PRÁCTICA 3: Volúmenes y Bind Mounts en Docker Compose

Objetivo

Entender la persistencia de datos en Docker usando volúmenes nombrados y bind mounts.

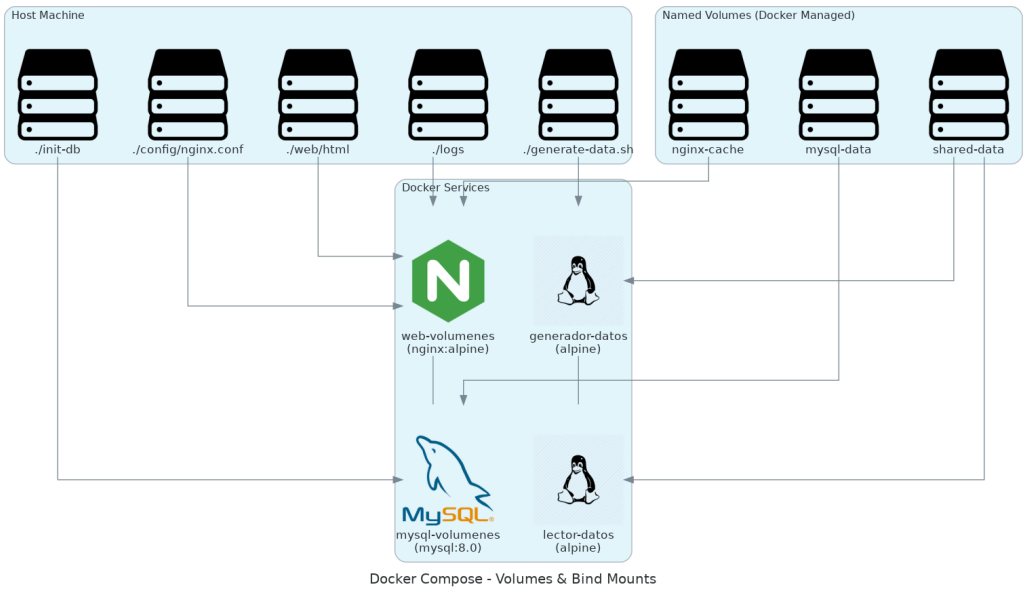
Esta aplicación contará con una página web la cual estará accesible en el puerto 8080, la cual al abrir http://localhost:8080/api/test generará logs ubicados en la carpeta logs. El docker compose generará tres contenedores uno para la base de datos la cual almacenera una tabla usuarios, y el contenedor data-generator y data-reader las cuales simularan el uso de volumenes para la escritura de logs.

Conceptos Teóricos

Tipos de almacenamiento en Docker:

1. **Volúmenes (Volumes)**: Gestionados por Docker, ideales para persistencia
2. **Bind Mounts**: Mapean directorios del host al contenedor

Diagrama General



Pasos de la Práctica

Paso 1: Crear estructura del proyecto

```
mkdir practica3-volumenes
cd practica3-volumenes
mkdir -p web/html
mkdir -p logs
mkdir -p config
mkdir -p init-db
```

Paso 2: Creación del contenido web

Crea `web/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>Práctica de Volúmenes</title>
```

```
  <style>
    body {
      font-family: Arial, sans-serif;
      max-width: 800px;
      margin: 50px auto;
      padding: 20px;
      background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
      color: white;
      line-height: 1.6;
    }
    .container {
      background: rgba(255, 255, 255, 0.1);
      padding: 30px;
      border-radius: 10px;
      backdrop-filter: blur(10px);
      box-shadow: 0 4px 15px rgba(0, 0, 0, 0.2);
    }
    h1 {
      text-align: center;
      margin-bottom: 20px;
    }
    .info {
      background: rgba(255, 255, 255, 0.2);
      padding: 15px;
      border-radius: 5px;
      margin: 15px 0;
    }
    ul {
      padding-left: 20px;
    }
    li {
      margin-bottom: 8px;
    }
  </style>
</head>
<body>
  <main class="container">
    <h1>Práctica de Volúmenes Docker</h1>

    <section class="info">
      <h2>Tipos de almacenamiento:</h2>
      <ul>
        <li><strong>Bind Mount:</strong> Este archivo HTML</li>
        <li><strong>Volume Nombrado:</strong> Logs de Nginx</li>
        <li><strong>Volume Anónimo:</strong> Cache temporal</li>
      </ul>
```

```
    </section>

    <section class="info">
      <p><strong>Tip:</strong> Modifica este archivo en tu host y
recarga la página. ¡Los cambios aparecerán inmediatamente!</p>
    </section>
  </main>
</body>
</html>
```

Paso 3: Crear configuración personalizada de Nginx

Nginx es un servidor web similar apache el cual es utilizado altamente para la publicación de sitios webs desarrollados en diferentes lenguajes de programación.

Crea `config/nginx.conf`:

```
server {
    listen 80;
    server_name localhost;

    # Habilitar logs detallados
    access_log /var/log/nginx/access.log;
    error_log /var/log/nginx/error.log;

    location / {
        root /usr/share/nginx/html;
        index index.html;
    }

    # Endpoint para generar logs de prueba
    location /api/test {
        access_log /var/log/nginx/api-access.log;
        return 200 "Log generado exitosamente\n";
        add_header Content-Type text/plain;
    }
}
```

Paso 4: Crear script para generar datos

El script generado sera utilizado para generar registros den el directorio `/data` el cual se encontrará en un volumen en docker.

Crea `generate-data.sh`:

```
#!/bin/bash
# Script para generar datos de prueba

echo "=== Generador de Datos de Prueba ==="
echo ""
# Crear archivo con timestamp
TIMESTAMP=$(date +"%Y-%m-%d %H:%M:%S")
echo "[$TIMESTAMP] Datos generados desde el contenedor" >>
/data/registro.txt
# Mostrar contenido
cat /data/registro.txt
echo ""
echo "Archivo guardado en volumen persistente"
```

Paso 5: Crear script de inicialización de base de datos

La base de datos en mysql (contenedor) puede ser pre cargada, para esto es necesario contar

Crea `init-db/01-create-tables.sql`:

```
-- Script de inicialización de base de datos
CREATE TABLE IF NOT EXISTS usuarios (
id INT AUTO_INCREMENT PRIMARY KEY,
nombre VARCHAR(100),
email VARCHAR(100),
fecha_registro TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Insertar datos de prueba
INSERT INTO usuarios (nombre, email) VALUES
('Juan Pérez', 'juan@ejemplo.com'),
('María García', 'maria@ejemplo.com'),
('Carlos López', 'carlos@ejemplo.com');

-- Mostrar tablas creadas
SELECT 'Tablas creadas exitosamente' AS mensaje;
```

Paso 6: Crear docker-compose.yml completo

```
version: '3.8'

services:

  web:
    image: nginx:alpine
    container_name: web-volumenes
    ports:
      - "8080:80"
    volumes:
      - ./web/html:/usr/share/nginx/html:ro # ro = read-only
      - ./config/nginx.conf:/etc/nginx/conf.d/default.conf:ro
      - ./logs:/var/log/nginx

      - nginx-cache:/var/cache/nginx
    depends_on:
      - database

  database:
    image: mysql:8.0
    container_name: mysql-volumenes
    environment:
      MYSQL_ROOT_PASSWORD: root123
      MYSQL_DATABASE: practica_volumenes
      MYSQL_USER: usuario
      MYSQL_PASSWORD: password123
    ports:
      - "3306:3306"
    volumes:
      - mysql-data:/var/lib/mysql
      - ./init-db:/docker-entrypoint-initdb.d:ro

  data-generator:
    image: alpine:latest
    container_name: generador-datos
    volumes:
      - shared-data:/data
      - ./generate-data.sh:/scripts/generate.sh
    command: sh -c "chmod +x /scripts/generate.sh &&
/scripts/generate.sh && tail -f /dev/null"


  data-reader:
```

```
    image: alpine:latest
    container_name: lector-datos
    volumes:
      - shared-data:/data:ro
    command: sh -c "echo 'Esperando datos...' && sleep 5 && echo '===
Contenido compartido ===' && cat /data/registro.txt && tail -f
/dev/null"
    depends_on:
      - data-generator


volumes:
  mysql-data:
    driver: local
    name: mysql-data

  nginx-cache:
    driver: local
    name: nginx-cache

  shared-data:
    driver: local
    name: shared-data
```

Paso 7: Ejecutar y explorar

```
chmod +x generate-data.sh
docker compose up -d
docker compose logs
docker compose logs data-generator
docker compose logs data-reader
```

Paso 8: Verificar funcionamiento

```
# 1. Verificar página web
Abrir en navegador: http://localhost:8080

# 2. Generar logs de prueba
Abrir la URL: http://localhost:8080/api/test

# 3. Ver logs generados en el host
cat logs/access.log
cat logs/api-access.log
```

```
# 4. Verificar volúmenes creados
docker volume ls | grep data

# 5. Inspeccionar un volumen
docker volume inspect mysql-data

# 6. Ver contenido del volumen compartido
docker compose exec data-reader cat /data/registro.txt
```

Paso 9: Probar persistencia de datos

Verifica que la base de datos cuenta con la información ingresada en la tabla usuarios:

```
docker compose exec database mysql -uroot -proot123 practica_volumenes
```

Verificaremos que usuarios se encuentran en la base de datos y posteriormente insertaremos un registro.

```
SELECT * FROM usuarios;

INSERT INTO usuarios (nombre, email) VALUES ('Nuevo Usuario',
'nuevo@ejemplo.com');

SELECT * FROM usuarios;
```

Paso 10: Limpieza completa

Para finalizar los contenedores ejecutar los comandos siguientes:

```
docker compose down
docker volume ls | grep data
```

Preguntas complementarias

1. ¿Cómo Mysql carga y genera de manera automática la BD y su contenido?
2. ¿En que locación dentro el contenedor de mysql se encuentra la base de datos?
3. ¿Donde se esta almacenando la base de datos de mysql en el servidor?

Para entregar:

1. Agregar un servicio con phpmyadmin para verificar la base de datos.
2. Agregar otro servicio de base de datos con su volumen correspondiente el cual deberá ser accesible desde el phpmyadmin creado en el punto 1.
3. Reiniciar el contenedor stop / start y verificar si la tabla permanece después del reinicio.
4. Eliminar la tabla y posteriormente detener los contenedores y volverlos a crear (stop, start).
5. ¿Qué sucedió con la tabla?¿Está presente?