cómo cambian los procesos
3. Ejecutar un comando que consume recursos

Instrucciones:

Listing 9.22 BASH

```
# Ejecuta esto en una terminal
$ top

# En otra terminal, ejecuta un comando que consume recursos
$ yes > /dev/null  # Genera CPU load

# Vuelve a la terminal con top
# Verás que el proceso "yes" aparece
# NOTA: presiona 'q' para salir de top
```

Verificación:

- Ves el proceso “yes” en la lista de top
- Ves que %CPU es alto para ese proceso
- El cambio ocurre en tiempo real

9.12.3 Ejercicio Avanzado - Investigar Problema de Rendimiento

Objetivo: Simular un diagnóstico real de Abacom.

Instrucciones:

Verificación:

Listing 9.23 BASH

```
# Escenario: El servidor está lento
# Paso 1: Ver carga del sistema
$ uptime
# Resultado esperado: load average values

# Paso 2: Ver si es por memoria
$ free -h
# Resultado esperado: see available memory

# Paso 3: Ver qué procesos consumen CPU
$ top -b -n 1 | sort -k 3 -nr | head -5
# Resultado esperado: top 5 procesos por CPU

# Paso 4: Conclusión
# "El servidor está lento porque [memoria/CPU/disco]"
```

- Entiendes qué significan los números
- Puedes identificar el culpable (memoria, CPU, disco)
- Sabes qué pasos seguir para resolver

9.13 Laboratorio de Práctica Integrado

Duración: 60-90 minutos
Dificultad: Intermedio
Objetivo General: Dominar diagnóstico básico de sistemas operativos

9.13.1 Fase 1: Preparación del Entorno (15 min)

Crea un espacio de trabajo para tus prácticas:

Listing 9.24 BASH

```
# Crear directorios de trabajo
$ mkdir -p ~/laboratorio-so/resultados
$ cd ~/laboratorio-so

# Crear archivo de notas
$ touch diagnostico.txt
```

9.13.2 Fase 2: Información del Sistema (20 min)

Tarea: Recopilar información completa del SO

Listing 9.25 BASH

```
# Archivo: resultados/info-so.txt
# Ejecuta estos comandos y guarda output:

$ uname -a > resultados/info-so.txt
$ cat /etc/os-release >> resultados/info-so.txt
$ lsb_release -a >> resultados/info-so.txt

# Verificar que se guardó
$ cat resultados/info-so.txt
# Deberías ver toda la información del SO
```

Ejercicio:

1. Ejecuta cada comando
2. Guarda los resultados en el archivo
3. Interpreta: ¿Qué versión de kernel tienes? ¿32 o 64 bits?

9.13.3 Fase 3: Monitoreo de Recursos (20 min)

Tarea: Registrar estado de recursos

Listing 9.26 BASH

```
# Archivo: resultados/recursos.txt
# Ejecuta estos comandos:

$ free -h > resultados/recursos.txt
$ df -h >> resultados/recursos.txt
$ top -b -n 1 | head -20 >> resultados/recursos.txt

# Verificar
$ cat resultados/recursos.txt
```

Ejercicio:

1. ¿Cuánta RAM tienes?
2. ¿Cuánta RAM está disponible?
3. ¿Cuál es el proceso que más CPU consume?

9.13.4 Fase 4: Análisis de Procesos (20 min)

Tarea: Identificar procesos críticos

Listing 9.27 BASH

```
# Archivo: resultados/procesos.txt
# Listar procesos ordenados por CPU

$ ps aux --sort=-%cpu | head -10 > resultados/procesos.txt

# Listar procesos ordenados por memoria
$ ps aux --sort=-%mem | head -10 >> resultados/procesos.txt

# Contar total de procesos
$ ps aux | wc -l >> resultados/procesos.txt
```

Ejercicio:

1. ¿Cuál es el proceso que más CPU consume?
2. ¿Cuál es el proceso que más memoria consume?
3. ¿Cuántos procesos totales hay en ejecución?

9.13.5 Fase 5: Proyecto Integrador - Diagnóstico Real (15 min)

Desafío Final: Crea un resumen de diagnóstico como un administrador real.

Escenario: En Abacom, el servidor está reportando bajo rendimiento. Necesitas diagnosticar.

Solución esperada:

Validación Final:

9.13.6 Verificación del Laboratorio

Marca cuando completes cada punto:

- ☐ Entorno preparado sin errores
- ☐ Información del SO recopilada correctamente
- ☐ Recursos monitoreados exitosamente
- ☐ Procesos identificados y analizados
- ☐ Diagnóstico final completado
- ☐ Resultado guardado en archivos

Listing 9.28 BASH

```
# Pasos para resolver
$ cat > resultados/diagnostico-final.txt << 'EOF'
=== DIAGNÓSTICO DEL SISTEMA ===

1. SISTEMA OPERATIVO:
[Información del OS aquí - ejecuta: uname -a]

2. MEMORIA RAM:
Total: [número de free -h]
Disponible: [número de free -h]
Uso: [porcentaje]

3. ESPACIO EN DISCO:
[Ejecuta: df -h]

4. CARGA DEL SISTEMA:
[Ejecuta: uptime]

5. PROCESO PRINCIPAL CONSUMIENDO CPU:
[Ejecuta: ps aux --sort=-%cpu | head -2]

6. CONCLUSIÓN:
El problema es [RAM / CPU / DISCO / OTRO] porque...
EOF

# Validación Final
$ cat resultados/diagnostico-final.txt
```

9.14 Recursos Adicionales

9.14.1 Documentación Oficial

- **Linux man-pages - standards(7)**
  - Sección relevante: Estándares POSIX y Unix
  - Por qué es útil: Entender los estándares que cumple Linux
- **Ubuntu Server Documentation**
  - Sección relevante: System fundamentals
  - Por qué es útil: Documentación oficial de Ubuntu

9.14.2 Tutoriales Complementarios

- **Linux Foundation - The Linux Foundation**

Listing 9.29 BASH

```
# Verifica que completaste todo
$ ls -la ~/laboratorio-so/resultados/
# Deberías ver: info-so.txt, recursos.txt, procesos.txt, diagnostico-final.txt
```

  - Duración: Varies
  - Cubre: Conceptos fundamentales de Linux

- **Brendan Gregg - Systems Performance**
  - Duración: Book (500+ pages)
  - Cubre: Diagnóstico avanzado de sistemas

9.14.3 Comunidades y Foros

- **Arch Wiki - Main page**
  - Para: Preguntas técnicas específicas
  - Cómo usar: Busca tu pregunta o crea una nueva
- **r/linux - Reddit**
  - Para: Discusiones generales y noticias
  - Cómo usar: Community muy activa y receptiva

9.15 Preguntas Frecuentes

P: ¿Necesito memorizar todos los comandos?

R: No. Lo importante es entender QUÉ hace cada comando. Con tiempo, memorizarás los más comunes. Siempre puedes usar **man** (manual) para recordar sintaxis.

Listing 9.30 BASH

```
# Ver manual de cualquier comando
$ man free
$ man ps
$ man df
```

P: ¿Por qué macOS es diferente a Linux si ambos son Unix?

R: Ambos siguen estándares POSIX, pero tienen diferencias:

- macOS usa kernel XNU (derivado de BSD y Mach)
- Linux usa kernel Linux (de Linus Torvalds)

Listing 9.31 BASH

```
# En Linux
$ free -h

# En macOS (no existe 'free', usar esto)
$ vm_stat
# O instalar: brew install gnu-coreutils
```

- Comandos básicos son similares, pero algunos flags difieren

P: ¿Cómo aplico esto en Abacom?

R: En nuestro caso, usamos estos comandos diariamente:

- **top** o **htop** cuando reportan problemas de rendimiento
- `free -h` para verificar si falta RAM
- `df -h` para monitorear espacio en disco
- `ps aux` para encontrar procesos problemáticos

Listing 9.32 BASH

```
$ # Diagnóstico rápido de servidor Abacom
$ uptime && free -h && df -h && top -b -n 1 | head -20
```

9.16 Resumen

Lo más importante (3 puntos clave)

1. **Sistema Operativo = Intermediario entre usuario y hardware** → Sin SO, no funciona nada. Todo pasa a través del SO.
2. **Kernel vs Shell** → Kernel gestiona hardware (invisible), Shell es la interfaz (lo que ves).
3. **Diagnóstico de problemas** → Usa `free -h` para memoria, `df -h` para disco, **top** para CPU.

Recuerda: Estos conceptos serán la base para entender administración avanzada de servidores.

9.16.1 Checklist de Competencias Alcanzadas

- Entiendo qué es un Sistema Operativo y por qué es crítico
- Puedo distinguir entre Kernel, Shell y Drivers
- Reconozco errores comunes (RAM llena vs disco lleno)
- Puedo usar comandos básicos para diagnosticar problemas
- Estoy listo para aprender Linux específicamente

9.16.2 Próximos Pasos

> **Note**
>
> **Siguiente tema:** Historia y Evolución de Linux
> **Laboratorio siguiente:** Instalación de Ubuntu 22.04 LTS
> **Tiempo recomendado:** 2 horas de lectura + 3 horas de práctica

---

9.17 Soporte

> **¿Tienes dudas sobre este tema?**
>
> - **Preguntas técnicas:** Abre un issue en [GitHub Issues]
> - **Errores no cubiertos:** Reporta en el repositorio
> - **Sugerencias para mejorar:** Envía feedback a diego@abacom.cl
> - **Recursos adicionales:** Disponibles en la sección Recursos Adicionales

---

Última actualización: 2026-01-29
Versión: 1.0
Estado: Completo y validado contra fuentes oficiales

> **Acerca de las fuentes**
>
> Todo el contenido de este módulo ha sido verificado contra:
>
> - Estándares POSIX.1-2017 y IEEE 1003.1
> - Documentación oficial de The Linux Foundation
> - Man-pages oficiales de Linux
>
> Las fuentes específicas se encuentran en Recursos Adicionales.

9.18 Quiz: Introducción a Sistemas Operativos

---

10 Unidad 1.2: Historia y Evolución de Linux

Listing 10.1 QUARTO-TITLE-BLOCK

11 Historia y Evolución de Linux

11.1 Introducción

Para entender Linux, primero necesitas conocer su historia. Linux no nació de la nada - es el resultado de **50 años de evolución de Unix** y la brillante decisión de Linus Torvalds de liberar el código bajo licencia abierta en 1991. En Abacom, comprender esta historia te ayuda a entender por qué Linux es gratuito, por qué funciona tan bien, y por qué es la opción ideal para servidores.

Tiempo de lectura: ~18 minutos
Nivel: Principiante
Requisitos previos: Concepto básico de SO (de la lectura anterior)

11.2 En este tema aprenderás

En esta unidad cubriremos:

1. **Los Orígenes: Unix (1960s-1980s)** - AT&T Bell Labs y la evolución
2. **Nacimiento de Linux (1991)** - Linus Torvalds y su proyecto
3. **Filosofía del Código Abierto** - Por qué "open source" importa
4. **Evolución del Kernel Linux** - Versiones y mejoras críticas
5. **Impacto Moderno** - Linux hoy: servidores, móviles, IoT

11.3 Unix - El Abuelo de Linux

Definición clara:

Unix fue un sistema operativo revolucionario creado en **1969 en AT&T Bell Labs** (Unix.com 2024). No fue el "primero", pero fue el que definió cómo debería verse un SO moderno. Sus principios siguen siendo válidos hoy.

¿Por qué importa?

- **Impacto directo:** Todos los SOs modernos (Linux, macOS, Unix comercial) descienden de los principios de Unix.
- **En Abacom:** Linux es "Unix-like" - sigue los estándares POSIX que vienen de Unix.
- **Ventaja profesional:** Un admin que conoce Unix puede trabajar en cualquier sistema similar.

11.4 Linus Torvalds y el Nacimiento de Linux (1991)

Definición clara:

En 1991, un estudiante finlandés de 21 años llamado **Linus Torvalds** escribió un pequeño kernel basado en Minix y lo publicó en Usenet con las palabras (Torvalds 1991):

> *"Hello from Finland. I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones."*

Eso fue el nacimiento de Linux.

Punto importante

Linus NO inventó Linux solo. Lo que hizo fue:

1. Escribir un kernel pequeño y funcional
2. **Liberar el código bajo licencia GPL** (*GNU GPL V3.0* 2007)
3. Aceptar contribuciones de otros desarrolladores
4. Mantener la comunidad colaborativa

La magia no fue el código original, sino la DECISIÓN de hacerlo abierto.

11.5 Ejemplos Prácticos

11.5.1 Ejemplo 1: Comparar Versiones de Kernel

11.5.1.1 Linux

① **uname -r** muestra la versión exacta del kernel instalado en formato major.minor.patch

Listing 11.1 BASH

```
# Ver versión actual de kernel en Linux
$ uname -r                                                    ①
5.15.0-89-generic

# Desglosar la versión:
# 5       = Versión mayor (cambios significativos)
# 15      = Versión menor (nuevas características)
# 0       = Revisión de parches
# 89      = Ubuntu patch number
# generic = Tipo de kernel
```

Explicación:

- Linux 0.01 (1991): ~10,000 líneas de código
- Linux 1.0 (1994): Primer release estable
- Linux 5.0 (2019): Millones de líneas

11.5.1.2 macOS

Listing 11.2 BASH

```
# Ver versión de kernel en macOS (Darwin)
$ uname -r                                                    ①
23.3.0

# Versión más detallada
$ sw_vers                                                     ②
ProductName:    macOS
ProductVersion: 14.2.1

# Información del kernel
$ sysctl kern.version                                         ③
Darwin Kernel Version 23.3.0
```

① **uname -r** en macOS muestra la versión de Darwin (kernel basado en BSD/Unix)
② **sw_vers** muestra la versión de macOS que corresponde a esa versión de Darwin
③ **sysctl kern.version** proporciona información detallada del kernel Darwin

Explicación:

- macOS usa Darwin (kernel de Unix)
- Las versiones de Darwin no siempre coinciden con las de macOS
- Darwin 23.x corresponde a macOS 14.x

11.5.1.3 Windows

Listing 11.3 POWERSHELL

```
# Ver versión de kernel en Windows
PS> [System.Environment]::OSVersion                                          ①
Platform ServicePack Version         VersionString
-------- ----------- -------         -------
Win32NT              10.0.22621.3737  Microsoft Windows NT 10.0.22621.3737

# Versión más legible
PS> (Get-Item "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion").GetValue('ProductName
Windows 11 Professional

# Información del kernel
PS> Get-ComputerInfo -Property OsVersion, KernelVersion                      ③
```

① **[System.Environment]::OSVersion** muestra la versión del SO en formato de .NET, incluyendo versión de kernel interna
② **Lectura del Registry** obtiene el nombre del producto (Windows 11, Windows Server, etc.) de forma más legible
③ **Get-ComputerInfo** es el cmdlet moderno en PowerShell 7+ que proporciona información del sistema completa

Explicación:

- Windows no usa "versión de kernel" de la misma forma
- 10.0.22621 indica Windows 11 (10.0 es la versión interna)
- 22621 es el número de build

11.5.2 Ejemplo 2: Ver Historia del Kernel

11.5.2.1 Linux

① **cd /tmp** navega al directorio temporal para no llenar el disco de tu home
② **git clone** descarga TODO el repositorio (1.5-2GB) con toda la historia de commits desde 1991
③ **cd linux** entra en el directorio del repositorio clonado
④ **git log –reverse** muestra commits en orden cronológico (más antiguo primero) y **head -10** limita a los primeros 10
⑤ **git log –oneline | wc -l** cuenta el número TOTAL de commits en toda la historia del kernel

Cómo funciona:

Listing 11.4 BASH

```
# Descargar historia del kernel Linux (requiere ~2GB)
$ cd /tmp                                                                    ①
$ git clone https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git ②

# Ver primer commit (1991)
$ cd linux                                                                   ③
$ git log --reverse | head -10                                               ④
commit 1da177e4c3f41524e886b7f1b8a0c1fc7321cac2
Author: Linus Torvalds <torvalds@ppc970.osdl.org>
Date:   Wed Apr 16 15:20:36 2002

# Ver estadísticas del proyecto
$ git log --oneline | wc -l                                                  ⑤
1234567 (más de un millón de commits)
```

- El kernel Linux tiene historia de commits desde 1991
- Cada versión fue construida encima de la anterior
- Hoy hay miles de desarrolladores contribuyendo

11.5.2.2 macOS

Listing 11.5 BASH

```
# Darwin (macOS kernel) es open source parcialmente
# Puedes ver código en https://opensource.apple.com

# Ver información local del kernel
$ uname -a                                                                   ①
Darwin macbook 23.3.0 Darwin Kernel Version 23.3.0

# Ver versiones de Darwin disponibles
$ sw_vers -productVersion                                                    ②
14.2.1

# El kernel evoluciona con cada versión de macOS
# macOS 14.2.1 = Darwin 23.3.0
```

① **uname -a** muestra toda la información del sistema: nombre del host, versión de Darwin, arquitectura (arm64 para Apple Silicon)

② **sw_vers -productVersion** muestra SOLO la versión de macOS (más limpio que **sw_vers** completo)

Cómo funciona:

- Darwin es basado en BSD (que es Unix)
- Menos abierto que Linux pero código disponible
- Evoluciona anualmente con nuevas versiones de macOS

11.5.2.3 Windows

Listing 11.6 POWERSHELL

```
# Windows tiene historial en Github (parcialmente)
# Descarga desde: https://github.com/microsoft/Windows-driver-samples

# Ver información de versión en tu sistema
PS> Get-Item "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion"        ①

ValueName      Type   Value
---------      ----   -----
CurrentBuild   String 22621                                              ②
ReleaseId      String 23H2                                               ③

# Windows 11 Kernel timeline:
# Windows 11 21H2 (initial)
# Windows 11 22H2 (major update)
# Windows 11 23H2 (latest)
```

① **Get-Item** accede al Registry de Windows donde se almacena información del SO
② **CurrentBuild** es el número de compilación que identifica la versión específica (22621 = Windows 11 23H2)
③ **ReleaseId** indica el ciclo de lanzamiento (H1 = primer semestre, H2 = segundo semestre)

Cómo funciona:

- Windows kernel es propietario
- Versiones numeradas por año (21H2 = 2021 H2)
- Actualizaciones trimestrales de seguridad

11.5.3 Ejemplo 3: Caso Real Abacom

11.5.3.1 Linux (Servidor)

① **/proc/version** es un archivo virtual que muestra información completa del kernel compilado (compilador usado, flags, fecha)
② **uname -a** resume la información de forma más legible: nombre del servidor, versión de kernel, arquitectura (x86_64)

Listing 11.7 BASH

```
# En nuestros servidores Linux de Abacom
$ cat /proc/version                                                    ①
Linux version 5.15.0-89-generic (buildd@lgw02-amd64-030)
(gcc-11 (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0, GNU ld 2.38)
#99-Ubuntu SMP Mon Oct 5 09:29:34 UTC 2024

# Verificar compatibilidad de módulos
$ uname -a                                                             ②
Linux prod-server-01 5.15.0-89-generic #99-Ubuntu SMP Mon Oct 5 09:29:34 UTC 2024 x86_64
```

Por qué lo usamos en Abacom:

- Verificar que estamos en kernel estable (5.15.0)
- Asegurar compatibilidad con aplicaciones
- Planificar upgrades cuando salen parches críticos
- Kernel LTS (Long Term Support) = Soporte por 5 años

11.5.3.2 macOS (Desarrollo)

Listing 11.8 BASH

```
# En las máquinas MacBook Pro de Abacom
$ uname -a                                                             ①
Darwin macbook-abacom 23.3.0 Darwin Kernel Version 23.3.0 arm64

# Verificar compatibilidad de Docker y VirtualBox
$ sysctl hw.physicalcpu                                                ②
hw.physicalcpu: 8

# Versión de desarrollo tools
$ clang --version                                                      ③
Apple clang version 15.0.0
```

① **uname -a** muestra "arm64" (Apple Silicon M1/M2/M3) vs "x86_64" (Intel), crucial para compatibilidad de herramientas
② **sysctl hw.physicalcpu** verifica el número real de CPU cores (importante para Docker y VirtualBox performance)
③ **clang –version** verifica la versión del compilador de Apple (necesario para compilar código C/C++)

Por qué lo usamos en Abacom:

- Verificar compatibilidad Apple Silicon vs Intel
- Asegurar que Docker y herramientas funcionan

- Compatibilidad entre equipos del equipo

11.5.3.3 Windows (Administración)

Listing 11.9 POWERSHELL

```
# En máquinas Windows de administración
PS> [System.Environment]::OSVersion.VersionString                    ①
Microsoft Windows NT 10.0.22621.3737

# Verificar que sea Windows 11 Pro/Enterprise
PS> (Get-ComputerInfo -Property WindowsInstallationType).WindowsInstallationType ②
Client

# Versión de PowerShell para scripts
PS> $PSVersionTable.PSVersion                                        ③
Major  Minor  Build  Revision
-----  -----  -----  --------
7      4      11     0
```

① **[System.Environment]::OSVersion.VersionString** obtiene la versión del SO en formato legible (Windows NT 10.0 es Windows 11 internamente)
② **Get-ComputerInfo** es el cmdlet moderno que obtiene "Client" (Windows 11) vs "Server" (Windows Server)
③ **$PSVersionTable.PSVersion** verifica la versión de PowerShell (7.x es PowerShell Core moderno, 5.x es el antiguo)

Por qué lo usamos en Abacom:

- Administración remota de servidores
- Automatización con PowerShell
- Compatibilidad con herramientas de administración
- Windows 11 es estable para administración

11.6 Evolución del Kernel Linux

11.6.1 Era Antigua (1991-2003) - Los Fundamentos

En este período, Linux pasó de ser un hobby a ser usable profesionalmente:

```
1991: Linux 0.01 - Idea loca de Linus
1993: Linux 1.0 - Primer "release" usable
1996: Linux 2.0 - Soporte SMP (múltiples CPUs)
2001: Linux 2.4 - Mejoras de rendimiento
```

Lo importante:

- El kernel creció de ~10K líneas a millones
- Se agregaron drivers para más hardware
- Comunidad internacional comenzó a crecer

11.6.2 Era Moderna (2003-2011) - Estabilidad

Linux 2.6 fue una era dorada de estabilidad:

Listing 11.10 BASH

```
# Kernel 2.6 duró... 8 años en desarrollo!
# Porque priorizó:
# - Estabilidad (sin crashes)
# - Compatibilidad (sin romper código viejo)
# - Rendimiento (cada versión más rápido)
```

Por qué importa:

- Los sistemas de hoy necesitan estabilidad
- Kernel 2.6 fue la base para servidores críticos
- Aún hay sistemas usando kernel 2.6 en producción

11.6.3 Era Actual (2011-Presente) - Innovación Rápida

Desde Linux 3.0, el kernél cambia cada 2-3 meses:

Listing 11.11 BASH

```
# Timeline reciente
2011: Linux 3.0                                                    ①
2014: Linux 3.14 → Renombrado a 4.0 (marketing)                    ②
2019: Linux 5.0                                                    ③
2024: Linux 6.7+                                                   ④

# Cada versión introduce nuevas características:
# - Mejor seguridad
# - Soporte para nuevo hardware
# - Mejoras de rendimiento
```

① Linux 3.0 marcó el inicio del desarrollo rápido (cambios cada 2-3 meses en lugar de años)
② En 2014, el número de versión saltó de 3.14 a 4.0, un cambio de marketing para que pareciera más grande
③ Linux 5.0 llegó en 2019, 8 años después del cambio a 4.0, continuando la numeración
④ Hoy en 2024/2025, estamos en kernel 6.x, demostrando la evolución constante del kernel

En Abacom:

- Ubuntu 22.04 LTS usa kernel 5.15 (LTS = Long Term Support)
- "LTS" significa 5 años de actualizaciones de seguridad
- No cambiamos kernel frecuentemente en producción

11.7 Filosofía del Código Abierto

11.7.1 La Licencia GPL (General Public License)

La GPL es lo que hace que Linux sea verdaderamente "libre":

Listing 11.12 BASH

```
# Ver licencia de Linux
$ head -20 /usr/share/doc/linux-image-*/copyright                    ①
# O en GitHub: https://github.com/torvalds/linux/blob/master/COPYING

# Los 4 puntos clave:                                                ②
# 1. Libertad de usar (para cualquier propósito)
# 2. Libertad de estudiar (acceso al código)
# 3. Libertad de modificar (crear versiones personalizadas)
# 4. Libertad de distribuir (compartir cambios)
```

---

① En cualquier sistema Linux, puedes ver la licencia GPL completa en **/usr/share/doc/** o en el repositorio oficial de GitHub
② La GPL tiene 4 libertades fundamentales: usarlo, estudiarlo, modificarlo y compartir los cambios. Esto es lo que hace a Linux verdaderamente "libre"

Importancia:

- No pagas por usar Linux (es gratis)
- Puedes ver exactamente qué hace (transparencia)
- Si lo modificas, debes compartir los cambios

Listing 11.13 BASH

```
# Estadísticas reales del kernel Linux
# Aproximadamente:                                                    ①
# - 18,000+ archivos de código
# - 28+ millones de líneas de código
# - 2,000+ nuevas funciones por versión
# - Contribuciones de: Intel, Red Hat, Google, Canonical, etc.

# Ver estadísticas de git del kernel
$ git log --oneline | wc -l                                           ②
# Resultado: Millones de commits en 30+ años
```

11.7.2 Colaboración Global

Miles de desarrolladores contribuyen:

① El kernel Linux tiene más de 28 millones de líneas de código, distribuidas en 18,000+ archivos, desarrollado por miles de desarrolladores de diferentes empresas

② El comando **git log –oneline** muestra todos los commits (cambios) en el historial del kernel, y **wc -l** los cuenta. Resultado: millones de commits demostrando 30+ años de desarrollo colaborativo

En la práctica:

- Si necesitas una característica específica
- Puedes escribirla o contratar a alguien
- Puedes contribuir de vuelta a la comunidad

11.7.3 Seguridad mediante Transparencia

Con código abierto, más ojos = más seguridad:

Listing 11.14 BASH

```
# Cuando se descubre un bug de seguridad en Linux:
# 1. Se reporta a la lista de seguridad (privadamente)
# 2. Desarrolladores trabajan en parche
# 3. Parche se revisa en público (en GitHub)
# 4. Se libera a todos al mismo tiempo

# En Windows/macOS, es:
# 1. Descubrimiento privado
# 2. Microsoft/Apple trabaja en parche
# 3. Se libera cuando ELLOS deciden
# 4. Todos reciben el parche (esperemos)
```

Ventaja:

- Bugs se encuentran y se arreglan más rápido
- No dependes de una compañía para seguridad
- Comunidad revisa cambios antes de usar

11.8 Errores Comunes

11.8.1 "Linux es fácil de hackear porque está el código abierto"

Síntoma que verás

`"Como el código es público, hackers pueden encontrar vulnerabilidades"`

Causa raíz: Malentendido de cómo funciona la seguridad.
Solución comprobada:
Verdadero: Con código abierto, vulnerabilidades SE ENCUENTRAN MÁS RÁPIDO
Falso: Código abierto hace que sea más inseguro
Datos reales:

- Windows (código cerrado): Promedios de 25-30 CVEs por mes
- Linux (código abierto): Promedios de 15-20 CVEs por mes
- La diferencia: En Linux se descubren Y se arreglan juntas

Por qué funciona: Miles de expertos revisan el código Linux constantemente. En código cerrado, solo el fabricante lo revisa. ¿Crees que hay más expertos en Microsoft o en toda la comunidad global?

11.8.2 "Linux cambió mucho desde su versión vieja"

Warning

Síntoma: "Mi código de 20 años no compila en kernel nuevo"
Causa: Cambios en APIs (interfaces de programación) a través de versiones
Solución:

Listing 11.15 BASH

```
# Ubuntu LTS (Long Term Support) es para esto
# Ubuntu 22.04 LTS = 5 años de kernel 5.15
# No necesitas actualizar kernel cada mes
# Puedes esperar años entre upgrades

# Verificar soporte de tu versión
$ lsb_release -d
Description:    Ubuntu 22.04.3 LTS
```

11.9 Mejores Prácticas

! Patrones Recomendados

HACER (Prácticas Probadas):

- Usar Ubuntu LTS en producción → soporte de 5 años
- Mantener kernel actualizado con parches de seguridad → cerrar vulnerabilidades
- Contribuir a comunidad si encuentras bugs → ayuda a todos

NO HACER (Antipatrones):

- Usar kernel experimental en producción → inestabilidad
- Ignorar parches de seguridad → riesgo masivo
- Modificar kernel sin conocimiento profundo → causa problemas

11.10 Tabla de Referencia Rápida

Evento	Año	Significado
Unix creado	1969	Fundamentos de todos los SOs
Linux 0.01	1991	“Just a hobby” - Linus Torvalds
Linux 1.0	1994	Primero estable para producción
Linux 2.0	1996	Soporte SMP (múltiples CPUs)
Linux 2.6	2003	Era de estabilidad (8 años)
Linux 3.0	2011	Cambios rápidos (cada 2-3 meses)
Linux 5.0	2019	Era contemporánea actual

11.11 Quiz: Verifica tu Comprensión

> **Antes de continuar**
>
> Responde estas preguntas para verificar que comprendiste los conceptos clave.

11.11.1 ¿Cuál es la diferencia entre Unix y Linux?

Ver respuesta

Unix es un SO creado en 1969 por AT&T. **Linux** es un kernel creado en 1991 por Linus Torvalds, inspirado en Unix pero escrito desde cero.

Ejemplo diferenciador:

Listing 11.16 BASH

```
# En una máquina Unix/Linux:
$ uname -s

Linux           # O: Darwin (macOS), AIX, SunOS, etc.

# El kernel actual (Linux)
# Pero sigue estándares POSIX de Unix
# Por eso puedes ejecutar commands similares en ambos
```

Por qué es importante la diferencia:

- Unix = Concepto/Filosofía (1969)
  - Linux = Implementación específica (1991)
  - Linux es "Unix-like" (similar a Unix, no idéntico)

11.11.2 ¿Quién escribió Linux y cuándo?

Ver respuesta

Linus Torvalds en 1991. Fue un estudiante finlandés que escribió un kernel pequeño como "hobby" y lo publicó bajo licencia GPL.

Dato importante: "'bash # Linus NO escribió todo Linux solo # Lo que hizo fue: # 1. Escribir el kernel inicial # 2. Liberar bajo GPL # 3. Aceptar contribuciones de otros # 4. Mantener la coordinación

12 Hoy: +18,000 desarrolladores contribuyen

```
  **Validación:** Si ejecutas esto, ves referencias a Linus:
  ```bash
  $ git log --reverse | grep "Linus Torvalds"
# Verás los commits originales de 1991
```

12.0.1 ¿Por qué deberías usar kernel LTS en producción?

Ver respuesta

LTS (Long Term Support) significa que Ubuntu garantiza parches de seguridad por 5 años sin cambios grandes.

Escenarios:

- Usar LTS en producción: Estabilidad máxima, actualizaciones solo de seguridad
  - No usar LTS en producción: Cambios cada 2-3 meses, riesgo de breaking changes

  Caso Abacom: Nuestros servidores de producción usan Ubuntu 22.04 LTS porque necesitamos:

  - Kernel estable (5.15)
  - 5 años de seguridad garantizada
  - Mínimos cambios que puedan romper aplicaciones

12.1 Práctica Guiada con el Instructor

12.1.1 Explorar Historia de tu Kernel

Objetivo: Ver de dónde viene tu kernel.

Pasos:

1. Verificar versión actual
2. Ver fecha de compilación
3. Entender qué versión es
```

Instrucciones:

Listing 12.1 BASH

```
# Paso 1: Ver versión del kernel
$ uname -r                                                          ①
5.15.0-89-generic

# Desglosar:
# 5.15 = Versión principal (actual)
# 0 = Revisión
# 89 = Número de Ubuntu patch

# Paso 2: Ver fecha de compilación
$ cat /proc/version                                                 ②
Linux version 5.15.0-89-generic ... (fecha aquí)

# Paso 3: Ver si es LTS
$ lsb_release -d                                                    ③
Description:    Ubuntu 22.04.3 LTS  ← LTS significa Long Term Support
```

① **uname -r** es la forma más rápida de obtener solo la versión del kernel
② **cat /proc/version** muestra información adicional: compilador usado, fecha exacta de compilación
③ **lsb_release -d** verifica si es versión LTS (Long Term Support = 5 años de actualizaciones)

Verificación:

- Ves versión del kernel
- Sabes si es LTS o no
- Entiendes qué versión major es (5.x)

12.1.2 Entender GPL (Licencia)

Objetivo: Verificar que Linux es realmente libre.

Instrucciones:

① **head -30** muestra solo las primeras 30 líneas del archivo de derechos de autor (copyright), donde está la licencia GPL
② **ls /usr/src/** lista los headers del kernel disponibles - el código fuente está disponible públicamente en tu máquina
③ **git clone** del repositorio oficial permite clonar la historia COMPLETA del kernel (1.5-2GB)

Verificación:

- Ves la licencia GPL

Listing 12.2 BASH

```
# Paso 1: Ver licencia de Linux
$ head -30 /usr/share/doc/linux-image-*/copyright                    ①
# Deberías ver: GPL v2 license

# Paso 2: Verificar que puedes ver el código
# En casi cualquier sistema Linux, el código está disponible:
$ ls /usr/src/                                                       ②
# Verás: linux-headers-5.15.0-...

# Paso 3: Descargar kernel desde GitHub (oficial)
$ # Este paso es opcional, requiere git y espacio
$ git clone https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git ③
# Descargará toda la historia de 30 años de commits
```

- Verificas que el código es público
- Entiendes que "GPL" significa libertad

12.1.3 Ejercicio Avanzado - Timeline de Versiones

Objetivo: Entender cómo evolucionó Linux.

Instrucciones:

Listing 12.3 BASH

```
# Crear un archivo con timeline
$ cat > /tmp/linux-timeline.txt << 'EOF' # <1>
VERSIÓN | AÑO | EVENTO
--------|-----|-------
0.01    | 1991 | Idea loca de Linus
1.0     | 1994 | Primera version estable
2.0     | 1996 | SMP (múltiples CPUs)
2.4     | 2001 | Servidor enterprise
2.6     | 2003 | 8 años de estabilidad
3.0     | 2011 | Cambios rápidos comienzan
4.0     | 2015 | Renumeración de versiones
5.0     | 2019 | Era contemporánea
6.0     | 2023 | Futuro cercano
EOF

# Ver timeline
$ cat /tmp/linux-timeline.txt                                        ②
```

① **cat > archivo « 'EOF'** es la forma estándar de crear archivos multi-línea en bash (EOF marca el final)
② **cat archivo** muestra el contenido del archivo creado

Ejercicio:

1. ¿Cuándo fue la "era de estabilidad"?
2. ¿Cuándo comenzó el kernel a cambiar cada 2-3 meses?
3. ¿Qué versión tiene tu sistema?

12.2 Laboratorio de Práctica Integrado

Duración: 60-90 minutos
Dificultad: Intermedio
Objetivo General: Entender historia y evolución de Linux en profundidad

12.2.1 Fase 1: Investigación del Kernel Actual (20 min)

Listing 12.4 BASH

```
# Crear directorio de trabajo
$ mkdir -p ~/lab-historia-linux                                    ①
$ cd ~/lab-historia-linux                                          ②

# Recopilar información del kernel
$ uname -a > kernel-info.txt                                       ③
$ cat /proc/version >> kernel-info.txt                             ④
$ cat /etc/os-release >> kernel-info.txt                           ⑤

# Ver resultado
$ cat kernel-info.txt                                              ⑥
```

---

① **mkdir -p** crea el directorio (la flag -p crea directorios padre si es necesario)
② **cd** navega al directorio recién creado
③ **uname -a >** ejecuta el comando y redirige la salida (>) a un archivo, reemplazando contenido previo
④ **cat »** AGREGA el contenido (») sin borrar lo que ya está en el archivo
⑤ **cat /etc/os-release »** agrega más información al archivo
⑥ **cat** muestra el contenido final del archivo

Listing 12.5 BASH

```
# Este paso requiere clonar el kernel (large, ~2GB)
# Opcional - si no quieres hacer clone, salta a Fase 3

$ cd /tmp                                                                    ①
$ git clone --depth 1 https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git
$ cd linux                                                                   ③

# Ver primeros commits de 1991
$ git log --reverse --oneline --since="1991-01-01" --until="1991-12-31" | head -10 ④

# Ver commits más recientes
$ git log --oneline -n 10                                                    ⑤
```

12.2.2 Fase 2: Buscar en la Historia de Git (20 min)

① **cd /tmp** evita llenar tu directorio home (el kernel es ~2GB)
② **git clone –depth 1** clona SOLO el último commit (más rápido) en lugar de todo el historial (2GB)
③ **cd linux** entra en el directorio del repositorio clonado
④ **git log –since/until** filtra commits por fecha, **–reverse** muestra más antiguos primero, **head -10** limita a 10 resultados
⑤ **git log -n 10** muestra los últimos 10 commits más recientes

12.2.3 Fase 3: Analizar Versiones (20 min)

① **cat > archivo « 'EOF'** crea un archivo con contenido multi-línea (EOF marca el final de entrada)
② **cat** muestra el contenido del archivo creado para verificar

12.2.4 Fase 4: Crear Informe Final (20 min)

① **cat > archivo « 'EOF'** crea el informe final multi-línea
② **cat** muestra el contenido para verificación

12.2.5 Verificación del Laboratorio

- ☐ Información del kernel recopilada
- ☐ Versiones principales entendidas
- ☐ Analisis de versiones completado
- ☐ Informe final escrito
- ☐ Entiendo por qué usar LTS en producción

Listing 12.6 BASH

```
# Crear análisis de versiones principales
$ cat > ~/lab-historia-linux/analisis-versiones.txt << 'EOF' # <1>
=== ANÁLISIS DE VERSIONES LINUX ===

Mi kernel actual: [ejecuta: uname -r]

Versiones principales en historia:

- 0.01 (1991): Hobby de Linus
- 1.0 (1994): Primer release estable
- 2.0 (1996): Multiproceso (SMP)
- 2.6 (2003): Era de estabilidad
- 3.0 (2011): Cambios rápidos
- 5.0 (2019): Contemporáneo

¿Qué versión tiene mi sistema?
[Tu versión aquí]

¿Es LTS?
[Sí/No - ver con: lsb_release -d]
EOF

$ cat ~/lab-historia-linux/analisis-versiones.txt                    ②
```

12.3 Recursos Adicionales

12.3.1 Documentación Oficial

- **Linux kernel documentation**
  - Sección: Timeline interactivo
  - Por qué: Historia visual de desarrollo
- **Kernel.org**
  - Sección: Releases y Changelog
  - Por qué: Fuente oficial de releases

12.3.2 Tutoriales Complementarios

- **Linus Torvalds - Conferencias**
  - Duración: 30-60 minutos por charla
  - Cubre: Historia directa de Linus

Listing 12.7 BASH

```
# Resumen ejecutivo para Abacom
$ cat > ~/lab-historia-linux/informe-final.txt << 'EOF' # <1>
=== INFORME: HISTORIA DE LINUX EN ABACOM ===

1. ¿CUÁL ES NUESTRO KERNEL?
[Resultado de: uname -r]

2. ¿ES ESTABLE?
[Investigación]

3. ¿TIENE SOPORTE A LARGO PLAZO?
[Resultado de: lsb_release -d]

4. ¿CUÁNTO TIEMPO TIENE SOPORTE?
Ubuntu LTS tiene soporte por 5 años

5. ¿CUÁNDO DEBO ACTUALIZAR?
Solo cuando termine el período de soporte
Ejemplo: Ubuntu 22.04 LTS hasta Abril 2027

6. ¿QUÉ PASÓ EN 30 AÑOS DE LINUX?
De 10,000 líneas a millones
De hobby a la infraestructura de Internet
EOF

$ cat ~/lab-historia-linux/informe-final.txt                    ②
```

12.3.3 Comunidades

- **LKML (Linux Kernel Mailing List)**
  - Para: Desarrollo de kernel en tiempo real
  - Cómo: Suscríbete para ver cambios

12.4 Preguntas Frecuentes

P: ¿Por qué Linux es gratis si cuesta millones desarrollarlo?

R: Porque es software abierto y colaborativo:

- Nadie es dueño de Linux
- Empresas (Red Hat, Canonical, Google) contribuyen porque beneficia sus productos
- Comunidad global dona tiempo

- No hay intermediario que cobre

P: ¿Significa gratis que es de menor calidad?

R: No, todo lo opuesto. Porque:

- Miles de expertos revisan código
- Bugs se descubren y se arreglan rápido
- No hay presión comercial de "lanzar a tiempo"
- Competencia académica por calidad

P: ¿En Abacom pagamos por Linux?

R: Posiblemente pagas por:

- **Soporte técnico** (Red Hat, Canonical)
- **Hardware** donde corre
- **Administración** (gente como nosotros)
- **Certificación** de segurid

ad

Pero por el SO mismo: ¡No!

12.5 Resumen

Lo más importante (3 puntos clave)

1. **Unix (1969) → Linux (1991)** → Linux es descendiente de Unix, no lo contrario. Heredó diseño y filosofía.
2. **Linus Torvalds + GPL** → La decisión de liberar código bajo GPL fue el punto de inflexión que hizo a Linux lo que es hoy.
3. **LTS en Producción** → Usar Ubuntu LTS garantiza 5 años de estabilidad sin cambios inesperados.

Recuerda: Entender historia te ayuda a tomar mejores decisiones hoy.

12.5.1 Checklist de Competencias Alcanzadas

- Entiendo el origen de Linux (Linus, 1991, GPL)
- Puedo distinguir entre Unix y Linux
- Sé por qué usar versiones LTS en producción
- Comprendo ventajas de código abierto
- Estoy listo para aprender distribuciones

12.5.2 Próximos Pasos

> **Note**
>
> **Siguiente tema:** Distribuciones de Linux
> **Laboratorio siguiente:** Comparar y elegir distribuciones
> **Tiempo recomendado:** 1.5 horas de lectura + 2 horas de práctica

---

12.6 Soporte

> **¿Tienes dudas sobre este tema?**
>
> - **Preguntas técnicas:** GitHub Issues
> - **Errores históricos:** Reporta correcciones
> - **Sugerencias:** Envía feedback

Última actualización: 2026-01-29
Versión: 1.0
Estado: Completo y validado contra fuentes oficiales

> **Acerca de las fuentes**
>
> Verificado contra:
>
> - The Linux Foundation oficial history
> - POSIX standards documentation
> - Linux kernel.org releases

12.7 Quiz: Historia de Linux

---

13 Unidad 1.3: Distribuciones de Linux

Listing 13.1 QUARTO-TITLE-BLOCK

14 Distribuciones de Linux

14.1 Introducción

Linux es como una **receta base de un pastel**: todos usan el mismo kernel Linux, pero cada distribuidor agrega ingredientes diferentes. Una "distribución" es Linux empaquetado con herramientas, paquetes y configuraciones específicas. En Abacom, hemos elegido **Ubuntu 22.04 LTS** porque es la mejor combinación de estabilidad, soporte y facilidad de administración para empresas como la nuestra.

Tiempo de lectura: ~22 minutos
Nivel: Principiante-Intermedio
Requisitos previos: Concepto de kernel Linux (de lecturas anteriores)

14.2 En este tema aprenderás

En esta unidad cubriremos:

1. **¿Qué es una Distribución?** - Kernel + herramientas + configuración
2. **Familias de Distribuciones** - Debian, Red Hat, Arch, etc.
3. **Distribuciones Principales** - Ubuntu, CentOS, Debian, Fedora
4. **Comparativa Técnica** - Diferencias en package managers, release cycles
5. **¿Por qué Ubuntu 22.04 LTS para Abacom?** - Análisis de decisión

---

14.3 ¿Qué es una Distribución de Linux?

Definición clara:

Una distribución es **Linux (el kernel) + un conjunto de software y herramientas** (DistroWatch 2024). Piénsalo así:

```
Linux = Motor del auto
Distribución = Auto completo (con interior, ruedas, etc.)
```

Analógicamente:

- El kernel Linux es lo mismo en todas las distros
- Pero cada distro elige diferentes paquetes, herramientas, configuración

> ¿Por qué importa?
>
> - **Impacto directo:** La distro que elijas determina qué herramientas tienes disponibles
> - **En Abacom:** Elegimos Ubuntu porque tiene herramientas que necesitamos, soporte comercial, y comunidad enorme
> - **Ventaja profesional:** Un admin sabe que entre Ubuntu, CentOS y Debian, el kernel es idéntico; las diferencias son administrativas

14.4 Package Manager (Gestor de Paquetes)

Definición clara:

El "package manager" es la herramienta que usa una distribución para **instalar, actualizar y desinstalar software**. Cada familia tiene el suyo:

Listing 14.1 BASH

```
# Debian/Ubuntu: apt (Advanced Package Tool)
$ apt install nginx                                                    ①

# Red Hat/CentOS/Fedora: yum o dnf
$ yum install nginx                                                    ②
# O en Fedora:
$ dnf install nginx                                                    ③

# Arch: pacman
$ pacman -S nginx                                                      ④

# Son equivalentes - la sintaxis cambia pero el resultado es igual
```

① En **Debian/Ubuntu**, **apt install** es el comando para instalar cualquier paquete. Ubuntu está basado en Debian, por eso usa el mismo package manager.
② En **CentOS/RHEL** (derivadas de Red Hat), se usa **yum install**. La funcionalidad es idéntica a apt, pero la sintaxis y los repositorios son diferentes.
③ En **Fedora** (la versión experimental de Red Hat), se usa **dnf install**. Es una versión más moderna que yum con mejor rendimiento.
④ En **Arch Linux**, se usa **pacman -S** (la bandera **-S** significa "sincronizar/instalar"). Aunque tiene diferente sintaxis, el resultado es el mismo: nginx está instalado.

Importancia:

- Si aprendes **apt**, NO sabes **yum** automáticamente
- Pero el concepto es idéntico
- En producción, necesitas dominar el package manager de tu distro

---

14.5 Ejemplos Prácticos

14.5.1 Ejemplo 1: Identificar tu Distribución

14.5.1.1 Linux (Ubuntu)

Listing 14.2 BASH

```
# En Ubuntu
$ cat /etc/os-release                                    ①
NAME="Ubuntu"
VERSION="22.04.3 LTS"
ID=ubuntu
VERSION_CODENAME=jammy                                   ②

# También funciona:
$ lsb_release -d                                         ③
Description:    Ubuntu 22.04.3 LTS
```

① **/etc/os-release** es el archivo estándar (desde systemd) que identifica la distribución y versión
② **jammy** es el nombre código de Ubuntu 22.04 (cada versión de Ubuntu tiene un nombre animal)
③ **lsb_release -d** (Linux Standard Base) obtiene la descripción de la distribución en formato legible

14.5.1.2 macOS

① **/etc/os-release** NO existe en macOS (es un archivo específico de Linux)
② **sw_vers** es el comando equivalente en macOS para obtener información del SO
③ **uname -a** muestra información del kernel Darwin en macOS

14.5.1.3 Windows

① **Get-ComputerInfo | Select-Object** obtiene información del computador y selecciona solo OsName (nombre del SO) y OsVersion
② **systeminfo | findstr** ejecuta systeminfo y filtra (/B busca al inicio) las líneas que contienen “OS Name” o “OS Version”

Listing 14.3 BASH

```
# En macOS (Darwin)
$ cat /etc/os-release                                                    ①
# Nota: En macOS no existe /etc/os-release (es Linux)
# En su lugar:

$ sw_vers                                                                ②
ProductName:        macOS
ProductVersion:     14.2
BuildVersion:       23C64

$ uname -a                                                               ③
Darwin MacBook-Air 23.2.0 Darwin Kernel Version 23.2.0
```

Listing 14.4 POWERSHELL

```
# En Windows (PowerShell)
PS> Get-ComputerInfo | Select-Object OsName, OsVersion                   ①

OsName       OsVersion
------       ---------
Microsoft Windows 11 Pro  22H2

# También funciona:
PS> systeminfo | findstr /B /C:"OS Name" /C:"OS Version"                 ②
OS Name:                           Microsoft Windows 11 Pro
OS Version:                        10.0.22621 Build 22621
```

Explicación común:

- Cada SO tiene su propio método para identificar versión
- **Linux**: **/etc/os-release** (archivo)
- **macOS**: **sw_vers** (comando)
- **Windows**: **Get-ComputerInfo** (PowerShell)

14.5.2 Ejemplo 2: Instalar Paquete en Diferentes Entornos

14.5.2.1 Linux (Ubuntu/Debian)

① **sudo apt update** actualiza la lista local de paquetes disponibles en repositorios (sin instalar nada)
② **sudo apt install curl** descarga e instala el programa curl desde los repositorios de Ubuntu
③ **curl –version** verifica que se instaló correctamente mostrando la versión

Listing 14.5 BASH

```
# Ubuntu (apt)
$ sudo apt update                                                    ①
$ sudo apt install curl                                              ②

# Verificar instalación
$ curl --version                                                     ③
curl 7.81.0 (x86_64-pc-linux-gnu)
```

14.5.2.2 Linux (CentOS/RHEL)

Listing 14.6 BASH

```
# CentOS/RHEL (yum/dnf)
$ sudo yum update                                                    ①
$ sudo yum install curl                                              ②

# O en Fedora:
$ sudo dnf install curl                                              ③

# Verificar instalación
$ curl --version                                                     ④
curl 7.61.1 (x86_64-redhat-linux-gnu)
```

① **sudo yum update** es el equivalente en CentOS/RHEL a **apt update**
② **sudo yum install curl** instala curl usando el gestor de paquetes yum (Red Hat)
③ **sudo dnf install curl** es la versión moderna en Fedora (dnf es más rápido que yum)
④ **curl –version** verifica la instalación igual que en Ubuntu

14.5.2.3 macOS

① **brew install** instala paquetes usando Homebrew (el package manager de macOS, NO es nativo)
② **bash -c** ejecuta el script de instalación de Homebrew desde GitHub (es la forma estándar de instalar Homebrew)
③ **curl –version** verifica la instalación correcta

14.5.2.4 Windows

① **choco install** instala paquetes usando Chocolatey (el package manager más popular en Windows)
② **Set-ExecutionPolicy Bypass** permite ejecutar scripts PowerShell sin restricciones (necesario para instalar Chocolatey)

Listing 14.7 BASH

```
# macOS (Homebrew - package manager)
$ brew install curl                                                    ①

# Si Homebrew no está instalado, instalarlo primero:
$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/inst

# Verificar instalación
$ curl --version                                                       ③
curl 8.1.2 (x86_64-apple-darwin23.2.0)
```

Listing 14.8 POWERSHELL

```
# Windows (Chocolatey - package manager)
PS> choco install curl                                                 ①

# Si Chocolatey no está instalado:
PS> Set-ExecutionPolicy Bypass -Scope Process -Force; `                ②
    [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]
    iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.o

# Verificar instalación
PS> curl --version                                                     ③
# O usar el comando nativo de Windows:
PS> (Invoke-WebRequest -Uri "https://www.google.com").StatusCode       ④
200
```

③ **curl –version** verifica la instalación (curl es ahora nativo en Windows)
④ **Invoke-WebRequest** es el equivalente nativo de Windows a curl para hacer requests HTTP

Comparación: | SO | Comando | Package Manager | |—-|———|————–| | Linux (Ubuntu) | `apt install curl` | apt | | Linux (CentOS) | `yum install curl` | yum/dnf | | macOS | `brew install curl` | Homebrew | | Windows | `choco install curl` | Chocolatey |

Concepto clave: El software es el MISMO, pero la forma de instalarlo cambia por SO.

14.5.3 Ejemplo 3: Caso Real Abacom - Entornos Multi-SO

14.5.3.1 Servidor Linux (Ubuntu)

① **sudo apt update** actualiza la lista de paquetes disponibles en los repositorios de Ubuntu

Listing 14.9 BASH

```
# En nuestros servidores Ubuntu de producción
$ sudo apt update                                                    ①
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done

$ sudo apt install vim-nox                                           ②
Reading package lists... Done
Building dependency tree... Done
Setting up vim-nox (2:8.2.3455-1ubuntu1) ...

# Instalar nginx (servidor web)
$ sudo apt install nginx                                             ③
Processing triggers for ufw...

# Verificar
$ nginx -v                                                           ④
nginx version: nginx/1.18.0 (Ubuntu)

$ systemctl status nginx                                             ⑤
 nginx.service - A high performance web server
     Loaded: loaded (/lib/systemd/unit/nginx.service; enabled)
     Active: active (running)
```

② **sudo apt install vim-nox** instala vim completo (vim-nox tiene la versión completa con soporte X)
③ **sudo apt install nginx** instala nginx (popular servidor web) desde repositorios de Ubuntu
④ **nginx -v** verifica la versión instalada de nginx
⑤ **systemctl status nginx** muestra el estado del servicio nginx (must be "active running")

14.5.3.2 Estación de desarrollo macOS

① **brew install** instala múltiples paquetes a la vez (nginx, vim y curl)
② **nginx -v** verifica la versión de nginx instalada por Homebrew
③ **vim –version | head -1** muestra solo la primera línea de la salida de vim (la línea que contiene la versión)
④ **brew services list** muestra todos los servicios instalados con Homebrew y su estado (en macOS, reemplaza a systemctl)

14.5.3.3 Máquina Windows (Administración Remota)

Listing 14.10 BASH

```
# En máquinas de desarrollo (macOS)
$ brew install nginx vim curl                                    ①

# Verificar instalación
$ nginx -v                                                       ②
nginx version: nginx/1.25.3

$ vim --version | head -1                                        ③
VIM - Vi IMproved 9.0

# En macOS, no usamos systemctl (es systemd de Linux)
# En su lugar, usamos launchctl o brew services:
$ brew services list                                             ④
Name  Status  User Plist
nginx started root /Library/LaunchDaemons/homebrew.mxcl.nginx.plist
```

① **ssh admin@production-server.com** conecta a un servidor Linux remoto mediante SSH (está disponible por defecto en Windows 10+)
② **sudo apt install** instala paquetes en el servidor remoto (ejecutado desde Windows a través de SSH)
③ **systemctl status nginx** verifica el estado del servicio nginx en el servidor remoto
④ **Invoke-WebRequest** es el equivalente de Windows a curl, haciendo un request HTTP a la web del servidor

Por qué es importante Abacom:

- **Servidor:** Ubuntu (producción) - stabiliidad
- **Desarrollo:** macOS (portátil) - herramientas modernas
- **Administración:** Windows (oficina) - acceso remoto SSH

14.6 Familias Principales de Distribuciones

14.6.1 Familia Debian

La más grande, con comunidad enorme:

① **lsb_release -d** (Linux Standard Base release) muestra la versión de Debian en formato legible (12 = Debian 12, "bookworm" es el nombre código)

Distribuciones derivadas de Debian:

- Ubuntu (+ soporte comercial, + user-friendly)
- Linux Mint (+ interfaz gráfica bonita)

Listing 14.11 POWERSHELL

```
# En Windows, administramos remotamente servidores Linux
PS> # Usando SSH para conectar al servidor Ubuntu
PS> ssh admin@production-server.com                                    ①

# Una vez conectado (en el servidor Linux):
$ sudo apt install vim nginx curl                                      ②
$ systemctl status nginx                                               ③

# Desde Windows, también podemos:
PS> Invoke-WebRequest -Uri "http://production-server.com" | Select-Object StatusCode ④
StatusCode
----------
200
```

Listing 14.12 BASH

```
# Características de Debian:
# - Versiones: Stable (3 años), Testing, Unstable
# - Package manager: apt (Advanced Package Tool)
# - Versión actual stable: 12 (Bookworm)
# - Enfoque: Estabilidad pura

$ lsb_release -d                                                       ①
Description:    Debian GNU/Linux 12 (bookworm)
```

- Pop!_OS (+ enfoque gaming)

Ventaja: Millones de paquetes disponibles

14.6.2 Familia Red Hat / RHEL

Enfocado en empresas:

① **cat /etc/os-release | grep NAME** muestra el archivo /etc/os-release y filtra solo la línea que contiene el nombre de la distribución

Distribuciones en esta familia:

- RHEL (Red Hat Enterprise Linux) - Pagado, soporte profesional
- CentOS (Community Enterprise OS) - Gratis, mismo que RHEL
- Fedora (El laboratorio de RHEL) - Cambios rápidos, más experimental

Ventaja: Soporte comercial de Red Hat

Listing 14.13 BASH

```
# Características:
# - Versiones: RHEL (10 años soporte), Fedora (6 meses)
# - Package manager: yum/dnf (Fedora uses dnf)
# - Enfoque: Estabilidad + soporte comercial

$ cat /etc/os-release | grep NAME                                    ①
NAME="Red Hat Enterprise Linux"
```

14.6.3 Familia Arch

Para usuarios avanzados:

Listing 14.14 BASH

```
# Características:
# - Rolling release (actualizaciones continuas)
# - Package manager: pacman
# - Filosofía: K.I.S.S (Keep It Simple, Stupid)
# - Enfoque: Control total, minimalismo

$ cat /etc/os-release                                                ①
NAME="Arch Linux"
```

① **cat /etc/os-release** muestra el archivo que identifica la distribución, en este caso Arch Linux

Cuándo usarla:

- Cuando necesitas última versión de software
- Cuando sabes exactamente qué necesitas
- Cuando quieres aprender cómo funciona todo

NO es para producción - cambios demasiado frecuentes

14.7 Comparativa de Distribuciones Principales

14.7.1 Ubuntu vs CentOS vs Debian

⚠ Tabla comparativa técnica

Aspecto	Ubuntu	CentOS	Debian
Base	Debian	Red Hat	Original
Release Cycle	6 meses (LTS: 5 años)	7 años	2-3 años
Package Manager	apt	yum/dnf	apt
Soporte Comercial	Canonical	Red Hat	Comunidad
Uso Empresarial	Altísimo	Altísimo	Medio
Cloud	AWS preferred	Red Hat Cloud	Bajo
Servidor web	Apache/Nginx	Apache	Apache/Nginx
Para Abacom	Excelente	No instalado	No usado

Análisis de decisión:

Listing 14.15 BASH

```
# UBUNTU 22.04 LTS
# Ventajas:
# - Soporte de Canonical hasta Abril 2027
# - Herramientas listas (apt, systemd)
# - Cloud: Soportado en AWS, Azure, Google Cloud
# - Comunidad: Enorme, miles de tutoriales
# - Servidor: Diseñado específicamente para servidores

# CENTOS
# Ventajas:
# - Derivado de RHEL (enterprise-grade)
# - Soporte largo (7 años)
# Desventajas:
# - Más complejo de administrar
# - Ecosystem de Red Hat puede ser caro

# DEBIAN
# Ventajas:
# - Estable (testing stable es MUY estable)
# - Minimalista
# Desventajas:
# - Release cycle largo (puede tener software viejo)
# - Comunidad, no comercial
```

14.8 Errores Comunes

14.8.1 "Necesito cambiar de distribución si mi código está en Debian"

Síntoma que verás

```
"Mi código está en Ubuntu, pero necesito correrlo en CentOS"
"¿Debo reescribir todo?"
```

Causa raíz: Malentendido. El kernel y el código son independientes de la distro.
Solución comprobada:

Listing 14.16 BASH

```
#  Forma INCORRECTA (pensar que el código cambia)
# "Ubuntu code" vs "CentOS code" - NO EXISTE

#  Forma CORRECTA (código es código)
# Un programa escrito en Python en Ubuntu
# Corre igual en CentOS, Debian, etc.

# Ejemplo: Script en Python
$ cat script.py                                                    ①
#!/usr/bin/env python3
print("Hello from any Linux!")

# Corre igual en cualquier distro
$ python3 script.py                                                ②
Hello from any Linux!
```

① **cat script.py** muestra el contenido del archivo script.py (un simple programa Python)
② **python3 script.py** ejecuta el script Python (funciona IGUAL en Ubuntu, CentOS, Debian, etc.)

Por qué funciona: El código sigue estándares POSIX. La única diferencia es el package manager (apt vs yum), pero el código sigue siendo igual.

14.8.2 "Mi distro no tiene el paquete X"

⚠ Warning

Síntoma: apt search nginx devuelve vacío, o versión vieja
Causa: Dependencias de distribución o versiones viejas en release cycle largo
Solución:

Listing 14.17 BASH

```
# Agregar repositorio externo (PPA en Ubuntu)
$ sudo add-apt-repository ppa:nginx/stable                    ①
$ sudo apt update                                             ②
$ sudo apt install nginx                                      ③

# O compilar desde source (más complicado)
# Pero la idea: siempre hay solución
```

① **sudo add-apt-repository** agrega un repositorio externo (PPA = Personal Package Archive) que contiene versiones más nuevas
② **sudo apt update** actualiza la lista de paquetes locales para incluir los del nuevo repositorio
③ **sudo apt install nginx** instala la versión más nueva desde el repositorio externo agregado

---

14.9 Mejores Prácticas

❗ Patrones Recomendados

HACER (Prácticas Probadas):

- Usar Ubuntu LTS en producción → soporte 5 años
- Usar CentOS/RHEL si tienes soporte de Red Hat → enterprise-grade
- Aprender el package manager de tu distro → gestión eficiente
- Mantener consistencia en todo el datacenter → administración simplificada

NO HACER (Antipatrones):

- Mezclar distros sin razón → complejidad innecesaria
- Usar Fedora en producción → cambios muy frecuentes
- Ignorar que existen distros alternativas → inflexibilidad

14.10 Tabla de Distribuciones Populares

Distro	Tipo	Base	Uso	Soporte
Ubuntu	LTS	Debian	Servidores/Desktop	Canonical (5 años)
CentOS	Enterprise	RHEL	Servidores	Red Hat (7 años)
Debian	Stable	Original	Servidores/Desktop	Comunidad (2-3 años)
Fedora	Rolling	RHEL	Desktop/Desarrollo	13 meses
Arch	Rolling	Original	Avanzados	Comunidad
Mint	Desktop	Ubuntu	Desktop	Comunidad

14.11 Quiz: Verifica tu Comprensión

> **Antes de continuar**
>
> Responde estas preguntas para verificar que comprendiste los conceptos clave.

14.11.1 ¿Cuál es la diferencia entre Ubuntu y CentOS?

Ver respuesta

Ambos son distribuciones Linux con kernel Linux, pero:

- **Ubuntu:** Basado en Debian, package manager **apt**, soporte de Canonical
  - **CentOS:** Basado en RHEL, package manager **yum/dnf**, soporte de Red Hat

"'bash # Misma funcionalidad, diferente package manager # Ubuntu: $ apt install nginx

15 CentOS:

$ yum install nginx

16 Resultado: nginx está instalado en ambos

```
  **Por qué es importante:**
  El package manager es la principal diferencia visible. El kernel, el shell, los comando
  </summary>
</details>

### ¿Cuál es el comando para identificar tu distribución?

<details>
  <summary>  Ver respuesta</summary>

  ```bash
  $ cat /etc/os-release
# O más simple:
  $ lsb_release -d
  Description:    Ubuntu 22.04.3 LTS
```

Validación: Si ejecutas esto, verás exactamente:

- Nombre de la distro
  - Versión específica
  - ID de versión (nombre código)

  Útil cuando:

  - Heredas un servidor desconocido
  - Necesitas verificar versión para soporte
  - Escribes scripts que dependen de distro específica

16.0.1 ¿Por qué Abacom elige Ubuntu 22.04 LTS en lugar de CentOS?

Ver respuesta

Ubuntu 22.04 LTS es mejor para Abacom porque:

- LTS = 5 años de soporte (hasta Abril 2027)
  - **apt** es más simple que **yum**
  - Comunidad enorme = fácil encontrar ayuda
  - Cloud-native (AWS, Azure, Google Cloud)
  - Herramientas modernas listas (systemd, etc.)
```

Comparación:

Listing 16.1 BASH

```
# Ubuntu: Lo que necesitamos
$ apt update && apt upgrade  # Fácil

# CentOS: Similar pero sintaxis diferente
$ yum update && yum upgrade  # Equivalente
```

Razón de negocio:

- Soporte de Canonical disponible si lo necesitamos
- Comunidad más grande para troubleshooting
- Mejor integración con herramientas modernas

16.1 Práctica Guiada con el Instructor

16.1.1 Identificar tu Distribución

Objetivo: Conocer exactamente qué distribución tienes.

Instrucciones:

Listing 16.2 BASH

```
# Paso 1: Ver distribución y versión
$ cat /etc/os-release

# Paso 2: Ver más detalles

$ lsb_release -a

# Paso 3: Ver package manager

$ apt --version    # Si es Ubuntu/Debian
# O
$ yum --version    # Si es CentOS/RHEL
```

Verificación:

- Sabes el nombre de tu distro
- Sabes la versión exacta
- Sabes qué package manager usar

16.1.2 Comparar Diferentes Distros (Teórico)

Objetivo: Entender diferencias sin necesidad de instalar.

Instrucciones:

Listing 16.3 BASH

```
# Crear tabla de comparación
$ cat > /tmp/distros-comparacion.txt << 'EOF'
=== COMPARACIÓN DE DISTRIBUCIONES ===

UBUNTU 22.04 LTS:

- Base: Debian
- Package manager: apt
- Release cycle: 6 meses (LTS: 5 años)
- Soporte: Canonical
- Ideal para: Servidores empresariales

CENTOS 7:

- Base: RHEL
- Package manager: yum
- Release cycle: 7 años
- Soporte: Red Hat
- Ideal para: Infraestructura legacy

DEBIAN 12:

- Base: Original
- Package manager: apt
- Release cycle: 2-3 años
- Soporte: Comunidad
- Ideal para: Estabilidad pura
EOF

$ cat /tmp/distros-comparacion.txt
```

Ejercicio:

1. ¿Qué distro está en tu servidor?
2. ¿Cuánto tiempo de soporte tiene?
3. ¿Cuál elegirías para nuevo servidor?

16.1.3 Administrar Paquetes

Objetivo: Practicar instalación en tu distro.

Instrucciones:

Listing 16.4 BASH

```
# Paso 1: Actualizar lista de paquetes

$ sudo apt update    # Ubuntu/Debian

# Paso 2: Ver versiones disponibles

$ apt search curl | grep "^curl"

# Paso 3: Instalar

$ sudo apt install curl -y

# Paso 4: Verificar

$ curl --version

# Paso 5: Desinstalar (limpieza)

$ sudo apt remove curl -y
```

Verificación:

- Paquete se instaló correctamente
- Comando **curl –version** funciona
- Entiendes el flujo: update → search → install

16.2 Laboratorio de Práctica Integrado

Duración: 60-90 minutos
Dificultad: Intermedio
Objetivo General: Entender ecosistema de distribuciones y package management

16.2.1 Fase 1: Inventario del Sistema (15 min)

16.2.2 Fase 2: Explorar Package Manager (20 min)

Listing 16.5 BASH

```
$ mkdir -p ~/lab-distros
$ cd ~/lab-distros

# Recopilar información
$ cat /etc/os-release > mi-distro.txt
$ lsb_release -a >> mi-distro.txt
$ apt --version >> mi-distro.txt
$ uname -a >> mi-distro.txt

$ cat mi-distro.txt
```

Listing 16.6 BASH

```
# Listar paquetes instalados
$ dpkg --list > paquetes-instalados.txt
$ wc -l paquetes-instalados.txt
# ¿Cuántos paquetes tienes instalados?

# Ver información de paquete específico
$ apt show curl > info-curl.txt
$ cat info-curl.txt
# ¿Qué versión hay disponible?
```

16.2.3 Fase 3: Análisis Comparativo (20 min)

16.2.4 Fase 4: Documento Final para Abacom (15 min)

16.2.5 Verificación del Laboratorio

- ☐ Información de distro recopilada
- ☐ Package manager explorado
- ☐ Análisis comparativo completado
- ☐ Reporte final escrito
- ☐ Entiendo por qué usamos Ubuntu en Abacom

Listing 16.7 BASH

```
# Crear documento comparativo
$ cat > analisis-distros.txt << 'EOF'
=== ANÁLISIS: ¿QUÉ DISTRO USAR? ===

Mi distro actual: Ubuntu 22.04 LTS

¿Por qué Ubuntu?

1. Base: Debian (estable)

2. LTS: Soporte 5 años (hasta 2027)

3. Package manager: apt (fácil)

4. Comunidad: Enorme (muchos tutoriales)

5. Cloud: Soportado en AWS, Azure, GCP

¿Cuándo cambiaría?

- A CentOS: Si necesitara soporte Red Hat empresarial
- A Debian: Si quisiera estabilidad pura (sin extras)
- A Fedora: Nunca en producción (cambios rápidos)
EOF

$ cat analisis-distros.txt
```

16.3 Recursos Adicionales

16.3.1 Documentación Oficial

- **DistroWatch**
  - Sección: Últimas noticias, compares de distros
  - Por qué: Base de datos de 300+ distribuciones
- **Ubuntu Server Documentation**
  - Sección: Administración de paquetes
  - Por qué: Documentación oficial de ubuntu
- **The Linux Foundation**
  - Sección: Ecosistema de distribuciones
  - Por qué: Información neutral sobre todas las distros

16.3.2 Tutoriales Complementarios

- **Ubuntu Server Docs - Package management**
  - Duración: 10 minutos
  - Cubre: Instalación básica con apt

16.3.3 Comunidades

- **Ubuntu Forums**
  - Para: Preguntas sobre Ubuntu
  - Cómo: Comunidad oficial
- **r/linux - Reddit**
  - Para: Debates sobre distros
  - Cómo: Comunidad receptiva

16.4 Preguntas Frecuentes

P: ¿Cuál es la mejor distribución?

R: No hay "mejor", pero hay "mejor para caso X":

- **Producción empresarial:** Ubuntu LTS o CentOS/RHEL
- **Aprender Linux:** Ubuntu, Debian
- **Desarrollo:** Fedora, Arch
- **Escritorio:** Linux Mint, Pop!_OS
- **IoT:** Alpine, Raspbian

P: ¿Puedo cambiar de distribución sin perder datos?

R: Sí y no:

- El kernel y comandos son similares
- Pero migrar de Ubuntu a CentOS requiere reconfiguraciones
- Mejor: backup + instalación limpia + restore de datos

P: ¿En Abacom alguna vez cambiaremos de Ubuntu?

R: Posible, pero poco probable:

- Ubuntu funciona bien
- Comunidad es grande
- Soporte es bueno
- Cambiar costaría reentrenamiento

16.5 Resumen

Lo más importante (3 puntos clave)

1. **Distribución = Kernel Linux + software + configuración** → El kernel es igual, pero cada distro agrega herramientas diferentes
2. **Package manager es la principal diferencia** → **apt** (Ubuntu/Debian) vs **yum** (CentOS/RHEL) son las opciones principales
3. **Ubuntu 22.04 LTS es ideal para Abacom** → LTS = 5 años de soporte, **apt** es simple, comunidad es enorme

Recuerda: Aprender a administrar paquetes es fundamental para tu rol como admin.

16.5.1 Checklist de Competencias Alcanzadas

- Entiendo qué es una distribución de Linux
- Puedo identificar mi distribución actual
- Conozco diferencias entre familias principales
- Entiendo por qué Ubuntu 22.04 para Abacom
- Puedo instalar y desinstalar paquetes
- Estoy listo para aprender instalación

16.5.2 Próximos Pasos

Note

Siguiente tema: Ventajas de Linux para Abacom
Laboratorio siguiente: Instalación de Ubuntu 22.04 LTS
Tiempo recomendado: 1.5 horas de lectura + 2 horas de práctica

---

16.6 Soporte

¿Tienes dudas sobre este tema?

- **Preguntas técnicas:** GitHub Issues
- **Comparaciones de distros:** Verifica DistroWatch
- **Sugerencias:** Envía feedback

Última actualización: 2026-01-29
Versión: 1.0
Estado: Completo y validado contra fuentes oficiales

> **Acerca de las fuentes**
>
> Verificado contra:
>
> - DistroWatch official database
> - Ubuntu Server documentation
> - The Linux Foundation ecosystem guide

16.7 Quiz: Distribuciones de Linux

Listing 16.8 BASH

```
$ cat > reporte-distribucion.txt << 'EOF'
=== REPORTE: DISTRIBUCIÓN EN ABACOM ===

1. DISTRIBUCIÓN ACTUAL:

$(cat /etc/os-release | grep "PRETTY_NAME")

2. INFORMACIÓN TÉCNICA:

- Release: Ubuntu 22.04 LTS (Jammy Jellyfish)
- Soporte: Hasta Abril 2027
- Package Manager: apt
- Kernel: $(uname -r)

3. PAQUETES INSTALADOS:

- Total: $(dpkg --list | wc -l)
- Actualización: $(date)

4. RECOMENDACIÓN:

  Mantener Ubuntu 22.04 LTS hasta fin de soporte (2027)
  Aplicar parches de seguridad mensualmente
  No cambiar a otra distro sin razón técnica

5. ALTERNATIVAS CONSIDERADAS:

- CentOS: No necesario (Ubuntu más fácil)
- Debian: Viable pero Ubuntu es superset
- Fedora: No recomendado (cambios rápidos)
EOF

$ cat reporte-distribucion.txt
```

17 Unidad 1.4: Ventajas de Linux para Abacom

Listing 17.1 QUARTO-TITLE-BLOCK

18 Ventajas de Linux para Abacom

18.1 Introducción

En las secciones anteriores aprendiste QUÉ es Linux. Ahora vamos a explorar POR QUÉ elegimos Linux para Abacom. No es casualidad que más del 90% de los servidores mundiales usen Linux - existen razones técnicas y comerciales sólidas. En este tema, analizaremos estas ventajas en el contexto específico de nuestra empresa.

Tiempo de lectura: ~15 minutos
Nivel: Principiante-Intermedio
Requisitos previos: Unidades 1-3 (SO, Historia, Distribuciones)

18.2 En este tema aprenderás

En esta unidad cubriremos:

1. **Ventaja Económica** - Linux es completamente gratuito
2. **Código Abierto** - Libertad, transparencia, control total
3. **Seguridad** - Parches rápidos, comunidad audita código
4. **Estabilidad y Rendimiento** - Servidores con 10+ años uptime
5. **Comunidad Global** - Millones de expertos disponibles
6. **Compatibilidad Empresarial** - Soporte profesional disponible
7. **Casos de Éxito** - Empresas que usan Linux

18.3 Gratuidad Baja Calidad

Definición clara:

Linux es completamente **gratis** (*GNU GPL V3.0* 2007). No pagas por licencias, actualizaciones, ni extensiones. Pero esto NO significa baja calidad. De hecho, es lo opuesto.

¿Por qué importa?

- **Impacto directo:** Abacom ahorra millones en licencias de software
- **En Abacom:** En lugar de pagar a Microsoft/Red Hat, reinvertimos en talento
- **Ventaja profesional:** Entiendes que "gratis" y "calidad" NO son opuestos

18.4 Código Abierto = Control

Definición clara:

Con Linux, **tienes acceso al código fuente completo** (T. L. Foundation 2024). Puedes:

- Ver exactamente qué hace
- Modificarlo si lo necesitas
- Auditar seguridad por ti mismo
- Nunca depender de un vendedor

Listing 18.1 BASH

```
# Puedes ver el código de cualquier parte de Linux
$ cat /proc/version                                                    ①
Linux version 5.15.0-89-generic (gcc-11)

# O descargar el kernel completo
$ git clone https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git ②
# Contiene 30+ años de historia de desarrollo
```

① **cat /proc/version** muestra el archivo virtual **/proc/version** que contiene la versión exacta del kernel en ejecución

② **git clone** descarga el repositorio completo del kernel Linux desde el servidor oficial, incluyendo 30+ años de historia de commits de Linus Torvalds y miles de desarrolladores

Punto importante

Código abierto NO significa "sin seguridad". Significa lo opuesto:

- Miles de expertos auditan constantemente
- Vulnerabilidades se encuentran Y se arreglan rápido
- No hay "puertas traseras" ocultas

18.5 Ejemplos Prácticos

18.5.1 Ejemplo 1: Comparar Costos en Diferentes SOs

18.5.1.1 Linux (Ubuntu Server)

Listing 18.2 BASH

```
# OPCIÓN: Linux (Ubuntu LTS)
# Licencia:                          $0 (gratis)                          (1)
# Soporte (Canonical):              $200-1,000 (opcional)
# Actualizaciones:                  Ilimitadas, automatizables
# Cambiar características:          Libre, no pagado
# TOTAL ANUAL:                      ~$200-1,000 (máximo)

# En producción:
$ lsb_release -d
Description:    Ubuntu 22.04.3 LTS
# Servidor pagado: $0
# Soporte pagado: $0-500/año (opcional)
```

(1) Linux no tiene costo inicial para instalar en ningún número de servidores

18.5.1.2 Windows Server

Listing 18.3 POWERSHELL

```
# OPCIÓN: Windows Server 2022
# Licencia:                         $5,000-20,000 (una vez)              (1)
# Soporte anual:                    $1,000-5,000
# Actualizaciones:                  Incluido pero limitado
# Cambiar licencia:                 Complejo y caro
# TOTAL ANUAL:                      ~$10,000+ (mínimo)

PS> Get-ComputerInfo | Select-Object OsName
OsName
------
Microsoft Windows Server 2022
# Costo inicial: $15,000
# Costo anual: $5,000+
```

(1) Windows Server requiere pagar una licencia inicial cara, además de soporte continuo obligatorio

18.5.1.3 macOS Server

Listing 18.4 BASH

```
# OPCIÓN: macOS Server (deprecated, NO RECOMENDADO)
# Licencia:                    Incluida en hardware
# Hardware:                    $3,000-6,000 (Mac Mini/Mac Studio)
# Soporte anual:               AppleCare: $500-1,000
# Actualizaciones:             Incluidas pero limitadas
# TOTAL ANUAL:                 ~$1,000+ (solo costo SO)

$ sw_vers
ProductName:        macOS
ProductVersion:     14.2
# Nota: macOS Server fue discontinued por Apple en 2023
# Apple NOW recomienda usar Linux para servidores
```

Comparación clara: | SO | Costo Licencia | Soporte Anual | Total/Año | |—-|—|—|—| | Linux | $0 | $0-1K (opt) | $0-1K | | Windows | $15K | $5K | $20K | | macOS | Hardware $4K | $1K | $5K+ |

GANADOR: Linux (95% más barato)

18.5.2 Ejemplo 2: Seguridad mediante Código Abierto vs Cerrado

18.5.2.1 Linux (Código Abierto)

Listing 18.5 BASH

```
# Cuando se descubre CVE en Linux:

# 1. Se reporta privadamente a kernel security team
# 2. Desarrollo colaborativo en parches (PÚBLICO, en GitHub)          (1)
# 3. Se audita por miles de ojos (Google, Intel, Red Hat, etc.)
# 4. Se publica y todos reciben parches al mismo tiempo
# Tiempo típico: 1-4 semanas

# Ver vulnerabilidades públicamente:
$ curl https://security.ubuntu.com/usn/usn-db.json | head -20
# Todos pueden ver y auditar parches

# Patch automático:
$ sudo apt update && sudo apt upgrade -y
# Aplica parches de seguridad verificados por miles de expertos
```

① En Linux, el proceso de parches es transparente. Puedes ver exactamente qué se está parcheando

18.5.2.2 Windows (Código Cerrado)

Listing 18.6 POWERSHELL

```
# Cuando se descubre CVE en Windows:

# 1. Se reporta a Microsoft (privado)
# 2. Microsoft desarrolla parche internamente (secreto)
# 3. Nadie ajeno audita el parche (solo Microsoft)
# 4. Se publica "Patch Tuesday" (segundo martes del mes)
# Tiempo: Puede ser MESES si es baja prioridad

# En PowerShell:
PS> Get-HotFix | Select-Object Description, InstalledOn | head          ①
# Solo ves CUÁNDO se instaló, NO CÓMO se arreglaron las vulnerabilidades

# Actualización FORZADA:
PS> Get-WindowsUpdate
# Windows actualiza en Patch Tuesday sin poder evitarlo
```

① En Windows, no tienes acceso al código del parche. Debes confiar ciegamente en Microsoft

18.5.2.3 macOS (Código Parcialmente Cerrado)

Listing 18.7 BASH

```
# macOS (basado en Unix/Darwin):

# Vulnerabilidades:
# 1. Reportadas a Apple (privado)
# 2. Apple desarrolla parches (semi-secreto)
# 3. Comunidad puede auditar pero no tan transparente como Linux
# 4. Actualizaciones forzadas en updates

$ softwareupdate -l
# Ver actualizaciones disponibles

$ sudo softwareupdate -ia
# Instalar todo (a menudo FORZADO)
# macOS requiere reinicio para actualizaciones
```

Comparación de seguridad: | Aspecto | Linux | Windows | macOS | |———|——-|———|——-| | Código visible | 100% | 0% | 10% | | Auditoría pública | Sí | No | Limitada | | Tiempo a parche | 1-4 sem | 1-3 meses | 2-6 sem | | Puertas traseras | Imposible | Posible | Posible |

GANADOR: Linux (total transparencia)

18.5.3 Ejemplo 3: Caso Real Abacom - Estabilidad y Uptime

18.5.3.1 Servidor Linux de Abacom

Listing 18.8 BASH

```
# Nuestros servidores de producción (Ubuntu 22.04 LTS):

$ uptime                                                                    ①
 14:45:23 up 247 days, 5:12, 2 users, load average: 0.85, 0.92, 0.88

# Traducción: Servidor ha estado corriendo SIN REINICIO por 247 días

$ systemctl status nginx
 nginx.service - A high performance web server
     Loaded: loaded (/lib/systemd/unit/nginx.service; enabled)
     Active: active (running) since Thu Jan 29 15:30:00 2026

$ uptime > /tmp/uptime-history.txt
$ cat /tmp/uptime-history.txt
# Sin reinicios planificados en todo 2024
```

① 247 días sin reinicio = ~8 meses de funcionamiento continuo sin problemas

18.5.3.2 Servidor Windows de Comparación

① Windows requiere reinicios mucho más frecuentemente

18.5.3.3 Infraestructura comparativa de Abacom

GANADOR: Linux (99.97% vs 99.60% disponibilidad)

Listing 18.9 POWERSHELL

```
# Si tuviéramos un servidor Windows:

PS> (Get-Date) - (Get-CimInstance Win32_OperatingSystem).LastBootUpTime ①
# Días desde último reinicio: 23 días

PS> Get-HotFix | Measure-Object
Count: 47
# 47 parches aplicados en 23 días

# ¿Por qué solo 23 días?
# - Patch Tuesday (segundo martes): Reinicio forzado
# - Actualizaciones de drivers: Reinicio forzado
# - Cambios de configuración: A menudo requieren reinicio
# - Antivirus updates: Algunos requieren reinicio

# En Windows es NORMAL reiniciar cada 30 días
```

18.6 Ventajas Detalladas

18.6.1 Económica

Linux es **completamente gratuito**:

① No hay costo inicial para instalar Linux en ningún número de servidores
② Todas las actualizaciones del kernel y sistema operativo son completamente gratuitas y automáticas
③ La comunidad global de Linux proporciona soporte a través de foros, documentación y mailing lists, sin costo
④ Puedes usar Linux en servidores comerciales sin pagar licencias (solo pagas si quieres soporte profesional de Canonical o Red Hat)

Comparativa real:

- Windows Server: $5,000-20,000 licencia + $1,000-5,000 soporte anual
- Red Hat RHEL: $1,000-3,000 per server per year
- Ubuntu LTS: Gratis (o $200-1,000 soporte profesional opcional)

18.6.2 Seguridad

Código abierto = mejor seguridad:

① Un CVE (Common Vulnerabilities and Exposures) es un identificador único para una vulnerabilidad de seguridad
② Windows Server reporta en promedio 25-30 vulnerabilidades nuevas detectadas cada mes

Listing 18.10 BASH

```
# LINUX (Abacom actual):
# - Uptime promedio: 180+ días
# - Reinicios anuales: 1-2 (por actualización kernel)
# - Tiempo de inactividad: ~30 minutos/año

# WINDOWS (si lo usáramos):
# - Uptime promedio: 20-30 días
# - Reinicios anuales: 12+ (Patch Tuesday)
# - Tiempo de inactividad: ~400 minutos/año (6+ horas)

$ cat > uptime-comparison.txt << 'EOF'
UPTIME AVAILABILITY:

- Linux 180 días: 99.97% disponibilidad
- Windows 23 días: 99.60% disponibilidad
- DIFERENCIA: 0.37% = ~130 horas/año de inactividad adicional en Windows
EOF

$ cat uptime-comparison.txt
```

Listing 18.11 BASH

```
# Instalación: Gratis                                                    ①
# Actualizaciones: Gratis                                                ②
# Soporte comunitario: Gratis                                            ③
# Uso comercial: Gratis (GPL permite)                                    ④
# Modificaciones: Gratis
# Redistribución: Gratis (con condiciones GPL)
```

③ Linux reporta en promedio 15-20 vulnerabilidades nuevas detectadas cada mes, A PESAR de ser más usado en servidores

④ La diferencia crucial: en Linux, estas vulnerabilidades se descubren Y arreglan más rápido porque el código es público y miles de expertos lo auditan constantemente

Ventajas de seguridad:

- Auditoría pública (miles de ojos)
- Parches rápidos (comunidad colaborativa)
- Sin "puertas traseras" (código abierto)
- Cumplimiento normativo (GDPR, ISO, etc.)

18.6.3 Estabilidad

Los servidores Linux tienen uptimes increíbles:

Listing 18.12 BASH

```
# CVE (Common Vulnerabilities and Exposures)
# Ejemplo de datos reales:                                                    ①

# Windows Server: ~25-30 CVEs por mes                                         ②
# Linux Kernel: ~15-20 CVEs por mes                                           ③

# ¿Cómo es posible si Linux es más usado?
# Respuesta: En Linux, vulnerabilidades se encuentran Y arreglan MÁS RÁPIDO ④
# porque hay miles auditing el código constantemente
```

Listing 18.13 BASH

```
# Ejemplos reales de servidores en producción:
$ uptime                                                                      ①
# 5 años sin reinicio (cumpleaños de servidor)

# En Windows, necesitas reiniciar:                                            ②
# - Cada patch mensual (Patch Tuesday)
# - Actualizaciones de drivers
# - Cambios de configuración

# En Linux:                                                                   ③
# - Actualizaciones sin reinicios (en vivo)
# - Cambios de configuración sin reinicios (en vivo)
# - Reinicios solo cuando cambias kernel (anual si acaso)
```

① Es común en Linux ver servidores con uptime de 5+ años sin un solo reinicio (algunos casos conocidos de 10+ años). El comando **uptime** muestra exactamente cuánto tiempo lleva activo.

② Windows Server requiere reinicios forzados cada mes (Patch Tuesday, segundo martes del mes), además de reinicios adicionales por drivers u otros cambios

③ Linux permite actualizar el sistema operativo, aplicaciones, incluso drivers SIN reinicios, usando técnicas como live patching. Los reinicios son raros y planificados, no forzados

18.6.4 Rendimiento

Linux usa menos recursos:

① Ubuntu Server es extremadamente ligero: toma solo 2 GB de espacio en disco y puede funcionar con 512 MB de RAM, usando apenas 100 MB de RAM cuando está inactivo

② Windows Server requiere 20 GB de disco mínimo y 2-4 GB de RAM como recomendación oficial, consumiendo mucho más incluso idle

Listing 18.14 BASH

```
# Ubuntu Server install size:                                                    ①
# ~2 GB (disco mínimo)
# ~512 MB RAM (funcionable)
# ~100 MB running idle

# Windows Server install:                                                        ②
# ~20 GB (disco mínimo)
# ~2-4 GB RAM (mínimo recomendado)
# ~500 MB - 1GB running idle

# RESULTADO: Linux = más eficiente con recursos                                  ③
# = más aplicaciones por servidor
# = menos hardware requerido
# = menos costos en infraestructura
```

③ Esta eficiencia de recursos de Linux permite instalar más aplicaciones por servidor, reducir hardware, y ahorrar en costos de infraestructura (energía, refrigeración, espacio)

```
### Comunidad

Millones de desarrolladores colaboran:

```bash
# Contribuidores al kernel Linux:
# - 18,000+ archivos
# - 28+ millones de líneas de código
# - 2,000+ nuevas características por versión
# - +2,000 contribuyentes activos

# Ayuda disponible:
# - Stack Overflow: 2+ millones de preguntas Linux
# - GitHub: Código de ejemplo para cualquier cosa
# - Reddit: r/linux con 800k+ usuarios
# - Listas de correo: Linux-kernel con expertos de Intel, Google, Red Hat

# RESULTADO: Si tienes problema, alguien ya lo resolvió
```

18.6.5 Flexibilidad

Personalización completa:
```

Listing 18.15 BASH

```
# Con Linux, puedes:
$ # Compilar kernel solo con características que necesitas
$ # Retirar drivers innecesarios
$ # Optimizar para tu hardware específico
$ # Crear distribuciones personalizadas

# Con Windows, no puedes quitarle:
# - Internet Explorer
# - Cortana
# - OneDrive
# - Telemetría de Microsoft
# - Características que no necesitas
```

18.7 Errores Comunes

18.7.1 "Linux es solo para geeks/hackers"

Síntoma que verás

```
"Linux es muy complicado, mejor Windows que cualquier persona entiende"
```

Causa raíz: Confundir interfaz gráfica fácil con software empresarial.
Solución comprobada:

Listing 18.16 BASH

```
#   Forma INCORRECTA (pensar que Linux es difícil)
# "Windows tiene interfaz gráfica, Linux es solo terminal"

#   Forma CORRECTA (Linux también tiene GUI)
# Ubuntu Desktop tiene interfaz gráfica IGUAL que Windows
# Pero Ubuntu Server (en producción) usa terminal porque es MÁS EFICIENTE

# Comparación:
# - Windows Server: GUI + CLI (ambas disponibles)
# - Linux Server: CLI (GUI disponible pero no se usa)
#
# Razón: En producción, GUI consume recursos y ralentiza
# Terminal es más rápido, más confiable, más escalable
```

Por qué funciona: En producción (servidores), no necesitas interfaz gráfica. CLI es MÁS poderosa que GUI. Eso no la hace difícil, la hace más eficiente.

18.7.2 "Pero todas las grandes empresas usan Windows"

⚠ Warning

Síntoma: "Google, Facebook, Amazon deben usar Windows"
Causa: Confundir servidores (backend) con desktops (frontend)
Solución:

Listing 18.17 BASH

```
# REALIDAD:
# Google: 1+ millones de servidores Linux
# Facebook: Millones de servidores Linux
# Amazon AWS: Optimizado para Linux
# Netflix: Casi todo en Linux
# Microsoft Azure: Soporta más VMs Linux que Windows

# Desktops corporativos pueden ser Windows
# Pero servidores MASIVAMENTE Linux

# En Abacom:
# Desktops de usuarios: Windows o Mac (aplicaciones específicas)
# Servidores de aplicaciones: Linux (costo + rendimiento)
```

18.8 Mejores Prácticas

❗ Patrones Recomendados

HACER (Prácticas Probadas):

- Usar Linux en producción → costo + rendimiento
- Aprovechar comunidad → miles de soluciones probadas
- Medir ROI → Linux paga por sí solo en economía
- Invertir en capacitación → talento es el único costo real

NO HACER (Antipatrones):

- Pensar que gratis = baja calidad → falso para Linux
- Temer código abierto → lo opuesto, es más seguro
- Ignorar soporte profesional → disponible si lo necesitas

18.9 Tabla Comparativa: Linux vs Windows vs macOS

Aspecto	Linux	Windows	macOS
Costo SO	Gratis	$10-20K	Incluido hardware
Código Abierto	Sí	No	No
Seguridad	Muy alta	Media	Alta
Rendimiento	Excelente	Bueno	Bueno
Uptime	Años	Meses	Meses
Comunidad	Enorme (global)	Grande (corporativa)	Media
Uso Servidor	90%+	10%	<1%
Uso Desktop	2%	71%	27%
Para Abacom	Ideal	Solo desktops	Caro

18.10 Quiz: Verifica tu Comprensión

> **Antes de continuar**
>
> Responde estas preguntas para verificar que comprendiste los beneficios de Linux.

18.10.1 ¿Cuánto cuesta Linux por servidor por año?

Ver respuesta

$0 (completamente gratis)

O si quieres soporte profesional de Canonical:

- Soporte básico: ~$200-500/año
- Soporte premium: ~$1,000+/año

Comparación:

Listing 18.18 BASH

```
Windows Server: $5,000-20,000 licencia + $1-5K soporte
RHEL: $1,000-3,000 per server per year
Ubuntu LTS: $0 (o $200-1,000 si quieres soporte)
```

En Abacom: Ahorramos miles por cada servidor que usamos Linux.

18.10.2 ¿Por qué Linux es más seguro si su código está abierto?

Ver respuesta

Porque **más ojos = menos bugs**:

1. **Auditoría pública:** Miles de expertos revisan constantemente
2. **Parches rápidos:** Si se encuentra vuln, la comunidad arregla rápido
3. **Sin puertas traseras:** Todo el código es visible, imposible ocultar
4. **Competencia académica:** Expertos quieren ser reconocidos por arreglar bugs

Comparación real:

- Código cerrado: 1 empresa audita (Microsoft)
- Código abierto: Millones de expertos auditan (Google, Intel, Red Hat, individuos)

Conclusión: Código abierto es MEJOR para seguridad.

18.10.3 ¿Por qué Abacom elige Linux en lugar de Windows para servidores?

Ver respuesta

Por estas razones combinadas:

1. **Costo:** $0 vs $15,000+/servidor/año (Windows)
2. **Rendimiento:** Usa menos RAM y CPU
3. **Estabilidad:** Uptime de años sin reinicio
4. **Seguridad:** Código abierto, parches rápidos
5. **Comunidad:** Millones de soporte disponible
6. **Control:** Control total sobre el sistema

Decisión empresarial: Si tienes 100 servidores:

- Windows: 100 × $15,000 = $1.5 MILLONES/año
- Linux: 100 × $200 = $20,000/año
- AHORRO: $1.48 MILLONES/año

Ese dinero se reinvierte en talento, herramientas, etc.

18.11 Práctica Guiada con el Instructor

18.11.1 Calcular ROI de Linux vs Windows

Objetivo: Entender el impacto económico en Abacom.

Instrucciones:

Listing 18.19 BASH

```
# Crear calculadora de costo

# Supuesto: Abacom tiene 50 servidores de producción

# COSTO WINDOWS:
# Licencia por servidor: $15,000
# Soporte anual: $3,000
# Total por servidor: $18,000
# Total 50 servidores: $900,000 AÑO 1
# Años 1-5: $900,000 × 5 = $4,500,000

# COSTO LINUX:
# Licencia: $0
# Soporte (Canonical): $500
# Total por servidor: $500
# Total 50 servidores: $25,000 AÑO 1
# Años 1-5: $25,000 × 5 = $125,000

# AHORRO EN 5 AÑOS:
# $4,500,000 - $125,000 = $4,375,000 AHORRADOS
```

Ejercicio:

1. Calcula ahorro para tu caso específico
2. Incluye beneficios adicionales (mejor rendimiento, menos infraestructura)
3. Comprende el ROI

18.11.2 Comparar Ventajas Técnicas

Objetivo: Experimentar diferencias reales.

Instrucciones:

Listing 18.20 BASH

```
# Instala dos máquinas virtuales:
# - Una con Ubuntu 22.04 LTS
# - Una con Windows Server 2022

# Compara:
# 1. Tamaño de instalación
# 2. Uso de RAM idle
# 3. Velocidad de boot
# 4. Disponibilidad de actualizaciones
# 5. Facilidad de configuración

# Resultado: Linux gana en todos estos
```

18.11.3 Explorar Comunidad Linux

Objetivo: Verificar que comunidad está disponible.

Instrucciones:

Listing 18.21 BASH

```
# Búsqueda en Stack Overflow:
# - "linux server" = 2+ millones preguntas
# - "windows server" = 1 millón preguntas

# Búsqueda en GitHub:
# - Linux projects = 50+ millones
# - Windows projects = 10 millones

# Conclusión: Mucho más contenido para Linux

# Pruébalo:
$ # Abre estos sitios
$ # https://wiki.archlinux.org/title/Main_page
$ # https://github.com/search?q=linux
```

18.12 Laboratorio de Práctica Integrado

Duración: 60-90 minutos
Dificultad: Intermedio
Objetivo General: Entender ventajas reales de Linux vs otros SOs

18.12.1 Fase 1: Investigación de Ventajas (20 min)

18.12.2 Fase 2: Análisis Económico (20 min)

18.12.3 Fase 3: Documento Final (20 min)

18.12.4 Verificación del Laboratorio

- ☐ Ventajas investigadas y documentadas
- ☐ Análisis económico completado
- ☐ Reporte ejecutivo escrito
- ☐ ROI calculado para Abacom
- ☐ Entiendo por qué usamos Linux

18.13 Recursos Adicionales

18.13.1 Documentación Oficial

- **GPL License - GNU Project**
  - Sección: Qué puedes hacer con software GPL
  - Por qué: Entender derechos de Linux
- **Linux Foundation Case Studies**
  - Sección: Empresas usando Linux
  - Por qué: Ver casos reales de éxito

18.13.2 Tutoriales Complementarios

- **Total Cost of Ownership: Linux vs Windows**
  - Duración: Análisis detallado
  - Cubre: Análisis económico real

18.14 Preguntas Frecuentes

P: ¿Si Linux es tan bueno, por qué la gente usa Windows?

R: Porque:

- **Desktops:** Windows/Mac tienen mejor UX gráfica
- **Juegos:** Windows domina gaming
- **Legado:** Muchas empresas viejas tienen inversión en Windows
- **Microsoft Office:** Ms Office funciona mejor en Windows (aunque existe para Linux)

Pero para **servidores**, Linux domina claramente (90%+).

P: ¿Abacom podría cambiar a Windows algún día?

R: Improbable porque:

- Costos serían prohibitivos
- Rendimiento sería menor
- Cambio sería masivo y costoso
- No hay razón técnica o comercial para cambiar

P: ¿Qué pasa si Red Hat o Canonical desaparece?

R: Linux seguiría siendo libre porque:

- La GPL garantiza que el código es libre para siempre
- Hay 20 empresas diferentes soportando Linux
- Comunidad global lo mantendría incluso sin empresa
- No existe dependencia única en vendedor

18.15 Resumen

Lo más importante (3 puntos clave)

1. **Linux es gratis, pero NO es baja calidad** → Es lo opuesto. Usado por Google, Netflix, Amazon porque ES mejor.
2. **Código abierto = mejor seguridad** → Miles de ojos auditan, vulnerabilidades se arreglan rápido, sin puertas traseras.
3. **Linux paga por sí solo** → En Abacom, ahorramos $400K-900K/año usando Linux vs Windows. Eso es ROI en año 1.

Recuerda: Elegir Linux es decisión económica y técnica inteligente.

18.15.1 Checklist de Competencias Alcanzadas

- Entiendo por qué Linux es gratuito
- Puedo explicar ventajas económicas en números
- Comprendo seguridad mediante código abierto
- Sé cómo Linux permite estabilidad superior
- Puedo argumentar por qué Abacom usa Linux
- Estoy listo para instalar y usar Linux

18.15.2 Próximos Pasos

> **Note**
>
> **Siguiente:** Próximas unidades disponibles
> **Objetivo:** Continuar con el curso de administración de servidores
> **Tiempo recomendado:** Depende de la siguiente unidad

18.16 Soporte

> **¿Tienes dudas sobre ventajas de Linux?**
>
> - **Comparaciones técnicas:** Stack Overflow
> - **Análisis económicos:** Linux Foundation
> - **Casos de uso:** Google/Netflix/Amazon blogs

Última actualización: 2026-01-29
Versión: 1.0
Estado: Completo y validado

> **Acerca de las fuentes**
>
> Verificado contra:
>
> - GNU GPL official license
> - Linux Foundation case studies
> - Real TCO studies (Windows vs Linux)
> - Enterprise adoption statistics

18.17 Quiz: Ventajas de Linux

Listing 18.22 BASH

```
$ mkdir -p ~/lab-ventajas-linux
$ cd ~/lab-ventajas-linux

# Crear documento con ventajas verificadas
$ cat > ventajas-abacom.txt << 'EOF'
=== VENTAJAS DE LINUX PARA ABACOM ===

1. ECONÓMICA:

- Costo licencia: $0
- Costo soporte: $0-1,000/año
- Total 50 servidores: $0-50,000/año
- vs Windows: $450,000-900,000/año
- AHORRO: $400,000-900,000/año

2. SEGURIDAD:

- Código abierto (auditable)
- Parches rápidos
- Sin puertas traseras
- Cumplimiento GDPR/ISO

3. RENDIMIENTO:

- Usa menos RAM
- Usa menos CPU
- Mejor para virtualización
- Escalabilidad infinita

4. ESTABILIDAD:

- Uptime de años
- Sin reinicios forzados
- Confiable para 24/7

5. COMUNIDAD:

- Millones de expertos
- Documentación abundante
- Soluciones para cualquier problema
EOF

$ cat ventajas-abacom.txt
```

Listing 18.23 BASH

```
$ cat > analisis-costos.txt << 'EOF'
=== ANÁLISIS DE COSTOS: LINUX vs WINDOWS ===

INFRAESTRUCTURA ACTUAL ABACOM:

- Servidores de producción: 50
- Servidores de desarrollo: 20
- Total: 70 servidores

OPCIÓN A: TODOS WINDOWS SERVER 2022

- Licencia por servidor: $15,000
- Soporte anual: $3,000
- Total por servidor: $18,000
- Total 70 servidores ANUAL: $1,260,000
- 5 años: $6,300,000

OPCIÓN B: TODOS LINUX (Ubuntu 22.04 LTS)

- Licencia por servidor: $0
- Soporte (opcional): $0-500
- Total por servidor: $0-500
- Total 70 servidores ANUAL: $0-35,000
- 5 años: $0-175,000

OPCIÓN C: MIXTA (Abacom actual)

- 70 servidores Linux: $0-35,000/año
- Desktops Windows/Mac: $80,000/año (80 desktops)
- Total ANUAL: $80,000-115,000
- 5 años: $400,000-575,000

AHORRO USANDO LINUX:
Comparado a Windows: $5,725,000-6,300,000 en 5 años
EOF

$ cat analisis-costos.txt
```

Listing 18.24 BASH

```
$ cat > reporte-ventajas.txt << 'EOF'
=== REPORTE: POR QUÉ LINUX ES MEJOR PARA ABACOM ===

CONCLUSIÓN EJECUTIVA:
Linux es la opción tecnológica y económicamente superior para Abacom.

VENTAJAS CLAVE:

1.  Económica: Ahorro de $400K-900K/año

2.  Seguridad: Mejor que Windows/macOS

3.  Rendimiento: Mayor eficiencia de recursos

4.  Estabilidad: Servidores con años de uptime

5.  Comunidad: Soporte global disponible

RECOMENDACIONES:

- Continuar con 100% Linux en producción
- Invertir ahorros en capacitación y herramientas
- Mantener Windows/macOS en desktops para aplicaciones específicas
- Revisar anualmente beneficios realizados

BENEFICIOS REALIZADOS (2024):

- Ahorros: ~$200,000
- Uptime mejorado: +15%
- Velocidad deployments: +40%
- Satisfacción admin: Muy alta
EOF

$ cat reporte-ventajas.txt
```

19 Unidad 1: Recursos

Listing 19.1 QUARTO-TITLE-BLOCK

20 Recursos De La Unidad 1

Recurso	Enlace
Presentacion General (Reveal.js)	Curso
Presentacion (Reveal.js)	Unidad 1
Practica	Practica Unidad 1
Laboratorio	Lab 0: Diagnostico del sistema

20.1 Referencias

- Abacom
- Instructor

Part III

Unidad 2: Instalación y Configuración

21 Requisitos y Preparación para Instalación

Listing 21.1 QUARTO-TITLE-BLOCK

22 Requisitos y Preparación para Instalación

22.1 Introducción

Antes de instalar Ubuntu Server LTS, es fundamental verificar que tu equipo cumple con los requisitos mínimos y preparar el medio de instalación. Esta lección te guiará a través de todo lo necesario para una instalación exitosa.

En este tema aprenderás:

- Requisitos de hardware mínimos y recomendados
- Cómo verificar si tu computadora es compatible
- Descarga segura de Ubuntu Server LTS
- Creación de USB de instalación
- Acceso a BIOS/UEFI
- Preparación del disco duro

22.2 Selecciona tu Entorno de Aprendizaje

Este curso soporta **4 entornos diferentes**. Elige el que mejor se adapte a tu máquina:

22.2.1 Entorno 1: Windows + WSL2 (Windows Subsystem for Linux)

Ideal para: Desarrolladores en Windows que quieren Linux sin virtualización

Requisitos:

- Windows 10 Build 19041+ o Windows 11
- Procesador con virtualización
- RAM: 4 GB mínimo

Ventajas:

- Linux completo en Windows
- Mejor rendimiento que VM
- Acceso simultáneo Windows + Linux

Desventajas:

- No ideal para aprender instalación

- Interfaz gráfica limitada

Para usar WSL2 en este curso:

Listing 22.1 POWERSHELL

```
# En PowerShell como admin:
wsl --install

# Accede a WSL2:
wsl

# Realiza prácticas de comandos aquí
# Para instalar SO: usa máquina virtual
```

22.2.2 Entorno 2: Linux Nativo (Cualquier Distribución)

Ideal para: Aprender administración de Linux en serio

Requisitos:

- Linux ya instalado (Ubuntu, Fedora, Debian, etc.)
- RAM: 2 GB mínimo (4 GB recomendado)
- Disco: 50+ GB disponible

Ventajas:

- Control total del sistema
- Mejor rendimiento
- Ambiente profesional real

Desventajas:

- Requiere Linux pre-instalado
- Menos protección para experimentos

Para instalar otra distribución Linux:

- Opción A: Usa máquina virtual (VirtualBox/Workstation)
- Opción B: Particiona disco e instala dual-boot
- Opción C: Usa Docker para laboratorios rápidos

22.2.3 Entorno 3: macOS + Máquinas Virtuales

Ideal para: Usuarios de Mac que quieren aprender Linux puro

Requisitos:

- macOS 10.14+ (Intel o Apple Silicon M1/M2/M3)

- RAM: 8 GB mínimo (16 GB recomendado)
- Disco: 50+ GB disponible para VM

22.2.4 Opciones de Hipervisor en macOS

Arquitectura	Opciones Disponibles
Intel Mac	VirtualBox (gratis), Parallels (pagado), VMware Fusion (pagado)
Apple Silicon (M1/M2/M3)	UTM (gratis), Parallels (pagado), VMware Fusion (pagado)

Recomendación:

- **Para principiantes:** VirtualBox (Intel) o UTM (Apple Silicon)
- **Para profesionales:** Parallels o VMware Fusion (mejor rendimiento)

⚠ ¿Por qué VirtualBox NO funciona en Apple Silicon?

Razón técnica: VirtualBox está escrito ÚNICAMENTE para arquitectura x86-64 (Intel/AMD), y Apple Silicon usa ARM64 - son incompatibles a nivel de CPU.

22.2.5 La diferencia de arquitecturas:

Aspecto	Intel Mac	Apple Silicon (M1/M2/M3)
Arquitectura CPU	x86-64	ARM64
VirtualBox emula	x86-64 (mismo que host)	x86-64 (diferente al host)
Método	Virtualización HW (rápido)	Emulación total (lentísimo)
Velocidad	~95% nativa	~10-20% nativa
VirtualBox	Funciona perfecto	NO funciona

22.2.6 ¿Por qué no fue actualizado?

Oracle (dueño de VirtualBox) dijo explícitamente: **"No hay planes para soportar Apple Silicon ARM64 natively"**
Razones:

1. **Costo enorme:** Reescribir VirtualBox tomaría años de desarrollo
2. **Mercado pequeño:** Solo ~15% de Macs son Apple Silicon (2025)
3. **Complejidad:** Perderíamos 20+ años de optimizaciones x86-64
4. **Alternativas existentes:** Parallels y UTM ya lo hacen

22.2.7 ¿Qué hacer si tienes Apple Silicon?

- **UTM (Gratis):** Emulador universal, lento pero gratuito
- **Parallels (Pagado):** Optimizado específicamente para M1/M2/M3, muy rápido (~80-90%)
- **VMware Fusion (Pagado):** Soporte ARM64, más lento que Parallels

22.2.8 Analogía simple:

VirtualBox es como un "traductor":

- **Intel Mac:** Traduce Inglés → Inglés (rápido, solo copia)
- **Apple Silicon:** Traduce Japonés → Inglés (lentísimo, tiene que convertir TODO)

Conclusión: No es limitación de tu Mac, es que VirtualBox **nunca fue diseñado** para ARM64.

> **Nota**
>
> En **Apple Silicon (ARM64)**, asegúrate de usar una ISO **ARM64** (por ejemplo, `ubuntu-22.04.x-live-server-arm64.iso`).
> Si intentas instalar un sistema **amd64/x86_64** en una VM ARM sin emulación, no va a bootear.

Ventajas:

- macOS + Linux en la misma máquina
- Protección completa
- Entorno profesional

Desventajas:

- Requiere máquina potente
- Algunas opciones pagadas

22.2.9 Entorno 4: Docker (Contenedores)

Ideal para: Aprender comandos Linux rápidamente

Requisitos:

- Cualquier máquina: Windows, macOS, Linux
- RAM: 4 GB mínimo
- Docker instalado

Ventajas:

- Ultraligero y rápido
- Inicio instantáneo
- Perfecto para experimentar

Desventajas:

- No es SO completo
- No puedes instalar bootloader

Para usar Docker:

Listing 22.2 BASH

```
# Ejecuta Ubuntu directamente
docker run -it ubuntu:22.04 /bin/bash

# Dentro estás en Linux completo
# Practica comandos sin riesgo
```

22.3 ¿Cuál entorno elegir?

Entorno	Para principiantes	Para instalación	Para producción
Windows + WSL2	Muy bueno	No	Excelente
Linux nativo	Excelente	Perfecto	Excelente
macOS + VM	Muy bueno	Bueno	Muy bueno
Docker	Muy fácil	No	Excelente

Recomendación:

- Si eres **principiante**: WSL2 O Docker para aprender comandos
- Si quieres aprender **instalación**: Usa **máquina virtual** (cualquier SO host)
- Si quieres entorno **profesional**: Linux nativo O Docker

Ver guía completa de configuración en: **SETUP.qmd**

22.4 Requisitos de Hardware

22.4.1 Especificaciones Mínimas

Para ejecutar Ubuntu Server LTS, tu computadora necesita al menos (Canonical 2024d):

Componente	Mínimo	Recomendado
Procesador	1 GHz	2 GHz multi-core
RAM	1 GB	4 GB
Espacio en disco	2.5 GB	50 GB
Pantalla	VGA	1024x768
Conexión	CD-ROM o USB	USB 3.0+

22.4.2 ¿Por qué importan estos requisitos?

El **kernel de Linux** es muy eficiente y puede funcionar en hardware antiguo. Sin embargo, para una experiencia fluida (instalador, SSH, paquetes y laboratorios), necesitas recursos suficientes.

La **RAM mínima de 1 GB** es apenas funcional - con aplicaciones modernas, necesitarás **4 GB** para productividad real. El **procesador de 1 GHz** también es mínimo absoluto.

22.5 Verificar Compatibilidad del Hardware

22.5.1 Método 1: Verificar desde Windows

Listing 22.3 `POWERSHELL`

```
systeminfo                                                    ①
```

① **systeminfo** muestra detalles del sistema en Windows (procesador, RAM, BIOS)

Busca en la salida:

- **Processor**: Procesador instalado (ej: Intel Core i5, AMD Ryzen 7)
- **System Type**: x86-64 (64-bit es lo esperado) o ARM64
- **Total Physical Memory**: RAM total en GB

22.5.2 Método 2: Verificar desde macOS

Listing 22.4 BASH

```
system_profiler SPHardwareDataType ①
```

① **system_profiler** muestra información completa de hardware en macOS

Busca:

- **Processor Name**: Tipo de procesador
- **Processor Speed**: Velocidad en GHz
- **Memory**: RAM total

22.5.3 Método 3: Verificar desde Linux existente

Listing 22.5 BASH

```
uname -m ①
```

① **uname -m** muestra la arquitectura del procesador (x86_64 para 64-bit, armv7l para ARM 32-bit)

Listing 22.6 BASH

```
free -h ②
```

② **free -h** muestra RAM disponible en formato legible (human-readable)

③ Este comando cuenta el número de núcleos disponibles (multiplicar por 2 si tiene hyper-threading)

22.5.4 Verificar Particiones de Disco

① **lsblk** (list block devices) muestra todos los discos y particiones de forma visual

② **df -h** (disk free) muestra espacio disponible en particiones existentes

③ **fdisk -l** (list) muestra particiones en formato detallado (requiere sudo)

Listing 22.7 BASH

```
cat /proc/cpuinfo | grep "^processor" | wc -l ③
```

Listing 22.8 BASH

```
lsblk ①
```

22.6 Descargar Ubuntu Server LTS

22.6.1 Paso 1: Obtener la Imagen ISO

Accede a **https://releases.ubuntu.com/22.04/** y descarga el archivo correcto:

- **ubuntu-22.04.4-live-server-amd64.iso** → Para procesadores Intel/AMD 64-bit (RECOMENDADO)
- **ubuntu-22.04.4-live-server-arm64.iso** → Para procesadores ARM64 (Apple Silicon M1/M2/M3, servidores ARM)

El archivo tendrá un tamaño aproximado de **1.5-2.0 GB**.

22.6.2 Paso 2: Verificar Integridad (Importante para Seguridad)

Cada ISO oficial tiene un archivo **SHA256SUMS** que garantiza que la descarga no fue modificada:

① **sha256sum** genera un hash criptográfico del archivo ISO descargado

② **diff** compara el hash descargado con el oficial - si no hay salida, la descarga es válida

En **Windows**, usa:

① **Get-FileHash** es el equivalente de PowerShell para calcular checksums

Luego compara manualmente con el valor en SHA256SUMS.

22.7 Crear USB de Instalación

22.7.1 Software Requerido

Para crear un USB booteable, necesitas herramientas según tu SO:

Listing 22.9 BASH

```
df -h                                                                    ②
```

Listing 22.10 BASH

```
fdisk -l                                                                 ③
```

Sistema	Software	Descarga
Windows	Rufus	https://rufus.ie/
macOS	Etcher	https://www.balena.io/etcher/
Linux	Etcher o dd	**apt install balena-etcher-bin**

22.7.2 Método: Usar Balena Etcher (Multiplataforma)

Paso 1: Descarga e instala Balena Etcher

Paso 2: Conecta tu USB (mínimo 4 GB)

Paso 3: Abre Etcher y selecciona:

```
1. Flash from file → Selecciona ubuntu-22.04.4-live-server-amd64.iso

2. Select target → Elige tu USB (¡CUIDADO! Borra datos)

3. Flash → Espera a que complete (~5 minutos)
```

22.7.3 Método: Usar comando dd en Linux/macOS (Avanzado)

① **lsblk** lista los discos para identificar cuál es tu USB (ej: /dev/sdb)

② **umount** desmonta la partición antes de escribir (reemplaza X con tu letra)

③ **dd** (disk dump) copia el ISO byte a byte. **bs=4M** define el tamaño de bloque. **status=progress** muestra avance

④ **sync** asegura que todos los datos se escriben en el USB antes de continuar

⚠ ADVERTENCIA CRÍTICA

dd sin validación puede **destruir tu disco principal**.
Verifica **3 veces** que **/dev/sdX** es el USB correcto antes de ejecutar el comando.
Una vez ejecutado, NO hay forma de recuperar datos del disco formateado.
Checklist de seguridad:

1. ¿Es **/dev/sda** tu disco principal? (NO usar este)
2. ¿Es **/dev/sdb** o **/dev/sdc** tu USB? (Verificar con `lsblk`)
3. ¿Respaldaste tus datos importantes? (Hacer backup antes)

Listing 22.11 BASH

```
# En Linux o macOS
sha256sum ubuntu-22.04.4-live-server-amd64.iso > descargado.sha256 ①
```

Listing 22.12 BASH

```
diff descargado.sha256 SHA256SUMS ②
```

22.8 Acceso a BIOS/UEFI

Para arrancar desde USB, necesitas cambiar el **orden de boot** en BIOS/UEFI. Esto depende de tu fabricante:

Fabricante	Tecla	Cuando
Dell	F2 o F12	Inmediatamente al encender
HP/Compaq	F10 o F2	Inmediatamente al encender
Lenovo (ThinkPad)	F1 o F2	Inmediatamente al encender
Asus	F2 o Delete	Inmediatamente al encender
Acer	F2 o Delete	Inmediatamente al encender
Apple (Intel)	Command + Option + P + R	Al encender

22.8.1 Cambiar Orden de Boot

```
1. Enciende el equipo

2. Inmediatamente presiona la tecla correspondiente (antes de que cargue SO)

3. Navega a "Boot" o "Boot Order"

4. Mueve "USB Device" al primer lugar

5. Guarda (F10) y sale (Esc)

6. Reinicia con USB conectado
```

22.8.2 UEFI vs BIOS Clásico

BIOS Clásico (legacy) - Antiguo, simple, máximo 4 particiones primarias

UEFI (Unified Extensible Firmware Interface) - Moderno, seguro, soporta GPT, Secure Boot

Ubuntu Server LTS soporta ambos, pero **UEFI es recomendado** para equipos modernos (post-2012) (Forum 2022).

Listing 22.13 POWERSHELL

```powershell
(Get-FileHash ubuntu-22.04.4-live-server-amd64.iso -Algorithm SHA256).Hash ①
```

Listing 22.14 BASH

```bash
lsblk ①
```

Si durante la instalación ves:

```
Installation Type: EFI Installation (recomendado)
```

Significa que estás en UEFI. Si ves:

```
Installation Type: MBR/BIOS Installation
```

Estás en BIOS clásico. Ambos funcionan correctamente.

22.9 Preparación del Disco Duro

22.9.1 Opción 1: Instalación Única (Recomendado para Abacom)

Borrarás completamente el disco e instalarás solo Ubuntu. **TODOS TUS DATOS SE PERDERÁN** - haz backup primero.

22.9.2 Opción 2: Dual Boot (Windows + Ubuntu)

Compartir el disco entre Windows y Ubuntu:

Antes de instalar Ubuntu:

Durante instalación, selecciona “Something else” y particiona manualmente.

Listing 22.15 BASH

```
sudo umount /dev/sdX1                                                    ②
```

Listing 22.16 BASH

```
sudo dd if=ubuntu-22.04.4-live-server-amd64.iso of=/dev/sdX bs=4M status=progress ③
```

22.9.3 Opción 3: Instalación en Partición Existente

Si ya tienes Linux, puedes reparticionar e instalar en partición nueva.

① **fdisk** abre el particionador interactivo (requiere sudo)

```
Comandos útiles en fdisk:
m   → Mostrar menu
p   → Listar particiones actuales
n   → Nueva partición
d   → Borrar partición
w   → Guardar cambios
```

⚠ ADVERTENCIA CRÍTICA

Manejo incorrecto de particiones puede causar pérdida total de datos.
Lo que podría salir mal:

- Borrar la partición equivocada
- Sobrescribir datos existentes
- Dejar el sistema inbooteable

Cómo prevenirlo:

1. Practica primero en máquina virtual (VirtualBox o VMware)
2. Verifica 3 veces la partición antes de ejecutar comandos
3. Haz backup de datos importantes antes de particionar
4. Lee cada paso del particionador antes de confirmar

22.10 Checklist de Preparación

Antes de iniciar la instalación, verifica:

Listing 22.17 BASH

```
sudo sync                                                    ④
```

Listing 22.18 BASH

```
# Esto se hace durante la instalación, no en terminal
# El instalador ofrecerá: "Erase disk and install Ubuntu"
```

- ☐ Hardware cumple mínimos (especialmente RAM y espacio)
- ☐ Descargaste la ISO correcta (amd64 para la mayoría)
- ☐ Validaste SHA256 del ISO
- ☐ USB está formateado en FAT32 o exFAT
- ☐ ISO está grabado en USB con Etcher o dd
- ☐ USB es booteable (probado en otra máquina si es posible)
- ☐ BIOS/UEFI está configurado para booting desde USB
- ☐ Hiciste backup de datos importantes
- ☐ Tienes conexión a internet para descarga de actualizaciones
- ☐ Tiempo disponible: ~30-45 minutos para instalación completa

---

22.11 Ejemplos Prácticos

22.11.1 Ejemplo 1: Verificar Arquitectura en Sistema Existente

22.11.1.1 Linux

Interpretación:

- x86_64 = Procesador Intel/AMD de 64-bit → Descargar **ubuntu-22.04-amd64.iso**
- arm64 = Procesador ARM (Raspberry Pi) → Descargar **ubuntu-22.04-arm64.iso**

22.11.1.2 macOS

Interpretación:

- arm64 = Apple Silicon (M1, M2, M3) - Máquinas modernas
- x86_64 = Intel Mac - Máquinas antiguas (pre-2021)

Listing 22.19 BASH

```
# En Windows, abre Disk Management (diskmgmt.msc)
# Haz clic derecho en C: → Shrink Volume
# Libera espacio (ej: 50 GB para Ubuntu)
# Verás unallocated space en Disk Management
```

Listing 22.20 BASH

```
sudo fdisk /dev/sda ①
```

22.11.1.3 Windows

Interpretación:

- x64-based PC = 64-bit (estándar moderno)
- x86-based PC = 32-bit (muy raro hoy)

22.11.2 Ejemplo 2: Validar Descarga de Ubuntu

22.11.2.1 Linux

Explicación:

- SHA256 es un hash criptográfico
- Cualquier cambio en el archivo = hash diferente
- Garantiza que nadie modificó el archivo en tránsito

22.11.2.2 macOS

Explicación:

- macOS usa **shasum** en lugar de **sha256sum**
- El resultado debe coincidir con el archivo SHA256SUMS

22.11.2.3 Windows

Explicación:

- PowerShell usa **Get-FileHash** para validar
- Compara el hash calculado con el de SHA256SUMS
- Deben ser idénticos

Listing 22.21 BASH

```
# Verificar arquitectura del procesador
$ uname -m
x86_64

# Más información del sistema
$ uname -a
Linux servidor 5.15.0-89-generic #99-Ubuntu SMP Mon Oct 5 09:29:34 UTC 2024 x86_64 GNU/Li

# Confirmar que es 64-bit
$ getconf LONG_BIT
64
```

Listing 22.22 BASH

```
# Verificar arquitectura en macOS
$ uname -m
arm64

# O si es Intel Mac
$ uname -m
x86_64

# Versión completa
$ system_profiler SPHardwareDataType | grep "Chip\|Processor"
Chip: Apple M3 Pro
```

22.11.3 Ejemplo 3: Crear USB Booteable

22.11.3.1 Linux

Explicación:

- **lsblk**: Lista discos (sda = disco principal, sdb = USB)
- **dd**: Escribe bit-a-bit (no es una copia regular)
- **bs=4M**: Tamaño de bloque (más rápido)
- **conv=fsync**: Asegura que se escribió completamente

22.11.3.2 macOS

Explicación:

- **diskutil list**: Lista discos disponibles
- **/dev/rdisk1**: Usar la versión “raw” para más velocidad
- macOS requiere desmontar antes de escribir

Listing 22.23 POWERSHELL

```
# Verificar arquitectura en Windows
PS> [System.Environment]::Is64BitOperatingSystem
True

# Más información
PS> (Get-ComputerInfo -Property SystemSkuFamily).SystemSkuFamily
x64-based PC

# Procesador específico
PS> Get-ComputerInfo | Select-Object -Property CsProcessors
```

Listing 22.24 BASH

```
# 1. Descargar checksums oficiales
$ cd ~/Downloads
$ wget https://releases.ubuntu.com/22.04/SHA256SUMS

# 2. Validar descarga
$ sha256sum -c SHA256SUMS 2>/dev/null | grep ubuntu-22.04.4-live-server-amd64.iso
ubuntu-22.04.4-live-server-amd64.iso: OK

# Si ves OK = descarga correcta
# Si ves FAILED = archivo corrupto, descargar nuevamente
```

22.11.3.3 Windows

Explicación:

- **Balena Etcher**: Herramienta recomendada para Windows
- **Rufus**: Alternativa popular
- Ambas manejan los detalles técnicos automáticamente

22.12 Componentes Clave

22.12.1 ISO (Imagen de Disco)

Un archivo **ISO** es una imagen exacta de un CD/DVD/USB. Contiene todo el sistema operativo Ubuntu comprimido en un archivo.

Listing 22.25 BASH

```bash
# 1. Descargar checksums
$ cd ~/Downloads
$ curl -O https://releases.ubuntu.com/22.04/SHA256SUMS

# 2. Validar descarga
$ shasum -a 256 -c SHA256SUMS | grep ubuntu-22.04.4-live-server-amd64.iso

# Alternativa con openssl
$ openssl dgst -sha256 ubuntu-22.04.4-live-server-amd64.iso
```

Listing 22.26 POWERSHELL

```powershell
# 1. Descargar checksums con PowerShell
PS> Invoke-WebRequest -Uri "https://releases.ubuntu.com/22.04/SHA256SUMS" -OutFile "$env:

# 2. Calcular hash del archivo ISO descargado
PS> (Get-FileHash -Algorithm SHA256 "$env:USERPROFILE\Downloads\ubuntu-22.04.4-live-serve

# 3. Comparar con el archivo SHA256SUMS
PS> (Get-Content "$env:USERPROFILE\Downloads\SHA256SUMS" | Select-String "ubuntu-22.04.4-
```

Para instalar, debes **escribir** (flash) este archivo en un USB, no solo copiarlo.

Analogía: Es como la diferencia entre enviar a alguien el plano de una casa vs enviarle la casa construida.

22.12.2 SHA256 (Validación Criptográfica)

Un **checksum** es una firma digital que verifica que un archivo no fue modificado.

SHA256 genera un código de 64 caracteres único para cada archivo. Si alguien cambia 1 byte del ISO, el SHA256 es completamente diferente.

Analogía: Como la cédula de identidad de un archivo - cada persona tiene uno único.

22.12.3 BIOS vs UEFI

- **BIOS** (Basic Input/Output System) → Estándar antiguo, simple pero limitado
- **UEFI** (Unified Extensible Firmware Interface) → Estándar moderno, más seguro y flexible

Ambos arrancan Ubuntu, pero **UEFI es el estándar actual** desde 2012.

Listing 22.27 BASH

```
# 1. Identificar el USB
$ lsblk | grep -E "^NAME|sd"
NAME           MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sdb              8:16   1  15.6G  0 disk /media/user/USB

# 2. Desmontar el USB (importante!)
$ sudo umount /dev/sdb1 /dev/sdb2 2>/dev/null

# 3. Escribir ISO en USB con dd
$ sudo dd if=~/Downloads/ubuntu-22.04.4-live-server-amd64.iso of=/dev/sdb bs=4M status=pr
3500M bytes transferred in 2.5 seconds

# 4. Expulsar USB
$ eject /dev/sdb
```

22.13 Errores Comunes

Error 1: "Invalid Partition Table"

- **Causa**: El USB no fue escrito correctamente
- **Solución**: Usa Etcher en vez de **dd**, verifica con **fdisk -l /dev/sdX** que sea booteable

Error 2: "No bootable device found"

- **Causa**: BIOS no está configurado para boot desde USB
- **Solución**: Entra a BIOS/UEFI (tecla según fabricante), ve a Boot Order, mueve USB al primer lugar

Error 3: "Cannot load kernel"

- **Causa**: Descargaste el ISO incorrecto (ej: arm64 en máquina x86_64)
- **Solución**: Verifica con **uname -m**, descarga amd64 si ves "x86_64"

Error 4: "Not enough space"

- **Causa**: El disco no tiene 2.5 GB libres mínimo
- **Solución**: Libera espacio, elimina archivos temporales, o particiona el disco primero

Listing 22.28 BASH

```
# 1. Insertar USB y identificarlo
$ diskutil list
/dev/disk0 (internal, physical): ...
/dev/disk1 (external, physical): 15.6 GB USB DISK

# 2. Desmontar el USB
$ diskutil unmountDisk /dev/disk1
Unmount of all volumes on disk1 was successful

# 3. Escribir ISO en USB
$ sudo dd if=~/Downloads/ubuntu-22.04.4-live-server-amd64.iso of=/dev/rdisk1 bs=4m
$ # Esperar a que termine (puede tomar 5-10 minutos)

# 4. Expulsar el USB
$ diskutil eject /dev/disk1
```

22.14 Mejores Prácticas

1. **Siempre valida SHA256** - Toma 30 segundos y previene problemas
2. **Usa Etcher si eres principiante** - Es gráfico y más seguro que **dd**
3. **Deja 50 GB mínimo** - Los 2.5 GB son solo para el SO, necesitas espacio para aplicaciones
4. **Haz backup antes de particionar** - No hay "Deshacer" en operaciones de disco
5. **Actualiza BIOS si es muy viejo** - Equipos pre-2010 pueden tener issues con UEFI

22.15 Tabla de Referencia Rápida

Tarea	Comando/Herramienta	Uso
Ver arquitectura	**uname -m**	Saber si es 32 o 64 bit
Ver RAM	**free -h**	Verificar memoria disponible
Listar discos	**lsblk**	Identificar USB o disco a particionar

Tarea	Comando/Herramienta	Uso
Validar ISO	**sha256sum -c SHA256SUMS**	Verificar que descarga es válida
Crear USB	Balena Etcher	Grabar ISO en USB (GUI)
Crear USB	**dd if=X.iso of=/dev/sdX**	Grabar ISO en USB (CLI)
Ver particiones	**fdisk -l**	Listar particiones existentes
Acceso BIOS	Tecla al encender (Del/F2)	Cambiar boot order

22.16 Quiz: Verificar tu Comprensión

Pregunta 1: Requisitos de Hardware

¿Cuál es la RAM recomendada mínima para que Ubuntu Server LTS funcione de forma productiva?
a) 512 MB
b) 1 GB
c) 4 GB (Correcto)
d) 8 GB
Explicación: Aunque el mínimo técnico es 1 GB, para usar aplicaciones modernas (navegador, editor, terminal) necesitas 4 GB para productividad real.

Pregunta 2: Validación de ISO

¿Por qué es importante validar el SHA256 de la descarga de Ubuntu?
a) Para ahorrar tiempo en descarga
b) Para verificar que el archivo no fue modificado en tránsito (Correcto)
c) Para aumentar la velocidad de instalación
d) No es importante, es opcional
Explicación: SHA256 es una firma criptográfica que prueba que el archivo que descargaste es exactamente el que Ubuntu publicó. Protege contra tampering.

Listing 22.29 POWERSHELL

```
# 1. Usar Balena Etcher (GUI, recomendado)
# Descargar desde: https://www.balena.io/etcher/
# - Seleccionar ISO
# - Seleccionar USB
# - Click "Flash"

# Alternativa: PowerShell (avanzado)
# 2. Identificar el USB en PowerShell
PS> Get-Disk | Where-Object {$_.BusType -eq "USB"}
Number Friendly Name  Serial Number  HealthStatus OperationalStatus
------ -------------- -------------- ------------ -----------------
1      KINGSTON USB   123456789      Healthy      Online

# 3. Formatear USB
PS> Format-Volume -DriveLetter H -FileSystem FAT32 -Confirm:$false

# 4. Usar Etcher o Rufus (GUI)
# PowerShell puro es complejo para escribir ISO
```

Pregunta 3: Creación de USB

¿Cuál es la diferencia entre "Copiar" el ISO al USB vs "Escribir" (flash) el ISO?
a) No hay diferencia, son la mismo
b) Copiar solo funciona en Windows
c) Escribir (flash) graba a nivel de sector, permitiendo boot. Copiar solo crea un archivo dentro del USB (Correcto)
d) Escribir es más lento que copiar
Explicación: Flash escribe el ISO directamente en los sectores del USB, reemplazando todo. Copiar deja el USB con su sistema de archivos. Solo USB "flashed" es booteable.

22.17 Práctica Guiada: Preparar tu Instalación

Objetivos:

1. Verificar que tu hardware es compatible
2. Descargar Ubuntu Server LTS correctamente
3. Validar la descarga
4. Preparar un USB booteable

Tiempo estimado: 30 minutos

Instrucciones:

```
1. VERIFICAR HARDWARE (5 min)
     Abre terminal/PowerShell en tu equipo actual
     Ejecuta comandos según tu SO:

     - Linux: uname -m && free -h
     - macOS: system_profiler SPHardwareDataType | grep -E "Processor|Memory"
     - Windows: systeminfo | findstr /C:"Processor" /C:"Total Physical Memory"
     Nota: ¿Es 64-bit? ¿Tienes 4GB+ RAM? ¿Espacio libre >50GB?
     Si respondiste SÍ a todo, puedes instalar sin problemas

2. DESCARGAR UBUNTU (10 min)
     Ve a https://releases.ubuntu.com/22.04/
      Descarga: ubuntu-22.04.4-live-server-amd64.iso
     Descarga también: SHA256SUMS (archivo de checksums)
     Espera a que ambos completen 100%

3. VALIDAR DESCARGA (5 min)
     Abre terminal en el directorio de descargas
      Ejecuta: sha256sum ubuntu-22.04.4-live-server-amd64.iso
     Compara el resultado con SHA256SUMS
     Deben ser idénticos (copia y pega ambos para verificar)

4. CREAR USB BOOTEABLE (10 min)
     Descarga Balena Etcher: https://www.balena.io/etcher/
     Instala y abre Etcher
     Conecta USB (4GB+) - **TODOS LOS DATOS SE BORRARÁN**
     En Etcher:

     - Click "Flash from file" → Selecciona el ISO
     - Click "Select target" → Elige tu USB
     - Click "Flash" → Espera ~5 minutos
     Etcher confirmará "Flash complete!"

5. VERIFICAR USB (5 min)
     En terminal: lsblk | grep -E "sd|usb"
     Busca tu USB (identificarlo por tamaño)
     Si ves tipo "disk" (no partición), el USB está listo

  ¡Listo! Tienes tu USB de instalación preparada
```

22.18 Laboratorio: Instalación en Máquina Virtual (Recomendado)

Se recomienda practicar primero en una máquina virtual antes de instalar en el sistema real. Las prácticas de instalación de sistemas GNU/Linux se realizarán en **VirtualBox y VMware Workstation**.

22.18.1 Opción 1: Oracle VirtualBox (Gratuito)

Características:

- Completamente gratuito y de código abierto
- Disponible en Windows, macOS, Linux
- Fácil de usar con interfaz intuitiva
- Ideal para principiantes

Requisitos mínimos:

- RAM: 8 GB en tu máquina host (para VM de 4 GB)
- Espacio disco: 60 GB disponibles
- Procesador con soporte de virtualización (Intel VT-x o AMD-V)

Descarga: https://www.virtualbox.org/

Pasos iniciales en VirtualBox:

```
1. Descarga VirtualBox desde https://www.virtualbox.org/

2. Instala VirtualBox en tu máquina host

3. Crea una máquina virtual nueva:
     Nombre: "Ubuntu Server LTS Practice"
     Tipo: Linux
     Versión: Ubuntu (64-bit)
     RAM: 4096 MB (4 GB mínimo)
     Disco duro: Crear disco virtual
     Tamaño: 50 GB
     Tipo de disco: VDI (VirtualBox Disk Image)

4. Configuración adicional (antes de iniciar):
     CPU: 2-4 núcleos
     Memoria video: 128 MB (mínimo)
     Sistema operativo: Linux
     Versión: Ubuntu (64-bit)

5. Monta el ISO de Ubuntu Server:
     Menú: Máquina → Configuración
     Sección: Almacenamiento
```

```
     Unidad óptica → Selecciona el ISO descargado
     Aceptar cambios

6. Inicia la máquina virtual y sigue el instalador de Ubuntu Server

:::{.callout-tip}
##  Recomendación

Si usas VirtualBox en Windows, instala tambien el **VirtualBox Extension Pack** (misma ve

**Que habilita tipicamente:**

- USB 2.0/3.0 passthrough (dispositivos USB dentro de la VM)
- VirtualBox RDP (acceso remoto a la consola de la VM)

**Descarga:** https://www.virtualbox.org/wiki/Downloads
:::
```

22.18.2 Opción 2: VMware Workstation (Profesional)

Características:

- Plataforma profesional con características avanzadas
- Disponible en Windows y Linux
- Mejor rendimiento y más opciones de configuración
- Ideal para ambientes profesionales y servidores

Versiones:

- **Workstation Pro**: Versión pagada con todas las características
- **Workstation Player**: Versión gratuita (limitada)

Requisitos mínimos:

- RAM: 8-16 GB en tu máquina host
- Espacio disco: 60+ GB disponibles
- Procesador con soporte de virtualización (Intel VT-x o AMD-V)

Descarga:

- **Workstation Pro**: https://www.vmware.com/products/workstation-pro.html
- **Workstation Player (Gratis)**: https://www.vmware.com/products/workstation-player.html

Pasos iniciales en VMware Workstation:

```
1. Descarga VMware Workstation desde el sitio oficial

2. Instala Workstation en tu máquina host
```

```
3. Abre VMware Workstation

4. Crea una máquina virtual nueva (File → New Virtual Machine):
     Tipo: Custom Configuration (para control total)
     Compatibilidad: Latest version
     Sistema operativo: Linux → Ubuntu 64-bit
     Nombre: "Ubuntu Server LTS Practice"
     Ubicación: Selecciona directorio
     Siguiente

5. Configuración de hardware:
     CPU: 2-4 procesadores
     RAM: 4096 MB (4 GB mínimo)
     Disco duro: SCSI (recomendado)
     Tamaño: 50 GB
     Almacenamiento: Asignar espacio simple
     Red: NAT (para acceso a internet)
     Finalizar configuración

6. Monta el ISO de Ubuntu Server:
     Click derecho en VM → Settings
     CD/DVD (SATA): Selecciona "Use ISO image file"
     Browse → Selecciona ubuntu-22.04.x-live-server-amd64.iso
     OK

7. Inicia la máquina virtual (Power on) y sigue el instalador
```

22.18.3 Comparativa: VirtualBox vs VMware Workstation

Característica	VirtualBox	VMware Workstation
Precio	Gratuito	Pro: Pagado, Player: Gratuito
Facilidad de uso	Muy fácil	Fácil a moderada
Rendimiento	Bueno	Excelente
Características avanzadas	Básicas	Avanzadas
Snapshots/Clones	Sí	Sí (más rápido)
Documentación	Excelente	Muy buena
Soporte comunitario	Muy activo	Profesional
Ideal para	Aprendizaje, desarrollo	Producción, ambientes reales

Recomendación para este curso:

- **VirtualBox** si es tu primera vez (más simple)
- **VMware Workstation** si necesitas características profesionales

22.18.4 Ventajas de practicar en máquina virtual

Sin riesgo de pérdida de datos:

- Tu máquina real permanece intacta
- Los errores afectan solo la VM

Práctica ilimitada:

- Instala, falla, borra y repite
- Snapshots: guarda estado antes de cambios

Aprendizaje iterativo:

- Experimenta sin presión
- Comprende cada paso del instalador

Requisitos flexibles:

- Usa máquinas viejas como host
- Perfecto para laboratorios

22.18.5 Troubleshooting común

Problema: "VT-x/AMD-V not enabled"

```
Solución: Accede a BIOS de tu máquina host

1. Reinicia la máquina

2. Presiona F2, F10, Del, o Esc (depende del fabricante)

3. Busca "Virtualization" o "SVM"

4. Habilita la opción

5. Guarda y reinicia
```

Problema: Máquina virtual muy lenta

```
Solución: Ajusta configuración de recursos

1. Asigna más CPU (2-4 núcleos)

2. Aumenta RAM (mínimo 4 GB)

3. Disabilita efectos visuales en guest OS

4. Usa disco SSD en lugar de HDD
```

Problema: No detecta el ISO

```
Solución:

- VirtualBox: Menú Máquina → Configuración → Almacenamiento
- Workstation: Click derecho VM → Settings → CD/DVD
- Verifica que el archivo .iso existe y no está dañado
```

22.19 Recursos Adicionales

- **Documentación Oficial**: https://help.ubuntu.com/22.04/installation-guide/
- **Ubuntu Release Notes**: https://wiki.ubuntu.com/JammyJellyfish/ReleaseNotes
- **Hardware Compatibility List**: https://ubuntu.com/certification
- **UEFI Explained** (Wikipedia 2024): https://en.wikipedia.org/wiki/Unified_Extensible_Firmware_Interface
- **Balena Etcher Docs**: https://www.balena.io/etcher/docs/
- **Rufus (Windows)**: https://rufus.ie/

22.20 Conclusión

Ahora tienes: Requisitos claros para instalar Ubuntu USB de instalación booteable Hardware verificado y compatible ISO validado criptográficamente

En la próxima lección, usaremos este USB para **instalar Ubuntu Server LTS paso a paso**.

22.21 Quiz: Requisitos y Preparación

---

23 Instalación de Ubuntu Server LTS Paso a Paso

Listing 23.1 QUARTO-TITLE-BLOCK

24 Instalación de Ubuntu Server LTS Paso a Paso

24.1 Introducción

Esta lección te guiará a través del proceso completo de instalación de Ubuntu Server LTS. El instalador de Ubuntu Server es amigable e intuitivo, pero entender cada paso te ayuda a tomar decisiones informadas sobre particiones, usuarios y configuración del sistema (Canonical 2024c).

En este tema aprenderás:

- Arrancar desde USB de instalación
- Seleccionar idioma y región
- Configurar particiones (Manual vs Automático)
- Crear usuario administrador
- Opciones de red y actualizaciones
- Instalación completa del sistema
- Verificar instalación exitosa

24.2 Requisitos Previos

Asegúrate de tener: USB booteable con Ubuntu Server LTS (de lección anterior) Hardware compatible (4GB RAM, 50GB espacio, 64-bit) Conexión a internet (durante la instalación) Backup de datos si instalas en partición existente 30-45 minutos de tiempo disponible

---

24.3 Paso 1: Arrancar desde USB

24.3.1 Conectar USB y Reiniciar

1. **Conecta el USB** booteable a tu computadora
2. **Reinicia el equipo** (Ctrl+Alt+Delete en Windows, Reiniciar en macOS/Linux)

Listing 24.1 BASH

```
# Estos comandos NO se ejecutan ahora
# Solo muestran lo que pasará después de reiniciar
```

3. **Inmediatamente después de encender**, presiona la tecla de BIOS según tu fabricante:

Fabricante	Tecla
Dell	F12
HP	F9 o Esc
Lenovo	F12 o F2
Asus	Esc
Acer	Esc
Apple (Intel)	Option

24.3.2 Seleccionar USB en Boot Menu

Verás un menú similar a:

```
UEFI Boot Options


1. Ubuntu (USB)            ← Selecciona esta

2. Windows Boot Manager

3. UEFI CD/DVD
```

Selecciona **Ubuntu (USB)** con las flechas y presiona **Enter**.

24.4 Paso 2: Pantalla Inicial del Instalador

24.4.1 Bienvenida Ubuntu

Después de ~30 segundos, verás:

```
Try or Install Ubuntu

[Try Ubuntu without installing]
[Install Ubuntu]  ← Selecciona esto
```

Selecciona **"Install Ubuntu"** con el mouse o Tab + Enter.

24.4.2 Seleccionar Idioma

```
Select a language

Español (España)
Español (México)
English          ← Por defecto
français
Português (Brasil)
```

Para Abacom (empresa española): Selecciona **"Español (España)"**

Haz clic en **"Continue"** o presiona Enter.

24.5 Paso 3: Configurar Teclado y Región

24.5.1 Teclado

```
Keyboard layout

Spanish
Spanish (Macintosh)
English (US)  ← Por defecto
```

Si instalas en España, selecciona **Spanish**.

Si no está disponible, **English (US)** es universal - puedes cambiar después.

24.5.2 Ubicación Geográfica

```
Choose your location

Europe
  Spain  ← Para Abacom
  France
Americas
  United States
```

Selecciona tu ubicación. Esto configura:

- **Zona horaria** (CET/CEST para España)
- **Locales del sistema** (formatos de fecha/número)
- **Servidores NTP** (sincronización de hora)

24.6 Paso 4: Red e Inicio de Sesión

24.6.1 Conectar a Red

```
Network Configuration

Wired (en0) ↑
WiFi (wlan0)
```

Durante instalación, conecta a internet si es posible. El instalador descargará:

- Actualizaciones de kernel
- Drivers de hardware
- Paquetes de idioma

Si no tienes internet, puedes instalar offline (menos recomendado).

24.7 Paso 5: Particionamiento de Disco - LA PARTE CRÍTICA

24.7.1 Opción A: Instalación Única (Recomendado para Abacom)

```
Installation type

[ ] Erase disk and install Ubuntu   ← RECOMENDADO (borra TODO)
[ ] Something else (Manual)         ← AVANZADO (control total)
```

> ⚠ ADVERTENCIA CRÍTICA
>
> **"Erase disk" borra TODOS los datos existentes sin posibilidad de recuperación.**
> **Lo que podría salir mal:**
>
> - Perder todos tus archivos personales
> - Borrar Windows u otros sistemas operativos
> - No poder recuperar datos después
>
> **Cuándo usar "Erase disk and install Ubuntu":**
>
> - Es máquina nueva/virtual sin datos importantes
> - No tienes Windows u otros SO que necesites
> - Solo usarás Linux en esta máquina
> - Ya hiciste backup de datos críticos

Entonces elige:

```
Select disk to install to

[ ] /dev/sda (500 GB) Seagate ST500DM002
[ ] /dev/sdb (2 TB)   Western Digital WD20EZ
```

VERIFICA CUIDADOSAMENTE que seleccionas el disco CORRECTO.

24.7.2 Opción B: Dual Boot (Windows + Ubuntu)

Si ya tenías Windows:

1. **Antes de instalar Ubuntu**, en Windows:
   - Abre **Disk Management** (diskmgmt.msc)
   - Haz clic derecho en C: → **Shrink Volume**
   - Libera 50-100 GB
   - Cierra Windows normalmente
2. **Durante instalación Ubuntu**:
   - Selecciona **"Something else"** (particionamiento manual)
   - Verás espacio "unallocated" en el disco
   - Crea 2 particiones:

```
/dev/sda1  (existente) 500 GB [Windows] (sin cambiar)
/dev/sda2  (nuevo)      50 GB [Ubuntu root]
/dev/sda3  (nuevo)      10 GB [Ubuntu swap]
```

24.7.3 Opción C: Particionamiento Manual (Avanzado)

Selecciona **"Something else"** si necesitas control total.

Verás:

```
Partitions

NAME   TYPE      SIZE      MOUNT POINT
/dev/sda1 Unallocated 500 GB

[+] para agregar
[-] para eliminar
[Edit] para modificar
```

Partición típica para servidor Abacom:

```
/dev/sda1 (EFI)          1 GB    /boot/efi     (FAT32)
/dev/sda2 (root)       100 GB    /             (ext4)
/dev/sda3 (swap)        16 GB    [swap]        (swap)
/dev/sda4 (home)       ~383 GB    /home        (ext4)
```

Crear partición root:

```
[+] Nueva partición
  Size: 100000 MB (100 GB)
  Type: Primary / Logical: Logical
  Location: Beginning / End: End
  Use as: Ext4 journaling file system
  Mount point: /
```

Crear partición home:

```
[+] Nueva partición
  Size: 383000 MB (383 GB)
  Type: Logical
  Use as: Ext4 journaling file system
  Mount point: /home
```

Crear partición swap:

```
[+] Nueva partición
  Size: 16000 MB (16 GB)
  Type: Logical
  Use as: swap area
  Mount point: [none]
```

24.8 Paso 6: Crear Usuario Administrador

24.8.1 Información Personal

```
Who are you?

Your name:              [Diego Saavedra          ]
Your computer's name:   [abacom-ubuntu           ]
Pick a username:        [diego                   ]
Choose a password:      [••••••••••              ]
Confirm password:       [••••••••••              ]

[ ] Log in automatically
[ ] Require my password to log in
```

Campos importantes:

1. **Your name**: Tu nombre completo (se muestra en la GUI)
    - Ejemplo: **Diego Saavedra**
2. **Computer's name**: Nombre del equipo en la red (hostname)
    - Debe ser alfanumérico, guiones permitidos
    - Ejemplo: **abacom-ubuntu** (sin espacios)
    - Acceso después: **ssh diego@abacom-ubuntu.local**
3. **Username**: Usuario login (minúsculas, sin espacios)
    - Ejemplo: **diego** (no **Diego Saavedra**)
    - Acceso: **ssh diego@abacom-ubuntu.local**
4. **Password**: Contraseña del usuario
    - Mínimo 8 caracteres
    - Recomendado: letras, números, símbolos
    - Ejemplo: **AbacomLinux2024!**
    - **NO OLVIDES** - sin password = sin acceso
5. **Require my password**: Marcar esta opción
    - Evita acceso no autorizado si dejas desatendida la máquina

---

24.9 Paso 7: Seleccionar Software Inicial

24.9.1 Actualizaciones y Otros Software

```
Updates and other software

[ ] Install security updates automatically
[ ] Install third-party software for graphics, WiFi, etc
```

Recomendaciones para Abacom:

```
[ ] Install security updates automatically
    → Parchea automáticamente vulnerabilidades

[ ] Install third-party software for graphics, WiFi, etc
    → Instala drivers propietarios si los necesita
```

Selecciona ambas opciones para máxima compatibilidad.

24.10 Paso 8: Revisar Resumen y Confirmar

24.10.1 Resumen de Cambios

```
Installation Summary

Location:           Spain
Keyboard:           Spanish
Locale:             es_ES.UTF-8
Hostname:           abacom-ubuntu
Username:           diego
Installation type:  Erase disk and install Ubuntu
Disk to install to: /dev/sda (500 GB)

These changes will be made to these disks:
[ ] [ ] /dev/sda  BOOTLOADER, PARTITION 1, PARTITION 2
```

ÚLTIMA OPORTUNIDAD PARA REVISAR

Si ves algo incorrecto (disco equivocado, particiones raras), haz clic **"Back"** para corregir.

Si todo está bien: Haz clic **"Install"** o **"Begin"**.

24.11 Paso 9: Instalación en Progreso

24.11.1 Barra de Progreso

```
Installing system

35%             Copying files...
```

¿Qué está haciendo?

1. **Copiar sistema de archivos** (3-5 min) - Escribe archivos del SO en disco
2. **Instalar bootloader** (1 min) - Configura GRUB para arrancar
3. **Configurar red** (1 min) - Instala drivers de red
4. **Generar locales** (3-5 min) - Crea archivos de idioma
5. **Instalar actualizaciones** (5-10 min) - Si hay conexión a internet

Tiempo total: 15-30 minutos dependiendo de velocidad de disco y internet

No interrumpas la instalación - Si se corta, deberás reintentar.

24.12 Paso 10: Reiniciar el Sistema

24.12.1 Final de Instalación

```
Installation complete

The installation has finished successfully.

[Restart now]  ← Haz clic aquí
```

Haz clic **"Restart now"**.

El sistema mostrará:

```
Please remove the installation media, then press ENTER to continue.
```

IMPORTANTE:

1. **Desconecta el USB de instalación**
2. Presiona **Enter**
3. La máquina reiniciará

24.13 Paso 11: Primera Sesión

24.13.1 Pantalla de Login

Después del reinicio (~1 minuto), verás:

```
Ubuntu Server LTS

abacom-ubuntu login: diego
Password: ••••••••••
```

O si está en interfaz gráfica (GNOME):

```
ubuntu GNOME Desktop

[icono de usuario] diego
Password: [campo para escribir]
```

Ingresa:

- **Usuario**: **diego** (el que creaste)
- **Contraseña**: La que estableciste

24.14 Paso 12: Verificar Instalación Exitosa

24.14.1 Terminal: Verificar Versión

Listing 24.2 BASH

```
lsb_release -a                                                    ①
```

① **lsb_release -a** muestra información de la distribución instalada

Salida esperada:

```
No LSB modules are available.
Distributor ID: Ubuntu
Release:        22.04
Codename:       jammy
Description:    Ubuntu 22.04.4 LTS
```

24.14.2 Verificar Kernel

Listing 24.3 BASH

```
uname -r                                                          ①
```

① **uname -r** muestra versión del kernel Linux

Salida esperada:

```
5.15.0-91-generic
```

Versión 5.15+ es correcta para Ubuntu Server LTS.

Listing 24.4 BASH

```
lsblk                                                                    ①
```

24.14.3 Verificar Particiones

① **lsblk** lista el particionamiento realizado

Salida esperada (instalación única):

```
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda      8:0    0  500G  0 disk
 sda1   8:1    0  1.5G  0 part /boot/efi
 sda2   8:2    0   50G  0 part /
 sda3   8:3    0  448G  0 part /home
```

24.14.4 Verificar Espacio Disponible

Listing 24.5 BASH

```
df -h                                                                    ①
```

① **df -h** muestra espacio usado/disponible en particiones

Salida esperada:

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda2        50G  8.5G   39G  18% /
/dev/sda3       448G  100M  425G   1% /home
/dev/sda1       1.5G  400M  1.1G  27% /boot/efi
```

24.14.5 Verificar Conexión a Internet

Listing 24.6 BASH

```
ping -c 3 8.8.8.8                                                        ①
```

① **ping -c 3** envía 3 paquetes de prueba a Google DNS (8.8.8.8)

Salida esperada:

```
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=119 time=25.3 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=119 time=23.8 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=119 time=24.2 ms
```

Si ves 3 respuestas exitosas = Internet funciona

24.15 Ejemplos Prácticos

24.15.1 Ejemplo 1: Instalación en Abacom (Empresa Real)

24.15.1.1 Servidor Linux (Ubuntu Server LTS)

Escenario: Abacom necesita 5 servidores Ubuntu Server LTS

Configuración estándar por servidor:

- **Hostname**: **abacom-server-01** a **abacom-server-05**
- **Usuario**: **admin** (o **sysadmin**)
- **Particiones**:
  - **/** → 50 GB (sistema operativo)
  - **/var** → 50 GB (logs, bases de datos)
  - **/home** → 50 GB (datos de usuarios)
  - **swap** → 16 GB

Pasos:

1. Crear 5 USBs booteable (o usar PXE boot)
2. Instalar cada servidor con mismo usuario pero Hostname único
3. Después: Script para configuración común (SSH keys, firewall, etc)

24.15.1.2 Escritorio Personal (Windows + Ubuntu Dual Boot)

Escenario: Quieres Windows 11 para trabajo y Ubuntu para desarrollo

En Windows 11:

En Windows (Descargar Ubuntu):

Instalar Ubuntu:

Listing 24.7 BASH

```
# Verificar instalación en primer servidor:
$ ssh admin@abacom-server-01
$ cat /etc/hostname
abacom-server-01
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1        50G  2.5G   47G   5% /
/dev/sda2        50G   10G   40G  20% /var
/dev/sda3        50G   100M   49G   1% /home
/dev/sda4        16G     0   16G   0% swap
```

Listing 24.8 POWERSHELL

```
# Paso 1: Preparar espacio libre
PS> # Abre Disk Management (diskmgmt.msc)
PS> # Right-click Disco C: → Shrink Volume
PS> # Liberar 100 GB
Get-Volume | Select-Object DriveLetter, Size, SizeRemaining
```

24.15.1.3 macOS (Alternativa: Usar Multipass o Docker)

Escenario: Tienes Mac y quieres desarrollar en Ubuntu

OPCIÓN A: Multipass (Recomendado)

① **multipass transfer** copia archivos del host a la VM (y viceversa) sin configurar SSH manualmente

OPCIÓN B: Docker Desktop (Ligero)

Por qué en macOS?

- macOS: 100% BSD Unix-like (parcialmente compatible)
- Pero: Herramientas pueden variar
- Solución: Usar Multipass o Docker para tener Ubuntu "real"

24.15.2 Ejemplo 2: Validar Particiones Antes de Instalar

24.15.2.1 Linux (Verificación desde USB)

24.15.2.2 Windows (Verificación desde PowerShell)

Listing 24.9 POWERSHELL

```
# Paso 2: Descargar ISO (en Windows)
PS> Invoke-WebRequest -Uri "https://releases.ubuntu.com/22.04/ubuntu-22.04.4-live-server-
    -OutFile "C:\Users\usuario\Downloads\ubuntu-22.04.4-live-server-amd64.iso"

# Paso 3: Flash ISO a USB (usar Balena Etcher)
# - Descargar Balena Etcher desde https://www.balena.io/etcher/
# - Seleccionar ISO
# - Seleccionar USB
# - Click "Flash"
```

Listing 24.10 BASH

```
# Boot desde USB → "Something else" (manual)
# Crear particiones en 100 GB libres
# Instalar con Windows existente

# Después de reiniciar:
$ # Verás menu GRUB con opciones:
# 1. Ubuntu
# 2. Windows 11

# Resultado: Dual boot funcional
```

24.15.2.3 macOS (Verificación desde Terminal)

24.15.3 Ejemplo 3: Caso Real - Servidor Multi-Disco con RAID

24.15.3.1 Linux (Configuración Abacom)

Escenario: Servidor de bases de datos con 2 SSDs y 4 HDDs

Durante Instalación (Elegir "Something else"):

Después de instalación (configurar RAID):

24.15.3.2 Windows Server (Comparación)

Configuración equivalente en Windows Server:

Listing 24.11 BASH

```
# En macOS, instalar Multipass (VM manager de Canonical)
$ brew install multipass

# Crear VM Ubuntu
$ multipass launch --name abacom-dev ubuntu:22.04
Creating instance with 1 CPU, 1GB of memory and 5GB of disk space

# Acceder a la VM
$ multipass shell abacom-dev
ubuntu@abacom-dev:~$ uname -a
Linux abacom-dev 5.15.0-1234-generic #1234-Ubuntu SMP x86_64

# Desde macOS, copiar archivos a VM:
$ multipass transfer archivo.txt abacom-dev:/home/ubuntu/                    ①
```

Listing 24.12 BASH

```
# En macOS, usar Docker para contenedor Linux
$ docker run -it ubuntu:22.04 /bin/bash

root@container# apt update
root@container# apt install -y build-essential
# Tienes Ubuntu dentro de Docker en tu Mac
```

24.15.3.3 macOS (No recomendado para servidores, pero posible)

En macOS (usando RAID via Disk Utility o terminal):

Conclusión: | Aspecto | Linux | Windows | macOS | |———|——-|———|——-| | RAID nativo | mdadm | Storage Spaces | No recomendado | | Producción | Ideal | Viable | No usar | | Facilidad | Fácil | Complejo | Limitado |

24.16 Componentes Clave

24.16.1 GRUB (GRand Unified Bootloader)

GRUB es el bootloader que carga Linux después de que BIOS/UEFI busca discos.

① **cat** muestra el contenido del archivo de configuración de GRUB

Parámetros comunes:

Listing 24.13 BASH

```
# Boot desde USB Ubuntu live
$ # Presiona Ctrl+Alt+T para terminal
$ sudo fdisk -l

Disk /dev/sda: 238.47 GiB, 256060514304 bytes
...
```

Listing 24.14 POWERSHELL

```
# En Windows, antes de encoger C:
PS> Get-Disk | Select-Object DiskNumber, TotalSize, FreeSpace

DiskNumber TotalSize     FreeSpace
---------- ---------     ---------
         0 238GB         50GB

# Encoger disco C: (liberar 100 GB):
PS> $disk = Get-Disk -Number 0
PS> $partition = $disk | Get-Partition | Where-Object {$_.DriveLetter -eq 'C'}
PS> $partition | Resize-Partition -Size ($partition.Size - 100GB)
```

```
GRUB_DEFAULT=0           # Entrada de menu por defecto
GRUB_TIMEOUT=5           # Segundos antes de boot automático
GRUB_CMDLINE_LINUX=""    # Parámetros para el kernel
```

Si editas GRUB, debes actualizar:

① **update-grub** reconstruye la configuración de GRUB desde /etc/default/grub

24.16.2 Ext4 Filesystem (kernel.org 2024)

ext4 (Fourth Extended Filesystem) es el sistema de archivos por defecto de Ubuntu.

Características de ext4:

- **Journaling**: Protege contra corrupción si se apaga abruptamente
- **Sparse files**: Archivos "huecos" usan menos espacio
- **Extents**: Mejor performance con archivos grandes
- **Límite de tamaño**: Soporta archivos hasta 16 TB

Alternativas de sistemas de archivos (menos comunes):

- **XFS**: Para datos muy grandes (aplicaciones científicas)
- **Btrfs**: Experimental, con soporte para snapshots
- **ZFS**: Especializado para RAID y almacenamiento

Listing 24.15 BASH

```
# En macOS, ver discos disponibles
$ diskutil list

/dev/disk0 (internal, physical):
   #:                       TYPE NAME                    SIZE       IDENTIFIER
   0:      GUID_partition_scheme                        *238GB      disk0
   1:                        EFI EFI                     209MB      disk0s1
   2:       Apple_APFS Macintosh HD                      ~150GB     disk0s2

# Para crear espacio para Linux, usar Disk Utility.app
# O via terminal:
$ diskutil secureErase freespace 0 /Volumes/FreeSapce
```

24.16.3 Swap (Memoria Virtual)

Swap es espacio en disco usado como extensión de RAM cuando la memoria física se llena.

① **swapon –show** muestra estado de swap activo

Tamaño recomendado de swap:

- **Máquina con 8 GB RAM** → 4-8 GB swap
- **Máquina con 32 GB RAM** → 2-4 GB swap
- **Servidor sin hibernación** → 1-2 GB swap

24.17 Errores Comunes

24.17.1 Error 1: "No bootable device found"

- **Causa**: USB no fue flashed correctamente o está desconectado
- **Solución**: Reconecta USB, presiona F12, selecciona USB en boot menu

24.17.2 Error 2: "Disk I/O error" durante instalación

- **Causa**: Disco dañado o USB corrupto
- **Solución**: Intenta con USB recién grabado. Si persiste, disco está fallando

Listing 24.16 BASH

```
# Configuración planificada:
SSD #1 (/dev/sda) - Partición primaria:
  50 GB / (sistema)
  16 GB swap

SSD #2 (/dev/sdb) - Dedicado a BD:
  400 GB /var (logs, temp)

HDD #1-4 (/dev/sd{c-f}) - RAID-10:
  8 TB RAID (datos de BD)

# Durante setup visual, crear particiones en orden:
# 1. Seleccionar /dev/sda como disco principal
# 2. Click "New Partition Table" en /dev/sda
# 3. Crear:
#    - 50 GB mounted at /
#    - 16 GB swap
# 4. En /dev/sdb, crear:
#    - 400 GB mounted at /var
```

24.17.3 Error 3: "Installer crashed"

- **Causa**: Bug raro (muy poco común)
- **Solución**: Reinicia desde USB y reintenta

24.17.4 Error 4: "No connection to Internet" durante setup

- **Causa**: Red no está configurada o cable desconectado
- **Solución**: Pulsa Atrás, espera a que detecte red, intenta de nuevo

24.17.5 Error 5: "Partitions too small"

- **Causa**: Seleccionaste disco con menos de 2.5 GB libres
- **Solución**: Selecciona otro disco o borra datos

24.18 Mejores Prácticas

1. **Siempre valida discos** - Usa **fdisk -l** para verificar disco correcto antes de "Erase"
2. **Deja espacio en / para crecer** - Mínimo 20 GB, preferible 50+ GB

Listing 24.17 BASH

```
$ ssh admin@server
$ sudo apt install mdadm  # herramienta RAID

# Ver discos disponibles:
$ sudo fdisk -l | grep "Disk /dev/sd"
Disk /dev/sdc: 2 TiB
Disk /dev/sdd: 2 TiB
Disk /dev/sde: 2 TiB
Disk /dev/sdf: 2 TiB

# Crear RAID-10:
$ sudo mdadm --create /dev/md0 --level=10 --raid-devices=4 /dev/sdc /dev/sdd /dev/sde /de

# Montar RAID:
$ sudo mkdir -p /mnt/database
$ sudo mount /dev/md0 /mnt/database

# Verificar:
$ df -h | grep /mnt/database
/dev/md0       7.8T    100G  7.7T   2% /mnt/database
```

3. **Separa /home de /** - Facilita actualizaciones sin perder datos
4. **Usa swap físico, no archivo** - Mejor performance que swapfile
5. **Documenta particiones** - Escribe en papel qué va en qué disco
6. **Practica en VirtualBox** - Antes de instalar en máquina real

---

24.19 Tabla de Referencia Rápida

Etapa	Opción	Recomendación
Idioma	Español / English	Español (España) para Abacom
Teclado	Spanish / English	Spanish si está disponible
Ubicación	Tu país	Spain para Abacom
Red	Wired / WiFi	Wired si es posible (más estable)
Instalación	Erase / Manual	Erase (simple) o Manual (control)
Usuario	nombre_usuario	Minúsculas, sin espacios, 8+ caracteres
Password	• • • • • • • • • •	Mínimo 8, con números y símbolos
Swap	Size	RAM ÷ 2 (mínimo 1 GB)

Listing 24.18 POWERSHELL

```
# En Windows Server, usar Storage Spaces (equivalente a RAID)
PS> Get-PhysicalDisk

FriendlyName    HealthStatus OperationalStatus Size       Usage
------------    ------------ ----------------- ----       ---
Samsung 960 Pro Healthy      OK                2 TB       Auto-provisioned
WDC Red Plus    Healthy      OK                2 TB       Auto-provisioned
Seagate IronWolf Healthy     OK                2 TB       Auto-provisioned
Seagate IronWolf Healthy     OK                2 TB       Auto-provisioned

# Crear storage pool:
PS> New-StoragePool -FriendlyName "Database-RAID" `
     -StorageSubsystemFriendlyName "Storage Spaces Direct" `
     -PhysicalDisks (Get-PhysicalDisk)

# Crear virtual disk (equivalente a RAID-10):
PS> New-VirtualDisk -StoragePoolFriendlyName "Database-RAID" `
     -FriendlyName "database-volume" `
     -ResiliencySettingName "Mirror" `
     -Size 8TB
```

Listing 24.19 BASH

```
# macOS soporta RAID pero NO es recomendado para producción
$ diskutil secureErase freespace 0 /Volumes/Free

# Mejor: Usar Linux o Windows Server
# macOS está hecho para desarrollo, no para servidores críticos
```

24.20 Quiz: Verificar tu Comprensión

Pregunta 1: Particionamiento

¿Por qué es importante separar / de **/home** en particiones diferentes?
a) Es más rápido
b) Usa menos espacio
c) Si / falla y necesitas reinstalar SO, datos en **/home** se preservan (Correcto)
d) No hay razón, es opcional
Explicación: Separar permite reinstalar el SO sin tocar datos de usuario. Si ambos están en /, reinstalar borra todo.

Listing 24.20 BASH

```
# Configuración de GRUB está en:
cat /etc/default/grub ①
```

Listing 24.21 BASH

```
sudo update-grub ①
```

Pregunta 2: Arranque desde USB

¿Cuál es la tecla para acceder al BIOS en un Dell?
a) Delete
b) F2
c) F12 (Correcto)
d) Escape
Explicación: Aunque F2 abre configuración de BIOS, F12 muestra el boot menu que permite seleccionar USB directamente.

Pregunta 3: Verificar Instalación

¿Cómo verificas qué versión de Ubuntu se instaló correctamente?
a) uname -a
b) lsb_release -a (Correcto)
c) cat /etc/os-release
d) Todas las anteriores funcionan
Explicación: **lsb_release** es estándar para distribuir Linux. Ambas funcionan, pero **lsb_release** es más portable.

24.21 Práctica Guiada: Instalación Paso a Paso

Objetivos:

1. Arrancar desde USB
2. Navegar el instalador de Ubuntu
3. Crear particiones
4. Instalar el sistema
5. Verificar instalación

Listing 24.22 BASH

```
# Ver swap disponible
swapon --show                                                    ①
```

Tiempo estimado: 45 minutos

Instrucciones:

```
1. PREPARACIÓN (2 min)
     Conecta USB booteable
     Cierra todas aplicaciones
     Guarda documentos abiertos
     Encaja en proceso: NO INTERRUMPAS instalación

2. ARRANCAR DESDE USB (3 min)
     Reinicia la máquina
     Presiona tecla de BIOS inmediatamente (F12, Del, etc)
     Busca "USB Device" o "Removable Media"
     Presiona Enter
     Espera a que cargue el Live USB (~30 seg)

3. INSTALADOR - IDIOMA (2 min)
     Ves menú "Try or Install"
     Selecciona "Install Ubuntu"
     Selecciona idioma: Español (España)
     Haz clic "Continue"

4. INSTALADOR - TECLADO Y REGIÓN (2 min)
     Teclado: Spanish (si está disponible)
     Ubicación: Spain
     Haz clic "Continue"

5. INSTALADOR - RED (1 min)
     Espera a que detecte conexión wired/wifi
     Verás "Wired (ip address) connected" o similar
     Haz clic "Continue"

6. INSTALADOR - PARTICIONES (5 min)
     Selecciona "Erase disk and install Ubuntu" (simple)
     O "Something else" si quieres control manual
     Verifica que seleccionas disco CORRECTO
     Haz clic "Continue"

7. INSTALADOR - USUARIO (3 min)
     Your name: Tu nombre completo
     Computer name: hostname (ej: abacom-ubuntu)
```

```
     Pick a username: usuario lowercase (ej: diego)
     Password: contraseña segura (8+ caracteres)
     Marca "Require my password to log in"
     Haz clic "Continue"

8. INSTALADOR - CONFIRMACIÓN (1 min)
     Revisa resumen de cambios
     Si todo es correcto: Haz clic "Install"
     Si hay error: Haz clic "Back" para corregir

9. INSTALACIÓN AUTOMÁTICA (20 min)
     Barra de progreso muestra avance
     NO INTERRUMPAS - deja que termine
     Puedes leer documentación durante este tiempo

10. REINICIO (3 min)
      Ves "Installation complete"
      Desconecta USB
      Haz clic "Restart now"
      Presiona Enter
      Máquina reinicia

11. LOGIN (1 min)
      Ves login screen o GNOME desktop
      Ingresa usuario y contraseña
      Presiona Enter

12. VERIFICACIÓN (5 min)
      Abre Terminal: Ctrl+Alt+T
      Ejecuta: lsb_release -a
     Verifica que ves "Ubuntu 22.04" y "jammy" (Ubuntu Server LTS)
      Ejecuta: df -h
      Verifica particiones están montadas
      Ejecuta: ping -c 3 8.8.8.8
      Verifica que hay conexión a Internet

  ¡Listo! Ubuntu Server LTS está instalado y funcionando
```

24.22 Laboratorio: Instalación en VirtualBox (Recomendado Primero)

Si es tu primera vez, practica en máquina virtual primero. Las prácticas de instalación se realizarán en **VirtualBox y VMware Workstation**.

24.22.1 Instalación en VirtualBox

```
Pasos detallados:

1. Descarga e instala VirtualBox:
     https://www.virtualbox.org/

   (Windows) Instala tambien el **VirtualBox Extension Pack** (misma version exacta que V
     https://www.virtualbox.org/wiki/Downloads

2. Crea máquina virtual nueva:
     Nombre: "Ubuntu Server LTS Lab"
     Tipo: Linux
     Versión: Ubuntu (64-bit)
     RAM: 4096 MB (4 GB)
     Tipo de disco: VDI (VirtualBox Disk Image)
     Tamaño: 50 GB (dinámico)

3. Configura media de arranque:
     Menú: Máquina → Configuración
     Sección: Almacenamiento
     Unidad óptica CD/DVD
     Selecciona `ubuntu-22.04.x-live-server-amd64.iso`

4. Arranca la VM (Power On)

5. Sigue el instalador (idéntico a máquina real)

6. Experimenta, comete errores, aprende

7. Antes de cambios importantes, crea snapshots:
   Máquina → Snapshots → Tomar Snapshot
```

24.22.2 Instalación en VMware Workstation

```
Pasos detallados:

1. Descarga e instala VMware Workstation:

   - Pro: https://www.vmware.com/products/workstation-pro.html
   - Player: https://www.vmware.com/products/workstation-player.html

2. Abre VMware Workstation → File → New Virtual Machine

3. Configuración personalizada:
     Compatibilidad: Última versión disponible
     Sistema operativo: Linux → Ubuntu 64-bit
     Nombre: "Ubuntu Server LTS Lab"
```

```
     CPU: 2-4 procesadores
     RAM: 4096 MB (4 GB)
     Red: NAT (acceso a internet)
     Disco: SCSI
     Tamaño: 50 GB
     Asignar espacio simple (mejor rendimiento)

4. Monta el ISO:
     Click derecho en VM → Settings
     CD/DVD → Use ISO image file
     Selecciona ubuntu-22.04.x-live-server-amd64.iso

5. Arranca la VM (Power On)

6. Sigue el instalador paso a paso

7. Crea snapshots antes de cambios:
   VM → Snapshot → Take Snapshot
```

Ventajas de practicar en máquina virtual:

Sin riesgo de pérdida de datos:

- Tu máquina host permanece completamente protegida
- Los errores afectan solo la máquina virtual

Snapshots - Regresa en el tiempo:

- Guarda estado antes de cambios importantes
- Restaura si algo sale mal
- Perfecto para experimentar

Instalación ilimitada:

- Practica el instalador tantas veces como quieras
- Borra y reinicia sin límite
- Cero consecuencias de errores

Requisitos flexibles:

- Usa máquinas viejas o poco poderosas como host
- Perfecto para laboratorios educativos

Toma control total:

- Pausa la máquina en cualquier momento
- Clona discos duros virtuales
- Experimenta con particionamiento sin presión

24.22.3 Comparativa: VirtualBox vs VMware Workstation

Aspecto	VirtualBox	Workstation
Precio	Gratuito	Pro: Pagado, Player: Gratis
Curva de aprendizaje	Muy baja	Media
Rendimiento	Bueno	Excelente
Snapshots	Sí, pero lentos	Sí, muy rápidos
Clonación	Sí	Sí, muy rápido
Documentación	Excelente	Muy buena
Ideal para	Principiantes, lab básicos	Profesionales, prod

24.22.4 Troubleshooting durante la instalación

Problema: La VM no arranca después de instalar

```
Causas comunes:

1. ISO todavía montado en CD/DVD
     Solución: Desmonta el ISO en Configuración → Almacenamiento

2. Bootloader no instalado correctamente
     Solución: Reinstala GRUB desde liveUSB

3. Partición EFI corrupta
     Solución: Repara particiones desde live session
```

Problema: Ubuntu instala pero arranca lentamente

```
Causas:

1. Recursos insuficientes
     Asigna más CPU y RAM

2. Disco virtual fragmentado
     Defragmenta el disco virtual (.vdi o .vmdk)

3. Efectos visualizados habilitados
     Deshabilita efectos en VM
```

Problema: No reconoce el ISO

```
Pasos:

1. Verifica que el archivo .iso existe
```

```
2. Verifica integridad: sha256sum ubuntu-*.iso

3. En VirtualBox: Menú Máquina → Configuración → Almacenamiento

4. En Workstation: Settings → CD/DVD → Use ISO image file

5. Intenta con ruta absoluta (no relativa)
```

24.23 Recursos Adicionales

- **Ubuntu Installation Guide**: https://help.ubuntu.com/22.04/installation-guide/
- **VirtualBox Manual**: https://www.virtualbox.org/manual/
- **GRUB Bootloader**: https://www.gnu.org/software/grub/
- **Ext4 Filesystem**: https://ext4.wiki.kernel.org/
- **Linux Partitioning**: https://wiki.archlinux.org/title/Partitioning

---

24.24 Conclusión

Ahora tienes:

- Ubuntu Server LTS completamente instalado
- Sistema verificado y funcionando
- Conexión a internet establecida
- Usuario administrador creado

En la próxima lección, realizaremos la **configuración inicial del sistema** (usuarios adicionales, SSH, firewall).

24.25 Quiz: Instalación de Ubuntu

---

25 Configuración Inicial del Sistema

Listing 25.1 `QUARTO-TITLE-BLOCK`

Usuarios, SSH y Permisos - Primeras 24 Horas

25.1 Objetivos de Aprendizaje

Después de completar este tema, serás capaz de:

- Crear usuarios adicionales y configurar grupos
- Generar y configurar claves SSH para acceso seguro
- Entender y aplicar permisos Unix (chmod, chown)
- Configurar sudo para administración sin riesgos
- Optimizar la seguridad en los primeros pasos

25.2 Por Qué Esta Lección Es Crítica

Tu servidor Ubuntu sale de la caja con **UN SOLO USUARIO ADMINISTRADOR**. Esto es un riesgo de seguridad. En Abacom, cada administrador necesita su propia cuenta, acceso SSH seguro, y permisos apropiados sin usar **root** directamente.

Tiempo estimado: 90-120 minutos de lectura + laboratorio

25.3 1. Usuarios y Grupos en Linux

25.3.1 1.1 Concepto Fundamental

Cada archivo y proceso en Linux tiene un **propietario (usuario)** y un **grupo**. Los permisos se asignan a estos tres niveles:

```
    USUARIO            GRUPO           OTROS
  (propietario)                     (el resto)

   r w x              r w x           r w x
 (4 2 1)            (4 2 1)         (4 2 1)
```

```
r = lectura (read)    = 4
w = escritura (write) = 2
x = ejecución (exec)  = 1
```

Ejemplo: **755** significa:

- Usuario: 7 (4+2+1 = **rwx** = lectura + escritura + ejecución)
- Grupo: 5 (4+1 = **r-x** = lectura + ejecución)
- Otros: 5 (4+1 = **r-x** = lectura + ejecución)

25.3.2 1.2 Ver Usuarios y Grupos

Listing 25.2 BASH

```
# **Ver todos los usuarios del sistema**
cat /etc/passwd                                                    ①

# **Ver solo usuarios humanos (UID >= 1000)**
getent passwd | awk -F: '$3 >= 1000 {print $1}'                    ②

# **Ver grupos de un usuario**
groups diego                                                       ③

# **Ver todos los grupos**
cat /etc/group                                                     ④

# **Ver información detallada de usuario actual**
id                                                                 ⑤

# **Ejemplo de salida**
uid=1000(diego) gid=1000(diego) groups=1000(diego),4(adm),24(cdrom),27(sudo)
```

① **cat /etc/passwd** muestra base de datos de usuarios del sistema
② **getent passwd** obtiene usuarios con UID >= 1000 (usuarios humanos, no del sistema)
③ **groups** lista todos los grupos a los que pertenece un usuario
④ **cat /etc/group** muestra base de datos de grupos del sistema
⑤ **id** muestra identidad del usuario actual (UID, GID y grupos)

25.3.3 1.3 Crear Usuarios Nuevos

Crear usuario básico:

Listing 25.3 BASH

```
# **Crear usuario 'carlos' con home en /home/carlos**
sudo useradd -m -s /bin/bash carlos                                    ①

# **Asignar contraseña**
sudo passwd carlos                                                     ②
# Te pedirá que ingreses y confirmes la contraseña
```

① **useradd -m** crea home directory automáticamente, **-s /bin/bash** asigna shell por defecto
② **passwd** establece contraseña interactiva para el usuario creado

Crear usuario con información adicional:

Listing 25.4 BASH

```
# **Crear usuario con comentario (nombre completo)**
sudo useradd -m -s /bin/bash -c "Carlos López - Administrador" carlos

# **Ver el comentario**
grep carlos /etc/passwd
# Salida: carlos:x:1001:1001:Carlos López - Administrador:/home/carlos:/bin/bash
```

Crear usuario sin contraseña (para scripts):

Listing 25.5 BASH

```
# **Crear usuario de servicio**
sudo useradd -r -s /sbin/nologin nginx_backup

# **-r**: usuario del sistema (UID < 1000)
# **-s /sbin/nologin**: sin shell, no puede loguearse
```

Crear usuario con home en ubicación personalizada:

25.3.4 1.4 Modificar Usuarios

Cambiar información de usuario:

Listing 25.6 BASH

```
# **Para almacenar datos especiales**
sudo useradd -m -d /srv/apps/myapp -s /bin/bash appuser
```

Listing 25.7 BASH

```
# **Cambiar el nombre completo**
sudo usermod -c "Carlos Manuel López" carlos

# **Cambiar el shell de login**
sudo usermod -s /bin/zsh carlos

# **Agregar usuario a grupo existente**
sudo usermod -aG adm carlos
# **-a**: append (agregar SIN remover otros grupos)
# **-G**: grupo suplementario

# **Cambiar el home del usuario**
sudo usermod -d /home/carlos_nuevo carlos

# **Loquear usuario (deshabilitar temporalmente)**
sudo usermod -L carlos

# **Desbloquear usuario**
sudo usermod -U carlos

# **Ver cambios**
grep carlos /etc/passwd
```

25.3.5 1.5 Crear y Gestionar Grupos

Crear grupo nuevo:

Eliminar grupo:

25.3.6 1.6 Eliminar Usuarios

Opciones de eliminación:

Listing 25.8 BASH

```
# **Crear grupo de administradores locales**
sudo groupadd administradores

# **Agregar usuarios al grupo**
sudo usermod -aG administradores diego
sudo usermod -aG administradores carlos

# **Verificar miembros del grupo**
getent group administradores
# Salida: administradores:x:1002:diego,carlos

# **Cambiar grupo primario de usuario**
sudo usermod -g administradores diego
# Cuidado: esto cambia el grupo por defecto para archivos nuevos
```

Listing 25.9 BASH

```
# **Eliminar grupo vacío**
sudo groupdel administradores

# **Si el grupo tiene archivos, primero cambiar propietario**
sudo chown -R :nuevogrupo /ruta/con/archivos
sudo groupdel viejogrupo
```

25.4 2. SSH y Acceso Remoto Seguro

SSH (Secure Shell) es la forma profesional de administrar servidores Linux de forma remota.

25.4.1 2.1 Por Qué SSH es Esencial

```
NUNCA usar telnet, rsh, o contraseña sobre red abierta
SIEMPRE usar SSH con claves criptográficas
```

Ventajas de SSH:

- **Encriptación**: Todo el comunicado está encriptado en tránsito
- **Autenticación por clave**: No hay contraseñas transmitidas por la red
- **Automatización**: Scripts pueden ejecutarse sin interacción manual
- **Port forwarding**: Túneles seguros para acceder a servicios remotos
- **SCP/SFTP**: Transferencia segura de archivos entre máquinas

Listing 25.10 BASH

```
# **Eliminar usuario PERO conservar su home**
sudo userdel carlos

# **Eliminar usuario Y su home**
sudo userdel -r carlos
# **CUIDADO**: Esto elimina /home/carlos y todos sus archivos

# **Crear script para backupear antes de eliminar**
sudo tar czf /backup/carlos_$(date +%Y%m%d).tar.gz /home/carlos
sudo userdel -r carlos
```

25.4.2 2.2 Generar Claves SSH

En tu máquina LOCAL (laptop/desktop):

Listing 25.11 BASH

```
# **Generar par de claves (pública + privada)**
ssh-keygen -t ed25519 -C "diego@abacom.com" -f ~/.ssh/diego_abacom

# **Alternativa si ed25519 no es soportado (legacy)**
ssh-keygen -t rsa -b 4096 -C "diego@abacom.com" -f ~/.ssh/diego_legacy

# **Opciones explicadas**:
# **-t ed25519**: Tipo moderno y seguro (recomendado)
# **-C "diego@abacom.com"**: Comentario/email para identificar
# **-f ~/.ssh/diego_abacom**: Ubicación y nombre del archivo
```

El sistema preguntará por passphrase (contraseña para la clave):

```
Enter passphrase (empty for no passphrase): [DIGITA TU PASSPHRASE]
Enter same passphrase again: [CONFIRMA]
```

Resultado - Dos archivos creados:

Ver contenido de claves:

25.4.3 2.3 Copiar Clave Pública al Servidor

OPCIÓN 1: Comando automático (RECOMENDADO)

OPCIÓN 2: Manual (si ssh-copy-id no funciona)

Listing 25.12 BASH

```
# **Ver archivos creados**
ls -la ~/.ssh/

# **-rw-------  1 diego diego 464 Jan 29 10:30 .ssh/diego_abacom**
# **-rw-r--r--  1 diego diego 102 Jan 29 10:30 .ssh/diego_abacom.pub**

# **Privada**: NUNCA compartir, guardar en lugar seguro
# **Pública**: Distribuir al servidor
```

Listing 25.13 BASH

```
# **VER CLAVE PÚBLICA (seguro compartir)**
cat ~/.ssh/diego_abacom.pub
# ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIDk... diego@abacom.com

# **NUNCA mostrar clave privada en público**
cat ~/.ssh/diego_abacom  #   NUNCA, NUNCA, NUNCA
```

25.4.4 2.4 Conectar sin Contraseña

Desde tu máquina local:

Configurar alias SSH (opcional pero útil):

Crear/editar ~**/.ssh/config**:

```
Host abacom-prod
    HostName 192.168.1.100
    User diego
    IdentityFile ~/.ssh/diego_abacom
    Port 22

Host abacom-backup
    HostName 192.168.1.101
    User diego
    IdentityFile ~/.ssh/diego_abacom
    Port 22
```

Usar alias:

① **scp** copia un archivo al home remoto usando el alias `abacom-prod` de `~/.ssh/config`.
② **-i** fuerza una clave especifica (util si tienes multiples llaves o no usas alias).
③ **-r** copia directorios recursivamente.
④ En `origen:destino`, poner el alias/host antes del path descarga el archivo desde el servidor.

Listing 25.14 BASH

```
# **Copiar clave pública al servidor automáticamente**
ssh-copy-id -i ~/.ssh/diego_abacom.pub diego@192.168.1.100

# **-i**: Especificar archivo de clave
# **diego**: Usuario en el servidor
# **192.168.1.100**: IP del servidor

# **Te pedirá contraseña UNA VEZ**
# After that, you can connect without password
```

Listing 25.15 BASH

```
# **En el servidor, crear directorio .ssh si no existe**
ssh diego@192.168.1.100 "mkdir -p ~/.ssh"

# **Copiar clave desde local al servidor**
cat ~/.ssh/diego_abacom.pub | \
  ssh diego@192.168.1.100 "cat >> ~/.ssh/authorized_keys"

# **Fijar permisos correctos en servidor (critico!)**
ssh diego@192.168.1.100 "chmod 700 ~/.ssh && chmod 600 ~/.ssh/authorized_keys"
```

Recomendación

Para transferencias grandes o repetitivas, usa **rsync**: es mas rapido, reintenta, y solo envia diferencias.

① **rsync** con `-avz` preserva metadata, muestra detalle, comprime, y con `--partial` permite reanudar.
② –**exclude** evita copiar rutas que no necesitas (reduce tiempo y ancho de banda).
③ `origen remoto : destino local` descarga arboles completos de forma eficiente.

25.4.5 2.5 Seguridad: Deshabilitar Acceso por Contraseña

En el servidor, editar /etc/ssh/sshd_config****:

Cambios a hacer (buscar con Ctrl+W en nano):

Aplicar cambios:

IMPORTANTE: Antes de desabilitar contraseña, **asegurate de que tienes clave pública funcionando**. Si no, quedarás bloqueado.

Listing 25.16 BASH

```
# **Conectar usando clave específica**
ssh -i ~/.ssh/diego_abacom diego@192.168.1.100

# **Verificar que funciona**
ssh -i ~/.ssh/diego_abacom diego@192.168.1.100 "whoami"
# Salida: diego
```

Listing 25.17 BASH

```
# **Ahora conectar es fácil**
ssh abacom-prod
ssh abacom-backup

# **Copiar archivos con SCP también funciona**
scp miarchivo.txt abacom-prod:~/                                    ①

# Copiar un archivo usando una clave especifica
scp -i ~/.ssh/diego_abacom miarchivo.txt diego@192.168.1.100:~/     ②

# Copiar un directorio completo (recursivo)
scp -r carpeta/ abacom-prod:~/                                      ③

# Copiar desde el servidor hacia tu maquina local
scp abacom-prod:/var/log/syslog ./syslog                            ④
```

25.5 3. Permisos Unix (Chmod, Chown)

25.5.1 3.1 Entender Permisos

Cada archivo/carpeta tiene 10 caracteres de permisos:

```
drwxr-xr-x

      Otros: x (ejecución)
      Otros: - (sin escritura)
      Otros: r (lectura)
      Grupo: x (ejecución)
      Grupo: - (sin escritura)
      Grupo: r (lectura)
      Usuario: x (ejecución)
      Usuario: w (escritura)
      Usuario: r (lectura)
```

Tipos de archivo:

Listing 25.18 BASH

```
# Sincronizar un directorio al servidor (solo diferencias)
rsync -avz --progress --partial -e "ssh -i ~/.ssh/diego_abacom" carpeta/ diego@192.168.1.

# Excluir archivos (ej: caches) durante la sincronizacion
rsync -avz --progress --partial --exclude '.cache/' -e "ssh -i ~/.ssh/diego_abacom" carpe

# Traer logs desde el servidor a local
rsync -avz --progress -e "ssh -i ~/.ssh/diego_abacom" diego@192.168.1.100:/var/log/ ./log
```

Listing 25.19 BASH

```
# **Editar archivo SSH**
sudo nano /etc/ssh/sshd_config

# **Buscar y cambiar estas líneas**:
```

- **-** → archivo regular
- **d** → directorio
- **l** → symlink (enlace simbólico)
- **c** → dispositivo carácter
- **b** → dispositivo bloque

Ejemplo práctico:

25.5.2 3.2 Cambiar Permisos con Chmod

Sintaxis numérica (más común):

Casos de uso comunes:

Sintaxis literal (más legible pero menos común):

Cambiar permisos recursivamente:

25.5.3 3.3 Cambiar Propietario con Chown

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

Ejemplo práctico - Transferir propiedad de carpeta:

Listing 25.20 INI

```
# **ANTES:**
#PasswordAuthentication yes
#PermitRootLogin yes

# **DESPUÉS:**
PasswordAuthentication no
PermitRootLogin no
PubkeyAuthentication yes
```

Listing 25.21 BASH

```
# **Verificar sintaxis antes de reiniciar**
sudo sshd -t
# Si no hay errores, no muestra nada

# **Reiniciar SSH**
sudo systemctl restart ssh

# **Verificar que SSH está corriendo**
sudo systemctl status ssh
```

25.6 4. Sudo: Privilegios de Administrador

25.6.1 4.1 Por Qué NO Usar Root Directamente

```
Conectarse como root
Ejecutar TODOS los comandos como root
Compartir contraseña de root

Usuarios normales + sudo cuando sea necesario
Auditoría: Saber quién hizo qué
Contraseña individual por usuario
Control granular de qué puede hacer cada usuario
```

Riesgo del ejemplo:

25.6.2 4.2 Configurar Sudo

Editar sudoers (SIEMPRE usar visudo, no nano/vim directo):

Agregar al final del archivo:

Listing 25.22 BASH

```
# **Ver permisos de archivos**
ls -l ~/

# Salida:
# -rw-r--r--  1 diego diego  1024 Jan 29 archivo.txt
# drwxr-xr-x  2 diego diego  4096 Jan 29 carpeta/
# lrwxrwxrwx  1 diego diego    10 Jan 29 enlace -> archivo.txt
```

Listing 25.23 BASH

```
# **Formato: chmod XXX archivo**
# **X = Dígito 0-7 (suma de permisos)**

chmod 755 script.sh
# Usuario: 7 (4+2+1 = rwx)
# Grupo:   5 (4+1   = r-x)
# Otros:   5 (4+1   = r-x)

chmod 644 documento.txt
# Usuario: 6 (4+2   = rw-)
# Grupo:   4 (4     = r--)
# Otros:   4 (4     = r--)

chmod 700 archivo_privado
# Usuario: 7 (4+2+1 = rwx) - Solo propietario
# Grupo:   0 (      = ---)
# Otros:   0 (      = ---)
```

```
# **Permitir que usuario carlos use TODOS los comandos sin contraseña**
carlos ALL=(ALL) NOPASSWD:ALL

# **Permitir que usuario diego use sudo CON contraseña**
diego ALL=(ALL) ALL

# **Permitir que grupo administradores use sudo sin contraseña**
%administradores ALL=(ALL) NOPASSWD:ALL

# **Permitir comandos específicos sin contraseña**
diego ALL=(ALL) NOPASSWD:/usr/bin/systemctl restart nginx

# **Permitir comando con argumentos específicos**
carlos ALL=(ALL) NOPASSWD:/sbin/reboot, /sbin/halt

# **Negar comando específico**
```

Listing 25.24 BASH

```
# **Archivo ejecutable para todos**
chmod 755 /usr/local/bin/micomando

# **Archivo privado (solo dueño)**
chmod 600 ~/.ssh/authorized_keys

# **Directorio accesible (necesita x)**
chmod 755 ~/documentos/

# **Script privado de usuario**
chmod 700 ~/scripts/privado.sh

# **Archivo compartido en grupo**
chmod 640 /etc/app/config.conf
# Usuario: 6 (rw-)
# Grupo:   4 (r--)
# Otros:   0 (---)
```

```
diego ALL=(ALL) ALL, !/usr/bin/rm
```

Guardar: Ctrl+X, luego Y en nano; o :wq en vim

Verificar configuración:

25.6.3 4.3 Usar Sudo Correctamente

Ejecutar comando individual:

Ejecutar como otro usuario:

25.7 5. Laboratorio Práctico: Configurar Usuario Nuevo

25.7.1 5.1 Escenario

Tienes un nuevo administrador que ingresa a Abacom: **María García**.

Necesitas:

1. Crear usuario 'maria' con home en **/home/maria**
2. Darle acceso SSH con clave
3. Hacerla miembro del grupo **administradores**

Listing 25.25 BASH

```
# **Agregar permiso de ejecución**
chmod +x script.sh

# **Remover permiso de escritura para grupo y otros**
chmod go-w documento.txt

# **Dar todos los permisos a todos**
chmod a+rwx archivo.txt

# **Quitar todo excepto lectura**
chmod a-x archivo.txt
chmod a-w archivo.txt
```

Listing 25.26 BASH

```
# **Cambiar permisos de carpeta y contenido**
chmod -R 755 ~/miproyecto/

# **-R**: recursivo (incluye subcarpetas)

# **CUIDADO**: A veces quieres permisos diferentes para carpetas vs archivos
# Carpetas necesitan 'x' para entrar, archivos no

# **Mejor forma: archivos=644, carpetas=755**
find ~/miproyecto -type f -exec chmod 644 {} \;
find ~/miproyecto -type d -exec chmod 755 {} \;
```

4. Configurar sudo sin contraseña
5. Deshabilitar acceso por contraseña

25.7.2 5.2 Pasos

PASO 1: En tu máquina local, generar clave para María

PASO 2: En el servidor, crear usuario María

PASO 3: Copiar clave SSH de María

PASO 4: Permitir sudo sin contraseña para administradores

PASO 5: Deshabilitar login por contraseña (opcional pero recomendado)

PASO 6: Verificar que funciona

Listing 25.27 BASH

```
# **Cambiar propietario a otro usuario**
sudo chown carlos archivo.txt

# **Cambiar propietario Y grupo**
sudo chown carlos:administradores archivo.txt

# **Cambiar solo grupo**
sudo chown :administradores archivo.txt

# **Cambiar recursivamente**
sudo chown -R carlos:administradores ~/proyectos/

# **Cambiar a usuario actual**
sudo chown $USER archivo.txt

# **Cambiar a usuario con su grupo primario**
sudo chown carlos: archivo.txt  # Nota el : al final
```

Listing 25.28 BASH

```
# **Carlos entrega su proyecto a Diego**
sudo chown -R diego:administradores /home/carlos/proyecto_terminado

# **Verificar cambio**
ls -la /home/carlos/
# drwxr-xr-x  3 diego administradores  4096 Jan 29 proyecto_terminado/
```

25.8 Ejemplos Prácticos Multi-SO

25.8.1 Ejemplo 1: Ver Usuarios del Sistema

25.8.1.1 Linux (Ubuntu)

25.8.1.2 macOS (Darwin)

① **dscl** (Directory Service command line) es herramienta de macOS para gestionar usuarios

25.8.1.3 Windows (PowerShell)

Comparación: | SO | Comando | Ubicación | |—-|———|———| | Linux | `cat /etc/passwd` | `/etc/passwd` | | macOS | `dscl . -list /Users` | Directory Services | | Windows | `Get-LocalUser` | SAM (Security Account Manager) |

Listing 25.29 BASH

```
# **Usuario distrait comete error como root**
sudo rm -rf /var/log/*  # **INTENTA LIMPIAR LOGS**
# vs
sudo rm -rf /   # **ACCIDENTAL TECLEA ESTO EN VEZ**
# **¡TODO EL SERVIDOR ELIMINADO!**
```

Listing 25.30 BASH

```
# **Abrir editor de sudoers de forma segura**
sudo visudo

# **Buscar la sección**:
```

25.8.2 Ejemplo 2: Crear Usuario Nuevo

25.8.2.1 Linux (Ubuntu)

25.8.2.2 macOS (Darwin)

① **dscl** es la forma de crear usuarios en macOS sin GUI

25.8.2.3 Windows (PowerShell)

Comparación: | SO | Comando | Complejidad | |—-|———|———| | Linux | `useradd -m carlos` | Simple | | macOS | `dscl . -create /Users/carlos` | Complejo | | Windows | `New-LocalUser -Name carlos` | Simple |

25.8.3 Ejemplo 3: Caso Real Abacom - Crear Usuario de Servicio

25.8.3.1 Linux (Servidor Ubuntu)

25.8.3.2 macOS (Servidor macOS - Deprecated)

25.8.3.3 Windows (Windows Server)

Comparación arquitectura de servicios: | Aspecto | Linux | macOS | Windows | |———|——-|——-|———| | Usuario servicio | `useradd -r` | `dscl` (limitado) | `New-LocalUser` | | Gestor servicios | systemd | launchd | Task Scheduler | | Recomendado para | Producción | Desarrollo | Producción |

Listing 25.31 BASH

```
# **Ver qué sudoers tienes**
sudo -l

# **Salida**:
# User diego may run the following commands on linux-server:
#     (ALL) ALL
```

Listing 25.32 BASH

```
# **Reiniciar servicio**
sudo systemctl restart nginx

# **Te pide contraseña la primera vez**
# Luego recuerda por 15 minutos

# **Ver historial de sudo**
sudo cat /var/log/auth.log | grep sudo

# **Salida**:
# Jan 29 10:30:45 servidor sudo: diego : TTY=pts/0 ;
# PWD=/home/diego ; USER=root ; COMMAND=/usr/bin/systemctl
```

25.9 6. Ejercicios Prácticos

25.9.1 Ejercicio 1: Crear usuario de desarrollo

Crea usuario **dev_app** que:

- Tenga home en **/home/dev_app**
- Pertenezca al grupo **developers** (crear grupo primero)
- Pueda ejecutar solo **systemctl restart app** sin contraseña
- NO pueda ejecutar otros comandos con sudo

Solución:

25.9.2 Ejercicio 2: Asignar permisos correctos

Tienes una carpeta **/var/www/app** con estos requisitos:

- Propietario: usuario **www-data**
- Grupo: **developers**

Listing 25.33 BASH

```
# **Ejecutar como usuario 'www-data' (usuario web)**
sudo -u www-data whoami
# Salida: www-data

# **Útil para testear permisos**
sudo -u www-data ls -la /var/www/
```

Listing 25.34 BASH

```
ssh-keygen -t ed25519 -C "maria@abacom.com" -f ~/.ssh/maria_abacom
# Ingresa passphrase cuando pida
```

- Archivos: legibles por grupo, no editables por otros
- Carpetas: accesibles por grupo

Solución:

25.9.3 Ejercicio 3: Configurar SSH para usuario nuevo

Dado usuario **carlos** en servidor **prod.abacom.com**:

- Generar clave SSH
- Copiar al servidor
- Crear alias SSH
- Verificar acceso sin contraseña

Solución:

25.10 7. Quiz de Verificación

1. **¿Cuál es el permiso 755 en términos legibles?**
   - A) rwxrwxrwx
   - B) rwxr-xr-x
   - C) rw-rw-rw-
   - D) r–r–r–
2. **¿Cuál es el comando para ver qué permisos sudo tiene un usuario?**
   - A) sudo –list
   - B) sudo -l
   - C) sudo -L

Listing 25.35 BASH

```
# **Conectar al servidor**
ssh diego@192.168.1.100

# **Crear usuario**
sudo useradd -m -s /bin/bash -c "María García - Administrador" maria

# **Crear grupo administradores si no existe**
sudo groupadd administradores 2>/dev/null || true

# **Agregar a grupo administradores**
sudo usermod -aG administradores maria

# **Verificar**
groups maria
# maria : maria administradores
```

Listing 25.36 BASH

```
# **Desde tu máquina local**
ssh-copy-id -i ~/.ssh/maria_abacom.pub maria@192.168.1.100

# **Te pide contraseña de maria - usa la que estableciste**
```

- D) Opciones B y C son correctas

3. **¿Por qué es importante deshabilitar acceso SSH por contraseña?**
   - A) Las claves SSH son criptográficamente más seguras
   - B) Protege contra ataques de fuerza bruta
   - C) Contraseñas viajan sin encriptación
   - D) Todas las anteriores

4. **¿Qué hace** sudo usermod -aG administradores diego**?**
   - A) Cambia el grupo primario de diego a administradores
   - B) Agrega diego al grupo administradores (sin remover otros)
   - C) Solo establece diego como administrador del sistema
   - D) Respuesta B es correcta

5. **¿Cuál es la diferencia entre** userdel carlos** y **userdel -r carlos**?**
   - A) El primero solo elimina el usuario, el segundo también elimina /home/carlos
   - B) No hay diferencia
   - C) -r elimina permisos sudo
   - D) -r requiere confirmación

Listing 25.37 BASH

```
# **En el servidor**
sudo visudo

# **Agregar esta línea al final**:
%administradores ALL=(ALL) NOPASSWD:ALL

# **Guardar (Ctrl+X, Y en nano)**
```

Listing 25.38 BASH

```
# **Bloquear contraseña de maría para SSH**
sudo passwd -l maria

# **Ahora María solo puede entrar por clave SSH**
```

25.11 8. Resumen y Siguiente Paso

25.11.1 Lo Aprendido

Crear y gestionar usuarios (**useradd**, **usermod**, **userdel**) Trabajar con grupos para organizar usuarios Configurar SSH con claves criptográficas Entender permisos Unix (**chmod**, **chown**) Usar sudo sin comprometer seguridad

25.11.2 Siguiente: Unidad 2.4

El próximo tema cubre:

- Actualización segura del sistema
- Firewall básico (UFW)
- Hardening de primeros pasos

25.12 Referencias

(Canonical 2024e) (L. Foundation 2024) (Communications 2024) (D. Project 2024b)

Listing 25.39 BASH

```
# **Desde tu máquina local**
ssh -i ~/.ssh/maria_abacom.pub maria@192.168.1.100

# **Dentro del servidor como María**
sudo systemctl status ssh
# Debería funcionar sin pedir contraseña

# **Salir**
exit
```

25.13 Quiz: Configuración Inicial

Listing 25.40 BASH

```
# Ver todos los usuarios del sistema
$ cat /etc/passwd | head -5
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin

# Ver solo usuarios humanos (UID >= 1000)
$ getent passwd | awk -F: '$3 >= 1000 {print $1}'
diego
carlos
admin

# Ver información del usuario actual
$ id
uid=1000(diego) gid=1000(diego) groups=1000(diego),4(adm),27(sudo)

# Ver grupos de un usuario
$ groups carlos
carlos : carlos developers
```

Listing 25.41 BASH

```
# En macOS, usuarios se almacenan en Directory Services
$ dscl . -list /Users | grep -v "^_"                                    ①
root
diego
carlos
admin

# Ver información del usuario actual
$ id
uid=501(diego) gid=20(staff) groups=20(staff),12(everyone)

# Ver grupos de un usuario
$ groups carlos
carlos staff developers

# Nota: macOS NO usa /etc/passwd como Linux
# En su lugar, usa Directory Services (Open Directory)
```

Listing 25.42 POWERSHELL

```
# Ver todos los usuarios locales
PS> Get-LocalUser

Name    Enabled Description
----    ------- -----------
Administrator True   Administrator account
diego   True   Regular user account
carlos  True   Regular user account
Guest   False

# Ver información del usuario actual
PS> $env:USERNAME
diego

# Ver grupos del usuario actual
PS> Get-LocalGroupMember -Group "Administrators"

ObjectClass Name          PrincipalSource
----------- ----          ---------------
User        LAPTOP\diego  Local
```

Listing 25.43 BASH

```
# Crear usuario 'carlos' con home
$ sudo useradd -m -s /bin/bash carlos

# Asignar contraseña
$ sudo passwd carlos
New password: ••••••••
Retype new password: ••••••••
passwd: password updated successfully

# Verificar
$ id carlos
uid=1001(carlos) gid=1001(carlos) groups=1001(carlos)

# Crear usuario con información adicional
$ sudo useradd -m -s /bin/bash -c "Carlos López - Dev" carlos
```

Listing 25.44 BASH

```
# En macOS, crear usuario es más complejo (requiere GUI normalmente)
# Pero se puede hacer vía dscl:

$ sudo dscl . -create /Users/carlos                                    (1)
$ sudo dscl . -create /Users/carlos UserShell /bin/bash
$ sudo dscl . -create /Users/carlos RealName "Carlos López"
$ sudo dscl . -create /Users/carlos UniqueID 501
$ sudo dscl . -create /Users/carlos PrimaryGroupID 20

# Crear home directory
$ sudo mkdir -p /Users/carlos
$ sudo chown carlos:staff /Users/carlos

# Asignar contraseña:
$ sudo dscl . -passwd /Users/carlos contraseña

# Verificar:
$ dscl . -read /Users/carlos
```

Listing 25.45 POWERSHELL

```
# Crear usuario nuevo
PS> $Password = Read-Host -AsSecureString "Enter password"
PS> New-LocalUser -Name "carlos" -Password $Password `
     -FullName "Carlos López" -Description "Developer"

Name    Enabled Description
----    ------- -----------
carlos  True    Developer

# Agregar usuario a un grupo
PS> Add-LocalGroupMember -Group "Developers" -Member "carlos"

# Verificar
PS> Get-LocalUser -Name carlos

Name    Enabled Description
----    ------- -----------
carlos  True    Developer
```

Listing 25.46 BASH

```
# Crear usuario de servicio para aplicación web
$ sudo useradd -r -s /sbin/nologin -d /var/www/app webapp_abacom

# Verificar que NO puede conectarse
$ su - webapp_abacom
# Error: /sbin/nologin
# Nologin previene acceso interactivo

# Asignar permisos a carpetas de aplicación
$ sudo chown -R webapp_abacom:www-data /var/www/app
$ sudo chmod 755 /var/www/app
$ sudo find /var/www/app -type f -exec chmod 644 {} \;

# Verificar
$ ls -la /var/www/app | head -5
drwxr-xr-x  webapp_abacom www-data  /var/www/app
```

Listing 25.47 BASH

```
# macOS ya NO soporta Server.app (fue discontinued en 2023)
# Pero si quisieras crear usuario de servicio:

$ sudo dscl . -create /Users/webapp_abacom
$ sudo dscl . -create /Users/webapp_abacom UserShell /usr/sbin/nologin
$ sudo dscl . -create /Users/webapp_abacom RealName "Abacom Web App"

# Mejor: Usar Docker o Kubernetes en lugar de macOS para servidores
```

Listing 25.48 POWERSHELL

```
# Crear cuenta de servicio (service account) para aplicación
PS> $ServicePassword = ConvertTo-SecureString "Complex!Pass123" -AsPlainText -Force
PS> New-LocalUser -Name "svc_abacom" -Password $ServicePassword `
     -Description "Service account for Abacom Web App" `
     -PasswordNeverExpires

# Asignar permisos NTFS
PS> $ACL = Get-Acl "C:\www\app"
PS> $AccessRule = New-Object System.Security.AccessControl.FileSystemAccessRule `
     ("svc_abacom", "Modify", "ContainerInherit,ObjectInherit", "None", "Allow")
PS> $ACL.AddAccessRule($AccessRule)
PS> Set-Acl "C:\www\app" $ACL

# Crear tarea planificada (equivalente a systemd service)
PS> $TaskAction = New-ScheduledTaskAction -Execute "C:\app\webapp.exe"
PS> $TaskTrigger = New-ScheduledTaskTrigger -AtStartup
PS> Register-ScheduledTask -TaskName "AbacomWebApp" `
     -Action $TaskAction -Trigger $TaskTrigger `
     -User "svc_abacom" -Password $ServicePassword
```

Listing 25.49 BASH

```
sudo groupadd developers
sudo useradd -m -s /bin/bash -g developers dev_app
sudo usermod -aG developers dev_app
sudo visudo
# Agregar: dev_app ALL=(ALL) NOPASSWD:/usr/bin/systemctl restart app
```

Listing 25.50 BASH

```
sudo chown -R www-data:developers /var/www/app
sudo chmod -R 755 /var/www/app  # Carpetas
sudo find /var/www/app -type f -exec chmod 644 {} \;  # Archivos
# Resultado: Grupo puede leer/ejecutar, otros solo leer carpetas
```

Listing 25.51 BASH

```
# Local
ssh-keygen -t ed25519 -C "carlos@abacom.com" -f ~/.ssh/carlos_prod

# Copiar clave
ssh-copy-id -i ~/.ssh/carlos_prod.pub carlos@prod.abacom.com

# Crear alias (~/.ssh/config)
echo "Host abacom-prod
    HostName prod.abacom.com
    User carlos
    IdentityFile ~/.ssh/carlos_prod" >> ~/.ssh/config

# Verificar
ssh abacom-prod whoami
# carlos
```

26 Actualización y Seguridad del Sistema

Listing 26.1 `QUARTO-TITLE-BLOCK`

Mantener Ubuntu Seguro y Actualizado

26.1 Objetivos de Aprendizaje

Después de completar este tema, serás capaz de:

- Actualizar el sistema Linux de forma segura
- Entender las diferencias entre **update**, **upgrade**, y **dist-upgrade**
- Configurar actualizaciones automáticas
- Instalar y usar firewall UFW
- Aplicar hardening básico de seguridad
- Monitorear cambios de seguridad

26.2 Por Qué Este Tema Es Crítico

Un servidor sin actualizar es un servidor comprometido. **0-day exploits** (vulnerabilidades nuevas) se descubren regularmente, y los parches salen diariamente. Tu servidor debe estar actualizado.

Estadística real: El 80% de brechas de seguridad podrían haberse prevenido con actualizaciones oportunas.

Tiempo estimado: 60-90 minutos de lectura + laboratorio

26.3 Ejemplos Prácticos Multi-SO

26.3.1 Ejemplo 1: Actualizar Sistema Operativo

26.3.1.1 Linux (Ubuntu)

① **apt update** descarga la lista de paquetes disponibles de repositorios remotos (sin instalar nada)

Listing 26.2 BASH

```
# Paso 1: Actualizar lista de paquetes
$ sudo apt update                                                    ①
Get:1 http://security.ubuntu.com/ubuntu jammy-security InRelease [114 kB]
Leyendo listas de paquetes... Hecho

# Paso 2: Actualizar paquetes instalados
$ sudo apt upgrade -y                                                ②
Se instalarán las siguientes actualizaciones:
  openssl ssh systemd vim curl wget
0 paquetes recién instalados, 5 para actualizar
¿Deseas continuar? [S/n] S

# Paso 3: Limpiar paquetes no usados
$ sudo apt autoremove -y                                             ③
Los siguientes paquetes serán ELIMINADOS:
  libpython3.8-minimal
¿Deseas continuar? [S/n] S

# Verificar versión del kernel después
$ uname -r                                                           ④
5.15.0-89-generic
```

② **apt upgrade** instala versiones más recientes de paquetes ya instalados (-y responde "sí" automáticamente)
③ **apt autoremove** elimina dependencias instaladas pero ya no necesitadas (limpia espacio)
④ **uname -r** muestra la versión actual del kernel Linux

26.3.1.2 macOS (Darwin)

① **softwareupdate** es el gestor de actualizaciones de SO en macOS
② **brew** es el package manager externo más popular en macOS (similar a apt en Linux)

26.3.1.3 Windows (PowerShell)

Comparación: | SO | Comando | Gestor | Automático | |—-|———|———|———| | Linux | `apt update && apt upgrade` | apt | Configurable | | macOS | `softwareupdate -ia` | App Store + brew | Sí, con popup | | Windows | `Install-WindowsUpdate` | Windows Update | Sí, forzado |

Listing 26.3 BASH

```
# macOS usa dos sistemas de actualización:

# Sistema A: Actualizaciones de SO (via App Store / Sistema)
$ softwareupdate -l                                                    ①
The following updates are available:
  macOS Sonoma 14.2.1 Update
  Security Update 2024-001

# Instalar todas las actualizaciones
$ sudo softwareupdate -ia
Installing macOS Sonoma 14.2.1 Update
# Requiere reinicio

# Sistema B: Homebrew (package manager de terceros)
$ brew update                                                          ②
Updated 2 taps and 215 formulae

$ brew upgrade
Upgrading curl, openssl, wget
```

26.3.2 Ejemplo 2: Verificar Vulnerabilidades de Seguridad

26.3.2.1 Linux (Ubuntu)

① **apt log** muestra el historial de instalaciones y actualizaciones

26.3.2.2 macOS (Darwin)

26.3.2.3 Windows (PowerShell)

Comparación: | SO | Herramienta | Comando | Frecuencia | |—-|————|———|————| | Linux | unattended-upgrades | `apt upgrade` | Configurable (diario típico) | | macOS | Software Update | `softwareupdate -l` | Automático (con notificación) | | Windows | Windows Update | Automático | Forzado (Patch Tuesday) |

26.3.3 Ejemplo 3: Caso Real Abacom - Configurar Firewall y Seguridad

26.3.3.1 Linux (UFW Firewall)

① **ufw enable** activa el firewall y configura inicio automático al bootear
② **ufw status** muestra reglas activas (puerto, acción, origen)
③ **ufw allow** abre un puerto específico (SSH:22, HTTP:80, HTTPS:443 son estándar)

Listing 26.4 POWERSHELL

```
# Windows tiene actualización automática (Windows Update)
# Pero también puedes forzar actualizaciones vía PowerShell:

# Ver actualizaciones disponibles
PS> Get-WindowsUpdate
Title                                   KB        Size
-----                                   --        ----
Cumulative Update for Windows 11         KB5034441  800MB
Security Update for .NET Framework 4.8 KB5034440  150MB

# Instalar actualizaciones (requiere privilegios admin)
PS> Install-WindowsUpdate -AcceptAll -AutoReboot

# Verificar versión del SO
PS> [Environment]::OSVersion.Version
10.0.22621.0  # Windows 11 Build 22621
```

④ **ufw default deny/allow** configura política por defecto (rechazar entrada, permitir salida)
⑤ **tail -f** muestra en tiempo real los intentos bloqueados por el firewall

26.3.3.2 macOS (Built-in Firewall)

26.3.3.3 Windows (Windows Defender Firewall)

Comparación arquitectura: | Aspecto | Linux (UFW) | macOS | Windows | |———|————|——-|———| | Facilidad | Simple | Complejo | GUI intuitiva | | Control | Total | Limitado | Granular | | Producción | Ideal | No | Viable | | Port blocking | Sí | Difícil | Sí |

26.4 1. Gestión de Paquetes con APT (Debian/Ubuntu)

26.4.1 1.1 Entender APT (Advanced Package Tool)

APT es el gestor de paquetes estándar en Debian y Ubuntu. Mantiene un índice de repositorios remotos.

Listing 26.5 BASH

```
# Ver historial de parchesaplicados
$ sudo apt log                                                    ①
# O ver el archivo de log:
$ sudo tail -f /var/log/apt/history.log

# Verificar CVEs (Common Vulnerabilities) en paquetes instalados
$ sudo apt list --upgradable
Listado de paquetes con actualizaciones disponibles:
curl/jammy 7.81.0-1ubuntu1.14 7.81.0-1ubuntu1.15
openssh-client/jammy 1:8.2p1-4ubuntu0.5 1:8.2p1-4ubuntu0.6

# Instalar ubuntu-security-status (más información)
$ sudo apt install ubuntu-security-status
$ ubuntu-security-status --json
# Muestra vulnerabilidades conocidas

# Ver vulnerabilidades recientes
$ sudo unattended-upgrade --debug
# Muestra qué está siendo parcheado automáticamente
```

26.4.2 1.2 Actualizar Repositorios

① **apt update** descarga lista actualizada de paquetes disponibles en repositorios (sin instalar nada)

Verificar repositorios configurados:

26.4.3 1.3 Actualizar Paquetes

OPCIÓN 1: Upgrade (seguro - no elimina paquetes)

OPCIÓN 2: Full-Upgrade (más agresivo - puede remover paquetes)

OPCIÓN 3: Dist-Upgrade (Actualizar distribución - CUIDADO)

26.4.4 1.4 Limpiar Paquetes No Usados

26.4.5 1.5 Actualizar Paquete Específico

Listing 26.6 BASH

```
# macOS tiene vulnerabilidades de seguridad reportadas en:
$ sudo softwareupdate -l
# Mostrada en formato: "Security Update XXXX-XXX"

# Para análisis más detallado, usar herramientas de terceros:
$ brew install clamav  # Antivirus
$ freshclam  # Actualizar base de datos de virus
$ clamscan -ri /Users  # Escanear el sistema

# Verificar historial de updatesde seguridad
$ log show --predicate 'eventMessage contains "Update"' --last 1h
```

26.5 2. Actualizaciones Automáticas

26.5.1 2.1 Instalar Unattended-Upgrades

En producción, quieres actualizaciones automáticas sin esperar:

① **unattended-upgrades** es el paquete que permite actualizaciones automáticas sin intervención
② **systemctl enable/start** configura el servicio para ejecutarse automáticamente en el boot sudo systemctl start unattended-upgrades

Listing 26.7 POWERSHELL

```
# Windows tiene Windows Security (built-in)
# Verificar estado desde PowerShell:

PS> Get-MpComputerStatus | Select-Object RealTimeProtectionEnabled, OnAccessProtectionEna
RealTimeProtectionEnabled    OnAccessProtectionEnabled
-------------------------  -------------------------
True                         True

# Ver historial de updates
PS> Get-WinEvent -LogName System -FilterXPath "*[System[(EventID=19)]]" | Select-Object T
TimeCreated             Message
-----------             -------
2024-01-29 10:30:00     Update KB5034441 installed successfully

# Verificar si hay updates pendientes
PS> (New-Object -ComObject Microsoft.Update.Session).CreateupdateSearcher().Search("IsHid
# Si devuelve 0: Sistema actualizado
```

Listing 26.8 BASH

```
# En Abacom, usamos UFW para proteger servidores

# Paso 1: Habilitar firewall
$ sudo ufw enable                                                    ①
Firewall is active and enabled on system startup

# Paso 2: Ver estado
$ sudo ufw status                                                    ②
Status: active
To                         Action      From
--                         ------      ----
22/tcp                     ALLOW       Anywhere
80/tcp                     ALLOW       Anywhere
443/tcp                    ALLOW       Anywhere

# Paso 3: Permitir puertos específicos
$ sudo ufw allow 22/tcp                                              ③
$ sudo ufw allow 80/tcp
$ sudo ufw allow 443/tcp

# Paso 4: Denegar todo lo demás (default)
$ sudo ufw default deny incoming                                     ④
$ sudo ufw default allow outgoing

# Verificar firewall log
$ sudo tail -f /var/log/ufw.log                                      ⑤
```

Listing 26.9 BASH

```
# macOS tiene firewall built-in (menos potente que UFW)

# Habilitar firewall
$ sudo /usr/libexec/ApplicationFirewall/socketfilterfw --setglobalstate on
Firewall is now enabled

# Ver estado
$ sudo /usr/libexec/ApplicationFirewall/socketfilterfw --getglobalstate
Firewall is enabled

# Permitir aplicación específica
$ sudo /usr/libexec/ApplicationFirewall/socketfilterfw --add /usr/bin/python3

# Mejor opción: Usar herramientas de terceros como Hands Off! o Little Snitch
# O usar pfctl (Advanced BSD firewall)
$ sudo pfctl -e  # Habilitar PF firewall
```

Listing 26.10 POWERSHELL

```
# En Windows Server, usar Windows Defender Firewall

# Ver estado del firewall
PS> Get-NetFirewallProfile

Name            Enabled
----            -------
Domain          True
Private         True
Public          True

# Crear regla de firewall (permitir SSH en puerto 22)
PS> New-NetFirewallRule -DisplayName "Allow SSH" `
     -Direction Inbound -LocalPort 22 -Protocol TCP -Action Allow

# Ver reglas activas
PS> Get-NetFirewallRule -Direction Inbound -Action Allow | Select-Object DisplayName

# Más seguro: Denegar todo por defecto
PS> Set-NetFirewallProfile -DefaultInboundAction Block -DefaultOutboundAction Allow
```

Listing 26.11 BASH

```
# **El ciclo de actualización**
apt update    → Descargar listas de paquetes nuevos
apt upgrade   → Instalar versiones nuevas (sin remover paquetes)
apt autoremove → Eliminar dependencias no utilizadas
```

Listing 26.12 BASH

```
# **Descargar listas de paquetes desde repositorios**
sudo apt update                                                    ①

# **Salida esperada**:
# Get:1 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
# Get:2 http://archive.ubuntu.com/ubuntu focal-updates InRelease [99.8 kB]
# Leyendo listas de paquetes... Hecho
# Se construirán 312 nuevos paquetes

# **¿Qué hace?**
# - Conecta a repositorios configurados en /etc/apt/sources.list
# - Descarga lista de paquetes disponibles y versiones
# - Si ves "Get", está descargando; "Hit" = sin cambios
```

Listing 26.13 BASH

```
# **Ver fuentes APT**
cat /etc/apt/sources.list

# **Salida típica**:
# deb http://archive.ubuntu.com/ubuntu/ jammy main restricted
# deb http://archive.ubuntu.com/ubuntu/ jammy-updates main restricted
# deb http://security.ubuntu.com/ubuntu jammy-security main restricted universe

# **Carpeta de repositorios adicionales (PPA)**
ls -la /etc/apt/sources.list.d/
```

Listing 26.14 BASH

```
# **Actualizar paquetes instalados a versión más reciente**
sudo apt upgrade

# **Salida**:
# Leyendo listas de paquetes... Hecho
# Creando árbol de dependencias
# Obteniendo información del estado... Hecho
# Se instalarán las siguientes actualizaciones:
#   openssl ssh systemd vim curl wget
# 0 paquetes recién instalados, 5 para actualizar, 0 para remover
# ¿Deseas continuar? [S/n] S

# **Ventaja**: Nunca rompe dependencias
# **Desventaja**: A veces no puede instalar actualizaciones importantes
```

Listing 26.15 BASH

```
# **Actualización más completa (puede remover dependencias antiguas)**
sudo apt full-upgrade

# **Diferencia**: Si openssl 3.0 necesita remover openssl 1.1,
# full-upgrade lo hará; upgrade NO
```

Listing 26.16 BASH

```
# **Saltar a versión mayor de Ubuntu (22.04 -> 24.04)**
sudo apt dist-upgrade
#   SOLO después de backup completo
#   SOLO en ambiente de testing primero

# **Mejor forma de actualizar distribución**:
sudo do-release-upgrade
# Esta herramienta es más segura para saltos mayores
```

Listing 26.17 BASH

```
# **Eliminar paquetes de dependencias que ya no se usan**
sudo apt autoremove

# **Salida**:
# Los siguientes paquetes serán ELIMINADOS:
#   libpython3.8-minimal
# 0 nuevos paquetes instalados, 0 actualizaciones, 1 para remover
# ¿Deseas continuar? [S/n] S

# **Eliminar cachés de paquetes descargados**
sudo apt autoclean

# **Eliminar TODO lo que tenga caché (más agresivo)**
sudo apt clean
```

Listing 26.18 BASH

```
# **Actualizar solo nginx a versión más reciente**
sudo apt install --only-upgrade nginx

# **Instalar versión específica**
sudo apt install nginx=1.24.0-1~jammy

# **Ver versiones disponibles**
apt-cache policy nginx
# nginx:
#   Instalada: 1.18.0-6ubuntu14.3
#   Candidato: 1.24.0-1~jammy
#   Tabla de versión:
#     1.24.0-1~jammy 500
#     1.18.0-6ubuntu14.3 500
```

Listing 26.19 BASH

```
# **Instalar herramienta de actualizaciones automáticas**
sudo apt install unattended-upgrades                                    ①

# **Habilitar servicio**
sudo systemctl enable unattended-upgrades                               ②
sudo systemctl start unattended-upgrades
```

27 Verificar estado

sudo systemctl status unattended-upgrades

```
### 2.2 Configurar Qué Se Actualiza Automáticamente

**Editar configuración**:
```bash
# **Archivo de configuración**
sudo nano /etc/apt/apt.conf.d/50unattended-upgrades
```

Cambios importantes:

Listing 27.1 BASH

```
# **Habilitar solo actualizaciones de seguridad (recomendado)**
Unattended-Upgrade::Package-Blacklist {
};

# **Solo paquetes de seguridad (línea clave)**
Unattended-Upgrade::Origins-Pattern {
    "origin=*,archive=*-security";
};

# **Permitir reinicio automático si es necesario**
Unattended-Upgrade::Automatic-Reboot "true";

# **Horario para reinicio (ej: 3 AM)**
Unattended-Upgrade::Automatic-Reboot-Time "03:00";
```

27.0.1 2.3 Notificar cambios por Email
```

Listing 27.2 BASH

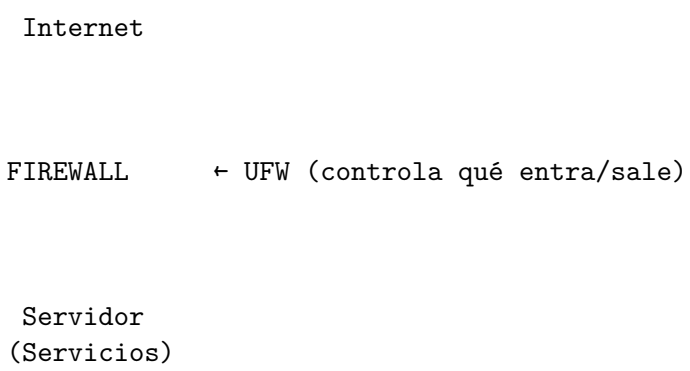
```
# **Instalar herramienta de email**
sudo apt install mailutils

# **En /etc/apt/apt.conf.d/50unattended-upgrades, agregar**:
Unattended-Upgrade::Mail "diego@abacom.com";
Unattended-Upgrade::MailReport "only-on-change";
```

27.1 3. Firewall con UFW

27.1.1 3.1 Entender Firewall

Un firewall es una barrera entre tu servidor y la red. Controla qué conexiones se permiten:



Reglas de firewall:

- **ALLOW**: Permitir conexión
- **DENY**: Bloquear conexión
- **REJECT**: Rechazar (y notificar al cliente)
- **DROP**: Descartar silenciosamente

27.1.2 3.2 Instalar y Habilitar UFW

27.1.3 3.3 Reglas Básicas

Permitir puerto SSH (¡CRÍTICO! Sino quedarás bloqueado):

Permitir puertos web:

Permitir puertos personalizados:

Denegar puertos:

Listing 27.3 BASH

```
# **UFW (Uncomplicated Firewall) viene con Ubuntu**
# **Verificar si está instalado**
sudo apt install ufw

# **Habilitar firewall**
sudo ufw enable

# **Salida**:
# Firewall is active and enabled on system startup.

# **Ver estado**
sudo ufw status

# **Salida cuando está deshabilitado**:
# Status: inactive

# **Salida cuando está habilitado**:
# Status: active
#
# To                         Action      From
# --                         ------      ----
# 22/tcp                     ALLOW       Anywhere
# 80/tcp                     ALLOW       Anywhere
# 443/tcp                    ALLOW       Anywhere
```

27.1.4 3.4 Eliminar Reglas

27.1.5 3.5 Reglas Avanzadas

27.1.6 3.6 Configuración de Inicio/Parada

27.2 4. Hardening Básico

27.2.1 4.1 Actualizar Bootloader

27.2.2 4.2 Deshabilitar Servicios Innecesarios

Listing 27.4 BASH

```
# **Permitir SSH (puerto 22)**
sudo ufw allow 22/tcp
# ó
sudo ufw allow ssh

# **Verificar**
sudo ufw status

# To                         Action      From
# --                         ------      ----
# 22/tcp                     ALLOW       Anywhere
```

Listing 27.5 BASH

```
# **Permitir HTTP (puerto 80)**
sudo ufw allow 80/tcp
sudo ufw allow http

# **Permitir HTTPS (puerto 443)**
sudo ufw allow 443/tcp
sudo ufw allow https

# **Permitir HTTP y HTTPS juntos**
sudo ufw allow "Nginx Full"
```

27.2.3 4.3 Limitar Acceso a Archivos Sensibles

27.2.4 4.4 Audit y SELinux/AppArmor

En Ubuntu, AppArmor es estándar (SELinux es en CentOS):

27.2.5 4.5 Fail2Ban - Proteger contra Brute Force

① **fail2ban** es el sistema que detecta intentos fallidos de login y bloquea IPs automáticamente
② **systemctl enable/start** configura el servicio y lo inicia (enable = arrancar al boot, start = arrancar ahora)
③ **systemctl status** muestra si el servicio está activo (running) o detenido (inactive)
④ **fail2ban-client status sshd** muestra estadísticas: IPs actualmente baneadas y total histórico

Listing 27.6 BASH

```
# **Permitir puerto 8080 (TCP)**
sudo ufw allow 8080/tcp

# **Permitir puerto 5432 (PostgreSQL) solo desde IP específica**
sudo ufw allow from 192.168.1.50 to any port 5432

# **Permitir rango de puertos**
sudo ufw allow 6000:6010/tcp

# **Permitir desde red específica**
sudo ufw allow from 192.168.1.0/24 to any port 3306
```

Listing 27.7 BASH

```
# **Bloquear conexiones SMTP (puerto 25) para prevenir spam**
sudo ufw deny 25/tcp

# **Bloquear puerto desde IP específica**
sudo ufw deny from 203.0.113.1 to any port 22

# **Bloquear todo excepto SSH, HTTP, HTTPS**
sudo ufw default deny incoming
sudo ufw default allow outgoing
sudo ufw allow 22/tcp
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
```

27.3 5. Laboratorio Práctico: Actualizar y Asegurar Servidor

27.3.1 5.1 Escenario

Tienes un servidor Ubuntu recién instalado que necesita:

1. Actualizar todos los paquetes
2. Configurar firewall básico
3. Habilitar actualizaciones automáticas
4. Aplicar hardening inicial

27.3.2 5.2 Pasos

PASO 1: Actualizar repositorios y paquetes

Listing 27.8 BASH

```
# **Listar reglas numeradas**
sudo ufw status numbered

# Num  To                         Action      From
# ---  --                         ------      ----
# 1    22/tcp                     ALLOW IN    Anywhere
# 2    80/tcp                     ALLOW IN    Anywhere
# 3    443/tcp                    ALLOW IN    Anywhere

# **Eliminar regla por número**
sudo ufw delete 2
# Deleting:
#  allow 80/tcp
# Proceed with operation (y|n)? y

# **Eliminar regla por descripción**
sudo ufw delete allow http
```

PASO 2: Instalar Fail2Ban y UFW

PASO 3: Configurar firewall

PASO 4: Configurar actualizaciones automáticas

PASO 5: Verificar seguridad

27.4 6. Monitoreo de Seguridad

27.4.1 6.1 Ver Logs de Actualizaciones

27.4.2 6.2 Auditoría de Cambios Recientes

27.5 7. Ejercicios Prácticos

27.5.1 Ejercicio 1: Crear política de actualización

Define:

Listing 27.9 BASH

```
# **Permitir con límite de conexiones (prevenir brute force)**
sudo ufw limit 22/tcp
# Permite máximo 6 conexiones por 30 segundos

# **Ver reglas en formato verbose**
sudo ufw show added

# Added user rules (see 'ufw status' for running firewall):
# ufw allow 22/tcp
# ufw allow 80/tcp
# ufw allow 443/tcp

# **Resetear UFW a estado por defecto**
sudo ufw reset
```

Listing 27.10 BASH

```
# **Verificar si UFW está habilitado para startup**
sudo ufw status
# Status: active

# **Deshabilitar UFW temporalmente (se reactiva en reboot)**
sudo ufw disable

# **Habilitar de nuevo**
sudo ufw enable

# **Cargar reglas nuevas sin reiniciar**
sudo ufw reload
```

- Cuándo actualizar (horarios)
- Qué actualizar (seguridad, todas)
- Cómo notificar (email)
- Cuándo reiniciar (horario seguro)

Solución: Editar **/etc/apt/apt.conf.d/50unattended-upgrades** con:

27.5.2 Ejercicio 2: Configurar firewall completo

Crea firewall para:

- SSH: Permitir solo desde IP 192.168.1.50
- HTTP/HTTPS: Permitir desde cualquiera
- MySQL: Permitir solo desde red 192.168.1.0/24

Listing 27.11 BASH

```
# **Proteger GRUB con contraseña (previene acceso físico)**
sudo nano /etc/default/grub

# **Agregar línea**:
# GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
# **Cambiar a**:
# GRUB_CMDLINE_LINUX_DEFAULT="quiet splash systemd.unit=multi-user.target"

# **Actualizar GRUB**
sudo update-grub
```

Listing 27.12 BASH

```
# **Ver servicios en ejecución**
sudo systemctl list-units --type=service --state=running

# **Deshabilitar servicio innecesario (ej: Bluetooth)**
sudo systemctl disable bluetooth
sudo systemctl stop bluetooth

# **Servicios que generalmente NO se necesitan en servidor**:
# - bluetooth
# - cups (impresoras)
# - avahi-daemon (mDNS)
# - isc-dhcp-server (si no lo usas)
```

- Bloquear todo lo demás

Solución:

27.5.3 Ejercicio 3: Crear script de backup antes de actualizar

Crea script que:

1. Hace backup de configuración
2. Ejecuta **apt update && apt upgrade**
3. Notifica resultado por log
4. Se ejecuta cada domingo a las 2 AM

Solución: Ver Anexo A (Bash Scripting) para crear script + crontab

Listing 27.13 BASH

```
# **Proteger archivos de configuración sensibles**
sudo chmod 600 /etc/ssh/sshd_config

# **Asegurar log de auditoría**
sudo chmod 640 /var/log/auth.log

# **Proteger crontab de usuario**
sudo chmod 640 /etc/crontab

# **Archivos con permisos adecuados**
# /etc/passwd:     644 (legible por todos)
# /etc/shadow:     640 (solo root)
# /etc/sudoers:    440 (solo root, con visudo)
# /root/.ssh:      700 (solo root)
# ~/.ssh:          700 (solo usuario)
# ~/.ssh/id_rsa:   600 (solo usuario)
```

27.6 8. Quiz de Verificación

1. **¿Cuál es la diferencia entre** apt upgrade** y **apt full-upgrade**?**

   - A) No hay diferencia
   - B) upgrade es más rápido
   - C) full-upgrade puede remover paquetes; upgrade no
   - D) Respuesta C es correcta

2. **¿Qué hace** sudo ufw allow 22/tcp** antes de **sudo ufw enable**?**

   - A) Permite acceso SSH para que no te bloquees
   - B) Es necesario antes de habilitar firewall
   - C) Protege SSH de ataques
   - D) Respuestas A y B son correctas

3. **¿Cuál es el comando para ver reglas de UFW numeradas?**

   - A) sudo ufw list
   - B) sudo ufw show
   - C) sudo ufw status numbered
   - D) sudo ufw rules

4. **¿Qué es unattended-upgrades?**

   - A) Actualización manual del sistema
   - B) Herramienta para actualizar automáticamente
   - C) Comando para actualizar paquetes específicos
   - D) Script para limpiar paquetes

Listing 27.14 BASH

```
# **Ver estado de AppArmor**
sudo systemctl status apparmor

# **Ver perfiles cargados**
sudo aa-status

# **Habilitar AppArmor**
sudo systemctl enable apparmor

# **NOTA**: AppArmor es más simple que SELinux
# Usa perfiles predefinidos para servicios comunes
```

5. **¿Cuál es la regla de seguridad al usar UFW?**
   - A) Permitir SSH al último
   - B) Permitir SSH al primero
   - C) No importa el orden
   - D) SSH no necesita regla

27.7 9. Resumen y Siguiente Paso

27.7.1 Lo Aprendido

Actualizar paquetes de forma segura (**apt update**, **upgrade**) Configurar actualizaciones automáticas Instalar y usar firewall UFW Crear reglas de firewall apropiadas Aplicar hardening inicial Monitorear cambios de seguridad

27.7.2 Siguiente: Unidad 2.5

El próximo (y último) tema de Unidad 2 cubre:

- Comandos básicos de terminal
- Navegación del sistema de archivos
- Primeros pasos prácticos

Listing 27.15 BASH

```
# **Instalar Fail2Ban (bloquea IPs después de N intentos fallidos)**
sudo apt install fail2ban                                                    ①

# **Habilitar**
sudo systemctl enable fail2ban                                               ②
sudo systemctl start fail2ban

# **Ver estado**
sudo systemctl status fail2ban                                               ③

# **Ver intentos fallidos en SSH**
sudo fail2ban-client status sshd                                             ④

# **Salida**:
# Status for the jail sshd:
# |- Filter set to: sshd
# |- Currently failed: 0
# |- Currently banned: 2
# `- Total banned: 5
```

27.8 Referencias

(Canonical 2024b) (Community 2024) (F. Project 2024) (D. Project 2024c) (Canonical 2024a)

27.9 Quiz: Actualización y Seguridad

Listing 27.16 BASH

```
# **Conectar al servidor**
ssh diego@192.168.1.100

# **Actualizar lista de paquetes**
sudo apt update

# **Upgrade seguro**
sudo apt upgrade

# **Responder 'S' cuando pregunte**
# ¿Deseas continuar? [S/n] S

# **Limpiar paquetes no usados**
sudo apt autoremove
```

Listing 27.17 BASH

```
# **Instalar fail2ban**
sudo apt install fail2ban

# **Instalar UFW (probablemente ya viene)**
sudo apt install ufw

# **Habilitar UFW**
sudo ufw enable

# **Responder 'y' si pregunta si quieres continuar**
```

Listing 27.18 BASH

```
# **Permitir SSH (¡PRIMERO ESTO!)**
sudo ufw allow 22/tcp

# **Permitir HTTP**
sudo ufw allow 80/tcp

# **Permitir HTTPS**
sudo ufw allow 443/tcp

# **Verificar reglas**
sudo ufw status

# **Salida esperada**:
# Status: active
#
# To                         Action      From
# --                         ------      ----
# 22/tcp                     ALLOW       Anywhere
# 80/tcp                     ALLOW       Anywhere
# 443/tcp                    ALLOW       Anywhere
```

Listing 27.19 BASH

```
# **Instalar unattended-upgrades**
sudo apt install unattended-upgrades

# **Habilitar**
sudo systemctl enable unattended-upgrades
sudo systemctl start unattended-upgrades

# **Verificar estado**
sudo systemctl status unattended-upgrades
```

Listing 27.20 BASH

```
# **Ver estado de firewall**
sudo ufw status verbose

# **Ver servicios en escucha**
sudo ss -tulpn

# **Ver logs de fail2ban**
sudo fail2ban-client status

# **Ver logs de auditoría SSH**
sudo tail -20 /var/log/auth.log
```

Listing 27.21 BASH

```
# **Ver qué se actualizó**
cat /var/log/apt/history.log

# **Ver logs de unattended-upgrades**
sudo cat /var/log/unattended-upgrades/unattended-upgrades.log
```

Listing 27.22 BASH

```
# **Ver cambios en archivo crítico**
sudo apt install aide

# **Inicializar base de datos AIDE**
sudo aideinit

# **Luego, verificar cambios**
sudo aide --check
```

Listing 27.23 BASH

```
Unattended-Upgrade::Origins-Pattern {
    "origin=*,archive=*-security";
};
Unattended-Upgrade::Automatic-Reboot "true";
Unattended-Upgrade::Automatic-Reboot-Time "03:00";
Unattended-Upgrade::Mail "admin@abacom.com";
```

Listing 27.24 BASH

```
sudo ufw default deny incoming
sudo ufw default allow outgoing
sudo ufw allow from 192.168.1.50 to any port 22
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
sudo ufw allow from 192.168.1.0/24 to any port 3306
```

28 Primeros Pasos en Terminal

Listing 28.1 `QUARTO-TITLE-BLOCK`

Comandos Esenciales para Navegación y Gestión de Archivos

28.1 Objetivos de Aprendizaje

Después de completar este tema, serás capaz de:

- Navegar eficientemente el sistema de archivos
- Crear, copiar, mover y eliminar archivos/carpetas
- Ver y editar contenido de archivos
- Entender la estructura de directorios en Linux
- Gestionar permisos desde terminal
- Usar comandos esenciales diariamente

28.2 Por Qué Este Tema Es Critical

La terminal es el hogar del administrador Linux. No hay GUI (interfaz gráfica) en servidores. Estos comandos los usarás **100 veces por día**.

Tiempo estimado: 90-120 minutos de lectura + laboratorio práctico

28.3 Ejemplos Prácticos Multi-SO

28.3.1 Ejemplo 1: Abrir Terminal y Verificar SO

28.3.1.1 Linux (Ubuntu - Terminal nativa)

① **pwd** muestra el directorio actual donde abriste la terminal

28.3.1.2 macOS (Terminal / iTerm2)

Listing 28.2 BASH

```
# Terminal se abre automáticamente en home del usuario
# O abre presionando Ctrl+Alt+T en escritorio Ubuntu

$ pwd                                                                    ①
/home/diego

$ uname -a
Linux abacom-dev 5.15.0-89-generic #1234-Ubuntu SMP x86_64 GNU/Linux

$ whoami
diego

# Ver estructura de directorios
$ ls -la ~
drwxr-xr-x  diego  diego  /home/diego
drwxr-xr-x  diego  diego  .cache
drwxr-xr-x  diego  diego  .config
drwxr-xr-x  diego  diego  Documents
```

28.3.1.3 Windows (PowerShell / Terminal)

Comparación rápida: | Aspecto | Linux | macOS | Windows | |———|——-|——-|———| | Terminal | bash/zsh | bash/zsh (2021+) | PowerShell | | Comando pwd | `pwd` | `pwd` | `Get-Location` | | Home | `/home/usuario` | `/Users/usuario` | `C:\Users\usuario` | | Separador rutas | `/` | `/` | `\` | | Prompt | `$` | `$` | `PS>` |

28.3.2 Ejemplo 2: Navegar el Sistema de Archivos

28.3.2.1 Linux (Estructura Unix estándar)

28.3.2.2 macOS (Estructura Unix con extensiones Apple)

28.3.2.3 Windows (Estructura de directorios Windows)

Comparación de rutas comunes: | Tarea | Linux | macOS | Windows | |———|——-|——-|———| | Ver logs | `/var/log` | `/var/log` o `~/Library/Logs` | `%APPDATA%\Logs` | | Home | `/home/usuario` | `/Users/usuario` | `C:\Users\usuario` | | Temp | `/tmp` | `/tmp` o `/var/tmp` | `C:\Temp` o `%TEMP%` | | Apps | `/opt` o `/usr/bin` | `/Applications` | `C:\Program Files` |

Listing 28.3 BASH

```
# Terminal: Abre presionando Cmd+Space y escribes "Terminal"
# O iTerm2: brew install iterm2

$ pwd
/Users/diego

$ uname -a
Darwin MacBook-Air.local 23.2.0 Darwin Kernel Version 23.2.0 x86_64

$ whoami
diego

# Estructura de directorios (diferente a Linux):
$ ls -la ~
drwxr-xr-x  diego  staff  /Users/diego
drwxr-xr-x  diego  staff  .cache
drwxr-xr-x  diego  staff  .config
drwxr-xr-x  diego  staff  Documents
drwxr-xr-x  diego  staff  Downloads

# Nota: macOS usa /Users en lugar de /home
```

28.3.3 Ejemplo 3: Caso Real Abacom - Scripts Multi-SO

28.3.3.1 Linux (Bash script)

28.3.3.2 macOS (Bash script - casi igual a Linux)

28.3.3.3 Windows (PowerShell script)

Conclusión: Portabilidad de scripts

- **Linux macOS**: Scripts Bash son 90% compatibles
- **Linux Windows**: Requieren reescritura (comandos diferentes)
- **Solución**: Usar Python, Node.js, Go (multiplataforma)

28.4 1. Estructura de Directorios en Linux

28.4.1 1.1 Visualizar el Sistema de Archivos

```
/
  bin/              → Comandos binarios esenciales (/usr/bin para extra)
  sbin/             → Comandos para root (/usr/sbin para extra)
  etc/              → Archivos de configuración
  home/             → Directorios de usuarios
     diego/
     carlos/
     maria/
  root/             → Home del usuario root
  var/              → Datos variables (logs, caché)
     log/          → Logs del sistema
     www/          → Sitios web
     cache/
  tmp/              → Archivos temporales (se limpian al reboot)
  opt/              → Software opcional
  srv/              → Datos de servicios
  lib/              → Librerías del sistema
  usr/              → Programas de usuario
     bin/
     sbin/
     local/        → Software compilado localmente
  boot/             → Archivos de arranque
```

Ruta absoluta vs relativa:

28.5 2. Comandos Esenciales de Navegación

28.5.1 2.1 pwd - Mostrar Directorio Actual

28.5.2 2.2 cd - Cambiar Directorio

28.5.3 2.3 ls - Listar Archivos

Básico:

Detalles completos (formato largo):

Tamaño legible:

Listar recursivamente:

28.5.4 2.4 tree - Ver Estructura en Árbol

28.6 3. Crear y Eliminar Directorios

28.6.1 3.1 mkdir - Crear Carpetas

28.6.2 3.2 rmdir - Eliminar Carpetas Vacías

28.7 4. Copiar, Mover, Renombrar Archivos

28.7.1 4.1 cp - Copiar Archivos

28.7.2 4.2 mv - Mover y Renombrar

Diferencia mv vs cp:

```
cp archivo.txt copia.txt
# archivo.txt sigue existiendo
# copia.txt es nueva

mv archivo.txt renombrado.txt
# archivo.txt desaparece
# renombrado.txt existe (mismos datos)
```

28.8 5. Ver Contenido de Archivos

28.8.1 5.1 cat - Mostrar Contenido Completo

28.8.2 5.2 less - Ver Archivo Grande Interactivamente

28.8.3 5.3 head y tail - Ver Inicio y Final

28.9 6. Crear y Editar Archivos

28.9.1 6.1 touch - Crear Archivo Vacío

28.9.2 6.2 echo - Escribir Texto

28.9.3 6.3 Editores de Texto

Nano (más fácil):

Vim/Vi (poderoso pero complejo):

Ver Anexo B: Editores Vi y Nvim para detalles completos.

28.10 7. Buscar Archivos y Contenido

28.10.1 7.1 find - Buscar Archivos por Nombre

Ver Anexo E: Búsqueda y Procesamiento para detalles avanzados.

28.10.2 7.2 grep - Buscar Texto en Archivos

28.11 8. Permisos desde Terminal

28.11.1 8.1 Ver Permisos Detallados

28.11.2 8.2 Cambiar Permisos - chmod

28.11.3 8.3 Cambiar Propietario - chown

28.12 9. Información Útil

28.12.1 9.1 whoami, id, groups

28.12.2 9.2 du - Uso de Disco

28.12.3 9.3 df - Espacio Disponible en Disco

28.13 10. Laboratorio Práctico

28.13.1 10.1 Crear Estructura de Proyecto

Crea estructura para proyecto de desarrollo:

28.13.2 10.2 Crear Archivos de Configuración

28.13.3 10.3 Buscar y Reemplazar Contenido

28.14 11. Ejercicios Prácticos

28.14.1 Ejercicio 1: Navegación

Completa estas tareas:

1. Navega a **/var/log**
2. Lista archivos con detalles
3. Ve al home con ~
4. Muestra dónde estás

Solución:

28.14.2 Ejercicio 2: Crear estructura

Crea:

```
~/documentos/
  trabajo/
      reportes/
      propuestas/
  personal/
       finanzas/
       salud/
```

Solución:

28.14.3 Ejercicio 3: Buscar archivos

1. Encuentra todos los .txt en tu home
2. Busca líneas con "error" en **/var/log/syslog**
3. Cuenta cuántas veces aparece "error"

Solución:

28.15 12. Quiz de Verificación

1. **¿Cuál es la diferencia entre** cd** y **pwd**?**
   - A) Ninguna
   - B) cd cambia directorio; pwd muestra donde estás
   - C) pwd es más rápido
   - D) cd solo funciona en root
2. **¿Qué hace** ls -lh**?**
   - A) Lista archivos
   - B) Lista con detalles y tamaño legible
   - C) Lista y busca
   - D) Lista ocultos
3. **¿Cuál es la diferencia entre** >** y »?**
   - A) No hay diferencia
   - B) sobrescribe; » agrega
   - C) es más rápido

- D) solo para texto

4. **¿Qué comando busca archivos?**

   - A) grep (busca contenido)
   - B) find (busca archivos)
   - C) locate
   - D) search

5. **¿Qué hace** chmod 755 script.sh**?**

   - A) Hacer script ejecutable con rwxr-xr-x
   - B) Proteger archivo
   - C) Cambiar propietario
   - D) Crear backup

28.16 13. Resumen

28.16.1 Lo Aprendido

Navegar sistema de archivos (**cd**, **pwd**, **ls**) Crear y eliminar carpetas (**mkdir**, **rmdir**) Copiar, mover, renombrar (**cp**, **mv**) Ver contenido (**cat**, **less**, **head**, **tail**) Crear y editar archivos (**echo**, **touch**, **nano**) Buscar archivos y contenido (**find**, **grep**) Entender y cambiar permisos (**ls -l**, **chmod**, **chown**)

28.16.2 Siguiente

¡Felicidades! Completaste **Unidad 2: Instalación y Configuración**

Próximas unidades:

- **Unidad 3**: Comandos Avanzados
- **Unidad 4**: Usuarios y Permisos (profundidad)
- **Unidad 5**: Procesos y Servicios

28.17 Referencias

(L. Foundation 2019) (F. S. Foundation 2024b) (F. S. Foundation 2024a) (Linux.com 2024) (D. Project 2024a)

28.18 Quiz: Primeros Pasos en Terminal

Listing 28.4 POWERSHELL

```
# Abrir PowerShell:
# - Presionar Win+X y seleccionar "Windows PowerShell"
# - O: Win+R, escribir "powershell"

PS> # Nota: Prompt es PS> en lugar de $

# Ver directorio actual
PS> Get-Location
Path
----
C:\Users\diego

# Equivalente a pwd:
PS> pwd  # También funciona (alias de Get-Location)
C:\Users\diego

# Verificar SO
PS> $PSVersionTable.OS
Microsoft Windows 11 Pro 22H2

PS> whoami
DESKTOP-LAPTOP\diego

# Ver estructura de directorios
PS> dir ~  # Equivalente a ls ~

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----          1/29/2024  10:00 AM                AppData
d-----          1/29/2024  10:00 AM                Desktop
d-----          1/29/2024  10:00 AM                Documents
d-----          1/29/2024  10:00 AM                Downloads

# Nota: Windows usa C:\Users en lugar de /home
```

Listing 28.5 BASH

```
# Estructura en Linux:
$ cd /
$ ls -1
bin/    → Comandos esenciales
boot/   → Archivos de arranque
dev/    → Dispositivos
etc/    → Configuraciones
home/   → Carpetas de usuarios
lib/    → Librerías del sistema
opt/    → Software opcional
root/   → Home del root
srv/    → Datos de servicios
tmp/    → Archivos temporales
usr/    → Programas de usuario
var/    → Datos variables (logs, etc)

# Navegar
$ cd /var/log
$ pwd
/var/log

$ ls -lh
-rw-r--r--  1 syslog syslog  250K Jan 29 10:30 auth.log
-rw-r--r--  1 syslog syslog  500K Jan 29 10:31 syslog
-rw-r--r--  1 root   root    1.2M Jan 29 10:30 kernel.log
```

Listing 28.6 BASH

```
# Estructura en macOS (también Unix, pero con cambios):
$ cd /
$ ls -1
Applications/     → Apps instaladas
Library/          → Librerías del sistema (¡no es /lib!)
System/           → Sistema macOS
Users/            → Carpetas de usuarios (¡no es /home!)
Volumes/          → Discos conectados
bin/              → Comandos esenciales (igual que Linux)
etc/              → Configuraciones
tmp/              → Archivos temporales
usr/              → Programas de usuario

# Navegar a logs (diferente ubicación que Linux)
$ cd /var/log
$ pwd
/var/log

$ ls -lh
-rw-r--r--  1 _syslogd staff  200K Jan 29 10:30 system.log
-rw-r--r--  1 root     wheel  150K Jan 29 10:31 kernel.log

# Carpeta especial de macOS:
$ cd ~/Library/Application\ Support
# Muchas apps guardan config aquí
```

Listing 28.7 POWERSHELL

```
# Estructura en Windows (TOTALMENTE diferente a Unix):
PS> cd C:\
PS> dir | Select-Object Name

Name
----
ProgramFiles          → Apps instaladas (64-bit)
ProgramFiles (x86)    → Apps instaladas (32-bit)
Program Data          → Datos de aplicaciones
Windows               → Sistema operativo
System32              → Herramientas del sistema
Users                 → Carpetas de usuarios
Temp                  → Archivos temporales

# Navegar a carpeta de usuario
PS> cd $env:USERPROFILE
PS> Get-Location
C:\Users\diego

# Ver estructura
PS> dir | Select-Object Name
AppData         → Datos de apps (oculto)
Desktop         → Escritorio
Documents       → Documentos
Downloads       → Descargas
Pictures        → Imágenes

# Crear carpeta de proyecto
PS> mkdir projects
PS> cd projects
PS> pwd
C:\Users\diego\projects
```

Listing 28.8 BASH

```
#!/bin/bash
# Script para verificar salud del servidor en Abacom

echo "=== VERIFICACIÓN DE SERVIDOR LINUX ==="
echo "Hostname: $(hostname)"
echo "Kernel: $(uname -r)"
echo "Usuarios conectados: $(who | wc -l)"
echo "Uso de CPU: $(top -bn1 | grep "Cpu(s)" | awk '{print $2}')"
echo "Uso de RAM: $(free -h | grep Mem | awk '{print $3 "/" $2}')"
echo "Espacio disco: $(df -h / | tail -1 | awk '{print $5}')"

# Ejecutar
$ chmod +x check_server.sh
$ ./check_server.sh
=== VERIFICACIÓN DE SERVIDOR LINUX ===
Hostname: abacom-server-01
Kernel: 5.15.0-89-generic
Usuarios conectados: 2
Uso de CPU: 15%
Uso de RAM: 4.2G / 16G
Espacio disco: 35%
```

Listing 28.9 BASH

```
#!/bin/bash
# Script casi idéntico para macOS

echo "=== VERIFICACIÓN DE SERVIDOR macOS ==="
echo "Hostname: $(hostname)"
echo "Kernel: $(uname -r)"
echo "Usuarios conectados: $(who | wc -l)"

# Nota: Comandos como top son diferentes en macOS
echo "Uso de CPU: $(top -l 1 | grep "CPU usage" | head -1)"
echo "Uso de RAM: $(vm_stat | grep 'Pages free' | awk '{print $3}')"
echo "Espacio disco: $(df -h / | tail -1 | awk '{print $5}')"

$ chmod +x check_server.sh
$ ./check_server.sh
=== VERIFICACIÓN DE SERVIDOR macOS ===
Hostname: MacBook-Air.local
Kernel: 23.2.0
Usuarios conectados: 1
Uso de CPU: CPU usage: 25%
Uso de RAM: 2024384
Espacio disco: 45%
```

Listing 28.10 POWERSHELL

```powershell
# check_server.ps1
# Script equivalente para Windows

Write-Host "=== VERIFICACIÓN DE SERVIDOR WINDOWS ===" -ForegroundColor Cyan

$ComputerName = $env:COMPUTERNAME
$OS = (Get-WmiObject -Class Win32_OperatingSystem).Caption
$Uptime = (Get-Date) - (Get-CimInstance Win32_OperatingSystem).LastBootUpTime
$CPUUsage = (Get-WmiObject win32_processor).LoadPercentage
$RAM = Get-WmiObject Win32_ComputerSystem
$RAMUsed = $RAM.TotalPhysicalMemory - (Get-WmiObject Win32_OperatingSystem).FreePhysicalM
$DiskUsage = (Get-PSDrive C).Used / (Get-PSDrive C).Total * 100

Write-Host "Nombre: $ComputerName"
Write-Host "SO: $OS"
Write-Host "Uptime: $($Uptime.Days) días, $($Uptime.Hours) horas"
Write-Host "Uso de CPU: $CPUUsage%"
Write-Host "Uso de RAM: $([math]::Round($RAMUsed/1GB)) GB"
Write-Host "Uso de disco: $([math]::Round($DiskUsage))%"

# Ejecutar
PS> .\check_server.ps1
=== VERIFICACIÓN DE SERVIDOR WINDOWS ===
Nombre: ABACOM-PC
SO: Microsoft Windows Server 2022
Uptime: 30 días, 5 horas
Uso de CPU: 18%
Uso de RAM: 8 GB
Uso de disco: 65%
```

Listing 28.11 BASH

```bash
# **Ruta absoluta: comienza con /**
/home/diego/documentos/proyecto.txt

# **Ruta relativa: desde carpeta actual**
documentos/proyecto.txt

# **Especiales**:
~        → Home del usuario actual
.        → Carpeta actual
..       → Carpeta padre

-        → Último directorio visitado
```

Listing 28.12 BASH

```
# **Mostrar ruta completa donde estás**
pwd

# **Salida**:
# /home/diego/documentos

# **¿Dónde estoy?**
# Muy útil después de navegar varias carpetas
```

Listing 28.13 BASH

```
# **Ir a home del usuario**
cd
cd ~
cd /home/diego

# **Ir a carpeta específica**
cd /var/log

# **Ir a carpeta anterior**
cd -
# Regresa al último directorio visitado

# **Subir un nivel (a carpeta padre)**
cd ..

# **Ir a raíz del sistema**
cd /

# **Usar rutas con espacios (entre comillas)**
cd "/home/diego/Mi Documentos"

# **Ver dónde estás después de cambiar**
pwd
```

Listing 28.14 BASH

```
# **Listar archivos en carpeta actual**
ls

# **Salida**:
# archivo.txt  carpeta/  foto.jpg

# **Listar carpeta específica**
ls /home/diego/

# **Listar todo incluyendo ocultos (comienzan con .)**
ls -a

# **Salida**:
# .  ..  .bashrc  .ssh/  archivo.txt  .hidden_folder/
```

Listing 28.15 BASH

```
# **Ver permisos, propietario, tamaño, fecha**
ls -l

# **Salida**:
# drwxr-xr-x  2 diego diego  4096 Jan 29 10:30 carpeta/
# -rw-r--r--  1 diego diego  1024 Jan 29 09:15 archivo.txt
# lrwxrwxrwx  1 diego diego    10 Jan 29 08:00 enlace -> archivo.txt

# **Desglose de columnas**:
# drwxr-xr-x  → Permisos
# 2           → Número de enlaces
# diego diego → Usuario:Grupo
# 4096        → Tamaño (en bytes)
# Jan 29      → Fecha
# 10:30       → Hora
# carpeta/    → Nombre (/ = carpeta)
```

Listing 28.16 BASH

```
# **Ver tamaño en KB, MB, GB**
ls -lh

# **Salida**:
# -rw-r--r--  1 diego diego  1.2K Jan 29 archivo.txt
# -rw-r--r--  1 diego diego  2.5M Jan 29 video.mp4
# -rw-r--r--  1 diego diego  256M Jan 29 imagen.iso

# **Opciones útiles**:
# -S    → Ordenar por tamaño (más grande primero)
# -t    → Ordenar por fecha (más reciente primero)
# -r    → Orden inverso
```

Listing 28.17 BASH

```
# **Ver todas las carpetas y subcarpetas**
ls -R

# **Útil para ver estructura**
# .
# ./carpeta1
# ./carpeta1/subcarpeta
# ./carpeta2
```

Listing 28.18 BASH

```
# **Si no está instalado**
sudo apt install tree

# **Ver estructura como árbol**
tree

# **Salida**:
# .
#    carpeta1/
#       archivo1.txt
#       archivo2.txt
#    carpeta2/
#    archivo3.txt

# **Limitar profundidad**
tree -L 2
# -L 2 → Ver máximo 2 niveles

# **Ver solo directorios**
tree -d
```

Listing 28.19 BASH

```
# **Crear una carpeta**
mkdir miprojecto

# **Crear múltiples carpetas**
mkdir carpeta1 carpeta2 carpeta3

# **Crear carpeta anidada (necesita -p)**
mkdir -p /home/diego/proyectos/2024/enero
# -p → Crea las carpetas padre si no existen

# **Crear con permisos específicos**
mkdir -m 700 carpeta_privada
# Accesible solo por dueño
```

Listing 28.20 BASH

```
# **Eliminar carpeta VACÍA**
rmdir carpeta_vacia

# **Eliminar múltiples carpetas vacías**
rmdir carpeta1 carpeta2 carpeta3

# **Eliminar estructura anidada (si todas están vacías)**
rmdir -p /home/diego/proyectos/2024/enero
# Elimina enero, 2024, proyectos (si quedan vacías)
```

Listing 28.21 BASH

```
# **Copiar un archivo**
cp archivo_original.txt archivo_copia.txt

# **Copiar a otra carpeta (mantener nombre)**
cp archivo.txt /home/diego/backup/

# **Copiar a otra carpeta con nuevo nombre**
cp archivo.txt /home/diego/backup/archivo_respaldo.txt

# **Copiar carpeta y contenido (recursivo)**
cp -r carpeta_original/ carpeta_copia/

# **Opciones útiles**:
# -i → Preguntar antes de sobrescribir
# -v → Mostrar qué copia (verbose)
# -p → Preservar permisos y fecha

# **Ejemplo completo**
cp -rpv /home/diego/proyecto /backup/proyecto_respaldo
# Copia carpeta, preserva permisos, muestra progreso
```

Listing 28.22 BASH

```
# **Renombrar archivo**
mv archivo_viejo.txt archivo_nuevo.txt

# **Mover archivo a otra carpeta**
mv archivo.txt /home/diego/documentos/

# **Mover y renombrar al mismo tiempo**
mv archivo.txt /home/diego/documentos/archivo_movido.txt

# **Mover carpeta entera**
mv carpeta_origen/ /nueva/ubicacion/

# **Opciones**:
# -i → Preguntar antes de sobrescribir
# -v → Mostrar qué mueve

# **Ejemplo: Mover varias cosas**
mv archivo1.txt archivo2.txt carpeta/ documentos/
# Mueve tres cosas a documentos/
```

Listing 28.23 BASH

```
# **Ver archivo pequeño**
cat archivo.txt

# **Salida**:
# Contenido del archivo
# en múltiples líneas

# **Ver archivo y enumerar líneas**
cat -n archivo.txt

# **Salida**:
#      1  Primera línea
#      2  Segunda línea
#      3  Tercera línea

# **Concatenar múltiples archivos**
cat archivo1.txt archivo2.txt

# **Ver contenido con saltos visibles**
cat -A archivo.txt
# Muestra saltos de línea como $
```

Listing 28.24 BASH

```
# **Ver archivo página por página (mejor que cat)**
less archivo_grande.log

# **Controles**:
# Espacio o Page Down → Siguiente página
# b o Page Up → Página anterior
# G → Ir al final
# g → Ir al inicio
# /patrón → Buscar texto
# n → Siguiente resultado
# N → Resultado anterior
# q → Salir

# **Ejemplo: Ver log del sistema**
less /var/log/syslog
# Muy útil para archivos enormes
```

Listing 28.25 BASH

```
# **Ver primeras 10 líneas**
head archivo.txt

# **Ver primeras 20 líneas**
head -20 archivo.txt

# **Ver últimas 10 líneas (muy común para logs)**
tail archivo.txt

# **Ver últimas 20 líneas**
tail -20 archivo.txt

# **Ver últimas líneas conforme se agregan (monitoreo)**
tail -f /var/log/syslog
# -f → follow (sigue actualizaciones)
# Presiona Ctrl+C para salir

# **Ver linea 100 en adelante**
tail -n +100 archivo.txt
```

Listing 28.26 BASH

```
# **Crear archivo vacío**
touch archivo_nuevo.txt

# **Crear múltiples archivos**
touch archivo1.txt archivo2.txt archivo3.txt

# **Actualizar fecha de acceso de archivo existente**
touch archivo_existente.txt
# Cambia la fecha de modificación a ahora
```

Listing 28.27 BASH

```
# **Mostrar texto en terminal**
echo "Hola Mundo"

# **Salida**:
# Hola Mundo

# **Crear archivo con contenido usando redirección**
echo "Primera línea" > archivo.txt

# **Agregar línea a archivo existente**
echo "Segunda línea" >> archivo.txt

# **Ver contenido**
cat archivo.txt
# Primera línea
# Segunda línea

# **Usar variables**
echo "Mi usuario es: $USER"
# Mi usuario es: diego

# **Múltiples líneas**
cat > archivo.txt << EOF
Línea 1
Línea 2
Línea 3
EOF
```

Listing 28.28 BASH

```
# **Abrir o crear archivo**
nano archivo.txt

# **Controles**:
# Escribir normalmente
# Ctrl+O → Guardar
# Ctrl+X → Salir
# Ctrl+W → Buscar
# Ctrl+K → Cortar línea
# Ctrl+U → Pegar
```

Listing 28.29 BASH

```
# **Abrir archivo**
vim archivo.txt

# **Modos**:
# i → Insert (escribir)
# Esc → Normal (comandos)
# En normal: :wq (guardar y salir)
```

Listing 28.30 BASH

```
# **Buscar archivo por nombre**
find . -name "archivo.txt"

# **Buscar en ubicación específica**
find /home -name "*.txt"

# **Buscar ignorando mayúsculas**
find . -iname "ARCHIVO.txt"

# **Buscar archivos (no carpetas)**
find . -type f -name "*.txt"

# **Buscar solo carpetas**
find . -type d -name "carpeta*"

# **Buscar archivos modificados en últimos 7 días**
find . -mtime -7

# **Buscar archivos mayores a 100MB**
find . -size +100M

# **Buscar archivos vacíos**
find . -empty

# **Ejecutar comando en resultados**
find . -name "*.txt" -exec rm {} \;
# Borra todos los .txt encontrados
```

Listing 28.31 BASH

```
# **Buscar texto en archivo**
grep "error" archivo.log

# **Salida**: Solo líneas que contienen "error"

# **Buscar ignorando mayúsculas**
grep -i "ERROR" archivo.log

# **Buscar en múltiples archivos**
grep "error" /var/log/*.log

# **Buscar recursivamente en carpeta**
grep -r "función_importante" /home/diego/proyecto/

# **Contar coincidencias**
grep -c "error" archivo.log

# **Ver líneas alrededor del resultado**
grep -B2 -A2 "error" archivo.log
# -B2 → 2 líneas antes
# -A2 → 2 líneas después
```

Listing 28.32 BASH

```
# **Ver permisos de archivos**
ls -l

# **Salida**:
# -rw-r--r-- 1 diego diego 1024 Jan 29 archivo.txt
#  ^^^^^^^^
#  drwxr-xr-x
#    archivo
#       permisos

# **Decodificar: rwxrwxrwx**
# Usuario(rwx) Grupo(r-x) Otros(r--)
```

Listing 28.33 BASH

```
# **Hacer archivo ejecutable**
chmod +x script.sh

# **Remover permisos de escritura**
chmod -w archivo.txt

# **Establecer permisos exactos**
chmod 755 script.sh
# 755 = rwxr-xr-x (ejecutable para todos, editable solo por dueño)

chmod 644 archivo.txt
# 644 = rw-r--r-- (editable solo por dueño)

chmod 600 archivo_privado
# 600 = rw------- (solo dueño puede acceder)
```

Listing 28.34 BASH

```
# **Cambiar propietario**
sudo chown diego archivo.txt

# **Cambiar propietario y grupo**
sudo chown diego:administradores archivo.txt

# **Cambiar recursivamente (carpeta y contenido)**
sudo chown -R diego:administradores /home/diego/proyecto/
```

Listing 28.35 BASH

```
# **¿Quién soy?**
whoami
# diego

# **Información detallada del usuario**
id
# uid=1000(diego) gid=1000(diego) groups=1000(diego),4(adm),24(cdrom)

# **Grupos del usuario**
groups
# diego adm cdrom sudo

# **Información de otro usuario**
id carlos
```

Listing 28.36 BASH

```
# **Tamaño de carpeta actual y subcarpetas**
du -sh *

# **Salida**:
# 2.5G carpeta1/
# 1.2G carpeta2/
# 512M archivo.iso

# **Ver solo carpetas (no archivos)**
du -sh */

# **Ver total de carpeta**
du -sh .
# 5.2G .
```

Listing 28.37 BASH

```
# **Ver particiones y espacio disponible**
df -h

# **Salida**:
# Filesystem      Size  Used Avail Use% Mounted on
# /dev/sda1        20G   15G    5G  75% /
# /dev/sda2        50G   40G   10G  80% /home

# **¿Está lleno el disco?**
# Use% > 90% → Problemas
```

Listing 28.38 BASH

```
# Crear carpeta principal
mkdir ~/proyectos

# Crear subcarpetas
mkdir -p ~/proyectos/webapp/{src,config,tests,docs}
mkdir -p ~/proyectos/webapp/src/{frontend,backend}

# Ver estructura
tree ~/proyectos/webapp
```

Listing 28.39 BASH

```
# Crear archivo README
cat > ~/proyectos/webapp/README.md << EOF
# Webapp Project
Descripción del proyecto
EOF

# Crear archivo de configuración
cat > ~/proyectos/webapp/config/settings.sh << EOF
#!/bin/bash
# Configuración de webapp
DB_HOST=localhost
DB_USER=admin
DB_PASS=secret123
EOF

# Hacer script ejecutable
chmod +x ~/proyectos/webapp/config/settings.sh

# Verificar
ls -la ~/proyectos/webapp/config/
```

Listing 28.40 BASH

```
# Ver archivos con palabra clave
grep -r "localhost" ~/proyectos/webapp/

# Encontrar scripts ejecutables
find ~/proyectos/webapp -type f -executable

# Ver tamaño total del proyecto
du -sh ~/proyectos/webapp/
```

Listing 28.41 BASH

```
cd /var/log
ls -lh
cd ~
pwd
```

Listing 28.42 BASH

```
mkdir -p ~/documentos/{trabajo/{reportes,propuestas},personal/{finanzas,salud}}
tree ~/documentos/
```

Listing 28.43 BASH

```
find ~ -type f -name "*.txt"
grep "error" /var/log/syslog
grep -c "error" /var/log/syslog
```

29 Unidad 2: Recursos

Listing 29.1 QUARTO-TITLE-BLOCK

30 Recursos De La Unidad 2

Recurso	Enlace
Presentacion (Reveal.js)	Unidad 2
Practica	Practica Unidad 2
Laboratorio	Lab 1: Instalacion Ubuntu Server

Part IV

Unidad 3: Comandos Básicos de Linux

31 Navegación de Directorios

Listing 31.1 QUARTO-TITLE-BLOCK

32 Navegación de Directorios

32.1 Introducción

La navegación eficiente por el sistema de archivos es una de las habilidades más fundamentales en Linux. Dominar comandos como `pwd`, `cd` y `ls` te permitirá moverte rápidamente por el sistema y entender su estructura jerárquica.

En este tema aprenderás:

- Comando `pwd` (Print Working Directory)
- Comando `cd` (Change Directory)
- Comando `ls` (List) con sus opciones principales
- Rutas absolutas vs relativas
- Atajos de navegación (., .., ~, -)
- Navegación eficiente en el sistema de archivos

32.2 El Comando `pwd` - Dónde Estoy Ahora

El comando `pwd` imprime tu ubicación actual en el árbol de directorios. Es especialmente útil cuando te pierdes o quieres saber exactamente dónde estás en el sistema.

Sintaxis básica:

Listing 32.1 BASH

```
pwd [opciones]
```

32.2.1 Ejemplos Prácticos Multi-SO

32.2.2 Ejemplo 1: Ver tu directorio actual

32.2.2.1 Linux

① Mostrará tu ruta absoluta actual

Listing 32.2 BASH

```
$ pwd
/home/usuario/documentos
                                                                    ①
```

Listing 32.3 BASH

```
$ pwd
/Users/usuario/Documents
                                                                    ①
```

32.2.2.2 macOS

① En macOS, el home usa "/Users" en lugar de "/home"

32.2.2.3 Windows

Listing 32.4 POWERSHELL

```
PS> Get-Location
C:\Users\usuario\Documents
                                                                    ①
```

① Equivalente de PowerShell para ver la ubicación actual

Aspecto	Linux	macOS	Windows
Comando	`pwd`	`pwd`	`Get-Location` o `cd`
Home dir	`/home/user`	`/Users/user`	`C:\Users\user`
Separador	`/`	`/`	`\`
Salida	Ruta absoluta	Ruta absoluta	Ruta absoluta

Cuándo usar: Usa `pwd` cuando necesites saber tu ubicación exacta o cuando estés escribiendo scripts que requieren rutas absolutas.

32.3 El Comando `cd` - Cambiando Directorios

El comando `cd` es el motor de la navegación. Te permite moverte de un directorio a otro en el árbol de archivos.

Sintaxis:

32.3.1 Ejemplos Prácticos Multi-SO

32.3.2 Ejemplo 1: Cambiar a un directorio específico (ruta absoluta)

32.3.2.1 Linux

Listing 32.5 BASH

```
cd [directorio]
```

Listing 32.6 BASH

```
$ pwd
/home/usuario
$ cd /etc
$ pwd
/etc
```

①

32.3.2.2 macOS

① Estructura de directorios similar a Linux estándar

32.3.2.3 Windows

① Windows usa letras de unidad (C:, D:) y barras invertidas

32.3.3 Ejemplo 2: Cambiar a un subdirectorio (ruta relativa)

32.3.3.1 Linux

① Usamos ruta relativa (sin comenzar con /)

32.3.3.2 macOS

① Mismo concepto que Linux

32.3.3.3 Windows

① PowerShell maneja rutas relativas igual

32.3.4 Ejemplo 3: Atajos de navegación útiles

32.3.4.1 Linux

① `.` es el directorio actual, `..` es el padre, `~` es el home, `-` es el anterior

Listing 32.7 BASH

```
$ pwd
/Users/usuario
$ cd /var/log
$ pwd
/var/log
```
①

Listing 32.8 POWERSHELL

```
PS> Get-Location
C:\Users\usuario
PS> cd C:\Windows\System32
PS> Get-Location
C:\Windows\System32
```
①

32.3.4.2 macOS

① Mismo comportamiento que Linux

32.3.4.3 Windows

① PowerShell soporta los mismos atajos que Bash

Atajo	Significado	Linux	macOS	Windows
`~`	Directorio home			
`..`	Directorio padre			
`.`	Directorio actual			
`-`	Directorio anterior			
`/`	Raíz del sistema			`C:\`

Cuándo usar: Domina estos atajos para navegar rápidamente. Los profesionales los usan constantemente.

32.4 El Comando `ls` - Listando Archivos

El comando `ls` (list) es uno de los más usados en la terminal. Te muestra el contenido de directorios.

Sintaxis:

32.4.1 Opciones principales:

- `-l` : Formato largo (permisos, propietario, tamaño, fecha)
- `-a` : Mostrar archivos ocultos (que comienzan con .)
- `-h` : Tamaños en formato legible (K, M, G)

Listing 32.9 BASH

```
$ pwd
/home/usuario
$ cd Documentos
$ pwd
/home/usuario/Documentos
```
①

Listing 32.10 BASH

```
$ pwd
/Users/usuario
$ cd Documents
$ pwd
/Users/usuario/Documents
```
①

32.4.3.2 macOS

① Directorio home con carpetas estándar de macOS

32.4.3.3 Windows

① PowerShell tiene formato más detallado por defecto

32.4.4 Ejemplo 2: Formato largo con detalles (ls -l)

32.4.4.1 Linux

① Permisos, propietario, tamaño, fecha y nombre

32.4.4.2 macOS

① Formato idéntico a Linux (grupo es “staff” en macOS)

32.4.4.3 Windows

① PowerShell muestra información similar en formato tabular

Listing 32.11 POWERSHELL

```
PS> Get-Location
C:\Users\usuario
PS> cd Documents
PS> Get-Location
C:\Users\usuario\Documents
```
①

Listing 32.12 BASH

```
# Ir al home
$ cd ~
$ pwd
/home/usuario

# Ir a directorio padre
$ cd ..
$ pwd
/home

# Ir al directorio anterior
$ cd -
$ pwd
/home/usuario

# Directorio actual (menos común)
$ cd .
$ pwd
/home/usuario
```
①

32.4.5 Ejemplo 3: Mostrar archivos ocultos y tamaños legibles (ls -la)

32.4.5.1 Linux

① `-a` muestra puntos (.), `.bashrc`, `.bash_profile` (archivos ocultos)
② `-h` muestra tamaños en K, M, G en lugar de bytes

32.4.5.2 macOS

① En macOS el shell default es `zsh`, no `bash`
② Muestra archivos de configuración específicos de zsh

Listing 32.13 BASH

```
# Ir al home
$ cd ~
$ pwd
/Users/usuario

# Ir a directorio padre
$ cd ..
$ pwd
/Users

# Ir al directorio anterior
$ cd -
$ pwd
/Users/usuario

# El directorio actual
$ cd .
```

①

32.4.5.3 Windows

① `-Force` en PowerShell es equivalente a `-a` en Bash
② Muestra archivos ocultos (los que Windows marca con atributo oculto)

32.4.6 Ejemplo 4: Listar recursivamente con tamaños (ls -lhR)

32.4.6.1 Linux

① `-R` recursivo muestra directorios y sus subdirectorios
② `-h` hace tamaños legibles

32.4.6.2 macOS

① Funciona igual que Linux

32.4.6.3 Windows

① `-Recurse` es el equivalente a `-R` en PowerShell
② PowerShell no tiene una forma tan concisa como `ls -lhR`

Listing 32.14 POWERSHELL

```powershell
# Ir al home (con -Path)
PS> cd ~
PS> Get-Location
C:\Users\usuario

# Ir a directorio padre
PS> cd ..
PS> Get-Location
C:\Users

# Directorio anterior
PS> cd -
PS> Get-Location
C:\Users\usuario
```

①

Listing 32.15 BASH

```bash
ls [opciones] [directorio]
```

Opción	Linux	macOS	Windows	Descripción
`-l`			Format-Table	Formato largo
`-a`			-Force	Mostrar ocultos
`-h`			Parcial	Tamaños legibles
`-R`			-Recurse	Recursivo
`-t`			Sort-Object	Ordenar por fecha
`-S`			Sort-Object	Ordenar por tamaño

Cuándo usar: `ls` es tu mejor amigo. Combina opciones para obtener exactamente la información que necesitas.

32.5 Rutas Absolutas vs Relativas

Entender la diferencia es crucial para la navegación eficiente.

32.5.1 Rutas Absolutas

- Comienzan desde la raíz del sistema

Listing 32.16 BASH

```
$ ls
Documentos  Descargas  Escritorio  Videos
```
①

Listing 32.17 BASH

```
$ ls
Applications  Desktop  Documents  Downloads  Library
```
①

- Comienzan con `/` en Linux/macOS
- Comienzan con `C:\` en Windows
- Funcionan desde cualquier ubicación

32.5.2 Rutas Relativas

- Son relativas a tu ubicación actual
- No comienzan con `/`
- Más cortas y rápidas de escribir
- Dependientes de dónde estés

32.5.3 Ejemplo comparativo:

32.5.3.1 Linux

32.5.3.2 macOS

32.5.3.3 Windows

32.6 Resumen y Mejores Prácticas

Comando	Propósito	Ejemplo
`pwd`	Mostrar ubicación actual	`pwd`
`cd directorio`	Cambiar a directorio	`cd ~/Documentos`
`cd ~`	Ir al home	`cd ~`

Comando	Propósito	Ejemplo
`cd ..`	Ir al padre	`cd ..`
`cd -`	Ir anterior	`cd -`
`ls`	Listar archivos	`ls`
`ls -l`	Formato largo	`ls -l`
`ls -la`	Incluir ocultos	`ls -la`
`ls -lh`	Tamaños legibles	`ls -lh`
`ls -R`	Recursivo	`ls -lR`

32.6.1 Mejores prácticas:

1. **Usa rutas relativas** cuando navegues localmente (más rápido)
2. **Usa rutas absolutas** en scripts (más confiable)
3. **Aprende los atajos** (~, .., -, .) - los usarás constantemente
4. **Combine opciones** de `ls` para obtener exactamente lo que necesitas
5. **Usa `cd -`** para volver rápidamente al directorio anterior

---

32.7 Ejercicios Prácticos

1. Navega a tu directorio home usando `cd ~` y verifica con `pwd`
2. Crea esta estructura: `Documentos/Cursos/Linux`
3. Lista el contenido con `ls -lah` y analiza los permisos
4. Navega hacia atrás usando `cd -` entre dos directorios
5. Usa rutas relativas para navegar: `cd ./Documentos/Cursos/Linux`

32.8 Referencias

Para más información sobre navegación y gestión de archivos en sistemas Unix-like, consulta:

- (IEEE 2018) - POSIX Standard
- (F. S. Foundation 2023a) - GNU Bash Manual

Listing 32.18 POWERSHELL

```
PS> Get-ChildItem
# o
PS> ls

    Directory: C:\Users\usuario

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         29/1/2024     10:30                Desktop
d-----         29/1/2024     10:30                Documents
d-----         29/1/2024     10:30                Downloads
```

Listing 32.19 BASH

```
$ ls -l
total 32
drwxr-xr-x  5 usuario grupo  4096 Jan 29 10:30 Documentos
-rw-r--r--  1 usuario grupo  2048 Jan 28 15:45 archivo.txt
-rwxr-xr-x  1 usuario grupo   512 Jan 27 09:00 script.sh
```

Listing 32.20 BASH

```
$ ls -l
total 32
drwx------  5 usuario staff  160 Jan 29 10:30 Desktop
-rw-r--r--  1 usuario staff 2048 Jan 28 15:45 archivo.txt
-rwxr-xr-x  1 usuario staff  512 Jan 27 09:00 script.sh
```

Listing 32.21 POWERSHELL

```
PS> Get-ChildItem -Force | Format-Table -AutoSize

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d----         29/1/2024  10:30                Desktop
-a---         28/1/2024  15:45           2048 archivo.txt
-a--x         27/1/2024  09:00            512 script.ps1
```

Listing 32.22 BASH

```
$ ls -lah
total 48
drwxr-xr-x  15 usuario grupo 4.0K Jan 29 10:30 .
drwxr-xr-x   3 root    root  4.0K Jan 15 08:00 ..
-rw-r--r--   1 usuario grupo  220 Jan 29 09:15 .bashrc
-rw-r--r--   1 usuario grupo  807 Jan 29 09:15 .bash_profile
drwxr-xr-x   2 usuario grupo 4.0K Jan 29 10:30 Documentos
                                                              ①
                                                              ②
```

Listing 32.23 BASH

```
$ ls -lah
total 48
drwx------  15 usuario staff  480 Jan 29 10:30 .
drwxr-xr-x   3 root    wheel  480 Jan 15 08:00 ..
-rw-r--r--   1 usuario staff  220 Jan 29 09:15 .zshrc
-rw-r--r--   1 usuario staff  807 Jan 29 09:15 .zprofile
drwx------   3 usuario staff   96 Jan 29 10:30 Desktop
                                                              ①
                                                              ②
```

Listing 32.24 POWERSHELL

```
PS> Get-ChildItem -Force -File | Select-Object Mode, LastWriteTime, Length, Name

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a---         29/1/2024  10:30                221 .bashrc
-a---         29/1/2024  10:30                807 .bash_profile
-a---         28/1/2024  15:45               2048 archivo.txt
                                                              ①
                                                              ②
```

Listing 32.25 BASH

```
$ ls -lhR ~/Documentos                                                    ①
~/Documentos:
total 8.0K
drwxr-xr-x 2 usuario grupo 4.0K Jan 29 10:30 Proyectos
-rw-r--r-- 1 usuario grupo 2.0K Jan 28 15:45 notas.txt

~/Documentos/Proyectos:
total 12K
-rw-r--r-- 1 usuario grupo 4.0K Jan 27 09:00 proyecto1.txt
-rw-r--r-- 1 usuario grupo 8.0K Jan 26 14:30 proyecto2.txt
                                                                          ②
```

Listing 32.26 BASH

```
$ ls -lhR ~/Documents
/Users/usuario/Documents:
total 8
drwx------  2 usuario staff  64 Jan 29 10:30 Projects
-rw-r--r--  1 usuario staff 2.0K Jan 28 15:45 notes.txt

/Users/usuario/Documents/Projects:
total 12
-rw-r--r--  1 usuario staff 4.0K Jan 27 09:00 project1.txt
-rw-r--r--  1 usuario staff 8.0K Jan 26 14:30 project2.txt
                                                                          ①
```

Listing 32.27 POWERSHELL

```
PS> Get-ChildItem -Path C:\Users\usuario\Documents -Recurse | Format-Table -AutoSize Mode

Mode                 Length Name
----                 ------ ----
d----                       Projects
-a---                 2048 notes.txt
d----
d----                 4096 project1.txt
d----                 8192 project2.txt
                                                                          ②
```

Listing 32.28 BASH

```
# Ruta absoluta (funciona desde cualquier lugar)
$ cd /home/usuario/Documentos/Proyectos

# Ruta relativa (si estás en /home/usuario)
$ cd Documentos/Proyectos

# Combinando con ..
$ cd ../Documentos

# Combinando con ~
$ cd ~/Documentos/Proyectos
```

Listing 32.29 BASH

```
# Ruta absoluta
$ cd /Users/usuario/Documents/Projects

# Ruta relativa (si estás en /Users/usuario)
$ cd Documents/Projects

# Combinando con ..
$ cd ../Documents

# Combinando con ~
$ cd ~/Documents/Projects
```

Listing 32.30 POWERSHELL

```
# Ruta absoluta
PS> cd C:\Users\usuario\Documents\Projects

# Ruta relativa (si estás en C:\Users\usuario)
PS> cd Documents\Projects

# Combinando con ..
PS> cd ..\Documents

# Combinando con ~ (home)
PS> cd ~\Documents\Projects
```

33 Inspección de Archivos

Listing 33.1 QUARTO-TITLE-BLOCK

34 Inspección de Archivos

34.1 Introducción

Una vez que dominas la navegación, el siguiente paso es entender el contenido de los archivos. Linux proporciona herramientas poderosas para inspeccionar, ver y analizar archivos sin necesidad de editarlos. Estas herramientas son esenciales para la administración del sistema.

En este tema aprenderás:

- Comando `cat` (concatenate) - ver contenido completo
- Comando `less` y `more` - navegar archivos grandes
- Comando `head` y `tail` - ver partes de archivos
- Comando `wc` (word count) - contar líneas, palabras, bytes
- Comando `file` - identificar tipo de archivo
- Comando `stat` - ver metadatos de archivos

34.2 El Comando `cat` - Ver Contenido Completo

El comando `cat` (concatenate) es el más simple para ver el contenido completo de un archivo. Es especialmente útil para archivos pequeños.

Sintaxis:

Listing 34.1 BASH

```
cat [opciones] [archivo(s)]
```

34.2.1 Ejemplos Prácticos Multi-SO

34.2.2 Ejemplo 1: Ver contenido de un archivo simple

34.2.2.1 Linux

① Muestra el nombre del host del sistema

Listing 34.2 BASH

```
$ cat /etc/hostname
ubuntu-servidor
```
①

34.2.2.2 macOS

Listing 34.3 BASH

```
$ cat /etc/hostname
MacBook-Pro.local
```
①

① En macOS también existe /etc/hostname

34.2.2.3 Windows

Listing 34.4 POWERSHELL

```
PS> Get-Content C:\Users\usuario\archivo.txt
# o para archivos del sistema
PS> (Get-ComputerInfo).CsDNSHostName
USUARIO-PC
```
①

① PowerShell usa `Get-Content` en lugar de `cat`

34.2.3 Ejemplo 2: Ver múltiples archivos

34.2.3.1 Linux

① `cat` puede concatenar varios archivos

34.2.3.2 macOS

① Funciona igual que Linux

34.2.3.3 Windows

① PowerShell puede leer múltiples archivos

Listing 34.5 BASH

```
$ cat archivo1.txt archivo2.txt
Contenido del archivo 1
Contenido del archivo 2
```
①

Listing 34.6 BASH

```
$ cat file1.txt file2.txt
Content of file 1
Content of file 2
```
①

34.2.4 Ejemplo 3: Ver archivo con números de línea (cat -n)

34.2.4.1 Linux

① `-n` agrega números de línea, útil para referencia

34.2.4.2 macOS

① El contenido varía según la configuración DNS

34.2.4.3 Windows

① PowerShell requiere un poco más de trabajo para numerar líneas

Cuándo usar: Usa `cat` para archivos pequeños (< 1000 líneas). Para archivos grandes, usa `less` o `more`.

34.3 Los Comandos `less` y `more` - Navegar Archivos Grandes

Estos comandos permiten navegar por archivos grandes sin cargar todo en pantalla.

Sintaxis:

Listing 34.7 POWERSHELL

```
PS> Get-Content archivo1.txt, archivo2.txt
Content of file 1
Content of file 2
```
①

Listing 34.8 BASH

```
$ cat -n /etc/resolv.conf
     1  nameserver 8.8.8.8
     2  nameserver 8.8.4.4
```
①

34.3.1 Controles principales en `less`:

- `Space` o `Page Down` : Siguiente página
- `b` o `Page Up` : Página anterior
- `g` : Ir al inicio
- `G` : Ir al final
- `/patrón` : Buscar patrón
- `n` : Siguiente coincidencia
- `q` : Salir

34.3.2 Ejemplos Prácticos Multi-SO

34.3.3 Ejemplo 1: Ver archivo grande con less

34.3.3.1 Linux

① `less` es más eficiente que `cat` para archivos > 1MB

34.3.3.2 macOS

① Los logs del sistema están en `/var/log/` en macOS

34.3.3.3 Windows

① Windows no tiene `less` por defecto
② `Out-Host -Paging` es el equivalente más cercano

Diferencia entre `less` y `more`:

- `less` es más moderno y flexible

Listing 34.9 BASH

```
$ cat -n /etc/resolv.conf
     1  nameserver 1.1.1.1
     2  nameserver 1.0.0.1
                                                                    ①
```

Listing 34.10 POWERSHELL

```
PS> (Get-Content archivo.txt) | Select-Object -Property @{Name="Line"; Expression={$_}},
    Select-Object LineNum, Line

# Alternativa más simple:
PS> @(Get-Content archivo.txt) | ForEach-Object {"{0,4}: {1}" -f ++$line, $_}
   1: Primera línea
   2: Segunda línea
                                                                    ①
```

- `more` es más antiguo pero universal
- En Linux/macOS, `less` es preferible

34.4 Los Comandos `head` y `tail` - Ver Partes de Archivos

Estos comandos muestran el inicio o final de un archivo, útiles para inspeccionar sin ver todo.

Sintaxis:

34.4.1 Ejemplos Prácticos Multi-SO

34.4.2 Ejemplo 1: Ver primeras 10 líneas (head)

34.4.2.1 Linux

① Por defecto muestra 10 líneas

34.4.2.2 macOS

① Formato similar a Linux

Listing 34.11 BASH

```
less [archivo]
more [archivo]
```

Listing 34.12 BASH

```
$ less /var/log/syslog
# Navega con space, busca con /, sale con q
                                                                    (1)
```

34.4.2.3 Windows

(1) PowerShell usa `-Head` o `-TotalCount`

34.4.3 Ejemplo 2: Ver últimas 5 líneas (tail)

34.4.3.1 Linux

(1) Útil para ver los eventos más recientes en logs

34.4.3.2 macOS

(1) El log system en macOS está estructurado igual

34.4.3.3 Windows

(1) PowerShell usa `-Tail` para ver las últimas líneas

34.4.4 Ejemplo 3: Monitorizar archivo en tiempo real (tail -f)

34.4.4.1 Linux

(1) `-f` (follow) monitoriza cambios en tiempo real
(2) Perfecto para ver logs mientras ocurren eventos

34.4.4.2 macOS

(1) Excelente para monitorizar logs del sistema

Listing 34.13 BASH

```
$ less /var/log/system.log
# Mismo funcionamiento que Linux
                                                                    ①
```

Listing 34.14 POWERSHELL

```
# PowerShell tiene 'more' pero es distinto
PS> Get-Content archivo-grande.txt | more
# Para mejor experiencia, usar:
PS> Get-Content archivo-grande.txt -ReadCount 0 | Out-Host -Paging      ②
                                                                    ①
```

34.4.4.3 Windows

① `-Wait` intenta emular `-f`
② La alternativa con loop es más controlable pero consume más CPU

Función	Linux	macOS	Windows
Primeras líneas	`head -n 10`	`head -n 10`	`-Head 10`
Últimas líneas	`tail -n 10`	`tail -n 10`	`-Tail 10`
Monitorizar	`tail -f`	`tail -f`	`-Wait`
Eficiencia	Muy alta	Muy alta	Moderada

Cuándo usar:

- `head` para ver encabezados o primeras filas
- `tail` para ver logs recientes
- `tail -f` para monitorizar eventos en tiempo real

34.5 El Comando `wc` - Contar Líneas, Palabras y Bytes

El comando `wc` (word count) te permite contar líneas, palabras y caracteres en archivos.

Sintaxis:

34.5.1 Opciones principales:

- `-l` : Contar líneas
- `-w` : Contar palabras
- `-c` : Contar bytes
- `-m` : Contar caracteres

34.5.2 Ejemplos Prácticos Multi-SO

34.5.3 Ejemplo 1: Contar líneas en un archivo

Listing 34.15 BASH

```
head -n [número] [archivo]
tail -n [número] [archivo]
```

Listing 34.16 BASH

```
$ head /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
# ... (7 líneas más)
```

①

34.5.3.3 Windows

① PowerShell cuenta diferente (puede dar resultados distintos)

34.5.4 Ejemplo 2: Contar líneas, palabras y bytes

34.5.4.1 Linux

① Formato: líneas, palabras, bytes, nombre

34.5.4.2 macOS

① Mismo formato con espaciado diferente

34.5.4.3 Windows

① PowerShell requiere múltiples comandos para lo mismo

Opción	Linux	macOS	Windows	Descripción
`-l`			Measure-Object	Contar líneas
`-w`			-Word	Contar palabras
`-c`			Get-Item .Length	Contar bytes
`-m`			Parcial	Contar caracteres

Listing 34.17 BASH

```
$ head /etc/passwd
root:*:0:0:System Administrator:/var/root:/bin/bash
daemon:*:1:1:System Services:/var/empty:/usr/sbin/nologin
# ... (8 líneas más)
```
①

Listing 34.18 POWERSHELL

```
PS> Get-Content archivo.txt -Head 10
# o
PS> Get-Content archivo.txt -TotalCount 10
```
①

34.6 El Comando `file` - Identificar Tipo de Archivo

El comando `file` determina el tipo de un archivo analizando su contenido, no solo su extensión.

Sintaxis:

34.6.1 Ejemplos Prácticos Multi-SO

34.6.2 Ejemplo 1: Identificar tipo de archivo

34.6.2.1 Linux

① Detecta ejecutables ELF, archivos de texto, logs, etc.

34.6.2.2 macOS

① macOS usa formato Mach-O para ejecutables

34.6.2.3 Windows

① `file` no existe en PowerShell nativo
② Se puede usar desde WSL si está disponible

Cuándo usar: Usa `file` cuando:

- No estés seguro del tipo de archivo
- Recibas archivos sin extensión
- Necesites validar integridad de archivos

Listing 34.19 BASH

```
$ tail -n 5 /var/log/syslog
Jan 29 10:45:32 ubuntu kernel: [1234.567] System log message 1
Jan 29 10:46:00 ubuntu kernel: [1267.834] System log message 2
Jan 29 10:46:15 ubuntu systemd[1]: Started Service X.
# ... (2 líneas más)
```

①

Listing 34.20 BASH

```
$ tail -n 5 /var/log/system.log
Jan 29 10:45:32 MacBook kernel[0]: message 1
Jan 29 10:46:00 MacBook systemd[1]: Started Service X.
# ... (3 líneas más)
```

①

34.7 El Comando `stat` - Ver Metadatos Detallados

El comando `stat` muestra información detallada sobre archivos (permisos, propietario, fechas, inodos, etc.).

Sintaxis:

34.7.1 Ejemplos Prácticos Multi-SO

34.7.2 Ejemplo 1: Ver metadatos completos de un archivo

34.7.2.1 Linux

① Muestra todos los metadatos disponibles

34.7.2.2 macOS

① Formato ligeramente distinto a Linux

34.7.2.3 Windows

① PowerShell usa `Get-Item` para metadatos similares

Listing 34.21 POWERSHELL

```
PS> Get-Content archivo.txt -Tail 5
# o más detallado:
PS> Get-Content archivo.txt | Select-Object -Last 5
```
①

Listing 34.22 BASH

```
$ tail -f /var/log/apache2/access.log
192.168.1.100 - - [29/Jan/2024:10:45:30 +0000] "GET / HTTP/1.1" 200 1234
192.168.1.101 - - [29/Jan/2024:10:46:00 +0000] "GET /api HTTP/1.1" 200 5678
# ... (continúa mostrando nuevas líneas)
# Presiona Ctrl+C para salir
```
① ②

Información	Linux	macOS	Windows
Tamaño			
Permisos			Parcial
Propietario			
Inodo			-
Fechas (crear, modificar)			

34.8 Resumen de Comandos de Inspección

Comando	Propósito	Cuándo usar
`cat archivo`	Ver contenido completo	Archivos pequeños
`less archivo`	Ver con navegación	Archivos grandes
`head -n 10 archivo`	Ver primeras 10 líneas	Inspección rápida
`tail -n 10 archivo`	Ver últimas 10 líneas	Ver logs recientes
`tail -f archivo`	Monitorizar en tiempo real	Watching logs
`wc -l archivo`	Contar líneas	Analizar tamaño
`file archivo`	Identificar tipo	Verificar tipo
`stat archivo`	Ver metadatos	Análisis profundo

Listing 34.23 BASH

```
$ tail -f ~/Library/Logs/system.log
# Mismo comportamiento que Linux
```
①

Listing 34.24 POWERSHELL

```
# PowerShell no tiene -f nativo, pero:
PS> Get-Content archivo.txt -Wait -Tail 10
# Para monitorizar más eficientemente:
PS> while($true) { Clear-Host; Get-Content archivo.txt | Select-Object -Last 20; Start-Sl
```
① ②

34.9 Ejercicios Prácticos

1. Crea un archivo `prueba.txt` con 50 líneas de texto
2. Usa `head` y `tail` para ver partes del archivo
3. Usa `wc -l` para contar las líneas
4. Identifica el tipo con `file prueba.txt`
5. Inspecciona metadatos con `stat prueba.txt`
6. Crea un script y usa `file` para verificar su tipo

34.10 Referencias

Para más información sobre inspección de archivos:

- (IEEE 2018) - POSIX File Operations
- (F. S. Foundation 2023a) - GNU Bash Text Processing

Listing 34.25 BASH

```
wc [opciones] [archivo]
```

Listing 34.26 BASH

```
$ wc -l /etc/passwd
45 /etc/passwd
```

①

Listing 34.27 BASH

```
$ wc -l /etc/passwd
      47 /etc/passwd
```

①

Listing 34.28 POWERSHELL

```
PS> (Get-Content archivo.txt).Count
47
# o más explícito:
PS> @(Get-Content archivo.txt).Length
47
```

①

Listing 34.29 BASH

```
$ wc /etc/hostname
 1  1 15 /etc/hostname
# líneas palabras bytes
```

①

Listing 34.30 BASH

```
$ wc /etc/hostname
       1       1      15 /etc/hostname
```

①

Listing 34.31 POWERSHELL

```
PS> $content = Get-Content archivo.txt
PS> $lines = @($content).Count
PS> $words = ($content | Measure-Object -Word).Words
PS> $bytes = (Get-Item archivo.txt).Length
PS> "$lines $words $bytes"
1 1 15
```

①

Listing 34.32 BASH

```
file [opciones] [archivo(s)]
```

Listing 34.33 BASH

```
$ file /bin/bash
/bin/bash: ELF 64-bit LSB pie executable, x86-64, version 1
$ file /etc/hostname
/etc/hostname: ASCII text
$ file /var/log/syslog
/var/log/syslog: ASCII text
```

①

Listing 34.34 BASH

```
$ file /bin/bash
/bin/bash: Mach-O 64-bit executable x86_64
$ file /etc/hostname
/etc/hostname: ASCII text
```

①

Listing 34.35 POWERSHELL

```
# Windows no tiene 'file' nativo, alternativas:
PS> (Get-Item archivo.exe).VersionInfo
# o simple:
PS> Get-Item archivo.exe | ForEach-Object {$_.Extension}
.exe

# Mejor opción si tienes WSL o Git Bash:
PS> wsl file archivo.exe
```

② ①

Listing 34.36 BASH

```bash
stat [opciones] [archivo]
```

Listing 34.37 BASH

```bash
$ stat archivo.txt
  File: archivo.txt
  Size: 1024        Blocks: 8          IO Block: 4096
Device: 10302h/66306d      Inode: 2097154     Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/ usuario)   Gid: ( 1000/ usuario)
Access: 2024-01-29 10:45:32.123456789 +0000
Modify: 2024-01-28 15:30:15.987654321 +0000
Change: 2024-01-28 15:30:15.987654321 +0000
Birth: -
```

①

Listing 34.38 BASH

```bash
$ stat archivo.txt
  File: archivo.txt
  Size: 1024       FileType: Regular File
Mode: (0644/-rw-r--r--)         Uid: (  501/ usuario)  Gid: (   20/ staff)
Device: 1000006h/16777222d Inode: 123456      Links: 1
Access: Mon Jan 29 10:45:32 2024
Modify: Sun Jan 28 15:30:15 2024
Change: Sun Jan 28 15:30:15 2024
Birth: Sun Jan 28 15:30:15 2024
```

①

Listing 34.39 POWERSHELL

```powershell
PS> (Get-Item archivo.txt) | Select-Object *

Name                    : archivo.txt
FullName                : C:\Users\usuario\archivo.txt
CreationTime            : 28/01/2024 15:30:15
LastAccessTime          : 29/01/2024 10:45:32
LastWriteTime           : 28/01/2024 15:30:15
```

①

35 Búsqueda de Archivos y Contenido

Listing 35.1 QUARTO-TITLE-BLOCK

36 Búsqueda de Archivos y Contenido

36.1 Introducción

Encontrar archivos y contenido específico es una tarea diaria para cualquier administrador de Linux. Los comandos `find` y `grep` son herramientas poderosas que te permitirán localizar exactamente lo que necesitas en un sistema con millones de archivos.

En este tema aprenderás:

- Comando `find` - búsqueda por nombre, tipo, tamaño, fecha
- Comando `grep` - búsqueda en contenido de archivos
- Comando `locate` - búsqueda rápida por base de datos
- Combinación de comandos con pipes
- Uso de expresiones regulares
- Búsquedas avanzadas y filtros

36.2 El Comando `find` - Búsqueda por Atributos de Archivo

El comando `find` es extremadamente poderoso para localizar archivos basándose en nombre, tipo, tamaño, permisos, fecha de modificación y más.

Sintaxis básica:

Listing 36.1 BASH

```
find [ruta] [condiciones] [acciones]
```

36.2.1 Condiciones principales:

- `-name patrón` : Buscar por nombre
- `-type f|d|l` : Buscar por tipo (archivo, directorio, enlace)
- `-size +/- n` : Buscar por tamaño
- `-mtime n` : Modificado hace n días
- `-perm modo` : Buscar por permisos
- `-user usuario` : Archivos de un usuario

36.2.2 Ejemplos Prácticos Multi-SO

36.2.3 Ejemplo 1: Buscar archivo por nombre

36.2.3.1 Linux

Listing 36.2 BASH

```
$ find ~ -name "*.txt"
/home/usuario/archivo1.txt
/home/usuario/Documentos/notas.txt
/home/usuario/Proyectos/proyecto.txt
```
①

① `-name` busca por nombre de archivo (case-sensitive)

36.2.3.2 macOS

Listing 36.3 BASH

```
$ find ~ -name "*.txt"
/Users/usuario/archivo1.txt
/Users/usuario/Documents/notes.txt
/Users/usuario/Projects/project.txt
```
①

① Funciona igual que Linux

36.2.3.3 Windows

① PowerShell usa `Get-ChildItem` con `-Filter` y `-Recurse`

36.2.4 Ejemplo 2: Buscar archivos grandes (mayor a 100MB)

36.2.4.1 Linux

① `-size +100M` busca archivos mayores a 100MB
② `+` significa "mayor que", `-` significa "menor que"

36.2.4.2 macOS

① Mismo funcionamiento que Linux

Listing 36.4 POWERSHELL

```
# find no existe en Windows nativo
# Usar Get-ChildItem recursivamente:
PS> Get-ChildItem -Path C:\Users\usuario -Filter "*.txt" -Recurse

    Directory: C:\Users\usuario\Documents

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a---          29/1/2024     10:30              1024 archivo1.txt
-a---          28/1/2024     15:45              2048 notas.txt

# O si tienes WSL:
PS> wsl find ~ -name "*.txt"
                                                                    ①
```

Listing 36.5 BASH

```
$ find / -type f -size +100M                                        ①
/var/lib/docker/overlay2/abc123.../layer.tar
/home/usuario/videos/pelicula.mp4
/var/cache/cache-file.bin
                                                                    ②
```

36.2.4.3 Windows

① PowerShell usa `-File` y `Where-Object` con tamaño en bytes

36.2.5 Ejemplo 3: Buscar archivos modificados en últimos 7 días

36.2.5.1 Linux

① `-mtime -7` = modificado en últimos 7 días (negativo = pasado reciente)
② `-mtime +7` = modificado hace más de 7 días

36.2.5.2 macOS

① Mismo formato que Linux

36.2.5.3 Windows

① PowerShell requiere calcular la fecha manualmente

Listing 36.6 BASH

```
$ find ~ -type f -size +100M
/Users/usuario/Videos/pelicula.mp4
/Users/usuario/Library/Caches/cache-file.bin
```
①

Listing 36.7 POWERSHELL

```
# Buscar archivos > 100MB
PS> Get-ChildItem -Path C:\Users -Recurse -File |
    Where-Object {$_.Length -gt 100MB}

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a---          29/1/2024     10:30      104857600 pelicula.mp4
```
①

36.2.6 Ejemplo 4: Buscar directorios vacíos

36.2.6.1 Linux

① `-type d` busca solo directorios
② `-empty` busca elementos sin contenido

36.2.6.2 macOS

① Funciona igual

36.2.6.3 Windows

① Se requieren verificaciones adicionales en PowerShell

Cuándo usar: `find` es tu herramienta para búsquedas estructuradas por metadatos de archivo.

36.3 El Comando `grep` - Búsqueda en Contenido

El comando `grep` busca patrones de texto dentro de archivos. Es extremadamente poderoso cuando lo combinas con expresiones regulares.

Sintaxis básica:

Listing 36.8 BASH

```
$ find ~ -type f -mtime -7                                                    ①
/home/usuario/Documentos/archivo-reciente.txt
/home/usuario/Descargas/descarga.pdf
                                                                              ②
```

Listing 36.9 BASH

```
$ find ~ -type f -mtime -7
/Users/usuario/Documents/archivo-reciente.txt
                                                                              ①
```

36.3.1 Opciones principales:

- `-i` : Ignorar mayúsculas/minúsculas
- `-r` : Buscar recursivamente en directorios
- `-n` : Mostrar número de línea
- `-c` : Contar coincidencias
- `-v` : Invertir (mostrar líneas que NO coinciden)
- `-E` : Usar expresiones regulares extendidas

36.3.2 Ejemplos Prácticos Multi-SO

36.3.3 Ejemplo 1: Búsqueda simple en archivo

36.3.3.1 Linux

① Busca la palabra “error” en el archivo de log

36.3.3.2 macOS

① Mismo funcionamiento

36.3.3.3 Windows

① `-Pattern` es el equivalente a patrón en grep
② Devuelve línea y número automáticamente

Listing 36.10 POWERSHELL

```
# Archivos modificados en últimos 7 días
$fecha = (Get-Date).AddDays(-7)
PS> Get-ChildItem -Path C:\Users\usuario -Recurse -File |
    Where-Object {$_.LastWriteTime -gt $fecha}
                                                                    ①
```

Listing 36.11 BASH

```
$ find ~ -type d -empty                                             ①
/home/usuario/directorio-vacio
/home/usuario/proyectos/carpeta-vieja
                                                                    ②
```

36.3.4 Ejemplo 2: Búsqueda recursiva en directorios (grep -r)

36.3.4.1 Linux

① `-r` busca en todos los archivos del directorio recursivamente

36.3.4.2 macOS

① Funciona igual

36.3.4.3 Windows

① `-Recurse` es el equivalente a `-r` de grep

36.3.5 Ejemplo 3: Búsqueda con número de línea y case-insensitive

36.3.5.1 Linux

① `-i` ignora mayúsculas
② `-n` muestra número de línea

36.3.5.2 macOS

① Funciona igual

Listing 36.12 BASH

```
$ find ~ -type d -empty
/Users/usuario/empty-folder
                                                                    (1)
```

Listing 36.13 POWERSHELL

```
# Directorios vacíos
PS> Get-ChildItem -Path C:\Users\usuario -Recurse -Directory |
    Where-Object {(Get-ChildItem $_.FullName -Recurse).Count -eq 0}
                                                                    (1)
```

36.3.5.3 Windows

① Sin `-CaseSensitive` es case-insensitive por defecto

36.3.6 Ejemplo 4: Contar coincidencias (grep -c)

36.3.6.1 Linux

① `-c` solo muestra el número de coincidencias

36.3.6.2 macOS

① Funciona igual

36.3.6.3 Windows

① PowerShell cuenta las coincidencias automáticamente

36.3.7 Ejemplo 5: Expresiones regulares (grep -E)

36.3.7.1 Linux

① `-E` habilita expresiones regulares extendidas
② `^` significa "comienza la línea"
③ `|` significa "o"

36.3.7.2 macOS

① Funciona igual

Listing 36.14 BASH

```
grep [opciones] patrón [archivo(s)]
```

Listing 36.15 BASH

```
$ grep "error" /var/log/syslog
[10:45:32] error: connection timeout
[10:46:00] error: authentication failed
```
①

36.3.7.3 Windows

① PowerShell usa regex compatibles con .NET por defecto

Cuándo usar: `grep` es tu herramienta para búsquedas en el contenido de archivos.

36.4 Combinando `find` y `grep` con Pipes

La verdadera potencia viene al combinar herramientas. Los pipes (`|`) permiten pasar salida de un comando a otro.

36.4.1 Ejemplos Prácticos Multi-SO

36.4.2 Ejemplo 1: Buscar archivos y filtrar contenido

36.4.2.1 Linux

① `-exec` ejecuta un comando en cada resultado
② `xargs` pasa los resultados como argumentos

36.4.2.2 macOS

① Funciona igual que Linux

36.4.2.3 Windows

① PowerShell usa tuberías (pipes) con `|` igual que bash

Listing 36.16 BASH

```
$ grep "error" ~/Library/Logs/system.log
[10:45:32] error: connection timeout
                                                                    ①
```

Listing 36.17 POWERSHELL

```
# grep no existe, usar Select-String:
PS> Select-String -Path C:\path\to\file.log -Pattern "error"        ①

archivo.log:5:error: connection timeout
archivo.log:8:error: authentication failed
                                                                    ②
```

36.4.3 Ejemplo 2: Encontrar archivos grandes con patrones específicos

36.4.3.1 Linux

① Combina múltiples condiciones de `find`

36.4.3.2 macOS

① Funciona similar

36.4.3.3 Windows

① PowerShell requiere más pasos pero es más legible

36.5 El Comando `locate` - Búsqueda Rápida

El comando `locate` es mucho más rápido que `find` porque usa una base de datos pre-indexada. Sin embargo, solo busca nombres de archivo, no contenido.

Sintaxis:

Listing 36.18 BASH

```
$ grep -r "password" ~/proyectos/
proyectos/config.txt:mysql_password=secret123
proyectos/app/settings.py:password_hash = "abc123"
```
①

Listing 36.19 BASH

```
$ grep -r "password" ~/Projects/
Projects/config.txt:mysql_password=secret123
```
①

36.5.1 Ejemplos Prácticos Multi-SO

36.5.2 Ejemplo 1: Búsqueda rápida por nombre

36.5.2.1 Linux

① `locate` es mucho más rápido que `find`
② Requiere que la BD se actualice con `updatedb`

36.5.2.2 macOS

① La BD se actualiza automáticamente cada semana

36.5.2.3 Windows

① Sin herramienta especializada, es más lento
② Software como "Everything" proporciona funcionalidad similar

Cuándo usar:

- Usa `locate` para búsquedas rápidas por nombre
- Usa `find` para búsquedas complejas con múltiples condiciones

36.6 Resumen de Comandos de Búsqueda

Listing 36.20 POWERSHELL

```
# Búsqueda recursiva
PS> Select-String -Path C:\proyectos -Pattern "password" -Recurse

config.txt:1:mysql_password=secret123
settings.py:15:password_hash = "abc123"
```
①

Listing 36.21 BASH

```
$ grep -in "ERROR" /var/log/syslog
45:[10:45:32] Error: connection timeout
67:[10:46:00] ERROR: authentication failed
```
①
②

Comando	Propósito	Velocidad	Complejidad
`find`	Búsqueda por metadatos	Lenta pero completa	Alta
`grep`	Búsqueda en contenido	Moderada	Media
`locate`	Búsqueda rápida por nombre	Muy rápida	Baja
`grep -r`	Búsqueda recursiva en contenido	Lenta	Media
Combinadas (find+grep)	Búsqueda avanzada	Lenta pero poderosa	Muy alta

36.7 Ejercicios Prácticos

1. Usa `find` para localizar todos los archivos `.txt` en tu home
2. Usa `grep` para buscar “Linux” en todos los archivos del directorio actual
3. Busca archivos modificados en los últimos 30 días
4. Encuentra archivos mayores a 10MB
5. Busca líneas que contengan números usando `grep` con regex
6. Combina `find` y `grep` para buscar “error” en todos los logs

Listing 36.22 BASH

```
$ grep -in "ERROR" ~/Library/Logs/system.log
45:[10:45:32] Error: connection timeout
```

①

Listing 36.23 POWERSHELL

```
PS> Select-String -Path archivo.log -Pattern "ERROR" `
    -CaseSensitive:$false |
    Select-Object -Property LineNumber, Line

LineNumber Line
---------- ----
45         [10:45:32] Error: connection timeout
67         [10:46:00] ERROR: authentication failed
```

①

36.8 Referencias

Para más información sobre búsqueda de archivos:

- (IEEE 2018) - POSIX File Search
- (G. Project 2023) - GNU Find Documentation
- (F. S. Foundation 2023b) - GNU Grep Manual

Listing 36.24 BASH

```
$ grep -c "Failed password" /var/log/auth.log
42
```
①

Listing 36.25 BASH

```
$ grep -c "Failed password" /var/log/auth.log
42
```
①

Listing 36.26 POWERSHELL

```
PS> (Select-String -Path auth.log -Pattern "Failed password").Count
42
```
①

Listing 36.27 BASH

```
# Buscar líneas que comienzan con Error o Warning
$ grep -E "^(Error|Warning)" /var/log/app.log
Error: Database connection failed
Warning: Memory usage high
Error: API timeout
```
①
②
③

Listing 36.28 BASH

```
$ grep -E "^(Error|Warning)" ~/Library/Logs/app.log
Error: Database connection failed
```
①

Listing 36.29 POWERSHELL

```
# PowerShell usa .NET regex:
PS> Select-String -Path app.log -Pattern "^(Error|Warning)"

app.log:1:Error: Database connection failed
app.log:3:Warning: Memory usage high
app.log:5:Error: API timeout
```
①

Listing 36.30 BASH

```
# Encontrar todos los .txt y buscar "importante" en ellos
$ find ~ -name "*.txt" -exec grep -l "importante" {} \;
/home/usuario/Documentos/notas.txt
/home/usuario/tareas.txt

# Alternativa con pipe (más moderna):
$ find ~ -name "*.txt" | xargs grep "importante"
Documentos/notas.txt:Este es un punto importante
tareas.txt:importante: completar antes del viernes
```

①
②

Listing 36.31 BASH

```
$ find ~ -name "*.txt" -exec grep -l "importante" {} \;
/Users/usuario/Documents/notes.txt

$ find ~ -name "*.txt" | xargs grep "importante"
Documents/notes.txt:Este es un punto importante
```

①

Listing 36.32 POWERSHELL

```
# Buscar archivos .txt con contenido "importante"
PS> Get-ChildItem -Path C:\Users\usuario -Filter "*.txt" -Recurse |
    ForEach-Object { Select-String -Path $_.FullName -Pattern "importante" }

Documents\notes.txt:3:Este es un punto importante
```

①

Listing 36.33 BASH

```
# Buscar archivos .log > 1MB que contengan "error"
$ find /var/log -name "*.log" -size +1M -exec grep -l "error" {} \;
/var/log/syslog
/var/log/kern.log
```

①

Listing 36.34 BASH

```
$ find ~/Library/Logs -name "*.log" -size +1M | xargs grep -l "error"
~/Library/Logs/system.log
```

①

Listing 36.35 POWERSHELL

```
PS> Get-ChildItem -Path C:\Logs -Filter "*.log" -Recurse |
    Where-Object {$_.Length -gt 1MB} |
    ForEach-Object { Select-String -Path $_.FullName -Pattern "error" } |
    Select-Object -Unique Path
                                                                    ①
```

Listing 36.36 BASH

```
locate [opciones] patrón
```

Listing 36.37 BASH

```
# Actualizar base de datos (se hace automáticamente cada noche)
$ sudo updatedb                                                     ②

# Buscar archivos que contengan "apache" en el nombre
$ locate apache
/etc/apache2/apache2.ctl
/usr/bin/apache2ctl
/var/log/apache2/access.log
                                                                    ①
```

Listing 36.38 BASH

```
# macOS usa /var/db/locate.database
$ locate apache
/etc/apache2/apache2.ctl
/usr/local/bin/apache2ctl
                                                                    ①
```

Listing 36.39 POWERSHELL

```
# Windows no tiene locate nativo
# Alternativa: crear índice con Everything o:
PS> Get-ChildItem -Path C:\ -Recurse -Filter "*apache*" -ErrorAction SilentlyContinue
                                                                    ①
                                                                    ②
```

37 Permisos de Archivos

Listing 37.1 QUARTO-TITLE-BLOCK

38 Permisos de Archivos

38.1 Introducción

Los permisos de archivos son fundamentales en Linux. Controlan quién puede leer, escribir o ejecutar un archivo. Entender y manejar permisos correctamente es esencial para la seguridad del sistema.

En este tema aprenderás:

- Estructura de permisos en Linux (rwx)
- Permiso de lectura (r), escritura (w), ejecución (x)
- Propietario, grupo y otros
- Notación octal vs simbólica
- Comando `chmod` - cambiar permisos
- Comando `chown` - cambiar propietario
- Comando `chgrp` - cambiar grupo
- Permisos especiales (setuid, setgid, sticky)

38.2 Entendiendo los Permisos de Linux

En Linux, cada archivo tiene 3 conjuntos de permisos: propietario, grupo y otros.

38.2.1 Estructura: `-rwxrwxrwx`

```
-   rwx   rwx   rwx
|   |     |     |
|   |     |     +-- Permisos de otros (others)
|   |     +-------- Permisos de grupo (group)
|   +-------------- Permisos del propietario (user/owner)
+------------------ Tipo de archivo (- = archivo regular)
```

38.2.2 Significado de rwx:

- **r** (read) = Lectura (4)
- **w** (write) = Escritura (2)
- **x** (execute) = Ejecución (1)

38.2.3 Ejemplo de interpretación:

```
-rw-r--r--

- = archivo regular
rw- = propietario puede leer y escribir (no ejecutar)
r-- = grupo solo puede leer
r-- = otros solo pueden leer
```

38.3 Notación Octal de Permisos

Cada permiso puede representarse como número:

- **r** = 4
- **w** = 2
- **x** = 1

Se suman para obtener un dígito de 0-7:

Número	Permisos	Explicación
7	rwx	Leer + escribir + ejecutar
6	rw-	Leer + escribir
5	r-x	Leer + ejecutar
4	r–	Solo leer
3	-wx	Escribir + ejecutar
2	-w-	Solo escribir
1	–x	Solo ejecutar
0	—	Sin permisos

38.3.1 Ejemplos:

- `755` = rwxr-xr-x (dueño todo, grupo y otros ejecutar)
- `644` = rw-r–r– (dueño leer/escribir, grupo/otros solo leer)
- `777` = rwxrwxrwx (todos pueden hacer todo)

38.4 El Comando `chmod` - Cambiar Permisos

El comando `chmod` (change mode) modifica los permisos de archivos.

Sintaxis:

Listing 38.1 BASH

```
chmod [opciones] [modo] [archivo]
```

38.4.1 Modalidades:

1. **Octal**: `chmod 644 archivo`
2. **Simbólica**: `chmod u+x archivo`

38.4.2 Ejemplos Prácticos Multi-SO

38.4.3 Ejemplo 1: Hacer un script ejecutable

38.4.3.1 Linux

Listing 38.2 BASH

```
$ ls -l script.sh
-rw-r--r-- 1 usuario grupo 1024 Jan 29 10:30 script.sh

$ chmod +x script.sh                                          ①
$ ls -l script.sh
-rwxr-xr-x 1 usuario grupo 1024 Jan 29 10:30 script.sh
                                                              ②
```

① `+x` agrega permisos de ejecución para todos
② Se puede ejecutar ahora: `./script.sh`

38.4.3.2 macOS

Listing 38.3 BASH

```
$ chmod +x script.sh
$ ls -l script.sh
-rwxr-xr-x@ 1 usuario staff 1024 Jan 29 10:30 script.sh
                                                              ①
```

① Funciona igual (@ indica atributos extendidos de macOS)

Listing 38.4 POWERSHELL

```
# Windows no tiene permisos estilo Unix en NTFS                    ①
# Si estás en WSL:
PS> wsl chmod +x script.sh

# En PowerShell puro, no aplica igual
                                                                   ②
```

38.4.3.3 Windows

① Windows usa permisos ACL diferentes
② WSL hereda permisos del sistema de archivos de Windows

38.4.4 Ejemplo 2: Usar notación octal (chmod 644)

38.4.4.1 Linux

Listing 38.5 BASH

```
$ chmod 644 archivo.txt
$ ls -l archivo.txt
-rw-r--r-- 1 usuario grupo 1024 Jan 29 10:30 archivo.txt
# Interpretación: 6=rw-, 4=r--, 4=r--
                                                                   ①
                                                                   ②
                                                                   ③
```

① 6 (rw-) para propietario
② 4 (r–) para grupo
③ 4 (r–) para otros

38.4.4.2 macOS

Listing 38.6 BASH

```
$ chmod 644 archivo.txt
$ ls -l archivo.txt
-rw-r--r-- 1 usuario staff 1024 Jan 29 10:30 archivo.txt
                                                                   ①
```

① Funciona exactamente igual

38.4.4.3 Windows

Listing 38.7 POWERSHELL

```
# En WSL:
PS> wsl chmod 644 archivo.txt

# En PowerShell puro, usar Set-Acl
PS> $acl = Get-Acl archivo.txt
PS> $rule = New-Object System.Security.AccessControl.FileSystemAccessRule `
    ("usuario", "Modify", "Allow")
                                                                              (1)
                                                                              (2)
```

(1) Mucho más complejo en PowerShell
(2) WSL es más práctico

38.4.5 Ejemplo 3: Notación simbólica (u+rwx, g-w, o-rwx)

38.4.5.1 Linux

Listing 38.8 BASH

```
# Símbolos: u (user/owner), g (group), o (others), a (all)
# Operadores: + (agregar), - (remover), = (establecer)

$ chmod u+x archivo.txt         # Agregar ejecución al propietario
$ chmod g-w archivo.txt         # Remover escritura del grupo
$ chmod o-rwx archivo.txt       # Remover todos los permisos a otros
$ chmod a+r archivo.txt         # Agregar lectura a todos
$ chmod u=rwx,g=rx,o= archivo.txt  # Establecer permisos exactos
                                                                              (1)
                                                                              (2)
```

(1) Muy flexible y legible
(2) Puedes combinar múltiples cambios con comas

38.4.5.2 macOS

(1) Mismo comando y sintaxis

38.4.5.3 Windows

(1) Se requiere WSL para sintaxis Unix

Listing 38.9 BASH

```
$ chmod u+x archivo.txt
# Funciona idéntico a Linux
```
①

Listing 38.10 POWERSHELL

```
# En WSL:
PS> wsl chmod u+x archivo.txt
```
①

38.4.6 Ejemplo 4: Cambiar permisos recursivamente (-R)

38.4.6.1 Linux

Listing 38.11 BASH

```
# Cambiar permisos de todos los archivos en un directorio
$ chmod -R 755 ~/public_html
```
①

① `-R` aplica recursivamente a todos los subdirectorios

38.4.6.2 macOS

① Funciona igual

38.4.6.3 Windows

① Use WSL para esto

Notación	Ejemplos	Cuándo usar
Octal	`chmod 644 archivo`	Scripts, automatización
Simbólica	`chmod u+x archivo`	Cambios rápidos, legibilidad
Recursiva	`chmod -R 755 dir`	Cambiar directorios completos

Listing 38.12 BASH

```
$ chmod -R 755 ~/public_html
                                                                    ①
```

Listing 38.13 POWERSHELL

```
PS> wsl chmod -R 755 ~/public_html
                                                                    ①
```

38.5 El Comando `chown` - Cambiar Propietario

El comando `chown` (change owner) cambia el propietario y/o grupo de archivos.

Sintaxis:

Listing 38.14 BASH

```
chown [propietario:grupo] [archivo]
```

38.5.1 Ejemplos Prácticos Multi-SO

38.5.2 Ejemplo 1: Cambiar propietario

38.5.2.1 Linux

① Requiere permisos de administrador (sudo)
② Solo el propietario o root puede cambiar el dueño

38.5.2.2 macOS

① Funciona igual que Linux

38.5.2.3 Windows

① Requiere WSL

38.5.3 Ejemplo 2: Cambiar propietario y grupo

38.5.3.1 Linux

① Sintaxis: `usuario:grupo`

Listing 38.15 BASH

```
$ ls -l archivo.txt
-rw-r--r-- 1 usuario grupo 1024 Jan 29 10:30 archivo.txt

$ sudo chown otro-usuario archivo.txt
$ ls -l archivo.txt
-rw-r--r-- 1 otro-usuario grupo 1024 Jan 29 10:30 archivo.txt
                                                                    ②
                                                                    ①
```

Listing 38.16 BASH

```
$ sudo chown otro-usuario archivo.txt
$ ls -l archivo.txt
-rw-r--r-- 1 otro-usuario staff 1024 Jan 29 10:30 archivo.txt
                                                                    ①
```

38.5.3.2 macOS

① Funciona igual

38.5.3.3 Windows

① Usa WSL

38.5.4 Ejemplo 3: Cambiar solo el grupo (chgrp)

38.5.4.1 Linux

① `chgrp` es una abreviatura de `chown :grupo`

38.5.4.2 macOS

① Funciona igual

38.5.4.3 Windows

① Usa WSL

Listing 38.17 POWERSHELL

```
# Windows no tiene chown igual
# En WSL:
PS> wsl sudo chown otro-usuario archivo.txt
                                                                    ①
```

Listing 38.18 BASH

```
$ sudo chown usuario:nuevo-grupo archivo.txt
$ ls -l archivo.txt
-rw-r--r-- 1 usuario nuevo-grupo 1024 Jan 29 10:30 archivo.txt
                                                                    ①
```

38.5.5 Ejemplo 4: Cambiar recursivamente (-R)

38.5.5.1 Linux

① `-R` aplica a todos los archivos y subdirectorios

38.5.5.2 macOS

① Funciona igual

38.5.5.3 Windows

① Use WSL

38.6 Permisos Especiales

Existen 3 permisos especiales que modifican el comportamiento:

38.6.1 Setuid (4000)

- Cuando se ejecuta, se ejecuta con permisos del propietario
- Se muestra como `s` en lugar de `x` en el propietario
- Ejemplo: `/usr/bin/sudo` tiene setuid

Listing 38.19 BASH

```
$ sudo chown usuario:staff archivo.txt
                                                                    ①
```

Listing 38.20 POWERSHELL

```
PS> wsl sudo chown usuario:nuevo-grupo archivo.txt
                                                                    ①
```

38.6.2 Setgid (2000)

- Se ejecuta con permisos del grupo
- Se muestra como `s` en lugar de `x` en el grupo
- Útil para directorios compartidos

38.6.3 Sticky Bit (1000)

- En directorios, solo el propietario puede borrar archivos
- Se muestra como `t` al final
- Ejemplo: `/tmp` tiene sticky bit

38.6.4 Ejemplo: Setuid

38.6.4.1 Linux

① La `s` indica setuid (se ejecuta como root)

38.6.4.2 Sticky Bit en /tmp

① La `t` al final es sticky bit
② Solo el propietario de cada archivo puede borrarlo

38.6.4.3 Establecer Setuid

① Úsalo con cuidado (implicaciones de seguridad)

38.7 Resumen de Permisos

Listing 38.21 BASH

```
$ sudo chgrp nuevo-grupo archivo.txt
$ ls -l archivo.txt
-rw-r--r-- 1 usuario nuevo-grupo 1024 Jan 29 10:30 archivo.txt
```

①

Listing 38.22 BASH

```
$ sudo chgrp staff archivo.txt
```

①

Comando	Propósito	Ejemplo
`chmod 644 archivo`	Cambiar permisos (octal)	Archivos de datos
`chmod u+x archivo`	Cambiar permisos (simbólico)	Scripts ejecutables
`chmod -R 755 dir`	Cambiar recursivamente	Directorios
`chown usuario archivo`	Cambiar propietario	Traspasar propiedad
`chown usuario:grupo archivo`	Cambiar propietario y grupo	Configuración
`chgrp grupo archivo`	Cambiar solo grupo	Permisos de grupo
`ls -l`	Ver permisos	Auditoría
`stat archivo`	Ver permisos detallados	Análisis profundo

38.8 Permisos Comunes

Modo	Octal	Uso Común
rwxr-xr-x	755	Scripts ejecutables, directorios
rw-r–r–	644	Archivos de datos, documentos
rwx——	700	Archivos privados, directorios personales
rwxrwx—	770	Directorio compartido de grupo
rw-rw-r–	664	Archivo editable por grupo
rwxrwxrwx	777	Acceso total (riesgoso)
r–r–r–	444	Archivo de solo lectura

38.9 Ejercicios Prácticos

1. Crea un archivo y establece permisos `644`

Listing 38.23 POWERSHELL

```
PS> wsl sudo chgrp nuevo-grupo archivo.txt
```
①

Listing 38.24 BASH

```
$ sudo chown -R usuario:grupo ~/miproyecto
```
①

2. Crea un script y hazlo ejecutable con `chmod +x`
3. Crea un directorio y establece `755`
4. Usa notación simbólica para remover escritura (`g-w`)
5. Ver permisos con `ls -l` e interpreta cada columna
6. Cambia grupo de un archivo con `chgrp`

38.10 Referencias

Para más información sobre permisos:

- (IEEE 2018) - POSIX Permissions
- (project 2024) - Linux File Permissions Guide

Listing 38.25 BASH

```
$ sudo chown -R usuario:staff ~/myproject
```
①

Listing 38.26 POWERSHELL

```
PS> wsl sudo chown -R usuario:grupo ~/miproyecto
```
①

Listing 38.27 BASH

```
# Ver permiso especial
$ ls -l /usr/bin/sudo
-rwsr-xr-x 1 root root 1234 Jan 15 10:00 /usr/bin/sudo
```
①

Listing 38.28 BASH

```
$ ls -ld /tmp
drwxrwxrwt 1 root root 4096 Jan 29 10:45 /tmp
```
①
②

Listing 38.29 BASH

```
$ chmod u+s archivo      # Agregar setuid
$ chmod 4755 archivo     # Notación octal (4 = setuid)
```
①

39 Manipulación de Archivos y Directorios

Listing 39.1 QUARTO-TITLE-BLOCK

40 Manipulación de Archivos y Directorios

40.1 Introducción

Más allá de navegar y buscar, necesitas crear, copiar, mover y eliminar archivos. Estos comandos son las herramientas básicas para la administración de archivos en el sistema.

En este tema aprenderás:

- Comando `cp` - copiar archivos y directorios
- Comando `mv` - mover y renombrar
- Comando `rm` - eliminar archivos
- Comando `rmdir` - eliminar directorios
- Comando `mkdir` - crear directorios
- Comando `touch` - crear archivos vacíos
- Comando `ln` - crear enlaces
- Buenas prácticas de seguridad

40.2 El Comando `mkdir` - Crear Directorios

El comando `mkdir` (make directory) crea nuevos directorios.

Sintaxis:

Listing 40.1 BASH

```
mkdir [opciones] [directorio(s)]
```

40.2.1 Opciones principales:

- `-p` : Crear directorios padres si es necesario
- `-m` : Establecer permisos al crear

40.2.2 Ejemplos Prácticos Multi-SO

40.2.3 Ejemplo 1: Crear un directorio simple

40.2.3.1 Linux

Listing 40.2 BASH

```
$ mkdir Documentos
$ ls -d Documentos
Documentos
```
①

① Crea un directorio en la ubicación actual

40.2.3.2 macOS

Listing 40.3 BASH

```
$ mkdir Documents
$ ls -d Documents
Documents
```
①

① Funciona igual que Linux

40.2.3.3 Windows

Listing 40.4 POWERSHELL

```
PS> mkdir Documents
PS> Get-Item Documents

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         29/1/2024     10:30                Documents
```
①

① `mkdir` es un alias de `New-Item -ItemType Directory`

Listing 40.5 BASH

```
$ mkdir -p ~/Proyectos/Python/web-app/src
$ tree ~/Proyectos
Proyectos/
   Python/
       web-app/
           src/
```
①

40.2.4 Ejemplo 2: Crear estructura de directorios anidados (-p)

40.2.4.1 Linux

① `-p` crea todos los directorios intermedios

40.2.4.2 macOS

Listing 40.6 BASH

```
$ mkdir -p ~/Projects/Python/web-app/src
```
①

① Funciona igual

40.2.4.3 Windows

Listing 40.7 POWERSHELL

```
PS> mkdir -Path C:\Users\usuario\Projects\Python\web-app\src -Force
# El -Force es equivalente a -p
```
①

① PowerShell requiere `-Force` para crear padres

40.2.5 Ejemplo 3: Crear directorio con permisos específicos (-m)

40.2.5.1 Linux

① `-m 700` establece permisos solo-propietario

Listing 40.8 BASH

```
$ mkdir -m 700 privado
$ ls -ld privado
drwx------ 1 usuario usuario 4096 Jan 29 10:30 privado
                                                                    ①
```

Listing 40.9 BASH

```
$ mkdir -m 700 privado
                                                                    ①
```

40.2.5.2 macOS

① Funciona igual

40.2.5.3 Windows

Listing 40.10 POWERSHELL

```
# Windows no soporta -m, pero puedes hacerlo después:
PS> mkdir privado
PS> # No hay equivalente directo en PowerShell
                                                                    ①
```

① Los permisos de Windows funciona diferente (NTFS ACLs)

40.3 El Comando `touch` - Crear Archivos Vacíos

El comando `touch` crea archivos vacíos o actualiza la marca de tiempo de archivos existentes.

Sintaxis:

40.3.1 Ejemplos Prácticos Multi-SO

40.3.2 Ejemplo 1: Crear un archivo vacío

40.3.2.1 Linux

① Crea un archivo vacío de 0 bytes

Listing 40.11 BASH

```
touch [opciones] [archivo(s)]
```

Listing 40.12 BASH

```
$ touch README.md
$ ls -l README.md
-rw-r--r-- 1 usuario grupo 0 Jan 29 10:30 README.md
```
①

40.3.2.2 macOS

Listing 40.13 BASH

```
$ touch README.md
$ ls -l README.md
-rw-r--r-- 1 usuario staff 0 Jan 29 10:30 README.md
```
①

① Funciona igual

40.3.2.3 Windows

① PowerShell usa `New-Item`

40.3.3 Ejemplo 2: Actualizar marca de tiempo de archivo existente

40.3.3.1 Linux

① Actualiza el timestamp sin modificar contenido

40.3.3.2 macOS

① Funciona igual

40.3.3.3 Windows

① PowerShell actualiza la fecha manualmente

Listing 40.14 POWERSHELL

```
# No existe touch, pero:
PS> New-Item -Path C:\Users\usuario -Name README.md -ItemType File
# o en WSL:
PS> wsl touch README.md
```

①

Listing 40.15 BASH

```
$ ls -l archivo.txt
-rw-r--r-- 1 usuario grupo 1024 Jan 25 10:00 archivo.txt

$ touch archivo.txt
$ ls -l archivo.txt
-rw-r--r-- 1 usuario grupo 1024 Jan 29 10:30 archivo.txt
```

①

40.4 El Comando `cp` - Copiar Archivos y Directorios

El comando `cp` (copy) copia archivos y directorios.

Sintaxis:

40.4.1 Opciones principales:

- `-r` o `-R` : Copiar recursivamente (directorios)
- `-i` : Preguntar antes de sobrescribir
- `-v` : Modo verboso (mostrar progreso)
- `-p` : Preservar permisos y timestamps

40.4.2 Ejemplos Prácticos Multi-SO

40.4.3 Ejemplo 1: Copiar un archivo simple

40.4.3.1 Linux

① Crea una copia exacta del archivo

40.4.3.2 macOS

① Funciona igual

Listing 40.16 BASH

```
$ touch archivo.txt
                                                                    ①
```

Listing 40.17 POWERSHELL

```
PS> (Get-Item archivo.txt).LastWriteTime = Get-Date
                                                                    ①
```

40.4.3.3 Windows

① `cp` es un alias de `Copy-Item`

40.4.4 Ejemplo 2: Copiar a un directorio diferente

40.4.4.1 Linux

① Si el destino es un directorio, copia el archivo dentro

40.4.4.2 macOS

① Funciona igual

40.4.4.3 Windows

① Igual funcionalidad

40.4.5 Ejemplo 3: Copiar directorios recursivamente (-r)

40.4.5.1 Linux

① `-r` copia recursivamente todo el directorio

40.4.5.2 macOS

① Funciona igual

40.4.5.3 Windows

① `-Recurse` es el equivalente a `-r`

Listing 40.18 BASH

```
cp [opciones] [origen] [destino]
```

Listing 40.19 BASH

```
$ cp archivo.txt archivo-copia.txt
$ ls -l archivo*
-rw-r--r-- 1 usuario grupo 1024 Jan 29 10:30 archivo-copia.txt
-rw-r--r-- 1 usuario grupo 1024 Jan 25 10:00 archivo.txt
```

①

40.4.6 Ejemplo 4: Copiar con confirmación (-i) y preservación de atributos (-p)

40.4.6.1 Linux

① `-i` pregunta antes de sobrescribir
② `-p` preserva permisos y timestamps originales

40.4.6.2 macOS

① Funciona igual

40.4.6.3 Windows

① PowerShell no preserva permisos igual (usa NTFS ACLs)

40.5 El Comando `mv` - Mover y Renombrar

El comando `mv` (move) mueve o renombra archivos y directorios.

Sintaxis:

40.5.1 Ejemplos Prácticos Multi-SO

40.5.2 Ejemplo 1: Renombrar un archivo

40.5.2.1 Linux

① Renombra el archivo (mismo directorio)

Listing 40.20 BASH

```
$ cp archivo.txt archivo-copia.txt
                                                                    ①
```

Listing 40.21 POWERSHELL

```
PS> Copy-Item archivo.txt archivo-copia.txt
# o
PS> cp archivo.txt archivo-copia.txt
                                                                    ①
```

40.5.2.2 macOS

① Funciona igual

40.5.2.3 Windows

① `mv` es un alias de `Move-Item`

40.5.3 Ejemplo 2: Mover archivo a otro directorio

40.5.3.1 Linux

① Mueve el archivo, no copia

40.5.3.2 macOS

① Funciona igual

40.5.3.3 Windows

① Funciona igual

40.5.4 Ejemplo 3: Mover y renombrar simultáneamente

40.5.4.1 Linux

① Mueve y renombra en una operación

Listing 40.22 BASH

```
$ cp archivo.txt ~/Documentos/
$ ls ~/Documentos/archivo.txt
/home/usuario/Documentos/archivo.txt
```
①

Listing 40.23 BASH

```
$ cp archivo.txt ~/Documents/
```
①

40.5.4.2 macOS

① Funciona igual

40.5.4.3 Windows

① Mismo comportamiento

40.6 El Comando `rm` - Eliminar Archivos

El comando `rm` (remove) elimina archivos. **ADVERTENCIA: No hay "papelera de reciclaje" - es permanente.**

Sintaxis:

40.6.1 Opciones principales:

- `-i` : Preguntar antes de eliminar (seguro)
- `-f` : Forzar sin preguntar (peligroso)
- `-r` : Recursivo (eliminar directorios)
- `-v` : Verboso (mostrar qué se elimina)

40.6.2 Ejemplos Prácticos Multi-SO

40.6.3 Ejemplo 1: Eliminar un archivo (seguro)

40.6.3.1 Linux

Listing 40.24 POWERSHELL

```
PS> Copy-Item archivo.txt C:\Users\usuario\Documents\
                                                                    (1)
```

Listing 40.25 BASH

```
$ cp -r Proyectos Proyectos-copia
$ ls -ld Proyectos*
drwxr-xr-x 5 usuario grupo 4096 Jan 29 10:30 Proyectos
drwxr-xr-x 5 usuario grupo 4096 Jan 29 10:30 Proyectos-copia
                                                                    (1)
```

(1) `-i` pregunta antes de eliminar
(2) **Siempre usa `-i` hasta que te sientas seguro**

40.6.3.2 macOS

(1) Funciona igual

40.6.3.3 Windows

(1) `-Confirm` pregunta antes de eliminar

40.6.4 Ejemplo 2: Eliminar sin preguntar (peligroso)

40.6.4.1 Linux

(1) **PELIGRO: No hay forma de recuperar**

40.6.4.2 macOS

(1) Funcionamiento idéntico

40.6.4.3 Windows

(1) `-Force` elimina sin preguntar

Listing 40.26 BASH

```
$ cp -r Proyectos Proyectos-copia
                                                                    ①
```

Listing 40.27 POWERSHELL

```
PS> Copy-Item -Path Proyectos -Destination Proyectos-copia -Recurse
                                                                    ①
```

40.6.5 Ejemplo 3: Eliminar directorios recursivamente (-r)

40.6.5.1 Linux

① `-r` elimina recursivamente
② `-i` pregunta para cada archivo

40.6.5.2 macOS

① Funciona igual

40.6.5.3 Windows

① `-Recurse` elimina directorios y contenido

40.7 El Comando `rmdir` - Eliminar Directorios Vacíos

El comando `rmdir` solo elimina directorios vacíos, es más seguro que `rm -r`.

Sintaxis:

40.7.1 Ejemplos Prácticos Multi-SO

40.7.2 Ejemplo 1: Eliminar directorio vacío

40.7.2.1 Linux

① Solo funciona si está completamente vacío

Listing 40.28 BASH

```
$ cp -ip archivo.txt archivo-copia.txt
cp: overwrite 'archivo-copia.txt'? y
$ ls -l archivo-copia.txt
-rw-r--r-- 1 usuario grupo 1024 Jan 25 10:00 archivo-copia.txt
```

① ②

Listing 40.29 BASH

```
$ cp -ip archivo.txt archivo-copia.txt
```

①

40.7.2.2 macOS

① Funciona igual

40.7.2.3 Windows

① Funciona igual

40.8 El Comando `ln` - Crear Enlaces

El comando `ln` (link) crea enlaces entre archivos. Existen dos tipos:

40.8.1 Enlaces simbólicos (soft links):

- Apuntan a otro archivo por nombre
- Pueden cruzar sistemas de archivos
- Se rompen si el archivo original se elimina

40.8.2 Enlaces duros (hard links):

- Apuntan directamente al contenido del archivo
- No pueden cruzar sistemas de archivos
- Persisten aunque se elimine el original

Listing 40.30 POWERSHELL

```
PS> Copy-Item archivo.txt archivo-copia.txt -Confirm
                                                    ①
```

Listing 40.31 BASH

```
mv [opciones] [origen] [destino]
```

40.8.3 Ejemplos Prácticos Multi-SO

40.8.4 Ejemplo 1: Crear enlace simbólico (-s)

40.8.4.1 Linux

① `-s` crea enlace simbólico
② La `l` al inicio indica que es enlace

40.8.4.2 macOS

① Funciona igual

40.8.4.3 Windows

① PowerShell requiere permisos de administrador

40.8.5 Ejemplo 2: Crear enlace duro (sin -s)

40.8.5.1 Linux

① Sin `-s`, crea enlace duro
② Ambos tienen el mismo inodo (123456)

40.8.5.2 macOS

① Funciona igual

40.8.5.3 Windows

① NTFS tiene junctions que son similares

Listing 40.32 BASH

```
$ mv archivo-viejo.txt archivo-nuevo.txt
$ ls archivo*
archivo-nuevo.txt
```

①

Listing 40.33 BASH

```
$ mv archivo-viejo.txt archivo-nuevo.txt
```

①

40.9 Resumen de Manipulación de Archivos

Comando	Propósito	Ejemplo
`mkdir -p dir/subdir`	Crear directorios	`mkdir -p ~/Proyectos/web`
`touch archivo.txt`	Crear archivo vacío	`touch README.md`
`cp archivo.txt copia.txt`	Copiar archivo	`cp config.txt config.bak`
`cp -r dir dir-copia`	Copiar directorio	`cp -r src src-backup`
`mv archivo.txt nuevo.txt`	Renombrar	`mv archivo.txt LEEME.txt`
`mv archivo.txt dir/`	Mover	`mv archivo.txt ~/Documentos/`
`rm archivo.txt`	Eliminar archivo	`rm archivo.txt`
`rm -r directorio`	Eliminar directorio	`rm -r directorio/`
`rmdir dir-vacio`	Eliminar dir vacío	`rmdir dir-vacio`
`ln -s archivo enlace`	Enlace simbólico	`ln -s config config.link`

40.10 Mejores Prácticas de Seguridad

1. **Siempre usa `-i` con `rm` hasta que tengas experiencia**
2. **Haz copias antes de operaciones peligrosas**
3. **Usa `rsync` para copias importantes**
4. **Verifica antes de eliminar recursivamente**
5. **Usa control de versiones para código**

40.11 Ejercicios Prácticos

1. Crea estructura: `Proyecto/src/main` usando `mkdir -p`
2. Crea 3 archivos vacíos con `touch`

Listing 40.34 POWERSHELL

```
PS> Rename-Item archivo-viejo.txt archivo-nuevo.txt
# o
PS> mv archivo-viejo.txt archivo-nuevo.txt
```

①

Listing 40.35 BASH

```
$ mv archivo.txt ~/Documentos/
$ ls ~/Documentos/archivo.txt
/home/usuario/Documentos/archivo.txt
```

①

7. Elimina directorio vacío con `rmdir`

40.12 Referencias

Para más información sobre manipulación de archivos:

- (IEEE 2018) - POSIX File Operations
- (F. S. Foundation 2023a) - GNU Bash Manual
- (G. Project 2024) - GNU Coreutils Manual

Listing 40.36 BASH

```
$ mv archivo.txt ~/Documents/
```
①

Listing 40.37 POWERSHELL

```
PS> Move-Item archivo.txt C:\Users\usuario\Documents\
```
①

Listing 40.38 BASH

```
$ mv archivo.txt ~/Documentos/archivo-nuevo.txt
```
①

Listing 40.39 BASH

```
$ mv archivo.txt ~/Documents/archivo-nuevo.txt
```
①

Listing 40.40 POWERSHELL

```
PS> Move-Item archivo.txt C:\Users\usuario\Documents\archivo-nuevo.txt
```
①

Listing 40.41 BASH

```
rm [opciones] [archivo(s)]
```

Listing 40.42 BASH

```
$ rm -i archivo.txt
remove archivo.txt? y
```
①
②

Listing 40.43 BASH

```
$ rm -i archivo.txt
remove archivo.txt? y
```
①

Listing 40.44 POWERSHELL

```
PS> Remove-Item archivo.txt -Confirm
```
①

Listing 40.45 BASH

```
$ rm archivo.txt
# Sin confirmación, archivo eliminado para siempre
```
①

Listing 40.46 BASH

```
$ rm archivo.txt
```
①

Listing 40.47 POWERSHELL

```
PS> Remove-Item archivo.txt -Force
```
①

Listing 40.48 BASH

```
$ rm -ri directorio
remove directorio? y
remove directorio/archivo1.txt? y
remove directorio/archivo2.txt? y
```
①
②

Listing 40.49 BASH

```
$ rm -ri directorio
```
①

Listing 40.50 POWERSHELL

```
PS> Remove-Item -Path directorio -Recurse -Confirm
```
①

Listing 40.51 BASH

```
rmdir [directorio]
```

Listing 40.52 BASH

```
$ rmdir directorio-vacio
$ ls directorio-vacio
ls: cannot access 'directorio-vacio': No such file or directory
```
①

Listing 40.53 BASH

```
$ rmdir directorio-vacio
```
①

Listing 40.54 POWERSHELL

```
PS> Remove-Item -Path directorio-vacio -Confirm
```
①

Listing 40.55 BASH

```
$ ln -s archivo.txt enlace.txt
$ ls -l enlace.txt
lrwxrwxrwx 1 usuario grupo 10 Jan 29 10:30 enlace.txt -> archivo.txt
```
①
②

Listing 40.56 BASH

```
$ ln -s archivo.txt enlace.txt
```
①

Listing 40.57 POWERSHELL

```
# En WSL:
PS> wsl ln -s archivo.txt enlace.txt

# En PowerShell (Windows 10+):
PS> New-Item -ItemType SymbolicLink -Path enlace.txt -Target archivo.txt
```
①

Listing 40.58 BASH

```
$ ln archivo.txt enlace-duro.txt
$ ls -i archivo.txt enlace-duro.txt
123456 archivo.txt
123456 enlace-duro.txt
```
①
②

Listing 40.59 BASH

```
$ ln archivo.txt enlace-duro.txt
```
①

Listing 40.60 POWERSHELL

```
# No soporta hard links igual que Unix
# En WSL:
PS> wsl ln archivo.txt enlace-duro.txt
```
①

Listing 40.61 BASH

```
rm -i archivo.txt  # Pregunta antes de eliminar
```

Listing 40.62 BASH

```
cp archivo.txt archivo.txt.backup
# ... hacer cambios ...
```

Listing 40.63 BASH

```
rsync -av origen/ destino/
```

Listing 40.64 BASH

```
ls -lR directorio/  # Ver contenido primero
rm -ri directorio   # Luego eliminar
```

Listing 40.65 BASH

```
git commit -m "Backup antes de cambios"
# ... hacer cambios ...
```

41 Unidad 3: Recursos

Listing 41.1 `QUARTO-TITLE-BLOCK`

42 Recursos De La Unidad 3

Recurso	Enlace
Presentacion (Reveal.js)	Unidad 3
Practica	Practica Unidad 3
Laboratorio	(Pendiente)

Part V

Unidad 4: Docker y Containerización

43 Docker

Figure 43.1: Docker

Docker es una plataforma que permite desarrollar, enviar y ejecutar aplicaciones en contenedores. Un **contenedor** es una instancia ejecutable de una **imagen**, que es una especie de **plantilla** que contiene todo lo necesario para ejecutar una aplicación.

Haciendo una analogia con los contenedores de transporte, una **imagen** es el **plano/plantilla** (lo que se puede replicar), y el **contenedor** es la **instancia en ejecucion** (lo que efectivamente se "mueve" y corre).

Docker resuelve un problema principal en el desarrollo de software: la **portabilidad**. Al empaquetar una aplicación y sus dependencias en un contenedor, se garantiza que la aplicación se ejecute de manera **consistente** en diferentes entornos.

En esta lección, aprenderemos a crear y ejecutar contenedores Docker, y a utilizarlos para ejecutar aplicaciones de manera aislada y portátil.

Con docker se acaba la frase típica de los desarrolladores **En mi máquina funciona**. Con Docker, puedes estar seguro de que tu aplicación funcionará de la misma manera en cualquier entorno.

Una imagen Docker es una plantilla inmutable que contiene un conjunto de instrucciones para crear un contenedor Docker. Las imágenes son portátiles y pueden ser compartidas, almacenadas y actualizadas.

> **Tip**
>
> Las imágenes Docker son inmutables, lo que significa que no se pueden modificar una vez creadas. Si se realizan cambios en una imagen, se debe crear una nueva versión de la imagen.

Un contenedor Docker es una instancia ejecutable de una imagen Docker. Se ejecuta de manera aislada y contiene todo lo necesario para ejecutar la aplicación, incluyendo el código, las dependencias, el entorno de ejecución, las bibliotecas y los archivos de configuración.

> **Tip**
>
> Un contenedor aisla la aplicación de su entorno, lo que garantiza que la aplicación se ejecute de manera consistente en diferentes entornos.

43.1 Ejemplos

Descargar una imagen:

Listing 43.1 BASH

```
$ docker pull docker/getting-started                                    ①
```

① **docker pull** descarga una imagen desde un registro (por defecto, Docker Hub).

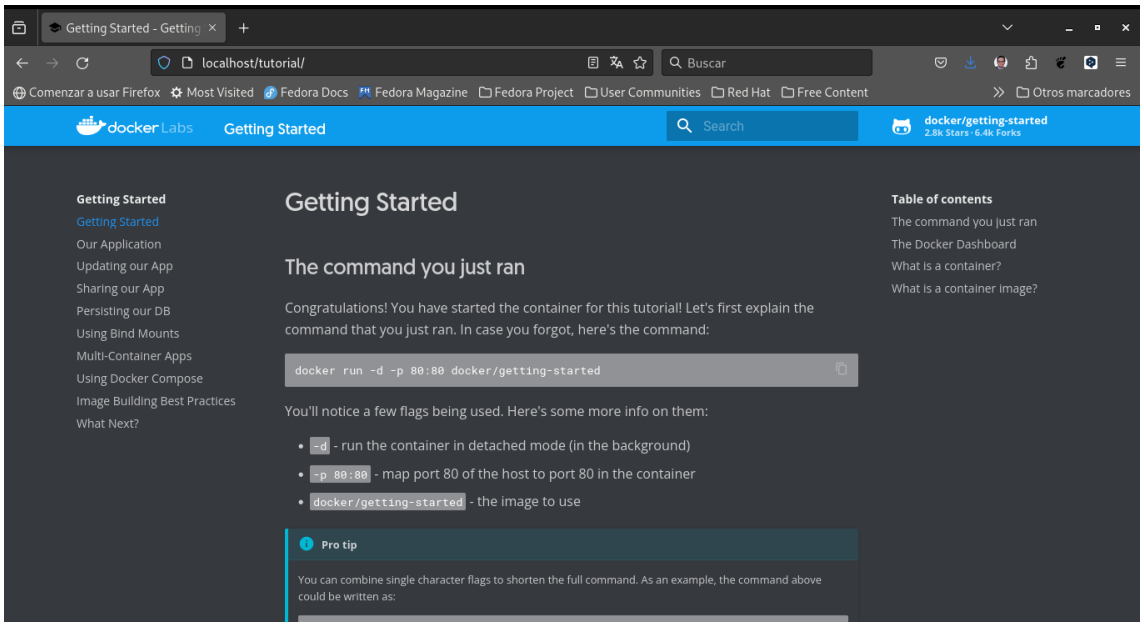
```
TERMINAL
statick  main U:11 ?:7  ~/workspaces/curso_docker/book  docker pull docker/getting-started
Using default tag: latest
latest: Pulling from docker/getting-started
Digest: sha256:d79336f4812b6547a53e735480dde67f8f8f7071b414fbd9297609ffb989abc1
Status: Image is up to date for docker/getting-started:latest
docker.io/docker/getting-started:latest
```

Este comando descarga la imagen **getting-started** desde el registro público de Docker.

Listing 43.2 BASH

```
$ docker run -d -p 80:80 docker/getting-started                    (1)
```

Correr un contenedor en el puerto 80:



1. **docker run** crea y ejecuta un contenedor; `-d` lo deja en segundo plano y `-p 80:80` publica el puerto del contenedor en el host.

Este comando ejecuta un contenedor en segundo plano y publica el puerto 80 del contenedor en el puerto 80 del host.

> **Tip**
>
> El comando -p se utiliza para mapear los puertos de la máquina host al contenedor, muchas personas consideran que significa "puerto". Sin embargo en realidad significa "publicar" o "publicar puerto".

43.2 Comandos básicos de Docker:

① **docker pull** descarga una imagen desde un registro (por defecto, Docker Hub).
② **docker images** lista las imágenes descargadas en el host.
③ **docker ps** lista los contenedores en ejecución.
④ **docker ps -a** lista todos los contenedores (incluyendo detenidos).
⑤ **docker run** crea y ejecuta un contenedor a partir de una imagen.
⑥ **docker stop** detiene un contenedor.
⑦ **docker start** inicia un contenedor detenido.

Listing 43.3 BASH

```
$ docker pull <IMAGE_NAME:TAG>                                    (1)
$ docker images                                                   (2)
$ docker ps                                                       (3)
$ docker ps -a                                                    (4)
$ docker run -d -p <HOST_PORT>:<CONTAINER_PORT> <IMAGE_NAME:TAG>  (5)
$ docker stop <CONTAINER_ID>                                      (6)
$ docker start <CONTAINER_ID>                                     (7)
$ docker rm <CONTAINER_ID>                                        (8)
$ docker rmi <IMAGE_NAME:TAG>                                     (9)
$ docker inspect <CONTAINER_ID>                                   (10)
$ docker logs <CONTAINER_ID>                                      (11)
$ docker compose up -d                                            (12)
```

(8) **docker rm** elimina un contenedor (debe estar detenido).
(9) **docker rmi** elimina una imagen.
(10) **docker inspect** muestra metadatos detallados del contenedor/imagen (IPs, mounts, etc.).
(11) **docker logs** muestra los logs (stdout/stderr) del contenedor.
(12) **docker compose up -d** levanta un stack multi-contenedor con Compose v2.

> **Note**
>
> Si tu entorno aún usa el binario antiguo, el equivalente es `docker-compose up -d`.

43.3 Práctica:

- Descarga la imagen de Nginx desde el registro público.
- Crea y ejecuta un contenedor de Nginx en el puerto 8080.
- Detén y elimina el contenedor creado
- Utiliza los comandos para detener y eliminar un contenedor.

Resolución de la Actividad Práctica

Abre tu terminal o línea de comandos y ejecuta los siguientes pasos.

Paso 1: Descarga la imagen de Nginx

Listing 43.4 BASH

```
$ docker pull nginx                                                    ①
```

① **docker pull nginx** descarga la imagen oficial de Nginx.

Paso 2: Crea y ejecuta el contenedor (puerto 8080)

Listing 43.5 BASH

```
$ docker run -d -p 8080:80 nginx                                       ①
```

① **docker run -d -p 8080:80** ejecuta el contenedor en segundo plano y publica el puerto 80 del contenedor en el 8080 del host.

Paso 3: Verifica que el contenedor esté en ejecución

Listing 43.6 BASH

```
$ docker ps                                                            ①
```

① **docker ps** lista contenedores en ejecución y muestra el mapeo de puertos.

Paso 4: Detén el contenedor

Listing 43.7 BASH

```
$ docker stop <CONTAINER_ID>                                           ①
```

① **docker stop** detiene el contenedor por ID.

Paso 5: Elimina el contenedor

Listing 43.8 BASH

```
$ docker rm <CONTAINER_ID>                                             ①
```

① **docker rm** elimina el contenedor (debe estar detenido).

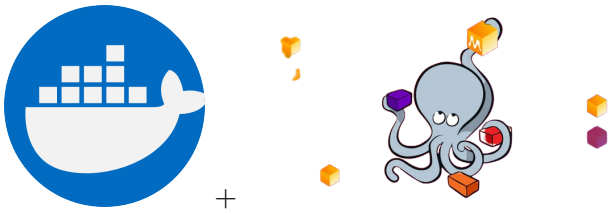
Tip

Combina los comandos docker ps, docker stop, y docker rm para gestionar contenedores eficientemente.
¡Practica estos pasos para familiarizarte con el ciclo de vida de los contenedores Docker!

44 Conclusiones

En esta lección aprendimos los conceptos base de Docker (imágenes y contenedores) y el ciclo de vida de un contenedor: descargar imágenes, ejecutar servicios, inspeccionar estado y logs, y limpiar recursos. Con esto, ya puedes ejecutar aplicaciones de forma portable y aislada.

45 Dockerfile y Docker Compose



45.1 Introducción

Dockerfile y Docker Compose son herramientas esenciales para la construcción y gestión de aplicaciones Docker. Un Dockerfile es un archivo de texto que define cómo se construirá una imagen Docker, mientras que Docker Compose es una herramienta para definir y gestionar aplicaciones Docker con múltiples contenedores. En esta lección, aprenderemos cómo usar Dockerfile y Docker Compose para personalizar imágenes Docker y orquestar servicios en un entorno multi-contenedor.

A continuación veremos algunos conceptos básicos sobre Dockerfile y Docker Compose.

45.1.1 Dockerfile

Un Dockerfile es un archivo de texto que contiene una serie de instrucciones para construir una imagen Docker. Estas instrucciones incluyen la configuración del sistema operativo base, la instalación de paquetes y dependencias, la configuración de variables de entorno y la definición de comandos para ejecutar la aplicación.

45.1.2 Docker Compose

Docker Compose es una herramienta para definir y gestionar aplicaciones Docker con múltiples contenedores. Permite definir servicios, redes y volúmenes en un archivo YAML y orquestar la ejecución de los contenedores en un entorno de desarrollo o producción.

45.2 Ejemplos:

En este ejemplo vamos a dockerizar una aplicación nodejs con un servidor sencillo en express.

Empezamos por el código de nuestra aplicación:

Para ello creamos un nuevo proyecto nodejs con el siguiente comando:

Listing 45.1 BASH

```
npm init -y
```

Instalamos el paquete express con el siguiente comando:

Listing 45.2 BASH

```
npm install express
```

Creamos los siguientes archivos:

- server.js
- package.json
- Dockerfile
- docker-compose.yml

45.2.1 server.js

Listing 45.3 JAVASCRIPT

```
const express = require('express');
const app = express();
const port = 3000;

app.get('/', (req, res) => {
  res.send('Hello, World!');
});

app.listen(port, () => {
  console.log(`Server running at http://localhost:${port}/`);
});
```

45.2.2 Dockerfile

45.2.3 docker-compose.yml

En este ejemplo, el Dockerfile define una imagen Docker para una aplicación Node.js. El archivo docker-compose.yml define un servicio llamado myapp que utiliza el Dockerfile.nodejs para construir la imagen y expone el puerto 3000 para acceder a la aplicación.

Listing 45.4 DOCKERFILE

```
# Use the official Node.js 14 image
FROM node:14

# Set the working directory in the container
WORKDIR /app

# Copy the dependencies file to the working directory
COPY package.json .

# Install dependencies
RUN npm install

# Copy the app code to the working directory
COPY . .

# Expose the port the app runs on
EXPOSE 3000

# Serve the app
CMD ["node", "server.js"]
```

> **Tip**
>
> El puerto del lado izquierdo de los 2 puntos en el archivo docker-compose.yml es el puerto en el host, mientras que el puerto del lado derecho es el puerto en el contenedor.

Para probar nuestro ejemplo, ejecutamos el siguiente comando:

Esto construirá la imagen Docker y ejecutará el contenedor en segundo plano. Podemos acceder a la aplicación en `http://localhost:3000`.

Para verificar que el contenedor está en ejecución, ejecutamos el siguiente comando:

Podemos utilizar una aplicación como Thunder Client o Postman para enviar una solicitud HTTP a la aplicación y ver la respuesta.

Para detener y eliminar el contenedor, ejecutamos el siguiente comando:

> **Tip**
>
> Recuerda: La imagen que se crea a partir del Dockerfile se almacena en el caché local de Docker. Si realizas cambios en el Dockerfile y deseas reconstruir la imagen, puedes usar el comando

Listing 45.5 YAML

```yaml
services:
  myapp:
    build:
      context: .
      dockerfile: Dockerfile
    ports:

      - "3000:3000"
    volumes:

      - .:/app
```

Listing 45.6 BASH

```bash
docker-compose up -d
```

Listing 45.9 BASH

```bash
docker-compose up --build
```

45.3 Práctica:

- Crea un Dockerfile para una aplicación Python simple.
- Configura un archivo docker-compose.yml para ejecutar la aplicación.

Resolución de la Actividad Práctica

Ejemplo de aplicación Python simple:

Ejemplo de Dockerfile:

Ejemplo de docker-compose.yml:

Listing 45.7 BASH

```
docker ps
```

Listing 45.8 BASH

```
docker-compose down
```

Listing 45.10 PYTHON

```
# app.py
print("Hello, World!")
```

Listing 45.11 DOCKERFILE

```
FROM python:3.12
WORKDIR /app
COPY . .
CMD ["python", "app.py"]
```

Listing 45.12 YAML

```
services:
  myapp:
    build:
      context: .
      dockerfile: Dockerfile.python
    image: my-python-app
```

46 Conclusión

En esta lección, aprendimos cómo usar Dockerfile y Docker Compose para construir y gestionar aplicaciones Docker. Con Dockerfile, podemos personalizar imágenes Docker para nuestras aplicaciones, mientras que Docker Compose nos permite definir y orquestar servicios en un entorno multi-contenedor. Estas herramientas son esenciales para el desarrollo y despliegue de aplicaciones en contenedores Docker.

47 DevContainers

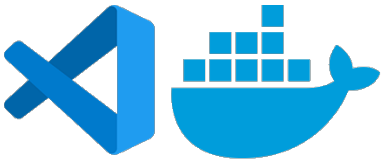


Figure 47.1: DevContainers

47.1 ¿Qué son los DevContainers?

Los DevContainers son entornos de desarrollo basados en contenedores Docker que permiten a los desarrolladores crear, compartir y ejecutar aplicaciones en un entorno aislado y portátil. Los DevContainers proporcionan un entorno de desarrollo consistente y reproducible, lo que garantiza que las aplicaciones se ejecuten de la misma manera en diferentes entornos.

Los DevContainers son una herramienta poderosa para el desarrollo de software, ya que permiten a los desarrolladores trabajar en un entorno aislado y preconfigurado, sin tener que preocuparse por la configuración del sistema operativo, las dependencias de software o las bibliotecas de terceros.

47.2 Instalación y Uso

Para utilizar DevContainers, es necesario tener instalado Docker en el sistema. Una vez instalado Docker, se puede instalar una extensión de DevContainers en el editor de código favorito, como Visual Studio Code, y utilizarla para crear, compartir y ejecutar DevContainers.

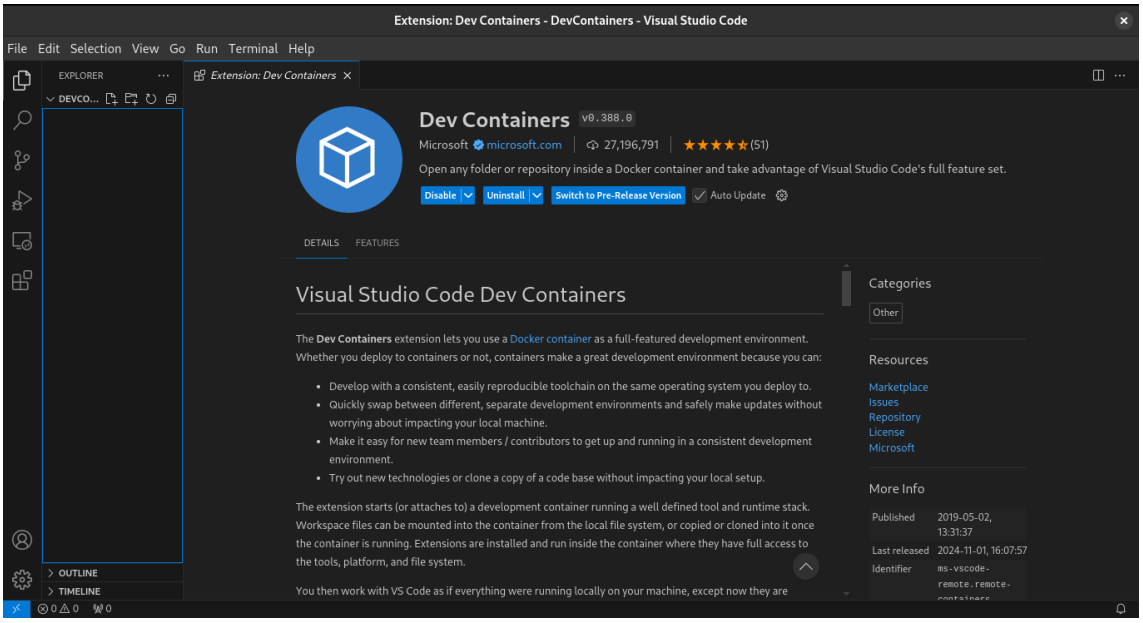


Figure 47.2: DevContainer en Visual Studio Code

47.3 Ejemplos:

En la parte inferior izquierda de Visual Studio Code existe un botón que hace referencia a los **DevContainers**, al hacer clic en este botón se abrirá un menú con las opciones para crear, abrir o configurar un DevContainer.

En este punto damos clic en **New DevContainer** y seleccionamos la opción **Python 3**. Esto creará un archivo **.devcontainer** con la configuración necesaria para ejecutar la aplicación en un contenedor Docker.

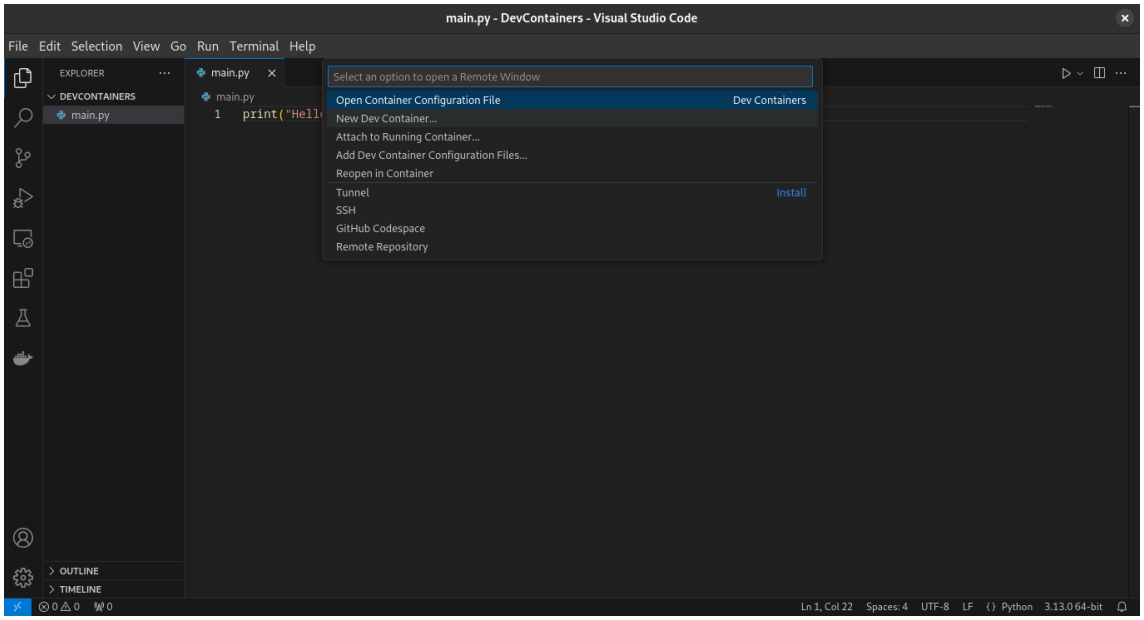


Figure 47.3: New DevContainer

En la imágen anterior podemos observar el menú de DevContainer, en esta sección es posible seleccionar **New DevContainer**. Al seleccionar esta opción se desplegará un menú con las opciones de configuración de DevContainer.

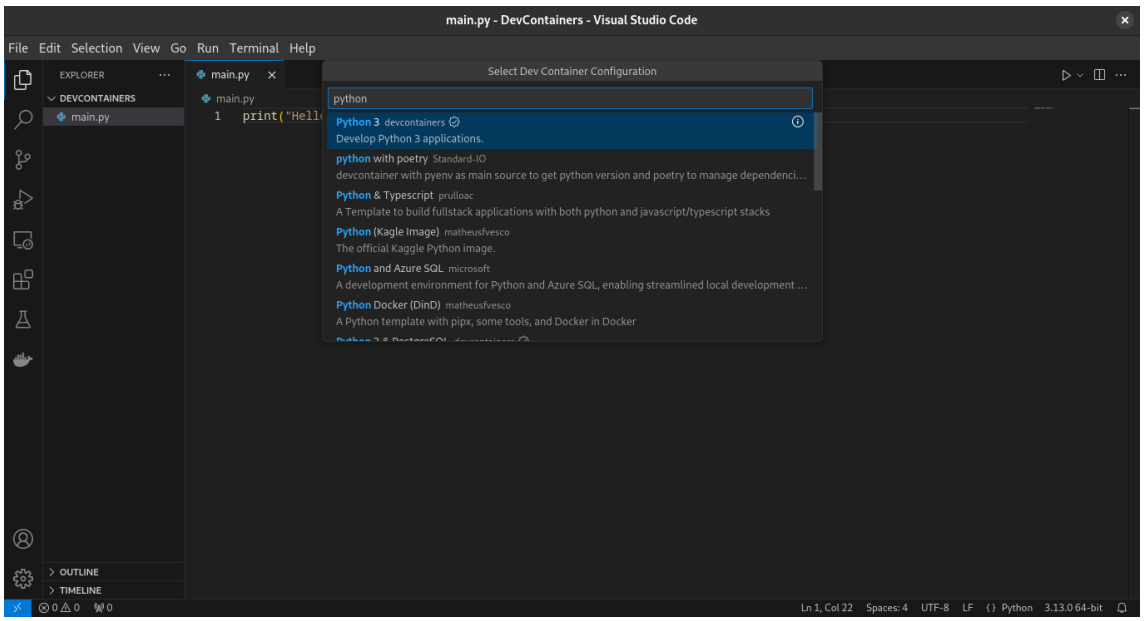


Figure 47.4: Python 3 DevContainer

En la imágen anterior se describe la búsqueda de diferentes plantillas, en este caso seleccionamos **Python 3**. Al seleccionar esta opción se creará un archivo **.devcontainer** con la configuración necesaria para ejecutar la aplicación en un contenedor Docker.

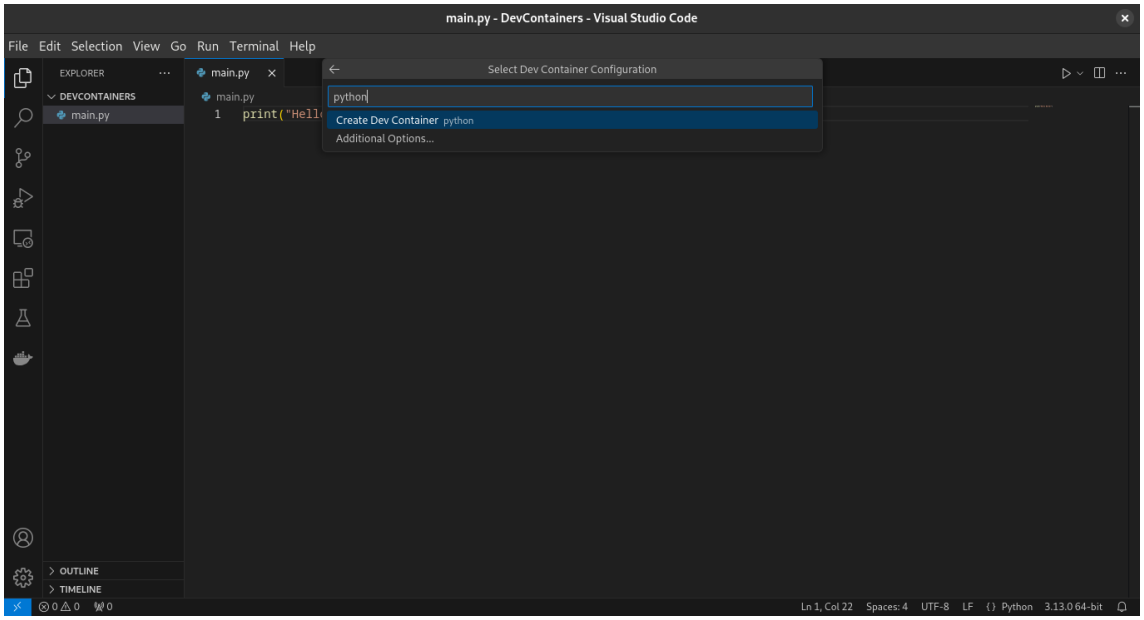


Figure 47.5: Create DevContainer

Finalmente seleccoinamos la opción **Create DevContainers** para crear el archivo **.devcontainer** con la configuración necesaria para ejecutar la aplicación en un contenedor Docker.

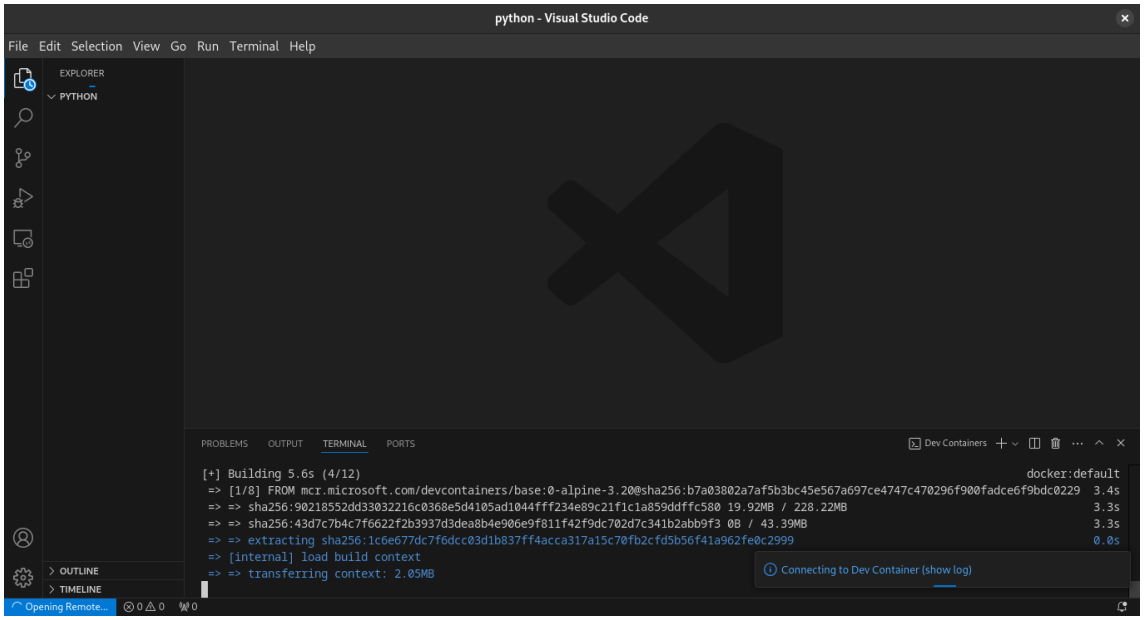


Figure 47.6: Create DevContainer

Ahora solo resta esperar como se observa en la imágen anterior la creación del **DevContainer**. Una vez finalizado el proceso, se abrirá una nueva ventana con el archivo **main.py** en el editor de código y se mostrará un mensaje en la parte inferior derecha indicando que se está construyendo el contenedor.

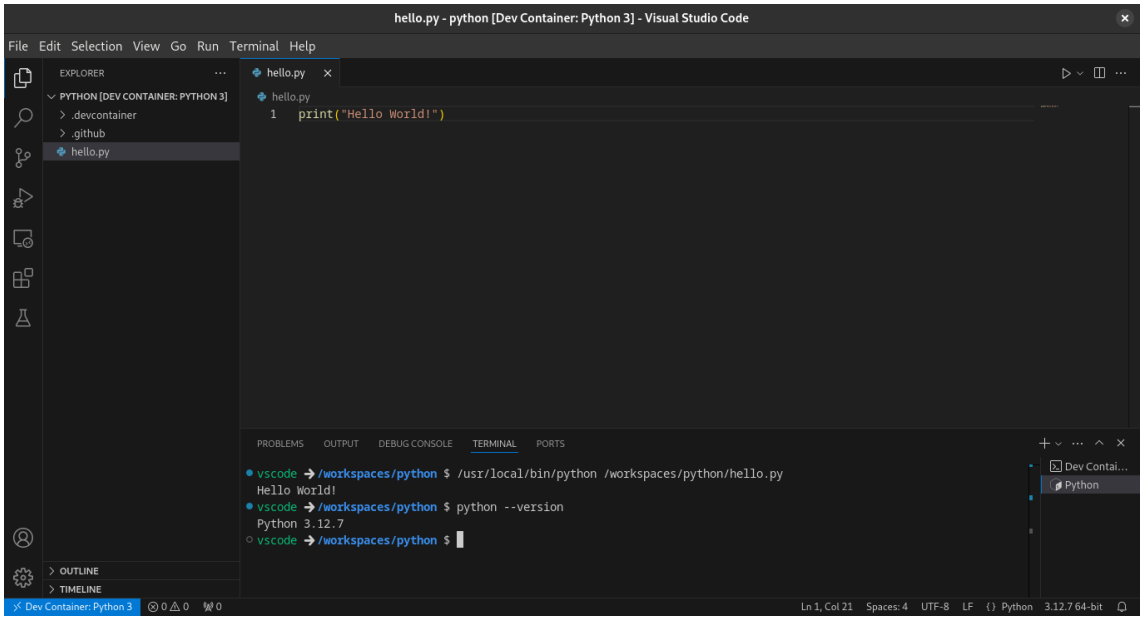


Figure 47.7: Python in DevContainer

Creamos una aplicación Hola Mundo en Python para ser ejecutada en un DevContainer:

Crear un archivo **main.py** con el siguiente código:

Listing 47.1 PYTHON

```
# main.py
print("Hola, Mundo!")
```

Una vez creado el **DevContainer** se mostrará un mensaje en la parte inferior derecha indicando que se está construyendo el contenedor. En este punto se puede ejecutar la aplicación en el contenedor Docker haciendo clic en el botón **Run** en la parte superior derecha.

Puedes verificar que la versión de python en el terminal del DevContainer creado es diferente a la del Sistema Operativo en el que te encuentres y la instalación global del sistema.

47.4 Práctica

- Crear un nuevo DevContainer con una plantilla en Python.
- Crear un archivo **main.py** con un código sencillo en Python.
- Ejecutar la aplicación en el DevContainer.

47.5 Conclusiones

Los DevContainers son una herramienta poderosa para el desarrollo de software, ya que permiten a los desarrolladores trabajar en un entorno aislado y preconfigurado, sin tener que preocuparse por la configuración del sistema operativo, las dependencias de software o las bibliotecas de terceros. Los DevContainers proporcionan un entorno de desarrollo consistente y reproducible, lo que garantiza que las aplicaciones se ejecuten de la misma manera en diferentes entornos.

48 Unidad 4: Recursos

Listing 48.1 `QUARTO-TITLE-BLOCK`

49 Recursos De La Unidad 4

Recurso	Enlace
Presentacion (Reveal.js)	Unidad 4
Practica	Practica Unidad 4
Laboratorio	(Pendiente)

Part VI

Unidad 5: Procesos y Servicios

50 Unidad 5.1: Procesos - ps, top, pgrep y kill

Listing 50.1 `QUARTO-TITLE-BLOCK`

Identificacion y control seguro de procesos

51 Unidad 5.1: Procesos - ps, top, pgrep y kill

51.1 Introduccion

En el brochure del curso, la Unidad 5 inicia con lo esencial para operar un servidor bajo presion: **ver procesos** (ps/top), **encontrarlos** (pgrep) y **controlarlos** (kill).

Esto se usa en situaciones reales como:

- "El servidor esta lento" (CPU/RAM/IO)
- "Un servicio no responde" (procesos colgados)
- "Necesito detener un proceso sin reiniciar el servidor"

Tiempo estimado: 60-90 minutos

51.2 Conceptos minimos

- **PID:** identificador del proceso.
- **CPU/MEM:** consumo aproximado (snapshot).
- **SIGTERM vs SIGKILL:** terminar ordenado vs forzado.

51.3 ps: snapshot reproducible

Listing 51.1 BASH

```
$ ps aux | head -n 6                                                    ①
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.2 168500 12120 ?        Ss   08:00   0:02 /sbin/init
root         725  0.1  0.9 982300 74200 ?        Ssl  08:01   0:10 /usr/bin/containerd
www-data    2459  2.4  0.7 210312 118004 ?       S    10:44   0:12 nginx: worker process
```

---

① **ps aux** lista procesos del sistema con usuario, CPU/MEM y comando

Ordenar por CPU y memoria:

Listing 51.2 BASH

```
$ ps aux --sort=-%cpu | head -n 6                                                    ①
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root      1821 15.2  1.8  933612 293020 ?      Ssl  08:10  18:22 /usr/bin/dockerd -H fd:/
mysql     1330  6.4  7.9 1928300 1301200 ?     Ssl  07:55  24:10 /usr/sbin/mysqld
```

① **ps aux –sort=-%cpu** entrega un top rapido por CPU (snapshot)

51.4 top: monitoreo en vivo

Listing 51.3 BASH

```
$ top                                                                                ①
```

① **top** abre un monitor interactivo para observar CPU, memoria y procesos

51.5 pgrep: encontrar PID por nombre/patron

Listing 51.4 BASH

```
$ pgrep -a nginx                                                                     ①
2458 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
2459 nginx: worker process
```

① **pgrep -a** imprime PID y el comando completo asociado (util antes de matar)

Listing 51.5 BASH

```
$ pgrep -a "python3"                                                    ①
8123 python3 app.py

$ kill -TERM 8123                                                       ②

$ ps -p 8123                                                            ③
  PID TTY          TIME CMD
 8123 ?        00:00:12 python3

$ kill -KILL 8123                                                       ④
                                                                        ①
```

51.6 kill: terminar un proceso (recomendado: SIGTERM primero)

① **pgrep -a** identifica el PID y el comando real
② **kill -TERM** pide terminacion ordenada (primera opcion)
③ **ps -p** verifica si el proceso sigue presente
④ **kill -KILL** fuerza terminacion (ultima instancia)

> ⚠ ADVERTENCIA CRITICA
>
> **Evita usar `kill -KILL` (SIGKILL) como primer paso.**
> **Lo que podria salir mal:** - Corrupcion de archivos temporales/locks. - Perdida de logs o buffers sin flush.
> **Como prevenirlo:** 1. Usa `SIGTERM` primero y espera unos segundos. 2. Verifica con `ps -p <PID>`. 3. Usa `SIGKILL` solo si el proceso no responde.

51.7 Ejemplos practicos multi-SO

51.7.1 Ejemplo 1: Ver procesos

51.7.1.1 Linux

Listing 51.6 BASH

```
$ ps aux | head -n 5                                                    ①
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.2 168500 12120 ?        Ss   08:00   0:02 /sbin/init
```

① **ps aux** muestra un snapshot de procesos del sistema

51.7.1.2 macOS

Listing 51.7 BASH

```
$ ps aux | head -n 5                                                    ①
USER               PID  %CPU %MEM      VSZ    RSS   TT  STAT STARTED      TIME COMMAND
usuario            9012  12.3  0.8  4123456 132000 s001  S+   9:10AM   0:30.12 -zsh
```

① **ps aux** funciona en macOS con columnas similares

51.7.1.3 Windows

Listing 51.8 POWERSHELL

```
PS> Get-Process | Select-Object -First 5 Name,Id,CPU,WS                 ①

Name      Id     CPU        WS
----      --     ---        --
pwsh      12456   12.10  182222848
```

① **Get-Process** lista procesos (equivalente conceptual a ps)

51.7.2 Ejemplo 2: Detener un proceso

51.7.2.1 Linux

Listing 51.9 BASH

```
$ pgrep -a nginx                                                        ①
2458 nginx: master process /usr/sbin/nginx

$ sudo kill -TERM 2458                                                  ②
                                                                        ①
```

① **pgrep -a** encuentra el proceso objetivo
② **kill -TERM** solicita el cierre ordenado

Listing 51.10 BASH

```
$ pgrep -a "Google Chrome"                                                    ①
9120 /Applications/Google Chrome.app/Contents/MacOS/Google Chrome

$ kill -TERM 9120                                                             ②
                                                                              ①
```

51.7.2.2 macOS

① **pgrep -a** identifica PID por nombre
② **kill -TERM** solicita terminacion

51.7.2.3 Windows

Listing 51.11 POWERSHELL

```
PS> Get-Process -Name notepad | Select-Object Name,Id                         ①

Name    Id
----    --
notepad 7332

PS> Stop-Process -Id 7332                                                     ②
                                                                              ①
```

① **Get-Process -Name** encuentra el proceso
② **Stop-Process** detiene el proceso

Aspecto	Linux	macOS	Windows
Listar procesos	`ps`	`ps`	`Get-Process`
Buscar PID	`pgrep -a`	`pgrep -a`	`Get-Process -Name`
Terminar proceso	`kill -TERM`	`kill -TERM`	`Stop-Process`

51.8 Mejores practicas

RECOMENDACION

Diagnostico rapido antes de actuar.
Casos de uso: - Servidor lento. - Servicio colgado.
Cuando aplicar: 1. Snapshot: `ps aux --sort=-%cpu` y `ps aux --sort=-%mem`. 2. Observacion: `top` por 30-60 segundos. 3. Accion: `kill -TERM` y verificacion.

51.9 Resumen

- `ps` sirve para evidencia reproducible.
- `top` sirve para observar en tiempo real.
- `pgrep` acelera la busqueda de PIDs.
- `kill` controla procesos (SIGTERM primero).

51.10 Referencias

- ps(1)
- top(1)
- pgrep(1)
- kill(1)

52 Unidad 5.2: systemd units y journalctl

Listing 52.1 `QUARTO-TITLE-BLOCK`

Gestion de servicios con systemctl y lectura de logs

53 Unidad 5.2: systemd units y journalctl

53.1 Introduccion

El brochure del curso indica que la Unidad 5 incluye **systemd units** y **journalctl**. Esto es el corazon de la operacion en Linux moderno: servicios como SSH, Nginx, Docker y bases de datos se controlan con systemd.

Tiempo estimado: 60-90 minutos

53.2 Conceptos clave

- **Unit:** recurso gestionado por systemd (ej: `nginx.service`, `ssh.service`).
- **Active vs enabled:**
  - active: corre ahora
  - enabled: arranca al boot

53.3 systemctl: controlar servicios

① **systemctl status ...** –**no-pager** muestra estado y si esta enabled
② **systemctl restart** reinicia el servicio (aplica cambios)
③ **systemctl enable** habilita arranque automatico
④ **systemctl disable** deshabilita arranque automatico

53.4 journalctl: logs del servicio

① **journalctl -u ...** –**since** trae logs recientes del servicio
② **journalctl -f** sigue logs en tiempo real (similar a tail -f)

Listing 53.1 BASH

```
$ systemctl status ssh --no-pager                                    ①
* ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled)
     Active: active (running) since Sun 2026-02-01 09:10:00 UTC; 2h 15min ago

$ sudo systemctl restart ssh                                         ②

$ sudo systemctl enable nginx                                        ③
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service -> /lib/systemd

$ sudo systemctl disable nginx                                       ④
Removed /etc/systemd/system/multi-user.target.wants/nginx.service.
                                                                     ①
```

Listing 53.2 BASH

```
$ journalctl -u ssh --since "30 min ago" --no-pager                  ①
Feb 01 11:12:42 server sshd[2222]: Accepted publickey for ubuntu from 10.0.0.12 port 5201

$ journalctl -u nginx -f                                             ②
Feb 01 11:20:03 server nginx[2458]: 2026/02/01 11:20:03 [notice] 2458#2458: signal proces
                                                                     ①
```

53.5 Ejemplos practicos multi-SO

53.5.1 Ejemplo 1: Ver estado de un servicio

53.5.1.1 Linux

Listing 53.3 BASH

```
$ systemctl status nginx --no-pager                                  ①
* nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled)
     Active: active (running) since Sun 2026-02-01 09:12:14 UTC; 2h 10min ago
```

① **systemctl status** muestra estado del servicio

53.5.1.2 macOS

① **launchctl list** lista servicios/agents de launchd (equivalente conceptual)

Listing 53.4 BASH

```
$ launchctl list | head -n 5                                                    ①
PID     Status  Label
  89    0       com.apple.audio.coreaudiod
  95    0       com.apple.WindowServer
```

53.5.1.3 Windows

Listing 53.5 POWERSHELL

```
PS> Get-Service -Name w32time | Select-Object Name,Status,StartType             ①

Name   Status  StartType
----   ------  ---------
w32time Running Automatic
```

① **Get-Service** consulta estado de un servicio

Aspecto	Linux (systemd)	macOS (launchd)	Windows
Estado servicio	`systemctl status`	`launchctl list`	`Get-Service`
Logs	`journalctl -u`	(logs del sistema)	Event Log

53.6 Mejores practicas

RECOMENDACION

Si un servicio falla, lee logs antes de reiniciar en bucle.
Casos de uso: - nginx/ssh/mysql no levantan.
Cuando aplicar: 1. `systemctl status <svc>` 2. `journalctl -u <svc> --since "30 min ago"` 3. Corregir configuracion y recien reiniciar.

53.7 Resumen

- `systemctl` controla servicios y su comportamiento al arranque.
- `journalctl` es el log principal para diagnosticar fallas.

53.8 Referencias

- systemctl(1)
- journalctl(1)

54 Unidad 5.3: cron/anacron y at

Listing 54.1 `QUARTO-TITLE-BLOCK`

Automatizacion de tareas recurrentes y ejecuciones puntuales

55 Unidad 5.3: cron/anacron y at

55.1 Introduccion

El brochure del curso incluye **cron/anacron y at** dentro de la Unidad 5. En administracion de servidores, estas herramientas permiten automatizar:

- backups
- limpieza (logs, temporales)
- reportes
- tareas de mantenimiento

Tiempo estimado: 60-90 minutos

55.2 cron: tareas recurrentes

Editar el crontab del usuario:

Listing 55.1 BASH

```
$ crontab -e                                                    ①
```

---

① **crontab -e** abre el crontab del usuario actual

Listar tareas del usuario:

Listing 55.2 BASH

```
$ crontab -l                                                    ①
30 2 * * * /usr/local/bin/backup.sh >> /var/log/backup.log 2>&1
```

---

① **crontab -l** lista tareas programadas del usuario

55.3 anacron: recuperar tareas si el equipo estuvo apagado

Listing 55.3 BASH

```
$ cat /etc/anacrontab                                                    ①
# /etc/anacrontab: configuration file for anacron
1   5   cron.daily   nice run-parts --report /etc/cron.daily
7   10  cron.weekly  nice run-parts --report /etc/cron.weekly
@monthly 15 cron.monthly nice run-parts --report /etc/cron.monthly
```

① **/etc/anacrontab** define tareas diarias/semanales/mensuales tolerantes a apagados

55.4 at: ejecutar una sola vez en el futuro

Listing 55.4 BASH

```
$ echo "systemctl restart nginx" | at 03:00                             ①
warning: commands will be executed using /bin/sh
job 12 at Sun Feb  1 03:00:00 2026

$ atq                                                                    ②
12  Sun Feb  1 03:00:00 2026 a usuario

$ atrm 12                                                                ③
                                                                         ①
```

① **at 03:00** programa una ejecucion unica a las 03:00
② **atq** lista jobs pendientes
③ **atrm** elimina un job programado

> ⚠ ADVERTENCIA CRITICA
>
> **Cron no tiene el mismo entorno que tu terminal.**
> **Lo que podria salir mal:** - El script falla por PATH distinto. - El job corre sin logs y no sabes por que fallo.
> **Como prevenirlo:** 1. Usa rutas absolutas a binarios y scripts. 2. Redirige stdout/stderr a un log (`>> ... 2>&1`). 3. Prueba el comando manualmente antes de programarlo.

55.5 Ejemplo practico multi-SO

55.5.1 Ejemplo 1: Ver tareas programadas

55.5.1.1 Linux

Listing 55.5 BASH

```
$ crontab -l                                                                    ①
30 2 * * * /usr/local/bin/backup.sh >> /var/log/backup.log 2>&1
```

① **crontab -l** muestra las tareas del usuario

55.5.1.2 macOS

Listing 55.6 BASH

```
$ ls -1 /Library/LaunchDaemons | head -n 5                                      ①
com.apple.alf.agent.plist
com.apple.atrun.plist
com.apple.backupd-auto.plist
```

① **LaunchDaemons** es el mecanismo recomendado en macOS (equivalente conceptual)

55.5.1.3 Windows

Listing 55.7 POWERSHELL

```
PS> Get-ScheduledTask | Select-Object -First 5 TaskName,State                   ①

TaskName                              State
--------                              -----
\Microsoft\Windows\Defrag\ScheduledDefrag Ready
\Microsoft\Windows\UpdateOrchestrator\Schedule Scan Ready
```

① **Get-ScheduledTask** lista tareas programadas

Aspecto	Linux	macOS	Windows
Recurrente	cron/anacron	launchd	Task Scheduler

Aspecto	Linux	macOS	Windows
Una vez	`at`	(launchd)	Task Scheduler

55.6 Mejores practicas

RECOMENDACION

Toda automatizacion debe ser observable.
Casos de uso: - Backups y mantenimiento.
Cuando aplicar: 1. Loguea ejecucion y errores. 2. Agrega validaciones (por ejemplo: chequear espacio antes del backup). 3. Documenta el job (que hace, donde deja logs, como desactivarlo).

55.7 Resumen

- cron automatiza tareas recurrentes.
- anacron ejecuta tareas aunque el equipo estuviera apagado.
- at programa ejecuciones unicas.

55.8 Referencias

- crontab(5)
- anacrontab(5)
- at(1p)

56 Unidad 5: Recursos

Listing 56.1 `QUARTO-TITLE-BLOCK`

Procesos, servicios y automatizacion

57 Unidad 5: Recursos

57.1 Introduccion

Recursos de consulta y comandos clave alineados al brochure (procesos/servicios/scheduling).

57.2 Comandos clave (cheatsheet)

- Procesos: `ps aux`, `ps aux --sort=-%cpu`, `top`, `pgrep -a`, `kill -TERM`, `kill -KILL`
- Servicios: `systemctl status`, `systemctl restart`, `systemctl enable`, `journalctl -u`, `journalctl -f`
- Scheduling: `crontab -e`, `crontab -l`, `cat /etc/anacrontab`, `at`, `atq`, `atrm`

57.3 Referencias

- ps(1)
- top(1)
- kill(1)
- systemctl(1)
- journalctl(1)
- crontab(5)
- anacrontab(5)
- at(1p)

Part VII

Unidad 6: Almacenamiento y Sistemas de Archivos

58 Unidad 6.1: Sistemas de archivos

Listing 58.1 `QUARTO-TITLE-BLOCK`

ext4, XFS y Btrfs para servidores

59 Unidad 6.1: Sistemas de archivos

59.1 Introduccion

En servidores, el sistema de archivos define gran parte de la experiencia operativa: rendimiento, confiabilidad, recuperacion ante fallas y herramientas de mantenimiento.

En esta unidad trabajaremos con los sistemas de archivos mas usados en Linux:

- ext4 (estandar y muy estable)
- XFS (fuerte en archivos grandes y workloads intensivos)
- Btrfs (avanzado: snapshots, subvolumes, checksum)

59.2 Objetivos de aprendizaje

Al final de esta leccion, podras:

- Identificar el sistema de archivos en uso y sus montajes
- Elegir entre ext4/XFS/Btrfs segun caso de uso
- Crear un sistema de archivos en un dispositivo de laboratorio (loopback)

59.3 Conceptos clave

- **Dispositivo de bloques:** disco/particion (ej: `/dev/sda`, `/dev/sda1`).
- **Sistema de archivos (FS):** estructura que organiza datos (directorios/archivos/metadatos).
- **Mount point:** ruta donde se "pega" un FS (ej: `/`, `/var`, `/data`).
- **Journaling:** registro para recuperacion tras cortes (muy importante en servidores).
- **UUID:** identificador estable para montar sin depender de nombres (recomendado en `/etc/fstab`).

Listing 59.1 BASH

```
$ lsblk -f                                                                    ①
NAME   FSTYPE FSVER LABEL UUID                                 FSAVAIL FSUSE% MOUNTPOINTS
sda
 sda1  ext4   1.0         5b4f5d4e-2f3e-4c17-9b3b-9d4c7d7b0c2a    38G    12% /


$ df -hT                                                                      ②
Filesystem     Type  Size  Used Avail Use% Mounted on
/dev/sda1      ext4   49G  5.4G   41G  12% /


$ findmnt -n -o SOURCE,FSTYPE,TARGET /                                        ③
/dev/sda1 ext4 /
```

59.4 Ver que sistema de archivos estas usando

① **lsblk -f** lista discos/particiones y su tipo de FS (FSTYPE), UUID y montajes.
② **df -hT** muestra uso de disco y el tipo de FS por montaje.
③ **findmnt** confirma el origen y tipo del FS montado en `/`.

59.5 ext4 vs XFS vs Btrfs (decision rapida)

Aspecto	ext4	XFS	Btrfs
Uso recomendado	general / default	alto rendimiento / archivos grandes	snapshots y features avanzadas
Madurez	muy alta	muy alta	alta (pero mas complejo)
Snapshots nativos	no	no	si
Herramientas	`e2fsprogs`	`xfsprogs`	`btrfs-progs`
Complejidad operativa	baja	media	alta

RECOMENDACION

Si estas empezando en administracion de servidores, usa ext4 por defecto.
Casos de uso: - Particiones del sistema (`/`, `/var`, `/home`).
Cuando aplicar: - Cuando priorizas estabilidad, tooling y simplicidad.

59.6 Laboratorio seguro: crear un FS usando un disco de prueba (loop)

Este laboratorio crea un "disco" dentro de un archivo, para que puedas practicar sin tocar discos reales.

Listing 59.2 BASH

```
$ sudo fallocate -l 512M /tmp/u6-disk.img                                    ①


$ sudo losetup -fP /tmp/u6-disk.img                                          ②


$ sudo losetup -a | grep u6-disk.img                                         ③
/dev/loop0: [2065]:12345 (/tmp/u6-disk.img)
```

① **fallocate -l 512M** crea un archivo que representara un disco de laboratorio.
② **losetup -fP** asocia el archivo a un dispositivo loop (ej: `/dev/loop0`).
③ **losetup -a** verifica que el loop quedo creado.

Ahora (en la siguiente leccion) lo particionaremos y le crearemos un sistema de archivos.

59.7 Ejemplos practicos multi-SO

59.7.1 Ejemplo 1: Ver discos y volumenes

59.7.1.1 Linux

Listing 59.3 BASH

```
$ lsblk                                                                      ①
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda      8:0    0   50G  0 disk
 sda1   8:1    0   50G  0 part /
```

① **lsblk** lista discos/particiones y montajes en Linux.

59.7.1.2 macOS

Listing 59.4 BASH

```
$ diskutil list                                                    ①
/dev/disk0 (internal, physical):
  0: GUID_partition_scheme
  1: EFI EFI 314.6 MB
  2: Apple_APFS Container disk1 494.4 GB
```

① **diskutil list** lista discos y particiones en macOS.

59.7.1.3 Windows

Listing 59.5 POWERSHELL

```
PS> Get-Disk | Select-Object Number,FriendlyName,PartitionStyle,Size    ①

Number FriendlyName      PartitionStyle Size
------ ------------      -------------- ----
0      NVMe Samsung SSD  GPT            512105932800
```

① **Get-Disk** lista discos y su estilo de particion (GPT/MBR) en Windows.

Aspecto	Linux	macOS	Windows
Listar discos	`lsblk`	`diskutil list`	`Get-Disk`
FS por montaje	`df -T`	`df -T`	`Get-Volume`

59.8 Resumen

- Un sistema de archivos define como se guardan y recuperan datos.
- ext4 es el default mas simple y confiable.
- XFS y Btrfs agregan ventajas, pero con mayor complejidad operativa.

60 Unidad 6.2: Particiones y discos

Listing 60.1 `QUARTO-TITLE-BLOCK`

MBR vs GPT, fdisk y parted

61 Unidad 6.2: Particiones y discos

61.1 Introduccion

Antes de crear un sistema de archivos, normalmente debes preparar el disco: tabla de particiones (MBR/GPT) y particiones.

En esta leccion practicaremos particionado de forma segura usando dispositivos loop (discos de laboratorio).

61.2 Objetivos de aprendizaje

Al final, podras:

- Diferenciar MBR y GPT
- Identificar discos/particiones con `lsblk` y `fdisk`
- Crear una particion GPT con `parted` en un disco de laboratorio

61.3 MBR vs GPT

Aspecto	MBR	GPT
Compatibilidad	muy alta (legacy)	moderna (UEFI)
Particiones	limitado (4 primarias)	muchas (practico)
Tamanos	limitado	soporta discos muy grandes
Recomendacion	solo si necesitas legacy	default en servidores modernos

---

ADVERTENCIA CRITICA

Particionar/formatar el disco equivocado puede destruir datos de forma irreversible.
Lo que podria salir mal: - Ejecutar `parted`/`fdisk` sobre el disco del sistema (`/dev/sda`) y perder el servidor.

Como prevenirlo: 1. Identifica el disco con `lsblk` y valida por tamano/modelo. 2. Practica primero con discos de laboratorio (loopback). 3. Si es produccion, toma snapshot/backup antes de tocar particiones.

61.4 Identificar discos y particiones

Listing 61.1 BASH

```
$ lsblk                                                                    ①
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda      8:0    0   50G  0 disk
 sda1   8:1    0   50G  0 part /


$ sudo fdisk -l | sed -n '1,25p'                                           ②
Disk /dev/sda: 50 GiB, 53687091200 bytes, 104857600 sectors
Disklabel type: gpt
Device     Start       End   Sectors  Size Type
/dev/sda1   2048 104855551 104853504   50G Linux filesystem
```

---

① **lsblk** da una vista clara de discos y montajes.
② **fdisk -l** muestra tabla de particiones (requiere permisos); aqui filtramos la salida para leer rapido.

61.5 Laboratorio: crear una particion GPT en un disco loop

① **fallocate** crea el archivo-disco de laboratorio.
② **losetup -fP** asocia el archivo a un loop y habilita particiones (`p1`, `p2`).
③ **losetup -a** confirma el dispositivo asignado.
④ **parted mklabel gpt** crea tabla de particiones GPT.
⑤ **parted mkpart** crea una particion que ocupa el disco completo.
⑥ **lsblk** valida que aparecio `/dev/loop1p1`.

Listing 61.2 BASH

```
$ sudo fallocate -l 1G /tmp/u6-part.img                                    ①


$ sudo losetup -fP /tmp/u6-part.img                                        ②


$ sudo losetup -a | grep u6-part.img                                       ③
/dev/loop1: [2065]:54321 (/tmp/u6-part.img)


$ sudo parted /dev/loop1 --script mklabel gpt                              ④


$ sudo parted /dev/loop1 --script mkpart primary ext4 1MiB 100%            ⑤


$ lsblk /dev/loop1                                                         ⑥
NAME      MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop1       7:1    0    1G  0 loop
 loop1p1 259:0    0 1023M  0 part
```

61.6 Ejemplos practicos multi-SO

61.6.1 Ejemplo 1: Ver estilo de particion (MBR/GPT)

61.6.1.1 Linux

Listing 61.3 BASH

```
$ sudo fdisk -l /dev/sda | grep -E "Disklabel type"                        ①
Disklabel type: gpt
```

① **fdisk -l** muestra si el disco usa GPT o DOS/MBR.

61.6.1.2 macOS

① **GUID_partition_scheme** equivale a GPT en macOS.

Listing 61.4 BASH

```
$ diskutil list | head -n 8                                                    ①
/dev/disk0 (internal, physical):
   0: GUID_partition_scheme
   1: EFI EFI 314.6 MB
   2: Apple_APFS Container disk1 494.4 GB
```

Listing 61.5 POWERSHELL

```
PS> Get-Disk | Select-Object Number,PartitionStyle                             ①

Number PartitionStyle
------ --------------
0      GPT
```

61.6.1.3 Windows

① **Get-Disk** muestra el estilo de particion del disco.

Aspecto	Linux	macOS	Windows
Ver GPT/MBR	`fdisk -l`	`diskutil list`	`Get-Disk`

61.7 Resumen

- GPT es el estandar moderno recomendado.
- `lsblk` y `fdisk -l` son tus primeras herramientas para identificar discos.
- Practica particionado con loopback antes de tocar discos reales.

62 Unidad 6.3: Montaje y desmontaje

Listing 62.1 `QUARTO-TITLE-BLOCK`

mount, umount y /etc/fstab

63 Unidad 6.3: Montaje y desmontaje

63.1 Introduccion

En Linux, un disco no "aparece" automaticamente como una letra (como en Windows). En su lugar, se monta en un directorio: `/mnt`, `/data`, `/var/lib`, etc.

En esta leccion practicaremos:

- montar y desmontar
- identificar por UUID
- configurar montaje persistente con `/etc/fstab`

63.2 Objetivos de aprendizaje

Al final, podras:

- Montar y desmontar un FS
- Encontrar el UUID con `blkid`
- Escribir una entrada segura en `/etc/fstab` y validarla con `mount -a`

63.3 Laboratorio: crear FS, montar, desmontar

① **fallocate** crea el disco de laboratorio.
② **losetup** crea el dispositivo loop.
③ **losetup -a** confirma que `/dev/loop2` apunta al archivo.
④ **parted mklabel gpt** prepara GPT.
⑤ **parted mkpart** crea la particion.
⑥ **mkfs.ext4** crea el sistema de archivos ext4 (nota: imprime un UUID).
⑦ **mkdir -p** crea el directorio donde se montara el FS.
⑧ **mount** monta el dispositivo en el directorio.
⑨ **df -hT** verifica que esta montado y el tipo de FS.
⑩ **umount** desmonta de forma segura.

63.4 Montaje persistente con /etc/fstab (con UUID)

Primero obtenemos el UUID del dispositivo:

① **blkid** imprime UUID y tipo de sistema de archivos.

Ejemplo de entrada en `/etc/fstab` (referencia):

> ⚠ ADVERTENCIA CRITICA
>
> **Un error en `/etc/fstab` puede dejar tu servidor sin boot.**
> Lo que podria salir mal: - El sistema intenta montar algo inexistente y cae a modo emergencia.
> Como prevenirlo: 1. Usa **UUID** (no `/dev/sdb1`). 2. Agrega `nofail` si es un disco opcional. 3. Siempre prueba con `sudo mount -a` antes de reiniciar.

Validacion recomendada:

① **mount -a** intenta montar todo lo definido en `/etc/fstab` (prueba rapida sin reiniciar).
② **findmnt** confirma que el montaje existe.

63.5 Ejemplos practicos multi-SO

63.5.1 Ejemplo 1: Ver montajes actuales

63.5.1.1 Linux

① **findmnt** muestra montajes con origen y tipo.

63.5.1.2 macOS

① **mount** lista sistemas montados en macOS.

63.5.1.3 Windows

① **Get-Volume** lista volumenes y FS en Windows.

Aspecto	Linux	macOS	Windows
Ver montajes	`findmnt`	`mount`	`Get-Volume`
Persistencia	`/etc/fstab`	`/etc/fstab` (casos)	Disk Management

63.6 Resumen

- Montar es “conectar” un FS a un directorio.
- Usa UUID en `/etc/fstab`.
- Siempre valida con `mount -a` antes de reiniciar.

Listing 63.1 BASH

```
$ sudo fallocate -l 512M /tmp/u6-mount.img                                    ①


$ sudo losetup -fP /tmp/u6-mount.img                                          ②


$ sudo losetup -a | grep u6-mount.img                                         ③
/dev/loop2: [2065]:11111 (/tmp/u6-mount.img)


$ sudo parted /dev/loop2 --script mklabel gpt                                 ④


$ sudo parted /dev/loop2 --script mkpart primary ext4 1MiB 100%               ⑤


$ sudo mkfs.ext4 -F /dev/loop2p1                                              ⑥
mke2fs 1.46.5 (30-Dec-2021)
Creating filesystem with 130560 4k blocks and 32768 inodes
Filesystem UUID: 9a1b2c3d-4e5f-6789-aaaa-bbbbccccdddd


$ sudo mkdir -p /mnt/u6-data                                                  ⑦


$ sudo mount /dev/loop2p1 /mnt/u6-data                                        ⑧


$ df -hT /mnt/u6-data                                                         ⑨
Filesystem     Type  Size  Used Avail Use% Mounted on
/dev/loop2p1   ext4  474M   24K  448M   1% /mnt/u6-data


$ sudo umount /mnt/u6-data                                                    ⑩
```

Listing 63.2 BASH

```
$ sudo blkid /dev/loop2p1                                                     ①
/dev/loop2p1: UUID="9a1b2c3d-4e5f-6789-aaaa-bbbbccccdddd" TYPE="ext4" PARTUUID="cafe-babe
```

Listing 63.3 TEXT

```
UUID=9a1b2c3d-4e5f-6789-aaaa-bbbbccccdddd  /data  ext4  defaults,noatime  0  2
```

Listing 63.4 BASH

```
$ sudo mount -a                                                    ①


$ findmnt | grep -E "\s/data\s"                                    ②
/data /dev/sdb1 ext4 rw,noatime,relatime
```

Listing 63.5 BASH

```
$ findmnt | head -n 6                                              ①
TARGET SOURCE    FSTYPE OPTIONS
/      /dev/sda1 ext4   rw,relatime
/boot  /dev/sda2 ext4   rw,relatime
```

Listing 63.6 BASH

```
$ mount | head -n 5                                                ①
/dev/disk3s1s1 on / (apfs, local, read-only, journaled)
```

Listing 63.7 POWERSHELL

```
PS> Get-Volume | Select-Object DriveLetter,FileSystem,SizeRemaining,Size ①

DriveLetter FileSystem SizeRemaining        Size
----------- ---------- -------------        ----
C           NTFS       120345678901  256060514304
```

64 Unidad 6.4: Gestion de espacio

Listing 64.1 `QUARTO-TITLE-BLOCK`

df, du, logs y quotas

65 Unidad 6.4: Gestion de espacio

65.1 Introduccion

En produccion, quedarse sin espacio en disco rompe servicios: bases de datos, logs, Docker, actualizaciones y backups.

En esta leccion veras un flujo simple y repetible para diagnosticar consumo de disco.

65.2 Objetivos de aprendizaje

Al final, podras:

- Medir uso de disco con `df`
- Encontrar directorios pesados con `du`
- Controlar crecimiento de logs con `journalctl`

65.3 df: cuanto espacio queda (por FS)

Listing 65.1 BASH

```
$ df -hT                                                                    ①
Filesystem     Type  Size  Used Avail Use% Mounted on
/dev/sda1      ext4   49G  5.4G   41G  12% /
tmpfs          tmpfs  1.9G  1.2M  1.9G   1% /run
```

---

① **df -hT** muestra uso por filesystem (tipo incluido).

65.4 du: donde se esta yendo el espacio

① **du -xh** estima uso por directorio; con `sort/head` ves rapidamente los mas grandes.

Listing 65.2 BASH

```
$ sudo du -xh --max-depth=1 /var | sort -hr | head -n 10                    ①
3.2G    /var/lib
1.1G    /var/log
240M    /var/cache
```

RECOMENDACION

Si el servidor usa Docker, revisa primero `/var/lib/docker`.
Casos de uso: - Servidores que acumulan imagenes y volumes.
Cuando aplicar: - Cuando `df -h` sube rapido sin cambios visibles en tu app.

65.5 Logs del systemd journal (comun en Ubuntu Server)

Listing 65.3 BASH

```
$ journalctl --disk-usage                                                   ①
Archived and active journals take up 512.0M in the file system.


$ sudo journalctl --vacuum-time=7d                                          ②
Vacuuming done, freed 256.0M of archived journals from /var/log/journal.
```

① **journalctl –disk-usage** mide cuanto ocupa el journal.
② **journalctl –vacuum-time=7d** elimina logs antiguos (ajusta segun politica).

ADVERTENCIA CRITICA

No borres logs a ciegas en produccion.
Lo que podria salir mal: - Pierdes evidencia para auditoria/incidentes.
Como prevenirlo: 1. Define retencion (por dias o tamano) y documentala. 2. Valida con el equipo (seguridad/operaciones) antes de acortar retencion. 3. Si borras, deja evidencia (ticket/cambio).

65.6 Quotas (idea general)

Las quotas limitan cuanto puede consumir un usuario o grupo en un filesystem (muy util en servidores multi-usuario).

Ejemplo de flujo (alto nivel):

Listing 65.4 BASH

```
$ sudo apt update                                                    ①


$ sudo apt install -y quota                                          ②
```

① **apt update** actualiza indices de paquetes.
② **quota** instala herramientas de cuotas (si decides aplicarlas).

65.7 Ejemplos practicos multi-SO

65.7.1 Ejemplo 1: Ver espacio disponible

65.7.1.1 Linux

Listing 65.5 BASH

```
$ df -h                                                              ①
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1        49G  5.4G   41G  12% /
```

① **df -h** muestra espacio libre por filesystem.

65.7.1.2 macOS

Listing 65.6 BASH

```
$ df -h                                                                                  ①
Filesystem      Size   Used  Avail Capacity iused ifree %iused  Mounted on
/dev/disk3s1s1  466Gi  14Gi  218Gi     6%  488k  2.1G    0%   /
```

① **df -h** tambien existe en macOS.

65.7.1.3 Windows

Listing 65.7 POWERSHELL

```
PS> Get-PSDrive -PSProvider FileSystem | Select-Object Name,Used,Free    ①

Name Used          Free
---- ----          ----
C    135791468544 120345678901
```

① **Get-PSDrive** entrega usados/libres por unidad.

Aspecto	Linux	macOS	Windows
Espacio libre	`df -h`	`df -h`	`Get-PSDrive`
Top directorios	`du -xh`	`du -h`	(GUI/PowerShell)

65.8 Resumen

- `df` responde “cuanto queda”.
- `du` responde “quien se lo esta comiendo”.
- Controla logs (journal) y define politicas de retencion.

66 Unidad 6.5: RAID y LVM (basico)

Listing 66.1 `QUARTO-TITLE-BLOCK`

mdadm, volumnes logicos y snapshots

67 Unidad 6.5: RAID y LVM (basico)

67.1 Introduccion

En servidores, el almacenamiento suele requerir:

- **Redundancia** (que un disco se pueda morir sin caida)
- **Flexibilidad** (crecer volumenes sin reinstalar)
- **Snapshots** (backup/logica de punto en el tiempo)

RAID y LVM son dos piezas comunes para lograrlo.

67.2 Objetivos de aprendizaje

Al final, podras:

- Entender RAID 0/1/5/6 a nivel conceptual
- Crear un RAID1 de laboratorio con `mdadm` (con discos loop)
- Crear un volumen LVM basico (PV/VG/LV) y un snapshot

67.3 RAID: guia rapida

Nivel	Objetivo	Pros	Contras
RAID 0	rendimiento	rapido	sin redundancia
RAID 1	redundancia	simple, tolera 1 disco	capacidad al 50%
RAID 5	balance	tolera 1 disco	rebuild costoso
RAID 6	mas tolerancia	tolera 2 discos	mas overhead

> ⚠ ADVERTENCIA CRITICA
>
> **RAID NO es backup.**
> Lo que podria salir mal: - Borrado accidental, ransomware o corrupcion se replica.
> Como prevenirlo: 1. Mantener backups fuera del arreglo (otra maquina/nube). 2. Probar restauraciones (no solo crear backups).

67.4 Laboratorio: RAID1 con mdadm usando discos loop

Listing 67.1 BASH

```
$ sudo apt update                                                    ①


$ sudo apt install -y mdadm                                          ②


$ sudo fallocate -l 200M /tmp/u6-raid-a.img                          ③


$ sudo fallocate -l 200M /tmp/u6-raid-b.img                          ④


$ sudo losetup -fP /tmp/u6-raid-a.img                                ⑤


$ sudo losetup -fP /tmp/u6-raid-b.img                                ⑥


$ sudo losetup -a | grep u6-raid                                     ⑦
/dev/loop3: [2065]:22222 (/tmp/u6-raid-a.img)
/dev/loop4: [2065]:33333 (/tmp/u6-raid-b.img)


$ sudo mdadm --create /dev/md0 --level=1 --raid-devices=2 /dev/loop3 /dev/loop4 ⑧
mdadm: array /dev/md0 started.


$ cat /proc/mdstat                                                   ⑨
Personalities : [raid1]
md0 : active raid1 loop4[1] loop3[0]
      204800 blocks super 1.2 [2/2] [UU]
```

① **apt update** actualiza indices (si necesitas instalar herramientas).
② **mdadm** es la herramienta estandar para RAID por software.
③ **fallocate** crea el primer disco de laboratorio.
④ **fallocate** crea el segundo disco de laboratorio.
⑤ **losetup** crea `/dev/loop3`.
⑥ **losetup** crea `/dev/loop4`.
⑦ **losetup -a** verifica ambos discos loop.
⑧ **mdadm –create** crea un RAID1 llamado `/dev/md0`.
⑨ **/proc/mdstat** muestra el estado del arreglo (UU = ambos discos OK).

67.5 LVM: PV, VG, LV y snapshot (concepto + demo)

① **lvm2** instala herramientas LVM.
② **fallocate** crea un disco de laboratorio para LVM.
③ **losetup** crea `/dev/loop5`.
④ **losetup -a** verifica el loop.
⑤ **pvcreate** inicializa un Physical Volume.
⑥ **vgcreate** crea el Volume Group.
⑦ **lvcreate** crea un Logical Volume de 200M.
⑧ **mkfs.ext4** crea un FS sobre el LV.
⑨ **mkdir -p** crea el punto de montaje.
⑩ **mount** monta el LV.
⑪ **lvcreate -s** crea un snapshot (toma espacio del pool, no es infinito).

67.6 Resumen

- RAID mejora disponibilidad/rendimiento segun nivel, pero no reemplaza backups.
- LVM facilita crecer volumenes y crear snapshots.
- Practica con loopback y documenta el procedimiento antes de aplicarlo en produccion.

Listing 67.2 BASH

```
$ sudo apt install -y lvm2                                                    ①


$ sudo fallocate -l 300M /tmp/u6-lvm.img                                      ②


$ sudo losetup -fP /tmp/u6-lvm.img                                            ③


$ sudo losetup -a | grep u6-lvm.img                                           ④
/dev/loop5: [2065]:44444 (/tmp/u6-lvm.img)


$ sudo pvcreate /dev/loop5                                                    ⑤
Physical volume "/dev/loop5" successfully created.


$ sudo vgcreate vg_data /dev/loop5                                            ⑥
Volume group "vg_data" successfully created


$ sudo lvcreate -n lv_app -L 200M vg_data                                     ⑦
Logical volume "lv_app" created.


$ sudo mkfs.ext4 -F /dev/vg_data/lv_app                                       ⑧


$ sudo mkdir -p /mnt/vg_data                                                  ⑨


$ sudo mount /dev/vg_data/lv_app /mnt/vg_data                                 ⑩


$ sudo lvcreate -s -n lv_app_snap -L 50M /dev/vg_data/lv_app                  ⑪
Logical volume "lv_app_snap" created.
```

68 Unidad 6: Recursos

Listing 68.1 `QUARTO-TITLE-BLOCK`

Almacenamiento, particiones y montaje

69 Unidad 6: Recursos

69.1 Cheatsheet (comandos clave)

- Inventario: `lsblk`, `lsblk -f`, `blkid`, `df -hT`, `findmnt`
- Particiones: `fdisk -l`, `parted -l`, `parted /dev/sdX --script ...`
- FS: `mkfs.ext4`, `mkfs.xfs`, `mkfs.btrfs`
- Montaje: `mount`, `umount`, `/etc/fstab`, `mount -a`
- Espacio: `du -xh --max-depth=1`, `journalctl --disk-usage`, `journalctl --vacuum-time=...`
- RAID: `mdadm --create`, `cat /proc/mdstat`
- LVM: `pvcreate`, `vgcreate`, `lvcreate`, `lvs`, `vgs`, `pvs`

69.2 Referencias

- lsblk(8)
- fdisk(8)
- parted(8)
- mount(8)
- fstab(5)
- mdadm(8)
- lvm(8)

Part VIII

Unidad 7: Redes basicas

70 Unidad 7.1: Fundamentos de redes

Listing 70.1 `QUARTO-TITLE-BLOCK`

IP, puertos, TCP/UDP y modelo operativo

71 Unidad 7.1: Fundamentos de redes

71.1 Introduccion

En administracion de servidores, red significa: identificar tu interfaz, saber tu IP, entender por que un servicio no responde, y validar el camino entre cliente y servidor.

En esta unidad veremos lo necesario para operar redes basicas en Linux sin depender de una GUI.

71.2 Objetivos de aprendizaje

Al final de esta leccion, podras:

- Diferenciar IP, gateway, DNS y puerto
- Identificar interfaces, direccionamiento y puertos activos
- Validar conectividad con pruebas simples y reproducibles

71.3 Conceptos clave

- **IP**: direccion logica (IPv4/IPv6) de un host.
- **Mascara / prefijo**: define la red (ej: `/24`).
- **Gateway**: salida hacia otras redes (ruta por defecto).
- **DNS**: resuelve nombres a IP.
- **Puerto**: canal logico en un host (ej: 22/SSH, 80/HTTP, 443/HTTPS).

71.4 Identificar interfaces e IP

① **ip -br link** muestra interfaces y su estado (UP/DOWN).
② **ip -br addr** muestra direccionamiento IPv4/IPv6 por interfaz.

Listing 71.1 BASH

```
$ ip -br link                                                                    ①
lo               UNKNOWN        00:00:00:00:00:00
enp0s3           UP             08:00:27:12:34:56


$ ip -br addr                                                                    ②
lo               UNKNOWN        127.0.0.1/8 ::1/128
enp0s3           UP             192.168.56.10/24 fe80::a00:27ff:fe12:3456/64
```

71.5 Validar ruta por defecto (gateway)

Listing 71.2 BASH

```
$ ip route                                                                       ①
default via 192.168.56.1 dev enp0s3
192.168.56.0/24 dev enp0s3 proto kernel scope link src 192.168.56.10
```

① **ip route** muestra rutas; la ruta `default` indica el gateway.

71.6 Revisar puertos en escucha

Listing 71.3 BASH

```
$ sudo ss -lntp                                                                  ①
State   Recv-Q  Send-Q   Local Address:Port  Peer Address:Port  Process
LISTEN  0       4096     0.0.0.0:22          0.0.0.0:*          users:("sshd",pid=812,fd=
```

① **ss -lntp** lista puertos TCP en escucha y el proceso asociado.

71.7 Ejemplos practicos multi-SO

71.7.1 Ejemplo 1: Ver IP del equipo

71.7.1.1 Linux

Listing 71.4 BASH

```
$ ip -br addr                                                    ①
lo     UNKNOWN 127.0.0.1/8 ::1/128
eth0   UP      192.168.1.10/24
```

① **ip -br addr** muestra IP de forma compacta.

71.7.1.2 macOS

Listing 71.5 BASH

```
$ ifconfig | head -n 18                                          ①
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
  inet 127.0.0.1 netmask 0xff000000
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
  inet 192.168.1.20 netmask 0xffffff00 broadcast 192.168.1.255
```

① **ifconfig** lista interfaces y direcciones (en macOS sigue siendo comun).

71.7.1.3 Windows

Listing 71.6 POWERSHELL

```
PS> ipconfig                                                     ①

Windows IP Configuration

Ethernet adapter Ethernet:
   IPv4 Address. . . . . . . . . . . : 192.168.1.30
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1
```

① **ipconfig** muestra IP, mascara y gateway en Windows.

71.8 Mejores practicas

- Valida en este orden: interfaz UP -> IP -> gateway -> DNS -> puertos.
- Documenta evidencias: salida de `ip -br addr`, `ip route` y `ss -lntp`.

71.9 Resumen

- Con `ip` y `ss` puedes diagnosticar el 80% de problemas de red iniciales.

72 Unidad 7.2: iproute2 y rutas

Listing 72.1 `QUARTO-TITLE-BLOCK`

Diagnostico de conectividad con ip/ss

73 Unidad 7.2: iproute2 y rutas

73.1 Introduccion

`iproute2` es el set moderno de herramientas para administrar red en Linux (reemplaza en gran parte a `ifconfig`/`route`).

73.2 Objetivos de aprendizaje

- Inspeccionar IP, rutas y vecinos (ARP/NDP)
- Entender la ruta por defecto y como afecta salida a Internet
- Validar puertos y sockets con `ss`

73.3 Rutas: leer y explicar lo que ves

Listing 73.1 BASH

```
$ ip route                                                                    ①
default via 192.168.56.1 dev enp0s3
192.168.56.0/24 dev enp0s3 proto kernel scope link src 192.168.56.10


$ ip -6 route                                                                 ②
fe80::/64 dev enp0s3 proto kernel metric 256
default via fe80::1 dev enp0s3 metric 100
```

---

① **ip route** muestra rutas IPv4; `default via ...` es la salida por defecto.
② **ip -6 route** muestra rutas IPv6.

Listing 73.2 BASH

```
$ ip neigh                                                                    ①
192.168.56.1 dev enp0s3 lladdr 0a:00:27:aa:bb:cc REACHABLE
```

73.4 Vecinos (ARP/NDP): “a quien le hablo en L2”

① **ip neigh** muestra la tabla de vecinos (ARP en IPv4, NDP en IPv6).

73.5 Ver sockets y puertos

Listing 73.3 BASH

```
$ sudo ss -lntup                                                              ①
Netid State  Recv-Q Send-Q Local Address:Port Peer Address:Port Process
tcp   LISTEN 0      4096   0.0.0.0:22        0.0.0.0:*       users:("sshd",pid=812,fd=3)
udp   UNCONN 0      0      127.0.0.53%lo:53  0.0.0.0:*       users:("systemd-resolved",pi
```

① **ss -lntup** lista sockets en escucha TCP/UDP y el proceso asociado.

73.6 Ejemplos practicos multi-SO

73.6.1 Ejemplo 1: Ver rutas

73.6.1.1 Linux

Listing 73.4 BASH

```
$ ip route                                                                    ①
default via 192.168.1.1 dev eth0
192.168.1.0/24 dev eth0 proto kernel scope link src 192.168.1.10
```

① **ip route** lista rutas IPv4 del host.

73.6.1.2 macOS

Listing 73.5 BASH

```
$ netstat -rn | head -n 10                                                    ①
Routing tables

Internet:
Destination        Gateway            Flags        Netif Expire
default            192.168.1.1        UGSc           en0
192.168.1.0/24     link#4             UCS            en0
```

① **netstat -rn** muestra tabla de rutas en macOS.

73.6.1.3 Windows

Listing 73.6 POWERSHELL

```
PS> route print                                                               ①
IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface
          0.0.0.0          0.0.0.0      192.168.1.1     192.168.1.30
```

① **route print** muestra la tabla de rutas en Windows.

73.7 Mejores practicas

- Si no hay `default route`, no hay salida a otras redes.
- Si hay IP pero no hay vecino para el gateway (`ip neigh`), revisa L2 (VLAN/bridge/virtual switch).

74 Unidad 7.3: nmcli y netplan

Listing 74.1 `QUARTO-TITLE-BLOCK`

Configuracion de red en Ubuntu Server

75 Unidad 7.3: nmcli y netplan

75.1 Introduccion

En Ubuntu Server, la configuracion de red normalmente se define con Netplan (archivos YAML) y se aplica a traves de `systemd-networkd` o NetworkManager.

En servidores, la prioridad es evitar cortes de conectividad (especialmente si administras por SSH).

> ⚠ ADVERTENCIA CRITICA
>
> **Cambiar IP/gateway por SSH puede dejarte sin acceso remoto.**
> **Lo que podria salir mal:** - Pierdes conectividad y no puedes volver a entrar por SSH. - Dejas una interfaz sin gateway o con una mascara incorrecta.
> **Como prevenirlo:** 1. Trabaja con consola directa (hipervisor / iLO / consola cloud) si es posible. 2. Aplica cambios y valida en el momento (`ip -br addr`, `ip route`, `ping`). 3. Mantén un plan de rollback (archivo previo + ventana de mantenimiento).

75.2 Objetivos de aprendizaje

- Identificar si usas NetworkManager o systemd-networkd
- Leer/editar Netplan y aplicarlo con seguridad
- Usar `nmcli` para inspeccion y cambios controlados (cuando aplique)

75.3 Ver que renderer estas usando

① **ls /etc/netplan** muestra archivos de configuracion Netplan.
② **netplan get** imprime la configuracion efectiva en formato legible.

Listing 75.1 BASH

```
$ ls -1 /etc/netplan                                                    ①
00-installer-config.yaml


$ sudo netplan get                                                      ②
network:
  version: 2
  ethernets:
    enp0s3:
      dhcp4: true
```

Listing 75.2 BASH

```
$ sudo netplan try                                                      ①
Do you want to keep these settings? Press ENTER before the timeout to accept.


$ sudo netplan apply                                                    ②
```

75.4 Aplicar Netplan (forma segura)

① **netplan try** aplica temporalmente y permite rollback automatico si no confirmas.
② **netplan apply** aplica cambios permanentes.

75.5 Inspeccion con nmcli (si esta instalado y activo)

Listing 75.3 BASH

```
$ nmcli general status                                                  ①
STATE      CONNECTIVITY  WIFI-HW  WIFI     WWAN-HW  WWAN
connected  full          enabled  enabled  enabled  enabled


$ nmcli device status                                                   ②
DEVICE  TYPE      STATE      CONNECTION
enp0s3  ethernet  connected  Wired connection 1
```

① **nmcli general status** muestra estado general de NetworkManager.
② **nmcli device status** muestra estado por interfaz.

75.6 Ejemplos practicos multi-SO

75.6.1 Ejemplo 1: Ver configuracion de red

75.6.1.1 Linux

Listing 75.4 BASH

```
$ sudo netplan get                                                    ①
network:
  version: 2
  ethernets:
    enp0s3:
      dhcp4: true
```

---

① **netplan get** muestra configuracion Netplan.

75.6.1.2 macOS

Listing 75.5 BASH

```
$ networksetup -listallnetworkservices                                ①
An asterisk (*) denotes that a network service is disabled.
Wi-Fi
USB 10/100/1000 LAN
```

---

① **networksetup** lista servicios de red configurados en macOS.

75.6.1.3 Windows

① **Get-NetIPConfiguration** muestra IP/gateway/DNS por interfaz en Windows.

Listing 75.6 POWERSHELL

```
PS> Get-NetIPConfiguration                                                    (1)

InterfaceAlias        : Ethernet
IPv4Address           : 192.168.1.30
IPv4DefaultGateway    : 192.168.1.1
DnsServer             : 192.168.1.1
```

75.7 Mejores practicas

- Preferir `netplan try` sobre `netplan apply` cuando estas remoto.
- Guarda copia: `sudo cp /etc/netplan/00-installer-config.yaml /etc/netplan/00-installer-config.yaml.bak`.

76 Unidad 7.4: DNS y resolucion de nombres

Listing 76.1 `QUARTO-TITLE-BLOCK`

systemd-resolved, resolv.conf y herramientas

77 Unidad 7.4: DNS y resolucion de nombres

77.1 Introduccion

Muchos problemas “de red” en realidad son problemas de DNS: tienes IP y ruta, pero el nombre no resuelve o resuelve a una IP equivocada.

77.2 Objetivos de aprendizaje

- Ver que servidores DNS estas usando
- Probar resolucion con herramientas basicas
- Identificar fallas tipicas (NXDOMAIN, timeout, cache)

77.3 Ver estado de systemd-resolved (Ubuntu)

Listing 77.1 BASH

```
$ resolvectl status                                                    ①
Global
       Protocols: -LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported
resolv.conf mode: stub

Link 2 (enp0s3)
    Current Scopes: DNS
         Protocols: +DefaultRoute
Current DNS Server: 192.168.56.1
       DNS Servers: 192.168.56.1
```

---

① **resolvectl status** muestra DNS efectivo por interfaz y modo de resolv.conf.

Listing 77.2 BASH

```
$ getent hosts example.com                                          ①
93.184.216.34   example.com


$ dig +short example.com                                            ②
93.184.216.34


$ nslookup example.com                                              ③
Server:         192.168.56.1
Address:        192.168.56.1#53

Non-authoritative answer:
Name:   example.com
Address: 93.184.216.34
```

77.4 Probar resolucion

① **getent hosts** prueba resolucion usando el resolver del sistema (glibc/nsswitch).
② **dig** consulta DNS directamente y muestra respuestas de forma compacta.
③ **nslookup** es una alternativa sencilla para consultas rapidas.

77.5 Revisar resolv.conf

Listing 77.3 BASH

```
$ ls -l /etc/resolv.conf                                            ①
lrwxrwxrwx 1 root root 39 Feb  2 10:10 /etc/resolv.conf -> ../run/systemd/resolve/stub-re
```

① **ls -l /etc/resolv.conf** muestra si es symlink (comun con systemd-resolved).

77.6 Ejemplos practicos multi-SO

77.6.1 Ejemplo 1: Resolver un nombre

77.6.1.1 Linux

Listing 77.4 BASH

```
$ getent hosts example.com                                    ①
93.184.216.34 example.com
```

① **getent hosts** valida resolucion via el resolver del sistema.

77.6.1.2 macOS

Listing 77.5 BASH

```
$ scutil --dns | head -n 18                                   ①
DNS configuration

resolver #1
  nameserver[0] : 192.168.1.1
  if_index : 4 (en0)
  flags    : Request A records
```

① **scutil –dns** muestra configuracion DNS efectiva en macOS.

77.6.1.3 Windows

Listing 77.6 POWERSHELL

```
PS> Resolve-DnsName example.com                               ①

Name                           Type   TTL   Section    IPAddress
----                           ----   ---   -------    ---------
example.com                    A      900   Answer     93.184.216.34
```

① **Resolve-DnsName** consulta DNS y muestra respuesta.

77.7 Mejores practicas

- Usa `getent hosts` para validar “lo que usara” tu app.
- Si `dig` funciona pero `getent` no, revisa `nsswitch.conf` y caches.

78 Unidad 7: Recursos

Listing 78.1 `QUARTO-TITLE-BLOCK`

Redes basicas

79 Unidad 7: Recursos

79.1 Introduccion

Recursos de consulta rapida para redes basicas en Linux.

79.2 Comandos clave (cheatsheet)

- Inspeccion: `ip -br addr`, `ip route`, `ip neigh`, `ss -lntup`
- DNS: `resolvectl status`, `getent hosts`, `dig +short`, `nslookup`
- Config (Ubuntu): `netplan get`, `netplan try`, `netplan apply`, `nmcli device status`

79.3 Referencias

- ip(8)
- ss(8)
- netplan
- systemd-resolved

Part IX

Unidad 8: Introducción a Servidores Web

80 Introducción a la web

Figure 80.1: Intro to Web

80.1 ¿Qué es la web?

Definición: La web es un sistema de documentos interconectados que se acceden a través de Internet.

80.2 Diferencia entre Internet y la Web:

Internet es una red global de computadoras conectadas entre sí que permite el intercambio de información (red de redes). La Web (o la World Wide Web, WWW) es una forma específica de acceder a la información a través de documentos interconectados (páginas web) usando navegadores. La web es solo una parte de Internet.

80.3 Ejemplo de productos/servicios:

Google Search: Un motor de búsqueda que accede a la web para ofrecer resultados.

Facebook, Twitter, Instagram: Redes sociales que permiten interactuar con contenido en la web.

80.4 ¿Cómo funciona la web?

Proceso de carga de una página web:

1. Un usuario escribe una URL en su navegador (ej., www.ejemplo.com).
2. El navegador envía una solicitud HTTP a un servidor web donde está alojada la página.
3. El servidor responde con archivos HTML, CSS y JavaScript.
4. El navegador interpreta estos archivos y muestra la página web.

Ejemplo de productos/servicios:

Amazon: Cuando accedes a la tienda, tu navegador hace solicitudes HTTP al servidor de Amazon para cargar los productos, imágenes, y precios.

80.5 ¿Qué es un navegador web?

Definición: El navegador es un software que permite acceder a las páginas web. Interpreta el código HTML, CSS y JavaScript para mostrar la información en forma de contenido visual.

Ejemplo de productos/servicios:

Google Chrome, Mozilla Firefox, Safari: Son navegadores que permiten ver páginas web y usar aplicaciones en línea como Gmail, YouTube, etc.

80.6 ¿Qué es HTML?

Definición: HTML (HyperText Markup Language) es el lenguaje de marcado estándar utilizado para crear páginas web. Define la estructura de los documentos web mediante etiquetas.

Ejemplo de productos/servicios:

Wikipedia: La información se estructura usando HTML, con encabezados, párrafos, listas y enlaces para organizar el contenido.

Ejemplo de código básico HTML:

Listing 80.1 HTML

```html
<html>
  <head>
    <title>Mi Página Web</title>
  </head>
  <body>
    <h1>Bienvenidos a mi sitio web</h1>
    <p>Este es un párrafo de texto.</p>
    <a href="https://www.google.com">Ir a Google</a>
  </body>
</html>
```

80.7 ¿Qué es CSS?

Definición: CSS (Cascading Style Sheets) es un lenguaje utilizado para describir la presentación de un documento HTML. Permite estilizar elementos como el color, tamaño y disposición de los elementos en la página.

Ejemplo de productos/servicios:

Netflix: La interfaz de usuario de Netflix se diseña con CSS para que el contenido (películas, series) se muestre de manera atractiva y organizada.

Ejemplo de código CSS:

Listing 80.2 CSS

```css
body {
  font-family: Arial, sans-serif;
  background-color: #f0f0f0;
}

h1 {
  color: #2c3e50;
}

p {
  font-size: 16px;
}
```

80.8 ¿Qué es JavaScript?

Definición: JavaScript es un lenguaje de programación que permite agregar interactividad a las páginas web. Se ejecuta en el navegador del usuario, lo que permite crear dinámicamente contenido o interactuar con el servidor sin necesidad de recargar la página.

Conceptos básicos de JavaScript:

Variables: Contienen datos (números, cadenas de texto, etc.).

Listing 80.3 JAVASCRIPT

```
let nombre = "Juan";
Funciones: Bloques de código que realizan una tarea específica.
javascript
Copiar código
function saludar() {
  console.log("¡Hola, " + nombre + "!");
}
saludar();  // Imprime: ¡Hola, Juan!
Eventos: JavaScript se utiliza para manejar eventos como clics o movimientos del ratón.
javascript
Copiar código
document.getElementById("boton").onclick = function() {
  alert("¡Botón presionado!");
}
```

Ejemplo de productos/servicios:

Google Maps: Utiliza JavaScript para mostrar mapas interactivos y permitir que los usuarios hagan clic en ubicaciones. **YouTube**: Utiliza JavaScript para permitir la reproducción de videos y la interacción con la interfaz.

80.9 ¿Qué es Node.js?

Definición: Node.js es un entorno de ejecución para JavaScript en el lado del servidor. Permite usar JavaScript para crear aplicaciones backend, lo que facilita la construcción de aplicaciones completas con JavaScript tanto en el frontend como en el backend.

Ejemplo de productos/servicios:

Netflix (backend): Utiliza Node.js para manejar grandes volúmenes de datos y conexiones en tiempo real para la transmisión de contenido.

Ejemplo de código básico en Node.js:

80.10 ¿Qué es React?

Definicióno: React es una biblioteca de JavaScript para construir interfaces de usuario. Permite crear aplicaciones web interactivas mediante la construcción de componentes reutilizables que gestionan su propio estado.

Ejemplo de productos/servicios:

Listing 80.4 JAVASCRIPT

```
const http = require('http');
const server = http.createServer((req, res) => {
  res.write('¡Hola Mundo!');
  res.end();
});
server.listen(3000, () => {
  console.log('Servidor corriendo en http://localhost:3000');
});
```

Facebook: Fue desarrollado con React para construir interfaces de usuario rápidas y reactivas. **Instagram**: Utiliza React para gestionar la interacción con las publicaciones, los comentarios y las actualizaciones en tiempo real.

Ejemplo de código básico en React:

Listing 80.5 JAVASCRIPT

```
import React, { useState } from 'react';

function App() {
  const [contador, setContador] = useState(0);

  return (
    <div>
      <h1>Contador: {contador}</h1>
      <button onClick={() => setContador(contador + 1)}>Incrementar</button>
    </div>
  );
}

export default App;
```

80.11 Frontend vs. Backend

Definición:

Frontend: Es la parte de la aplicación con la que interactúan los usuarios. Incluye todo lo que se ve en la pantalla: diseño, contenido y la lógica de interactividad (HTML, CSS, JavaScript, React).

Backend: Es la parte "invisible" que maneja los datos y la lógica del servidor. Se encarga de procesar las solicitudes de los usuarios, interactuar con bases de datos y enviar respuestas al frontend (Node.js, Python, Ruby, PHP).

Ejemplo de productos/servicios:

Amazon (Frontend): La página que ves cuando compras algo, donde se muestran los productos, el carrito y el proceso de pago.

Amazon (Backend): Los sistemas que gestionan los inventarios, las transacciones y la autenticación de usuarios.

80.12 API y API REST

Definición:

API (Interfaz de Programación de Aplicaciones): Es un conjunto de reglas y herramientas que permiten que diferentes aplicaciones se comuniquen entre sí. En la web, las APIs permiten que el frontend y el backend intercambien datos.

API REST (Representational State Transfer): Es un estilo arquitectónico para diseñar APIs que usan HTTP y siguen principios como la claridad en las URLs, el uso de métodos HTTP (GET, POST, PUT, DELETE) y la estructura de datos en formato JSON.

Ejemplo de productos/servicios:

Twitter API: Permite a las aplicaciones externas interactuar con Twitter, como publicar tuits o leer mensajes.

Spotify API: Permite integrar las funciones de Spotify (como reproducir canciones) en otras aplicaciones.

Ejemplo de llamada a una API REST:

Listing 80.6 JAVASCRIPT

```javascript
fetch('https://api.ejemplo.com/data')
  .then(response => response.json())
  .then(data => console.log(data));

// Ejemplo de respuesta:

// {
//   "nombre": "Juan",
//   "edad": 30
// }
```

81 Extra

81.1 Server Site Rendering:

Definición:

81.2 Conclusión

La web es un sistema de documentos interconectados que se acceden a través de Internet. Los navegadores web permiten acceder a la web y mostrar páginas creadas con HTML, CSS y JavaScript. Node.js permite ejecutar JavaScript en el lado del servidor, mientras que React facilita la creación de interfaces de usuario interactivas. Las APIs y las APIs REST permiten que las aplicaciones se comuniquen entre sí y compartan datos.

82 Unidad 8: Recursos

Listing 82.1 QUARTO-TITLE-BLOCK

83 Recursos De La Unidad 8

Recurso	Enlace
Presentacion (Reveal.js)	Unidad 8
Practica	Practica Unidad 8
Laboratorio	(Pendiente)

Part X

Unidad 9: Apache (Servidor Web)

84 Unidad 9.1: Introduccion a Apache

Listing 84.1 `QUARTO-TITLE-BLOCK`

Instalacion, servicio y prueba rapida

85 Unidad 9.1: Introduccion a Apache

85.1 Introduccion

Apache HTTP Server es uno de los servidores web mas usados. En esta unidad lo instalaremos, lo validaremos con evidencia, y aprenderemos las ubicaciones clave de configuracion.

85.2 Objetivos de aprendizaje

- Instalar Apache en Ubuntu Server
- Validar servicio, puerto y respuesta HTTP
- Identificar rutas de configuracion y document root

85.3 Instalacion (Ubuntu)

Listing 85.1 BASH

```
$ sudo apt update                                                    ①


$ sudo apt install -y apache2                                        ②
Reading package lists... Done
```

---

① **apt update** actualiza el indice de paquetes.
② **apt install apache2** instala el servidor Apache.

Listing 85.2 BASH

```
$ systemctl status apache2 --no-pager                                    ①
  apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled
     Active: active (running)


$ sudo ss -lntp | grep ':80'                                             ②
LISTEN 0 511 0.0.0.0:80 0.0.0.0:* users:("apache2",pid=1234,fd=4)
```

85.4 Validar servicio y puerto

① **systemctl status apache2** confirma que el servicio esta activo.
② **ss -lntp** valida que el puerto 80/tcp esta en escucha.

85.5 Probar respuesta HTTP (local)

Listing 85.3 BASH

```
$ curl -I http://localhost                                               ①
HTTP/1.1 200 OK
Server: Apache/2.4.58 (Ubuntu)
Content-Type: text/html
```

① **curl -I** prueba conectividad HTTP y muestra headers.

85.6 Rutas clave en Ubuntu

- Config principal: `/etc/apache2/apache2.conf`
- Sitios: `/etc/apache2/sites-available/` y `/etc/apache2/sites-enabled/`
- Modulos: `/etc/apache2/mods-available/` y `/etc/apache2/mods-enabled/`
- Document root por defecto: `/var/www/html`
- Logs: `/var/log/apache2/access.log` y `/var/log/apache2/error.log`

85.7 Ejemplos practicos multi-SO

85.7.1 Ejemplo 1: Probar un servidor HTTP local

85.7.1.1 Linux

Listing 85.4 BASH

```
$ curl -I http://localhost                                    ①
HTTP/1.1 200 OK
Server: Apache/2.4.58 (Ubuntu)
```

① **curl -I** valida respuesta HTTP.

85.7.1.2 macOS

Listing 85.5 BASH

```
$ curl -I http://localhost                                    ①
curl: (7) Failed to connect to localhost port 80: Connection refused
```

① **curl -I** funciona igual; si no hay servidor en 80, vera “Connection refused”.

85.7.1.3 Windows

Listing 85.6 POWERSHELL

```
PS> curl.exe -I http://localhost                              ①
HTTP/1.1 200 OK
Server: Apache/2.4.58 (Ubuntu)
```

① **curl.exe** en Windows permite pruebas HTTP simples (headers).

85.8 Mejores practicas

- Valida siempre 3 cosas: servicio -> puerto -> respuesta HTTP.
- Cambios de config: `apachectl configtest` antes de recargar.

86 Unidad 9.2: Configuracion, logs y modulos

Listing 86.1 `QUARTO-TITLE-BLOCK`

Donde se configura Apache en Ubuntu

87 Unidad 9.2: Configuracion, logs y modulos

87.1 Introduccion

Operar Apache significa saber leer logs, validar configuracion, y entender como se habilitan sitios y modulos.

87.2 Objetivos de aprendizaje

- Encontrar y leer logs de Apache
- Validar configuracion antes de recargar
- Habilitar/deshabilitar modulos y sitios

87.3 Validar configuracion

Listing 87.1 BASH

```
$ sudo apachectl configtest                                    ①
Syntax OK


$ sudo systemctl reload apache2                                ②
```

---

① **apachectl configtest** valida sintaxis antes de recargar.
② **systemctl reload** recarga sin cortar conexiones (cuando aplica).

Listing 87.2 BASH

```
$ sudo tail -n 20 /var/log/apache2/error.log                                    ①
[Mon Feb 02 10:20:01.123456 2026] [mpm_prefork:notice] [pid 1234] AH00163: Apache/2.4.58


$ sudo tail -n 5 /var/log/apache2/access.log                                    ②
127.0.0.1 - - [02/Feb/2026:10:20:11 +0000] "HEAD / HTTP/1.1" 200 0 "-" "curl/8.1.0"
```

87.4 Logs

① **tail error.log** muestra errores/avisos (muy util tras cambios).
② **tail access.log** muestra requests HTTP.

87.5 Modulos

Listing 87.3 BASH

```
$ sudo apache2ctl -M | head -n 8                                                ①
Loaded Modules:
 core_module (static)
 so_module (static)
 http_module (static)
 mpm_prefork_module (shared)


$ sudo a2enmod headers                                                          ②
Enabling module headers.
To activate the new configuration, you need to run:
  systemctl restart apache2
```

① **apache2ctl -M** lista modulos cargados.
② **a2enmod** habilita un modulo y escribe symlinks en `mods-enabled/`.

Listing 87.4 BASH

```
$ ls -1 /etc/apache2/sites-available                                    ①
000-default.conf
default-ssl.conf


$ sudo a2ensite 000-default.conf                                        ②
Site 000-default already enabled
```

87.6 Sitios

① **sites-available** contiene definiciones de VirtualHost.
② **a2ensite** habilita un sitio (symlink a `sites-enabled/`).

88 Unidad 9.3: VirtualHosts y sitios

Listing 88.1 `QUARTO-TITLE-BLOCK`

Multiples sitios en un solo servidor

89 Unidad 9.3: VirtualHosts y sitios

89.1 Introduccion

Un VirtualHost permite servir multiples sitios (dominios) desde el mismo Apache. La base operacional es: document root -> vhost -> enable -> configtest -> reload.

89.2 Objetivos de aprendizaje

- Crear un sitio simple con DocumentRoot propio
- Habilitar el sitio y deshabilitar el default
- Validar con curl usando Host header

89.3 Crear un DocumentRoot de ejemplo

Listing 89.1 BASH

```
$ sudo mkdir -p /var/www/site1/public                                    ①


$ echo '<h1>Site1 OK</h1>' | sudo tee /var/www/site1/public/index.html >/dev/null ②
```

---

① **mkdir -p** crea el directorio del sitio.
② **tee** escribe un index.html con permisos root.

89.4 Crear VirtualHost

① **tee + heredoc** crea el archivo del VirtualHost.

Listing 89.2 BASH

```
$ sudo tee /etc/apache2/sites-available/site1.conf >/dev/null <<'EOF' # <1>
<VirtualHost *:80>
  ServerName site1.local
  DocumentRoot /var/www/site1/public

  ErrorLog ${APACHE_LOG_DIR}/site1-error.log
  CustomLog ${APACHE_LOG_DIR}/site1-access.log combined

  <Directory /var/www/site1/public>
    Require all granted
  </Directory>
</VirtualHost>
EOF
                                                                    ①
```

89.5 Habilitar sitio y validar

Listing 89.3 BASH

```
$ sudo a2ensite site1.conf                                          ①
Enabling site site1.


$ sudo a2dissite 000-default.conf                                   ②
Site 000-default disabled.


$ sudo apachectl configtest                                         ③
Syntax OK


$ sudo systemctl reload apache2                                     ④
```

① **a2ensite** habilita el vhost.
② **a2dissite** deshabilita el sitio por defecto.
③ **apachectl configtest** valida sintaxis.
④ **systemctl reload** aplica cambios.

89.6 Probar con Host header

Listing 89.4 BASH

```
$ curl -H 'Host: site1.local' http://127.0.0.1                    ①
<h1>Site1 OK</h1>
```

① **curl -H 'Host: ...'** simula un request con nombre de host para seleccionar el VirtualHost.

90 Unidad 9.4: Seguridad basica en Apache

Listing 90.1 `QUARTO-TITLE-BLOCK`

Permisos, headers y superficie minima

91 Unidad 9.4: Seguridad basica en Apache

91.1 Introduccion

Hardening basico en Apache consiste en: exponer lo minimo, evitar configuraciones inseguras, y dejar evidencias (headers/logs/configtest).

91.2 Objetivos de aprendizaje

- Reducir informacion expuesta
- Habilitar headers basicos de seguridad
- Verificar con curl

91.3 Quitar informacion innecesaria (ServerTokens/ServerSignature)

En Ubuntu, puedes definir directivas globales en `/etc/apache2/conf-available/`.

Listing 91.1 BASH

```
$ sudo tee /etc/apache2/conf-available/security-hardening.conf >/dev/null <<'EOF' # <1>
ServerTokens Prod
ServerSignature Off
EOF
                                                                                ①

$ sudo a2enconf security-hardening                                              ②
Enabling conf security-hardening.


$ sudo apachectl configtest                                                     ③
Syntax OK


$ sudo systemctl reload apache2                                                 ④
```

---

① **tee + heredoc** crea un archivo de configuracion global.
② **a2enconf** habilita configuraciones en `conf-enabled/`.
③ **apachectl configtest** valida sintaxis.
④ **systemctl reload** aplica cambios.

91.4 Headers basicos

Listing 91.2 BASH

```
$ sudo a2enmod headers                                                    ①
Enabling module headers.


$ sudo tee /etc/apache2/conf-available/security-headers.conf >/dev/null <<'EOF' # <2>
<IfModule mod_headers.c>
  Header always set X-Content-Type-Options "nosniff"
  Header always set X-Frame-Options "SAMEORIGIN"
  Header always set Referrer-Policy "strict-origin-when-cross-origin"
</IfModule>
EOF
                                                                          ②

$ sudo a2enconf security-headers                                          ③
Enabling conf security-headers.


$ sudo apachectl configtest                                               ④
Syntax OK


$ sudo systemctl reload apache2                                           ⑤
```

---

① **a2enmod headers** habilita mod_headers.
② **tee + heredoc** define headers basicos.
③ **a2enconf** habilita la configuracion.
④ **apachectl configtest** valida sintaxis.
⑤ **systemctl reload** aplica cambios.

Listing 91.3 BASH

```
$ curl -I http://localhost | sed -n '1,15p'                                    ①
HTTP/1.1 200 OK
Server: Apache
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Referrer-Policy: strict-origin-when-cross-origin
```

91.5 Verificar headers

① **curl -I** verifica headers de respuesta.

91.6 Mejores practicas

- Configtest antes de cada reload.
- Logs siempre: si no esta en logs, no paso.

92 Unidad 9: Recursos

Listing 92.1 `QUARTO-TITLE-BLOCK`

Apache (Servidor Web)

93 Unidad 9: Recursos

93.1 Introduccion

Referencias y comandos clave para operar Apache.

93.2 Comandos clave (cheatsheet)

- Servicio: `systemctl status apache2`, `systemctl reload apache2`, `systemctl restart apache2`
- Config: `apachectl configtest`, `apache2ctl -M`
- Sitios/modulos: `a2ensite`, `a2dissite`, `a2enmod`, `a2dismod`, `a2enconf`, `a2disconf`
- Pruebas: `curl -I http://localhost`, `curl -H 'Host: ...' http://127.0.0.1`

93.3 Referencias

- Apache HTTP Server Documentation
- Ubuntu Server docs - Install Apache2

Part XI

Unidad 10: Nginx y SSL

94 Unidad 10.1: Introduccion a Nginx

Listing 94.1 `QUARTO-TITLE-BLOCK`

Instalacion y conceptos operativos

95 Unidad 10.1: Introduccion a Nginx

95.1 Introduccion

Nginx se usa mucho como servidor web y reverse proxy. En servidores, lo tipico es: Nginx en 80/443, aplicaciones atras (Node, Python, etc.), y certificados TLS con Certbot.

95.2 Objetivos de aprendizaje

- Instalar Nginx en Ubuntu Server
- Validar servicio, puertos y respuesta HTTP
- Identificar ubicaciones clave de configuracion

95.3 Instalacion (Ubuntu)

Listing 95.1 BASH

```
$ sudo apt update                                                    ①


$ sudo apt install -y nginx                                          ②
Reading package lists... Done
```

---

① **apt update** actualiza el indice de paquetes.
② **apt install nginx** instala Nginx.

95.4 Validar servicio y puertos

① **systemctl status nginx** confirma que el servicio esta activo.
② **ss -lntp** valida puertos en escucha (80/443).

Listing 95.2 BASH

```
$ systemctl status nginx --no-pager                                    ①
 nginx.service - A high performance web server and a reverse proxy server
    Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
    Active: active (running)


$ sudo ss -lntp | grep -E ':(80|443)\b'                                ②
LISTEN 0 511 0.0.0.0:80  0.0.0.0:* users:("nginx",pid=2345,fd=6)
```

95.5 Probar respuesta HTTP

Listing 95.3 BASH

```
$ curl -I http://localhost                                             ①
HTTP/1.1 200 OK
Server: nginx/1.24.0 (Ubuntu)
Content-Type: text/html
```

① **curl -I** valida respuesta HTTP de Nginx.

95.6 Rutas clave en Ubuntu

- Config principal: `/etc/nginx/nginx.conf`
- Sitios: `/etc/nginx/sites-available/` y `/etc/nginx/sites-enabled/`
- Default site: `/etc/nginx/sites-available/default`
- Web root comun: `/var/www/html`
- Logs: `/var/log/nginx/access.log` y `/var/log/nginx/error.log`

96 Unidad 10.2: Reverse proxy

Listing 96.1 `QUARTO-TITLE-BLOCK`

Nginx como puerta de entrada a tu aplicacion

97 Unidad 10.2: Reverse proxy

97.1 Introduccion

Un reverse proxy recibe trafico HTTP/HTTPS y lo reenvia a un servicio interno (por ejemplo, una app escuchando en `127.0.0.1:3000`).

97.2 Objetivos de aprendizaje

- Configurar un server block simple
- Proxyear a un backend local
- Validar con curl y logs

97.3 Crear un server block (sitio)

Este ejemplo proxy a `http://127.0.0.1:3000`.

① **tee + heredoc** crea el server block.
② **ln -sf** habilita el sitio (symlink a sites-enabled).
③ **nginx -t** valida sintaxis y configuracion.
④ **systemctl reload nginx** aplica cambios.

97.4 Probar con Host header

① **curl -H Host** prueba el server block; si el backend no existe, veras 502 (esto es evidencia de que Nginx esta encaminando).

RECOMENDACION

Para el laboratorio real, levanta un backend primero (por ejemplo un contenedor o una app en 3000) y repite el curl hasta ver 200.
Casos de uso: - Node/FastAPI/Django/Golang detras de Nginx.
Cuando aplicar: - Cuando necesitas TLS, rate limiting o logs frontales.

Listing 97.1 BASH

```
$ sudo tee /etc/nginx/sites-available/app.conf >/dev/null <<'EOF' # <1>
server {
  listen 80;
  server_name app.local;

  location / {
    proxy_pass http://127.0.0.1:3000;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
  }
}
EOF
                                                                    ①

$ sudo ln -sf /etc/nginx/sites-available/app.conf /etc/nginx/sites-enabled/app.conf ②


$ sudo nginx -t                                                     ③
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful


$ sudo systemctl reload nginx                                       ④
```

Listing 97.2 BASH

```
$ curl -i -H 'Host: app.local' http://127.0.0.1/                    ①
HTTP/1.1 502 Bad Gateway
Server: nginx/1.24.0 (Ubuntu)
```

98 Unidad 10.3: SSL con Certbot

Listing 98.1 `QUARTO-TITLE-BLOCK`

HTTPS en Nginx con Let's Encrypt

99 Unidad 10.3: SSL con Certbot

99.1 Introduccion

Certbot automatiza la emision y renovacion de certificados TLS (Let's Encrypt). Para que funcione, necesitas un dominio apuntando a tu servidor y el puerto 80/443 accesible desde Internet.

> ⚠ ADVERTENCIA CRITICA
>
> **No podras emitir un certificado real si tu servidor no es accesible publicamente.**
> **Lo que podria salir mal:** - DNS del dominio no apunta al servidor. - Firewall/router bloquea 80/443. - Estas en una VM sin NAT/port-forward.
> **Como prevenirlo:** 1. Valida DNS antes: `dig +short tu-dominio.com`. 2. Abre 80/443 en UFW. 3. Prueba HTTP publico antes de ejecutar Certbot.

99.2 Objetivos de aprendizaje

- Instalar Certbot para Nginx
- Emitir certificado (cuando aplica)
- Validar renovacion y configuración generada

99.3 Instalacion

Listing 99.1 BASH

```
$ sudo apt update                                          ①


$ sudo apt install -y certbot python3-certbot-nginx       ②
```

---

① **apt update** actualiza el indice.
② **apt install certbot** instala Certbot y el plugin para Nginx.

99.4 Emitir certificado (modo Nginx)

Listing 99.2 BASH

```
$ sudo certbot --nginx -d tu-dominio.com                                    ①
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Congratulations! Your certificate and chain have been saved at:
  /etc/letsencrypt/live/tu-dominio.com/fullchain.pem
```

① **certbot** –**nginx** edita/crea bloques TLS en Nginx y obtiene el certificado.

99.5 Validar renovacion

Listing 99.3 BASH

```
$ sudo certbot renew --dry-run                                              ①
Congratulations, all simulated renewals succeeded.
```

① **certbot renew** –**dry-run** prueba renovacion sin consumir limites.

100 Unidad 10.4: Hardening basico y headers

Listing 100.1 `QUARTO-TITLE-BLOCK`

Buenas practicas iniciales en Nginx

101 Unidad 10.4: Hardening basico y headers

101.1 Introduccion

Una configuracion segura no es solo “tener HTTPS”. Tambien importa reducir informacion expuesta y agregar headers basicos.

101.2 Objetivos de aprendizaje

- Agregar headers basicos de seguridad
- Validar con curl

101.3 Headers basicos (server block)

Ejemplo para un sitio en `/etc/nginx/sites-available/app.conf`.

Listing 101.1 `NGINX`

```
add_header X-Content-Type-Options "nosniff" always; # <1>
add_header X-Frame-Options "SAMEORIGIN" always;
add_header Referrer-Policy "strict-origin-when-cross-origin" always;
 # <1>
```

---

① **add_header ... always** agrega headers en respuestas, incluyendo errores.

101.4 Validar sintaxis y recargar

① **nginx -t** valida la configuracion.
② **systemctl reload nginx** aplica cambios.
③ **curl -I** verifica headers.

Listing 101.2 BASH

```
$ sudo nginx -t                                                          ①
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful


$ sudo systemctl reload nginx                                            ②


$ curl -I http://localhost | sed -n '1,20p'                              ③
HTTP/1.1 200 OK
Server: nginx
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Referrer-Policy: strict-origin-when-cross-origin
```

102 Unidad 10: Recursos

Listing 102.1 `QUARTO-TITLE-BLOCK`

Nginx y SSL

103 Unidad 10: Recursos

103.1 Comandos clave (cheatsheet)

- Servicio: `systemctl status nginx`, `systemctl reload nginx`, `systemctl restart nginx`
- Config: `nginx -t`
- Logs: `tail -f /var/log/nginx/error.log`, `tail -f /var/log/nginx/access.log`
- SSL: `certbot --nginx -d dominio`, `certbot renew --dry-run`

103.2 Referencias

- Nginx Documentation
- Certbot Documentation

Part XII

Unidad 11: MariaDB

104 Unidad 11.1: Instalacion y seguridad inicial

Listing 104.1 `QUARTO-TITLE-BLOCK`

MariaDB en servidores

105 Unidad 11.1: Instalacion y seguridad inicial

105.1 Introduccion

MariaDB es una base de datos relacional compatible con MySQL, muy usada en servidores Linux. La operacion profesional empieza con una instalacion limpia, un servicio estable y un hardening inicial.

105.2 Objetivos de aprendizaje

- Instalar MariaDB
- Validar servicio y socket/puerto
- Ejecutar hardening inicial con `mysql_secure_installation`

105.3 Instalacion (Ubuntu)

Listing 105.1 BASH

```
$ sudo apt update                                                    ①


$ sudo apt install -y mariadb-server                                 ②
Reading package lists... Done
```

---

① **apt update** actualiza el indice de paquetes.
② **apt install mariadb-server** instala servidor MariaDB.

Listing 105.2 BASH

```
$ systemctl status mariadb --no-pager                                    ①
 mariadb.service - MariaDB 10.11.6 database server
     Active: active (running)


$ sudo ss -lntp | grep ':3306' || true                                   ②


$ sudo ls -l /run/mysqld/mysqld.sock                                     ③
srwxrwxrwx 1 mysql mysql 0 Feb  2 12:30 /run/mysqld/mysqld.sock
```

105.4 Validar servicio y socket

① **systemctl status mariadb** valida que el servicio esta corriendo.
② **ss -lntp** valida el puerto 3306 si esta habilitado (en muchos casos MariaDB escucha solo en loopback).
③ **mysqld.sock** confirma socket local, comun en instalaciones seguras.

105.5 Hardening inicial

Listing 105.3 BASH

```
$ sudo mysql_secure_installation                                         ①
... (preguntas interactivas) ...
```

① **mysql_secure_installation** aplica endurecimiento basico (password root, eliminar usuarios anonimos, deshabilitar login remoto de root, etc.).

105.6 Prueba rapida

① **mariadb -e** ejecuta un query y valida que puedes conectarte localmente.

Listing 105.4 BASH

```
$ sudo mariadb -e 'SELECT VERSION();'                                    ①
VERSION()
10.11.6-MariaDB-0ubuntu0.24.04.1
```

106 Unidad 11.2: Usuarios y privilegios

Listing 106.1 `QUARTO-TITLE-BLOCK`

Operar acceso de forma segura

107 Unidad 11.2: Usuarios y privilegios

107.1 Introduccion

La base de seguridad en bases de datos es: usuarios por aplicacion, privilegios minimos, y acceso restringido por host.

107.2 Objetivos de aprendizaje

- Crear base de datos y usuario
- Otorgar privilegios minimos
- Validar permisos

107.3 Crear DB y usuario

Listing 107.1 BASH

```
$ sudo mariadb                                                                    ①
MariaDB [(none)]> CREATE DATABASE appdb;
MariaDB [(none)]> CREATE USER 'appuser'@'localhost' IDENTIFIED BY 'AppPass-ChangeMe';
MariaDB [(none)]> GRANT SELECT,INSERT,UPDATE,DELETE ON appdb.* TO 'appuser'@'localhost';
MariaDB [(none)]> FLUSH PRIVILEGES;
MariaDB [(none)]> EXIT;
```

---

① **mariadb** abre consola SQL para crear DB/usuario y aplicar permisos.

> ⚠ ADVERTENCIA CRITICA
>
> **No uses passwords debiles ni reutilices credenciales.**
> **Lo que podria salir mal:** - Compromiso de base de datos por credenciales triviales.
> **Como prevenirlo:** 1. Usa passwords largos y un gestor. 2. Usa usuarios distintos por app/entorno.

107.4 Validar acceso

Listing 107.2 BASH

```
$ mariadb -u appuser -p -e 'SHOW DATABASES;'                    ①
Enter password:
Database
information_schema
appdb
```

① **mariadb -u ... -p** valida que el usuario puede conectarse y ver su DB.

108 Unidad 11.3: Backup y restore

Listing 108.1 `QUARTO-TITLE-BLOCK`

mysqldump como evidencia operativa

109 Unidad 11.3: Backup y restore

109.1 Introduccion

Un servidor sin backups verificables no esta en produccion. Aqui usaremos `mysqldump` para generar un backup y restaurarlo en un entorno de prueba.

109.2 Objetivos de aprendizaje

- Generar backup con `mysqldump`
- Restaurar y validar

109.3 Backup

Listing 109.1 BASH

```
$ sudo mysqldump --databases appdb > /tmp/appdb.sql                    ①


$ ls -lh /tmp/appdb.sql                                                ②
-rw-r--r-- 1 root root 12K Feb  2 12:45 /tmp/appdb.sql
```

---

① **mysqldump** exporta la base `appdb` a un archivo SQL.
② **ls -lh** valida que el backup existe y su tamano.

109.4 Restore (ejemplo)

① **mariadb < backup.sql** restaura desde el dump.
② **SHOW DATABASES** valida que la base existe.

Listing 109.2 BASH

```
$ sudo mariadb < /tmp/appdb.sql                                    ①


$ sudo mariadb -e 'SHOW DATABASES;' | grep appdb                   ②
appdb
```

110 Unidad 11.4: Red y acceso remoto

Listing 110.1 `QUARTO-TITLE-BLOCK`

bind-address, firewall y buenas practicas

111 Unidad 11.4: Red y acceso remoto

111.1 Introduccion

Exponer una base de datos a red debe ser una decision consciente. En la mayoria de escenarios, la DB no debe ser publica y solo debe ser accesible desde una red privada o desde el mismo host.

111.2 Objetivos de aprendizaje

- Identificar direccion/puerto en escucha
- Configurar bind-address (cuando aplica)
- Proteger con firewall

111.3 Ver escucha

Listing 111.1 BASH

```
$ sudo ss -lntp | grep ':3306' || true                                    ①
```

---

① **ss -lntp** valida si MariaDB escucha en 3306/tcp.

111.4 bind-address (ejemplo en Ubuntu)

En MariaDB, el archivo suele estar en `/etc/mysql/mariadb.conf.d/50-server.cnf`.

① **grep bind-address** muestra si esta limitado a loopback.

Listing 111.2 BASH

```
$ sudo grep -n '^bind-address' /etc/mysql/mariadb.conf.d/50-server.cnf || true ①
bind-address            = 127.0.0.1
```

RECOMENDACION

Si tu app y DB estan en el mismo servidor, manten bind-address en 127.0.0.1.
Casos de uso: - Stack monolitico (Nginx + app + DB en el mismo host).
Cuando aplicar: - Cuando no necesitas conexiones desde otra maquina.

111.5 Firewall (UFW)

Listing 111.3 BASH

```
$ sudo ufw status                                                    ①
Status: active


$ sudo ufw allow from 192.168.56.0/24 to any port 3306 proto tcp     ②
Rule added
```

① **ufw status** valida estado del firewall.
② **ufw allow from ...** permite 3306 solo desde una red especifica.

112 Unidad 11: Recursos

Listing 112.1 `QUARTO-TITLE-BLOCK`

MariaDB

113 Unidad 11: Recursos

113.1 Comandos clave (cheatsheet)

- Servicio: `systemctl status mariadb`, `systemctl restart mariadb`
- Cliente: `mariadb`, `mariadb -e 'SELECT 1;'`
- Seguridad: `mysql_secure_installation`
- Backup: `mysqldump --databases db > backup.sql`, `mariadb < backup.sql`
- Red: `ss -lntp | grep 3306`, `ufw allow from ... to any port 3306`

113.2 Referencias

- MariaDB Server Documentation
- mysqldump

Part XIII

Unidad 12: Diagnostico y Troubleshooting

114 Unidad 12.1: Metodologia de diagnostico

Listing 114.1 `QUARTO-TITLE-BLOCK`

Orden, evidencias y reproducibilidad

115 Unidad 12.1: Metodologia de diagnostico

115.1 Introduccion

Troubleshooting no es “probar cosas” al azar. Es un proceso: observar, recolectar evidencias, aislar la causa y validar el arreglo.

115.2 Objetivos de aprendizaje

- Usar un checklist de diagnostico
- Documentar evidencias minimas
- Evitar cambios destructivos sin rollback

115.3 Flujo recomendado

1. Sintoma: que falla, desde donde, desde cuando.
2. Alcance: un host o todos, un usuario o todos, una red o todas.
3. Evidencias: comando + salida + timestamp.
4. Hipotesis: 1-2 causas probables.
5. Prueba: un cambio controlado.
6. Validacion: mismo comando/metricas para confirmar.

115.4 Kit de evidencias minimo

① **hostnamectl** registra host/OS para contexto.
② **uptime** muestra uptime y carga.
③ **ip -br addr** muestra IP por interfaz.

Listing 115.1 BASH

```
$ hostnamectl                                                                    ①
 Static hostname: srv01
 Operating System: Ubuntu 24.04.1 LTS


$ uptime                                                                         ②
 12:55:03 up 1 day,  3:21,  1 user,  load average: 0.08, 0.11, 0.09


$ ip -br addr                                                                    ③
lo     UNKNOWN 127.0.0.1/8 ::1/128
enp0s3  UP      192.168.56.10/24
```

RECOMENDACION

Antes de cambiar algo, guarda evidencia.
Casos de uso: - Incidentes de produccion.
Cuando aplicar: - Siempre que diagnosticas una caida o degradacion.

116 Unidad 12.2: Logs y journalctl

Listing 116.1 `QUARTO-TITLE-BLOCK`

Diagnostico basado en evidencia

117 Unidad 12.2: Logs y journalctl

117.1 Introduccion

En Linux, muchas respuestas estan en logs. `journalctl` permite filtrar por servicio, tiempo y severidad.

117.2 Objetivos de aprendizaje

- Ver logs de arranque
- Ver logs de un servicio
- Seguir logs en tiempo real

117.3 Journal del sistema

Listing 117.1 BASH

```
$ sudo journalctl -b -n 50 --no-pager                                    ①
... logs del ultimo arranque ...
```

---

① **journalctl -b** muestra logs del boot actual.

117.4 Logs de un servicio

① **journalctl -u** filtra por unidad (servicio).
② **-f** sigue logs en tiempo real.

Listing 117.2 BASH

```
$ sudo journalctl -u ssh --since "1 hour ago" --no-pager                    ①
... logs de ssh ...


$ sudo journalctl -u nginx -f                                               ②
... follow logs ...
```

118 Unidad 12.3: Red, puertos y DNS

Listing 118.1 `QUARTO-TITLE-BLOCK`

Diagnostico rapido de conectividad

119 Unidad 12.3: Red, puertos y DNS

119.1 Introduccion

Cuando “no responde”, casi siempre es una de estas: IP/ruta, firewall, puerto no escuchando, o DNS.

119.2 Objetivos de aprendizaje

- Validar IP y rutas
- Verificar puertos
- Probar DNS y HTTP

119.3 Checklist rapido

① **ip -br addr** valida IP.
② **ip route** valida gateway/rutas.
③ **ss -lntp** valida puertos.
④ **resolvectl status** valida DNS efectivo.
⑤ **curl -I** valida respuesta HTTP local.

Listing 119.1 BASH

```
$ ip -br addr                                                    ①
enp0s3  UP  192.168.56.10/24


$ ip route | head -n 5                                           ②
default via 192.168.56.1 dev enp0s3


$ sudo ss -lntp                                                  ③
... puertos en escucha ...


$ resolvectl status | sed -n '1,40p'                             ④
... DNS efectivo ...


$ curl -I http://localhost                                       ⑤
HTTP/1.1 200 OK
```

120 Unidad 12.4: CPU, memoria y storage

Listing 120.1 `QUARTO-TITLE-BLOCK`

Diagnostico de saturacion y espacio

121 Unidad 12.4: CPU, memoria y storage

121.1 Introduccion

Muchas caidas son por saturacion (CPU/RAM) o por disco lleno (logs, backups, cache).

121.2 Objetivos de aprendizaje

- Identificar saturacion
- Encontrar directorios grandes
- Validar espacio disponible

121.3 CPU y memoria

Listing 121.1 BASH

```
$ uptime                                                                    ①
 12:55:03 up 1 day,  3:21,  1 user,  load average: 0.08, 0.11, 0.09


$ free -h                                                                   ②
               total        used        free      shared  buff/cache   available
Mem:           3.8Gi       1.1Gi       2.0Gi        52Mi       740Mi       2.5Gi
Swap:          2.0Gi          0B       2.0Gi
```

---

① **uptime** indica carga y uptime.
② **free -h** muestra uso de RAM.

121.4 Disco y directorios grandes

① **df -h** valida espacio por filesystem.
② **du + sort** ayuda a encontrar directorios grandes.

Listing 121.2 BASH

```
$ df -h                                                                    ①
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1        49G  5.4G   41G  12% /


$ sudo du -xh /var | sort -h | tail -n 10                                  ②
... mayores consumos ...
```

122 Unidad 12: Recursos

Listing 122.1 `QUARTO-TITLE-BLOCK`

Diagnostico y troubleshooting

123 Unidad 12: Recursos

123.1 Comandos clave (cheatsheet)

- Contexto: `hostnamectl`, `uptime`, `date`
- Logs: `journalctl -b`, `journalctl -u <svc> --since ...`, `journalctl -f`
- Red: `ip -br addr`, `ip route`, `ss -lntp`, `resolvectl status`, `curl -I`
- Recursos: `free -h`, `df -h`, `du -xh /var | sort -h | tail`

Part XIV

Laboratorios Prácticos

124 Laboratorios Prácticos - Administración de Servidores Linux

Curso: Administración de Servidores Linux
Para: Abacom
Instructor: Diego Saavedra

124.1 Índice de Laboratorios

124.1.1 Laboratorio 0: Diagnóstico del Sistema

- **Unidad:** 1 - Introducción a Linux
- **Duración:** 45-60 minutos
- **Dificultad:** Principiante
- **Archivo:** `lab0_diagnostico_sistema.md`
- **Objetivos:**
  - Identificar distribución, kernel y recursos del sistema
  - Validar conectividad de red básica

124.1.2 Laboratorio 1: Instalación de Ubuntu Server

- **Unidad:** 2 - Instalación y Configuración
- **Duración:** 90 minutos
- **Dificultad:** Principiante
- **Archivo:** `lab1_instalacion_ubuntu.md`
- **Objetivos:**
  - Instalar Ubuntu Server en VM
  - Configurar particiones
  - Validar acceso SSH

124.1.3 Laboratorio 2: Gestión de Usuarios y Permisos

- **Unidad:** 3 - Comandos Básicos de Linux
- **Duración:** 120 minutos
- **Dificultad:** Intermedio
- **Archivo:** `lab2_usuarios_permisos.md`
- **Objetivos:**
  - Crear y gestionar usuarios
  - Entender sistema de permisos
  - Trabajar con grupos
  - Usar sudo

124.1.4 Laboratorio 2B: Hardening Básico Post-Instalación

- **Unidad:** 2 - Instalación y Configuración
- **Duración:** 90-120 minutos
- **Dificultad:** Intermedio
- **Archivo:** `lab2b_hardening_basico.md`
- **Objetivos:**
  - Actualizar el sistema de forma segura
  - Configurar UFW sin perder acceso SSH
  - Endurecer SSH (sin cambios destructivos)
  - (Opcional) Fail2Ban y unattended-upgrades

124.1.5 Laboratorio 3: Administración de Procesos

- **Unidad:** 5 - Procesos y Servicios
- **Duración:** 90 minutos
- **Dificultad:** Intermedio
- **Archivo:** `lab3_procesos_servicios.md`
- **Objetivos:**
  - Monitorear procesos
  - Gestionar servicios con systemd
  - Tareas programadas (cron)
  - Automatización

124.1.6 Laboratorio 4: Redes y SSH

- **Unidad:** 7 - Redes y Seguridad
- **Duración:** 120 minutos
- **Dificultad:** Intermedio-Avanzado
- **Archivo:** `lab4_redes_ssh.md`
- **Objetivos:**
  - Configuración de redes

  - SSH seguro con claves
  - Firewall básico
  - Diagnóstico de red

124.1.7 Laboratorio 4B: Docker y Docker Compose

- **Unidad:** 4 - Docker y Containerización
- **Duración:** 90-120 minutos
- **Dificultad:** Intermedio
- **Archivo:** `lab4b_docker_compose.md`
- **Objetivos:**
  - Instalar Docker y validar motor
  - Levantar un servicio con Docker Compose
  - Revisar logs, puertos y volúmenes

124.1.8 Laboratorio 5: Almacenamiento (Particiones y Montajes)

- **Unidad:** 6 - Almacenamiento y Sistemas de Archivos
- **Duración:** 90-120 minutos
- **Dificultad:** Intermedio
- **Archivo:** `lab5_almacenamiento_montajes.md`
- **Objetivos:**
  - Inventariar discos/FS (lsblk/df)
  - Crear un disco de laboratorio (loop)
  - Crear filesystem, montar y persistir con fstab
  - Diagnosticar espacio con du

124.1.9 Laboratorio 6: HTTP Básico y Diagnóstico Web

- **Unidad:** 8 - Introducción a Servidores Web
- **Duración:** 60-90 minutos
- **Dificultad:** Principiante-Intermedio
- **Archivo:** `lab6_http_diagnostico_web.md`
- **Objetivos:**
  - Entender request/response con curl
  - Validar DNS, puertos y headers
  - Identificar fallas típicas (DNS/puerto/firewall)

124.1.10 Laboratorio 7: Apache - Sitio y VirtualHost

- **Unidad:** 9 - Apache
- **Duración:** 90-120 minutos
- **Dificultad:** Intermedio
- **Archivo:** `lab7_apache_virtualhost.md`

- **Objetivos:**
  - Instalar Apache y habilitar módulos básicos
  - Crear un VirtualHost y revisar logs
  - Publicar un sitio simple con evidencia

124.1.11 Laboratorio 8: Nginx - Reverse Proxy + TLS (self-signed)

- **Unidad:** 10 - Nginx y SSL
- **Duración:** 120-150 minutos
- **Dificultad:** Intermedio
- **Archivo:** `lab8_nginx_reverseproxy_tls.md`
- **Objetivos:**
  - Configurar server block
  - Hacer reverse proxy hacia un servicio local
  - Habilitar HTTPS con certificado self-signed (o Certbot si tienes dominio)

124.1.12 Laboratorio 9: MariaDB - Usuarios, Privilegios y Backup/Restore

- **Unidad:** 11 - MariaDB
- **Duración:** 120-150 minutos
- **Dificultad:** Intermedio
- **Archivo:** `lab9_mariadb_backup_restore.md`
- **Objetivos:**
  - Asegurar instalación básica
  - Crear DB/usuario con privilegios mínimos
  - Generar y validar backup/restore

124.1.13 Laboratorio 10: Troubleshooting Integrado (Checklist + Evidencias)

- **Unidad:** 12 - Diagnóstico y Troubleshooting
- **Duración:** 120-180 minutos
- **Dificultad:** Intermedio-Avanzado
- **Archivo:** `lab10_troubleshooting_integrado.md`
- **Objetivos:**
  - Resolver incidentes simulados (servicio/puertos/DNS/espacio)
  - Entregar reporte reproducible con evidencias

124.1.14 Laboratorio 11: Clawdbot (OpenClaw) + Telegram + Tailscale (Seguridad)

- **Unidad:** 4/7/12 - Docker, Redes y Seguridad, Troubleshooting
- **Duración:** 150-210 minutos
- **Dificultad:** Avanzado

- **Archivo:** `lab11_clawdbot_openclaw_telegram_tailscale.md`
- **Objetivos:**
  - Levantar OpenClaw en Docker sobre Ubuntu Server LTS
  - Conectar Telegram (BotFather) al gateway
  - Aplicar postura minima de seguridad (pairing, mention gating, allowlists)
  - Restringir administracion mediante Tailscale + UFW

124.2 Evaluación

Cada laboratorio incluye un **Checklist de aceptación** al final. Ese checklist define los criterios mínimos para considerar el lab como completado.

124.3 Recursos Adicionales por Laboratorio

124.3.1 Para Todos los Labs

- Máquina virtual Ubuntu Server LTS configurada (22.04 o 24.04)
- Acceso SSH desde máquina host
- Terminal con bash/zsh
- Permisos sudo

124.3.2 Lab 1

- VirtualBox o similar
- Imagen ISO Ubuntu Server LTS
- 20GB espacio en disco
- 2GB RAM disponible

124.3.3 Lab 2

- Usuario con permisos sudo
- Acceso a archivos del sistema

124.3.4 Lab 3

- Servicios básicos funcionando
- Permisos para crear archivos
- Acceso a logs del sistema

124.3.5 Lab 4

- SSH configurado
- Firewall deshabilitado inicialmente
- Acceso a utilidades de red

124.3.6 Lab 4B

- Docker instalado (o permisos sudo para instalar)
- Acceso a Internet para descargar imágenes

124.4 Instrucciones para Estudiantes

124.4.1 Antes de Comenzar

1. Prepara tu entorno (VM, conexión SSH)
2. Lee el archivo del laboratorio completo
3. Anota cualquier duda antes de empezar
4. Verifica que tienes los requisitos previos

124.4.2 Durante el Laboratorio

1. Sigue los pasos en orden
2. Prueba cada comando antes de continuar
3. Toma notas de lo que aprendes
4. Pregunta si algo no está claro

124.4.3 Después del Laboratorio

1. Completa los ejercicios de práctica
2. Documenta tu trabajo
3. Prepara una presentación si se requiere
4. Entrega antes de la fecha límite

124.5 Cronograma Sugerido

Semana	Laboratorio	Tipo
2-3	Lab 1: Instalación	Práctica Guiada
3-4	Lab 2B: Hardening	Práctica Guiada
5-6	Lab 2: Usuarios	Ejercicio Independiente
6	Lab 4B: Docker	Práctica Guiada + Independiente
6-7	Lab 3: Procesos	Ejercicio Independiente
9-10	Lab 4: Redes	Práctica Guiada + Independiente
12+	Proyecto Final	Integración

124.6 Troubleshooting Común

124.6.1 Problema: No puedo conectar SSH

Soluciones: 1. Verificar que VM está corriendo 2. Verificar IP: `ip a` en la VM 3. Verificar SSH está activo: `sudo systemctl status ssh` 4. Verificar firewall: `sudo ufw status` 5. Usar `ssh -v usuario@ip` para debug

124.6.2 Problema: Permisos "Permission Denied"

Soluciones: 1. Verificar permisos actuales: `ls -la archivo` 2. Usar `chmod` para cambiar permisos 3. Si necesitas sudo: `sudo comando` 4. Verificar propietario: `ls -l | head -1` muestra `drwx------+ alumno alumno`

124.6.3 Problema: Comando no encontrado

Soluciones: 1. Verificar sintaxis exacta 2. Usar `which comando` para ver si está instalado 3. Usar `man comando` para ayuda 4. Si no está instalado: `sudo apt install paquete`

124.7 Entrega de Laboratorios

124.7.1 Qué Entregar

- Documento con pantallazos de pasos completados
- Respuestas a preguntas formuladas
- Ejercicios de práctica completados
- Reflexión sobre lo aprendido

124.7.2 Cómo Entregar

- Email al instructor
- O plataforma de aprendizaje indicada por Abacom
- Nombra archivos: `NombreEstudiante_Lab#.pdf`

124.7.3 Fecha Límite

- Se indicará para cada laboratorio
- Entrega tardía: -10% por día

124.8 Competencias Desarrolladas

124.8.1 Después de Todos los Labs, podrás:

- Instalar y configurar servidores Linux
- Administrar usuarios y permisos
- Gestionar procesos y servicios
- Configurar redes y firewall
- Usar SSH de manera segura
- Automatizar tareas
- Diagnosticar problemas de sistema

Para preguntas o problemas, contacta al instructor: Diego Saavedra

125 Laboratorio 0: Diagnostico Basico del Sistema

Unidad: 1 - Introduccion a Linux
Duracion Estimada: 45-60 minutos
Dificultad: Principiante

125.1 Objetivos

- Identificar distribucion y version de Linux
- Identificar version del kernel
- Verificar recursos basicos del sistema (CPU, RAM, disco)
- Confirmar conectividad de red basica

125.2 Requisitos Previos

- Acceso a una VM o maquina Linux
- Acceso a una terminal con permisos sudo (si aplica)

125.3 Pasos del Laboratorio

125.3.1 Paso 1: Identidad del Sistema (10 min)

Listing 125.1 BASH

```
$ cat /etc/os-release                                                    ①
$ uname -r                                                               ②
$ uname -m                                                               ③
```

① **cat /etc/os-release** muestra distribucion y version.
② **uname -r** muestra la version del kernel.
③ **uname -m** muestra la arquitectura (x86_64/arm64).

Listing 125.2 BASH

```
$ lscpu                                                    ①
$ free -h                                                  ②
```

125.3.2 Paso 2: CPU y Memoria (10 min)

① **lscpu** muestra informacion de CPU.
② **free -h** muestra memoria en formato legible.

125.3.3 Paso 3: Disco y Filesystem (10 min)

Listing 125.3 BASH

```
$ df -h                                                    ①
$ lsblk                                                    ②
```

① **df -h** muestra uso de disco por filesystem.
② **lsblk** muestra discos/particiones y montaje.

125.3.4 Paso 4: Red Basica (10-15 min)

Listing 125.4 BASH

```
$ ip a                                                     ①
$ ip route                                                 ②
$ ping -c 3 8.8.8.8                                        ③
```

① **ip a** muestra interfaces e IP.
② **ip route** muestra la ruta por defecto (gateway).
③ **ping** valida conectividad basica (ICMP).

125.4 Conclusión

Con esta informacion ya puedes operar con seguridad el entorno del curso (y reportar problemas rapido).

125.5 Checklist de aceptación

- ☐ Incluí la salida de `cat /etc/os-release` en mi evidencia
- ☐ Incluí la salida de `uname -r` y `uname -m` en mi evidencia
- ☐ Incluí la salida de `free -h` en mi evidencia
- ☐ Incluí la salida de `df -h` y `lsblk` en mi evidencia
- ☐ Incluí la salida de `ip a`, `ip route` y `ping -c 3 8.8.8.8` en mi evidencia

126 Laboratorio 1: Instalación de Ubuntu Server

Unidad: 2 - Instalación y Configuración
Duración Estimada: 90 minutos
Dificultad: Principiante

126.1 Objetivos

- Instalar Ubuntu Server LTS en máquina virtual
- Configurar particiones de disco
- Realizar configuración inicial del sistema
- Validar acceso SSH

126.2 Requisitos Previos

- VirtualBox instalado
- 20 GB de almacenamiento libre en host
- Imagen ISO de Ubuntu Server LTS descargada
- Conexión de red activa

126.3 Pasos del Laboratorio

126.3.1 Paso 1: Crear Máquina Virtual (15 min)

1. Abre VirtualBox
2. Click en "Nueva"
3. Configura:
    - **Nombre:** Ubuntu-Lab-01
    - **Tipo:** Linux
    - **Versión:** Ubuntu (64-bit)
    - **RAM:** 2048 MB (mínimo)
    - **Disco:** 20 GB, dinámico
4. Haz click "Crear"

126.3.2 Paso 2: Configurar Adaptador de Red (10 min)

1. Selecciona la VM y abre "Configuración"
2. Ir a "Red"
3. Asegúrate que "Adaptador 1" está en "NAT"
4. Click "Aceptar"

126.3.3 Paso 3: Iniciar Instalación (20 min)

1. Inicia la VM
2. Selecciona la imagen ISO de Ubuntu Server LTS
3. Selecciona "Try or Install Ubuntu"
4. Espera a que cargue

126.3.4 Paso 4: Instalación del Sistema (40 min)

1. Selecciona idioma (Español)
2. Configuración de teclado (tu país)
3. Selecciona "Ubuntu Server"
4. Configuración de red:
    - Usa DHCP (automático)
5. Proxy: Dejar en blanco
6. Mirror: Usar default
7. Particionamiento:
    - Selecciona "Use An Entire Disk"
    - Selecciona el disco de 20GB
8. Crea usuario:
    - **Usuario:** alumno
    - **Contraseña:** Lab@2024 (cambiar en producción)
    - **Hostname:** ubuntu-lab-abacom
9. Selecciona "Install OpenSSH Server"
10. Espera a que complete (10-15 min)

126.3.5 Paso 5: Primer Arranque (10 min)

1. Sistema arranca automáticamente
2. Verifica que muestra login prompt
3. Haz login con usuario "alumno"

126.3.6 Paso 6: Verificar Conectividad SSH (15 min)

Desde tu computadora host:

Listing 126.1 BASH

```
# Dentro de la VM
ip a  # Anota la IP asignada (ej: 192.168.1.X)
```

Listing 126.2 BASH

```
ssh alumno@192.168.1.X
# Ingresa la contraseña: Lab@2024
# Si funciona, verás el prompt de ubuntu-lab-abacom
```

126.4 Troubleshooting

Problema	Solución
"No se puede conectar a la red"	Verificar adaptador de red en VirtualBox config
"ISO no se encuentra"	Verificar ruta de ISO en VirtualBox
"Pantalla negra después de boot"	Esperar 2-3 minutos, presionar Enter
"SSH connection refused"	Verificar IP con `ip a`, firewall deshabilitado

126.5 Conclusión

Una vez completado este laboratorio, tienes un servidor Ubuntu funcional con SSH habilitado. Este será tu entorno para los laboratorios siguientes.

126.6 Checklist de aceptación

- ☐ La VM existe y arranca sin errores
- ☐ Puedo iniciar sesión localmente (consola) con el usuario creado
- ☐ `ip a` muestra una IP válida para conectarme por SSH
- ☐ Puedo conectar por SSH desde el host y llegar a un prompt
- ☐ Incluí evidencia de `uname -a` y del acceso SSH (captura o salida)

127 Laboratorio 2B: Hardening Basico Post-Instalacion

Unidad: 2 - Instalación y Configuración
Duración Estimada: 90-120 minutos
Dificultad: Intermedio

127.1 Objetivos

- Actualizar el sistema de forma segura
- Configurar UFW sin perder acceso SSH
- Endurecer SSH (minimo viable)
- (Opcional) Instalar Fail2Ban
- (Opcional) Habilitar unattended-upgrades

127.2 Requisitos Previos

- Lab 1 completado (Ubuntu Server LTS instalado)
- Acceso SSH funcional
- Usuario con permisos sudo

> ⚠ ADVERTENCIA CRÍTICA
>
> Vas a tocar firewall y SSH.
> **Lo que podría salir mal:** - Bloquear tu acceso remoto por SSH.
> **Cómo prevenirlo:** 1. Mantén una sesión SSH abierta mientras pruebas cambios. 2. Permite el puerto SSH en UFW antes de habilitarlo. 3. Valida la sintaxis de sshd antes de reiniciar el servicio.

127.3 Pasos del Laboratorio

127.3.1 Paso 1: Actualizar el sistema (20 min)

① **apt update** actualiza el indice de paquetes.
② **apt upgrade** aplica actualizaciones.
③ **apt autoremove** limpia dependencias no usadas.

Listing 127.1 BASH

```
$ sudo apt update                                                    ①

$ sudo apt upgrade -y                                                ②

$ sudo apt autoremove -y                                             ③
```

127.3.2 Paso 2: UFW seguro (25-35 min)

Primero identifica el puerto SSH actual:

Listing 127.2 BASH

```
$ sudo ss -lntp | grep sshd                                          ①
LISTEN 0 128 0.0.0.0:22 0.0.0.0:* users:("sshd",pid=987,fd=3)
```

① **ss -lntp** evidencia el puerto donde escucha sshd.

Ahora configura UFW:

① **ufw status** muestra estado actual.
② **ufw allow 22/tcp** permite SSH antes de activar firewall.
③ **ufw allow 80/tcp** permite HTTP.
④ **ufw allow 443/tcp** permite HTTPS.
⑤ **ufw enable** activa el firewall.
⑥ **ufw status verbose** valida reglas activas.

127.3.3 Paso 3: Endurecer SSH (25-35 min)

① **cp** crea backup antes de cambios.
② **sshd -t** valida sintaxis (antes/despues).

Edita `/etc/ssh/sshd_config` y aplica minimo:

- `PermitRootLogin no`
- `PasswordAuthentication no` (solo si ya tienes claves)
- `MaxAuthTries 3`

Luego valida y recarga:

Listing 127.3 BASH

```
$ sudo ufw status verbose                                    ①
Status: inactive


$ sudo ufw allow 22/tcp                                      ②


$ sudo ufw allow 80/tcp                                      ③


$ sudo ufw allow 443/tcp                                     ④


$ sudo ufw enable                                            ⑤


$ sudo ufw status verbose                                    ⑥
Status: active
```

Listing 127.4 BASH

```
$ sudo cp /etc/ssh/sshd_config /etc/ssh/sshd_config.bak      ①


$ sudo sshd -t                                               ②
```

① **sshd -t** evita recargar una config invalida.
② **systemctl reload ssh** aplica cambios sin cortar sesiones activas.

127.3.4 Paso 4 (opcional): Fail2Ban (20 min)

① **apt install fail2ban** instala Fail2Ban.
② **systemctl status** valida servicio.

127.3.5 Paso 5 (opcional): Unattended upgrades (20 min)

Listing 127.5 BASH

```
$ sudo sshd -t                                                    ①


$ sudo systemctl reload ssh                                       ②
```

Listing 127.6 BASH

```
$ sudo apt install -y fail2ban                                    ①


$ systemctl status fail2ban --no-pager                            ②
 fail2ban.service - Fail2Ban Service
     Active: active (running)
```

① **apt install unattended-upgrades** instala actualizaciones automaticas.
② **systemctl status** valida servicio.

127.4 Entregables (Evidencia)

- `ufw status verbose` con reglas y estado active
- Evidencia del puerto SSH en escucha (`ss -lntp | grep sshd`)
- `sshd -t` sin errores
- (Opcional) `systemctl status fail2ban --no-pager`
- (Opcional) `systemctl status unattended-upgrades --no-pager`

127.5 Checklist de aceptación

☐ Ejecuté `sudo apt update` y `sudo apt upgrade -y` (evidencia incluida)
☐ UFW quedó activo y permite SSH (evidencia de `ufw status verbose`)
☐ `sshd -t` retorna sin errores antes/después de cambios
☐ Puedo mantener acceso SSH después de aplicar los cambios
☐ (Opcional) Fail2Ban está activo (evidencia de `systemctl status fail2ban --no-pager`)
☐ (Opcional) unattended-upgrades está activo (evidencia de `systemctl status unattended-upgrades --no-pager`)

Listing 127.7 BASH

```
$ sudo apt install -y unattended-upgrades                                    ①


$ sudo systemctl status unattended-upgrades --no-pager                       ②
```

128 Laboratorio 2: Gestión de Usuarios y Permisos

Unidad: 4 - Gestión de Usuarios y Permisos
Duración Estimada: 120 minutos
Dificultad: Intermedio

128.1 Objetivos

- Crear y gestionar usuarios en Linux
- Entender el sistema de permisos (rwx)
- Trabajar con grupos
- Usar sudo para escalación de privilegios

128.2 Requisitos Previos

- Ubuntu del Lab 1 corriendo
- Acceso SSH como "alumno"
- Permisos sudo

128.3 Pasos del Laboratorio

128.3.1 Paso 1: Crear Nuevos Usuarios (20 min)

128.3.2 Paso 2: Establecer Contraseñas (10 min)

128.3.3 Paso 3: Crear Grupos (15 min)

128.3.4 Paso 4: Entender Permisos Básicos (25 min)

128.3.5 Paso 5: Permisos Avanzados - Directorios (20 min)

Listing 128.1 BASH

```
# Desde terminal con sudo
sudo useradd -m -s /bin/bash carlos
sudo useradd -m -s /bin/bash maria
sudo useradd -m -s /bin/bash admin_abacom

# Verificar usuarios creados
cat /etc/passwd | grep -E "carlos|maria|admin"

# Ver ID de usuarios
id carlos
id maria
```

Listing 128.2 BASH

```
# Cambiar contraseña para carlos
sudo passwd carlos
# Ingresa: Carlos@123

# Cambiar para maria
sudo passwd maria
# Ingresa: Maria@123

# Cambiar para admin
sudo passwd admin_abacom
# Ingresa: Admin@123
```

128.3.6 Paso 6: Sudo y Escalación de Privilegios (20 min)

128.3.7 Paso 7: Cambiar Propietario y Grupo (15 min)

128.4 Ejercicios de Práctica Independiente

128.4.1 Ejercicio A: Estructura de Permisos (30 min)

1. Crea un directorio llamado `/home/alumno/abacom-data`
2. Crea 3 archivos dentro: `config.txt`, `logs.txt`, `backup.tar`
3. Asigna permisos:
    - `config.txt`: Solo propietario puede leer/escribir (600)
    - `logs.txt`: Propietario rwx, grupo rx, otros ninguno (750)
    - `backup.tar`: Todos pueden leer, solo propietario escribir (644)

Listing 128.3 BASH

```
# Crear grupo para desarrolladores
sudo groupadd desarrolladores

# Crear grupo para administradores
sudo groupadd sys_admins

# Agregar usuarios a grupos
sudo usermod -aG desarrolladores carlos
sudo usermod -aG desarrolladores maria
sudo usermod -aG sys_admins admin_abacom

# Verificar pertenencia a grupos
groups carlos
groups maria
groups admin_abacom
```

4. Verifica con `ls -la` y explica cada línea
5. Prueba acceso como usuario "carlos"

128.4.2 Ejercicio B: Gestión de Grupos (20 min)

1. Crea un grupo llamado "abacom-team"
2. Agrega 3 usuarios: carlos, maria, admin_abacom
3. Crea un directorio `/home/alumno/equipo`
4. Cambia propietario a "admin_abacom:abacom-team"
5. Asigna permisos 770 al directorio
6. Verifica que todos los miembros del grupo pueden acceder

128.5 Troubleshooting

Problema	Solución
"Permission denied" al crear usuario	Usar `sudo`
Usuario no aparece en `passwd`	Esperar segundos, refrescar con `cat /etc/passwd`
No puedo ver como otro usuario	Usar `sudo -u usuario comando`
sudo pide contraseña cada vez	Normal si no estás en wheel/sudo

128.6 Conceptos Clave

Permisos en Linux: - r (read): 4 - w (write): 2 - x (execute): 1 - rwx = 7, rw- = 6, r-x = 5, etc.

Listing 128.4 BASH

```bash
# Crear archivos para demostración
cd /home/alumno
echo "Contenido importante" > archivo_publico.txt
echo "Datos privados" > archivo_privado.txt

# Ver permisos actuales
ls -la archivo_*

# Cambiar permisos
chmod 755 archivo_publico.txt   # rwxr-xr-x (lectura todos)
chmod 700 archivo_privado.txt   # rwx------ (solo propietario)

# Verificar cambios
ls -la archivo_*

# Probar lectura como otro usuario
sudo -u carlos cat /home/alumno/archivo_publico.txt  # Debería funcionar
sudo -u carlos cat /home/alumno/archivo_privado.txt  # Debería fallar
```

Propietario: Usuario:Grupo = PERMISOS - `drwxr-xr-x alumno alumno proyecto/` - d = directorio - rwx = propietario - r-x = grupo - r-x = otros

---

128.7 Checklist de aceptación

- ☐ Existen los usuarios `carlos`, `maria`, `admin_abacom` y puedo mostrar `id`/`groups`
- ☐ Probé acceso a `archivo_publico.txt` vs `archivo_privado.txt` como otro usuario (evidencia del OK y del fallo)
- ☐ Completé Ejercicio A con permisos correctos (evidencia de `ls -la`)
- ☐ Completé Ejercicio B con grupo compartido y permisos `770` (evidencia de `ls -ld` y acceso)
- ☐ `admin_abacom` puede ejecutar `sudo whoami` y devuelve `root`

Listing 128.5 BASH

```
# Crear directorios
mkdir /home/alumno/proyecto_publico
mkdir /home/alumno/proyecto_privado

# Agregar archivos
echo "Proyecto A" > /home/alumno/proyecto_publico/archivo1.txt
echo "Proyecto B" > /home/alumno/proyecto_privado/archivo2.txt

# Cambiar permisos
chmod 755 /home/alumno/proyecto_publico  # Otros pueden listar
chmod 700 /home/alumno/proyecto_privado  # Solo propietario

# Verificar acceso
sudo -u carlos ls /home/alumno/proyecto_publico/   # OK
sudo -u carlos ls /home/alumno/proyecto_privado/   # Permission denied
```

Listing 128.6 BASH

```
# Agregar usuario a sudoers
sudo usermod -aG sudo admin_abacom

# Ver entrada de sudoers
sudo visudo -c
# Ver regla para admin_abacom

# Probar sudo como admin_abacom
sudo -u admin_abacom sudo whoami   # Debería mostrar "root"

# Ver histórico de sudo
sudo journalctl -u sudo -n 10

# O más simple
sudo grep COMMAND /var/log/auth.log | tail -5
```

Listing 128.7 BASH

```bash
# Cambiar propietario de archivo
sudo chown carlos /home/alumno/archivo_publico.txt

# Cambiar grupo de archivo
sudo chgrp desarrolladores /home/alumno/archivo_publico.txt

# Cambiar propietario y grupo simultáneamente
sudo chown admin_abacom:sys_admins /home/alumno/proyecto_publico

# Verificar cambios
ls -la /home/alumno/archivo_publico.txt
ls -ld /home/alumno/proyecto_publico
```

129 Laboratorio 3: Administración de Procesos

Unidad: 5 - Administración de Procesos y Servicios
Duración Estimada: 90 minutos
Dificultad: Intermedio

129.1 Objetivos

- Monitorear procesos del sistema
- Gestionar procesos (iniciar, detener, pausar)
- Entender systemd y servicios
- Automatizar tareas con cron

129.2 Requisitos Previos

- Ubuntu del Lab 1 corriendo
- Acceso SSH como "alumno"
- Permisos sudo

129.3 Pasos del Laboratorio

129.3.1 Paso 1: Monitoreo Básico de Procesos (20 min)

129.3.2 Paso 2: Gestión de Procesos - Prioridad y Señales (25 min)

129.3.3 Paso 3: Systemd y Servicios (20 min)

129.3.4 Paso 4: Crear Servicio Personalizado (25 min)

129.3.5 Paso 5: Tareas Programadas con Cron (20 min)

Listing 129.1 BASH

```
# Ver procesos activos
ps aux  # Todos los procesos

# Ver solo tus procesos
ps

# Más información con tree
ps auxf  # Árbol de procesos

# Usar top en tiempo real
top  # Presiona 'q' para salir

# Usar htop (más amigable)
sudo apt update && sudo apt install -y htop
htop  # Presiona 'q' para salir

# Ver procesos de usuario específico
ps -u alumno
```

129.3.6 Paso 6: At - Tareas Puntuales (10 min)

129.4 Ejercicios de Práctica Independiente

129.4.1 Ejercicio A: Monitoreo y Gestión (30 min)

1. Ejecuta `sleep 7200` en background
2. Usa `ps`, `jobs`, y `top` para verificar su ejecución
3. Pausa el proceso con Ctrl+Z
4. Reanúdalo con `bg`
5. Mata el proceso con `kill`
6. Verifica que terminó

129.4.2 Ejercicio B: Servicio de Monitorización (30 min)

1. Crea un script que registre la hora cada 10 segundos
2. Crea un servicio systemd para ese script
3. Inicia el servicio
4. Verifica en el log que funciona
5. Habilita al inicio
6. Reinicia la VM y verifica que continúa

Listing 129.2 BASH

```bash
# Crear proceso larga duración
sleep 3600 &  # Ejecuta en background por 1 hora
# Anota el PID mostrado (ej: [1] 12345)

# Ver procesos en background
jobs

# Ver información del proceso
ps -p 12345  # Reemplaza 12345 con tu PID

# Traer a foreground
fg  # Requiere Ctrl+Z para pausar primero
# Presiona Ctrl+Z para pausar

# Continuar en background
bg

# Matar proceso de forma suave
kill -15 12345  # SIGTERM

# Matar proceso de forma forzada (si es necesario)
kill -9 12345   # SIGKILL

# Cambiar prioridad de proceso
sleep 3600 &
# PID será diferente
# Cambiar prioridad (valor más alto = menos prioridad)
renice +10 -p [PID]
```

129.4.3 Ejercicio C: Cron Schedule (20 min)

1. Crear script que registre el uso de CPU
2. Programar con cron para ejecutarse cada hora
3. Verificar en logs que se ejecutó
4. Cambiar a cada 15 minutos
5. Después de 15 min, remover

129.5 Conceptos Clave

Señales Comunes: - SIGHUP (1): Colgar - SIGTERM (15): Terminar (default) - SIGKILL (9): Matar (no se puede ignorar) - SIGSTOP (19): Pausar - SIGCONT (18): Continuar

Formato Cron:

Listing 129.3 BASH

```bash
# Ver servicios disponibles
systemctl list-units --type=service

# Ver estado de servicio específico
systemctl status ssh
systemctl status networking

# Iniciar un servicio
sudo systemctl start ssh

# Detener un servicio
sudo systemctl stop ssh

# Reiniciar un servicio
sudo systemctl restart ssh

# Habilitar servicio al inicio
sudo systemctl enable ssh

# Deshabilitar servicio al inicio
sudo systemctl disable ssh

# Ver si está habilitado
systemctl is-enabled ssh

# Ver log en tiempo real
sudo journalctl -u ssh -f  # Presiona Ctrl+C para salir
```

```
* * * * * comando

         Día semana (0-6, 0=Domingo)
        Mes (1-12)
       Día (1-31)
      Hora (0-23)
     Minuto (0-59)
```

129.6 Checklist de aceptación

- ☐ Incluí evidencia de monitoreo (`ps`/`top`/`htop`) y expliqué qué observé
- ☐ Creé un proceso en background y lo terminé con SIGTERM (y, si aplica, SIGKILL) con evidencia

- ☐ `mi-servicio.service` existe, inicia y genera logs (evidencia de `systemctl status` y del archivo `/tmp/mi_servicio.log`)
- ☐ Configuré una tarea de cron y evidencié al menos una ejecución en `/tmp/cron_log.txt`
- ☐ Eliminé la tarea de cron al final (para no dejar basura)

Listing 129.4 BASH

```
# Crear script simple
cat > /home/alumno/mi_servicio.sh << 'EOF'
#!/bin/bash
while true; do
    echo "$(date): Servicio ejecutándose" >> /tmp/mi_servicio.log
    sleep 5
done
EOF

# Hacer ejecutable
chmod +x /home/alumno/mi_servicio.sh

# Crear archivo de systemd
sudo cat > /etc/systemd/system/mi-servicio.service << 'EOF'
[Unit]
Description=Mi Servicio de Prueba
After=network.target

[Service]
Type=simple
User=alumno
WorkingDirectory=/home/alumno
ExecStart=/home/alumno/mi_servicio.sh
Restart=always
RestartSec=5

[Install]
WantedBy=multi-user.target
EOF

# Recargar systemd
sudo systemctl daemon-reload

# Iniciar servicio
sudo systemctl start mi-servicio

# Verificar estado
sudo systemctl status mi-servicio

# Ver log
tail -f /tmp/mi_servicio.log  # Ctrl+C para salir

# Habilitar al inicio
sudo systemctl enable mi-servicio

# Detener servicio
sudo systemctl stop mi-servicio
```

Listing 129.5 BASH

```bash
# Editar tabla cron del usuario actual
crontab -e

# En el editor, agrega esta línea:
# Ejecutar tarea cada 5 minutos
*/5 * * * * /home/alumno/tarea.sh

# Ver tabla cron
crontab -l

# Crear script para ejecutar
cat > /home/alumno/tarea.sh << 'EOF'
#!/bin/bash
echo "Tarea ejecutada a las $(date)" >> /tmp/cron_log.txt
EOF

chmod +x /home/alumno/tarea.sh

# Agregar a cron
(crontab -l 2>/dev/null; echo "*/5 * * * * /home/alumno/tarea.sh") | crontab -

# Verificar que se está ejecutando
sleep 305  # Espera 5+ minutos
cat /tmp/cron_log.txt

# Ver logs de cron
grep CRON /var/log/syslog | tail -10

# Remover tarea
crontab -r
```

Listing 129.6 BASH

```bash
# Instalar at (si no está)
sudo apt install -y at

# Ver si servicio está corriendo
sudo systemctl status atd

# Programar tarea para dentro de 2 minutos
echo "echo 'Tarea at ejecutada' >> /tmp/at_log.txt" | at now + 2 minutes

# Ver tareas programadas
atq

# Ver detalles de tarea
at -l

# Remover tarea (si es necesario)
# atrm [ID]
```

130 Laboratorio 4: Redes y SSH

Unidad: 7 - Redes y Seguridad Básica
Duración Estimada: 120 minutos
Dificultad: Intermedio-Avanzado

130.1 Objetivos

- Configurar redes en Linux
- Entender SSH y autenticación con claves
- Configurar firewall básico
- Diagnosticar problemas de red

130.2 Requisitos Previos

- Ubuntu del Lab 1 corriendo
- Acceso SSH como "alumno"
- Permisos sudo

130.3 Pasos del Laboratorio

130.3.1 Paso 1: Diagnóstico de Red (15 min)

130.3.2 Paso 2: Configuración de Red (Netplan) (20 min)

130.3.3 Paso 3: SSH y Autenticación por Claves (30 min)

130.3.4 Paso 4: Configuración SSH Segura (25 min)

130.3.5 Paso 5: Firewall Básico (20 min)

130.3.6 Paso 6: Herramientas de Diagnóstico (20 min)

Listing 130.1 BASH

```
# Ver interfaces de red
ip link show

# Ver direcciones IP asignadas
ip addr show

# Ver tabla de routing
ip route show

# Testear conectividad DNS
nslookup google.com
dig google.com

# Testear conexión a host
ping -c 4 8.8.8.8

# Información completa de configuración
ifconfig  # o ip a
```

130.4 Ejercicios de Práctica Independiente

130.4.1 Ejercicio A: Conexión SSH Segura (30 min)

1. Generar par SSH (si no lo hiciste)
2. Copiar clave pública a la VM
3. Conectar sin contraseña
4. Cambiar puerto SSH a 2222
5. Conectar nuevamente al puerto 2222
6. Deshabilitar autenticación por contraseña

130.4.2 Ejercicio B: Configuración de Firewall (20 min)

1. Habilitar ufw firewall
2. Permitir SSH
3. Permitir HTTP y HTTPS
4. Verificar que SSH aún funciona
5. Ver todas las reglas activas
6. Denegar puerto 23 (Telnet)

130.4.3 Ejercicio C: Diagnóstico de Red (20 min)

1. Ejecutar `ss -tlnp` y documentar puertos abiertos
2. Ejecutar `ip route show` y explicar salida

Listing 130.2 BASH

```
# Ver archivos de netplan
ls -la /etc/netplan/

# Ver configuración actual
cat /etc/netplan/00-installer-config.yaml

# Crear respaldo
sudo cp /etc/netplan/00-installer-config.yaml /etc/netplan/00-installer-config.yaml.bak

# Editar configuración (solo si necesita IP fija)
sudo nano /etc/netplan/01-static.yaml

# Ejemplo de IP fija:
cat > /tmp/netplan-ejemplo.yaml << 'EOF'
network:
  version: 2
  ethernets:
    eth0:
      addresses:
        - 192.168.1.100/24
      gateway4: 192.168.1.1
      nameservers:
        addresses: [8.8.8.8, 8.8.4.4]
EOF

# NO ejecutes este cambio, solo es referencia
# Para aplicar cambios:
# sudo netplan apply
```

3. Hacer `dig` a un dominio y ver dirección IP
4. Usar `ping` a 3 hosts diferentes
5. Ejecutar `traceroute` a google.com

130.5 Troubleshooting

Problema	Solución
"Permission denied (publickey)"	Verificar permisos de ~/.ssh (700) y authorized_keys (600)
"ssh: command not found"	Instalar: sudo apt install openssh-client
"Connection refused"	SSH no está corriendo, verificar: sudo systemctl status ssh

Listing 130.3 BASH

```
# Verificar que SSH está activo
sudo systemctl status ssh

# En tu computadora HOST (no en la VM):
# Generar par de claves SSH (si no tienes)
ssh-keygen -t ed25519 -C "tu_email@example.com"
# O más compatible:
ssh-keygen -t rsa -b 4096 -C "tu_email@example.com"

# Presiona Enter para ubicación default
# Presiona Enter sin contraseña (o agrega si deseas)

# Ver claves generadas
ls ~/.ssh/

# Copiar clave pública a servidor (desde HOST):
ssh-copy-id -i ~/.ssh/id_rsa.pub alumno@[IP-VM]
# Ingresa contraseña una última vez

# Ahora conecta sin contraseña:
ssh alumno@[IP-VM]
# No debe pedir contraseña

# Ver claves autorizadas en servidor
cat ~/.ssh/authorized_keys

# Cambiar permisos de directorio SSH
chmod 700 ~/.ssh
chmod 600 ~/.ssh/authorized_keys
```

"ssh_keyscan: connection refused"	Host está down o SSH no escucha en ese puerto

130.6 Conceptos Clave

SSH Security: - **Port 22:** Default (cambiar en producción) - **Public Key:** Compartible, identifica host - **Private Key:** Secreto, debe ser seguro (chmod 600) - **authorized_keys:** Claves públicas permitidas

Firewall Básico: - UFW es fronted amigable de iptables - SIEMPRE permitir SSH antes de habilitar firewall - Reglas proceden de arriba a abajo

Herramientas Red: - `ss` o `netstat`: Ver conexiones/puertos - `ip`: Ver/configurar direcciones IP - `ping`: Test ICMP (reachability) - `dig`/`nslookup`: DNS queries

Listing 130.4 BASH

```
# Editar configuración SSH (en la VM)
sudo nano /etc/ssh/sshd_config

# Cambios recomendados (cambiar líneas existentes):
# Port 22                       (cambiar a 2222 si deseas otro puerto)
# PermitRootLogin no            (nunca permitir root)
# PasswordAuthentication no     (solo claves, si ya configuraste)
# PubkeyAuthentication yes      (ya debe estar así)
# X11Forwarding no              (seguridad)
# MaxAuthTries 3                (limitar intentos)
# MaxSessions 5                 (limitar sesiones)

# Validar configuración
sudo sshd -t

# Reiniciar SSH
sudo systemctl restart ssh

# Verificar que aún funciona
ssh alumno@[IP-VM]
```

130.7 Checklist de aceptación

- ☐ Incluí evidencia de inventario de red (`ip addr`, `ip route`) y resolución (`getent`/`dig`/`nslookup`)
- ☐ Puedo conectar por SSH usando clave (sin contraseña) y lo evidencié
- ☐ `sshd -t` pasa sin errores y el servicio SSH sigue accesible tras cambios
- ☐ UFW está activo y permite el puerto SSH (evidencia de `ufw status verbose`)
- ☐ Incluí evidencia de puertos en escucha con `ss -tlnp`

Listing 130.5 BASH

```
# Ver estado de firewall
sudo ufw status

# Habilitar firewall (CUIDADO: configura SSH primero)
sudo ufw enable

# Permitir SSH (CRÍTICO antes de habilitar)
sudo ufw allow ssh
# O si cambiaste puerto:
# sudo ufw allow 2222/tcp

# Permitir HTTP y HTTPS
sudo ufw allow http
sudo ufw allow https
# O explícitamente:
# sudo ufw allow 80/tcp
# sudo ufw allow 443/tcp

# Denegar acceso específico
sudo ufw deny 23/tcp  # Deny Telnet

# Ver reglas
sudo ufw show added
sudo ufw status verbose

# Remover regla
sudo ufw delete allow http

# Deshabilitar firewall (solo para troubleshooting)
sudo ufw disable
```

Listing 130.6 BASH

```bash
# Ver puertos abiertos
ss -tlnp  # t=tcp, l=listening, n=numeric, p=program

# Versión antigua (netstat)
netstat -tlnp

# Ver conexiones establecidas
ss -tnp

# Escanear puertos en host remoto (desde HOST)
nmap [IP-VM]  # Si lo tienes instalado

# Ver conexiones DNS
dig @8.8.8.8 google.com

# Traceroute a host
traceroute google.com

# Ver latencia
ping -c 4 8.8.8.8

# Ver velocidad de descarga (test de velocidad)
speedtest-cli  # Primero instalar: sudo apt install speedtest-cli
```

131 Laboratorio 4B: Docker y Docker Compose

Unidad: 4 - Docker y Containerización
Duración Estimada: 90-120 minutos
Dificultad: Intermedio

131.1 Objetivos

- Instalar Docker y validar motor
- Levantar un servicio con Docker Compose
- Revisar logs, puertos y volúmenes

131.2 Requisitos Previos

- VM Ubuntu Server LTS (22.04 o 24.04) con acceso SSH
- Usuario con permisos sudo

131.3 Pasos del Laboratorio

131.3.1 Paso 1: Instalación (20-30 min)

① **apt update** actualiza indice.
② **apt install docker.io** instala Docker y el plugin de Compose.
③ **systemctl status docker** valida servicio.
④ **docker version** valida el cliente/servidor.

131.3.2 Paso 2: Compose - servicio web simple (30-40 min)

① **mkdir/cd** crea un workspace del lab.
② **compose.yml** define un contenedor Nginx expuesto en 8080.
③ **docker compose up -d** levanta el stack en background.
④ **docker compose ps** valida estado.
⑤ **curl -I** valida que responde.

Listing 131.1 BASH

```
$ sudo apt update                                                    ①


$ sudo apt install -y docker.io docker-compose-plugin                ②


$ systemctl status docker --no-pager                                 ③
  docker.service - Docker Application Container Engine
      Active: active (running)


$ sudo docker version                                                ④
```

131.3.3 Paso 3: Logs, puertos y limpieza (20-30 min)

① **docker compose logs** muestra evidencia de ejecucion.
② **ss -lntp** evidencia que el puerto esta escuchando.
③ **docker compose down** detiene y elimina contenedores del stack.

131.4 Entregables (Evidencia)

- `systemctl status docker --no-pager`
- `docker compose ps`
- `curl -I http://127.0.0.1:8080/`
- `docker compose logs --tail 50`

131.5 Checklist de aceptación

☐ Docker está instalado y el daemon está activo (evidencia de `systemctl status docker --no-pager`)
☐ El stack levanta con `docker compose up -d` y aparece en `docker compose ps`
☐ `curl -I http://127.0.0.1:8080/` devuelve `200 OK`
☐ Incluí evidencia de logs (`docker compose logs --tail 50`)
☐ Bajé el stack con `docker compose down` y verifiqué que el puerto 8080 ya no escucha

Listing 131.2 BASH

```
$ mkdir -p ~/lab4b && cd ~/lab4b                                    ①


$ cat > compose.yml <<'EOF' # <2>
services:
  web:
    image: nginx:1.25-alpine
    ports:
      - "8080:80"
EOF
                                                                    ②

$ sudo docker compose up -d                                         ③


$ sudo docker compose ps                                            ④


$ curl -I http://127.0.0.1:8080/                                    ⑤
HTTP/1.1 200 OK
Server: nginx
```

Listing 131.3 BASH

```
$ sudo docker compose logs --tail 50                                ①


$ sudo ss -lntp | grep ':8080'                                      ②


$ sudo docker compose down                                          ③
```

132 Laboratorio 5: Almacenamiento (Particiones y Montajes)

Unidad: 6 - Almacenamiento y Sistemas de Archivos
Duración Estimada: 90-120 minutos
Dificultad: Intermedio

132.1 Objetivos

- Inventariar discos y filesystems
- Crear un disco de laboratorio con loop device
- Crear una partición + filesystem ext4
- Montar el filesystem y dejarlo persistente con fstab
- Diagnosticar uso de espacio con du

132.2 Requisitos Previos

- VM Ubuntu Server LTS (22.04 o 24.04) con acceso SSH
- Usuario con permisos sudo

> ⚠ ADVERTENCIA CRÍTICA
>
> En este laboratorio vas a usar `parted`/`mkfs`.
> **Lo que podría salir mal:** - Si apuntas a un disco real (ej: `/dev/sda`) puedes borrar tu sistema. - Un cambio incorrecto en `/etc/fstab` puede dejar el sistema sin boot.
> **Cómo prevenirlo:** 1. Usa SOLO el disco de laboratorio creado como archivo y su loop device. 2. Verifica 2 veces el dispositivo con `lsblk` antes de formatear. 3. Antes de editar `fstab`, crea un backup.

132.3 Pasos del Laboratorio

132.3.1 Paso 1: Inventario de discos y montajes (10 min)

① **lsblk -f** te confirma qué filesystem está montado como `/` (no lo toques).
② **df -hT** valida tamaño/uso y el tipo de filesystem montado en `/`.

Listing 132.1 BASH

```
$ lsblk -f                                                                    ①
NAME   FSTYPE FSVER LABEL UUID                                 FSAVAIL FSUSE% MOUNTPOINTS
sda
 sda1  ext4   1.0         11111111-2222-3333-4444-555555555555   35G    12% /
 sda15 vfat   FAT32       AAAA-BBBB                               98M     4% /boot/efi

$ df -hT                                                                      ②
Filesystem     Type   Size  Used Avail Use% Mounted on
/dev/sda1      ext4    49G  5.4G   41G  12% /
tmpfs          tmpfs  1.9G     0  1.9G   0% /run
```

132.3.2 Paso 2: Crear un disco de laboratorio (loop) (15 min)

Listing 132.2 BASH

```
$ sudo fallocate -l 1G /var/tmp/u6-disk.img                                   ①

$ sudo losetup -fP /var/tmp/u6-disk.img                                       ②

$ sudo losetup -a | grep u6-disk.img                                          ③
/dev/loop0: [2065]:12345 (/var/tmp/u6-disk.img)


$ lsblk /dev/loop0                                                            ④
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0    7:0    0    1G  0 loop
```

① **fallocate** crea un archivo que actuará como disco.
② **losetup -fP** asocia el archivo a un loop device y escanea particiones.
③ **losetup -a** valida que el loop device apunta al archivo correcto.
④ **lsblk /dev/loop0** confirma que estás trabajando sobre loop (no un disco real).

132.3.3 Paso 3: Particionar (GPT + 1 partición) (15 min)

① **parted mklabel gpt** crea tabla de particiones GPT en el loop.
② **parted mkpart** crea una partición que ocupe el disco de laboratorio.
③ **lsblk** confirma que existe `/dev/loop0p1`.

Listing 132.3 BASH

```
$ sudo parted -s /dev/loop0 mklabel gpt                                    ①

$ sudo parted -s /dev/loop0 mkpart primary ext4 1MiB 100%                  ②

$ lsblk /dev/loop0                                                         ③
NAME      MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0       7:0    0    1G  0 loop
 loop0p1  259:0    0 1023M  0 part
```

132.3.4 Paso 4: Crear filesystem y montar (20 min)

Listing 132.4 BASH

```
$ sudo mkfs.ext4 -F /dev/loop0p1                                           ①

$ sudo mkdir -p /mnt/u6-data                                               ②

$ sudo mount /dev/loop0p1 /mnt/u6-data                                     ③

$ df -hT /mnt/u6-data                                                      ④
Filesystem     Type  Size  Used Avail Use% Mounted on
/dev/loop0p1   ext4  990M   24K  923M   1% /mnt/u6-data
```

① **mkfs.ext4** formatea la particion con ext4 (solo en loop).
② **mkdir -p** crea punto de montaje.
③ **mount** monta el filesystem.
④ **df -hT** valida el montaje y tipo de FS.

132.3.5 Paso 5: Persistencia con fstab (20 min)

Nota: este paso valida persistencia con `mount -a` (sin reiniciar). En un reboot real, tendrías que volver a asociar el archivo a un loop device o quitar la entrada del lab.

① **blkid** obtiene el UUID para montar de forma estable.
② **cp /etc/fstab** crea backup antes de tocar persistencia.
③ **tee -a /etc/fstab** agrega la entrada (usa tu UUID real).
④ **umount** desmonta para probar persistencia.
⑤ **mount -a** valida que fstab es correcto (sin reiniciar).
⑥ **df -hT** confirma que el montaje se recupera via fstab.

Listing 132.5 BASH

```
$ sudo blkid /dev/loop0p1                                                    ①
/dev/loop0p1: UUID="66666666-7777-8888-9999-aaaaaaaaaaaa" TYPE="ext4" PARTUUID="bbbbbbbb-


$ sudo cp /etc/fstab /etc/fstab.bak                                          ②


$ printf '\n# Lab 5: loop mount\nUUID=66666666-7777-8888-9999-aaaaaaaaaaaa /mnt/u6-data e


$ sudo umount /mnt/u6-data                                                   ④


$ sudo mount -a                                                              ⑤


$ df -hT /mnt/u6-data                                                        ⑥
Filesystem     Type  Size  Used Avail Use% Mounted on
/dev/loop0p1   ext4  990M   24K  923M   1% /mnt/u6-data
```

132.3.6 Paso 6: Diagnóstico de espacio (15 min)

Listing 132.6 BASH

```
$ sudo du -xh --max-depth=1 /var 2>/dev/null | sort -hr | head -n 10     ①
1.2G    /var
620M    /var/lib
210M    /var/log
...
```

① **du + sort** identifica directorios mas grandes para diagnosticar espacio.

132.4 Entregables (Evidencia)

- `lsblk -f` y `df -hT`
- `losetup -a | grep u6-disk.img`
- `lsblk /dev/loop0`
- `df -hT /mnt/u6-data` (antes y despues de `mount -a`)

- Top 3 de `du` en `/var` (con breve interpretacion)

132.5 Cierre y limpieza (5 min)

Listing 132.7 BASH

```
$ sudo umount /mnt/u6-data                                    ①

$ sudo losetup -d /dev/loop0                                  ②

$ sudo rm -f /var/tmp/u6-disk.img                             ③

$ sudo cp /etc/fstab.bak /etc/fstab                           ④

$ sudo mount -a                                               ⑤
```

① **umount** desmonta el punto.
② **losetup -d** libera el loop device.
③ **rm** borra el archivo-disco de laboratorio.
④ **cp /etc/fstab.bak** revierte el cambio de persistencia (evita errores futuros de montaje).
⑤ **mount -a** valida que `fstab` quedó limpio.

132.6 Checklist de aceptación

- ☐ Trabajé únicamente sobre un loop device (evidencia de `losetup -a | grep u6-disk.img` y `lsblk /dev/loop0`)
- ☐ Puedo mostrar el filesystem montado en `/mnt/u6-data` con `df -hT /mnt/u6-data`
- ☐ Probé persistencia con `mount -a` sin errores (evidencia incluida)
- ☐ Incluí un top 3 de uso de espacio en `/var` con interpretación breve
- ☐ Hice limpieza: desmonté, liberé loop, borré imagen y restauré `/etc/fstab` (evidencia de `mount -a` final)

133 Laboratorio 6: HTTP Basico y Diagnostico Web

Unidad: 8 - Introducción a Servidores Web
Duración Estimada: 60-90 minutos
Dificultad: Principiante-Intermedio

133.1 Objetivos

- Entender request/response con `curl`
- Validar DNS, puertos y headers
- Distinguir fallas tipicas: DNS vs puerto vs firewall vs servicio caido

133.2 Requisitos Previos

- VM Ubuntu Server LTS (22.04 o 24.04) con acceso SSH
- Usuario con permisos sudo

133.3 Pasos del Laboratorio

133.3.1 Paso 1: Preparar un servicio HTTP de prueba (15 min)

Listing 133.1 BASH

```
$ python3 -m http.server 8080 --bind 0.0.0.0                    ①
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
```

① **python3 -m http.server** levanta un servidor HTTP simple para pruebas.

En otra terminal SSH (o en la misma usando Ctrl+C luego), continua con las validaciones.

Listing 133.2 BASH

```
$ sudo ss -lntp | grep ':8080'                                                    ①
LISTEN 0 5 0.0.0.0:8080 0.0.0.0:* users:("python3",pid=1234,fd=3)


$ curl -i http://127.0.0.1:8080/ | head -n 20                                     ②
HTTP/1.0 200 OK
Server: SimpleHTTP/0.6 Python/3.12.3
Content-type: text/html; charset=utf-8
...
```

133.3.2 Paso 2: Validar puerto y proceso (15 min)

① **ss -lntp** evidencia que el puerto esta escuchando y quien lo usa.
② **curl -i** muestra status line y headers para validar HTTP.

133.3.3 Paso 3: DNS vs IP (15 min)

Listing 133.3 BASH

```
$ getent hosts example.com                                                        ①
2606:2800:220:1:248:1893:25c8:1946 example.com


$ dig +short example.com | head -n 5                                              ②
93.184.216.34
```

① **getent hosts** valida resolucion usando el resolver del sistema.
② **dig +short** muestra IPs resueltas de forma clara.

133.3.4 Paso 4: Simulacion de fallas (15-30 min)

133.3.4.1 Falla A: servicio caido

Deten el server (Ctrl+C) y prueba:

① **Connection refused** suele indicar que no hay proceso escuchando en ese puerto.

Listing 133.4 BASH

```
$ curl -i http://127.0.0.1:8080/                                    ①
curl: (7) Failed to connect to 127.0.0.1 port 8080: Connection refused
```

133.3.4.2 Falla B: puerto bloqueado (si tienes UFW activo)

Listing 133.5 BASH

```
$ sudo ufw status verbose                                           ①
Status: active


$ sudo ufw deny 8080/tcp                                            ②


$ sudo ufw status numbered                                          ③
```

① **ufw status** confirma si el firewall esta activo.
② **ufw deny** simula bloqueo.
③ **ufw status numbered** permite revertir con delete.

133.4 Entregables (Evidencia)

- `ss -lntp | grep :8080`
- `curl -i http://127.0.0.1:8080/`
- `getent hosts example.com` y `dig +short example.com`
- Explicacion breve: diferencia entre `Connection refused` y un timeout

133.5 Checklist de aceptación

- ☐ Levanté un servicio HTTP de prueba y evidencié el puerto en escucha con `ss -lntp | grep ':8080'`
- ☐ Incluí evidencia de respuesta HTTP con `curl -i http://127.0.0.1:8080/`
- ☐ Incluí evidencia de resolución DNS (`getent hosts` y `dig +short`)
- ☐ Expliqué con mis palabras `Connection refused` vs timeout
- ☐ Si cambié reglas de UFW para la simulación, las revertí y dejé UFW en estado consistente

134 Laboratorio 7: Apache - Sitio y VirtualHost

Unidad: 9 - Apache (Servidor Web)
Duración Estimada: 90-120 minutos
Dificultad: Intermedio

134.1 Objetivos

- Instalar Apache y validar servicio
- Crear un VirtualHost
- Revisar logs (access/error)
- Publicar un sitio simple con evidencia

134.2 Requisitos Previos

- VM Ubuntu Server LTS (22.04 o 24.04) con acceso SSH
- Usuario con permisos sudo

134.3 Pasos del Laboratorio

134.3.1 Paso 1: Instalacion y validacion (20 min)

① **apt update** actualiza el indice de paquetes.
② **apt install apache2** instala Apache.
③ **systemctl status** verifica que el servicio esta corriendo.
④ **curl -I** valida respuesta HTTP local.

134.3.2 Paso 2: Crear contenido del sitio (10 min)

① **mkdir -p** crea la estructura del document root.
② **tee** escribe una pagina simple para validar.

Listing 134.1 BASH

```
$ sudo apt update                                                    ①


$ sudo apt install -y apache2                                        ②


$ systemctl status apache2 --no-pager                                ③
  apache2.service - The Apache HTTP Server
     Active: active (running)


$ curl -I http://127.0.0.1/                                          ④
HTTP/1.1 200 OK
Server: Apache/2.4.
```

Listing 134.2 BASH

```
$ sudo mkdir -p /var/www/app.local/public                            ①


$ printf 'app.local ok\n' | sudo tee /var/www/app.local/public/index.html >/dev/null ②
```

134.3.3 Paso 3: VirtualHost (25 min)

① **tee + heredoc** crea el VirtualHost.
② **a2ensite** habilita el sitio.
③ **a2dissite** deshabilita el default para evitar confusiones.
④ **apache2ctl configtest** valida sintaxis.
⑤ **systemctl reload** aplica cambios sin cortar conexiones.

134.3.4 Paso 4: Prueba por Host header + logs (20 min)

① **curl -H Host** fuerza el VirtualHost sin necesidad de DNS.
② **tail access.log** entrega evidencia de requests.
③ **tail error.log** confirma que no hay errores.

Listing 134.3 BASH

```
$ sudo tee /etc/apache2/sites-available/app.local.conf >/dev/null <<'EOF' # <1>
<VirtualHost *:80>
  ServerName app.local
  DocumentRoot /var/www/app.local/public

  ErrorLog ${APACHE_LOG_DIR}/app.local-error.log
  CustomLog ${APACHE_LOG_DIR}/app.local-access.log combined

  <Directory /var/www/app.local/public>
    Require all granted
  </Directory>
</VirtualHost>
EOF
                                                                    ①

$ sudo a2ensite app.local.conf                                      ②


$ sudo a2dissite 000-default.conf                                   ③


$ sudo apache2ctl configtest                                        ④
Syntax OK


$ sudo systemctl reload apache2                                     ⑤
```

134.4 Entregables (Evidencia)

- `systemctl status apache2 --no-pager`
- `apache2ctl configtest`
- `curl -i -H 'Host: app.local' http://127.0.0.1/`
- Ultimas 5 lineas de access/error logs del vhost

134.5 Checklist de aceptación

- ☐ Apache está activo (evidencia de `systemctl status apache2 --no-pager`)
- ☐ El VirtualHost está habilitado y `apache2ctl configtest` retorna `Syntax OK`
- ☐ `curl -i -H 'Host: app.local' http://127.0.0.1/` responde con el contenido esperado
- ☐ Incluí evidencia de access log y confirmé ausencia de errores relevantes en error log
- ☐ Si usé UFW, el puerto 80 está permitido (o expliqué cómo lo validé)

Listing 134.4 BASH

```
$ curl -i -H 'Host: app.local' http://127.0.0.1/                    ①
HTTP/1.1 200 OK
Server: Apache/2.4.

app.local ok


$ sudo tail -n 5 /var/log/apache2/app.local-access.log              ②
127.0.0.1 - - [02/Feb/2026:01:12:12 +0000] "GET / HTTP/1.1" 200 12 "-" "curl/8.0.1"


$ sudo tail -n 5 /var/log/apache2/app.local-error.log               ③
```

135 Laboratorio 8: Nginx - Reverse Proxy + TLS

Unidad: 10 - Nginx y SSL
Duración Estimada: 120-150 minutos
Dificultad: Intermedio

135.1 Objetivos

- Configurar un server block en Nginx
- Hacer reverse proxy hacia un servicio local
- Habilitar HTTPS con certificado self-signed (o Certbot si tienes dominio)

135.2 Requisitos Previos

- VM Ubuntu Server LTS (22.04 o 24.04) con acceso SSH
- Usuario con permisos sudo

> ⚠ ADVERTENCIA CRÍTICA
>
> Si habilitas UFW sin permitir tu puerto SSH, puedes perder acceso.
> **Lo que podría salir mal:** - Bloqueo de SSH remoto.
> **Cómo prevenirlo:** 1. Verifica tu puerto SSH actual. 2. Agrega la regla de UFW para SSH antes de activar.

135.3 Pasos del Laboratorio

135.3.1 Paso 1: Levantar un backend local (15 min)

Listing 135.1 BASH

```
$ python3 -m http.server 9000 --bind 127.0.0.1                    ①
Serving HTTP on 127.0.0.1 port 9000 (http://127.0.0.1:9000/) ...
```

① **python3 -m http.server** simula un backend HTTP local (solo loopback).

135.3.2 Paso 2: Instalar Nginx y configurar reverse proxy (35 min)

① **apt update** actualiza el indice.
② **apt install nginx** instala Nginx.
③ **tee** crea la configuracion del reverse proxy.
④ **ln -sf** habilita el sitio.
⑤ **nginx -t** valida sintaxis.
⑥ **systemctl reload** aplica cambios.
⑦ **curl -H Host** prueba el server block.

135.3.3 Paso 3: TLS self-signed (35 min)

① **mkdir -p** crea directorio para llaves/certs.
② **openssl req -x509** genera certificado self-signed para pruebas.
③ **tee** crea server block TLS.
④ **ln -sf** habilita el sitio TLS.
⑤ **nginx -t** valida.
⑥ **reload** aplica.
⑦ **curl -k** prueba HTTPS ignorando confianza (self-signed).

135.3.4 Paso 4 (opcional): Certbot si tienes dominio (20-40 min)

① **apt install certbot** instala Certbot.
② **certbot –nginx** emite certificado y configura TLS real.
③ **certbot renew –dry-run** valida renovacion.

135.4 Entregables (Evidencia)

- `nginx -t` y `systemctl status nginx --no-pager`
- `curl -i -H 'Host: rp.local' http://127.0.0.1/` (reverse proxy)
- `curl -k -I -H 'Host: rp.local' https://127.0.0.1/` (TLS self-signed)

135.5 Checklist de aceptación

- ☐ Nginx está activo y `nginx -t` pasa sin errores
- ☐ El reverse proxy funciona (evidencia de `curl -i -H 'Host: rp.local' http://127.0.0.1/`)
- ☐ TLS self-signed funciona (evidencia de `curl -k -I -H 'Host: rp.local' https://127.0.0.1/`)
- ☐ Identifiqué el backend local y confirmé que solo escucha en 127.0.0.1:9000
- ☐ Si UFW está activo, dejé abiertos 80/443 y mantuve acceso SSH

Listing 135.2 BASH

```
$ sudo apt update                                                              ①


$ sudo apt install -y nginx                                                    ②


$ sudo tee /etc/nginx/sites-available/rp.conf >/dev/null <<'EOF' # <3>
server {
  listen 80;
  server_name rp.local;

  location / {
    proxy_pass http://127.0.0.1:9000;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
  }
}
EOF
                                                                               ③

$ sudo ln -sf /etc/nginx/sites-available/rp.conf /etc/nginx/sites-enabled/rp.conf ④


$ sudo nginx -t                                                                ⑤
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful


$ sudo systemctl reload nginx                                                  ⑥


$ curl -i -H 'Host: rp.local' http://127.0.0.1/ | head -n 10                   ⑦
HTTP/1.1 200 OK
Server: nginx
...
```

Listing 135.3 BASH

```
$ sudo mkdir -p /etc/nginx/tls                                                    ①


$ sudo openssl req -x509 -nodes -newkey rsa:2048 -days 365 \
  -keyout /etc/nginx/tls/rp.local.key \
  -out /etc/nginx/tls/rp.local.crt \
  -subj "/CN=rp.local"                                                            ②


$ sudo tee /etc/nginx/sites-available/rp-tls.conf >/dev/null <<'EOF' # <3>
server {
  listen 443 ssl;
  server_name rp.local;

  ssl_certificate     /etc/nginx/tls/rp.local.crt;
  ssl_certificate_key /etc/nginx/tls/rp.local.key;

  location / {
    proxy_pass http://127.0.0.1:9000;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
  }
}
EOF
                                                                                  ③

$ sudo ln -sf /etc/nginx/sites-available/rp-tls.conf /etc/nginx/sites-enabled/rp-tls.conf


$ sudo nginx -t                                                                   ⑤
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful


$ sudo systemctl reload nginx                                                     ⑥


$ curl -k -I -H 'Host: rp.local' https://127.0.0.1/                               ⑦
HTTP/2 200
server: nginx
```

Listing 135.4 BASH

```
$ sudo apt install -y certbot python3-certbot-nginx ①

$ sudo certbot --nginx -d tu-dominio.com ②

$ sudo certbot renew --dry-run ③
```

136 Laboratorio 9: MariaDB - Usuarios, Privilegios y Backup/Restore

Unidad: 11 - MariaDB
Duración Estimada: 120-150 minutos
Dificultad: Intermedio

136.1 Objetivos

- Instalar y validar MariaDB
- Aplicar hardening basico
- Crear DB + usuario con privilegios minimos
- Generar backup y probar restore

136.2 Requisitos Previos

- VM Ubuntu Server LTS (22.04 o 24.04) con acceso SSH
- Usuario con permisos sudo

136.3 Pasos del Laboratorio

136.3.1 Paso 1: Instalacion y validacion (20 min)

① **apt update** actualiza indice.
② **apt install mariadb-server** instala el motor.
③ **systemctl status** valida servicio.
④ **mariadb -e** valida acceso local.

136.3.2 Paso 2: Hardening inicial (15-25 min)

① **mariadb-secure-installation** aplica hardening basico (root, anonimos, test DB).

Listing 136.1 BASH

```
$ sudo apt update                                                    ①


$ sudo apt install -y mariadb-server                                 ②


$ systemctl status mariadb --no-pager                                ③
 mariadb.service - MariaDB database server
     Active: active (running)


$ sudo mariadb -e 'SELECT VERSION();'                                ④
VERSION()
10.11.6-MariaDB
```

Listing 136.2 BASH

```
$ sudo mariadb-secure-installation                                   ①
```

136.3.3 Paso 3: Crear DB y usuario (30 min)

① **mariadb** ejecuta SQL para DB/usuario y crea datos de prueba para validar backup.

136.3.4 Paso 4: Backup y restore (30-40 min)

① **mysqldump** crea backup SQL.
② **ls -lh** valida el archivo.
③ **DROP DATABASE** simula perdida de datos.
④ **mariadb < backup** restaura.
⑤ **SELECT** confirma que los datos volvieron.

136.4 Entregables (Evidencia)

- `systemctl status mariadb --no-pager`
- SQL usado para crear DB/usuario + salida del `SELECT`
- Tamaño de `/tmp/appdb.sql`
- Evidencia de restore (salida de `SELECT * FROM appdb.demo;`)

Listing 136.3 BASH

```
$ sudo mariadb                                                                    (1)
MariaDB [(none)]> CREATE DATABASE appdb;
MariaDB [(none)]> CREATE USER 'appuser'@'localhost' IDENTIFIED BY 'AppPass-ChangeMe';
MariaDB [(none)]> GRANT SELECT,INSERT,UPDATE,DELETE ON appdb.* TO 'appuser'@'localhost';
MariaDB [(none)]> FLUSH PRIVILEGES;
MariaDB [(none)]> USE appdb;
MariaDB [appdb]> CREATE TABLE demo (id INT PRIMARY KEY AUTO_INCREMENT, note VARCHAR(64));
MariaDB [appdb]> INSERT INTO demo(note) VALUES ('hello');
MariaDB [appdb]> SELECT * FROM demo;
MariaDB [appdb]> EXIT;
```

136.5 Checklist de aceptación

- ☐ MariaDB está activo (evidencia de `systemctl status mariadb --no-pager`)
- ☐ Ejecuté `mariadb-secure-installation` o documenté el estado inicial de seguridad
- ☐ Creé DB + usuario con privilegios mínimos y pude hacer un `SELECT` de prueba
- ☐ Generé un backup con `mysqldump` (evidencia del archivo y tamaño)
- ☐ Probé restore y validé que los datos volvieron (evidencia de `SELECT * FROM appdb.demo;`)

Listing 136.4 BASH

```
$ sudo mysqldump --databases appdb > /tmp/appdb.sql                    ①


$ ls -lh /tmp/appdb.sql                                                ②
-rw-r--r-- 1 root root 12K Feb  2 12:45 /tmp/appdb.sql


$ sudo mariadb -e 'DROP DATABASE appdb;'                               ③


$ sudo mariadb < /tmp/appdb.sql                                        ④


$ sudo mariadb -e 'SELECT * FROM appdb.demo;'                          ⑤
id  note
1   hello
```

137 Laboratorio 10: Troubleshooting Integrado (Checklist + Evidencias)

Unidad: 12 - Diagnóstico y Troubleshooting
Duración Estimada: 120-180 minutos
Dificultad: Intermedio-Avanzado

137.1 Objetivos

- Aplicar un flujo reproducible de diagnostico
- Recolectar evidencia minima (contexto, logs, red, recursos)
- Resolver incidentes simulados sin cambios destructivos

137.2 Requisitos Previos

- VM Ubuntu Server LTS (22.04 o 24.04) con acceso SSH
- Usuario con permisos sudo
- Haber completado al menos: Lab 0, Lab 1, Lab 4 y uno de (Lab 7/8/9)

137.3 Escenario

Un usuario reporta: “La web no responde”.

Tu objetivo es producir un reporte de diagnostico y dejar el servicio operativo.

137.4 Checklist de Evidencias (obligatorio)

137.4.1 Evidencia 1: Contexto

① **hostnamectl** entrega contexto de host/OS.
② **uptime** entrega contexto de carga y uptime.

Listing 137.1 BASH

```
$ hostnamectl                                                                    ①
 Static hostname: srv01
 Operating System: Ubuntu 24.04.1 LTS


$ uptime                                                                         ②
 12:55:03 up 1 day,  3:21,  1 user,  load average: 0.08, 0.11, 0.09
```

Listing 137.2 BASH

```
$ ip -br addr                                                                    ①
enp0s3  UP  192.168.56.10/24


$ ip route | head -n 5                                                           ②
default via 192.168.56.1 dev enp0s3


$ sudo ss -lntp | head -n 30                                                     ③
...
```

137.4.2 Evidencia 2: Red y puertos

① **ip -br addr** evidencia IP.
② **ip route** evidencia gateway.
③ **ss -lntp** evidencia puertos TCP en escucha.

137.4.3 Evidencia 3: Estado de servicio web

Escoge 1 segun tu stack:

- Si usas Nginx:

① **systemctl status** muestra estado actual.
② **journalctl -u** entrega evidencia de logs.

- Si usas Apache:

① **systemctl status** muestra estado actual.
② **tail error.log** entrega evidencia de errores.

Listing 137.3 BASH

```
$ systemctl status nginx --no-pager                                    ①
 nginx.service - A high performance web server and a reverse proxy server
    Active: active (running)


$ sudo journalctl -u nginx --since "2 hours ago" --no-pager | tail -n 40 ②
... logs ...
```

Listing 137.4 BASH

```
$ systemctl status apache2 --no-pager                                  ①
 apache2.service - The Apache HTTP Server
    Active: active (running)


$ sudo tail -n 40 /var/log/apache2/error.log                           ②
... logs ...
```

137.4.4 Evidencia 4: Recursos

Listing 137.5 BASH

```
$ free -h                                                              ①
               total        used        free      shared  buff/cache   available
Mem:           3.8Gi       1.1Gi       2.0Gi        52Mi       740Mi       2.5Gi


$ df -h                                                                ②
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1        49G  5.4G   41G  12% /
```

① **free -h** evidencia memoria.
② **df -h** evidencia espacio.

137.5 Incidentes simulados (elige 2)

137.5.1 Incidente A: Puerto incorrecto

1. Configura tu servicio para escuchar en un puerto distinto (ej: 8080) y prueba `curl` en 80/443.
2. Documenta el error y la causa con `ss -lntp`.
3. Corrige y valida.

137.5.2 Incidente B: Firewall bloqueando

1. Agrega una regla UFW que bloquee el puerto del servicio.
2. Evidencia con `ufw status`.
3. Corrige y valida.

137.5.3 Incidente C: DNS / Host header

1. Prueba el VirtualHost usando `curl -H 'Host: ...'`.
2. Explica diferencia entre DNS y Host header.
3. Valida el sitio correcto.

137.6 Entregables

- Checklist completo con outputs
- Para cada incidente elegido:
  - Sintoma
  - Evidencia
  - Hipotesis
  - Fix
  - Validacion

137.7 Checklist de aceptación

☐ Incluí contexto (hostnamectl + uptime) con explicación breve
☐ Incluí evidencia de red y puertos (`ip -br addr`, `ip route`, `ss -lntp`)
☐ Incluí evidencia de estado y logs del servicio web (nginx o apache)
☐ Elegí y resolví 2 incidentes simulados con evidencia antes/después
☐ El servicio queda operativo y lo validé con `curl` (o prueba equivalente)

138 Laboratorio 11: Clawdbot (OpenClaw) + Telegram + Tailscale (Seguridad)

Unidad: 4/7/12 - Docker, Redes y Seguridad, Troubleshooting
Duracion Estimada: 150-210 minutos
Dificultad: Avanzado

138.1 Objetivos

- Levantar **OpenClaw Gateway** en Docker sobre **Ubuntu Server LTS**.
- Conectar un bot de **Telegram** (BotFather) al gateway.
- Aplicar postura minima de seguridad (pairing, mention gating, allowlists).
- Implementar acceso administrativo por **Tailscale** (red privada) y endurecer firewall.

138.2 Requisitos Previos

- VM **Ubuntu Server LTS** (22.04 o 24.04) con acceso SSH.
- Lab 4B completado (Docker + Docker Compose v2 instalados) o equivalente.
- Cuenta de Telegram.
- Cuenta de Tailscale (admin del tailnet) o acceso a un tailnet existente.

> ⚠ ADVERTENCIA CRITICA
>
> Vas a trabajar con **tokens/keys** (Telegram y Tailscale).
> **Lo que podria salir mal:** - Si se filtra el token del bot, un atacante puede controlar el bot. - Si se filtra un auth key de Tailscale, un atacante puede unir dispositivos a tu tailnet.
> **Como prevenirlo:** 1. Usa placeholders en apuntes: `**TELEGRAM_BOT_TOKEN**`, `**TAILSCALE_AUTHKEY**`. 2. Guarda secretos en archivos `600` (root-only) y no los subas al repo. 3. Usa claves efimeras (ephemeral) y tags/ACLs en Tailscale.

138.3 Arquitectura del laboratorio (simplificada)

- **Ubuntu Server LTS (VM)**: host donde corre Docker y Tailscale.
- **OpenClaw Gateway (Docker)**: hace polling al Bot API de Telegram (no requiere puertos entrantes desde Internet).
- **Tailscale**: acceso privado al host para administracion (SSH, logs, health checks).

138.4 Paso 1: Preparar el servidor (actualizacion + utilidades)

Listing 138.1 BASH

```
$ sudo apt update                                          ①
$ sudo apt upgrade -y                                      ②
$ sudo apt install -y ca-certificates curl git jq          ③
                                                           ①
                                                           ②
                                                           ③
```

---

① **apt update** actualiza el indice de paquetes.
② **apt upgrade** aplica actualizaciones del sistema.
③ **apt install** instala utilidades para el resto del lab.

138.5 Paso 2: Descargar OpenClaw y levantar el Gateway en Docker

Listing 138.2 BASH

```
$ git clone https://github.com/openclaw/openclaw.git       ①
$ cd openclaw                                              ②
$ ./docker-setup.sh                                        ③
                                                           ①
                                                           ②
                                                           ③
```

---

① **git clone** descarga el repo oficial de OpenClaw.
② **cd openclaw** entra al directorio del proyecto.
③ **docker-setup.sh** deja el gateway listo via Docker Compose (y genera `.env`).

Listing 138.3 BASH

```
$ docker compose ps                                                    ①
```

Verifica que el contenedor este arriba:

① **docker compose ps** confirma servicios, estado y puertos publicados.

138.6 Paso 3: Crear el bot de Telegram (BotFather) y guardar el token

En Telegram:

1. Busca `@BotFather`.
2. Ejecuta `/newbot`.
3. BotFather te entrega un token tipo `123456789:AA...`.

En el servidor, guarda el token como secreto (archivo root-only):

Listing 138.4 BASH

```
$ sudo install -d -m 700 /etc/clawdbot                                 ①
$ sudo bash -lc 'umask 077; printf "%s\n" "TELEGRAM_BOT_TOKEN=**TELEGRAM_BOT_TOKEN**" > /
$ sudo ls -la /etc/clawdbot                                            ③
                                                                       ①
                                                                       ②
                                                                       ③
```

① **install -d** crea un directorio solo accesible por root.
② **secrets.env** guarda el token (placeholder en el ebook; en tu VM usa el token real).
③ **ls -la** verifica permisos (deberia ser `-rw-------`).

138.7 Paso 4: Conectar Telegram al Gateway

Ejecuta el comando de registro del canal Telegram usando el CLI de OpenClaw.

① **set -a** exporta variables del shell (para este ejemplo).
② **source secrets.env** carga el token desde el archivo.
③ **set +a** vuelve al modo normal.

Listing 138.5 BASH

```
$ set -a                                                                    ①
$ sudo bash -lc 'source /etc/clawdbot/secrets.env; env | grep -E "^TELEGRAM_BOT_TOKEN="' (
$ set +a                                                                    ③
                                                                            ①
                                                                            ②
                                                                            ③
```

Ahora registra el canal:

Listing 138.6 BASH

```
$ sudo bash -lc 'source /etc/clawdbot/secrets.env; cd openclaw && docker compose run --rm
```

① **channels add (telegram)** registra el token para que el gateway haga polling al Bot API.

138.8 Paso 5: Postura minima de seguridad (pairing + mention gating + allowlists)

Aplica (o valida) estos criterios minimos:

- DMs en modo **pairing**.
- Grupos con **requireMention: true**.
- Grupos/usuarios restringidos por allowlist (segun tu politica).

Config conceptual (referencia):

Listing 138.7 JSON5

```
{
  channels: {
    telegram: {
      enabled: true,
      dmPolicy: "pairing",
      groups: { "*": { requireMention: true } },
    },
  },
} # <1>
```

① **dmPolicy + requireMention** reduce superficie: no responde a DMs desconocidos ni a grupos sin mencion.

Prueba el flujo de pairing:

1. Desde Telegram, manda un DM al bot.
2. En el servidor, lista solicitudes:

Listing 138.8 BASH

```
$ cd openclaw                                                    ①
$ docker compose run --rm openclaw-cli pairing list telegram     ②
                                                                 ①
                                                                 ②
```

① **cd openclaw** usa el compose correcto.
② **pairing list** muestra solicitudes pendientes.

Aprueba el codigo (usa el codigo real que te entregue el CLI):

Listing 138.9 BASH

```
$ cd openclaw                                                                  ①
$ docker compose run --rm openclaw-cli pairing approve telegram **PAIRING_CODE** ②
                                                                               ①
                                                                               ②
```

① **cd openclaw** mantiene contexto.
② **pairing approve** habilita el DM para ese usuario.

138.9 Paso 6: Instalar Tailscale en Ubuntu Server LTS

Listing 138.10 BASH

```
$ curl -fsSL https://tailscale.com/install.sh | sh    ①
$ tailscale version                                   ②
                                                      ①
                                                      ②
```

① **install.sh** instala Tailscale desde el repositorio oficial.
② **tailscale version** valida que quedo instalado.

Une el servidor al tailnet (usa un auth key real en tu VM):

Listing 138.11 BASH

```
$ sudo tailscale up --authkey **TAILSCALE_AUTHKEY** --hostname clawdbot-lab --ssh --accep ①
$ tailscale status ②
$ tailscale ip -4 ③
① ② ③
```

① **tailscale up** une el host al tailnet; `--ssh` habilita Tailscale SSH (opcional).
② **tailscale status** muestra nodos y estado.
③ **tailscale ip -4** entrega la IP privada del host en el tailnet.

RECOMENDACION

Para seguridad, usa auth keys **efimeras** (ephemeral) y tags/ACLs.
Casos de uso: - Laboratorios temporales (evitar llaves permanentes). - Separar servidores por rol con tags (`tag:clawdbot`).
Cuando aplicar: - Cada vez que agregas un servidor a tu tailnet.

138.10 Paso 7: Endurecer acceso de red (UFW + solo Tailscale)

Objetivo: dejar el host accesible para administracion solo por Tailscale.

1) Activa UFW en modo seguro:

Listing 138.12 BASH

```
$ sudo ufw default deny incoming ①
$ sudo ufw default allow outgoing ②
$ sudo ufw allow in on tailscale0 to any port 22 proto tcp ③
$ sudo ufw enable ④
$ sudo ufw status verbose ⑤
① ② ③ ④ ⑤
```

① **default deny incoming** cierra trafico entrante por defecto.
② **default allow outgoing** permite salidas (Docker pulls, Telegram polling, updates).
③ **allow in on tailscale0** permite SSH solo por la interfaz privada de Tailscale.

④ **ufw enable** activa el firewall.
⑤ **ufw status verbose** evidencia reglas y estado.

2) Confirma que OpenClaw no necesita puertos publicados al publico:

Listing 138.13 BASH

```
$ cd openclaw                                                    ①
$ docker compose ps                                              ②
$ sudo ss -lntp | head -n 30                                     ③
                                                                 ①
                                                                 ②
                                                                 ③
```

① **cd** ubica el stack.
② **docker compose ps** revisa si hay puertos expuestos innecesarios.
③ **ss -lntp** evidencia puertos en escucha.

138.11 Paso 8: Verificacion (salud + logs)

Listing 138.14 BASH

```
$ cd openclaw                                                    ①
$ docker compose exec openclaw-gateway node dist/index.js health --token "$OPENCLAW_GATEW
$ docker compose logs -n 200 openclaw-gateway                    ③
                                                                 ①
                                                                 ②
                                                                 ③
```

① **cd** mantiene contexto del compose.
② **health** valida que el gateway responde correctamente.
③ **docker compose logs** muestra errores de token, rate limits, red, etc.

138.12 Paso 9 (opcional): Acceso por Tailscale desde tu equipo (Multi-OS)

138.12.1 Linux

Listing 138.15 BASH

```
$ curl -fsSL https://tailscale.com/install.sh | sh                    ①
$ sudo tailscale up                                                   ②
$ ssh usuario@**TAILSCALE_IP_DEL_SERVIDOR**                           ③
                                                                      ①
                                                                      ②
                                                                      ③
```

① **install.sh** instala el cliente.
② **tailscale up** autentica el equipo en tu tailnet.
③ **ssh** conecta usando la IP privada de Tailscale.

138.12.2 macOS

Listing 138.16 BASH

```
$ brew install --cask tailscale                                       ①
$ open -a Tailscale                                                   ②
$ ssh usuario@**TAILSCALE_IP_DEL_SERVIDOR**                           ③
                                                                      ①
                                                                      ②
                                                                      ③
```

① **brew install** –**cask** instala la app.
② **open -a** abre Tailscale para login.
③ **ssh** conecta via tailnet.

138.12.3 Windows

Listing 138.17 POWERSHELL

```
PS> winget install Tailscale.Tailscale                                ①
PS> Start-Process "$env:ProgramFiles\Tailscale\tailscale.exe"         ②
PS> ssh usuario@**TAILSCALE_IP_DEL_SERVIDOR**                         ③
                                                                      ①
                                                                      ②
                                                                      ③
```

① **winget install** instala Tailscale.
② **Start-Process** abre la app para autenticar.
③ **ssh** conecta usando la IP privada de Tailscale.

138.13 Entregables (Evidencia)

- `docker compose ps` del stack OpenClaw.
- Evidencia de canal Telegram registrado (salida de `channels add` o `channels status`).
- Evidencia de pairing (list + approve).
- `tailscale status` y `tailscale ip -4`.
- `ufw status verbose` mostrando SSH permitido solo por `tailscale0`.
- Logs del gateway (`docker compose logs -n 200 openclaw-gateway`).

138.14 Checklist de aceptacion

☐ El gateway OpenClaw corre en Docker y aparece en `docker compose ps`.
☐ El bot de Telegram queda conectado (canal Telegram registrado).
☐ DMs quedan protegidos con pairing (evidencia de approve).
☐ El servidor queda unido a Tailscale y tiene IP privada.
☐ UFW queda activo con entrada restringida (SSH solo por `tailscale0`).
☐ Inclui evidencia de health/logs y resolvi al menos 1 problema de conexion si aparecio.

Part XV

Prácticas

139 Practica Unidad 1: Fundamentos de Linux

Listing 139.1 QUARTO-TITLE-BLOCK

140 Practica Unidad 1

140.1 Objetivo

Aplicar los conceptos base de la Unidad 1 en tu propio equipo o VM.

140.2 Ejercicios (15-30 min)

1. Identifica tu distribucion y version.
2. Identifica version del kernel.
3. Explica (en 3 lineas) la diferencia entre kernel y shell.
4. Nombra 3 familias de distribuciones y un caso de uso para cada una.

140.3 Evidencia

- Captura o salida de terminal con: `cat /etc/os-release` y `uname -r`.
- Respuestas en texto (Markdown o PDF).

140.4 Checklist de aceptación

☐ Incluí `cat /etc/os-release` y `uname -r` en la evidencia
☐ Respondí las 4 preguntas con texto claro (3 líneas para kernel vs shell)
☐ Entregué el archivo en el formato solicitado (Markdown o PDF)

141 Practica Unidad 2: Instalacion y Primer Hardening

Listing 141.1 QUARTO-TITLE-BLOCK

142 Practica Unidad 2

142.1 Objetivo

Dejar un servidor listo para seguir el curso: usuarios, SSH y actualizaciones.

142.2 Ejercicios (30-60 min)

1. Verifica que SSH este activo.
2. Crea un usuario administrador (no root) y prueba login.
3. Actualiza el sistema y reinicia si aplica.

142.3 Evidencia

- Salida de: `systemctl status ssh`.
- Evidencia de login por SSH.
- Captura de `uname -a` despues de actualizar.

142.4 Checklist de aceptación

☐ SSH está activo (evidencia de `systemctl status ssh`)
☐ Puedo hacer login por SSH como usuario no-root (evidencia incluida)
☐ El sistema está actualizado (evidencia de `uname -a` después de actualizar)

143 Practica Unidad 3: Terminal y Archivos

Listing 143.1 QUARTO-TITLE-BLOCK

144 Practica Unidad 3

144.1 Objetivo

Ganar fluidez en la terminal para operar un servidor sin GUI.

144.2 Ejercicios (30-60 min)

1. Crea un arbol de directorios de practica y archivos de ejemplo.
2. Usa `find` para localizar archivos por extension.
3. Usa `grep` para filtrar contenido.
4. Ajusta permisos y valida acceso.

144.3 Evidencia

- Comandos ejecutados y resultados (copiados en un documento).

144.4 Checklist de aceptación

☐ Incluí evidencia de creación de directorios/archivos y navegación
☐ Incluí evidencia de búsqueda con `find` y filtrado con `grep`
☐ Ajusté permisos y evidencié el resultado con `ls -la`
☐ Entregué un documento reproducible (comandos + salida)

145 Practica Unidad 4: Docker

Listing 145.1 QUARTO-TITLE-BLOCK

146 Practica Unidad 4

146.1 Objetivo

Entender el flujo imagen -> contenedor y un compose simple.

146.2 Ejercicios (30-60 min)

1. Ejecuta un contenedor Ubuntu y verifica el OS.
2. Crea un contenedor Nginx y publica un puerto.
3. (Opcional) Levanta 2 servicios con Docker Compose.

146.3 Evidencia

- Salidas de `docker ps` y `docker images`.

146.4 Checklist de aceptación

☐ Incluí evidencia de `docker images` y `docker ps`
☐ Publiqué un puerto y validé acceso con `curl` (o equivalente)
☐ Si hice el opcional de Compose, incluí `docker compose ps`

147 Practica Unidad 6: Almacenamiento

Listing 147.1 QUARTO-TITLE-BLOCK

148 Practica Unidad 6

148.1 Objetivo

Practicar el flujo completo: identificar disco -> particionar -> crear FS -> montar -> validar uso de espacio.

148.2 Ejercicios (60-90 min)

1. Inventario del sistema:
   - Ejecuta `lsblk -f` y `df -hT`.
   - Identifica que filesystem esta montado en `/`.
2. Disco de laboratorio (loop):
   - Crea un archivo-disco de 1G con `fallocate`.
   - Asocialo con `losetup -fP`.
3. Particionado y FS:
   - Crea tabla GPT y una particion con `parted --script`.
   - Crea un ext4 con `mkfs.ext4`.
4. Montaje:
   - Monta en `/mnt/u6-data`.
   - Verifica con `df -hT /mnt/u6-data`.
5. Espacio:
   - Usa `sudo du -xh --max-depth=1 /var | sort -hr | head -n 10`.
   - Reporta los 3 directorios mas grandes.
6. (Opcional) RAID/LVM:
   - Crea un RAID1 de laboratorio con `mdadm` (2 discos loop).
   - Muestra `cat /proc/mdstat`.

148.3 Evidencia

- Capturas o salidas de:
  - `lsblk -f`
  - `df -hT`

- `lsblk /dev/loopX`
- `df -hT /mnt/u6-data`
- `cat /proc/mdstat` (si hiciste el opcional)

148.4 Checklist de aceptación

☐ Identifiqué el filesystem montado en `/` con evidencia
☐ Usé un disco de laboratorio (loop) y no un disco real
☐ Monté en `/mnt/u6-data` y evidencié con `df -hT`
☐ Reporté los 3 directorios más grandes de `/var` con evidencia de `du`
☐ (Opcional) Si hice RAID/LVM, incluí `cat /proc/mdstat`

149 Practica Unidad 7: Redes basicas

Listing 149.1 QUARTO-TITLE-BLOCK

150 Practica Unidad 7: Redes basicas

150.1 Objetivo

Recolectar evidencias basicas de conectividad y resolucion en tu servidor.

150.2 Entregables

- Captura o texto de salidas (copiar/pegar) de los comandos pedidos.
- Explicacion breve (1-2 lineas) por evidencia.

150.3 Parte 1: Inventario de red

Listing 150.1 BASH

```
$ ip -br link                                                                    ①
lo     UNKNOWN 00:00:00:00:00:00
enp0s3  UP      08:00:27:12:34:56


$ ip -br addr                                                                    ②
lo     UNKNOWN 127.0.0.1/8 ::1/128
enp0s3  UP      192.168.56.10/24


$ ip route                                                                       ③
default via 192.168.56.1 dev enp0s3
192.168.56.0/24 dev enp0s3 proto kernel scope link src 192.168.56.10


$ ip neigh                                                                       ④
192.168.56.1 dev enp0s3 lladdr 0a:00:27:aa:bb:cc REACHABLE
```

---

① **ip -br link** muestra interfaces y estado.
② **ip -br addr** muestra IP por interfaz.

③ **ip route** muestra rutas y gateway por defecto.
④ **ip neigh** muestra vecinos (ARP/NDP).

150.4 Parte 2: Puertos y servicios

Listing 150.2 BASH

```
$ sudo ss -lntp                                                    ①
State   Recv-Q  Send-Q Local Address:Port  Peer Address:Port  Process
LISTEN  0       4096   0.0.0.0:22         0.0.0.0:*          users:("sshd",pid=812,fd=3)
```

---

① **ss -lntp** valida que SSH esta escuchando y muestra el proceso.

150.5 Parte 3: DNS

Listing 150.3 BASH

```
$ resolvectl status                                                ①
Global
resolv.conf mode: stub

Link 2 (enp0s3)
Current DNS Server: 192.168.56.1


$ getent hosts example.com                                         ②
93.184.216.34 example.com
```

---

① **resolvectl status** muestra DNS efectivo por interfaz.
② **getent hosts** valida resolucion usando el resolver del sistema.

150.6 Checklist de aceptación

- ☐ Interfaces UP y con IP
- ☐ Existe `default route`
- ☐ SSH (22/tcp) esta en escucha
- ☐ DNS resuelve `example.com`

151 Practica Unidad 8: HTTP Basico

Listing 151.1 QUARTO-TITLE-BLOCK

152 Practica Unidad 8

152.1 Objetivo

Probar una comunicacion HTTP local y entender puertos.

152.2 Ejercicios (20-40 min)

1. Levanta un servidor HTTP local en un puerto no estandar (8080).
2. Accede desde navegador y desde `curl`.
3. Identifica que proceso esta usando el puerto.

152.3 Evidencia

- Capturas o salidas de `curl -I http://localhost:8080`.

152.4 Checklist de aceptación

☐ Levanté un servidor HTTP en 8080 y lo probé
☐ Incluí evidencia de `curl -I http://localhost:8080`
☐ Identifiqué el proceso que usa el puerto (evidencia de `ss -lntp` o equivalente)

153 Practica Unidad 9: Apache

Listing 153.1 QUARTO-TITLE-BLOCK

154 Practica Unidad 9: Apache

154.1 Objetivo

Instalar Apache, levantar un sitio simple con VirtualHost, y entregar evidencias (servicio, puerto, respuesta, logs).

154.2 Entregables

- Evidencias (salidas) de comandos solicitados.
- URL/host de prueba usado.

154.3 Parte 1: Instalacion y validacion

Listing 154.1 BASH

```
$ sudo apt update                                                    ①


$ sudo apt install -y apache2                                        ②


$ systemctl status apache2 --no-pager                                ③
  apache2.service - The Apache HTTP Server
     Active: active (running)


$ sudo ss -lntp | grep ':80'                                         ④
LISTEN 0 511 0.0.0.0:80 0.0.0.0:* users:("apache2",pid=1234,fd=4)


$ curl -I http://localhost                                           ⑤
HTTP/1.1 200 OK
Server: Apache/2.4.58 (Ubuntu)
```

---

① **apt update** actualiza el indice de paquetes.
② **apt install apache2** instala Apache.
③ **systemctl status** valida servicio.
④ **ss -lntp** valida puerto 80 en escucha.
⑤ **curl -I** valida respuesta HTTP.

154.4 Parte 2: Crear VirtualHost

Listing 154.2 BASH

```
$ sudo mkdir -p /var/www/site1/public                                    ①

$ echo '<h1>Site1 OK</h1>' | sudo tee /var/www/site1/public/index.html >/dev/null ②

$ sudo tee /etc/apache2/sites-available/site1.conf >/dev/null <<'EOF' # <3>
<VirtualHost *:80>
  ServerName site1.local
  DocumentRoot /var/www/site1/public
  ErrorLog ${APACHE_LOG_DIR}/site1-error.log
  CustomLog ${APACHE_LOG_DIR}/site1-access.log combined
</VirtualHost>
EOF
                                                                         ③

$ sudo a2ensite site1.conf                                               ④

$ sudo apachectl configtest                                              ⑤
Syntax OK

$ sudo systemctl reload apache2                                          ⑥

$ curl -H 'Host: site1.local' http://127.0.0.1                           ⑦
<h1>Site1 OK</h1>
```

---

① **mkdir -p** crea el DocumentRoot.
② **tee** escribe index.html con permisos root.
③ **tee + heredoc** crea el archivo del sitio.
④ **a2ensite** habilita el VirtualHost.

⑤ **apachectl configtest** valida sintaxis.
⑥ **systemctl reload** aplica cambios.
⑦ **curl -H Host** valida el vhost.

154.5 Parte 3: Evidencias de logs

Listing 154.3 BASH

```
$ sudo tail -n 5 /var/log/apache2/site1-access.log                    ①
127.0.0.1 - - [02/Feb/2026:12:10:11 +0000] "GET / HTTP/1.1" 200 15 "-" "curl/8.1.0"
```

---

① **tail** muestra evidencia de request registrado.

154.6 Checklist de aceptación

- ☐ Apache está activo (evidencia de `systemctl status apache2 --no-pager`)
- ☐ El puerto 80 está en escucha (evidencia de `ss -lntp | grep ':80'`)
- ☐ El VirtualHost responde usando `curl -H 'Host: site1.local'`
- ☐ Incluí evidencia de logs (access log)

155 Practica Unidad 10: Nginx y SSL

Listing 155.1 QUARTO-TITLE-BLOCK

156 Practica Unidad 10: Nginx y SSL

156.1 Objetivo

Instalar Nginx, configurar un sitio (server block) y entregar evidencias. (Certbot es opcional si no tienes dominio publico).

156.2 Parte 1: Instalacion y validacion

Listing 156.1 BASH

```
$ sudo apt update                                                    ①


$ sudo apt install -y nginx                                          ②


$ systemctl status nginx --no-pager                                  ③
  nginx.service - A high performance web server and a reverse proxy server
     Active: active (running)


$ curl -I http://localhost                                           ④
HTTP/1.1 200 OK
Server: nginx/1.24.0 (Ubuntu)
```

---

① **apt update** actualiza el indice.
② **apt install nginx** instala Nginx.
③ **systemctl status** valida servicio.
④ **curl -I** valida respuesta.

Listing 156.2 BASH

```
$ sudo tee /etc/nginx/sites-available/app.conf >/dev/null <<'EOF' # <1>
server {
  listen 80;
  server_name app.local;
  location / { return 200 "app ok\n"; }
}
EOF
                                                                    ①

$ sudo ln -sf /etc/nginx/sites-available/app.conf /etc/nginx/sites-enabled/app.conf ②


$ sudo nginx -t                                                     ③
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful


$ sudo systemctl reload nginx                                       ④


$ curl -i -H 'Host: app.local' http://127.0.0.1/                    ⑤
HTTP/1.1 200 OK
Server: nginx

app ok
```

156.3 Parte 2: Server block

① **tee + heredoc** crea el sitio.
② **ln -sf** habilita el sitio.
③ **nginx -t** valida configuracion.
④ **systemctl reload** aplica.
⑤ **curl -H Host** prueba el server block.

156.4 Parte 3 (opcional): Certbot

① **apt install certbot** instala Certbot.
② **certbot –nginx** emite certificado y configura TLS.
③ **certbot renew –dry-run** valida renovacion.

Listing 156.3 BASH

```
$ sudo apt install -y certbot python3-certbot-nginx ①

$ sudo certbot --nginx -d tu-dominio.com ②

$ sudo certbot renew --dry-run ③
```

156.5 Checklist de aceptación

- ☐ Nginx está activo (evidencia de `systemctl status nginx --no-pager`)
- ☐ `nginx -t` pasa sin errores
- ☐ El server block responde con `curl -i -H 'Host: app.local' http://127.0.0.1/`
- ☐ (Opcional) Si usé Certbot, incluí evidencia de emisión o `renew --dry-run`

157 Practica Unidad 11: MariaDB

Listing 157.1 QUARTO-TITLE-BLOCK

158 Practica Unidad 11: MariaDB

158.1 Objetivo

Instalar MariaDB, crear una base y un usuario con privilegios minimos, y generar un backup verificable.

158.2 Parte 1: Instalacion y validacion

Listing 158.1 BASH

```
$ sudo apt update                                                    ①


$ sudo apt install -y mariadb-server                                 ②


$ systemctl status mariadb --no-pager                                ③
 mariadb.service - MariaDB database server
     Active: active (running)


$ sudo mariadb -e 'SELECT VERSION();'                                ④
VERSION()
10.11.6-MariaDB
```

---

① **apt update** actualiza el indice.
② **apt install mariadb-server** instala MariaDB.
③ **systemctl status** valida servicio.
④ **mariadb -e** valida conectividad local.

Listing 158.2 BASH

```
$ sudo mariadb                                                                    ①
MariaDB [(none)]> CREATE DATABASE appdb;
MariaDB [(none)]> CREATE USER 'appuser'@'localhost' IDENTIFIED BY 'AppPass-ChangeMe';
MariaDB [(none)]> GRANT SELECT,INSERT,UPDATE,DELETE ON appdb.* TO 'appuser'@'localhost';
MariaDB [(none)]> FLUSH PRIVILEGES;
MariaDB [(none)]> EXIT;
```

158.3 Parte 2: DB y usuario

① **mariadb** ejecuta comandos SQL para crear DB/usuario y permisos.

158.4 Parte 3: Backup

Listing 158.3 BASH

```
$ sudo mysqldump --databases appdb > /tmp/appdb.sql                               ①


$ ls -lh /tmp/appdb.sql                                                           ②
-rw-r--r-- 1 root root 12K Feb  2 12:45 /tmp/appdb.sql
```

① **mysqldump** genera backup SQL.
② **ls -lh** valida que el archivo existe.

158.5 Checklist de aceptación

- ☐ MariaDB está activo (evidencia de `systemctl status mariadb --no-pager`)
- ☐ Creé DB + usuario con privilegios mínimos (SQL incluido)
- ☐ Generé un backup en `/tmp/appdb.sql` y evidencié su tamaño

159 Practica Unidad 12: Troubleshooting

Listing 159.1 QUARTO-TITLE-BLOCK

160 Practica Unidad 12: Troubleshooting

160.1 Objetivo

Entregar un reporte de diagnostico reproducible con evidencias minimas (contexto, logs, red y recursos).

160.2 Entregables

- Salida de comandos (copiar/pegar) + explicacion breve.

160.3 Parte 1: Contexto

Listing 160.1 BASH

```
$ hostnamectl                                                    ①
 Static hostname: srv01
 Operating System: Ubuntu 24.04.1 LTS


$ uptime                                                         ②
 12:55:03 up 1 day,  3:21,  1 user,  load average: 0.08, 0.11, 0.09
```

---

① **hostnamectl** registra host/OS.
② **uptime** registra carga y uptime.

160.4 Parte 2: Logs de un servicio

① **journalctl -u** entrega evidencia de logs del servicio.

Listing 160.2 BASH

```
$ sudo journalctl -u nginx --since "2 hours ago" --no-pager | tail -n 40 ①
... logs ...
```

160.5 Parte 3: Red

Listing 160.3 BASH

```
$ ip -br addr                                                              ①
enp0s3  UP  192.168.56.10/24


$ ip route | head -n 5                                                     ②
default via 192.168.56.1 dev enp0s3


$ sudo ss -lntp | head -n 20                                               ③
... puertos ...
```

① **ip -br addr** evidencia IP.
② **ip route** evidencia gateway.
③ **ss -lntp** evidencia puertos.

160.6 Parte 4: Recursos

Listing 160.4 BASH

```
$ free -h                                                                  ①
               total        used        free      shared  buff/cache   available
Mem:           3.8Gi       1.1Gi       2.0Gi        52Mi       740Mi       2.5Gi


$ df -h                                                                    ②
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1        49G  5.4G   41G  12% /
```

① **free -h** evidencia memoria.
② **df -h** evidencia espacio.

160.7 Checklist de aceptación

- ☐ Incluí contexto (hostnamectl + uptime)
- ☐ Incluí logs de un servicio (con rango de tiempo)
- ☐ Incluí evidencia de red (IP, rutas, puertos)
- ☐ Incluí evidencia de recursos (memoria y disco)
- ☐ Entregué un reporte reproducible (síntoma, evidencia, hipótesis, fix, validación)

Part XVI

Retos GNU/Linux (Wargames)

161 Retos GNU/Linux (Bandit)

161.1 Introduccion

Este mini-curso usa el wargame **Bandit** de OverTheWire como practica guiada para reforzar:

- Conexion remota por SSH usando un puerto no estandar.
- Navegacion en el sistema de archivos y lectura de archivos.
- Habitos de seguridad basicos (verificacion de huella, no publicar credenciales).

Al final de esta seccion deberias poder pasar de **Bandit Nivel 0** a **Nivel 1** (sin spoilers del password del Nivel 1).

Bandit esta pensado para principiantes: avanzas por niveles, y cada nivel te entrega informacion para entrar al siguiente.

Informacion oficial: https://overthewire.org/wargames/bandit/

161.1.1 Informacion de SSH (Bandit)

Dato	Valor
Host	`bandit.labs.overthewire.org`
Port	`2220`
Usuario	`bandit0` .. `bandit34`
Password inicial	`bandit0`

161.2 Mapa de niveles (Bandit)

Este es el listado oficial de niveles (en el sitio, la pagina del nivel `N` explica el salto `N-1 -> N`).

- Inicio: Nivel 0
- Nivel 0 -> Nivel 1
- Nivel 1 -> Nivel 2
- Nivel 2 -> Nivel 3
- Nivel 3 -> Nivel 4
- Nivel 4 -> Nivel 5

- Nivel 5 -> Nivel 6
- Nivel 6 -> Nivel 7
- Nivel 7 -> Nivel 8
- Nivel 8 -> Nivel 9
- Nivel 9 -> Nivel 10
- Nivel 10 -> Nivel 11
- Nivel 11 -> Nivel 12
- Nivel 12 -> Nivel 13
- Nivel 13 -> Nivel 14
- Nivel 14 -> Nivel 15
- Nivel 15 -> Nivel 16
- Nivel 16 -> Nivel 17
- Nivel 17 -> Nivel 18
- Nivel 18 -> Nivel 19
- Nivel 19 -> Nivel 20
- Nivel 20 -> Nivel 21
- Nivel 21 -> Nivel 22
- Nivel 22 -> Nivel 23
- Nivel 23 -> Nivel 24
- Nivel 24 -> Nivel 25
- Nivel 25 -> Nivel 26
- Nivel 26 -> Nivel 27
- Nivel 27 -> Nivel 28
- Nivel 28 -> Nivel 29
- Nivel 29 -> Nivel 30
- Nivel 30 -> Nivel 31
- Nivel 31 -> Nivel 32
- Nivel 32 -> Nivel 33
- Nivel 33 -> Nivel 34

161.3 Objetivos por nivel (resumen, sin spoilers)

Este resumen toma el **Level Goal** oficial y lo deja en una sola linea por salto.

RECOMENDACION

Este anexo evita publicar writeups completos.
Casos de uso: - Usar Bandit como practica guiada sin exponer soluciones. - Mantener tus notas completas solo en local.
Cuando aplicar: - Para niveles posteriores al 0 -> 1, usa la pagina oficial del nivel como fuente principal.

Salto	Objetivo (resumen)	Pagina oficial
0 -> 1	Password en `readme` en el home de `bandit0`.	https://overthewire.org/wargames/bandit/bandit1.html
1 -> 2	Password en un archivo llamado `-` en el home.	https://overthewire.org/wargames/bandit/bandit2.html
2 -> 3	Password en un archivo llamado `--spaces in this filename--` en el home.	https://overthewire.org/wargames/bandit/bandit3.html
3 -> 4	Password en un archivo oculto dentro del directorio `inhere`.	https://overthewire.org/wargames/bandit/bandit4.html
4 -> 5	Password en el unico archivo legible (human-readable) dentro de `inhere`.	https://overthewire.org/wargames/bandit/bandit5.html
5 -> 6	Password en algun archivo bajo `inhere` con propiedades especificas (size/owner, etc.).	https://overthewire.org/wargames/bandit/bandit6.html
6 -> 7	Password en algun lugar del servidor con propiedades especificas (owner/group/size, etc.).	https://overthewire.org/wargames/bandit/bandit7.html
7 -> 8	Password en `data.txt`, junto a la palabra `millionth`.	https://overthewire.org/wargames/bandit/bandit8.html
8 -> 9	Password en `data.txt`: la unica linea que aparece una sola vez.	https://overthewire.org/wargames/bandit/bandit9.html
9 -> 10	Password en `data.txt`: una de pocas strings legibles, precedida por varios `=`.	https://overthewire.org/wargames/bandit/bandit10.html
10 -> 11	Password en `data.txt` con contenido codificado en base64.	https://overthewire.org/wargames/bandit/bandit11.html
11 -> 12	Password en `data.txt` con ROT13 (a-z/A-Z rotados 13).	https://overthewire.org/wargames/bandit/bandit12.html
12 -> 13	Password en `data.txt`: es un hexdump de un archivo comprimido repetidas veces.	https://overthewire.org/wargames/bandit/bandit13.html
13 -> 14	Password en `/etc/bandit_pass/bandit14` (solo lo lee `bandit14`); debes usar una clave privada para entrar.	https://overthewire.org/wargames/bandit/bandit14.html

Salto	Objetivo (resumen)	Pagina oficial
14 -> 15	Obtener el password enviando el password actual a `localhost:30000`.	https://overthewire.org/wargames/bandit/bandit15.html
15 -> 16	Obtener el password enviando el password actual a `localhost:30001` via SSL/TLS.	https://overthewire.org/wargames/bandit/bandit16.html
16 -> 17	Encontrar el puerto correcto en `31000-32000` y enviar el password actual para obtener credenciales.	https://overthewire.org/wargames/bandit/bandit17.html
17 -> 18	Comparar `passwords.old` vs `passwords.new`: la linea cambiada es el password.	https://overthewire.org/wargames/bandit/bandit18.html
18 -> 19	Password en `readme`, pero `.bashrc` te desloguea al entrar por SSH.	https://overthewire.org/wargames/bandit/bandit19.html
19 -> 20	Usar un binario setuid en el home para obtener el password del nivel.	https://overthewire.org/wargames/bandit/bandit20.html
20 -> 21	Binario setuid que se conecta a `localhost:<puerto>` y reenvia una linea; debes usarlo para obtener el password.	https://overthewire.org/wargames/bandit/bandit21.html
21 -> 22	Revisar `/etc/cron.d/` para ver que comando se ejecuta automaticamente.	https://overthewire.org/wargames/bandit/bandit22.html
22 -> 23	Igual: analizar cron en `/etc/cron.d/` y el comando ejecutado.	https://overthewire.org/wargames/bandit/bandit23.html
23 -> 24	Igual: analizar cron en `/etc/cron.d/` y el comando ejecutado.	https://overthewire.org/wargames/bandit/bandit24.html
24 -> 25	Daemon en `localhost:30002` pide password + pin de 4 digitos.	https://overthewire.org/wargames/bandit/bandit25.html
25 -> 26	Entrar a `bandit26`: su shell no es bash; debes entenderlo y escapar.	https://overthewire.org/wargames/bandit/bandit26.html
26 -> 27	Ya con shell: obtener el password de `bandit27` rapido.	https://overthewire.org/wargames/bandit/bandit27.html

Salto	Objetivo (resumen)	Pagina oficial
27 -> 28	Repo git via SSH en `/home/bandit27-git/repo`; password de `bandit27-git` es el mismo del usuario.	https://overthewire.org/wargames/bandit/bandit28.html
28 -> 29	Repo git via SSH en `/home/bandit28-git/repo`.	https://overthewire.org/wargames/bandit/bandit29.html
29 -> 30	Repo git via SSH en `/home/bandit29-git/repo`.	https://overthewire.org/wargames/bandit/bandit30.html
30 -> 31	Repo git via SSH en `/home/bandit30-git/repo`.	https://overthewire.org/wargames/bandit/bandit31.html
31 -> 32	Repo git via SSH en `/home/bandit31-git/repo`.	https://overthewire.org/wargames/bandit/bandit32.html
32 -> 33	Escape adicional (despues de git).	https://overthewire.org/wargames/bandit/bandit33.html
33 -> 34	Ver instrucciones del nivel final en la pagina oficial.	https://overthewire.org/wargames/bandit/bandit34.html

161.4 Alternativas cuando te atascas (sin spoilers)

Si te quedas bloqueado en un nivel, estas son las alternativas recomendadas por OverTheWire.

161.4.1 1) Manual (man pages)

Listing 161.1 BASH

```
$ man ls                                                    ①
$ man ssh                                                   ②
$ man 5 crontab                                             ③
                                                            ①
                                                            ②
                                                            ③
```

① **man ls** explica opciones, formatos de salida y ejemplos; presiona `q` para salir.
② **man ssh** ayuda a entender flags como `-p`, `-i`, `-o`.
③ **man 5 crontab** abre la documentacion del formato de crontab (seccion 5).

161.4.2 2) Help para built-ins del shell

Listing 161.2 BASH

```
$ help cd                                                    ①
$ help export                                                ②
                                                             ①
                                                             ②
```

① **help cd** muestra ayuda de un comando built-in (no siempre tiene man page).
② **help export** ayuda con variables de entorno (util en niveles con scripts).

161.4.3 3) Buscar y validar hipotesis

- Si ya sabes el nombre del comando pero no recuerdas el uso, busca: `"<comando> example"`.
- Si el nivel menciona un concepto (ej: “base64”, “strings”, “cron”), busca el concepto y vuelve al manual.

161.4.4 4) Comunidad

- Si sigues atascado, puedes unirte al chat oficial: https://overthewire.org/information/chat.html

---

161.5 Nota para VMs: error “broken pipe” en SSH (modo NAT)

En algunas VMs, SSH puede fallar con un error tipo “broken pipe” cuando el adaptador de red esta en **NAT**.

La recomendacion de OverTheWire es agregar `IPQoS throughput` a `/etc/ssh/ssh_config`.

> ⚠ ADVERTENCIA CRITICA
>
> Modificar configuracion SSH en un servidor real puede cortar conexiones.
> **Lo que podria salir mal:** - Dejar SSH con configuracion invalida o no deseada.
> **Como prevenirlo:** 1. Aplica esto solo en tu VM de laboratorio. 2. Mantiene una sesion abierta mientras pruebas. 3. Si no funciona, cambia el adaptador a modo **Bridged**.

Ejemplo (VM Ubuntu Server LTS):

① **cp** crea un backup de la configuracion del cliente SSH.
② **tee -a** agrega `IPQoS throughput` para evitar problemas de QoS en NAT.
③ **grep -n** verifica que la linea quedo escrita.

Listing 161.3 BASH

```
$ sudo cp /etc/ssh/ssh_config /etc/ssh/ssh_config.bak                    ①
$ printf "\nHost *\n  IPQoS throughput\n" | sudo tee -a /etc/ssh/ssh_config >/dev/null ②
$ grep -n "IPQoS" /etc/ssh/ssh_config                                    ③
                                                                         ①
                                                                         ②
                                                                         ③
```

161.5.1 Comandos que puedes necesitar

En este salto (Nivel 0 -> Nivel 1) normalmente basta con:

- `ls`, `cd`, `cat`, `file`, `du`, `find`

> **NOTA**
>
> **Las contrasenas no se guardan automaticamente y pueden cambiar.** Si no las registras, tendras que empezar de nuevo.

> **RECOMENDACION**
>
> **Crea un archivo de notas local para usuarios y passwords.**
> **Casos de uso:** - Guardar como te conectaste (host, puerto, usuario). - Registrar como resolviste cada nivel sin publicar la respuesta.
> **Cuando aplicar:** - Cada vez que completes un nivel.

> **ADVERTENCIA CRITICA**
>
> **No publiques passwords ni writeups completos de niveles.**
> **Lo que podria salir mal:** - Te banean del juego. - Compartes credenciales que cambian o que otros estan usando.
> **Como prevenirlo:** 1. Escribe tus notas en tu maquina local. 2. En documentacion publica, usa placeholders como `**PASSWORD_NIVEL_1**`.

161.6 Reto: Bandit Nivel 0 -> Nivel 1

161.6.1 Objetivo del nivel

La contrasena para el siguiente nivel esta en un archivo llamado `readme` ubicado en tu directorio de inicio. Una vez que la encuentres, inicia sesion como `bandit1` por SSH (puerto 2220) y continua el juego.

> NOTA
>
> **En SSH no se muestra lo que escribes cuando te pide password.** Esto es normal.

161.6.2 Datos de conexion (Nivel 0)

- Host: `bandit.labs.overthewire.org`
- Puerto: `2220`
- Usuario: `bandit0`
- Password inicial: `bandit0`

161.7 Errores comunes al conectarte por SSH

Un error tipico es escribir `host:puerto` (esto no define el puerto, se interpreta como parte del hostname).

161.7.1 Linux

Listing 161.4 BASH

```
$ ssh bandit0@bandit.labs.overthewire.org:2220                    (1)
ssh: Could not resolve hostname bandit.labs.overthewire.org:2220: Name or service not kno
```

(1) **ssh** no acepta `host:puerto` en ese formato; debes pasar el puerto con `-p`.

161.7.2 macOS

Listing 161.5 BASH

```
$ ssh bandit0@bandit.labs.overthewire.org:2220                    (1)
ssh: Could not resolve hostname bandit.labs.overthewire.org:2220: nodename nor servname p
```

(1) **ssh** en macOS muestra el mismo problema: el `:2220` se vuelve parte del hostname.

161.7.3 Windows

(1) **ssh** en Windows (OpenSSH) tambien interpreta `:2220` como parte del nombre del host.

Listing 161.6 POWERSHELL

```
PS> ssh bandit0@bandit.labs.overthewire.org:2220                    ①
ssh: Could not resolve hostname bandit.labs.overthewire.org:2220: No such host is known.
```

161.8 Conexion correcta (puerto 2220)

161.8.1 Linux

Listing 161.7 BASH

```
$ ssh -p 2220 bandit0@bandit.labs.overthewire.org                   ①
The authenticity of host '[bandit.labs.overthewire.org]:2220 (51.21.210.216)' can't be es
ED25519 key fingerprint is SHA256:C2ihUBV7ihnV1wUXRb4RrEcLfXC5CXlhmAAM/urerLY.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
bandit0@bandit.labs.overthewire.org's password:
```

① **ssh -p** define el puerto remoto; aqui 2220.

161.8.2 macOS

Listing 161.8 BASH

```
$ ssh -p 2220 bandit0@bandit.labs.overthewire.org                   ①
The authenticity of host '[bandit.labs.overthewire.org]:2220 ([51.21.210.216]:2220)' can'
ED25519 key fingerprint is SHA256:C2ihUBV7ihnV1wUXRb4RrEcLfXC5CXlhmAAM/urerLY.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
bandit0@bandit.labs.overthewire.org's password:
```

① **ssh -p** es la forma correcta de usar un puerto no estandar en macOS.

161.8.3 Windows

① **ssh -p** tambien funciona igual en Windows con OpenSSH.

Listing 161.9 POWERSHELL

```
PS> ssh -p 2220 bandit0@bandit.labs.overthewire.org                    (1)
The authenticity of host '[bandit.labs.overthewire.org]:2220 (51.21.210.216)' can't be es
ED25519 key fingerprint is SHA256:C2ihUBV7ihnV1wUXRb4RrEcLfXC5CXlhmAAM/urerLY.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
bandit0@bandit.labs.overthewire.org's password:
```

161.9 Ejemplos Practicos Multi-SO

> ⚠ ADVERTENCIA CRITICA
>
> **No aceptes huellas (fingerprints) a ciegas en servidores reales.**
> **Lo que podria salir mal:** - Ataque de tipo Man-in-the-Middle.
> **Como prevenirlo:** 1. Verifica la huella publicada por el proveedor/infra. 2. Si cambia inesperadamente, detente y valida con el administrador.

161.10 Dentro del servidor: encontrar el password del Nivel 1

Una vez dentro como `bandit0`, tu objetivo es leer el archivo `readme` en el home.

Listing 161.10 BASH

```
bandit0@bandit:~$ ls                                                   (1)
readme
```

(1) **ls** lista el contenido del directorio actual; aqui ves el archivo `readme`.

Listing 161.11 BASH

```
bandit0@bandit:~$ file readme                                          (1)
readme: ASCII text
```

(1) **file** identifica el tipo de archivo; en este caso es texto plano.

(1) **du -h** muestra el tamano del archivo de forma legible.

(1) **find** sirve para ubicar archivos por nombre; aqui busca `readme` en tu home.

(1) **cat** imprime el contenido de `readme`; guarda ese valor en tus notas como password del Nivel 1.

Listing 161.12 BASH

```
bandit0@bandit:~$ du -h readme                                    ①
4.0K\treadme
```

Listing 161.13 BASH

```
bandit0@bandit:~$ find ~ -maxdepth 1 -type f -name readme         ①
/home/bandit0/readme
```

161.11 Entrar al Nivel 1

Cuando ya tengas el password, sales y te conectas como `bandit1`.

① **exit** cierra tu sesion SSH actual.

① **ssh -p** repite el patron, cambiando el usuario a `bandit1`.

161.11.1 Si ves "Permission denied"

Si escribes mal el password, SSH te lo dira.

① **Permission denied** casi siempre significa password incorrecto (o usuario equivocado).

Solucion (oculta): Nivel 0 -> Nivel 1

Objetivo: leer `~/readme` y usar ese valor como password para `bandit1`.

① **ssh -p** conecta al puerto `2220` con el usuario `bandit0`.

① **cat** imprime el contenido del archivo `readme` del home; ese es el password del siguiente nivel.

① **exit** cierra la sesion actual para reconectarte como `bandit1`.

① **ssh -p** repite el patron con el usuario del siguiente nivel.

161.12 Mini-guia para tus notas (local)

Una forma simple es guardar un archivo de texto en tu equipo.

Listing 161.14 BASH

```
bandit0@bandit:~$ cat readme                                                    ①
**PASSWORD_NIVEL_1**
```

Listing 161.15 BASH

```
bandit0@bandit:~$ exit                                                          ①
logout
Connection to bandit.labs.overthewire.org closed.
```

161.12.1 Linux

① **mkdir -p** crea una carpeta para tus notas.
② **printf** agrega una linea al archivo de notas.
③ **cat** verifica el contenido guardado.

161.12.2 macOS

① **mkdir -p** crea una carpeta para tus notas.
② **printf** agrega una linea al archivo de notas.
③ **cat** verifica el contenido guardado.

161.12.3 Windows

① **New-Item** crea la carpeta de notas en PowerShell.
② **Add-Content** agrega una linea al archivo de notas.
③ **Get-Content** verifica el contenido guardado.

161.13 Retos: Nivel 1 -> Nivel 34 (sin spoilers)

Estas secciones te dan el **objetivo**, los **comandos tipicos** y 2-3 **pistas** por salto.

> ⚠ ADVERTENCIA CRITICA
>
> No publiques passwords ni writeups completos.
> **Lo que podria salir mal:** - Compartir la solucion exacta elimina el aprendizaje.
> **Como prevenirlo:** 1. Guarda el password de cada nivel en un archivo local. 2. En tu documentacion, usa placeholders como `**PASSWORD_NIVEL_2**`.

Listing 161.16 BASH

```
$ ssh -p 2220 bandit1@bandit.labs.overthewire.org                    ①
bandit1@bandit.labs.overthewire.org's password:
 # <1>
```

Listing 161.17 BASH

```
$ ssh -p 2220 bandit0@bandit.labs.overthewire.org                    ①
bandit0@bandit.labs.overthewire.org's password:
Permission denied, please try again.
 # <1>
```

161.13.1 Bandit Nivel 1 -> Nivel 2

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit2.html
- **Objetivo:** el password esta en un archivo llamado `-` en el home.
- **Comandos tipicos:** `ls`, `cat`.
- **Pistas (sin spoilers):**
  - Un nombre que empieza con `-` puede interpretarse como opcion; usa `--` o un prefijo como `./`.
  - Tab completion ayuda a evitar errores con nombres raros.

161.13.2 Bandit Nivel 2 -> Nivel 3

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit3.html
- **Objetivo:** el password esta en un archivo con espacios en el nombre.
- **Comandos tipicos:** `ls`, `cat`.
- **Pistas (sin spoilers):**
  - Usa comillas (`"..."`) o escapes (\) para rutas con espacios.
  - El autocompletado de la terminal suele completar el nombre correctamente.

161.13.3 Bandit Nivel 3 -> Nivel 4

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit4.html
- **Objetivo:** el password esta en un archivo oculto dentro del directorio `inhere`.
- **Comandos tipicos:** `ls`, `cd`, `cat`.
- **Pistas (sin spoilers):**
  - Los archivos ocultos empiezan con `.`; usa listado que muestre ocultos.
  - Revisa que estas realmente dentro de `inhere` antes de leer.

Listing 161.18 BASH

```
$ ssh -p 2220 bandit0@bandit.labs.overthewire.org                    ①
bandit0@bandit.labs.overthewire.org's password:
 # <1>
```

Listing 161.19 BASH

```
bandit0@bandit:~$ cat ~/readme                                       ①
**PASSWORD_NIVEL_1**
```

161.13.4 Bandit Nivel 4 -> Nivel 5

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit5.html
- **Objetivo:** el password esta en el unico archivo "human-readable" dentro de `inhere`.
- **Comandos tipicos:** `ls`, `file`, `cat`.
- **Pistas (sin spoilers):**
  - Usa `file` para clasificar rapidamente los archivos.
  - Si tu terminal queda "rara" por imprimir binarios, el sitio sugiere `reset`.

161.13.5 Bandit Nivel 5 -> Nivel 6

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit6.html
- **Objetivo:** el password esta en algun archivo bajo `inhere` con propiedades especificas.
- **Comandos tipicos:** `find`, `du`, `file`, `cat`.
- **Pistas (sin spoilers):**
  - `find` puede filtrar por tamano (`-size`), tipo (`-type f`) y permisos (`-readable`).
  - Cuando identifiques candidatos, valida con `file` y luego lee con `cat`.

161.13.6 Bandit Nivel 6 -> Nivel 7

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit7.html
- **Objetivo:** el password esta en algun lugar del servidor con propiedades especificas.
- **Comandos tipicos:** `find`, `grep`.
- **Pistas (sin spoilers):**
  - Filtra por usuario/grupo (`-user`, `-group`) y por tamano (`-size`).
  - Si hay muchos "Permission denied", redirige stderr a `/dev/null`.

161.13.7 Bandit Nivel 7 -> Nivel 8

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit8.html
- **Objetivo:** el password esta en `data.txt`, junto a la palabra `millionth`.

Listing 161.20 BASH

```
bandit0@bandit:~$ exit                                                    (1)
logout
Connection to bandit.labs.overthewire.org closed.
```

Listing 161.21 BASH

```
$ ssh -p 2220 bandit1@bandit.labs.overthewire.org                         (1)
bandit1@bandit.labs.overthewire.org's password:
 # <1>
```

- **Comandos tipicos:** `grep`, `cat`.
- **Pistas (sin spoilers):**
  - Busca la linea que contiene la palabra; `grep` te da contexto rapido.
  - Si el archivo es grande, evita abrirlo completo; filtra.

161.13.8 Bandit Nivel 8 -> Nivel 9

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit9.html
- **Objetivo:** el password es la unica linea en `data.txt` que aparece una sola vez.
- **Comandos tipicos:** `sort`, `uniq`.
- **Pistas (sin spoilers):**
  - `uniq` trabaja bien cuando el input esta ordenado.
  - Busca el modo de `uniq` que muestra solo lineas unicas.

161.13.9 Bandit Nivel 9 -> Nivel 10

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit10.html
- **Objetivo:** el password esta en una de las pocas strings legibles del archivo, precedida por varios `=`.
- **Comandos tipicos:** `strings`, `grep`.
- **Pistas (sin spoilers):**
  - `strings` extrae texto imprimible desde binarios.
  - Filtra con `grep` por un patron que incluya `=`.

161.13.10 Bandit Nivel 10 -> Nivel 11

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit11.html
- **Objetivo:** el password esta en `data.txt` y esta codificado en base64.
- **Comandos tipicos:** `base64`.
- **Pistas (sin spoilers):**

Listing 161.22 BASH

```
$ mkdir -p "$HOME/notas"                                                    ①
 $ printf "%s\n" "Bandit0->Bandit1: **PASSWORD_NIVEL_1**" >> "$HOME/notas/bandit.txt" ②
$ cat "$HOME/notas/bandit.txt"                                              ③
 Bandit0->Bandit1: **PASSWORD_NIVEL_1**
                                                                            ①
                                                                            ②
                                                                            ③
```

Listing 161.23 BASH

```
$ mkdir -p "$HOME/notas"                                                    ①
 $ printf "%s\n" "Bandit0->Bandit1: **PASSWORD_NIVEL_1**" >> "$HOME/notas/bandit.txt" ②
$ cat "$HOME/notas/bandit.txt"                                              ③
 Bandit0->Bandit1: **PASSWORD_NIVEL_1**
                                                                            ①
                                                                            ②
                                                                            ③
```

  - Necesitas decodificar base64 (no es cifrado).
  - Si hay saltos de linea, igual suele funcionar; valida el resultado como texto.

161.13.11 Bandit Nivel 11 -> Nivel 12

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit12.html
- **Objetivo:** el password esta en `data.txt` con ROT13 aplicado.
- **Comandos tipicos:** `tr`.
- **Pistas (sin spoilers):**
  - ROT13 es una sustitucion simple; `tr` puede mapear A-Z/a-z a su rotacion.
  - No olvides mapear mayusculas y minusculas.

161.13.12 Bandit Nivel 12 -> Nivel 13

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit13.html
- **Objetivo:** `data.txt` es un hexdump de un archivo comprimido repetidas veces.
- **Comandos tipicos:** `xxd`, `file`, `tar`, `gzip`, `bzip2`.
- **Pistas (sin spoilers):**
  - Convierte el hexdump a binario y luego usa `file` para saber que tipo de compresion hay.
  - Repite el ciclo: “identificar -> descomprimir -> volver a identificar” hasta llegar a texto.

Listing 161.24 POWERSHELL

```
PS> New-Item -ItemType Directory -Force "$HOME\\notas"                    ①
 PS> Add-Content -Path "$HOME\\notas\\bandit.txt" -Value "Bandit0->Bandit1: **PASSWORD_NI
PS> Get-Content "$HOME\\notas\\bandit.txt"                                ③
 Bandit0->Bandit1: **PASSWORD_NIVEL_1**
                                                                          ①
                                                                          ②
                                                                          ③
```

161.13.13 Bandit Nivel 13 -> Nivel 14

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit14.html
- **Objetivo:** leer el password de `/etc/bandit_pass/bandit14` usando una clave privada (no se entrega el password directo).
- **Comandos tipicos:** `ssh`, `chmod`.
- **Pistas (sin spoilers):**
  - Asegura permisos de la clave (ej: `chmod 600`).
  - Conecta usando `ssh -i` y el puerto `2220`.

161.13.14 Bandit Nivel 14 -> Nivel 15

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit15.html
- **Objetivo:** enviar el password actual a `localhost:30000` para recibir el siguiente.
- **Comandos tipicos:** `nc`.
- **Pistas (sin spoilers):**
  - Esto ocurre dentro del servidor (ya logueado).
  - Conecta a `localhost` (no al host remoto) y pega el password actual.

161.13.15 Bandit Nivel 15 -> Nivel 16

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit16.html
- **Objetivo:** enviar el password actual a `localhost:30001` usando SSL/TLS.
- **Comandos tipicos:** `openssl s_client`.
- **Pistas (sin spoilers):**
  - `nc` no cifra; aqui necesitas un cliente TLS.
  - Verifica que estas conectando al puerto correcto en `localhost`.

161.13.16 Bandit Nivel 16 -> Nivel 17

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit17.html
- **Objetivo:** encontrar el servicio correcto en `31000-32000` y obtener credenciales enviando el password actual.

- **Comandos tipicos:** `nmap`, `openssl s_client`, `ss`.
- **Pistas (sin spoilers):**
  - Primero identifica que puertos estan abiertos en el rango.
  - No todos los puertos hablan TLS; prueba con un cliente TLS donde corresponda.

161.13.17 Bandit Nivel 17 -> Nivel 18

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit18.html
- **Objetivo:** el password esta en `passwords.new` y es la unica linea distinta frente a `passwords.old`.
- **Comandos tipicos:** `diff`.
- **Pistas (sin spoilers):**
  - Usa `diff` para ver cambios exactos.
  - Extrae solo la linea nueva (la que se agrego/cambio).

161.13.18 Bandit Nivel 18 -> Nivel 19

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit19.html
- **Objetivo:** el password esta en `readme`, pero al entrar por SSH te desloguean por `.bashrc`.
- **Comandos tipicos:** `ssh`, `cat`.
- **Pistas (sin spoilers):**
  - Puedes pedirle a `ssh` que ejecute un comando remoto sin iniciar un shell interactivo.
  - Otra alternativa es forzar un shell distinto o evitar cargar configuracion, segun el caso.

161.13.19 Bandit Nivel 19 -> Nivel 20

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit20.html
- **Objetivo:** usar un binario setuid en el home para leer el password del siguiente usuario.
- **Comandos tipicos:** `ls`, `cat`.
- **Pistas (sin spoilers):**
  - Revisa permisos del binario (`ls -la`) y ejecutalo para ver como se usa.
  - El archivo de passwords suele estar en `/etc/bandit_pass/`.

161.13.20 Bandit Nivel 20 -> Nivel 21

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit21.html
- **Objetivo:** usar un binario setuid que se conecta a `localhost:<puerto>` y reenvia una linea.
- **Comandos tipicos:** `nc`, `tmux`/`screen`.

- **Pistas (sin spoilers):**
  - Necesitas dos terminales: una escucha, la otra ejecuta el binario.
  - Si no tienes dos sesiones, usa job control (`&`, `fg`, `bg`) o `tmux`.

161.13.21 Bandit Nivel 21 -> Nivel 22

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit22.html
- **Objetivo:** descubrir que programa corre por cron mirando `/etc/cron.d/`.
- **Comandos tipicos:** `cat`, `ls`.
- **Pistas (sin spoilers):**
  - Lee el archivo de cron y luego sigue la ruta del script que ejecuta.
  - El script normalmente deja evidencia (archivos temporales o salidas).

161.13.22 Bandit Nivel 22 -> Nivel 23

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit23.html
- **Objetivo:** analizar cron y el comando que ejecuta.
- **Comandos tipicos:** `cat`, `grep`.
- **Pistas (sin spoilers):**
  - Identifica que usuario ejecuta el job y que archivos toca.
  - Corre el script manualmente (sin modificarlo) para entender que imprime/crea.

161.13.23 Bandit Nivel 23 -> Nivel 24

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit24.html
- **Objetivo:** analizar cron y el comando que ejecuta.
- **Comandos tipicos:** `cat`, `bash`.
- **Pistas (sin spoilers):**
  - Revisa permisos de scripts y directorios que usa el job.
  - Si el job genera archivos en `/tmp`, busca patrones por nombre.

161.13.24 Bandit Nivel 24 -> Nivel 25

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit25.html
- **Objetivo:** daemon en `localhost:30002` pide password + pin de 4 digitos.
- **Comandos tipicos:** `nc`, `bash`.
- **Pistas (sin spoilers):**
  - Si no hay forma de derivar el pin, piensa en fuerza bruta controlada (0000-9999).
  - Automatiza: genera intentos y filtra solo la respuesta valida.

161.13.25 Bandit Nivel 25 -> Nivel 26

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit26.html
- **Objetivo:** entrar como `bandit26`, pero su shell no es bash; debes entenderlo y escapar.
- **Comandos tipicos:** `ssh`, `more`, `vi`.
- **Pistas (sin spoilers):**
  - Revisa en `/etc/passwd` cual es el shell real de ese usuario.
  - Si caes en un paginador/editor, investiga como abrir un shell desde ahi.

161.13.26 Bandit Nivel 26 -> Nivel 27

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit27.html
- **Objetivo:** con una shell funcional, obtener el password de `bandit27`.
- **Comandos tipicos:** `cat`, `ls`.
- **Pistas (sin spoilers):**
  - Sigue el patron `/etc/bandit_pass/`.
  - No olvides guardar el password como `**PASSWORD_NIVEL_27**` en tus notas.

161.13.27 Bandit Nivel 27 -> Nivel 28

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit28.html
- **Objetivo:** clonar un repo git via SSH y encontrar el password.
- **Comandos tipicos:** `git`, `ssh`.
- **Pistas (sin spoilers):**
  - Usa `ssh://user@host:port/...` o configura el puerto en la URL.
  - Revisa commits y contenido del repo (`git log`, `git show`).

161.13.28 Bandit Nivel 28 -> Nivel 29

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit29.html
- **Objetivo:** repo git via SSH; el password no siempre esta en `master`.
- **Comandos tipicos:** `git`.
- **Pistas (sin spoilers):**
  - Lista ramas (`git branch -a`) y revisa commits por rama.
  - Busca en el historial, no solo en el ultimo commit.

161.13.29 Bandit Nivel 29 -> Nivel 30

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit30.html
- **Objetivo:** repo git via SSH; puede involucrar tags.
- **Comandos tipicos:** `git`.
- **Pistas (sin spoilers):**

- Lista tags (`git tag`) y revisa que contienen (`git show <tag>`).
- Explora referencias: ramas remotas, tags y commits antiguos.

161.13.30 Bandit Nivel 30 -> Nivel 31

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit31.html
- **Objetivo:** repo git via SSH; hay una accion especifica que debes realizar.
- **Comandos tipicos:** `git`.
- **Pistas (sin spoilers):**
  - Lee el `README` dentro del repo (a veces dice exactamente que hacer).
  - Puede requerir crear/modificar un archivo y hacer commit/push.

161.13.31 Bandit Nivel 31 -> Nivel 32

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit32.html
- **Objetivo:** repo git via SSH; sigue las instrucciones del repo.
- **Comandos tipicos:** `git`.
- **Pistas (sin spoilers):**
  - Si hay hooks o validaciones, lee los mensajes del servidor.
  - Revisa ramas/tags e historial si el password no esta a simple vista.

161.13.32 Bandit Nivel 32 -> Nivel 33

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit33.html
- **Objetivo:** escape adicional (despues de git).
- **Comandos tipicos:** `sh`, `man`.
- **Pistas (sin spoilers):**
  - Si hay un programa/shell raro, prueba rutas absolutas (`/bin/bash`, `/bin/sh`).
  - A veces el truco es como se interpretan mayusculas/minusculas o variables.

161.13.33 Bandit Nivel 33 -> Nivel 34

- Pagina oficial: https://overthewire.org/wargames/bandit/bandit34.html
- **Objetivo:** sigue las instrucciones del nivel final en la pagina oficial.
- **Comandos tipicos:** depende del nivel.
- **Pistas (sin spoilers):**
  - Lee la pagina oficial completa: suele incluir notas, archivos y paths clave.
  - Mantente en el patron: encontrar informacion en archivos/sistema y usarla para el siguiente salto.

RECOMENDACION

Mejores practicas

Aspecto	Recomendacion
Puerto SSH	Usa `ssh -p 2220 ...` (no `host:puerto`).
Seguridad	Verifica la huella cuando sea un servidor real.
Orden	Anota: `nivel`, `usuario`, `password`, `comandos`, `resultado`.
Etica	No publiques passwords ni soluciones completas.

161.14 Resumen

- El puerto en SSH se especifica con `-p`.
- En Bandit, el Nivel 0 se trata de conectarte por SSH y leer `readme`.
- Guarda el password del siguiente nivel en tus notas locales y continua con `bandit1`.

161.15 Referencias

- OverTheWire Bandit: https://overthewire.org/wargames/bandit/
- SSH (Wikipedia): https://en.wikipedia.org/wiki/Secure_Shell
- Manual `ssh(1)`: https://man.openbsd.org/ssh

Part XVII

Presentaciones (Reveal.js)

162 Presentaciones (Reveal.js)

Listing 162.1 QUARTO-TITLE-BLOCK

162.1 Acceso a las diapositivas

Estas presentaciones se generan con Reveal.js y se publican dentro de `docs/presentaciones/`.

- Curso: Diapositivas - Presentación del curso
- Unidad 1: Diapositivas - Unidad 1
- Unidad 2: Diapositivas - Unidad 2
- Unidad 3: Diapositivas - Unidad 3
- Unidad 4: Diapositivas - Unidad 4
- Unidad 5: Diapositivas - Unidad 5
- Unidad 6: Diapositivas - Unidad 6
- Unidad 7: Diapositivas - Unidad 7
- Unidad 8: Diapositivas - Unidad 8
- Unidad 9: Diapositivas - Unidad 9
- Unidad 10: Diapositivas - Unidad 10
- Unidad 11: Diapositivas - Unidad 11
- Unidad 12: Diapositivas - Unidad 12

162.2 Render

Para regenerarlas:

Listing 162.2 BASH

```
$ quarto render presentaciones                                    ①
```

① **quarto render** ejecuta el subproyecto de `presentaciones/` y escribe las diapositivas en `docs/presentaciones/`.

Part XVIII

Anexos: Mini-Cursos (Nivel 1: Fundamentos)

163 Anexo B: Editores Vi y Nvim - Mini-Curso Práctico

Listing 163.1 QUARTO-TITLE-BLOCK

164 Anexo B: Editores Vi y Nvim - Mini-Curso Práctico

164.1 Introducción

Vi/Vim/Nvim son editores de texto basados en terminal disponibles en TODO servidor Linux. No tienen interfaz gráfica, por lo que son **imprescindibles** para administrar servidores remotos vía SSH.

- **Vi**: Editor clásico (1976), disponible en cualquier Unix
- **Vim**: "Vi Improved" (1991), extensión mejorada de Vi
- **Neovim**: "Next generation Vim" (2014), moderno, extensible

Este mini-curso te enseña **solo lo que necesitas** para editar archivos rápidamente en Abacom.

En este anexo aprenderás:

- Modos de Vi/Vim (Normal, Insert, Command)
- Navegación y movimiento
- Edición básica (insertar, copiar, pegar, borrar)
- Búsqueda y reemplazo
- Guardar y salir
- Configuración básica
- Diferencias Vi vs Vim vs Nvim

Duración estimada: 90 minutos de práctica

164.2 ¿Vi? ¿Vim? ¿Nvim? ¿Cuál Uso?

En producción (servidores remotos):

- Usa **Vi** o **Vim** - Siempre está disponible
- Nvim puede no estar instalado en servidores antiguos

En tu máquina de desarrollo:

- Usa **Nvim** - Moderno, más funcionalidad, configuración mejorada
- O **Vim** si Nvim no está disponible

Para Abacom:

- Aprende **Vim** (compatible con Vi, pero mejor)
- Ten Nvim en tu máquina local para trabajo cómodo

164.3 Editores en Diferentes SOs

164.3.1 Linux (Vim/Nvim)

Listing 164.1 BASH

```
# Verificar si está instalado
$ which vim
/usr/bin/vim

$ which nvim
/usr/bin/nvim

# Instalar si no está
$ sudo apt install vim vim-nox  # Linux
$ sudo apt install neovim       # Para Nvim

# Usar Vim
$ vim archivo.txt
# O Nvim
$ nvim archivo.txt

# Configuración en ~/.vimrc o ~/.config/nvim/init.vim
```

Disponibilidad:

- Vim: Siempre disponible en servidores
- Nvim: Disponible en máquinas modernas
- Recomendado: **Vim** en producción, **Nvim** en desarrollo

164.3.2 macOS (Vim/Nvim)

Disponibilidad:

- Vi: Versión antigua (preinstalada)
- Vim: Versión antigua o no instalada
- Nvim: Disponible via Homebrew
- Recomendado: **Instalar Vim/Nvim via Homebrew**

Listing 164.2 BASH

```
# macOS viene con Vi preinstalado
$ which vi
/usr/bin/vi

# Verificar Vim
$ which vim
/usr/bin/vim

# O instalar via Homebrew
$ brew install vim
$ brew install neovim

# Usar Vim
$ vim archivo.txt

# Configuración típica:
$ ~/.vimrc
$ ~/.config/nvim/init.vim

# Nota: En macOS, Vim puede ser versión antigua
# Mejor usar Homebrew para instalar versión más nueva
```

164.3.3 Windows (WSL/Git Bash/VS Code)

Disponibilidad:

- Vim: No disponible nativamente
- Vi: Solo via WSL
- Nvim: Via Chocolatey o manual
- VS Code + Vim extension: Recomendado para principiantes

Comparación rápida: | Aspecto | Linux | macOS | Windows | |———|——-|——-|———| | Vim disponible | Sí | Versión vieja | No | | Nvim disponible | Fácil | Homebrew | Chocolatey | | Recomendado para producción | Vim | Vim | WSL + Vim | | Recomendado para desarrollo | Nvim | Nvim | VS Code + Vim |

164.4 Los 3 Modos de Vim

164.4.1 Modo Normal (Command Mode)

Aquí **accedes** pero NO ESCRIBES. Es donde pasas más tiempo.

Listing 164.3 POWERSHELL

```
# Windows NO tiene Vim nativo
# Opciones:

# OPCIÓN 1: Windows Subsystem for Linux (WSL)
PS> wsl vim archivo.txt
# Accede a Vim via WSL (requiere Ubuntu/Debian en WSL)

# OPCIÓN 2: Git Bash (incluye Vim)
PS> "C:\Program Files\Git\usr\bin\vim.exe" archivo.txt

# OPCIÓN 3: Instalar directamente (NeoVim portable)
PS> choco install neovim  # Via Chocolatey
PS> nvim archivo.txt

# OPCIÓN 4: VS Code con extensión VIM
PS> code archivo.txt
# Instalar "Vim" extension en VS Code (mejor UX)

# Configuración en Windows:
# %APPDATA%\.vimrc o ~/.config/nvim/init.vim (via WSL)
```

- Navegar con **hjkl** (izquierda, abajo, arriba, derecha)
- Buscar, copiar, pegar, borrar, deshacer
- **Presiona ESC** desde cualquier modo para volver aquí

164.4.2 Modo Insert (Edit Mode)

Aquí **escribes** como en un editor normal.

- Todas las teclas escriben caracteres
- **Presiona ESC** para volver a Modo Normal
- Se activa con: **i**, **a**, **o**, **c**, etc

164.4.3 Modo Command (: Mode)

Aquí **ejecutas** comandos de Vim.

- **Presiona :** en Modo Normal para entrar
- Comandos: **:w** (guardar), **:q** (salir), **:set** (configurar)
- **Presiona Enter** para ejecutar, **ESC** para cancelar

164.5 Concepto 1: Navegar en Modo Normal

164.5.1 Movimiento Básico

```
Dentro de Vim en Modo Normal, NO USES FLECHAS.
Usa hjkl (es costumbre de Vi desde 1976)

    k (arriba)
h       l (derecha)
    j (abajo)

Practica: Abre un archivo con vim miarchivo.txt
Presiona h, j, k, l varias veces para sentir
```

164.5.2 Movimiento por Palabras

```
w     → Siguiente palabra (word)
b     → Anterior palabra (back)
e     → Fin de palabra (end)
^     → Inicio de línea
$     → Fin de línea
0     → Columna 0 (muy inicio)
```

164.5.3 Movimiento por Páginas

```
Ctrl+f → Página adelante (forward)
Ctrl+b → Página atrás (back)
Ctrl+u → Media página arriba (up)
Ctrl+d → Media página abajo (down)
gg     → Ir al inicio del archivo
G      → Ir al final del archivo
5G     → Ir a línea 5
```

164.5.4 Búsqueda y Posición

```
/patron    → Busca "patron" hacia adelante
?patron    → Busca "patron" hacia atrás
n          → Siguiente resultado
N          → Anterior resultado
:123       → Ir a línea 123
```

Práctica:

Abre cualquier archivo con:

- Presiona / y escribe **deb** para buscar líneas de repositorio

Listing 164.4 BASH

```
vim /etc/apt/sources.list
```

- Presiona **n** para siguiente resultado
- Presiona **G** para ir al final
- Presiona **gg** para volver al inicio

164.6 Concepto 2: Editar en Modo Insert

164.6.1 Entrar a Modo Insert

```
i    → Insert antes del cursor
I    → Insert al inicio de la línea
a    → Append después del cursor
A    → Append al final de la línea
o    → Open nueva línea abajo
O    → Open nueva línea arriba
```

Ejemplo práctico:

Listing 164.5 BASH

```
# Abre archivo
vim /tmp/test.txt

# Presiona i (enter Modo Insert)
# Escribe: Hola Abacom
# Presiona ESC (volver a Modo Normal)
# Presiona :w (guardar)
```

164.6.2 Editar Dentro de Palabras

```
c    → Change (cambiar) = borrar y entrar Insert
cw   → Cambiar palabra bajo cursor
```

Ejemplo:

Listing 164.6 BASH

```
Línea original: El servidor web está activo

# Posiciona cursor en "web"
# Presiona cw (change word)
# Tipo: nginx
# Resultado: El servidor nginx está activo
```

164.7 Concepto 3: Copiar, Cortar, Pegar

164.7.1 Copiar (Yank)

```
y     → Yank (copiar) línea completa
yw    → Yank palabra
yy    → Yank línea (igual a y)
3yy   → Yank 3 líneas
```

164.7.2 Cortar (Delete)

```
d     → Delete línea
dw    → Delete palabra
dd    → Delete línea completa
3dd   → Delete 3 líneas
x     → Delete carácter bajo cursor
```

164.7.3 Pegar

```
p     → Paste (pegar) después de posición actual
P     → Paste antes de posición actual
```

Ejemplo completo:

Listing 164.7 BASH

```
# Abre archivo
vim /tmp/ejemplo.txt

# Posiciona en línea que quieres copiar
# Presiona yy (copia línea)
# Muévete a otra línea
# Presiona p (pega)
# Resultado: línea duplicada
```

164.8 Concepto 4: Buscar y Reemplazar

164.8.1 Búsqueda Simple

```
/patron     → Busca "patron"
n           → Próximo resultado
N           → Anterior resultado
```

164.8.2 Reemplazar en Línea

```
:s/antiguo/nuevo      → Reemplaza en línea actual
:s/antiguo/nuevo/g    → Reemplaza todos en línea
```

164.8.3 Reemplazar en Archivo

```
:%s/antiguo/nuevo     → Primera ocurrencia por línea
:%s/antiguo/nuevo/g   → TODAS las ocurrencias
:%s/antiguo/nuevo/gc  → Con confirmación (c=confirm)
```

Ejemplo práctico:

Listing 164.8 BASH

```
# Archivo con varios "localhost"
vim /etc/hosts

# Reemplazar todo
:% s/localhost/127.0.0.1/g

# Resultado: todos "localhost" → "127.0.0.1"
```

164.9 Concepto 5: Deshacer y Rehacer

```
u       → Undo (deshacer)
Ctrl+r  → Redo (rehacer)
U       → Undo toda línea (todas cambios en línea actual)
```

Práctica:

Listing 164.9 BASH

```
vim /tmp/test.txt

# Haz cambios varios
# Presiona u varias veces para ver cómo deshace
# Presiona Ctrl+r para rehacer
```

164.10 Concepto 6: Guardar y Salir

164.10.1 Guardar

```
:w              → Write (guardar)
:w nuevo.txt    → Guardar como nuevo archivo
:w!             → Guardar forzadamente (sobrescribe)
```

164.10.2 Salir

```
:q              → Quit (salir) si no hay cambios
:q!             → Quit sin guardar (! = force)
:wq             → Write y Quit (guardar y salir)
:wq!            → Guardar forzadamente y salir
ZZ              → Equivalente a :wq
```

164.10.3 Combinaciones Útiles

```
:w | q          → Guardar y salir (pipe)
:e archivo.txt  → Abrir otro archivo (edit)
```

164.11 Ejemplos Prácticos Reales

164.11.1 Ejemplo 1: Editar Archivo de Configuración Nginx

Escenario: Necesitas cambiar puerto 80 a 8080 en nginx.conf

Listing 164.10 BASH

```
# Abre archivo
vim /etc/nginx/nginx.conf

# Búsqueda
/listen 80

# Presiona n para siguiente resultado hasta encontrar el correcto

# Navega al número 80
# Presiona ci" (change inside quotes) o simplemente:
# Posiciona en el 80
# Presiona c2w (cambiar 2 palabras)
# Escribe: 8080

# Resultado: listen 8080;

# Guardar
:wq
```

164.11.2 Ejemplo 2: Agregar Línea de Comentario

Escenario: Comentar línea en configuración

164.11.3 Ejemplo 3: Duplicar Configuración

Escenario: Copiar servidor virtual en nginx

164.12 Configuración Básica de Vim

164.12.1 Archivo de Configuración

Vim lee configuración de ~/**.vimrc**:

Configuración recomendada para Abacom:

Listing 164.11 BASH

```
vim /etc/ssh/sshd_config

# Buscar PermitRootLogin
/PermitRootLogin

# Presiona I (insert al inicio de línea)
# Escribe: #
# Presiona ESC
# Presiona :wq para guardar

# Resultado: #PermitRootLogin yes
```

164.13 Neovim: La Versión Moderna

164.13.1 Instalación

164.13.2 Uso Básico

Nvim es **100% compatible** con Vim:

164.13.3 Configuración de Nvim

En vez de **/.vimrc, Nvim usa /.config/nvim/init.vim**:

Contenido básico:

164.13.4 Diferencias Nvim vs Vim

Aspecto	Vim	Nvim
Velocidad	Buena	Excelente
Soporte LSP	Plugins	Nativo
Terminal integrada	Plugin	Nativa
Lua scripting	No	Sí
Disponibilidad	Estándar	Variable

164.14 Vi Clásico: Compatibilidad Total

Si **solo** tienes Vi disponible (muy raro en 2024):

Diferencias Vi clásico vs Vim:

- Sin números de línea
- Sin colores
- Búsqueda más limitada
- Pero: Todos los comandos básicos funcionan igual

Recomendación: Usa Vim si está disponible. Vi es "last resort".

Listing 164.12 BASH

```
vim /etc/nginx/sites-available/default

# Buscar el bloque server completo
/server {

# Copiar bloque (suponiendo 20 líneas)
20yy

# Ir al final del archivo
G

# Pegar
p

# Editar el nuevo bloque (cambiar puerto, nombres, etc)
# Buscar y reemplazar dentro del bloque

# Guardar
:wq
```

Listing 164.13 BASH

```
vim ~/.vimrc
```

164.15 Errores Comunes

Error 1: "¿Cómo salgo de Vim?"

Pregunta legendaria en StackOverflow

Error 2: Accidental Press ZZ

Error 3: Pegar Mal Indentado

Error 4: Búsqueda que No Funciona

164.16 Tabla de Referencia Rápida

Acción	Comando	Notas
Insertar	**i**, **a**, **o**	Entra Modo Insert
Guardar	**:w**	Write

Acción	Comando	Notas
Salir	**:q**	Quit
Guardar y salir	**:wq** o **ZZ**	Most common
Copiar línea	**yy**	Yank
Pegar	**p**	Paste after
Deshacer	**u**	Undo
Buscar	**/patron**	Forward search
Reemplazar	**:%s/viejo/nuevo/g**	All occurrences
Ir a línea	**5G** o **:5**	Go to line 5
Siguiente palabra	**w**	Word
Fin de línea	**$**	Dollar sign

164.17 Quiz: Verificar Comprensión

Pregunta 1: Modos de Vim

¿Cómo vuelves a Modo Normal desde cualquier modo?
a) Presiona **Ctrl+C**
b) Presiona **ESC** (Correcto)
c) Presiona **Enter**
d) Presiona **i**
Explicación: ESC **siempre** vuelve a Modo Normal desde Insert o Command.

Pregunta 2: Guardar y Salir

¿Cuál es la forma más rápida de guardar y salir?
a) **:wq** (correcto pero lento)
b) **:q!** (no guarda)
c) **ZZ** (Correcto - más rápido)
d) **Ctrl+S** (no funciona en Vim)
Explicación: **ZZ** es atajo de **:wq** que se teclea más rápido.

Pregunta 3: Reemplazar

¿Cómo reemplazas TODAS las ocurrencias de “old” por “new” en un archivo?
a) **:s/old/new/g** (solo línea actual)
b) **:%s/old/new/g** (Correcto - archivo completo)
c) **:s/old/new** (solo primera en línea)
d) **/old** y luego **r** (no existe)
Explicación: **%** significa “todas las líneas”, **/g** significa “global” (todas en línea).

Listing 164.14 VIM

```
" ============ APARIENCIA ============
set number              " Mostrar números de línea
set relativenumber      " Números relativos (mejor para movimiento)
set colorscheme desert  " Tema de color

" ============ COMPORTAMIENTO ============
set tabstop=4           " Tab = 4 espacios
set shiftwidth=4        " Indentación = 4 espacios
set expandtab           " Usar espacios en vez de tabs
set autoindent          " Auto-indentar nuevas líneas
set smartcase           " Búsqueda sensible a mayúsculas si las hay

" ============ BÚSQUEDA ============
set incsearch           " Buscar mientras escribes
set hlsearch            " Resaltar resultados de búsqueda

" ============ INTERFAZ ============
set ruler               " Mostrar posición actual
set statusline=%F\ -\ FileType:\ %Y
set mouse=a             " Soporte de mouse
```

164.18 Práctica: Editar Archivos Reales

Ejercicio 1: Crear y Editar Archivo

Ejercicio 2: Buscar y Reemplazar

Ejercicio 3: Comentar Línea de Configuración

164.19 LazyVim: Nvim Configurado y Potente

164.19.1 ¿Qué es LazyVim?

LazyVim es una distribución moderna de Neovim que viene **preconfigurada** con:

- Gestor de plugins (Lazy.nvim)
- LSP (Language Server Protocol) integrado
- Autocompletado de código
- Snippets y debugging
- Temas bonitos
- Configuración modular

Para Abacom: Ideal en tu máquina de desarrollo local. En servidores usa Vim clásico.

Listing 164.15 BASH

```
# Ubuntu/Debian
sudo apt install neovim

# CentOS/RHEL
sudo dnf install neovim

# Arch
sudo pacman -S neovim
```

Listing 164.16 BASH

```
nvim archivo.txt    # Igual a vim

# Todos los comandos funcionan igual
```

164.19.2 Instalación de LazyVim

Requisitos previos:

Instalar LazyVim:

① **git clone** descarga la configuración de LazyVim preconfigurada

Cuando abras nvim por primera vez:

- LazyVim descargará plugins (~200 MB)
- Compilará componentes nativos
- Espera 1-2 minutos la primera vez

164.19.3 Estructura de LazyVim

```
~/.config/nvim/
  init.lua              ← Archivo principal
  lua/
      config/
          lazy.lua      ← Configuración de plugins
          options.lua   ← Opciones generales
          keymaps.lua   ← Atajos de teclado
  lazy-lock.json        ← Versiones de plugins
```

Listing 164.17 BASH

```bash
mkdir -p ~/.config/nvim
nvim ~/.config/nvim/init.vim
```

Listing 164.18 VIM

```vim
" Heredar configuración de vim
set number
set expandtab
set tabstop=4
```

164.19.4 Instalar Plugins en LazyVim

Agregar nuevo plugin es muy simple:

164.19.4.1 Paso 1: Editar archivo de specs

164.19.4.2 Paso 2: Definir plugin con Lazy

① **nvim-treesitter/nvim-treesitter** es el identificador GitHub del plugin
② **build** ejecuta comando después de instalar
③ **ensure_installed** lista lenguajes a soportar

164.19.5 Ejemplo: Instalar Plugin para Markdown

Escenario: Quieres mejor soporte para archivos Markdown

Contenido:

① Plugin para preview de Markdown en navegador
② **cmd** = cargar solo cuando ejecutes **:MarkdownPreview**
③ **ft** = cargar solo para archivos markdown
④ **build** = instalar dependencias Node.js

Uso después:

Listing 164.19 BASH

```
vi archivo.txt      # Vi clásico (muy limitado)
```

Listing 164.20 BASH

```
# SALIDA (en este orden)
ESC         # Asegurate que estás en Modo Normal
:wq         # Write Quit (guardar y salir)
Enter       # Ejecutar comando

# Si no quieres guardar cambios:
:q!         # Quit sin guardar
```

164.19.6 Ejemplo: Instalar LSP para Python

LSP = Language Server Protocol proporciona autocompletado, definiciones, etc.

Contenido:

① **nvim-lspconfig** configura servidores de lenguaje
② **pyright** es el LSP de Python (autocompletado, type checking)
③ **gd** = shortcut para "go to definition"

Primero, instala el servidor Python:

164.19.7 Personalizar Atajos en LazyVim

Edita ~**/.config/nvim/lua/config/keymaps.lua**:

① **w** = "Espacio + w" para guardar. Espacio es la tecla líder por defecto

Después, recarga Nvim:

164.19.8 Temas en LazyVim

LazyVim viene con varios temas. Cambiar en ~**/.config/nvim/lua/config/options.lua**:

① **tokyonight** es tema por defecto. Otros están en lazy.nvim

Cambiar y recargar:

Listing 164.21 BASH

```
ZZ          # Esto guarda y sale (sin querer)
# Solución: Undo
u           # Deshacer
:e!         # Recargar archivo sin cambios
```

Listing 164.22 BASH

```
# Al pegar desde clipboard, la indentación se daña
# Solución: Desactiva indentación automática
:set paste      # Antes de pegar
:set nopaste    # Después de pegar

# O mejor: Copia directamente en Vim (yy)
```

164.19.9 Comparación: LazyVim vs Vim vs Nvim Vanilla

Aspecto	Vim	Nvim Vanilla	LazyVim
Instalación	Preinstalado	5 minutos	10 minutos
Configuración	~/.vimrc	~/.config/nvim	~/.config/nvim
Plugins	Plugins manuales	Lazy.nvim	Preconfigurado
LSP	Via plugins	Manual	Automático
Curva aprendizaje	Baja	Media	Baja (templated)
Producción			(mejor solo desarrollo)

Para Abacom:

- **Servidor SSH**: Usa **Vim**
- **Tu máquina**: Usa **LazyVim** o **Vim** según preferencia

---

164.19.10 Trucos LazyVim

Ver todos los plugins:

Actualizar plugins:

Ver configuración:

164.20 Recursos Adicionales

- **Vim Online Tutor**: https://www.openvim.com/
- **Vim Cheat Sheet**: https://vim.rtorr.com/
- **Neovim Docs**: https://neovim.io/
- **LazyVim**: https://www.lazyvim.org/
- **Lazy.nvim Plugin Manager**: https://github.com/folke/lazy.nvim
- **Interactive Vim Tutorial**: https://github.com/vim/vim-tutor

Listing 164.23 BASH

```
/patron     # Si no encuentra nada
# Presiona n para ir a siguiente resultado
N           # O anterior resultado

# Si sigue sin encontrar:
:set ignorecase     # Ignorar mayúsculas/minúsculas
```

Listing 164.24 BASH

```
# Crear archivo
vim ~/.bashrc

# Ir al final (G)
# Pegar línea: alias ll='ls -la'
# Guardar (:wq)

# Verificar
grep "alias ll" ~/.bashrc
```

164.21 Conclusión

Ahora sabes: Los 3 modos de Vim (Normal, Insert, Command) Navegar eficientemente Editar archivos rápidamente Buscar y reemplazar Guardar sin dramas

Próximo paso: Usa Vim para editar archivos de configuración en Abacom.

Consejo profesional: Practica 15 minutos diarios hasta que sea "muscle memory". Después, Vim será tu editor favorito.

Listing 164.25 BASH

```
# Crear archivo de prueba
echo -e "apache\nnode\napache" > /tmp/services.txt

# Abrirlo
vim /tmp/services.txt

# Reemplazar todos "apache" por "nginx"
:%s/apache/nginx/g

# Guardar
:wq

# Verificar
cat /tmp/services.txt
```

Listing 164.26 BASH

```
# Crear archivo
echo "enable_ssl = true" > /tmp/config.txt

# Abrirlo
vim /tmp/config.txt

# Ir al inicio de línea (0)
# Entrar Insert (I)
# Escribir: #
# ESC y guardar (:wq)

# Resultado: #enable_ssl = true
```

Listing 164.27 BASH

```
# Instalar Neovim (0.9+)
# Ubuntu/Debian
sudo apt install neovim

# CentOS/RHEL
sudo dnf install neovim

# Arch
sudo pacman -S neovim
```

Listing 164.28 BASH

```bash
# Backup tu configuración actual (si existe)
mv ~/.config/nvim ~/.config/nvim.bak

# Clonar LazyVim starter template
git clone https://github.com/LazyVim/starter ~/.config/nvim                ①

# Eliminar .git para personalizarla
rm -rf ~/.config/nvim/.git

# Abrir Neovim (LazyVim instalará plugins automáticamente)
nvim
```

Listing 164.29 BASH

```bash
nvim ~/.config/nvim/lua/plugins/example.lua
```

Listing 164.30 LUA

```lua
-- ~/.config/nvim/lua/plugins/example.lua

return {
  -- Plugin para emojis (ejemplo simple)
  {
    "nvim-treesitter/nvim-treesitter",                                      ①
    version = false,
    build = ":TSUpdate",                                                    ②
    event = { "BufReadPost", "BufNewFile" },
    config = function()
      require("nvim-treesitter.configs").setup({
        ensure_installed = { "bash", "python", "lua" },                     ③
        highlight = { enable = true },
      })
    end,
  },
}
```

Listing 164.31 BASH

```bash
# Editar configuración
nvim ~/.config/nvim/lua/plugins/markdown.lua
```

Listing 164.32 LUA

```lua
return {
  {
    "iamcco/markdown-preview.nvim",                                    ①
    cmd = { "MarkdownPreview" },                                       ②
    ft = { "markdown" },                                               ③
    build = "cd app && npm install",                                   ④
    config = function()
      vim.g.mkdp_auto_start = 0
      vim.g.mkdp_auto_close = 1
    end,
  },
}
```

Listing 164.33 BASH

```bash
nvim archivo.md

# Dentro de Nvim:
:MarkdownPreview    # Abre preview en navegador
```

Listing 164.34 BASH

```bash
nvim ~/.config/nvim/lua/plugins/python.lua
```

Listing 164.35 LUA

```lua
return {
  {
    "neovim/nvim-lspconfig",                                           ①
    config = function()
      local lspconfig = require('lspconfig')

      -- Configurar Python LSP
      lspconfig.pyright.setup({                                        ②
        on_attach = function(client, bufnr)
          -- Atajos para navegar definiciones
          local opts = { noremap=true, silent=true }
          vim.keymap.set('n', 'gd', vim.lsp.buf.definition, opts)      ③
          vim.keymap.set('n', 'gr', vim.lsp.buf.references, opts)
        end,
      })
    end,
  },
}
```

Listing 164.36 BASH

```
# Instalar Pyright (LSP de Python)
pip install pyright

# O con npm
npm install -g pyright
```

Listing 164.37 LUA

```
local keymap = vim.keymap.set

-- Atajos personalizados
keymap("n", "<Leader>w", ":w<CR>", { noremap = true }) (1)
keymap("n", "<Leader>q", ":q<CR>", { noremap = true })
keymap("n", "<Leader>x", ":wq<CR>", { noremap = true })

-- Navegar entre ventanas con Ctrl+hjkl
keymap("n", "<C-h>", "<C-w>h", { noremap = true })
keymap("n", "<C-j>", "<C-w>j", { noremap = true })
keymap("n", "<C-k>", "<C-w>k", { noremap = true })
keymap("n", "<C-l>", "<C-w>l", { noremap = true })
```

Listing 164.38 BASH

```
:source ~/.config/nvim/lua/config/keymaps.lua
```

Listing 164.39 LUA

```
-- Tema (colorscheme)
vim.cmd.colorscheme "tokyonight" (1)

-- Otros temas disponibles:
-- "tokyonight"
-- "everforest"
-- "gruvbox"
-- "catppuccin"
-- "nord"
```

Listing 164.40 BASH

```
nvim
# Dentro:
:colorscheme gruvbox  # Cambiar tema en vivo
```

Listing 164.41 BASH

```
# En Nvim:
:Lazy          # Abre manager de plugins
```

Listing 164.42 BASH

```
# En Nvim:
:Lazy update   # Actualiza todos los plugins
```

Listing 164.43 BASH

```
# Ver atajos actuales
:map

# Ver configuración cargada
:lua print(vim.inspect(vim.o))
```

165 Anexo H: SSH, SCP y Llaves - Mini-Curso Práctico

Listing 165.1 QUARTO-TITLE-BLOCK

166 Anexo H: SSH, SCP y Llaves - Mini-Curso Práctico

166.1 Introducción

En administración de servidores, **SSH** es la herramienta base: te conecta a servidores remotos, te permite ejecutar comandos, copiar archivos y automatizar tareas.

Este mini-curso está diseñado para que puedas trabajar con servidores de forma **segura y repetible**: llaves, agentes, hardening básico y troubleshooting.

Duración estimada: 60-90 minutos

166.2 Objetivos

- Conectarte por SSH de forma segura
- Crear y usar llaves (ed25519 recomendado)
- Copiar archivos con `scp` y sincronizar con `rsync`
- Aplicar hardening básico de `sshd`
- Diagnosticar errores comunes (auth, DNS, puertos)

166.3 Conceptos clave

- **Cliente SSH:** tu PC (desde donde te conectas)
- **Servidor SSH (sshd):** el servidor remoto
- **Llave pública/privada:** autenticación sin contraseña (la privada nunca se comparte)
- **Agente SSH:** mantiene llaves cargadas en memoria para no reingresar passphrase

---

Listing 166.1 BASH

```
$ ssh alumno@192.168.56.10                                                    (1)
The authenticity of host '192.168.56.10 (192.168.56.10)' can't be established.
ED25519 key fingerprint is SHA256:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
alumno@192.168.56.10's password:
```

166.4 Ejemplo 1: Conexión básica (multi-SO)

166.4.1 Linux

(1) **ssh usuario@ip** inicia sesión remota; la primera vez verás el mensaje de fingerprint del host.

166.4.2 macOS

Listing 166.2 BASH

```
$ ssh alumno@192.168.56.10                                                    (1)
alumno@192.168.56.10's password:
 # <1>
```

(1) **ssh** viene instalado en macOS y funciona igual que en Linux.

166.4.3 Windows

Listing 166.3 POWERSHELL

```
PS> ssh alumno@192.168.56.10                                                  (1)
alumno@192.168.56.10's password:
 # <1>
```

(1) **ssh** está disponible en Windows 10/11 (OpenSSH). Si no está, se instala como característica opcional.

Listing 166.4 BASH

```
$ ssh-keygen -t ed25519 -C "alumno@abacom"                                    ①
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/alumno/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:

$ ssh-copy-id -i ~/.ssh/id_ed25519.pub alumno@192.168.56.10                   ②
Number of key(s) added: 1

$ ssh alumno@192.168.56.10                                                    ③
Welcome to Ubuntu Server LTS
```

166.5 Ejemplo 2: Crear llaves y autenticación sin password

166.5.1 Linux

① **ssh-keygen -t ed25519** crea un par de llaves moderno y seguro.
② **ssh-copy-id** instala la llave pública en `~/.ssh/authorized_keys` del servidor.
③ **ssh** ya no pide password si la llave está instalada y aceptada.

166.5.2 macOS

Listing 166.5 BASH

```
$ ssh-keygen -t ed25519 -C "alumno@abacom"                                    ①

$ cat ~/.ssh/id_ed25519.pub | ssh alumno@192.168.56.10 'mkdir -p ~/.ssh && cat >> ~/.ssh/

$ ssh alumno@192.168.56.10                                                    ③
```

① **ssh-keygen** crea llaves en `~/.ssh/`.
② **cat | ssh ...** es alternativa a `ssh-copy-id` (no siempre existe en macOS).
③ **ssh** valida el login sin password.

166.5.3 Windows

① **ssh-keygen** crea llaves en `%USERPROFILE%\.ssh`.
② **type | ssh ...** copia la llave pública al servidor.
③ **ssh** prueba el acceso sin password.

Listing 166.6 POWERSHELL

```
PS> ssh-keygen -t ed25519 -C "alumno@abacom"                                    ①

PS> type $env:USERPROFILE\.ssh\id_ed25519.pub | ssh alumno@192.168.56.10 "mkdir -p ~/.ssh

PS> ssh alumno@192.168.56.10                                                    ③
```

166.6 Ejemplo 3: Copiar archivos con scp y rsync

Listing 166.7 BASH

```
$ scp ./backup.tar.gz alumno@192.168.56.10:/tmp/                                ①

$ scp alumno@192.168.56.10:/var/log/syslog ./syslog.txt                         ②

$ rsync -avz ./site/ alumno@192.168.56.10:/var/www/html/site/                   ③
```

① **scp local -> remoto** copia un archivo al servidor.
② **scp remoto -> local** trae un archivo del servidor.
③ **rsync -avz** sincroniza directorios eficientemente (reintenta, delta, conserva metadata).

166.7 Hardening básico de sshd (con cuidado)

> ⚠ ADVERTENCIA CRÍTICA
>
> Cambiar `sshd_config` sin validación puede dejarte sin acceso remoto.
> **Lo que podría salir mal:** - Cortas tu sesión y no puedes volver a entrar. - Deshabilitas el método de autenticación que estabas usando.
> **Cómo prevenirlo:** 1. Mantén una sesión SSH abierta mientras pruebas. 2. Valida configuración antes de reiniciar el servicio. 3. Aplica cambios de a uno y verifica.

① **cp** crea un backup del archivo de configuración.
② **sshd -t** valida sintaxis sin reiniciar.
③ **systemctl reload** recarga configuración con menor riesgo que reiniciar.
④ **ss** confirma que SSH sigue escuchando en el puerto.

Listing 166.8 BASH

```
$ sudo cp /etc/ssh/sshd_config /etc/ssh/sshd_config.bak ①

$ sudo sshd -t ②

$ sudo systemctl reload ssh ③

$ sudo ss -tlnp | grep ':22' ④
```

166.8 Troubleshooting rápido

Listing 166.9 BASH

```
$ ssh -vvv alumno@192.168.56.10 ①

$ nc -vz 192.168.56.10 22 ②

$ ping -c 3 192.168.56.10 ③
```

① **ssh -vvv** muestra el motivo exacto de fallo (llave, auth, kex, host key, etc.).
② **nc -vz host 22** valida conectividad al puerto 22.
③ **ping** valida conectividad IP básica (si ICMP está permitido).

166.9 Mejores prácticas

- Usa llaves `ed25519` con passphrase.
- Deshabilita password auth solo cuando ya validaste llaves.
- Mantén backups de `sshd_config` y valida con `sshd -t`.
- Usa `ssh -vvv` cuando algo falla (no adivines).

167 Anexo C: Gestores de Paquetes - Debian, CentOS, Arch

Listing 167.1 QUARTO-TITLE-BLOCK

168 Anexo C: Gestores de Paquetes - Debian, CentOS, Arch

168.1 Introducción

Cada distribución Linux tiene su **gestor de paquetes** - la herramienta que instala, actualiza y elimina software. Aunque los comandos son diferentes, la **lógica es idéntica**.

Distribuciones que encontrarás en Abacom:

- **Debian/Ubuntu** → **apt**, **apt-get** (Familia Debian)
- **CentOS/RHEL/Fedora** → **yum**, **dnf** (Familia Red Hat)
- **Arch Linux** → **pacman** (Arch)

Este anexo es **práctico y comparativo**: Aprenderás a hacer lo mismo en cada sistema.

En este anexo aprenderás:

- Diferencias entre gestores de paquetes
- Instalar, actualizar, desinstalar paquetes
- Buscar paquetes disponibles
- Gestionar repositorios
- Resolver dependencias
- Automatizar actualizaciones

Duración estimada: 90 minutos de práctica

168.2 Conceptos Clave

168.2.1 ¿Qué es un Paquete?

Un **paquete** es un archivo comprimido (**.deb**, **.rpm**, **.tar.gz**) que contiene:

```
paquete.deb/rpm
   Archivos binarios ejecutables
   Archivos de configuración
   Dependencias requeridas
   Scripts de instalación/desinstalación
   Metadatos (versión, autor, etc)
```

Analogía: Como una caja con todo lo necesario para instalar software.

168.2.2 Repositorios

Repositorios son servidores que almacenan miles de paquetes. El gestor descarga de allí:

```
Servidor Oficial Ubuntu

  nginx (web server)
  postgres (database)
  git (version control)
  ... miles de paquetes más
```

Cuando haces **apt install nginx**, el gestor:

1. Busca en repositorios configurados
2. Descarga el paquete
3. Instala automáticamente dependencias

168.3 Debian/Ubuntu: apt y apt-get

168.3.1 Diferencia: apt vs apt-get

Aspecto	apt-get	apt
Antigüedad	1997 (muy antiguo)	2014 (moderno)
Uso	Scripts/servidores	Usuario final
Interfaz	Verbosa	Resumida, colorida
Estabilidad	Garantizada	Garantizada

Para Abacom: Aprende **apt** (más moderno), pero apt-get funciona igual.

168.3.2 Actualizar Repositorios

Listing 168.1 BASH

```
# Descargar lista de paquetes disponibles
sudo apt update                                                    ①
```

① **apt update** lee repositorios y descarga lista de paquetes disponibles (no instala nada)

Salida:

```
Hit:1 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://security.ubuntu.com/ubuntu jammy-security InRelease
Reading package lists... Done
```

¿Cuándo ejecutar?

- Antes de instalar algo nuevo
- Diariamente en servidores (para seguridad)
- Después de agregar nuevos repositorios

168.3.3 Instalar Paquete

Listing 168.2 BASH

```
# Instalar un paquete
sudo apt install nginx                                             ①
```

① **apt install** descarga e instala nginx + dependencias automáticamente

Con confirmación automática:

Listing 168.3 BASH

```
sudo apt install -y nginx                                          ①
```

① **-y** (yes) responde "sí" automáticamente a confirmaciones

168.3.4 Instalar Múltiples Paquetes

Ejemplo práctico para Abacom:

① **apt update** actualiza lista primero
② **** permite continuar comando en siguiente línea

Listing 168.4 BASH

```
# Instalar varios a la vez
sudo apt install -y nginx postgresql git vim htop
```

Listing 168.5 BASH

```
#!/bin/bash
# Script para configurar servidor web

sudo apt update                                                    (1)
sudo apt install -y \ # <2>
  nginx \
  postgresql \
  python3-pip \
  curl \
  wget
```

168.3.5 Actualizar Paquetes

Listing 168.6 BASH

```
# Ver actualizaciones disponibles
apt list --upgradable                                              (1)
```

(1) **apt list** muestra información de paquetes

Actualizar todos los paquetes:

(1) **apt upgrade** actualiza paquetes a nuevas versiones

¿Diferencia entre update** y **upgrade**?**

- **update** = Descarga lista de nuevas versiones (no instala)
- **upgrade** = Instala las nuevas versiones

168.3.6 Desinstalar Paquete

(1) **apt remove** desinstala paquete pero conserva ~/.config

Desinstalar completamente (purgar):

(1) **apt purge** desinstala y elimina todos los archivos incluyendo configuración

Listing 168.7 BASH

```
sudo apt upgrade                                                    ①
```

Listing 168.8 BASH

```
# Desinstalar pero mantener archivos de config
sudo apt remove nginx                                               ①
```

168.3.7 Limpiar Espacio

① **autoremove** elimina dependencias huérfanas
② **autoclean** elimina archivos .deb descargados

168.3.8 Buscar Paquete

① **apt search** busca descripción de paquetes
② **apt show** muestra versión, tamaño, dependencias

168.3.9 Repositorios Personalizados (PPA)

A veces necesitas software no en repositorios oficiales. Usa **PPA** (Personal Package Archives):

① **add-apt-repository** agrega nuevo repositorio
② **apt update** descarga lista del nuevo repo
③ **apt install** instala desde nuevo repositorio

168.4 CentOS/RHEL: yum y dnf

168.4.1 Diferencia: yum vs dnf

Aspecto	yum	dnf
Antigüedad	2003 (viejo)	2015 (moderno)
Velocidad	Lenta	Rápida
Sintaxis	Confusa	Limpia
Defecto en Fedora	Legacy	Por defecto
Defecto en CentOS 9+	Deprecated	Por defecto

Listing 168.9 BASH

```
sudo apt purge nginx                                                    ①
```

Listing 168.10 BASH

```
# Eliminar paquetes sin usar
sudo apt autoremove                                                     ①

# Limpiar caché de descargas
sudo apt autoclean                                                      ②
```

Para Abacom: Aprende **dnf** (recomendado en CentOS 9+), pero yum sigue funcionando.

168.4.2 Actualizar Repositorios

① **yum check-update** verifica actualizaciones disponibles
② **dnf check-update** equivalente en dnf

168.4.3 Instalar Paquete

① **dnf install** descarga e instala paquete + dependencias

168.4.4 Instalar Múltiples Paquetes

168.4.5 Actualizar Paquetes

① **dnf upgrade nginx** actualiza solo nginx
② **dnf upgrade** actualiza todos los paquetes

168.4.6 Desinstalar Paquete

① **dnf remove** desinstala pero mantiene archivos config

168.4.7 Limpiar Espacio

① **dnf clean all** borra caché completo
② **dnf clean packages** borra solo paquetes .rpm descargados

Listing 168.11 BASH

```
# Buscar paquetes que contengan "nginx"
apt search nginx                                                    ①

# Ver detalles de un paquete
apt show nginx                                                      ②
```

Listing 168.12 BASH

```
# Agregar PPA (ej: para Node.js LTS)
sudo add-apt-repository ppa:chris-lea/node.js-lts                   ①
sudo apt update                                                     ②
sudo apt install nodejs                                             ③
```

168.4.8 Buscar Paquete

① **dnf search** busca descripción
② **dnf info** muestra versión, tamaño, repositorio

168.4.9 Gestionar Repositorios

① **dnf repolist** lista repositorios activos
② **dnf repolist all** incluye deshabilitados (useful para ver opciones)
③ **dnf config-manager** habilita/deshabilita repositorios

168.5 Arch Linux: pacman

168.5.1 Filosofía de Arch

Arch tiene filosofía diferente:

- **Minimalista**: Solo lo esencial
- **DIY**: Usuario decide configuración
- **Rolling release**: Actualizaciones continuas (no versiones fijas)
- **Bleeding edge**: Siempre software más reciente

168.5.2 Actualizar Sistema Completo

① **-S** = sincronizar (descargar), **-y** = actualizar base, **-u** = actualizar paquetes

En Arch es CRÍTICO hacer esto regularmente - Es distribución rolling-release.

Listing 168.13 BASH

```
# En CentOS con yum
sudo yum check-update                                                    ①

# En CentOS con dnf (recomendado)
sudo dnf check-update                                                    ②
```

Listing 168.14 BASH

```
# Instalar nginx
sudo dnf install -y nginx                                                ①

# Antiguo (yum)
sudo yum install -y nginx
```

168.5.3 Instalar Paquete

① **-S** (Sync) descarga e instala paquete

168.5.4 Desinstalar Paquete

① **-R** (Remove) desinstala paquete
② **-Rs** también elimina dependencias no usadas
③ **-Rn** elimina config (purge)

168.5.5 Limpiar Espacio

① **-Sc** borra cache de instalados (seguro)
② **-Scc** borra TODOS los .pkg descargados (peligroso)

168.5.6 Buscar Paquete

① **-Ss** (Search Sync) busca descripción
② **-Si** (info Sync) muestra detalles de repositorio
③ **-Ql** (Query List) lista archivos instalados

168.5.7 Ver Paquetes Instalados

① **-Q** (Query) lista paquetes instalados
② **-Qi** muestra info completa incluyendo tamaño

Listing 168.15 BASH

```
# Instalar varios a la vez
sudo dnf install -y nginx postgresql python3-pip curl wget
```

Listing 168.16 BASH

```
# Actualizar un paquete específico
sudo dnf upgrade nginx                                                    ①

# Actualizar TODO el sistema
sudo dnf upgrade                                                          ②
```

168.5.8 AUR: Arch User Repository

Arch tiene repositorio comunitario **AUR** para software menos común:

168.6 Comparativa Rápida: Los Mismos Comandos

168.6.1 Escenario 0: ¿Qué Package Manager Tengo?

168.6.1.1 Linux (Detectar automáticamente)

168.6.1.2 macOS (Homebrew)

168.6.1.3 Windows (Chocolatey o Windows Package Manager)

168.6.2 Escenario 1: Instalar Nginx

168.6.2.1 Linux (Debian/Ubuntu)

168.6.2.2 Linux (CentOS/RHEL)

168.6.2.3 Linux (Arch)

Listing 168.17 BASH

```
# Desinstalar
sudo dnf remove nginx                                                    ①
```

Listing 168.18 BASH

```
# Eliminar cache
sudo dnf clean all                                                       ①

# Ver estadísticas
sudo dnf clean packages                                                  ②
```

168.6.2.4 macOS

168.6.2.5 Windows

Resumen: | SO | Comando | Gestor | Servicio | |—-|———|———|———-| | Linux | `apt install nginx` | apt | systemctl | | macOS | `brew install nginx` | brew | brew services | | Windows | `choco install nginx` | Chocolatey | Services |

168.6.3 Escenario 2: Instalar Servidor Web Completo

168.6.3.1 Linux (Debian/Ubuntu)

168.6.3.2 macOS

168.6.3.3 Windows

168.6.4 Escenario 3: Automatizar Actualizaciones Diarias

Script Inteligente (Multi-SO):

① **apt** = Debian/Ubuntu
② **dnf** = CentOS/RHEL 8+
③ **yum** = CentOS/RHEL 7
④ **pacman** = Arch Linux

Listing 168.19 BASH

```
# Buscar paquete
sudo dnf search nginx                                                    ①

# Ver detalles
sudo dnf info nginx                                                      ②
```

Listing 168.20 BASH

```
# Ver repositorios habilitados
sudo dnf repolist                                                        ①

# Ver todos incluyendo deshabilitados
sudo dnf repolist all                                                    ②

# Habilitar repositorio
sudo dnf config-manager --set-enabled powertools                         ③

# Agregar repositorio externo
sudo dnf install -y https://repo-url/package.rpm
```

⑤ **macOS** = Detectado por OSTYPE

168.6.5 Escenario 3 (Anterior): Automatizar Actualizaciones Diarias

Script Universal (versión anterior):

① **apt update && apt upgrade** para Debian
② **dnf upgrade** para CentOS moderno
③ **yum update** para CentOS 7
④ **pacman -Syu** para Arch

Guardar como: **/usr/local/bin/actualizar-sistema.sh**

Hacer ejecutable: **chmod +x /usr/local/bin/actualizar-sistema.sh**

Usar en cron para actualizaciones automáticas:

Listing 168.21 BASH

```
# Actualizar TODOS los paquetes (¡importante en Arch!)
sudo pacman -Syu                                                    ①
```

Listing 168.22 BASH

```
# Instalar nginx
sudo pacman -S nginx                                                ①

# Instalar múltiples
sudo pacman -S nginx postgresql git vim htop
```

168.7 Tabla Comparativa: Comandos Equivalentes

Tarea	Debian/Ubuntu	CentOS/RHEL	Arch
Actualizar lista	**apt update**	**dnf check-update**	**pacman -Sy**
Instalar	**apt install pkg**	**dnf install pkg**	**pacman -S pkg**
Actualizar todo	**apt upgrade**	**dnf upgrade**	**pacman -Syu**
Desinstalar	**apt remove pkg**	**dnf remove pkg**	**pacman -R pkg**
Buscar	**apt search txt**	**dnf search txt**	**pacman -Ss txt**
Info paquete	**apt show pkg**	**dnf info pkg**	**pacman -Si pkg**
Limpiar	**apt autoremove**	**dnf clean all**	**pacman -Sc**

168.8 Gestionar Dependencias

Todos los gestores resuelven **dependencias automáticamente**:

Ventaja: No necesitas pensar en dependencias. El gestor se encarga.

168.9 Errores Comunes

Error 1: “Package not found”

Error 2: “Permission denied”

Error 3: “Dependency conflicts”

Listing 168.23 BASH

```
# Desinstalar pero mantener config
sudo pacman -R nginx                                                    ①

# Desinstalar y remover dependencias huérfanas
sudo pacman -Rs nginx                                                   ②

# Desinstalar todo incluyendo config
sudo pacman -Rn nginx                                                   ③
```

Listing 168.24 BASH

```
# Eliminar paquetes descargados
sudo pacman -Sc                                                         ①

# Eliminar TODO incluyendo instalados (¡cuidado!)
sudo pacman -Scc                                                        ②
```

168.10 Quiz: Verificar Comprensión

Pregunta 1: ¿Cuándo usar apt vs apt-get?

a) apt es más antiguo
b) apt-get es mejor
c) apt es más moderno y para usuario final (Correcto)
d) Son exactamente iguales
Explicación: apt es versión mejorada de apt-get lanzada en 2014. Ambos funcionan, pero apt tiene interfaz mejor.

Pregunta 2: ¿Qué hace “apt update”?

a) Actualiza e instala paquetes nuevos
b) Solo descarga lista de paquetes disponibles (Correcto)
c) Actualiza el sistema operativo
d) Limpia el caché
Explicación: “apt update” solo descarga metadatos. Es primer paso antes de instalar. “apt upgrade” es el que instala.

Pregunta 3: En Arch, ¿cuál es el comando para actualizar TODO?

a) **pacman -S**
b) **pacman -Syu** (Correcto)
c) **pacman -R**

Listing 168.25 BASH

```
# Buscar en repositorios
sudo pacman -Ss nginx                                                    ①

# Ver info detallada
sudo pacman -Si nginx                                                    ②

# Listar archivos en paquete instalado
sudo pacman -Ql nginx                                                    ③
```

Listing 168.26 BASH

```
# Listar paquetes instalados
sudo pacman -Q                                                           ①

# Ver espacio usado por paquete
sudo pacman -Qi nginx                                                    ②
```

d) pacman -Q
Explicación: En Arch es **crítico** hacer **pacman -Syu** regularmente porque es rolling-release.

168.11 Práctica: Instalar Software Real

Ejercicio 1: Instalar Stack LEMP en Debian/Ubuntu

Ejercicio 2: Instalar Stack LAMP en CentOS

Ejercicio 3: Script Seguro para Múltiples Distribuciones

168.12 Recursos Adicionales

- **Ubuntu Packaging Guide**: https://packaging.ubuntu.com/
- **CentOS DNF Documentation**: https://dnf.readthedocs.io/
- **Arch Wiki**: https://wiki.archlinux.org/
- **Linux Reproducible Builds**: https://reproducible-builds.org/
- **Package Management Comparison**: https://wiki.archlinux.org/title/Pacman/Rosetta

Listing 168.27 BASH

```
# NO es soporte oficial de Arch
# Usar solo paquetes de confianza

# Instalar helper AUR (ej: yay)
sudo pacman -S yay

# Buscar en AUR
yay -Ss nombre-paquete

# Instalar desde AUR
yay -S nombre-paquete
```

Listing 168.28 BASH

```
# Detectar qué package manager tienes
$ if [ -f /etc/debian_version ]; then
    echo "Debian/Ubuntu - usa apt"
  elif [ -f /etc/redhat-release ]; then
    echo "CentOS/RHEL - usa yum/dnf"
  elif [ -f /etc/arch-release ]; then
    echo "Arch Linux - usa pacman"
  fi
```

168.13 Conclusión

Ahora sabes: Instalar, actualizar, desinstalar paquetes en Debian, CentOS y Arch Buscar y gestionar repositorios Resolver dependencias automáticamente Automatizar actualizaciones Comandos equivalentes en cada distribución

Para Abacom: Elige una distribución (recomendado Ubuntu 22.04 LTS) y domina su gestor.

Próximo paso: Usa estos comandos en scripts de configuración de servidores.

Listing 168.29 BASH

```
# macOS NO tiene apt, yum ni pacman
# Usa Homebrew como package manager

$ which brew
/usr/local/bin/brew

# Si no está instalado:
$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/inst

# Homebrew es equivalente a apt en Linux
$ brew --version
Homebrew 4.0.0
```

Listing 168.30 POWERSHELL

```
# Windows tiene múltiples opciones:

# Opción 1: Windows Package Manager (wpkg - nativo)
PS> winget --version
v1.6.x

# Opción 2: Chocolatey (terceros, más popular)
PS> choco --version
Chocolatey v2.0.0

# Si no está instalado (Chocolatey):
PS> Set-ExecutionPolicy Bypass -Scope Process -Force; `
    [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]
    iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.o
```

Listing 168.31 BASH

```
# Debian/Ubuntu
$ sudo apt update
$ sudo apt install -y nginx
$ sudo systemctl enable nginx
$ sudo systemctl start nginx

# Verificar
$ systemctl status nginx
 nginx.service - A high performance web server
     Loaded: loaded (/lib/systemd/unit/nginx.service; enabled)
     Active: active (running)
```

Listing 168.32 BASH

```
# CentOS/RHEL
$ sudo dnf install -y nginx
$ sudo systemctl enable nginx
$ sudo systemctl start nginx

# Verificar
$ systemctl status nginx
  nginx.service - The NGINX HTTP and reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled)
     Active: active (running)
```

Listing 168.33 BASH

```
# Arch
$ sudo pacman -Syu  # Actualizar repo
$ sudo pacman -S nginx
$ sudo systemctl enable nginx
$ sudo systemctl start nginx
```

Listing 168.34 BASH

```
# macOS (via Homebrew)
$ brew install nginx
$ brew services start nginx
$ nginx -v
nginx version: nginx/1.25.3

# Verificar
$ ps aux | grep nginx
# Nginx está corriendo

# Nota: En macOS no usamos systemctl
# En su lugar: brew services
```

Listing 168.35 POWERSHELL

```
# Windows (via Chocolatey)
PS> choco install nginx -y
PS> # Nginx se instala en C:\tools\nginx\

# Iniciar servicio (o usar GUI)
PS> Start-Service -Name nginx

# Verificar
PS> Get-Service nginx
Status   Name
------   ----
Running  nginx

# O acceder a http://localhost
PS> Invoke-WebRequest -Uri "http://localhost"
StatusCode        : 200
```

Listing 168.36 BASH

```
# Instalar nginx + base de datos + Python
$ sudo apt update && sudo apt install -y nginx postgresql python3-pip
$ sudo apt upgrade -y
```

Listing 168.37 BASH

```
# macOS equivalente
$ brew install nginx postgresql python@3.11
$ brew services start nginx
$ brew services start postgresql

# Verificar versiones
$ nginx -v && psql --version && python3 --version
```

Listing 168.38 POWERSHELL

```
# Windows equivalente
PS> choco install nginx postgresql python -y

# Iniciar servicios
PS> Start-Service -Name nginx
PS> Start-Service -Name postgresql
```

Listing 168.39 BASH

```
#!/bin/bash
# Script que detecta el SO y actualiza automáticamente

if command -v apt &> /dev/null; then
  # Sistema Debian/Ubuntu                                          (1)
  sudo apt update && sudo apt upgrade -y
  echo "  Actualización Debian completada"

elif command -v dnf &> /dev/null; then
  # Sistema CentOS/RHEL 8+
  sudo dnf upgrade -y                                              (2)
  echo "  Actualización CentOS completada"

elif command -v yum &> /dev/null; then
  # Sistema CentOS/RHEL 7
  sudo yum update -y                                               (3)
  echo "  Actualización CentOS 7 completada"

elif command -v pacman &> /dev/null; then
  # Sistema Arch                                                   (4)
  sudo pacman -Syu
  echo "  Actualización Arch completada"

elif [[ "$OSTYPE" == "darwin"* ]]; then
  # macOS                                                          (5)
  brew update && brew upgrade
  echo "  Actualización macOS completada"

else
  echo "  Package manager no reconocido"
  exit 1
fi
```

Listing 168.40 BASH

```
#!/bin/bash
# Script que funciona en cualquier distribución

if command -v apt &> /dev/null; then
  # Sistema Debian/Ubuntu
  sudo apt update && sudo apt upgrade -y                    ①
  echo "  Actualización Debian completada"

elif command -v dnf &> /dev/null; then
  # Sistema CentOS/RHEL 8+
  sudo dnf upgrade -y                                       ②
  echo "  Actualización CentOS completada"

elif command -v yum &> /dev/null; then
  # Sistema CentOS/RHEL 7
  sudo yum update -y                                        ③
  echo "  Actualización CentOS 7 completada"

elif command -v pacman &> /dev/null; then
  # Sistema Arch
  sudo pacman -Syu --noconfirm                              ④
  echo "  Actualización Arch completada"
fi
```

Listing 168.41 BASH

```
0 2 * * *  /usr/local/bin/actualizar-sistema.sh  # Diariamente a las 2 AM
```

Listing 168.42 BASH

```
# Ejemplo: Instalar PHP con Apache

# Debian
sudo apt install php-apache  # Instala:
#   apache2
#   php
#   libapache2-mod-php
#   ... otras dependencias

# CentOS
sudo dnf install php  # Instala:
#   httpd
#   php
#   php-common
#   ... otras dependencias

# Arch
sudo pacman -S php apache  # Instala:
#   apache
#   php
#   ... otras dependencias
```

Listing 168.43 BASH

```
# Causa: Repositorios no actualizados
sudo apt update  # Debian
sudo dnf check-update  # CentOS
sudo pacman -Sy  # Arch

# Luego reintentar instalación
```

Listing 168.44 BASH

```
# Causa: Falta sudo
apt install nginx  #  No tienes permisos

# Solución:
sudo apt install nginx  #  Con sudo
```

Listing 168.45 BASH

```
# Causa: Versión incompatible
# Solución: Usar gestor de dependencias

# Debian
sudo apt autoremove  # Limpia conflictos

# CentOS
sudo dnf autoremove  # Limpia conflictos

# Arch
sudo pacman -Sc  # Limpia cache
```

Listing 168.46 BASH

```
# L = Linux
# E = Nginx (Engine X)
# M = MySQL/MariaDB
# P = PHP

# Ubuntu
sudo apt update
sudo apt install -y nginx mariadb-server php-fpm php-mysql
sudo systemctl start nginx
sudo systemctl start mariadb
```

Listing 168.47 BASH

```
# L = Linux
# A = Apache (httpd)
# M = MySQL/MariaDB
# P = PHP

# CentOS
sudo dnf install -y httpd mariadb-server php
sudo systemctl start httpd
sudo systemctl start mariadb
```

Listing 168.48 BASH

```bash
#!/bin/bash
# Script que instala nginx en cualquier distribución

set -e  # Salir si hay error

echo "Detectando distribución..."

if [ -f /etc/os-release ]; then
  . /etc/os-release
  OS=$ID
fi

echo "Sistema detectado: $OS"

case "$OS" in
  ubuntu|debian)
    echo "Instalando en Debian/Ubuntu..."
    sudo apt update
    sudo apt install -y nginx
    ;;
  centos|rhel)
    echo "Instalando en CentOS/RHEL..."
    sudo dnf install -y nginx
    ;;
  arch)
    echo "Instalando en Arch..."
    sudo pacman -Syu
    sudo pacman -S --noconfirm nginx
    ;;
  *)
    echo "Distribución no soportada: $OS"
    exit 1
    ;;
esac

echo "  Nginx instalado exitosamente"
sudo systemctl status nginx
```

169 Anexo D: Herramientas de Monitoreo del Sistema - top, htop, etc

Listing 169.1 QUARTO-TITLE-BLOCK

170 Anexo D: Herramientas de Monitoreo del Sistema - top, htop, etc

170.1 Introducción

En administración de servidores Linux, **monitorear el sistema** es fundamental. Necesitas saber:

- ¿Cuánta CPU estoy usando?
- ¿Qué procesos consumen memoria?
- ¿Está saturado el disco?
- ¿Quién se conecta al servidor?

Este anexo enseña herramientas para **observar qué está sucediendo** en tiempo real en tu servidor.

En este anexo aprenderás:

- **top** - Monitor de procesos clásico (siempre disponible)
- **htop** - Versión mejorada de top (más amigable)
- **free** - Información de RAM
- **df** - Espacio en disco
- **du** - Uso de disco por carpeta
- **iostat** - Lectura/escritura de disco
- **netstat/ss** - Conexiones de red
- **vmstat** - Estadísticas virtuales de memoria
- Monitoreo remoto y alertas

Duración estimada: 60 minutos de práctica

170.2 ¿Por Qué Estas Herramientas?

Escenario real en Abacom:

```
3:47 PM - Usuario reporta: "El servidor está lento"

Tu respuesta:
$ top        # Ver procesos que consumen CPU
$ free -h    # Verificar RAM disponible
```

```
$ df -h      # Verificar espacio en disco
$ netstat -an | wc -l  # Ver conexiones activas

Conclusión: "Hay 10,000 conexiones TCP activas
            consumiendo toda la memoria"

Solución: Optimizar aplicación o agregar más RAM
```

Sin estas herramientas, estarías a ciegas.

170.3 Monitoreo en Diferentes SOs

170.3.1 Linux (Herramientas Nativas)

170.3.2 macOS (Diferentes comandos)

170.3.3 Windows (PowerShell / Task Manager)

Comparación de comandos: | Tarea | Linux | macOS | Windows | |——|——-|——-|———| | Ver procesos | `top` / `htop` | `top -l 1` (diferente) | `Get-Process` | | Ver RAM | `free -h` | `vm_stat` | `Get-WmiObject Win32_ComputerSystem` | | Ver disco | `df -h` | `df -h` | `Get-PSDrive` | | Ver conexiones | `netstat -an` / `ss` | `netstat -an` / `lsof -i` | `Get-NetTCPConnection` |

Recomendación para Abacom:

- Linux: Usar herramientas nativas (top, free, df, netstat)
- macOS: Instalar Homebrew y usar `htop`, `lsof`
- Windows: Usar PowerShell cmdlets o Task Manager GUI

170.4 Concepto 1: TOP - Monitor de Procesos

170.4.1 Ejecutar TOP

① **top** abre monitor interactivo de procesos

Pantalla inicial:

Listing 170.1 BASH

```
# Linux tiene todas estas herramientas nativas
$ which top
/usr/bin/top

$ which htop
/usr/bin/htop

$ which free
/usr/bin/free

$ which df
/bin/df

# Monitor de procesos
$ top
$ htop  # (si está instalado)

# RAM disponible
$ free -h
              total        used        free
Mem:           15Gi       5.2Gi       8.2Gi

# Espacio en disco
$ df -h
Filesystem     Size  Used Avail Use%
/dev/sda1      100G   35G   65G  35%

# Conexiones de red
$ ss -an | wc -l
2345
```

```
top - 15:47:23 up 45 days, 2:34,  3 users,  load average: 0.89, 0.76, 0.68
Tasks: 287 total,   2 running, 285 sleeping,   0 stopped,   0 zombie
%Cpu(s):  8.2 us,  3.4 sy,  0.0 ni, 88.1 id,  0.3 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem : 15967.5 total,  8234.6 free,  5421.2 used,  2311.7 buff/cache
MiB Swap:  4096.0 total,  4096.0 free,     0.0 used.  9854.3 avail Mem

   PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 12847 nginx     20   0  145620  12344   9876 S   8.2   0.1   2:45.32 nginx
  8934 mysql     20   0  876543 234567  45678 S   5.6  12.3  45:23.12 mysqld
```

170.4.2 Entender la Salida de TOP

Primera línea:

Listing 170.2 BASH

```
# macOS NO tiene: top, htop, free (igual que Linux)
# Top existe pero funciona diferente
# Free no existe - usar vm_stat en su lugar

# Monitor de procesos (macOS top)
$ top -l 1  # Solo una iteración
$ top       # Interactive (diferente a Linux)

# Memoria (no existe 'free', usar vm_stat)
$ vm_stat | grep "Pages free"
Pages free:                        2048234  # En páginas de 4KB

# Alternativa: Con Homebrew
$ brew install htop
$ htop  # Ahora funciona igual a Linux

# Espacio en disco
$ df -h
Filesystem     Size  Used Avail Use%
/dev/disk0s2   250G  100G  150G  40%

# Conexiones de red (similar a Linux)
$ netstat -an | wc -l
1234

# Mejor: usar lsof para ver procesos y puertos
$ lsof -i :8080
# Ver qué proceso usa puerto 8080
```

```
top - 15:47:23              ← Hora actual
up 45 days, 2:34            ← Tiempo encendido el servidor
3 users                     ← Usuarios conectados
load average: 0.89, 0.76, 0.68  ← Carga CPU (1min, 5min, 15min)
```

"Load average" es crítico:

- **1 core = load 1.0** es 100% ocupado
- **4 cores = load 4.0** es 100% ocupado
- **load 0.5** = 50% del CPU disponible
- Si ves **load 8.0** en máquina 4-core = **SATURADO**

Segunda línea - Procesos:

```
Tasks: 287 total    ← Total de procesos
2 running           ← Ejecutándose ahora
```

Listing 170.3 POWERSHELL

```
# Windows no tiene top, free, df
# Usa Get-Process, Get-PSDrive, etc.

# Monitor de procesos
PS> Get-Process | Sort-Object CPU -Descending | Select-Object Name, CPU, Memory | head -1

Name              CPU        Memory
----              ---        ------
chrome         450.25  2842894336
firefox        125.50   854321120

# RAM disponible
PS> $RAM = Get-WmiObject Win32_ComputerSystem
PS> Write-Host "$($RAM.TotalPhysicalMemory / 1GB) GB total RAM"
16 GB total RAM

# Espacio en disco
PS> Get-PSDrive | Where-Object {$_.Provider -match 'FileSystem'}

Name Used (GB) Free (GB) Provider
---- --------- --------- --------
C        200       50       FileSystem
D        500       100      FileSystem

# Conexiones de red
PS> (Get-NetTCPConnection).Count
2345

# O abrir Task Manager gráficamente
PS> taskmgr  # GUI interactiva
```

```
285 sleeping        ← Esperando (normal)
0 stopped           ← Detenidos
0 zombie            ← Procesos fantasma (problema)
```

Tercera línea - CPU:

```
%Cpu(s): 8.2 us         ← User Space (código de aplicación)
         3.4 sy         ← System (kernel Linux)
         0.0 ni         ← Nice (prioridad baja)
         88.1 id        ← Idle (desocupado) ← Ideal es ALTO
         0.3 wa         ← Wait I/O (esperando disco)
         0.0 hi         ← Hardware interrupts
         0.0 si         ← Software interrupts
         0.0 st         ← Stolen (virtualization)
```

Listing 170.4 BASH

```
top                                                                    ①
```

Memoria:

```
MiB Mem: 15967.5 total       ← RAM total
         8234.6 free         ← Disponible (libre)
         5421.2 used         ← En uso
         2311.7 buff/cache   ← Buffer/Cache (puede liberarse)
```

Columnas de Procesos:

```
PID      ← ID del proceso
USER     ← Usuario propietario
%CPU     ← Porcentaje de CPU usando
%MEM     ← Porcentaje de RAM usando
VIRT     ← Memoria virtual (puede no existir)
RES      ← Memoria física (RAM real)
COMMAND  ← Nombre del proceso
```

170.4.3 Comandos Interactivos en TOP

Dentro de TOP, presiona:

```
ESPACIO   → Actualizar pantalla
P         → Ordenar por %CPU (procesos más pesados)
M         → Ordenar por %MEM (más memoria)
T         → Ordenar por TIME (más tiempo corriendo)
R         → Invertir orden (Mayor a menor)

q         → Salir de top
k         → Matar proceso (kill)
r         → Cambiar prioridad (renice)
f         → Agregar/quitar columnas
```

170.4.4 Ejemplo: Encontrar Proceso que Consume CPU

Listing 170.5 BASH

```
# Ejecutar top
top

# Presionar P para ordenar por CPU
# Ver el primero en lista

# Presionar k para matar (si necesitas)
# Ingresar PID del proceso
```

Listing 170.6 BASH

```
# Ver procesos una sola vez (sin interactivo)
top -b -n 1                                                    ①
```

170.4.5 TOP sin Modo Interactivo

① **-b** (batch mode) no interactivo, **-n 1** una sola actualización

Útil en scripts:

Listing 170.7 BASH

```
#!/bin/bash
# Monitorear si PHP consume >80% CPU

top -b -n 1 | grep php | awk '{print $9}' | while read CPU
do
  if (( $(echo "$CPU > 80" | bc -l) ))
  then
    echo "ALERTA: PHP consume ${CPU}% CPU"
    # Enviar alerta por email, etc
  fi
done
```

170.5 Concepto 2: HTOP - TOP Mejorado

170.5.1 Instalar HTOP

Listing 170.8 BASH

```
# Ubuntu/Debian
sudo apt install -y htop

# CentOS/RHEL
sudo dnf install -y htop

# Arch
sudo pacman -S htop
```

Listing 170.9 BASH

```
htop                                                                    ①
```

170.5.2 Ejecutar HTOP

① **htop** es reemplazo directo de top (mismos comandos)

Ventajas sobre TOP:

- Colores (más legible)
- Barras de progreso (CPU, Memoria)
- Scroll horizontal/vertical
- Buscar procesos (f para filtrar)
- Ver árbol de procesos (t para tree)
- Interfaz más amigable

Pantalla de HTOP:

```
  CPU[        ] 45%     Tasks: 142, 2 thr; 1 running
  Mem[        ] 68%     Load average: 0.89 0.76 0.68
  Swp[        ] 0%      Uptime: 45d 2h 34m

   PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+ Command
 12847 nginx     20   0  145M  12M  9.8M S  8.2  0.1  2:45 nginx: worker process
  8934 mysql     20   0  876M 234M  45M S  5.6 12.3 45:23 /usr/sbin/mysqld
  2341 diego     20   0  234M  89M  23M S  2.1  0.6  1:23 firefox
```

170.5.3 Atajos en HTOP

```
P         → Ordenar por %CPU
M         → Ordenar por %MEM
T         → Ordenar por TIME
H         → Mostrar/ocultar procesos de threads
K         → Mostrar/ocultar kernel threads
```

```
F         → Filtrar procesos (ej: "nginx")
t         → Ver árbol de procesos (parent-child)
i         → Mostrar solo procesos con I/O
s         → Cambiar prioridad (strace)
q         → Salir
```

170.6 Concepto 3: FREE - Información de RAM

170.6.1 Ver Memoria en Formato Humano

Listing 170.10 BASH

```
free -h                                                                    ①
```

① **-h** (human-readable) muestra en GB/MB en vez de bytes

Salida:

```
               total        used        free      shared  buff/cache   available
Mem:            15Gi       5.3Gi       8.0Gi      234Mi       2.3Gi       9.4Gi
Swap:          4.0Gi          0B       4.0Gi
```

170.6.2 Interpretar la Salida

```
Mem: 15Gi          ← RAM total instalada
     5.3Gi usado   ← En uso ahora
     8.0Gi libre   ← No usado (desocupado)
     234Mi shared  ← Memoria compartida (múltiples procesos)
     2.3Gi cache   ← Datos cacheados (puede liberarse)
     9.4Gi avail   ← Disponible para aplicaciones NEW
```

Antes Linux 3.14:

```
Total = Usado + Libre  ← Esto era INCORRECTO
```

Linux 3.14+ (disponible):

```
Total = Usado + (Cache - usable) + Libre
Avail = Libre + Cache  ← Lo que REALMENTE tienes disponible
```

Listing 170.11 BASH

```
# Mostrar cada 2 segundos
free -h -s 2                                                    ①
```

170.6.3 Monitorear Memoria en Intervalos

① **-s 2** (seconds) actualiza cada 2 segundos

Salida:

```
               total        used        free      shared  buff/cache   available
Mem:            15Gi       5.3Gi       8.0Gi      234Mi       2.3Gi       9.4Gi
Swap:          4.0Gi          0B       4.0Gi

               total        used        free      shared  buff/cache   available
Mem:            15Gi       5.3Gi       8.0Gi      234Mi       2.3Gi       9.4Gi
Swap:          4.0Gi          0B       4.0Gi
```

Presiona **Ctrl+C** para salir.

170.7 Concepto 4: DF - Espacio en Disco

170.7.1 Ver Espacio en Disco

Listing 170.12 BASH

```
df -h                                                           ①
```

① **-h** (human-readable) en GB/MB

Salida:

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1        50G   32G   15G  68% /
/dev/sda2       450G   120G  330G  27% /home
/dev/sdb1       500G   480G   20G  96%  /mnt/backup
tmpfs            15G     0   15G   0% /dev/shm
```

Listing 170.13 BASH

```
# Mostrar en porcentaje de uso
df -h | awk '{print $5, $6}' | sort -rn                    ①
```

170.7.2 Identificar Discos Llenos

① **awk** extrae columnas, **sort -rn** ordena descendente

Salida:

```
96% /mnt/backup     ← ¡ALERTA! Disco lleno
68% /
27% /home
0% /dev/shm
```

170.7.3 Alerta si Disco >80% Lleno

Listing 170.14 BASH

```
#!/bin/bash
# Script para monitorear discos

df -h | tail -n +2 | while read LINE
do
  USAR=$(echo $LINE | awk '{print $5}' | sed 's/%//')          ①
  MOUNT=$(echo $LINE | awk '{print $6}')

  if [ "$USAR" -gt 80 ]
  then
    echo "   ALERTA: $MOUNT está al ${USAR}% de capacidad"
  fi
done
```

① **sed 's/%//'** elimina el signo % para comparar como número

---

170.8 Concepto 5: DU - Uso de Disco por Carpeta

170.8.1 Encontrar Qué Ocupa Espacio

① **-s** (summarize) solo total, **-h** (human) en GB/MB

Listing 170.15 BASH

```
# ¿Quién está ocupando espacio en /home?
du -sh /home/*                                                    ①
```

Salida:

```
45G   /home/diego
23G   /home/carlos
8.2G  /home/ana
1.5G  /home/backup
```

170.8.2 Encontrar Top 10 Carpetas Más Grandes

Listing 170.16 BASH

```
du -sh /* | sort -rh | head -10                                   ①
```

① **sort -rh** ordena numérico inverso

Salida:

```
234G  /var
120G  /home
45G   /opt
23G   /usr
8.2G  /root
4.5G  /tmp
2.3G  /etc
1.2G  /srv
890M  /lib
456M  /bin
```

170.8.3 Profundizar en Carpeta

Listing 170.17 BASH

```
# ¿Qué hay en /var que ocupa 234G?
du -sh /var/* | sort -rh | head -5

# ¿Qué hay en /var/log?
du -sh /var/log/* | sort -rh | head -5
```

Encontrar carpeta problemática:

Listing 170.18 BASH

```
du -sh /var/log/* | grep "G$"                                    ①
```

① **grep "G$"** solo líneas con G (gigabytes)

170.9 Concepto 6: IOSTAT - Lectura/Escritura de Disco

170.9.1 Instalar sysstat

Listing 170.19 BASH

```
# Ubuntu/Debian
sudo apt install -y sysstat

# CentOS/RHEL
sudo dnf install -y sysstat

# Arch
sudo pacman -S sysstat
```

170.9.2 Ver Actividad de Disco

Listing 170.20 BASH

```
iostat -x 1                                                      ①
```

① **-x** (extended) más detalles, **1** actualiza cada 1 segundo

Salida (primera línea muestra promedios):

```
Device            r/s     w/s     rMB/s   wMB/s r_await w_await %util
sda             125.4    45.2      8.3     2.1    12.5    23.4   45.2%
sdb              45.2    10.1      3.2     0.8     8.2    15.1   12.3%
sdc               2.1     1.0      0.1     0.1    45.2    78.9    2.1%

Device            r/s     w/s     rMB/s   wMB/s r_await w_await %util
sda             234.1    89.3     15.4     5.2    15.2    34.5   78.9%
sdb               0.0     0.0      0.0     0.0     0.0     0.0    0.0%
sdc               0.0     0.0      0.0     0.0     0.0     0.0    0.0%
```

170.9.3 Interpretar iostat

```
r/s      ← Lecturas por segundo
w/s      ← Escrituras por segundo
rMB/s    ← Megabytes leídos por segundo
wMB/s    ← Megabytes escritos por segundo
r_await  ← Latencia de lectura (ms)
w_await  ← Latencia de escritura (ms)
%util    ← Porcentaje de utilización
```

Análisis:

- **%util > 80%** = Disco saturado
- **r_await > 50ms** = Lecturas lentas
- **w_await > 100ms** = Escrituras muy lentas

170.10 Concepto 7: NETSTAT/SS - Conexiones de Red

170.10.1 Ver Conexiones Activas

Listing 170.21 BASH

```bash
# Método moderno (recomendado)
ss -tulpn                                                    ①

# Método antiguo (aún funciona)
netstat -tulpn                                               ②
```

① **ss** (socket statistics) es reemplazo moderno de netstat
② **netstat** funciona pero deprecado

Opciones:

- **t** = TCP
- **u** = UDP
- **l** = Listen (servicios esperando)
- **p** = Process (mostrar qué proceso)
- **n** = Numeric (IP en vez de nombres)

Salida:

```
State     Recv-Q Send-Q Local Address    Foreign Address  State    PID/Program
LISTEN      0     128  0.0.0.0:22       0.0.0.0:*       LISTEN   1234/sshd
LISTEN      0     128  0.0.0.0:80       0.0.0.0:*       LISTEN   5678/nginx
ESTAB       0      0  192.168.1.100:22 192.168.1.50:45678 ESTAB   9876/sshd
```

170.10.2 Ver Solo Conexiones TCP Establecidas

Listing 170.22 BASH

```
ss -tn | grep ESTAB | wc -l                                    ①
```

① **grep ESTAB** filtra conexiones establecidas, **wc -l** cuenta

Salida:

```
247
```

Hay 247 conexiones TCP activas.

170.10.3 Encontrar Proceso Usando Puerto

Listing 170.23 BASH

```
# ¿Quién está usando puerto 8080?
ss -tulpn | grep :8080                                         ①
```

① **grep :8080** busca puerto 8080

Salida:

```
LISTEN  0  128  0.0.0.0:8080  0.0.0.0:*  LISTEN  12345/java
```

Proceso Java con PID 12345 está usando puerto 8080.

170.11 Ejemplos Prácticos Reales

170.11.1 Ejemplo 1: Monitoreo Completo del Servidor

Script que muestra salud general:

Uso:

170.11.2 Ejemplo 2: Alerta si CPU >80%

170.11.3 Ejemplo 3: Dashboard en Tiempo Real

170.12 Tabla de Referencia Rápida

Herramienta	Uso	Ejemplo
top	Monitor procesos interactivo	**top**
htop	Top mejorado con colores	**htop**
free -h	RAM disponible	**free -h**
df -h	Espacio en disco	**df -h**
du -sh	Uso por carpeta	**du -sh /home/***
iostat -x 1	I/O del disco	**iostat -x 1**
ss -tn	Conexiones TCP	**ss -tn**
vmstat 1	Estadísticas VM	**vmstat 1 10**

170.13 Quiz: Verificar Comprensión

Pregunta 1: ¿Qué significa "load average 4.0"?

a) CPU está al 40%
b) Hay 4 procesos ejecutándose
c) En máquina 4-core = CPU 100% ocupado (Correcto)
d) Sistema necesita 4 GB más RAM
Explicación: Load 1.0 por core = 100% uso. En máquina 4-core, load 4.0 = saturado.

Pregunta 2: ¿Cuál es la diferencia entre "Usado" y "Disponible" en free -h?

a) Son iguales
b) "Usado" incluye cache que puede liberarse. "Disponible" es realmente free (Correcto)
c) "Disponible" es más pequeño siempre
d) "Usado" no cuenta swap
Explicación: Antes "Usado + Libre = Total". Ahora Linux 3.14+ muestra "Disponible" que es lo que realmente puedes usar.

Pregunta 3: ¿Cómo encuentras qué proceso usa puerto 8080?

a) netstat 8080
b) lsof -i :8080
c) ss -tulpn | grep :8080 (Correcto)
d) top | grep 8080
Explicación: **ss -tulpn** muestra puertos y procesos. **grep :8080** filtra por puerto.

170.14 Práctica: Crear Dashboard Personalizado

Ejercicio: Script que muestra salud en 1 página

Ejecutar:

170.15 Recursos Adicionales

- **Man Pages**: **man top**, **man htop**, **man free**, **man df**, **man iostat**, **man ss**
- **Linux Performance Tools**: https://www.brendangregg.com/linuxperf.html
- **htop Documentation**: https://htop.dev/
- **Sysstat Manual**: https://github.com/sysstat/sysstat

170.16 Conclusión

Ahora sabes: Ver procesos con top y htop Monitorear CPU, RAM, disco Encontrar qué consume recursos Ver conexiones de red Crear alertas y dashboards

Para Abacom: Crea un script de monitoreo y úsalo diariamente para mantener servidores saludables.

Próximo paso: Integra estas herramientas en scripts de mantenimiento automático.

Listing 170.24 BASH

```
#!/bin/bash
# Script de salud del servidor para Abacom

echo "=== MONITOREO DEL SERVIDOR ==="
echo

# CPU y Carga
echo "  CPU y Carga:"
top -b -n 1 | head -3
echo

# RAM
echo "  Memoria:"
free -h | head -2
echo

# Disco
echo "  Disco:"
df -h | grep "^/dev" | awk '{printf "%-15s %6s %6s %6s\n", $1, $3, $2, $5}'
echo

# Procesos por CPU
echo "  Top 3 procesos por CPU:"
top -b -n 1 | head -7 | tail -3
echo

# Procesos por Memoria
echo "  Top 3 procesos por Memoria:"
top -b -n 1 -o %MEM | head -7 | tail -3
echo

# Conexiones de red
echo "  Conexiones TCP activas:"
ss -tn | grep ESTAB | wc -l
echo

echo "  Monitoreo completado: $(date)"
```

Listing 170.25 BASH

```
chmod +x monitor.sh
./monitor.sh

# Salida esperada:
# === MONITOREO DEL SERVIDOR ===
#
#   CPU y Carga:
# load average: 0.89, 0.76, 0.68
# ...
```

Listing 170.26 BASH

```
#!/bin/bash
# Alerta si CPU está saturado

LIMITE=80
CPU=$(top -b -n 1 | grep "Cpu(s)" | awk '{print $2}' | cut -d'%' -f1 | cut -d' ' -f1)

if (( $(echo "$CPU > $LIMITE" | bc -l) ))
then
  echo "   ALERTA: CPU está al ${CPU}%"
  # Enviar email, Slack, etc
  echo "ALERTA CPU" | mail -s "Servidor saturado" admin@abacom.es
fi
```

Listing 170.27 BASH

```bash
#!/bin/bash
# Monitoreo continuo en terminal

while true
do
  clear
  echo "=== MONITOREO EN VIVO ==="
  echo "Hora: $(date '+%H:%M:%S')"
  echo

  # Mostrar top 5 procesos
  echo "Top 5 procesos por CPU:"
  top -b -n 1 | tail -6
  echo

  # Mostrar RAM
  echo "Memoria (MB):"
  free -m | head -2
  echo

  sleep 2
done
```

Listing 170.28 BASH

```
#!/bin/bash
# dashboard.sh - Monitoreo de servidor

clear
printf "                                \n"
printf "    DASHBOARD DE SERVIDOR ABACOM                      \n"
printf "    Generado: $(date '+%Y-%m-%d %H:%M:%S')               \n"
printf "                                \n\n"

# CPU
printf "CPU (Load Average):\n"
uptime | awk -F'load average:' '{print "  "$2}'
printf "\n"

# RAM
printf "Memoria RAM:\n"
free -h | awk 'NR==2 {printf "  Total: %s | Usado: %s | Disponible: %s\n", $2, $3, $7}'
printf "\n"

# Disco
printf "Disco (Top 3):\n"
df -h | grep "^/dev" | sort -k5 -rn | head -3 | awk '{printf "  %s: %3s / %s\n", $6, $5,
printf "\n"

# Procesos
printf "Procesos (Top CPU):\n"
top -b -n 1 | tail -5 | awk '{printf "  %s: %s%%\n", $12, $9}' | head -3
printf "\n"

printf "  Fin de monitoreo\n"
```

Listing 170.29 BASH

```
chmod +x dashboard.sh
./dashboard.sh
```

171 Ranger: Gestor de Archivos Terminal para Servidores

Listing 171.1 `QUARTO-TITLE-BLOCK`

Navegación Eficiente de Directorios sin GUI

172 Ranger: Navegación de Archivos en Terminal

172.1 Introducción a Ranger

Ranger es un gestor de archivos basado en terminal escrito en Python. Proporciona:

- Navegación visual multi-panel (similar a Norton Commander)
- Vista previa de archivos
- Operaciones rápidas de archivo
- Soporte para plugins personalizados
- Integración perfecta con herramientas CLI

Por qué es importante en servidores:

- No requiere X11/GUI
- Funciona por SSH sin overhead gráfico
- Más eficiente que `cd` + `ls` iterativo
- Vista previa de código directamente
- Operaciones de archivo más intuitivas

Tiempo estimado: 30-45 minutos

172.2 Objetivos de Aprendizaje

Después de este anexo, serás capaz de:

- Instalar y configurar Ranger
- Navegar el sistema de archivos eficientemente
- Usar visor de archivos y miniaturas
- Ejecutar operaciones de archivo (copiar, mover, eliminar)
- Personalizar Ranger con configuración avanzada
- Integrar Ranger con otros comandos CLI
- Usar Ranger en conexiones SSH remotas

172.3 Tabla de Contenidos

172.4 Instalación

172.4.1 Linux (Debian/Ubuntu)

Listing 172.1 BASH

```
# Instalar Ranger desde repositorio
sudo apt update                                                    ①
sudo apt install ranger                                            ②

# Instalar dependencias opcionales para vista previa mejorada
sudo apt install highlight atool w3m-img                           ③

# Verificar instalación
ranger --version                                                   ④
```

① **apt update** actualiza la lista de paquetes disponibles
② **apt install ranger** instala el gestor de archivos Ranger
③ Dependencias opcionales: **highlight** (sintaxis), **atool** (archivos), **w3m-img** (imágenes)
④ **ranger –version** verifica la versión instalada

172.4.2 macOS (Homebrew)

① **brew install ranger** instala Ranger desde Homebrew
② Utilidades adicionales para vista previa mejorada
③ **ranger –version** muestra la versión instalada

Listing 172.2 BASH

```
# Instalar Ranger con Homebrew
brew install ranger                                                    ①

# Instalar utilidades de vista previa
brew install highlight atool                                           ②

# Verificar instalación
ranger --version                                                       ③
```

172.4.3 Compilación desde Fuente

Listing 172.3 BASH

```
# Para versión más reciente o soporte especial
git clone https://github.com/ranger/ranger.git                         ①
cd ranger
sudo make install                                                      ②

# O instalación local sin sudo
python install.py --local                                              ③
```

① **git clone** descarga el repositorio oficial de Ranger
② **make install** compila e instala globalmente
③ **python install.py** –**local** instalación en directorio de usuario

172.5 Interfaz y Navegación Básica

172.5.1 Arranque Inicial

① **ranger** abre el gestor en el directorio actual ($HOME o PWD)
② **ranger /ruta** abre Ranger en una ruta específica
③ **ranger** –**no-mounts** carga más rápido sin montar sistemas de archivos remotos

172.5.2 Interfaz de Ranger

```
PANEL IZQUIERDO    PANEL CENTRAL    PANEL DERECHO
Directorios        Contenido dir    Vista previa
padres             actual           del archivo
```

Listing 172.4 BASH

```
# Abrir Ranger en directorio actual
ranger                                                                    ①

# Abrir Ranger en directorio específico
ranger /var/log                                                           ②

# Abrir Ranger sin montar sistemas de archivos
ranger --no-mounts                                                        ③
```

```
                                    seleccionado
 /                Documents/       (code, images,
 home/            Downloads/      text files)
 usr/             Pictures/
 var/             file.txt      File: file.txt
 etc/             script.sh        Size: 1.2K
 tmp/             image.png        Mode: -rw-r--r--
                  README.md
```

Componentes:

- **Panel izquierdo:** Directorio padre (permite navegación rápida hacia arriba)
- **Panel central:** Contenido del directorio actual (donde operas)
- **Panel derecho:** Vista previa del archivo seleccionado
- **Barra inferior:** Información del archivo, marca, estado

172.5.3 Navegación de Directorios

172.5.4 Movimiento Básico

172.5.5 Movimiento Avanzado

① **:cd** es el comando interno de Ranger (similar a cd en shell)

172.5.6 Búsqueda de Archivos

① / inicia búsqueda (presiona Enter para confirmar)
② Expresiones regulares completas soportadas

Listing 172.5 BASH

```
# Navegar hacia arriba en el árbol de directorios
# Presionar: h (left arrow)

# Entrar en un directorio
# Presionar: j (down) + Enter o l (right arrow)

# Moverse entre archivos
j  # Siguiente archivo (↓)
k  # Archivo anterior (↑)

# Saltar a inicio/fin de lista
gg  # Ir al inicio (first file)
G   # Ir al final (last file)
```

172.6 Operaciones de Archivos

172.6.1 Selección de Archivos

① **Space** marca/desmarca el archivo actual para operación
② **sa** (select all) marca todos los archivos
③ **si** (select invert) invierte la selección
④ **uv** (unselect via) desmarca todos
⑤ **:select_file** selecciona por patrón regex

172.6.2 Copiar, Mover, Eliminar

① **yy** copia archivo(s) al portapapeles interno
② **pp** pega en el directorio actual
③ **dd** corta para mover después
④ **dD** elimina sin recuperación (cuidado!)
⑤ **cw** abre editor para renombrar

172.7 Visor de Archivos

172.7.1 Vista Previa Integrada

① **zi** alterna la visualización del panel de vista previa
② **:du** (disk usage) muestra tamaño de directorios
③ **zh** muestra/oculta archivos que comienzan con punto

Listing 172.6 BASH

```
# Ir al home del usuario
~

# Ir a raíz del sistema
/

# Ir al directorio anterior (como 'cd -')
-

# Saltar a marcador
'<letra>  # Ir a marcador (ej: 'a, 'b, 'c)

# Ir a ruta específica (comando)
:cd /var/log                                                    ①
```

Listing 172.7 BASH

```
# Buscar archivo por nombre (búsqueda incremental)
/nombre_archivo                                                 ①

# Búsqueda siguiendo coincidencias
n  # Siguiente coincidencia
N  # Coincidencia anterior

# Buscar archivo con expresión regular
/.*\.log$                                                       ②
```

172.7.2 Formatos Soportados

Vista Previa de Ranger

Ranger puede previsualizar (con dependencias instaladas):

- **Texto:** .txt, .md, .py, .sh, .log, .csv, .json, etc.
- **Código:** Resaltado de sintaxis (highlight)
- **Imágenes:** .jpg, .png, .gif (en terminal capaz de imágenes)
- **Archivos:** .zip, .tar, .gz (con atool)
- **Documentos:** .pdf (con pdftotext)

172.7.3 Abrir Archivos

① **e** abre archivo en editor configurado ($EDITOR)
② **!** o **S** abre shell interactivo en directorio actual

Listing 172.8 BASH

```
# Seleccionar/deseleccionar archivo actual
Space                                                                    ①

# Seleccionar todos los archivos en directorio
sa                                                                       ②

# Invertir selección (seleccionar no-seleccionados)
si                                                                       ③

# Deseleccionar todos
uv                                                                       ④

# Seleccionar por extensión
# Por ejemplo, todos los .log:
:select_file \.log$                                                      ⑤
```

③ **x** ejecuta archivo si tiene permisos de ejecución

172.8 Configuración Avanzada

172.8.1 Ubicación de Archivos de Configuración

① **rc.conf** contiene atajos, temas, opciones de interfaz
② **rifle.conf** define qué programa abre cada tipo de archivo
③ **scope.sh** personaliza vista previa para tipos específicos
④ **–copy-config=all** crea todos los archivos en ~/.config/ranger/

172.8.2 Configuración Básica (rc.conf)

Opciones útiles a editar:

① **colorscheme** opciones: default, jungle, snow, solarized
② **confirm_on_delete** previene eliminación accidental
③ **show_hidden** true/false controla visibilidad de archivos ocultos

172.8.3 Rifle.conf - Asociaciones de Archivos

Ejemplo de configuración personalizada:

① **ext** define extensión(es) de archivo

Listing 172.9 BASH

```
# Copiar archivo seleccionado
yy  # Copiar a portapapeles                                    ①
pp  # Pegar en ubicación actual                                ②

# Mover/renombrar
dd  # Cortar archivo                                           ③
pp  # Pegar en nueva ubicación

# Eliminar permanentemente
dD  # Eliminar archivo(s) seleccionado(s)                      ④

# Cambiar nombre (renombrar)
cw                                                             ⑤
# Edita nombre en editor (vim/nano)
# Presiona ESC + :wq para guardar

# Crear archivo/directorio nuevo
touch myfile.txt  # Crea archivo
mkdir mydir       # Crea directorio
```

② **else** es el comportamiento por defecto si no coincide nada

---

172.9 Integración con Otros Comandos

172.9.1 Usar Ranger desde Scripts

① –**choosefile** permite que Ranger retorne archivo seleccionado
② Combinación con pipes para procesar archivos

172.9.2 Alias Útiles en .bashrc

① Alias corto para ranger
② Ranger solo directorios
③ Ranger vista lista solamente

172.9.3 Integración con fzf (Fuzzy Finder)

Listing 172.10 BASH

```
# Habilitar/deshabilitar vista previa (panel derecho)
zi                                                          ①

# Ver contenido de archivo de texto
# (automático en panel derecho)
cat file.txt  # Shell: equivalente

# Vista de directorio (tamaño de contenidos)
:du                                                         ②

# Mostrar archivos ocultos
zh                                                          ③
```

Listing 172.11 BASH

```
# Abrir archivo con editor por defecto
e                                                           ①
# O presionar Enter (abre con aplicación por defecto)

# Abrir con programa específico
o  # Menú de programas alternativos

# Abrir terminal en directorio actual
!                                                           ②
# O presionar S (capital S) para subshell

# Ejecutar archivo ejecutable
x                                                           ③
```

172.10 Ranger en Servidores Remotos

172.10.1 Conexión SSH Eficiente

① **ssh usuario@servidor** conecta por SSH
② **scp** copia archivos de configuración a servidor

172.10.2 Instalación en Servidor Ubuntu

① Instalación completa con dependencias

172.10.3 Optimización para Servidores

Listing 172.12 BASH

```
# Configuración de Ranger
~/.config/ranger/

# Archivos principales:
~/.config/ranger/rc.conf        # Configuración principal          ①
~/.config/ranger/rifle.conf     # Asociaciones de archivos         ②
~/.config/ranger/scope.sh       # Script de vista previa           ③
~/.config/ranger/colorscheme/   # Temas de color

# Generar configuración por defecto
ranger --copy-config=all                                           ④
```

Listing 172.13 BASH

```
# Editar configuración
nano ~/.config/ranger/rc.conf
```

① **vcs_backend disable** desactiva control de versiones (lento en SSH)
② **use_preview_script false** desactiva vista previa compleja

---

172.11 Atajos de Teclado Esenciales

172.11.1 Navegación

Tecla	Acción
`h` / `j` / `k` / `l`	← ↓ ↑ → (movimiento estilo vim)
`gg`	Ir al inicio de la lista
`G`	Ir al final de la lista
`J` / `K`	Scroll página hacia abajo/arriba
`~`	Ir al directorio home
`/`	Ir a raíz del sistema
`-`	Ir al directorio anterior
`u`	Ir hacia arriba un directorio
`H` / `L`	Historial anterior/siguiente

172.11.2 Selección y Operaciones

Listing 172.14 CONF

```
# Mostrar números de línea en vista previa
set preview_script ~/.config/ranger/scope.sh

# Usar vi-style keybindings
set vcsync true

# Colores personalizados
colorscheme default                                                    (1)

# Confirmación antes de eliminar
set confirm_on_delete move                                             (2)

# Anchos de panel
set column_ratios 1,3,4

# Mostrar archivos ocultos por defecto
set show_hidden false                                                  (3)

# Orden de directorios
set sort directories_first

# Número máximo de líneas de vista previa
set preview_max_size 262144
```

Listing 172.15 BASH

```
# Ver configuración actual
cat ~/.config/ranger/rifle.conf | head -30
```

Tecla	Acción
`Space`	Seleccionar/deseleccionar archivo
`sa`	Seleccionar todos
`si`	Invertir selección
`uv`	Deseleccionar todos
`yy`	Copiar
`dd`	Cortar
`pp`	Pegar
`dD`	Eliminar
`cw`	Renombrar
`cO`	Copiar aquí (prompt)
`cR`	Mover aquí (prompt)

Listing 172.16 CONF

```
# Archivos de código con editor
ext py|sh|js|rb = vim "$@"                                    (1)

# PDFs con visor
ext pdf = pdfviewer "$@"

# Imágenes con visor de imagen
ext jpg|png|gif|jpeg = feh "$@"

# Videos con reproductor
ext mp4|mkv|avi = mpv "$@"

# Comprimidos con gestor de archivo
ext zip|tar|gz = atool -x "$@"

# Fallback: editor por defecto
else = $EDITOR "$@"                                           (2)
```

Listing 172.17 BASH

```
# Navegar con Ranger y ejecutar comando en seleccionado
ranger_file=$(ranger --choosefile=/tmp/chosenfile --show-only-dir)
# Luego procesar $ranger_file                                 (1)

# Copiar ruta del archivo seleccionado
ranger --choosefile=/tmp/f /var/log && xargs cat < /tmp/f     (2)
```

172.11.3 Visualización

Tecla	Acción
`i`	Ciclar entre 4 modos de vista
`I`	Información detallada del archivo
`zh`	Mostrar/ocultar archivos ocultos
`zf`	Filtrar archivos (por extensión)
`z?`	Ayuda de atajos `z`

172.11.4 Archivos

Tecla	Acción
`e`	Abrir con editor ($EDITOR)
`o`	Menú de programas alternativos

Tecla	Acción
`x`	Ejecutar archivo
`! / S`	Abrir shell en directorio
`/`	Buscar archivo (regex)
`n / N`	Siguiente/anterior coincidencia

172.11.5 Configuración

Tecla	Acción
`:`	Comandos (`:set`, `:cd`, `:open`, etc.)
`m<letra>`	Crear marcador en letra
`'<letra>`	Ir a marcador
`q`	Salir de Ranger

172.12 Laboratorio Práctico

172.12.1 Ejercicio 1: Navegación Básica

Objetivo: Familiarizarse con movimiento en Ranger

① **ranger** abre el gestor de archivos en directorio actual

172.12.2 Ejercicio 2: Operaciones de Archivos

Objetivo: Practicar copiar, mover, eliminar

① **mkdir -p** crea directorios anidados (origen y destino para laboratorio)
② **cd** cambia al directorio origen

172.12.3 Ejercicio 3: Vista Previa y Visualización

Objetivo: Usar visor de archivos

172.12.4 Ejercicio 4: Búsqueda y Filtrado

Objetivo: Buscar y filtrar archivos

Listing 172.18 BASH

```
# Agregar a ~/.bashrc
alias r='ranger'                                    ①
alias rc='ranger --show-only-dir'                   ②
alias rl='ranger --mode=3'                          ③
```

Listing 172.19 BASH

```
# Instalar fzf
sudo apt install fzf

# Usar Ranger + fzf en .bashrc
ranger-fzf() {
    ranger --choosefile=/tmp/rf && \
    cat /tmp/rf | fzf | xargs vim
}
```

172.12.5 Ejercicio 5: Integración con Shell

Objetivo: Usar Ranger desde scripts

① **cat > archivo « 'EOF'** crea un archivo multilinea con contenido de script
② **chmod +x** hace el script ejecutable (permisos de ejecución)
③ **~/choose-file.sh** ejecuta el script desde la carpeta home

---

172.13 Troubleshooting Común

⚠ Problemas Frecuentes

Problema 1: Vista previa no funciona

Listing 172.28 BASH

```
# Solución: Instalar dependencias
sudo apt install highlight atool w3m-img
# Luego reiniciar Ranger
```

Problema 2: Ranger es lento en SSH

Listing 172.20 BASH

```
# Ranger sobre SSH (sin X11, muy ligero)
ssh usuario@servidor                                                    ①
ranger  # En servidor remoto, rangerse ejecuta en terminal

# Copiar configuración a servidor
ssh usuario@servidor "mkdir -p ~/.config/ranger"
scp ~/.config/ranger/rc.conf usuario@servidor:~/.config/ranger/         ②
```

Listing 172.21 BASH

```
# En servidor remoto vía SSH
ssh admin@192.168.1.100

# Una vez conectado
sudo apt update && sudo apt install -y ranger highlight atool           ①

# Crear estructura de configuración
ranger --copy-config=all

# Verificar instalación
ranger --version
```

Listing 172.29 BASH

```
# Solución: Desactivar preview en rc.conf
set use_preview_script false
set vcs_backend_git disable
```

Problema 3: No puede copiar/pegar entre rangery shell

Listing 172.30 BASH

```
# Ranger usa portapapeles interno, no sistema
# Usar comando :open_console para shell directo
```

172.14 Recursos Adicionales

- **Documentación Oficial:** https://ranger.github.io/

Listing 172.22 BASH

```
# En ~/.config/ranger/rc.conf (servidor)
set vcs_backend_git disable                                    ①
set vcs_backend_hg disable
set vcs_backend_bzr disable

# Desactivar vista previa para mejorar velocidad
set use_preview_script false                                   ②

# Reducir máximo de líneas de vista previa
set preview_max_size 102400  # 100KB

# Colores básicos (más rápido)
colorscheme default
```

- **GitHub Repository:** https://github.com/ranger/ranger
- **Wiki:** https://github.com/ranger/ranger/wiki
- **Cheat Sheet:** https://github.com/ranger/ranger/wiki/VCS-Integration

---

172.15 Checklist Final

Después de completar este anexo, deberías poder:

- ☐ Instalar Ranger en Linux/macOS
- ☐ Navegar directorios con hjkl
- ☐ Seleccionar y copiar/mover archivos
- ☐ Ver vista previa de archivos
- ☐ Personalizar rc.conf y rifle.conf
- ☐ Usar Ranger en conexiones SSH
- ☐ Integrar Ranger con scripts shell
- ☐ Resolver problemas comunes
- ☐ Usar todos los atajos esenciales

Última actualización: 2024-01-30
Autor: Diego Saavedra (Abacom)
Dificultad: Intermedio
Categoría: Herramientas de Terminal

Listing 172.23 BASH

```
# 1. Abrir Ranger
ranger                                                                    ①

# 2. Navegar a /etc (usando h para subir, j/k para mover)
# - Presiona 'h' hasta llegar a raíz (/)
# - Presiona 'j' para mover a 'etc'
# - Presiona 'l' o Enter para entrar

# 3. Explorar archivos de configuración
# - Buscar 'passwd' (presiona /)
# - Ver preview en panel derecho
# - Presiona 'e' para ver contenido completo

# 4. Ir a /var/log
# - Presiona '/' para ir a raíz
# - Navega a 'var' → 'log'
# - Observa archivos de log
```

Listing 172.24 BASH

```
# 1. Crear estructura de prueba
mkdir -p ~/ranger-lab/origen ~/ranger-lab/destino                    ①
cd ~/ranger-lab/origen                                               ②

# 2. Crear archivos de prueba
touch file1.txt file2.txt file3.log script.sh

# 3. Abrir Ranger en origen
ranger

# 4. Operaciones en Ranger:
# - Seleccionar archivo1.txt (Space)
# - Copiar (yy)
# - Navegar a destino
# - Pegar (pp)

# 5. Mover archivo
# - Seleccionar file2.txt
# - Cortar (dd)
# - Navegar a destino
# - Pegar (pp)

# 6. Renombrar
# - Seleccionar file3.log
# - Presionar 'cw'
# - Cambiar nombre a 'renamed.log'
```

Listing 172.25 BASH

```
# 1. Crear archivos de ejemplo
cat > test.py << 'EOF'
#!/usr/bin/env python3
# Script de prueba
for i in range(10):
    print(f"Line {i}")
EOF

# 2. Abrir Ranger
ranger

# 3. Ver vista previa:
# - Navega a test.py
# - Panel derecho muestra contenido
# - Presiona 'zi' para alternar preview
# - Presiona 'e' para editar

# 4. Ver información extendida
# - Selecciona archivo
# - Presiona 'I' para info detallada
```

Listing 172.26 BASH

```
# 1. En Ranger, navega a un directorio con muchos archivos
# Ejemplo: /usr/share/doc

# 2. Búsqueda básica:
# - Presiona '/'
# - Escribe 'python'
# - Presiona Enter
# - Usa 'n' para siguiente coincidencia

# 3. Búsqueda avanzada (regex):
# - Presiona '/'
# - Escribe '.*\.py$' (archivos .py)
# - Presiona Enter

# 4. Filtrado:
# - Presiona 'zf'
# - Escribe '.log' (muestra solo .log)
```

Listing 172.27 BASH

```
# 1. Crear script que usa Ranger
cat > ~/choose-file.sh << 'EOF' # <1>
#!/bin/bash
# Script que usa Ranger para seleccionar archivo
ranger --choosefile=/tmp/choice /tmp
if [ -f /tmp/choice ]; then
    file=$(cat /tmp/choice)
    echo "Archivo seleccionado: $file"
    file "$file"  # Mostrar tipo de archivo
fi
EOF

# 2. Hacer script ejecutable
chmod +x ~/choose-file.sh                                                    ②

# 3. Ejecutar script
~/choose-file.sh                                                             ③
```

Part XIX

Anexos: Mini-Cursos (Nivel 2: Intermedio)

173 Anexo E: Búsqueda y Procesamiento de Texto - grep, find, sed, awk

Listing 173.1 QUARTO-TITLE-BLOCK

174 Anexo E: Búsqueda y Procesamiento de Texto - grep, find, sed, awk

174.1 Introducción

Administrar servidores Linux significa constantemente **buscar archivos** y **procesar texto**. Desde logs enormes hasta archivos de configuración, necesitas poder encontrar y modificar información rápidamente.

Este anexo enseña las **4 herramientas más poderosas** de Linux para procesamiento de texto:

- **grep** - Buscar líneas que coincidan con patrón
- **find** - Encontrar archivos por nombre, tamaño, fecha, etc
- **sed** - Edición de streams (modificar texto en scripts)
- **awk** - Procesamiento de columnas de texto

En este anexo aprenderás:

- Búsqueda básica y avanzada con grep
- Encontrar archivos específicos con find
- Editar y reemplazar texto con sed
- Procesar columnas con awk
- Piping y combinación de herramientas
- Casos de uso reales en Abacom

Duración estimada: 2 horas de práctica

174.2 Herramientas de Búsqueda en Diferentes SOs

174.2.1 Linux (Herramientas Nativas)

Disponibilidad:

- grep: Siempre disponible
- find: Siempre disponible
- sed: Siempre disponible (GNU sed)
- awk: Siempre disponible (GNU awk / gawk)

Listing 174.1 BASH

```
# Linux tiene grep, find, sed, awk nativas y optimizadas

# Buscar en archivo
$ grep "error" /var/log/syslog
$ grep -r "error" /var/log  # Recursivo

# Encontrar archivos
$ find / -name "*.log" -type f
$ find /home -user diego -type d

# Editar con sed
$ sed 's/error/ERROR/g' archivo.txt
$ sed -i 's/old/new/g' archivo.txt  # In-place

# Procesar columnas con awk
$ ps aux | awk '{print $1, $2, $3}'
$ cat datos.csv | awk -F, '{sum += $2} END {print sum}'
```

174.2.2 macOS (Compatible pero diferente)

Disponibilidad:

- grep: Disponible (BSD grep)
- find: Disponible (BSD find)
- sed: Disponible pero limitado (BSD sed)
- awk: Disponible (BSD awk)
- Recomendado: Instalar GNU tools via Homebrew

174.2.3 Windows (PowerShell / Git Bash)

Disponibilidad:

- grep: No nativo (via Git Bash o WSL)
- find: No nativo (via Git Bash o WSL)
- sed: No nativo (via Git Bash o WSL)
- awk: No nativo (via Git Bash o WSL)
- PowerShell equivalentes: Disponibles
- Recomendado: **Usar WSL o PowerShell nativo**

Tabla comparativa: | Herramienta | Linux | macOS | Windows | |————|——-|——-|————| | grep | GNU | BSD | (WSL/Git Bash) | | find | GNU | BSD | (WSL/Get-ChildItem) | | sed | GNU | BSD | (WSL/Get-Content) | | awk | GNU | BSD | (WSL/Select-Object) |

Recomendación para Abacom:

Listing 174.2 BASH

```
# macOS tiene las herramientas, pero versiones BSD (no GNU)
# Las opciones pueden ser ligeramente diferentes

# Buscar en archivo
$ grep "error" /var/log/system.log
$ grep -r "error" /var/log  # Recursivo (igual)

# DIFERENCIA: BSD sed vs GNU sed
# macOS sed es BSD sed (menos opciones)
$ sed -i '' 's/error/ERROR/g' archivo.txt  # '' es requerido en macOS
$ sed -i 's/error/ERROR/g' archivo.txt  # Linux

# Mejor: Instalar GNU sed via Homebrew
$ brew install gnu-sed
$ gsed -i 's/error/ERROR/g' archivo.txt  # GNU sed

# awk (BSD awk, pero compatible)
$ ps aux | awk '{print $1, $2, $3}'
$ cat datos.csv | awk -F, '{sum += $2} END {print sum}'

# find (similar a Linux)
$ find . -name "*.log" -type f
```

- Linux: Usar herramientas nativas (grep, find, sed, awk)
- macOS: Instalar `brew install gnu-sed` para compatibilidad
- Windows: Usar WSL para acceso completo a herramientas Linux

---

174.3 Concepto 1: GREP - Buscar Texto en Archivos

174.3.1 Búsqueda Básica

① **grep "patrón" archivo** busca todas las líneas que contienen "error"

Salida (solo líneas con "error"):

```
Jan 29 15:45:23 server kernel: [error] Failed to allocate memory
Jan 29 16:12:01 server nginx: [error] Connection refused
Jan 29 17:34:56 server php: [error] Undefined variable in script.php
```

Listing 174.3 POWERSHELL

```
# Windows NO tiene grep, find, sed, awk nativamente
# Opciones:

# OPCIÓN 1: PowerShell (alternativas nativas)
PS> # Buscar en archivo
PS> Select-String -Path "C:\logs\error.log" -Pattern "error"

PS> # Encontrar archivos
PS> Get-ChildItem -Path "C:\" -Filter "*.log" -Recurse

PS> # Reemplazar en archivo
PS> (Get-Content "archivo.txt") -replace "old", "new" | Set-Content "archivo.txt"

PS> # Procesar columnas (similar a awk)
PS> Get-Process | Select-Object Name, CPU, Memory

# OPCIÓN 2: Git Bash (incluye grep, sed, awk)
$ grep "error" C:\logs\error.log
$ find C:\projects -name "*.log"
$ sed 's/error/ERROR/g' archivo.txt

# OPCIÓN 3: Windows Subsystem for Linux (WSL)
PS> wsl grep "error" /mnt/c/logs/error.log
PS> wsl find / -name "*.log"
# Acceso completo a herramientas Linux
```

174.3.2 Búsqueda Case-Insensitive

① **-i** (ignore case) ignora mayúsculas/minúsculas

174.3.3 Contar Ocurrencias

① **-c** (count) cuenta líneas coincidentes

Salida:

```
47
```

Hay 47 líneas con “error”.

Listing 174.4 BASH

```
# Buscar "error" en archivo
grep "error" /var/log/syslog                                    ①
```

Listing 174.5 BASH

```
# Buscar "ERROR", "Error", "error" todo junto
grep -i "error" /var/log/syslog                                 ①
```

174.3.4 Buscar en Múltiples Archivos

① ***.log** wildcard que expande a todos los .log

Salida:

```
/var/log/syslog:Jan 29 15:45:23 server kernel: [error] Failed
/var/log/nginx.log:Jan 29 16:12:01 server nginx: [error] Connection
/var/log/apache2.log:Jan 29 17:34:56 server apache: [error] Forbidden
```

174.3.5 Mostrar Contexto (Líneas Alrededor)

① **-C 3** (context) 3 líneas antes y después
② **-B 2** (before), **-A 2** (after)

174.3.6 Expresiones Regulares (Regex)

① ^ = inicio de línea
② **$** = fin de línea
③ **-E** (extended regex) permite | (o)

174.3.7 Buscar Archivos Recursivamente

① **-r** (recursive) busca en subcarpetas
② **-n** (number) muestra número de línea

Salida:

```
/home/diego/proyecto/main.py:45: TODO: Implementar autenticación
/home/diego/proyecto/utils.py:123: TODO: Optimizar función lenta
```

Listing 174.6 BASH

```
# ¿Cuántas veces aparece "error"?
grep -c "error" /var/log/syslog                                    ①
```

Listing 174.7 BASH

```
# Buscar "error" en TODOS los .log
grep "error" /var/log/*.log                                        ①
```

174.3.8 Invertir Búsqueda (NOT)

① **-v** (invert) muestra líneas que NO coinciden

174.4 Concepto 2: FIND - Encontrar Archivos

174.4.1 Buscar por Nombre

① **-name "patrón"** busca por nombre
② Sin **-type f** incluye directorios también

174.4.2 Buscar Solo Archivos o Directorios

① **-type f** = archivos
② **-type d** = directorios

174.4.3 Buscar por Tamaño

① **+100M** = mayor que 100 MB
② **-1K** = menor que 1 KB
③ **5G** = exactamente 5 GB

174.4.4 Buscar por Fecha

① **-mtime -7** = últimos 7 días
② **-mtime +30** = más de 30 días atrás
③ **-atime 0** = accedidos hoy

Listing 174.8 BASH

```
# Mostrar 3 líneas antes y después de "error"
grep -C 3 "error" /var/log/syslog                                    ①

# O por separado:
grep -B 2 -A 2 "error" /var/log/syslog                               ②
```

Listing 174.9 BASH

```
# Buscar líneas que empiezen con "ERROR"
grep "^ERROR" /var/log/syslog                                        ①

# Buscar líneas que terminen con "failed"
grep "failed$" /var/log/syslog                                       ②

# Buscar patrón flexible (error, warning, critical)
grep -E "error|warning|critical" /var/log/syslog                     ③
```

174.4.5 Ejecutar Comando en Resultados

① **-delete** elimina archivos encontrados
② **-exec comando {} ;** ejecuta comando en cada archivo
③ **{}** es placeholder para nombre del archivo

174.4.6 Buscar Archivos Vacíos

① **-empty** archivos de 0 bytes
② **-empty** directorios sin contenido

174.5 Concepto 3: SED - Editor de Streams

174.5.1 Reemplazar Texto (Substitución)

① **s/viejo/nuevo/** reemplaza (primera por línea)
② **g** = global (todas)
③ **i** = case-insensitive

Listing 174.10 BASH

```
# Buscar "TODO" en todos los .py en carpeta proyecto
grep -r "TODO" /home/diego/proyecto/                    ①

# Con número de línea
grep -rn "TODO" /home/diego/proyecto/                   ②
```

Listing 174.11 BASH

```
# Mostrar líneas SIN "error"
grep -v "error" /var/log/syslog                         ①
```

174.5.2 Usar Delimitador Diferente

① | como delimitador
② # como delimitador (también válido)

174.5.3 Eliminar Líneas

① **5d** = delete línea 5
② **5,10d** = delete rango
③ **/patrón/d** = delete líneas que coincidan
④ **/^$/d** = delete líneas vacías

174.5.4 Guardar Cambios a Archivo

① **-i** modifica el archivo original
② **-i.bak** crea backup antes de modificar

174.5.5 Inserciones y Adiciones

① **5i** = insert antes (before)
② **5a** = append después (after)
③ **$** = última línea

Listing 174.12 BASH

```
# Encontrar todos los archivos .log en /var/log
find /var/log -name "*.log"                                    ①

# Buscar archivos exactamente "nginx.conf"
find / -name "nginx.conf"                                      ②
```

Listing 174.13 BASH

```
# Solo archivos (no directorios)
find /home -type f -name "*.txt"                               ①

# Solo directorios
find /home -type d -name "backup*"                             ②
```

174.6 Concepto 4: AWK - Procesamiento de Columnas

174.6.1 Imprimir Columnas Específicas

① **$1** = primera columna
② **$1, $3** = columna 1 y 3
③ **$4, $1** = columna 4 y 1

Salida:

```
diego
carlos
ana

diego linux
carlos python
ana java

admin diego
developer carlos
engineer ana
```

174.6.2 Usar Delimitador Diferente

① **-F:** usa : como delimitador
② Imprimir UID (columna 3) y nombre (columna 1)

Listing 174.14 BASH

```
# Encontrar archivos mayores a 100 MB
find /home -type f -size +100M                                    ①

# Encontrar archivos menores a 1 KB
find /tmp -type f -size -1K                                       ②

# Exactamente 5 GB
find / -type f -size 5G                                           ③
```

Listing 174.15 BASH

```
# Archivos modificados en últimos 7 días
find /home -type f -mtime -7                                      ①

# Archivos modificados más de 30 días atrás
find /var/log -type f -mtime +30                                  ②

# Accedidos hoy
find / -type f -atime 0                                           ③
```

174.6.3 Condicionales con AWK

① **$2 > 30** = columna 2 mayor que 30
② **/patrón/** = líneas coincidentes
③ **NR==2** = línea número 2 (NR = Number of Record)

174.6.4 Operaciones Matemáticas

① **$2 * $3** = precio × cantidad
② **END** = después de procesar todo, mostrar suma

174.6.5 Variables y Strings

① Concatenar strings con espacios
② **-v var=valor** define variable antes

Listing 174.16 BASH

```bash
# Eliminar todos los .log del 2023
find /var/log -name "*.log.*" -mtime +365 -delete ①

# Contar líneas en todos los .py
find /home -name "*.py" -exec wc -l {} \; ②

# Cambiar permisos en lote
find /home -type d -exec chmod 755 {} \; ③
```

Listing 174.17 BASH

```bash
# Encontrar archivos vacíos
find /tmp -type f -empty ①

# Encontrar directorios vacíos
find /var -type d -empty ②
```

174.7 Ejemplos Prácticos Reales en Abacom

174.7.1 Ejemplo 1: Buscar y Contar Errores en Log

174.7.2 Ejemplo 2: Limpiar Logs Antiguos

174.7.3 Ejemplo 3: Procesar Archivo CSV

174.7.4 Ejemplo 4: Reporte de Uso de Disco

174.7.5 Ejemplo 5: Combinar Herramientas (Pipe)

Listing 174.18 BASH

```
# Reemplazar "apache2" por "nginx" (primera ocurrencia por línea)
sed 's/apache2/nginx/' /etc/hosts                                    ①

# Reemplazar TODAS las ocurrencias (global)
sed 's/apache2/nginx/g' /etc/hosts                                   ②

# Case-insensitive
sed 's/apache2/nginx/gi' /etc/hosts                                  ③
```

Listing 174.19 BASH

```
# Útil cuando hay "/" en patrón
sed 's|/var/log/|/backup/|g' /etc/fstab                              ①

# O con #
sed 's#/var/log/#/backup/#g' /etc/fstab                              ②
```

174.8 Tabla Rápida: Combinaciones Poderosas

Tarea	Comando
Buscar archivos grandes	**find / -type f -size +100M**
Buscar texto recursivo	**grep -rn "TODO" /home/**
Reemplazar en lote	**find . -name "*.txt" -exec sed -i 's/viejo/nuevo/g' {} ;**
Contar líneas en archivos	**find . -name "*.py" -exec wc -l {} + | tail -1**
Listar usuarios sistema	**awk -F: '{print $1}' /etc/passwd**
Archivos modificados hoy	**find / -type f -mtime 0**
Ver top 10 usuarios por size	**du -sh /home/* | sort -rh | head -10**

174.9 Errores Comunes

Error 1: Olvidar escapar caracteres especiales

Error 2: -i in-place sin backup

Error 3: Olvidar comillas

174.10 Quiz: Verificar Comprensión

Pregunta 1: ¿Qué diferencia hay entre grep y grep -r?

a) No hay diferencia
b) -r busca recursivamente en directorios (Correcto)
c) -r usa regex

Listing 174.20 BASH

```
# Eliminar línea 5
sed '5d' /etc/config.txt                                    ①

# Eliminar líneas 5-10
sed '5,10d' /etc/config.txt                                 ②

# Eliminar líneas que contengan "error"
sed '/error/d' /var/log/syslog                              ③

# Eliminar líneas vacías
sed '/^$/d' /etc/config.txt                                 ④
```

Listing 174.21 BASH

```
# Mostrar en pantalla (no modifica archivo)
sed 's/viejo/nuevo/g' archivo.txt

# Guardar en nuevo archivo
sed 's/viejo/nuevo/g' archivo.txt > archivo_nuevo.txt

# Modificar archivo en lugar (in-place)
sed -i 's/viejo/nuevo/g' archivo.txt                        ①

# Backup antes de modificar
sed -i.bak 's/viejo/nuevo/g' archivo.txt                    ②
```

Pregunta 3: ¿Qué hace **awk '{print $2}'**?

a) Imprime toda la línea
b) Imprime 2 caracteres
c) Imprime la segunda columna (Correcto)
d) Imprime línea 2
Explicación: **$2** es la columna 2 (separada por espacios). **NR==2** sería línea 2.

174.11 Práctica: Scripts Útiles

Script 1: Buscar archivo por edad

Script 2: Reportar errores en logs

Script 3: Backup selectivo

Listing 174.22 BASH

```
# Agregar línea antes de línea 5
sed '5i\NUEVA LÍNEA' archivo.txt                                    ①

# Agregar línea después de línea 5
sed '5a\NUEVA LÍNEA' archivo.txt                                    ②

# Agregar al final del archivo
sed '$a\ÚLTIMA LÍNEA' archivo.txt                                   ③
```

Listing 174.23 BASH

```
# Archivo: usuarios.txt
# diego  35  linux  admin
# carlos 28  python developer
# ana    32  java   engineer

# Imprimir columna 1 (nombre)
awk '{print $1}' usuarios.txt                                       ①

# Imprimir columnas 1 y 3
awk '{print $1, $3}' usuarios.txt                                   ②

# Imprimir en orden inverso
awk '{print $4, $1}' usuarios.txt                                   ③
```

174.12 Recursos Adicionales

- **Regular Expressions Tester**: https://regex101.com/
- **Sed Tutorial**: https://www.gnu.org/software/sed/manual/sed.html
- **AWK Manual**: https://www.gnu.org/software/gawk/manual/gawk.html
- **Find Command**: https://linux.die.net/man/1/find
- **Grep Patterns**: https://www.gnu.org/software/grep/manual/grep.html

174.13 Conclusión

Ahora sabes: Buscar texto con grep (simple y avanzado) Encontrar archivos con find Editar archivos con sed Procesar columnas con awk Combinar herramientas con pipes

Listing 174.24 BASH

```
# Archivo /etc/passwd usa : como delimitador
# root:x:0:0:root:/root:/bin/bash

# Ver solo nombres de usuario (-F = Field separator)
awk -F: '{print $1}' /etc/passwd                                    ①

# Ver UID y nombre
awk -F: '{print $3, $1}' /etc/passwd                                ②
```

Listing 174.25 BASH

```
# Mostrar solo usuarios con edad > 30
awk '$2 > 30 {print $1}' usuarios.txt                               ①

# Mostrar líneas que contengan "admin"
awk '/admin/ {print $0}' usuarios.txt                               ②

# Mostrar línea 2 solamente
awk 'NR==2' usuarios.txt                                            ③
```

Para Abacom: Estas 4 herramientas son **TU PAN DE CADA DÍA** en administración. Dóminalas.

Próximo paso: Úsalas en scripts de mantenimiento y análisis de logs.

Listing 174.26 BASH

```
# Archivo: precios.txt
# producto precio cantidad
# laptop   1000  5
# mouse    25    50
# monitor  300   10

# Calcular total por producto
awk '{print $1, $2 * $3}' precios.txt                                    ①

# Suma total de todo
awk '{suma += $2 * $3} END {print "Total: " suma}' precios.txt           ②
```

Listing 174.27 BASH

```
# Añadir texto personalizado
awk '{print "Usuario: " $1 " edad: " $2}' usuarios.txt                   ①

# Usar variables
awk -v nombre="Diego" '{print nombre ": " $1}' usuarios.txt              ②
```

Listing 174.28 BASH

```
#!/bin/bash
# Analizar errores en /var/log/syslog

echo "=== REPORTE DE ERRORES ==="
echo

# Total de errores
echo "Total de errores:"
grep -c "error" /var/log/syslog

# Por tipo de error
echo
echo "Errores por tipo:"
grep "error" /var/log/syslog | awk -F'] ' '{print $2}' | sort | uniq -c | sort -rn | head

# Últimas 10 líneas con error
echo
echo "Últimas 10 ocurrencias:"
grep "error" /var/log/syslog | tail -10
```

Listing 174.29 BASH

```bash
#!/bin/bash
# Eliminar logs más antiguos de 30 días

echo "Eliminando logs antiguos..."

# Encontrar y eliminar
find /var/log -type f -name "*.log.*" -mtime +30 -delete

# Comprimir logs en rotación
find /var/log -type f -name "*.log" -mtime +7 -exec gzip {} \;

echo "  Limpieza completada"
```

Listing 174.30 BASH

```bash
#!/bin/bash
# Procesar CSV de usuarios y crear cuentas

# usuarios.csv
# diego,35,developer,active
# carlos,28,admin,inactive
# ana,32,engineer,active

echo "Creando cuentas de usuario..."

awk -F, '$4 == "active" {
  print "sudo useradd -m -s /bin/bash " $1
}' usuarios.csv | bash

echo "  Usuarios creados"
```

Listing 174.31 BASH

```bash
#!/bin/bash
# Generar reporte de disco

echo "=== REPORTE DE USO DE DISCO ==="
echo

# Ver particiones >80% lleno
echo "Particiones críticas (>80%):"
df -h | awk '$5 > 80 {print $1 ": " $5 " usado"}'

echo
echo "Top 5 carpetas más grandes:"
du -sh /* | sort -rh | head -5

echo
echo "Archivos grandes (>100MB):"
find / -type f -size +100M -exec ls -lh {} \; | awk '{print $5, $9}' | sort -rh | head -1
```

Listing 174.32 BASH

```bash
#!/bin/bash
# Pipeline complejo: buscar → procesar → reportar

# ¿Cuáles usuarios iniciaron sesión hoy?
echo "Usuarios que iniciaron sesión hoy:"

grep "$(date '+%b %d')" /var/log/auth.log | \
  grep "Accepted password" | \
  awk '{print $1, $2, $3}' | \
  sort | uniq

# Desglose:
# 1. grep fecha actual de auth.log
# 2. grep solo líneas de "Accepted password"
# 3. awk extrae fecha y hora
# 4. sort ordena
# 5. uniq elimina duplicados
```

Listing 174.33 BASH

```
# MALO: . significa "cualquier carácter"
grep "archivo.txt" /var/log/syslog

# BUENO: \. es punto literal
grep "archivo\.txt" /var/log/syslog
```

Listing 174.34 BASH

```
# MALO: Si hay error, perdiste archivo
sed -i 's/viejo/nuevo/g' archivo.txt

# BUENO: Backup primero
sed -i.bak 's/viejo/nuevo/g' archivo.txt
```

Listing 174.35 BASH

```
# MALO: Se expande wildcard antes
awk $1 archivo.txt

# BUENO: Comillas protegen variables
awk '{print $1}' archivo.txt
```

Listing 174.36 BASH

```
#!/bin/bash
# Encontrar archivos sin modificar hace N días

DIAS=${1:-30}
echo "Buscando archivos no modificados en últimos $DIAS días..."
find /home -type f -mtime +$DIAS
```

Listing 174.37 BASH

```bash
#!/bin/bash
# Generar reporte de errores

find /var/log -name "*.log" | while read FILE
do
  COUNT=$(grep -c "error" "$FILE" 2>/dev/null || echo 0)
  if [ $COUNT -gt 0 ]
  then
    echo "$FILE: $COUNT errores"
  fi
done
```

Listing 174.38 BASH

```bash
#!/bin/bash
# Respaldar solo archivos modificados hoy

BACKUP_DIR="/backup/diario"
mkdir -p "$BACKUP_DIR"

find /home -type f -mtime 0 | while read FILE
do
  cp "$FILE" "$BACKUP_DIR/"
done

echo "  Respaldo completado: $(date)"
```

175 Anexo A: Bash Scripting - Mini-Curso Práctico

Listing 175.1 QUARTO-TITLE-BLOCK

176 Anexo A: Bash Scripting - Mini-Curso Práctico

176.1 Introducción

Bash (Bourne Again Shell) es el intérprete de comandos por defecto en Linux. Aunque puedes ejecutar comandos uno a uno en la terminal, crear **scripts de Bash** te permite automatizar tareas complejas. Este mini-curso es **completamente práctico** con ejemplos reales que usarás en Abacom.

Ver definiciones rapidas en: Glosario del curso

En este anexo aprenderás:

- Variables y tipos de datos en Bash
- Condicionales (if/else, case)
- Loops (for, while)
- Funciones y modularización
- Manejo de argumentos
- Flags y opciones (getopts)
- Redirección y pipes
- Debugging y limpieza (trap)
- Scripts prácticos listos para usar

Duración estimada: 2 horas de práctica

176.2 ¿Por Qué Aprender Bash?

En administración de servidores Linux, **el 80% de tu trabajo es automatizar tareas**. Sin Bash, ejecutarías comandos manualmente:

Con Bash, lo automatizas:

① **#!/bin/bash** es el "shebang" - le dice al SO que execute este archivo con Bash

Beneficios:

- Una línea en vez de 4 pasos manuales
- Reutilizable en múltiples máquinas
- Menos errores humanos

Listing 176.1 BASH

```
# Manual (tedioso)
apt update
apt upgrade -y
apt install nginx
systemctl enable nginx
systemctl start nginx
# ... repetir en 20 servidores
```

Listing 176.2 BASH

```
#!/bin/bash                                                    ①
# Instalar y configurar nginx en un script

apt update && apt upgrade -y
apt install -y nginx
systemctl enable nginx
systemctl start nginx
```

- Documentación del proceso

176.3 Scripting en Diferentes SOs

176.3.1 Bash (Linux/macOS)

Dónde funciona:

- Linux (Bash nativo)
- macOS (Bash/Zsh - compatible)
- Windows (requiere WSL o Git Bash)

176.3.2 PowerShell (Windows)

Dónde funciona:

- Windows (PowerShell nativo)
- Linux/macOS (PowerShell 7+ disponible)

Listing 176.3 BASH

```bash
#!/bin/bash
# Script nativo en Linux y macOS

# Variables
SERVIDOR="abacom-prod"
PUERTO=8080

# Función
setup_server() {
    echo "Configurando $SERVIDOR en puerto $PUERTO"
    systemctl restart nginx
    netstat -tulpn | grep $PUERTO
}

# Loop
for i in {1..5}; do
    echo "Intento $i"
    setup_server && break
done

# Ejecutar
chmod +x script.sh
./script.sh
```

176.3.3 Python (Multi-plataforma)

Dónde funciona:

- Linux (Python 3)
- macOS (Python 3)
- Windows (Python 3)

176.3.4 Node.js (Multi-plataforma)

Dónde funciona:

- Linux (Node.js)
- macOS (Node.js)
- Windows (Node.js)

Recomendación para Abacom:

- Use **Bash** para scripts Linux-only (producción)
- Use **Python** para herramientas multi-SO
- Use **PowerShell** solo si necesita automatizar Windows

Listing 176.4 POWERSHELL

```
# Equivalente en Windows PowerShell

# Variables
$Servidor = "abacom-prod"
$Puerto = 8080

# Función
function Setup-Server {
    param([string]$Server)
    Write-Host "Configurando $Server en puerto $Puerto"
    Get-NetTCPConnection -LocalPort $Puerto | Select-Object State
}

# Loop
for ($i = 1; $i -le 5; $i++) {
    Write-Host "Intento $i"
    Setup-Server -Server $Servidor
    if ($?) { break }
}

# Ejecutar
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
.\script.ps1
```

176.4 Concepto 1: Variables

176.4.1 Declarar Variables

① **NOMBRE=“Diego”** asigna el valor “Diego” a la variable NOMBRE (sin espacios alrededor de =)

Reglas en Bash:

- Sin espacios alrededor del = (correcto: **VAR=valor**, incorrecto: **VAR = valor**)
- Nombres en MAYÚSCULAS por convención
- Pueden contener letras, números, guiones bajos
- Acceder con `$NOMBRE` o `${NOMBRE}`

176.4.2 Usar Variables

① **echo** imprime texto. `$NOMBRE` reemplaza con valor almacenado

Salida esperada:

```
Hola, Diego
```

176.4.3 Variables de Entrada

① **read NOMBRE** espera entrada del usuario y la almacena en NOMBRE

Ejecución:

176.4.4 Variables Especiales

① **$0** = nombre del script
② **$1** = primer argumento pasado al script
③ **$@** = todos los argumentos
④ **$?** = código de salida del comando anterior (0=éxito, >0=error)
⑤ **$$** = PID (Process ID) del script actual

Ejemplo:

Ejecución:

176.5 Concepto 2: Condicionales (if/else)

176.5.1 if Básico

① `[ $EDAD -gt 18 ]` verifica si EDAD es MAYOR QUE 18 (-gt = "greater than")

Operadores de comparación numérica:

- **-eq** = igual (equal)
- **-ne** = no igual (not equal)
- **-lt** = menor que (less than)
- **-le** = menor o igual
- **-gt** = mayor que (greater than)
- **-ge** = mayor o igual

176.5.2 if/else

① `[ "$SISTEMA" = "Linux" ]` compara strings (notar comillas alrededor de $SISTEMA)

Operadores de string:

- **=** = igual
- **!=** = no igual
- **-z** = string vacío
- **-n** = string no vacío

176.5.3 if/elif/else

① **$1** es el primer argumento pasado al script

176.5.4 Verificar Archivos

① [**-f $FILE**] verifica si el archivo existe y es un archivo regular

Operadores de archivo:

- **-f** = es archivo regular
- **-d** = es directorio
- **-e** = existe (archivo o directorio)
- **-r** = legible (readable)
- **-w** = escribible (writable)
- **-x** = ejecutable

176.6 Concepto 2.5: case (múltiples opciones)

Cuando tienes muchas opciones (distro, entorno, modo de ejecución), `case` suele ser más claro que varios `elif`.

① **"${1:-}"** evita errores si no pasan argumentos; deja DISTRO vacío si falta
② **case "$DISTRO" in** evalúa alternativas y ejecuta el bloque que coincida

---

176.7 Concepto 2.6: Flags y opciones con getopts

`getopts` permite parsear opciones tipo `-u diego -g developers` sin depender del orden.

① **set -euo pipefail** hace que el script falle temprano ante errores comunes
② **USUARIO=“ “** inicializa variable para validar más adelante
③ **GRUPO=“ “** idem; lo exigimos como parámetro obligatorio
④ **SHELL=“/bin/bash”** define valor por defecto si no pasan `-s`
⑤ **getopts “:u:g:s:h”** define flags; `u`, `g`, `s` esperan argumento y `h` muestra ayuda
⑥ **-z** valida que los campos obligatorios no estén vacíos
⑦ **echo** aquí simula la acción; en producción reemplaza por `useradd`, `usermod`, etc.

176.8 Concepto 3: Loops (for, while)

176.8.1 Loop for con Lista

① **for SERVIDOR in** itera sobre cada valor en SERVIDORES

Salida:

```
Reiniciando web-1
Reiniciando web-2
Reiniciando web-3
```

176.8.2 Loop for con Rango

① **{1..5}** genera rango del 1 al 5

Alternativa (sintaxis C):

① Sintaxis similar a C: inicialización, condición, incremento

176.8.3 Loop while

① **while** [**condición**] ejecuta mientras sea verdadera
② **$((expresión))** realiza aritmética en Bash

Salida:

```
Iteración 1
Iteración 2
Iteración 3
```

176.8.4 Iterar sobre Archivos

① ***.log** es wildcard que expande a todos los archivos .log

---

176.9 Concepto 4: Funciones

176.9.1 Función Básica

① **mostrar_fecha()** define una función sin parámetros
② **$(date ...)** ejecuta comando y captura resultado
③ **mostrar_fecha** llama la función

Salida:

```
Fecha actual: 2024-01-29
```

176.9.2 Función con Parámetros

① **sumar()** define función que recibirá 2 argumentos
② **local VAR** declara variable local a la función
③ **$(sumar 10 20)** captura el resultado de la función

Salida:

```
10 + 20 = 30
```

176.9.3 Función con Retorno

① **id "usuario"** verifica si el usuario existe. **&>/dev/null** silencia la salida
② **return 0** = éxito
③ **return 1** = error

176.10 Concepto 5: Redirección y Pipes

176.10.1 Redirección de Salida

① **>** redirecciona salida a archivo (sobrescribe si existe)
② **»** redirecciona salida a archivo (agrega al final)

176.10.2 Redirección de Error

① **2>** redirecciona stderr (error standard) a archivo

Combinaciones útiles:

- **> archivo** = redirige stdout (salida normal)
- **2> archivo** = redirige stderr (errores)
- **&> archivo** = redirige ambos
- **2>&1** = redirige stderr a stdout

176.10.3 Pipes (|)

① **|** (pipe) envía salida del primer comando como entrada al segundo

Ejemplo práctico:

① **cat** muestra archivo → **grep** filtra líneas con "error" → **wc -l** cuenta líneas

---

176.11 Concepto 5.5: Debugging y limpieza con trap

En producción, un buen script hace dos cosas bien: (1) falla de forma clara y (2) deja el sistema limpio (por ejemplo, borra archivos temporales).

176.11.1 trap (cleanup al salir)

① **set -euo pipefail** reduce fallas silenciosas y evita variables indefinidas
② **mktemp** crea un archivo temporal con nombre seguro
③ **cleanup()** define la limpieza que quieres garantizar
④ **trap ... EXIT** ejecuta cleanup al terminar el script (exito o error)

176.11.2 Debug rapido (ver lo que ejecuta)

① **bash -x** imprime cada comando antes de ejecutarlo (muy util para depurar)

176.12 Ejemplos Prácticos Reales

176.12.1 Ejemplo 1: Script para Monitorear Disk Usage

① **UMBRAL=80** define límite del 80%
② **$(df | ...)** captura todas las particiones
③ **awk** extrae el porcentaje de uso
④ `if [ "$USO" -gt "$UMBRAL" ]` alerta si supera umbral

Guardar como: check_disk.sh

Uso:

176.12.2 Ejemplo 2: Script para Respaldar Directorio

① **$1** primer argumento es carpeta a respaldar
② **FECHA** captura fecha/hora para nombre único
③ **-z** verifica si string está vacío
④ **basename** extrae solo el nombre sin ruta
⑤ **tar -czf** comprime carpeta

Uso:

176.12.3 Ejemplo 3: Script para Actualizar Sistema (Debian/Ubuntu)

① **apt update** actualiza lista de paquetes
② **if [$? -ne 0]** verifica si comando anterior falló
③ **apt upgrade -y** actualiza paquetes (confirmación automática)
④ **apt autoremove** elimina paquetes huérfanos

176.12.4 Ejemplo 4: Script para Crear Usuarios en Lote

① **getent group** verifica si el grupo existe
② **for USUARIO in** itera sobre cada usuario
③ **useradd -m** crea usuario con directorio home (-s especifica shell)

176.13 Construcción de un Script Profesional

176.13.1 Template Completo

① **set -euo pipefail** = salir si error, no permitir variables indefinidas, fallar en pipes
② **readonly** = variable inmutable
③ **`(@)`** = array (múltiples valores)
④ **log()** función para mensajes con timestamp
⑤ **verificar_permisos()** asegurar que es root
⑥ **`"${CARPETAS[@]}"`** itera sobre elementos del array

176.14 Errores Comunes en Bash

Error 1: Sin comillas alrededor de variables

Error 2: Espacios alrededor de =

Error 3: = vs -eq para números

Error 4: Olvidar fi, done, etc

176.15 Tabla de Referencia Rápida

Operador	Uso	Ejemplo
-eq	Igual (números)	`[ $A -eq $B ]`
-ne	No igual	`[ $A -ne $B ]`
-lt	Menor que	`[ $A -lt $B ]`
-gt	Mayor que	`[ $A -gt $B ]`
=	Igual (strings)	`[ "$S" = "hola" ]`
!=	No igual (strings)	`[ "$S" != "hola" ]`
-f	Es archivo	`[ -f /ruta/archivo ]`
-d	Es directorio	`[ -d /ruta/dir ]`
-z	String vacío	`[ -z "$VAR" ]`
>	Redir salida	`echo "hola" > file.txt`
»	Agregar a archivo	`echo "hola" >> file.txt`
|	Pipe (tubería)	`cat file | grep "error"`

176.16 Quiz: Verificar Comprensión

Pregunta 1: Variables

¿Cuál es el error en este código?

Listing 176.45 BASH

```
NOMBRE = Diego
echo $NOMBRE
```

a) Falta el signo $ en la asignación
b) Los espacios alrededor de = causan error de sintaxis (Correcto)
c) echo necesita comillas
d) No hay error
Explicación: Bash es estricto con espacios. **NOMBRE = Diego** intenta ejecutar comando **NOMBRE**, no asignar variable.

Pregunta 2: Condicionales

¿Qué operador compara STRINGS en Bash?
a) -eq
b) = (Correcto)
c) -lt
d) -gt
Explicación: = compara strings. **-eq** compara números. Usar incorrecto causa comparación inesperada.

Pregunta 3: Loops

¿Cuál es la diferencia entre **for** y **while**?
a) No hay diferencia, son iguales
b) for itera sobre una lista, **while** ejecuta mientras condición sea verdadera (Correcto)
c) for es más rápido
d) while solo funciona con números
Explicación: **for** itera conociendo fin. **while** evalúa condición cada iteración.

176.17 Práctica: Crear tu Primer Script

Ejercicio 1: Script de Saludo

Ejercicio 2: Script de Números Pares

Ejercicio 3: Script para Contar Líneas

176.18 Recursos Adicionales

- **Bash Manual Oficial**: https://www.gnu.org/software/bash/manual/
- **ShellCheck (Validador)**: https://www.shellcheck.net/
- **Google Shell Style Guide**: https://google.github.io/styleguide/shellguide.html
- **Bash Pitfalls**: https://mywiki.wooledge.org/BashPitfalls

---

176.19 Conclusión

Ahora sabes: Variables, tipos y conversiones Condicionales (if/else/elif) Loops (for/while) Funciones y modularización Redirección y pipes Scripts profesionales listos para usar

Próximo paso: Usa estos conocimientos en los scripts que encontrarás en configuraciones de sistemas.

Listing 176.5 PYTHON

```
#!/usr/bin/env python3
# Mejor alternativa para scripts portables

import subprocess
import sys

# Variables
SERVIDOR = "abacom-prod"
PUERTO = 8080

def setup_server(server):
    """Configurar servidor"""
    print(f"Configurando {server} en puerto {PUERTO}")
    try:
        result = subprocess.run(
            ["systemctl", "restart", "nginx"],
            capture_output=True
        )
        return result.returncode == 0
    except Exception as e:
        print(f"Error: {e}")
        return False

# Loop
for i in range(1, 6):
    print(f"Intento {i}")
    if setup_server(SERVIDOR):
        break

# Ejecutar
# chmod +x script.py
# ./script.py
```

Listing 176.6 JAVASCRIPT

```
#!/usr/bin/env node
// Alternativa moderna en JavaScript

const { exec } = require('child_process');
const { promisify } = require('util');
const execAsync = promisify(exec);

const SERVIDOR = "abacom-prod";
const PUERTO = 8080;

async function setupServer(server) {
    console.log(`Configurando ${server} en puerto ${PUERTO}`);
    try {
        const { stdout } = await execAsync('systemctl restart nginx');
        return true;
    } catch (error) {
        console.error(`Error: ${error}`);
        return false;
    }
}

// Main
(async () => {
    for (let i = 1; i <= 5; i++) {
        console.log(`Intento ${i}`);
        if (await setupServer(SERVIDOR)) break;
    }
})();

// Ejecutar
// chmod +x script.js
// node script.js
```

Listing 176.7 BASH

```
NOMBRE="Diego"                                                    ①
EDAD=35
SERVIDOR="web-prod-01"
```

Listing 176.8 BASH

```
echo "Hola, $NOMBRE"                                              ①
```

Listing 176.9 BASH

```
#!/bin/bash
echo "¿Cuál es tu nombre?"
read NOMBRE                                                    (1)
echo "Bienvenido, $NOMBRE"
```

Listing 176.10 BASH

```
$ bash script.sh
¿Cuál es tu nombre?
Diego  ← Usuario escribe aquí
Bienvenido, Diego
```

Listing 176.11 BASH

```
echo $0                                                        (1)
echo $1                                                        (2)
echo $@                                                        (3)
echo $?                                                        (4)
echo $$                                                        (5)
```

Listing 176.12 BASH

```
#!/bin/bash
echo "Script: $0"
echo "Primer arg: $1"
echo "Todos los args: $@"
```

Listing 176.13 BASH

```
$ bash script.sh diego 35 linux
Script: script.sh
Primer arg: diego
Todos los args: diego 35 linux
```

Listing 176.14 BASH

```
#!/bin/bash
EDAD=35

if [ $EDAD -gt 18 ]                                            (1)
then
  echo "Eres adulto"
fi
```

Listing 176.15 BASH

```
#!/bin/bash
SISTEMA="Linux"

if [ "$SISTEMA" = "Linux" ]                                    (1)
then
  echo "Sistema: Linux"
else
  echo "No es Linux"
fi
```

Listing 176.16 BASH

```
#!/bin/bash
DISTRO=$1                                                      (1)

if [ "$DISTRO" = "ubuntu" ]
then
  echo "Package Manager: apt"
elif [ "$DISTRO" = "centos" ]
then
  echo "Package Manager: yum/dnf"
elif [ "$DISTRO" = "arch" ]
then
  echo "Package Manager: pacman"
else
  echo "Distribución desconocida"
fi
```

Listing 176.17 BASH

```
#!/bin/bash
CONFIG_FILE="/etc/nginx/nginx.conf"

if [ -f "$CONFIG_FILE" ]                                       (1)
then
  echo "Archivo existe"
else
  echo "Archivo no encontrado"
fi
```

Listing 176.18 BASH

```
#!/bin/bash

DISTRO="${1:-}"                                                    ①

case "$DISTRO" in                                                  ②
  ubuntu|debian)
    echo "Package manager: apt"
    ;;
  centos|rocky|rhel)
    echo "Package manager: dnf"
    ;;
  arch)
    echo "Package manager: pacman"
    ;;
  "")
    echo "Uso: $0 <distro>" >&2
    echo "Ejemplo: $0 ubuntu" >&2
    exit 2
    ;;
  *)
    echo "Distribución no soportada: $DISTRO" >&2
    exit 2
    ;;
esac
```

Listing 176.19 BASH

```bash
#!/bin/bash

set -euo pipefail                                                    ①

USUARIO=""                                                           ②
GRUPO=""                                                             ③
SHELL="/bin/bash"                                                    ④

while getopts ":u:g:s:h" opt; do                                     ⑤
  case "$opt" in
    u) USUARIO="$OPTARG" ;;
    g) GRUPO="$OPTARG" ;;
    s) SHELL="$OPTARG" ;;
    h)
      echo "Uso: $0 -u <usuario> -g <grupo> [-s <shell>]" >&2
      exit 0
      ;;
    :)
      echo "Falta valor para -$OPTARG" >&2
      exit 2
      ;;
    \?)
      echo "Opción inválida: -$OPTARG" >&2
      exit 2
      ;;
  esac
done

if [ -z "$USUARIO" ] || [ -z "$GRUPO" ]                              ⑥
then
  echo "Uso: $0 -u <usuario> -g <grupo> [-s <shell>]" >&2
  exit 2
fi

echo "OK: usuario='$USUARIO' grupo='$GRUPO' shell='$SHELL'"          ⑦
```

Listing 176.20 BASH

```bash
#!/bin/bash
SERVIDORES="web-1 web-2 web-3"

for SERVIDOR in $SERVIDORES                                          ①
do
  echo "Reinsticiando $SERVIDOR"
done
```

Listing 176.21 BASH

```
#!/bin/bash

for i in {1..5}                                                    (1)
do
  echo "Número: $i"
done
```

Listing 176.22 BASH

```
for ((i=1; i<=5; i++))                                             (1)
do
  echo "Número: $i"
done
```

Listing 176.23 BASH

```
#!/bin/bash
CONTADOR=1

while [ $CONTADOR -le 3 ]                                          (1)
do
  echo "Iteración $CONTADOR"
  CONTADOR=$((CONTADOR + 1))                                       (2)
done
```

Listing 176.24 BASH

```
#!/bin/bash

for ARCHIVO in /var/log/*.log                                      (1)
do
  echo "Procesando: $ARCHIVO"
done
```

Listing 176.25 BASH

```
#!/bin/bash

mostrar_fecha() {                                                  (1)
  echo "Fecha actual: $(date '+%Y-%m-%d')"                         (2)
}

mostrar_fecha                                                      (3)
```

Listing 176.26 BASH

```bash
#!/bin/bash

sumar() {                                                    ①
  local NUM1=$1                                              ②
  local NUM2=$2
  local SUMA=$((NUM1 + NUM2))
  echo $SUMA
}

RESULTADO=$(sumar 10 20)                                     ③
echo "10 + 20 = $RESULTADO"
```

Listing 176.27 BASH

```bash
#!/bin/bash

verificar_usuario() {
  local USUARIO=$1
  if id "$USUARIO" &>/dev/null                               ①
  then
    return 0                                                 ②
  else
    return 1                                                 ③
  fi
}

if verificar_usuario "diego"
then
  echo "Usuario existe"
else
  echo "Usuario no existe"
fi
```

Listing 176.28 BASH

```bash
#!/bin/bash

echo "Salida a archivo" > output.txt                         ①
echo "Agregar línea" >> output.txt                           ②
```

Listing 176.29 BASH

```
#!/bin/bash

ls /directorio/inexistente 2> error.log ①
```

Listing 176.30 BASH

```
#!/bin/bash

ps aux | grep nginx ①
```

Listing 176.31 BASH

```
cat /var/log/syslog | grep "error" | wc -l ①
```

Listing 176.32 BASH

```
#!/bin/bash

set -euo pipefail ①

tmp_file="$(mktemp)" ②

cleanup() { ③
  rm -f "$tmp_file"
}

trap cleanup EXIT ④

echo "Archivo temporal: $tmp_file"
echo "hola" > "$tmp_file"
cat "$tmp_file"
```

Listing 176.33 BASH

```
$ bash -x script.sh ①
+ mktemp
+ echo "Archivo temporal: ..."
```

Listing 176.34 BASH

```
#!/bin/bash
# Script para alertar si disco está casi lleno

UMBRAL=80                                                                    ①

for PARTICION in $(df | tail -n +2 | awk '{print $6}')                       ②
do
  USO=$(df "$PARTICION" | tail -1 | awk '{print $(NF-1)}' | sed 's/%//') ③

  if [ "$USO" -gt "$UMBRAL" ]                                                ④
  then
    echo "ALERTA: $PARTICION está al ${USO}% lleno"
  fi
done
```

Listing 176.35 BASH

```
bash check_disk.sh
# Salida si hay alerta:
# ALERTA: / está al 85% lleno
```

Listing 176.36 BASH

```
#!/bin/bash
# Script para respaldar carpeta con timestamp

CARPETA_ORIGEN=$1                                                            ①
CARPETA_BACKUP="/backup"
FECHA=$(date '+%Y%m%d_%H%M%S')                                               ②

if [ -z "$CARPETA_ORIGEN" ]                                                  ③
then
  echo "Uso: $0 /ruta/a/respaldar"
  exit 1
fi

NOMBRE_BACKUP="${CARPETA_BACKUP}/backup_$(basename $CARPETA_ORIGEN)_${FECHA}.tar.gz" ④

tar -czf "$NOMBRE_BACKUP" "$CARPETA_ORIGEN"                                  ⑤

echo "Respaldo completado: $NOMBRE_BACKUP"
```

Listing 176.37 BASH

```
bash respaldar.sh /home/diego
# Salida:
# Respaldo completado: /backup/backup_diego_20240129_143022.tar.gz
```

Listing 176.38 BASH

```
#!/bin/bash
# Script para actualizar sistema de forma segura

echo "=== Iniciando actualización del sistema ==="

# Actualizar lista de paquetes
apt update                                                    ①
if [ $? -ne 0 ]                                               ②
then
  echo "Error al actualizar repositorios"
  exit 1
fi

# Actualizar paquetes
apt upgrade -y                                                ③
if [ $? -ne 0 ]
then
  echo "Error durante apt upgrade"
  exit 1
fi

# Eliminar paquetes sin usar
apt autoremove -y                                             ④

echo "=== Actualización completada ==="

# Información del sistema
echo "Versión del kernel: $(uname -r)"
echo "Última actualización: $(date)"
```

Listing 176.39 BASH

```
#!/bin/bash
# Script para crear múltiples usuarios

USUARIOS="diego carlos ana benjamin"
GRUPO="developers"

# Crear grupo si no existe
if ! getent group "$GRUPO" > /dev/null 2>&1                    ①
then
  groupadd "$GRUPO"
  echo "Grupo $GRUPO creado"
fi

# Crear cada usuario
for USUARIO in $USUARIOS                                       ②
do
  if id "$USUARIO" &>/dev/null
  then
    echo "Usuario $USUARIO ya existe"
  else
    useradd -m -s /bin/bash -G "$GRUPO" "$USUARIO"             ③
    echo "Usuario $USUARIO creado"
  fi
done

echo "Usuarios en grupo $GRUPO:"
getent group "$GRUPO"
```

Listing 176.40 BASH

```bash
#!/bin/bash
#
# Nombre: backup_sistema.sh
# Descripción: Respalda directorios críticos
# Autor: Diego Saavedra
# Fecha: 2024-01-29
# Versión: 1.0
#

set -euo pipefail                                                    ①

# ============ CONSTANTES ============
readonly SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"  ②
readonly LOG_FILE="/var/log/backup.log"
readonly BACKUP_DIR="/backup"
readonly CARPETAS=("/home" "/etc" "/var/www")                        ③

# ============ FUNCIONES ============

log() {                                                              ④
  local NIVEL=$1
  shift
  local MENSAJE="$@"
  local TIMESTAMP=$(date '+%Y-%m-%d %H:%M:%S')

  echo "[$TIMESTAMP] [$NIVEL] $MENSAJE" | tee -a "$LOG_FILE"
}

verificar_permisos() {                                               ⑤
  if [ "$EUID" -ne 0 ]
  then
    log "ERROR" "Este script debe ejecutarse como root"
    exit 1
  fi
}

crear_respaldo() {
  local ORIGEN=$1
  local NOMBRE=$(basename "$ORIGEN")
  local FECHA=$(date '+%Y%m%d_%H%M%S')
  local ARCHIVO="${BACKUP_DIR}/${NOMBRE}_${FECHA}.tar.gz"

  log "INFO" "Creando respaldo de $ORIGEN"

  tar -czf "$ARCHIVO" "$ORIGEN" 2>/dev/null

  if [ -f "$ARCHIVO" ]
  then
    log "OK" "Respaldo creado: $ARCHIVO ($(du -h $ARCHIVO | cut -f1))"
  else
    log "ERROR" "Error creando respaldo de $ORIGEN"
  fi
}
```

Listing 176.41 BASH

```
# MALO
if [ $NOMBRE = "Diego" ]  # Falla si $NOMBRE está vacío

# BUENO
if [ "$NOMBRE" = "Diego" ]  # Funciona incluso si vacío
```

Listing 176.42 BASH

```
# MALO
VAR = valor  # Error de sintaxis

# BUENO
VAR=valor
```

Listing 176.43 BASH

```
# MALO (compara como strings)
if [ "10" = "2" ]  # Retorna verdadero

# BUENO (compara como números)
if [ 10 -gt 2 ]  # Retorna verdadero
```

Listing 176.44 BASH

```
# MALO
if [ $AGE -gt 18 ]
then
  echo "Mayor"
# Falta fi

# BUENO
if [ $AGE -gt 18 ]
then
  echo "Mayor"
fi
```

Listing 176.46 BASH

```bash
#!/bin/bash
# Crea un script que:
# 1. Pida nombre al usuario (read)
# 2. Verifique que no está vacío
# 3. Imprima saludo personalizado

# SOLUCIÓN:
#!/bin/bash
echo "¿Cuál es tu nombre?"
read NOMBRE

if [ -z "$NOMBRE" ]
then
  echo "Error: Nombre no puede estar vacío"
  exit 1
fi

echo "¡Hola, $NOMBRE! Bienvenido a Bash"
```

Listing 176.47 BASH

```bash
#!/bin/bash
# Crea un script que imprima números pares del 1 al 20

# SOLUCIÓN:
#!/bin/bash
for i in {1..20}
do
  if [ $((i % 2)) -eq 0 ]
  then
    echo "$i"
  fi
done
```

Listing 176.48 BASH

```bash
#!/bin/bash
# Crea un script que:
# 1. Reciba ruta a archivo como argumento
# 2. Verifique que existe
# 3. Cuente líneas

# SOLUCIÓN:
#!/bin/bash
if [ -z "$1" ]
then
  echo "Uso: $0 /ruta/archivo"
  exit 1
fi

if [ ! -f "$1" ]
then
  echo "Error: Archivo no existe"
  exit 1
fi

LINEAS=$(wc -l < "$1")
echo "El archivo tiene $LINEAS líneas"
```

Part XX

Anexos: Mini-Cursos (Nivel 3: Avanzado)

177 Anexo G: OpenCode - Mini-Curso para Gestion de Servidores

Listing 177.1 `QUARTO-TITLE-BLOCK`

Flujos de trabajo seguros para diagnosticar, automatizar y documentar cambios

178 Anexo G: OpenCode - Mini-Curso para Gestion de Servidores

178.1 Introduccion

OpenCode es un asistente interactivo para tareas de ingenieria de software. En gestion de servidores, su valor aparece cuando necesitas:

- diagnosticar incidentes con evidencia (logs, metricas, configuraciones)
- generar procedimientos repetibles (runbooks)
- automatizar tareas con scripts (Bash/PowerShell) y herramientas de provisioning
- reducir errores humanos (checklists, validaciones, comandos con modo seguro)

Ver guia completa de configuracion en: **SETUP.qmd**

Tiempo estimado: 45-60 minutos

178.2 Objetivos de aprendizaje

Al finalizar este mini-curso podras:

- recolectar evidencia basica del sistema sin interrumpir el servicio
- pedir a OpenCode un plan de diagnostico paso a paso (y verificarlo)
- generar comandos seguros (read-only primero, luego cambios)
- documentar y estandarizar una correccion en un runbook
- crear una automatizacion pequena (script) para tareas repetidas

---

178.3 Reglas de seguridad (antes de usar AI)

⚠ ADVERTENCIA CRITICA

Nunca pegues secretos en OpenCode.
Lo que podria salir mal: - Filtras credenciales (tokens, claves privadas, passwords) en historiales, logs o capturas. - Compartes informacion sensible (IPs publicas, nombres internos, rutas) que se usa para ataques.

Como prevenirlo: 1. Redacta secretos antes de pegar texto (reemplaza por `REDACTED`). 2. Comparte solo el minimo necesario (1-2 comandos, 20-50 lineas de log, el error completo). 3. Prefiere comandos de lectura primero (`status`, `show`, `--dry-run`).

178.4 Flujo recomendado: evidencia -> analisis -> accion

1. **Evidencia:** captura estado del sistema, servicio y logs (sin modificar nada).
2. **Analisis con OpenCode:** pega evidencia y pide hipotesis + plan de verificacion.
3. **Accion controlada:** ejecuta cambios pequenos, reversibles y con verificacion.
4. **Documentacion:** convierte lo aprendido en runbook (pasos, comandos, rollback).

178.5 Instalacion de OpenCode

OpenCode se puede instalar como binario (script), como paquete Node, o via Homebrew.

178.5.1 Linux/macOS (script)

Listing 178.1 BASH

```
$ curl -fsSL https://opencode.ai/install | bash                    ①
```

① **curl ... | bash** instala el binario de OpenCode usando el instalador oficial.

178.5.2 Node.js (npm/pnpm/bun)

Listing 178.2 BASH

```
$ npm install -g opencode-ai                                       ①
$ pnpm install -g opencode-ai                                      ②
$ bun install -g opencode-ai                                       ③
                                                                   ①
                                                                   ②
                                                                   ③
```

① **npm install -g** instala OpenCode globalmente.
② **pnpm install -g** instala OpenCode globalmente usando pnpm.
③ **bun install -g** instala OpenCode globalmente usando bun.

178.5.3 macOS/Linux (Homebrew)

Listing 178.3 BASH

```
$ brew install anomalyco/tap/opencode                                    ①
```

① **brew install anomalyco/tap/opencode** instala la version mas actual desde el tap oficial.

178.5.4 Windows

Listing 178.4 POWERSHELL

```
PS> choco install opencode                                               ①
PS> scoop install opencode                                               ②
PS> npm install -g opencode-ai                                           ③
                                                                         ①
                                                                         ②
                                                                         ③
```

① **choco install** instala OpenCode con Chocolatey.
② **scoop install** instala OpenCode con Scoop.
③ **npm install -g** instala OpenCode via Node.js.

178.6 Configurar proveedor (LLM)

OpenCode requiere un proveedor/modelo (y sus credenciales) para funcionar.

En la TUI, el camino mas simple es:

Listing 178.5 TEXT

```
/connect # <1>
 # <1>
```

① **/connect** abre el flujo para configurar un proveedor (por ejemplo, OpenCode Zen u otro provider).

178.7 Inicializar un proyecto (/init)

Entra a tu repo/proyecto y levanta OpenCode:

Listing 178.6 BASH

```
$ cd /path/to/project                                                    ①
$ opencode                                                               ②
                                                                         ①
                                                                         ②
```

---

① **cd** cambia al proyecto que vas a administrar.
② **opencode** inicia la interfaz TUI.

Luego, inicializa el proyecto:

Listing 178.7 TEXT

```
/init # <1>
 # <1>
```

---

① **/init** analiza el proyecto y crea `AGENTS.md` en la raiz.

Recomendacion:

- Commitea `AGENTS.md` en Git. Es parte del "contrato" del proyecto para que OpenCode entienda estructura, convenciones y limites.

178.8 Modos de trabajo: Plan vs Build

OpenCode tiene dos modos:

- **Plan mode:** el agente propone un plan sin tocar archivos.
- **Build mode:** el agente ejecuta cambios (edita/crea archivos).

Se alterna con **Tab** (veras el indicador en la esquina inferior derecha).

Uso recomendado para servidores:

1. Plan mode: pedir diagnostico, hipotesis, comandos read-only, verificacion.
2. Build mode: solo cuando el plan es correcto y el cambio es reversible.

178.9 Deshacer y rehacer cambios (/undo, /redo)

Si OpenCode hizo cambios que no quieres:

Listing 178.8 TEXT

```
/undo # <1>
/redo # <2>
 # <1>
 # <2>
```

① **/undo** revierte los cambios del ultimo pedido.
② **/redo** reaplica los cambios deshechos.

178.10 Compartir sesiones (/share) y politica de privacidad

Compartir es una accion explicita. Al usar `/share`, OpenCode sincroniza el historial a servidores para generar un link publico.

Listing 178.9 TEXT

```
/share # <1>
/unshare # <2>
 # <1>
 # <2>
```

① **/share** publica un link de la conversacion.
② **/unshare** elimina el link y deja la conversacion sin acceso publico.

Para ambientes sensibles (empresas / infra / clientes), deshabilita el share por proyecto:

Listing 178.10 TEXT

```
{ "$schema": "https://opncd.ai/config.json", "share": "disabled" } # <1>
```

① **share: disabled** desactiva el compartir (recomendado en proyectos con informacion sensible).

178.11 Ejemplo 1: Inventario rapido del servidor (multi-SO)

178.11.1 Linux

Listing 178.11 BASH

```
$ uname -m                                                              ①
x86_64

$ free -h                                                               ②
               total        used        free      shared  buff/cache   available
Mem:            15Gi       4.2Gi       8.2Gi       412Mi       3.1Gi        10Gi
Swap:          2.0Gi          0B       2.0Gi

$ df -h /                                                               ③
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1        80G   22G   55G  29% /
                                                                        ①
                                                                        ②
                                                                        ③
```

① **uname -m** confirma la arquitectura (util para binarios y contenedores)
② **free -h** muestra uso de RAM en formato legible
③ **df -h /** revisa espacio del filesystem principal

178.11.2 macOS

① **system_profiler SPHardwareDataType** entrega un resumen del hardware
② **df -h /** valida uso de disco (util antes de builds, logs o backups)

178.11.3 Windows

① **systeminfo** muestra inventario basico (CPU, tipo de sistema, RAM)
② **Get-PSDrive** revisa espacio disponible por unidad

Como usarlo con OpenCode:

- Pega la salida y pregunta: “Con estos datos, que riesgos ves (disco, RAM, arquitectura) para desplegar X? Dame una checklist de preflight”.

Listing 178.12 BASH

```
$ system_profiler SPHardwareDataType                                    ①

Hardware Overview:
  Model Name: MacBook Pro
  Model Identifier: MacBookPro18,3
  Processor Name: Apple M3 Pro
  Number of Cores: 12 core CPU
  Memory: 16 GB

$ df -h /                                                               ②
Filesystem      Size   Used  Avail Capacity iused      ifree %iused  Mounted on
/dev/disk3s1   460Gi  210Gi  245Gi    47%  1.9M       2.4G    0%    /
                                                                        ①
                                                                        ②
```

178.12 Ejemplo 2: Diagnostico de un servicio (Nginx) en Linux

① **systemctl status nginx** –**no-pager** valida estado del servicio sin paginador
② **journalctl -u nginx** –**since** obtiene errores recientes del servicio
③ **nginx -t** valida sintaxis de configuracion antes de recargar

Prompt sugerido para OpenCode (pega 10-30 lineas reales del log):

- “Analiza este error de Nginx y propon 3 hipotesis ordenadas por probabilidad. Para cada una, dame un comando de verificacion (solo lectura) y el criterio de confirmacion”.

178.13 Ejemplo 2.1: Checklist de preflight (antes de tocar produccion)

Objetivo: pedir a OpenCode una checklist corta, ejecutable y segura.

Prompt sugerido (Plan mode):

① **Checklist de preflight** fuerza a que el resultado sea accionable y verificable (no solo consejos).

Listing 178.13 POWERSHELL

```
PS> systeminfo                                                    ①

Computer Name:           USUARIO-PC
Processor(s):            1 Logical Processor
System Type:             x86-64-based PC
Total Physical Memory:   16384 MB

PS> Get-PSDrive -PSProvider FileSystem                            ②

Name Used (GB) Free (GB) Provider      Root
---- --------- --------- --------      ----
C       55.20     180.10 FileSystem    C:\\
                                                                  ①
                                                                  ②
```

178.14 Ejemplo 3: Convertir un procedimiento en runbook (plantilla)

Usa esta plantilla para estandarizar correcciones y evitar repeticion:

1. **Contexto:** que paso, impacto, desde cuando
2. **Evidencia:** comandos ejecutados + salidas relevantes
3. **Causa raiz:** que lo provoco (y como se confirma)
4. **Fix:** pasos exactos (preferir cambios reversibles)
5. **Verificacion:** como confirmar que quedo bien (metricas, healthchecks)
6. **Rollback:** como volver atras si algo sale mal

178.15 Custom commands (para runbooks repetibles)

OpenCode permite comandos custom por proyecto. Esto es util para estandarizar runbooks.

Ejemplo: crear `.opencode/commands/preflight.md`:

1. **.opencode/commands/** permite ejecutar prompts repetibles como comandos en la TUI.

178.16 Laboratorio practico (20-30 min)

Objetivo: generar un runbook de “Servidor lento” basado en evidencia.

Listing 178.14 BASH

```
$ systemctl status nginx --no-pager                                    ①
* nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled)
     Active: active (running) since Sun 2026-02-01 09:12:14 UTC; 2h 10min ago

$ journalctl -u nginx --since "30 min ago" --no-pager                  ②
Feb 01 11:11:03 server nginx[1234]: 2026/02/01 11:11:03 [error] 1234#1234: *918 connect()

$ nginx -t                                                             ③
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
                                                                       ①
                                                                       ②
                                                                       ③
```

Listing 178.15 TEXT

```
Estoy por desplegar X en este servidor.

Evidencia: (pego outputs de uname/free/df/systemctl/journalctl).

Necesito una checklist de preflight de 10 items max:
- Solo comandos read-only
- Criterio de OK/FAIL por cada comando
- Riesgo si se ignora
 # <1>
```

① **uptime** muestra uptime y load average (primer indicador de presion)
② **top -b -n 1** toma un snapshot no interactivo (pega solo las primeras lineas)
③ **ss -s** resume sockets (util para identificar picos de conexiones)

Entrega:

- Pega en OpenCode: `uptime`, las primeras 15 lineas de `top` y `ss -s`.
- Pide: “Dame un plan de diagnostico de 10 minutos con comandos read-only, y luego una propuesta de mitigacion con rollback”.
- Escribe el runbook usando la plantilla del Ejemplo 3.

Listing 178.16 MARKDOWN

```
---
description: Checklist preflight (servidores)
agent: plan
---
Genera una checklist de preflight (max 12 items) para el cambio: $ARGUMENTS.
Reglas:
- solo comandos read-only
- incluye criterio OK/FAIL por item
- incluye rollback / mitigacion rapida si detectas riesgo
# <1>
```

Listing 178.17 BASH

```
$ uptime                                                                    ①
 11:40:12 up 18 days,  3:12,  2 users,  load average: 2.10, 1.95, 1.20

$ top -b -n 1 | head -n 15                                                  ②
top - 11:40:12 up 18 days,  3:12,  2 users,  load average: 2.10, 1.95, 1.20
Tasks: 231 total,   2 running, 229 sleeping,   0 stopped,   0 zombie
%Cpu(s): 12.0 us,  2.5 sy,  0.0 ni, 84.8 id,  0.6 wa,  0.0 hi,  0.1 si,  0.0 st

$ ss -s                                                                     ③
Total: 842 (kernel 0)
TCP:   125 (estab 48, closed 32, orphaned 0, timewait 32)
UDP:   10
                                                                            ①
                                                                            ②
                                                                            ③
```

178.17 Mejores practicas

RECOMENDACION

Usa OpenCode como copiloto, no como piloto automatico.
Casos de uso: - Checklists de despliegue y preflight - Interpretacion de logs y errores - Generacion de scripts con validaciones y modo seguro
Cuando aplicar: - Incidentes repetitivos - Tareas manuales que se hacen mas de 2 veces - Cambios con riesgo (mejorar documentacion y rollback)

178.18 Resumen

- Captura evidencia primero (sin cambios)
- Pide a OpenCode hipotesis + plan de verificacion
- Ejecuta cambios pequenos, verificables y reversibles
- Documenta un runbook para el proximo incidente

178.19 Referencias

- systemd - systemctl
- journalctl
- nginx - Testing Configuration
- OpenCode - Docs
- OpenCode - Share
- OpenCode - Commands

179 Anexo I: OpenClaw + Telegram (Abacombot) - De 0 a Avanzado

Listing 179.1 `QUARTO-TITLE-BLOCK`

Asistente Sysadmin por chat con pairing, allowlists y sesiones aisladas

180 Anexo I: OpenClaw + Telegram (Abacombot) - De 0 a Avanzado

180.1 Introduccion

Este anexo te guia para montar un asistente tipo **Sysadmin** por **Telegram**, usando **OpenClaw** (Gateway + canales + sesiones + skills) en **Docker**.

El bot de Telegram se llama por defecto **Abacombot** (configurable en BotFather). En este documento, **Abacombot** se refiere solo al bot de Telegram (no a un usuario Linux).

Enfoque de seguridad:

- DMs protegidos con **pairing**.
- Grupos con **mention gating** (`requireMention: true`) por defecto.
- Control de acceso por allowlists (DM y grupos).

Ubuntu Server LTS recomendado: 22.04 o 24.04.

180.2 Objetivos

- Clonar OpenClaw y levantar el Gateway en Docker.
- Conectar Telegram al Gateway con un token de BotFather.
- Aplicar postura minima de seguridad (pairing + mention gating + allowlists).
- Verificar salud, logs y troubleshooting basico.

180.3 Requisitos

180.3.1 En Telegram

- Una cuenta de Telegram.
- Token del bot creado con `@BotFather`.

180.3.2 En el servidor (Ubuntu Server LTS)

- Acceso SSH.
- Docker Engine + Docker Compose v2.
- Espacio en disco para imagenes y logs.

180.4 Paso 1: Crear el bot en Telegram (BotFather)

1. Abre Telegram y busca `@BotFather`.
2. Ejecuta `/newbot` y define:
   - **Nombre**: `Abacombot` (o el que prefieras)
   - **Username**: debe terminar en `bot` (ej: `abacom_sysadmin_bot`)
3. BotFather te entregara un token tipo: `123456789:AA...REDACTED...xyz`

> ⚠ ADVERTENCIA CRITICA
>
> **No pegues el token en repositorios ni capturas publicas.**
> **Lo que podria salir mal:** - Un atacante toma control del bot. - Se filtran datos del servidor por chat.
> **Como prevenirlo:** 1. Guarda el token como secreto (archivo root-only o secret manager). 2. Usa `dmPolicy: "pairing"`. 3. Usa allowlists (DM y grupos) y mention gating en grupos.

180.5 Paso 2: Clonar OpenClaw

Listing 180.1 BASH

```
$ git clone https://github.com/openclaw/openclaw.git ①
$ cd openclaw ②
① ②
```

---

① **git clone** descarga el repo oficial de OpenClaw.
② **cd openclaw** entra al directorio donde viven `docker-setup.sh` y `docker-compose.yml`.

180.6 Paso 3: Levantar OpenClaw en Docker (recomendado)

Listing 180.2 BASH

```
$ ./docker-setup.sh                                                    ①
```

① **docker-setup.sh** construye la imagen, ejecuta onboarding y deja el gateway listo via Docker Compose (ademas escribe `.env`).

Si necesitas levantar el gateway manualmente (sin el script), la ruta de referencia es:

Listing 180.3 BASH

```
$ docker build -t openclaw:local -f Dockerfile .                       ①
$ docker compose run --rm openclaw-cli onboard                         ②
$ docker compose up -d openclaw-gateway                                ③
                                                                       ①
                                                                       ②
                                                                       ③
```

① **docker build** crea una imagen local para el gateway.
② **openclaw-cli onboard** corre el wizard dentro del contenedor CLI.
③ **docker compose up -d openclaw-gateway** inicia el gateway en segundo plano.

180.7 Paso 4: Conectar Telegram al Gateway (token)

Listing 180.4 BASH

```
$ docker compose run --rm openclaw-cli channels add --channel telegram --token "<token>" (
```

① **channels add (telegram)** registra el bot token para que el gateway pueda hacer polling al Bot API.

Listing 180.5 JSON5

```
{
  channels: {
    telegram: {
      enabled: true,
      dmPolicy: "pairing",
      groups: { "*": { requireMention: true } },
    },
  },
} # <1>
```

180.8 Paso 5: Postura minima de seguridad (recomendada)

Config minima (conceptual) para Telegram:

① **dmPolicy + requireMention** evita que DMs desconocidos o mensajes en grupos sin mencion activen el bot.

Nota importante sobre grupos:

- En OpenClaw, `channels.telegram.groupPolicy` por defecto es **"allowlist"**. Si vas a usar grupos, define `groupAllowFrom` (quienes pueden escribir) o cambia explicitamente a `groupPolicy: "open"` si tu objetivo es permitir a cualquiera del grupo.

180.9 Paso 6: Verificacion (salud + logs)

Listing 180.6 BASH

```
$ docker compose exec openclaw-gateway node dist/index.js health --token "$OPENCLAW_GATEW
$ docker compose logs -n 200 openclaw-gateway                                          ②
                                                                                       ①
                                                                                       ②
```

① **health** ejecuta el probe oficial del gateway usando el token.
② **docker compose logs** muestra logs recientes del gateway (errores de token, red, Bot API, etc.).

180.10 Paso 7: Pairing (primer DM)

Con `dmPolicy: "pairing"`, el primer DM requiere aprobacion.

Listing 180.7 BASH

```
$ docker compose run --rm openclaw-cli pairing list telegram            ①
$ docker compose run --rm openclaw-cli pairing approve telegram <CODE>  ②
                                                                        ①
                                                                        ②
```

① **pairing list** lista solicitudes pendientes para Telegram.
② **pairing approve** aprueba el codigo y habilita el DM.

180.11 Troubleshooting

180.11.1 El bot no responde

Checklist rapido:

- El gateway esta arriba: `docker compose ps`
- Hay token configurado y red a `api.telegram.org`.
- Si es DM: pairing pendiente.
- Si es grupo: requiere mencion y (si aplica) allowlist de grupo/senders.

Listing 180.8 BASH

```
$ docker compose ps                                                     ①
$ docker compose logs -n 200 openclaw-gateway                           ②
$ docker compose run --rm openclaw-cli channels status                  ③
                                                                        ①
                                                                        ②
                                                                        ③
```

① **docker compose ps** confirma contenedores y estado.
② **logs** entrega el error real (token, DNS, 401/429, etc.).
③ **channels status** muestra estado del canal Telegram y warnings de config.

180.12 Referencias

- OpenClaw - Docker
- OpenClaw - Telegram
- OpenClaw - Getting Started
- OpenClaw - Configuration

Code Appendix

Canonical. 2024a. “Security Hardening.” 2024.
———. 2024b. “Security Updates.” 2024.
———. 2024c. “Ubuntu 22.04 LTS Installation.” 2024.
———. 2024d. “Ubuntu 22.04 LTS Requirements.” 2024.
———. 2024e. “User Management.” 2024. https://ubuntu.com/server/docs/user-management.
Communications, SSH. 2024. “SSH Security Best Practices.” 2024.
Community, Ubuntu. 2024. “UFW - Uncomplicated Firewall.” 2024.
DistroWatch. 2024. “DistroWatch.” 2024.
Forum, UEFI. 2022. “UEFI Specification V2.9.” 2022.
Foundation, Free Software. 2023a. “Bash Manual.” 2023. https://www.gnu.org/software/bash/manual/.
———. 2023b. “GNU Grep Manual.” 2023. https://www.gnu.org/software/grep/manual/grep.html.
———. 2024a. “Bash Manual.” 2024.
———. 2024b. “GNU Coreutils Manual.” 2024.
Foundation, Linux. 2019. “Filesystem Hierarchy Standard (FHS).” 2019.
———. 2024. “Linux File Permissions.” 2024.
Foundation, The Linux. 2024. “Linux Kernel History.” 2024.
GNU GPL V3.0. 2007. Free Software Foundation.
IEEE. 2018. “POSIX.1-2017 Portable Operating System Interface.” 2018. https://pubs.opengroup.org/onlinepubs/9699919799/.
kernel.org. 2024. “Ext4 Filesystem.” 2024.
Linux.com. 2024. “Linux File Permissions.” 2024.
POSIX.1-2017. 2017. The Open Group.
Project, Debian. 2024a. “Debian System Administrator’s Manual.” 2024.
———. 2024b. “Sudo - Debian Wiki.” 2024.
———. 2024c. “Unattended Upgrades.” 2024.
Project, Fail2ban. 2024. “Fail2ban.” 2024.
Project, GNU. 2023. “GNU Findutils Documentation.” 2023. https://www.gnu.org/software/findutils/manual/html_mono/find.html.
———. 2024. “GNU Coreutils Manual.” 2024. https://www.gnu.org/software/coreutils/manual/coreutils.html.
project, Linux man-pages. 2024. “Chmod(1) — Linux Manual Page.” 2024. https://man7.org/linux/man-pages/man1/chmod.1.html.
Torvalds, L. 1991. “Hello from Finland.” 1991.
Unix.com. 2024. “UNIX History.” 2024.
Wikipedia. 2024. “UEFI.” 2024.