

03 - Volúmenes

Persistir datos y compartir archivos

Objetivo

Aprender a usar volúmenes para persistir datos y compartir archivos entre el host y el contenedor.

¿Qué aprenderás?

- Tipos de volúmenes: bind mounts y named volumes
- Cómo persistir datos de bases de datos
- Compartir código entre host y contenedor para desarrollo

¿Por qué necesitamos Volúmenes?

Por defecto, los datos dentro de un contenedor se **pierden** cuando el contenedor se elimina.

Solución: Usar volúmenes para **persistir** datos fuera del contenedor.

Conceptos Clave

Bind Mount

Monta un directorio o archivo del **host** en el contenedor.

- **Uso:** Desarrollo, compartir archivos
- **Ubicación:** Ruta específica en tu máquina

Named Volume

Volumen **gestionado por Docker**, independiente del sistema de archivos del host.

- **Uso:** Bases de datos, datos de producción
- **Ubicación:** Gestionada por Docker

Analogía Simple

Bind Mount = Carpeta compartida entre tu PC y el contenedor

Named Volume = Disco externo gestionado por Docker

Ejercicio 1: Bind Mount

Ejecutar con bind mount

```
# Crear directorio en el host
mkdir -p ./data

# Ejecutar contenedor montando el directorio
docker run --rm -v $(pwd)/data:/data \
  python:3.9-slim python -c "
import os
from datetime import datetime

data_dir = '/data'
os.makedirs(data_dir, exist_ok=True)

timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
message = f'Contenedor ejecutado el: {timestamp}\n'

log_file = os.path.join(data_dir, 'log.txt')
with open(log_file, 'a') as f:
    f.write(message)

print(f'Log guardado en: {log_file}')
```

Explicación:

- `-v $(pwd)/data:/data`: Monta `./data` del host en `/data` del contenedor
- Los datos persisten después de que el contenedor se elimina

Ejercicio 2: Named Volume

Ejecutar PostgreSQL con volumen

```
# Crear un volumen nombrado
docker volume create postgres-data

# Ejecutar PostgreSQL usando el volumen
docker run -d \
  --name postgres-db \
  -e POSTGRES_PASSWORD=mi_password \
  -e POSTGRES_DB=mi_db \
  -v postgres-data:/var/lib/postgresql/data \
  postgres:15
```

Ventajas:

- Gestionado por Docker
- Independiente del sistema de archivos del host
- Ideal para bases de datos

Ejercicio 3: Desarrollo con Hot-Reload

Ejecutar con bind mount para desarrollo

```
docker run -d \  
  -p 3000:3000 \  
  -v $(pwd):/app \  
  -v /app/node_modules \  
  node:18-alpine npm start
```

Explicación:

- `-v $(pwd):/app` : Monta el código actual
- `-v /app/node_modules` : Volumen anónimo para node_modules

Gestionar Volúmenes

```
# Listar volúmenes
docker volume ls

# Inspeccionar un volumen
docker volume inspect postgres-data

# Eliminar un volumen (¡cuidado, se pierden los datos!)
docker volume rm postgres-data

# Eliminar volúmenes no usados
docker volume prune
```

Casos de Uso

Tipo	Cuándo Usar
Bind Mount	Desarrollo local, compartir configuración
Named Volume	Bases de datos, datos que deben persistir
tmpfs	Datos temporales en memoria (solo Linux)

Comandos Clave

- `-v /ruta/host:/ruta/container` : Bind mount
- `-v nombre-volumen:/ruta/container` : Named volume
- `docker volume create` : Crear volumen nombrado
- `docker volume ls` : Listar volúmenes
- `docker volume inspect` : Ver detalles de un volumen

Práctica

1. Crea un bind mount para desarrollo
2. Crea un named volume para PostgreSQL
3. Verifica que los datos persisten
4. Lista y gestiona volúmenes

Siguiente Paso

Aprende sobre redes Docker para conectar contenedores.

Módulo 04: Redes

Preguntas?

¡Tiempo para preguntas y práctica!