# Trabajo Práctico - Administración de Sistemas Linux con Vagrant

## Entrega Final

**Fecha límite:** 23:59 del 4 de diciembre de 2025 (inclusive)

⚠️ **IMPORTANTE:** Los commits realizados después del 4 de diciembre de 2025  a las 23:59 no serán tenidos en cuenta para la evaluación.

---

## ⚠️ IMPORTANTE - Usuarios de Mac M1/M2

Si tienes un Mac con chip M1 o M2, debes usar QEMU en lugar de VirtualBox. Sigue estas instrucciones:

1. **Instalar QEMU:**

brew install qemu

2. **Usar el Vagrantfile específico para ARM** disponible en:
    - https://github.com/upszot/UTN-FRA_SO_Vagrant/blob/master/VagrantFiles/Mac/Vagrantfile

---

## 📊 Distribución de Calificaciones

- **30%** - Tareas Individuales
- **40%** - Tareas Grupales/Colaborativas
- **10%** - Claridad en Presentación y Documentación
- **20%** - Ejercicio Bonus (Servidor LAMP)

---

## Introducción

Este trabajo práctico está diseñado para equipos de **3 integrantes** que trabajarán de forma colaborativa en un entorno Linux virtualizado. Si tu grupo tiene menos de 3 personas, **deberás asumir los roles y tareas de los perfiles faltantes**.

---

## Objetivos de Aprendizaje

- Virtualización con Vagrant
- Control de versiones colaborativo con Git (comandos básicos)
- Administración básica de sistemas Linux
- Gestión de permisos y usuarios
- Administración de volúmenes lógicos con LVM
- Contenedores con Docker y Docker Compose
- Monitoreo de sistemas con Grafana, Loki y Prometheus

---

## ⚠️ Nota sobre Scripts

Este trabajo **NO requiere** conocimiento previo de scripting. Todos los comandos se proporcionan paso a paso.

---

## Configuración Inicial

### Vagrantfile para Intel/AMD (VirtualBox)

```
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/jammy64"
  config.vm.hostname = "linux-practice"

  config.vm.provider "virtualbox" do |vb|
    vb.memory = "2048"
    vb.cpus = 2

    # Agregar disco adicional de 2GB para ejercicios de LVM
    unless File.exist?('./disk1.vdi')
      vb.customize ['createhd', '--filename', './disk1.vdi', '--size',
2048]
    end
    vb.customize ['storageattach', :id, '--storagectl', 'SCSI',
'--port', 2, '--device', 0, '--type', 'hdd', '--medium', './disk1.vdi']
  end

  config.vm.network "public_network"

  config.vm.provision "shell", inline: <<-SHELL
```

```
    apt-get update
    apt-get install -y fastfetch git docker.io docker-compose lvm2
    usermod -aG docker vagrant
    systemctl enable docker
    systemctl start docker
  SHELL
end
```

### Vagrantfile para Mac M1/M2 (QEMU)

Usar el Vagrantfile disponible en:
**https://github.com/upszot/UTN-FRA_SO_Vagrant/blob/master/VagrantFiles/Mac/Vagrantfile**

Asegurarse de modificar la sección de provisioning para incluir:

```
config.vm.provision "shell", inline: <<-SHELL
  apt-get update
  apt-get install -y fastfetch git docker.io docker-compose lvm2
  usermod -aG docker vagrant
  systemctl enable docker
  systemctl start docker
  # Crear archivo que simule un disco adicional
  dd if=/dev/zero of=/tmp/disk1.img bs=1M count=2048
  losetup /dev/loop10 /tmp/disk1.img
SHELL
```

---

# Distribución de Roles

## 👤 Alumno A - "Administrador"

- Responsable de la configuración inicial del sistema
- Gestión de usuarios y permisos
- Administración de volúmenes lógicos (LVM)

## 👤 Alumno B - "Desarrollador"

- Configuración de repositorio Git
- Gestión de contenedores Docker
- Automatización de tareas

## 👤 Alumno C - "Operador"

- Desarrollo de comandos
- Gestión de archivos y directorios
- Testing y validación

---

# Ejercicios

## 0. Descubrimiento de la IP de la VM (Colaborativo)

**Objetivo:** Aprender a identificar la dirección IP de su máquina virtual en modo bridge.

**Pasos:**

1. Una vez que la VM esté levantada, conectarse con `vagrant ssh`

2. Ejecutar el siguiente comando para ver todas las interfaces de red:

ip addr show

3. Identificar la interfaz de red principal (generalmente `enp0s8` o `eth1`)

4. Anotar la dirección IP asignada (será una IP de tu red local, por ejemplo: `192.168.1.XXX`)

5. Verificar conectividad:

```
ping -c 4 8.8.8.8
```

6. Cada alumno debe guardar su IP en el archivo `informacion/ip_vm.txt`:

```
echo "========== IP DE LA VM - [APELLIDO] ==========" >>
informacion/ip_vm.txt
ip addr show | grep "inet " | grep -v "127.0.0.1" >>
informacion/ip_vm.txt
echo "" >> informacion/ip_vm.txt
```

**Nota importante:** Esta IP será necesaria para acceder a los servicios web (Grafana, Apache) desde tu navegador. Anótala bien.

**Entregable:** Archivo `informacion/ip_vm.txt` con las IPs de cada alumno

---

## 1. Configuración Inicial y Repositorio (Colaborativo)

**Pasos:**

1. **Alumno B** crea un repositorio público en GitHub llamado `practica-linux-[apellidos-equipo]`
2. Cada alumno clona el repositorio en su máquina virtual
3. Configurar Git con sus datos personales en cada VM
4. Crear la estructura inicial:

```
proyecto/
├── informacion/
├── permisos/
├── lvm/
├── archivos/
└── contenedores/
```

**Entregable:** Screenshot del repositorio creado y estructura de carpetas

---

## 2. Fastfetch Colaborativo (Colaborativo)

**Objetivo:** Cada alumno debe ejecutar `fastfetch` en su VM y agregar la salida al mismo archivo sin sobrescribir el contenido anterior.

**Pasos:**

1. Cada alumno ejecuta: `fastfetch > temp_fastfetch_[nombre].txt`
2. **Alumno A** inicia creando `informacion/system_info.txt` con su fastfetch
3. **Alumno B** hace pull, agrega su fastfetch al final del archivo usando `cat`
4. **Alumno C** repite el proceso
5. **IMPORTANTE:** Todos los alumnos deben usar `>>` para agregar su información al mismo archivo `system_info.txt`, de manera que el archivo final contenga los 3 fastfetch claramente separados

**Script de ayuda:**

```
#!/bin/bash
echo "========== FASTFETCH DE [NOMBRE] ==========" >>
informacion/system_info.txt
fastfetch >> informacion/system_info.txt
echo "" >> informacion/system_info.txt
```

**Entregable:** Archivo `system_info.txt` con los 3 fastfetch y historial de commits

---

## 3. Gestión de Permisos (Individual + Colaborativo)

**Parte Individual (cada alumno):**

Esta sección te ayudará a comprender cómo funcionan los permisos en Linux, quiénes son los usuarios y cómo pueden colaborar de forma segura en el mismo sistema.

**¿Qué vamos a hacer?**

- Crear tu propio espacio de trabajo personal
- Establecer archivos con diferentes niveles de privacidad
- Simular un entorno de trabajo real con múltiples usuarios

**Pasos:**

1. Crear un directorio personal: `/home/vagrant/[apellido]_espacio/`

2. Crear un archivo `privado.txt` con permisos 600 (solo el dueño puede leer y escribir)

    - **¿Por qué 600?** Estos permisos significan que SOLO tú puedes ver y modificar el archivo. Nadie más en el sistema puede acceder a él.
3. Crear un archivo `publico.txt` con permisos 644 (dueño lee/escribe, otros solo leen)

    - **¿Por qué 644?** El dueño tiene control completo, pero otros usuarios del sistema pueden leer el contenido (útil para compartir información sin permitir modificaciones).
4. Crear usuarios locales para simular el trabajo colaborativo

**Parte Colaborativa - Creación de usuarios locales:**

**¿Por qué creamos usuarios locales?** En un entorno de trabajo real, múltiples personas necesitan acceder al mismo servidor. Cada persona tiene su propio usuario con sus propios permisos. Vamos a simular esto creando usuarios de prueba.

Cada alumno debe crear usuarios locales en su máquina para simular el trabajo en equipo:

```
# Cada alumno ejecuta estos comandos para crear usuarios de prueba:
sudo useradd -m estudiante1
sudo useradd -m estudiante2
sudo useradd -m estudiante3

# Verificar que se crearon correctamente:
cat /etc/passwd | grep estudiante
```

**Trabajo con grupos:**

**¿Qué es un grupo?** Un grupo en Linux permite que múltiples usuarios trabajen juntos en los mismos archivos. Es como un "equipo" donde todos los miembros tienen los mismos permisos sobre ciertos recursos.

1. **Alumno A** crea un grupo llamado `equipotrabajo`:

```
sudo groupadd equipotrabajo
sudo usermod -a -G equipotrabajo estudiante1
sudo usermod -a -G equipotrabajo estudiante2
sudo usermod -a -G equipotrabajo estudiante3
sudo usermod -a -G equipotrabajo vagrant
```

2. Crear directorio colaborativo:

```
sudo mkdir /tmp/colaborativo
sudo chgrp equipotrabajo /tmp/colaborativo
sudo chmod 770 /tmp/colaborativo
```

**¿Qué significan estos comandos?**

- `chgrp`: Cambia el grupo dueño del directorio
- `chmod 770`: Permite que el dueño y el grupo tengan permisos completos (lectura, escritura, ejecución), pero otros usuarios no tienen acceso

3. Cada alumno debe crear su archivo personal de verificación mostrando sus grupos:

# Crear archivo con información personal de usuarios y grupos

```
echo "========== INFORMACIÓN DE USUARIOS Y GRUPOS - [APELLIDO]
==========" > permisos/usuarios_[apellido].txt
echo "Usuario actual:" >> permisos/usuarios_[apellido].txt
whoami >> permisos/usuarios_[apellido].txt
echo "Grupos del usuario:" >> permisos/usuarios_[apellido].txt
groups >> permisos/usuarios_[apellido].txt
echo "ID del usuario:" >> permisos/usuarios_[apellido].txt
id >> permisos/usuarios_[apellido].txt
echo "Usuarios del sistema:" >> permisos/usuarios_[apellido].txt
cat /etc/passwd | grep estudiante >> permisos/usuarios_[apellido].txt
echo "" >> permisos/usuarios_[apellido].txt
```

**¿Cómo hacer la verificación?**

La verificación consiste en ejecutar comandos y guardar su salida en un archivo de texto. El símbolo `>>` significa "agregar al final del archivo" y `>` significa "crear archivo nuevo o sobrescribir".

**Verificación de permisos:**

```
# Cada alumno ejecuta estos comandos para verificar los permisos:
echo "========== VERIFICACIÓN DE PERMISOS - [APELLIDO] ==========" >>
permisos/verificacion_permisos.txt
echo "Archivos en mi directorio personal:" >>
permisos/verificacion_permisos.txt
ls -la /home/vagrant/[apellido]_espacio/ >>
permisos/verificacion_permisos.txt
echo "Archivos en directorio colaborativo:" >>
permisos/verificacion_permisos.txt
ls -la /tmp/colaborativo/ >> permisos/verificacion_permisos.txt
echo "Información del grupo equipotrabajo:" >>
permisos/verificacion_permisos.txt
getent group equipotrabajo >> permisos/verificacion_permisos.txt
echo "" >> permisos/verificacion_permisos.txt
```

**Entregable:** Archivos `usuarios_[apellido].txt` y `verificacion_permisos.txt` de cada alumno

---

## 4. Operaciones con LVM - Logical Volume Manager (Individual)

**Objetivo:** Cada alumno debe crear volúmenes lógicos usando LVM para gestionar el disco adicional de forma flexible.

**¿Qué es LVM?** LVM (Logical Volume Manager) permite gestionar discos de forma más flexible que las particiones tradicionales. Puedes redimensionar, crear snapshots, y agregar más discos sin perder datos.

**Pasos para VirtualBox:**

1. Identificar el disco nuevo: `/dev/sdc`

2. Crear Physical Volume (PV):

```
sudo pvcreate /dev/sdc
```

3. Crear Volume Group (VG):

```
sudo vgcreate vg_datos_[apellido] /dev/sdc
```

4. Crear Logical Volume (LV):

```
sudo lvcreate -L 1.5G -n lv_storage_[apellido] vg_datos_[apellido]
```

5. Formatear con ext4:

```
sudo mkfs.ext4 /dev/vg_datos_[apellido]/lv_storage_[apellido]
```

6. Crear punto de montaje y montar:

```
sudo mkdir /mnt/lvm_storage_[apellido]
sudo mount /dev/vg_datos_[apellido]/lv_storage_[apellido]
/mnt/lvm_storage_[apellido]
```

7. Agregar al fstab para montaje automático:

```
echo "/dev/vg_datos_[apellido]/lv_storage_[apellido]
/mnt/lvm_storage_[apellido] ext4 defaults 0 0" | sudo tee -a /etc/fstab
```

**Pasos para QEMU (Mac M1/M2):**

1. Usar el loop device: `/dev/loop10`

2. Crear Physical Volume:

```
sudo pvcreate /dev/loop10
```

3. Continuar con los mismos pasos 3-7 que VirtualBox

**¿Cómo hacer la verificación de LVM?**

La verificación consiste en capturar tres estados diferentes del sistema:

1. **Estado inicial:** espacio en disco antes de montar el LVM
2. **Estado con LVM montado:** espacio disponible después del montaje
3. **Información de LVM:** detalles de los volúmenes creados

**Verificación:**

```
# Cada alumno ejecuta estos comandos para verificar LVM:
echo "========== VERIFICACIÓN DE LVM - [APELLIDO] ==========" >
lvm/lvm-[apellido].txt

echo "=== ESPACIO EN DISCO SIN LVM MONTADO ===" >>
lvm/lvm-[apellido].txt
# Desmontar temporalmente para mostrar el estado inicial
sudo umount /mnt/lvm_storage_[apellido]
df -h >> lvm/lvm-[apellido].txt
echo "" >> lvm/lvm-[apellido].txt

echo "=== ESPACIO EN DISCO CON LVM MONTADO ===" >>
lvm/lvm-[apellido].txt
sudo mount /dev/vg_datos_[apellido]/lv_storage_[apellido]
/mnt/lvm_storage_[apellido]
df -h >> lvm/lvm-[apellido].txt
echo "" >> lvm/lvm-[apellido].txt

echo "=== ESCANEO DE LOGICAL VOLUMES ===" >> lvm/lvm-[apellido].txt
sudo lvscan >> lvm/lvm-[apellido].txt
echo "" >> lvm/lvm-[apellido].txt

echo "=== ESCANEO DE PHYSICAL VOLUMES ===" >> lvm/lvm-[apellido].txt
sudo pvscan >> lvm/lvm-[apellido].txt
echo "" >> lvm/lvm-[apellido].txt
```

```
echo "=== ESCANEO DE VOLUME GROUPS ===" >> lvm/lvm-[apellido].txt
sudo vgscan >> lvm/lvm-[apellido].txt
echo "" >> lvm/lvm-[apellido].txt
```

**Entregable:** Archivo `lvm-[apellido].txt` con la verificación de cada alumno

---

## 5. Gestión de Archivos y Directorios (Individual)

**Objetivo:** Operaciones avanzadas con archivos y directorios.

**Tareas por alumno:**

1. Crear estructura de directorios:

```
/mnt/lvm_storage_[apellido]/
├── proyectos/
│   ├── activos/
│   └── archivados/
├── respaldos/
└── temporal/
```

2. Crear 10 archivos de prueba en `temporal/`:

```
cd /mnt/lvm_storage_[apellido]/temporal/
for i in {01..10}; do
  touch documento_$i.txt
  echo "Contenido del documento $i" > documento_$i.txt
done
```

3. Operaciones de copia y movimiento:

```
# Copiar archivos 1-5 a proyectos/activos/
cp documento_01.txt documento_02.txt documento_03.txt documento_04.txt
documento_05.txt ../proyectos/activos/

# Mover archivos 6-8 a proyectos/archivados/
mv documento_06.txt documento_07.txt documento_08.txt
../proyectos/archivados/

# Crear backup de archivos 9-10 en respaldos/
```

```
cp documento_09.txt documento_10.txt ../respaldos/

# Eliminar los archivos originales restantes de temporal/
rm documento_09.txt documento_10.txt
```

4. **¿Cómo hacer la verificación de archivos?**

La verificación muestra todos los archivos creados y sus ubicaciones finales.

**Verificación:**

```
# Cada alumno ejecuta estos comandos para verificar los archivos:
echo "========== VERIFICACIÓN DE ARCHIVOS - [APELLIDO] ==========" >>
archivos/verificacion_archivos.txt
echo "Estructura de directorios creada:" >>
archivos/verificacion_archivos.txt
find /mnt/lvm_storage_[apellido] -type d >>
archivos/verificacion_archivos.txt
echo "Archivos en cada directorio:" >>
archivos/verificacion_archivos.txt
find /mnt/lvm_storage_[apellido] -type f -exec ls -la {} \; >>
archivos/verificacion_archivos.txt
echo "" >> archivos/verificacion_archivos.txt
```

**Entregable:** Archivo `verificacion_archivos.txt` con la verificación de cada alumno

---

## 6. Contenedores y Monitoreo con Docker Compose (Colaborativo)

**Objetivo:** Levantar servicios con Docker Compose y monitorearlos con Grafana, Loki y Prometheus.

**Asignación de tareas:**

- **Alumno A:** Investigar y crear el archivo `docker-compose.yml`
- **Alumno B:** Configurar variables de entorno y networking
- **Alumno C:** Testing, documentación y capturas de pantalla

**Archivo docker-compose.yml:**

```
version: '3.8'
```

```yaml
services:
  # Servicio 1: Nginx - Servidor web
  nginx:
    container_name: nginx-practica
    image: nginx:alpine
    ports:
      - "8081:80"
    restart: unless-stopped
    networks:
      - monitoring

  # Servicio 2: Redis - Base de datos en memoria
  redis:
    container_name: redis-practica
    image: redis:alpine
    ports:
      - "6379:6379"
    restart: unless-stopped
    networks:
      - monitoring-network
  # Servicio 3: Postgres - Base de datos
  postgres:
    container_name: postgres-practica
    image: postgres:alpine
    environment:
      POSTGRES_USER: alumno
      POSTGRES_PASSWORD: practica123
      POSTGRES_DB: tp_final
    ports:
      - "5432:5432"
    restart: unless-stopped
    networks:
      - monitoring

  # Prometheus - Recolección de métricas
  prometheus:
    container_name: prometheus-practica
    image: prom/prometheus:latest
    ports:
      - "9090:9090"
    volumes:
      - ./prometheus.yml:/etc/prometheus/prometheus.yml
    command:
      - '--config.file=/etc/prometheus/prometheus.yml'
    restart: unless-stopped
    networks:
```

```
      - monitoring
    depends_on:
      - nginx
      - redis

  # Loki - Agregación de logs
  loki:
    container_name: loki-practica
    image: grafana/loki:latest
    ports:
      - "3100:3100"
    command: -config.file=/etc/loki/local-config.yaml
    restart: unless-stopped
    networks:
      - monitoring

  # Grafana - Visualización de métricas y logs
  grafana:
    container_name: grafana-practica
    image: grafana/grafana:latest
    ports:
      - "3000:3000"
    environment:
      - GF_SECURITY_ADMIN_PASSWORD=practica123
      - GF_USERS_ALLOW_SIGN_UP=false
    volumes:
      - grafana-data:/var/lib/grafana
    restart: unless-stopped
    depends_on:
      - prometheus
      - loki
    networks:
      - monitoring

networks:
  monitoring:
    driver: bridge

volumes:
  grafana-storage:
```

Crear el archivo en `contenedores/docker-compose.yml`:

⚠️ **NOTA IMPORTANTE:** Este archivo contiene errores intencionales que deberán encontrar y corregir como parte del ejercicio. Utilicen los logs de Docker para identificar los problemas.

**Pistas para el debug:**

- Revisen los nombres de las redes en todos los servicios
- Revisen los nombres de los volúmenes declarados vs utilizados
- Usen `docker-compose config` para validar la sintaxis
- Usen `docker-compose logs [nombre-servicio]` para ver errores específicos

**Archivo prometheus.yml:**

Crear el archivo en `contenedores/prometheus.yml`:

```
global:
  scrape_interval: 15s
  evaluation_interval: 15s
scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']

  - job_name: 'nginx'
    static_configs:
      - targets: ['nginx:9113']  # ERROR INTENCIONAL: nginx no expone
métricas en este puerto por defecto
```

⚠️ **NOTA:** Este archivo contiene errores intencionales. Prometheus se levantará, pero algunos targets aparecerán como "DOWN". Deberán investigar por qué.

**Pistas para el debug:**

- Verifiquen los targets en Prometheus UI (http://localhost:9090/targets)
- Nginx alpine no expone métricas por defecto - pueden comentar ese job o agregar nginx-prometheus-exporter
- Los logs de Prometheus mostrarán warnings útiles

**Tareas de verificación:**

1. Navegar al directorio: `cd contenedores/`

2. **IMPORTANTE - Proceso de Debug:**

    - Primero intentar ejecutar: `docker-compose up -d`
    - Si encuentran errores, documentarlos
    - Usar `docker-compose config` para validar la sintaxis
    - Revisar logs: `docker-compose logs [nombre-servicio]`
    - Corregir los errores encontrados
    - Volver a ejecutar: `docker-compose up -d`

3. Verificar que los servicios estén corriendo: `docker ps`
    - Deben aparecer 6 contenedores corriendo
4. Verificar que no haya contenedores con errores:

```
docker ps -a | grep -i exit
```

5. Acceder a Grafana: http://localhost:3000 (o usar la IP de tu VM si accedes desde otro equipo)
    - Usuario: `admin`
    - Contraseña: `practica123`
6. Configurar Prometheus como datasource en Grafana:
    - Ir a Configuration → Data Sources → Add data source
    - Seleccionar Prometheus
    - URL: `http://prometheus:9090`
    - Click en "Save & Test"
7. Configurar Loki como datasource en Grafana:
    - Add data source → Loki
    - URL: `http://loki:3100`
    - Click en "Save & Test"
8. Verificar targets en Prometheus:
    - Acceder a http://localhost:9090/targets
    - Documentar qué targets están UP y cuáles están DOWN
    - Explicar por qué algunos pueden estar DOWN
9. Tomar screenshots de:
    - La interfaz de Grafana con datasources configurados
    - La página de targets de Prometheus
    - Un dashboard simple creado en Grafana (puede usar el dashboard de Prometheus)
    - Los logs mostrando los errores iniciales y su resolución
10. Verificar logs de todos los servicios:

```
docker-compose logs > logs_completos.txt
```

**¿Cómo hacer la verificación de contenedores?**

La verificación muestra que los contenedores están funcionando correctamente y responden a las peticiones.

**Verificación:**

```
# Cada alumno ejecuta estos comandos para verificar Docker:
echo "========== VERIFICACIÓN CONTENEDORES - [APELLIDO] ==========" >>
contenedores/verificacion_contenedores.txt

echo "=== IP DE LA VM ===" >> contenedores/verificacion_contenedores.txt
hostname -I >> contenedores/verificacion_contenedores.txt
echo "" >> contenedores/verificacion_contenedores.txt

echo "=== CONTENEDORES EN EJECUCIÓN ===" >>
contenedores/verificacion_contenedores.txt
docker ps >> contenedores/verificacion_contenedores.txt
echo "" >> contenedores/verificacion_contenedores.txt

echo "=== ESTADO DE DOCKER-COMPOSE ===" >>
contenedores/verificacion_contenedores.txt
docker-compose ps >> contenedores/verificacion_contenedores.txt
echo "" >> contenedores/verificacion_contenedores.txt

echo "=== REDES DE DOCKER ===" >>
contenedores/verificacion_contenedores.txt
docker network ls >> contenedores/verificacion_contenedores.txt
echo "" >> contenedores/verificacion_contenedores.txt

echo "=== VOLÚMENES DE DOCKER ===" >>
contenedores/verificacion_contenedores.txt
docker volume ls >> contenedores/verificacion_contenedores.txt
echo "" >> contenedores/verificacion_contenedores.txt
```

**Entregable:**

- Archivo `docker-compose.yml` **CORREGIDO**
- Archivo `prometheus.yml` **CORREGIDO**
- Archivo `errores_encontrados.md` documentando:
  - Errores encontrados en el docker-compose.yml original
  - Cómo los identificaron (comandos usados)
  - Cómo los solucionaron
  - Problemas con Prometheus targets y su explicación
- Capturas de pantalla de:
  - `docker ps` mostrando todos los contenedores corriendo (6 contenedores)

    - Grafana con datasources configurados y en estado "Working"
    - Dashboard de Grafana mostrando métricas de Prometheus
    - Prometheus targets page (http://localhost:9090/targets)
    - Logs mostrando la resolución de errores
- Archivo `verificacion_contenedores.txt`

---

# 🚀 Ejercicio Bonus (Opcional)

## Implementación de Servidor LAMP Completo

**Objetivo:** Uno de los integrantes del equipo configura un servidor LAMP (Linux + Apache + MySQL + PHP) completo desde cero e implementa una página web.

**Candidato:** El alumno con mejor rendimiento en ejercicios anteriores o por sorteo.

**Pasos:**

1. **Instalar los componentes del stack LAMP:**

```
# Actualizar repositorios
sudo apt-get update

# Instalar Apache
sudo apt-get install -y apache2

# Instalar MySQL
sudo apt-get install -y mysql-server

# Instalar PHP y módulos necesarios
sudo apt-get install -y php libapache2-mod-php php-mysql

# Verificar versiones instaladas
apache2 -v
mysql --version
php -v
```

2. **Configurar MySQL:**

```
# Iniciar servicio MySQL
sudo systemctl start mysql
sudo systemctl enable mysql

# Ejecutar script de seguridad
```

```
sudo mysql_secure_installation

# Crear base de datos de prueba
sudo mysql -e "CREATE DATABASE tp_final_db;"
sudo mysql -e "CREATE USER 'alumno'@'localhost' IDENTIFIED BY
'practica123';"
sudo mysql -e "GRANT ALL PRIVILEGES ON tp_final_db.* TO
'alumno'@'localhost';"
sudo mysql -e "FLUSH PRIVILEGES;"
```

3. **Configurar Apache:**

```
# Habilitar módulos necesarios
sudo a2enmod rewrite
sudo a2enmod php8.1

# Reiniciar Apache
sudo systemctl restart apache2
sudo systemctl enable apache2
```

4. **Crear el sitio web:**

Crear el archivo `/var/www/html/index.html`:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>TP Final - Arquitectura y Sistemas Operativos</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana,
sans-serif;
            background: linear-gradient(135deg, #667eea 0%, #764ba2
100%);
```

```
            color: #333;
            line-height: 1.6;
        }

        .container {
            max-width: 1200px;
            margin: 0 auto;
            padding: 20px;
        }

        header {
            background: rgba(255, 255, 255, 0.95);
            padding: 30px;
            border-radius: 15px;
            box-shadow: 0 10px 30px rgba(0, 0, 0, 0.2);
            margin-bottom: 30px;
            text-align: center;
        }

        h1 {
            color: #667eea;
            font-size: 2.5em;
            margin-bottom: 10px;
        }

        .subtitle {
            color: #764ba2;
            font-size: 1.2em;
            font-weight: 500;
        }

        .content {
            background: rgba(255, 255, 255, 0.95);
            padding: 40px;
            border-radius: 15px;
            box-shadow: 0 10px 30px rgba(0, 0, 0, 0.2);
        }

        h2 {
            color: #667eea;
            margin-top: 30px;
            margin-bottom: 15px;
            font-size: 1.8em;
        }

        p {
```

```
            margin-bottom: 15px;
            text-align: justify;
        }

        .info-box {
            background: #f8f9fa;
            border-left: 4px solid #667eea;
            padding: 20px;
            margin: 20px 0;
            border-radius: 5px;
        }

        .tech-stack {
            display: grid;
            grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
            gap: 20px;
            margin-top: 30px;
        }

        .tech-card {
            background: linear-gradient(135deg, #667eea 0%, #764ba2
100%);
            color: white;
            padding: 20px;
            border-radius: 10px;
            text-align: center;
            transition: transform 0.3s;
        }

        .tech-card:hover {
            transform: translateY(-5px);
        }

        footer {
            text-align: center;
            margin-top: 40px;
            color: white;
            font-size: 0.9em;
        }
    </style>
</head>
<body>
    <div class="container">
        <header>
            <h1>TP Final Arq y Sis Op</h1>
            <p class="subtitle">Arquitectura y Sistemas Operativos -
```

```
Servidor LAMP</p>
        </header>

        <div class="content">
            <div class="info-box">
                <h2>Sobre este Proyecto</h2>
                <p>Lorem ipsum dolor sit amet, consectetur adipiscing
elit. Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi
ut aliquip ex ea commodo consequat.</p>
            </div>

            <h2>Descripción del Stack LAMP</h2>
            <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum
dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
proident, sunt in culpa qui officia deserunt mollit anim id est
laborum.</p>

            <p>Sed ut perspiciatis unde omnis iste natus error sit
voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque
ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae
dicta sunt explicabo.</p>

            <h2>Componentes Implementados</h2>
            <div class="tech-stack">
                <div class="tech-card">
                    <h3>Linux</h3>
                    <p>Sistema Operativo Base</p>
                </div>
                <div class="tech-card">
                    <h3>Apache</h3>
                    <p>Servidor Web</p>
                </div>
                <div class="tech-card">
                    <h3>MySQL</h3>
                    <p>Base de Datos</p>
                </div>
                <div class="tech-card">
                    <h3>PHP</h3>
                    <p>Lenguaje de Programación</p>
                </div>
            </div>

            <h2>Proceso de Implementación</h2>
            <p>Nemo enim ipsam voluptatem quia voluptas sit aspernatur
```

```
aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit.</p>

            <p>At vero eos et accusamus et iusto odio dignissimos ducimus qui blanditiis praesentium voluptatum deleniti atque corrupti quos dolores et quas molestias excepturi sint occaecati cupiditate non provident, similique sunt in culpa qui officia deserunt mollitia animi, id est laborum et dolorum fuga.</p>

            <div class="info-box">
                <h2>Configuración del Servidor</h2>
                <p>Et harum quidem rerum facilis est et expedita distinctio. Nam libero tempore, cum soluta nobis est eligendi optio cumque nihil impedit quo minus id quod maxime placeat facere possimus, omnis voluptas assumenda est, omnis dolor repellendus.</p>
            </div>

            <h2>Conclusiones</h2>
            <p>Temporibus autem quibusdam et aut officiis debitis aut rerum necessitatibus saepe eveniet ut et voluptates repudiandae sint et molestiae non recusandae. Itaque earum rerum hic tenetur a sapiente delectus, ut aut reiciendis voluptatibus maiores alias consequatur aut perferendis doloribus asperiores repellat.</p>

            <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.</p>
        </div>

        <footer>
            <p>&copy; 2025 - Trabajo Práctico Final - Arquitectura y Sistemas Operativos</p>
            <p>Implementación de Servidor LAMP Completo</p>
        </footer>
    </div>
</body>
</html>
```

5. **Crear archivo de prueba PHP:**

Crear el archivo `/var/www/html/info.php`:

```
<?php
```

```
// Información del sistema
phpinfo();
?>
```

6. **Crear archivo de conexión a MySQL:**

Crear el archivo `/var/www/html/test_db.php`:

```
<?php
$servername = "localhost";
$username = "alumno";
$password = "practica123";
$dbname = "tp_final_db";

// Crear conexión
$conn = new mysqli($servername, $username, $password, $dbname);

// Verificar conexión
if ($conn->connect_error) {
    die("Conexión fallida: " . $conn->connect_error);
}
echo "<h2>Conexión exitosa a MySQL</h2>";
echo "<p>Base de datos: " . $dbname . "</p>";
echo "<p>Servidor: " . $servername . "</p>";

$conn->close();
?>
```

7. **Configurar permisos:**

```
sudo chown -R www-data:www-data /var/www/html/
sudo chmod -R 755 /var/www/html/
```

8. **Verificar servicios:**

```
sudo systemctl status apache2
sudo systemctl status mysql
```

**Verificación:**

```
# Crear archivo de verificación
echo "========== VERIFICACIÓN SERVIDOR LAMP - [APELLIDO] ==========" >
lamp/verificacion_lamp.txt

echo "=== VERSIONES INSTALADAS ===" >> lamp/verificacion_lamp.txt
apache2 -v >> lamp/verificacion_lamp.txt
mysql --version >> lamp/verificacion_lamp.txt
php -v >> lamp/verificacion_lamp.txt
echo "" >> lamp/verificacion_lamp.txt

echo "=== ESTADO DE SERVICIOS ===" >> lamp/verificacion_lamp.txt
sudo systemctl status apache2 --no-pager >> lamp/verificacion_lamp.txt
echo "" >> lamp/verificacion_lamp.txt
sudo systemctl status mysql --no-pager >> lamp/verificacion_lamp.txt
echo "" >> lamp/verificacion_lamp.txt

echo "=== PUERTOS EN ESCUCHA ===" >> lamp/verificacion_lamp.txt
sudo netstat -tlnp | grep -E '(apache|mysql)' >>
lamp/verificacion_lamp.txt
echo "" >> lamp/verificacion_lamp.txt

echo "=== ARCHIVOS WEB CREADOS ===" >> lamp/verificacion_lamp.txt
ls -la /var/www/html/ >> lamp/verificacion_lamp.txt
echo "" >> lamp/verificacion_lamp.txt
```

**Entregable:**

- Archivo `verificacion_lamp.txt` con la verificación completa
- Capturas de pantalla de:
    - Página principal (http://localhost o usar IP de la VM)
    - Página de info.php (http://localhost/info.php)
    - Página de test_db.php mostrando conexión exitosa
- Comandos ejecutados documentados

---

# Criterios de Evaluación

### Trabajo Individual (30%)

- Completitud de ejercicios individuales
- Correcta ejecución de comandos
- Resolución de problemas básicos

### Trabajo Colaborativo (40%)

- Uso efectivo de Git (add, commit, push, pull)
- Resolución de conflictos básicos
- Comunicación y coordinación del equipo

### Presentación y Documentación (10%)

- Claridad de los archivos de verificación
- Screenshots y evidencias
- README del proyecto explicando el proceso

### Ejercicio Bonus (20%)

- Implementación exitosa del servidor LAMP
- Funcionamiento correcto de todos los componentes
- Documentación del proceso
- Capturas de pantalla de las páginas funcionando

---

# Entrega Final

**Fecha límite:** 23:59 del 4 de diciembre de 2025  (inclusive)

⚠️ **IMPORTANTE:** Los commits realizados después del 4 de diciembre de 2025  a las 23:59 no serán tenidos en cuenta para la evaluación.

### Formato:

1. Repositorio Git con todos los archivos
2. README.md explicando el proceso y conclusiones

### Estructura del repositorio final:

```
proyecto/
├── README.md
├── Vagrantfile
├── informacion/
│   ├── ip_vm.txt
│   └── system_info.txt
├── permisos/
│   ├── usuarios_[apellido].txt (uno por alumno)
│   └── verificacion_permisos.txt
├── lvm/
│   └── lvm-[apellido].txt (uno por alumno)
├── archivos/
│   └── verificacion_archivos.txt
├── contenedores/
│   ├── docker-compose.yml (corregido)
│   ├── prometheus.yml (corregido)
│   ├── errores_encontrados.md
│   ├── logs_completos.txt
│   ├── capturas/
│   │   ├── docker_ps.png
│   │   ├── grafana_datasources.png
│   │   ├── grafana_dashboard.png
│   │   ├── prometheus_targets.png
│   │   └── resolucion_errores.png
│   └── verificacion_contenedores.txt
└── lamp/ (bonus)
    ├── capturas/
    │   ├── index_html.png
    │   ├── info_php.png
    │   └── test_db_php.png
    └── verificacion_lamp.txt
```

---

## Recursos de Ayuda

- Vagrant Documentation
- Git Collaboration Guide
- Docker Compose Reference
- LVM Administrator Guide
- Grafana Documentation
- Apache Documentation

---

¡Buena suerte y que aprendan mucho! 🐧