Taller: Volúmenes y Bind Mounts en Docker (Linux/macOS)

David Fernando Cifuentes Bohórquez

UNIVERSIDAD PEDAGOGICA Y TECNOLOGICA DE COLOMBIA
INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
BASES DE DATOS II
TUNJA
2025

Ejercicio 1 - Bind mount en modo lectura con Nginx

1. Cree una carpeta y un archivo:
    1. Crear carpeta:

```
~ mkdir -p ~/web
```
```
~ cd web/
```

    2. Crear archivo:

```
~ echo "<h1>Hola desde bind mount</h1>" > ~/web/index.html
```
```
➜ web ls
index.html
```

2. Levante Nginx con bind mount de solo lectura:

    Al no tener la imagen alpine de **ngnix**, se hace el pull para tenerla instalada en mi host y después hacer el levantamiento.

```
➜ ~ docker run -d --name web-ro -p 8080:80 \
  -v ~/web:/usr/share/nginx/html:ro \
  nginx:alpine

Unable to find image 'nginx:alpine' locally
alpine: Pulling from library/nginx
9824c27679d3: Already exists
6bc572a340ec: Pull complete
403e3f251637: Pull complete
9adfbae99cb7: Pull complete
7a8a46741e18: Pull complete
c9ebe2ff2d2c: Pull complete
a992fbc61ecc: Pull complete
cb1ff4086f82: Pull complete
Digest: sha256:42a516af16b852e33b7682d5ef8acbd5d13fe08fecadc7ed98605ba5e3b26ab8
Status: Downloaded newer image for nginx:alpine
96076adb69c98b2cdf8ae5e397255682d93c79f7fd3a6766d6b681127d7f7a5c
```

3. Abra http://localhost:8080 y verifique

4. Edite index.html en el host y recargue el navegador, el cambio debe verse.

```
→ ~ echo "<h1> SI se ve el cambio </h1>" > ~/web/index.html
```

5. Intente crear un archivo dentro del contenedor:

```
→ ~ docker exec -it web-ro sh -lc 'echo test > /usr/share/nginx/html/test.txt'
sh: can't create /usr/share/nginx/html/test.txt: Read-only file system
```

Ejercicio 2 — Named volume con PostgreSQL

1. Crear el volumen:

```
→ ~ docker volume create pgdata
pgdata
```

2. Ejecute PostgreSQL: Al no tener la imagen alpine de postgres, se hace el pull para tenerla instalada en mi host y después hacer el levantamiento.

```
→ ~ docker run -d --name pg -e POSTGRES_PASSWORD=postgres \
  -p 5432:5432 \
  -v pgdata:/var/lib/postgresql/data \
  postgres:16-alpine

Unable to find image 'postgres:16-alpine' locally
16-alpine: Pulling from library/postgres
9824c27679d3: Already exists
01ef787617d5: Pull complete
d444581c5dc1: Pull complete
127625cab66d: Pull complete
7f8bf47818a2: Pull complete
0951477387e1: Pull complete
878e28e3ecd5: Pull complete
d079e32a74cc: Pull complete
cb87d3c01966: Pull complete
40af0ccd9733: Pull complete
0b003ba20c51: Pull complete
Digest: sha256:8ffca822c1933bdc8be7dbbe9c2330974bdb43f5027f47717772fa35925412b0
Status: Downloaded newer image for postgres:16-alpine
e9fef2feecc8358fb210f72f36806056d6c3764a02243ba0a8bb0462ecd135d0
```

3. Cree una tabla y agregue datos:

```
→ ~ docker exec -it pg psql -U postgres -c "CREATE TABLE
test(id serial, nombre text);"
CREATE TABLE
→ ~ docker exec -it pg psql -U postgres -c "INSERT INTO test(nombre) VALUES ('Ada'),('Linus');"
INSERT 0 2
→ ~ docker exec -it pg psql -U postgres -c "SELECT * FROM test;"
 id | nombre
----+--------
  1 | Ada
  2 | Linus
(2 rows)
```

4. Elimine el contenedor:

```
→ ~ docker rm -f pg
pg
```

5. Vuelva a levantarlo usando el mismo volumen y verifique que los datos siguen allí.

```
→ ~ docker run -d --name pg -e POSTGRES_PASSWORD=postgres \
  -p 5432:5432 \
  -v pgdata:/var/lib/postgresql/data \
  postgres:16-alpine
bc45a27593227519f8324aa3c3e5d7905c42ca0bdb7e7ebe120135b8cd6508d3
→ ~ docker exec -it pg psql -U postgres -c "SELECT * FROM test;"
 id | nombre
----+--------
  1 | Ada
  2 | Linus
(2 rows)
```

Ejercicio 3 — Volumen compartido entre dos contenedores

1) Cree un volumen:

```
→ ~ docker volume create sharedlogs
sharedlogs
```

2) Lanzar el productor (escribe timestamps cada segundo):

Al no tener la imagen alpine necesaria, se hace el pull para tenerla instalada en mi host y después hacer el levantamiento

```
→ ~ docker run -d --name writer -v sharedlogs:/data \
  alpine:3.20 sh -c 'while true; do date >> /data/log.txt; sleep 1; done'

Unable to find image 'alpine:3.20' locally
3.20: Pulling from library/alpine
01d036902a3c: Pull complete
Digest: sha256:b3119ef930faabb6b7b976780c0c7a9c1aa24d0c75e9179ac10e6bc9ac080d0d
Status: Downloaded newer image for alpine:3.20
7362c9fd730b7527918c343dc3b3b6a4202220cfad831adc5923962dcaadad4a
```

3) Lanzar el consumidor (lee en tiempo real):

```
➜ ~ docker run -it --rm --name reader -v sharedlogs:/data \
  alpine:3.20 tail -f /data/log.txt

Wed Sep  3 00:42:03 UTC 2025
Wed Sep  3 00:42:04 UTC 2025
Wed Sep  3 00:42:05 UTC 2025
Wed Sep  3 00:42:06 UTC 2025
Wed Sep  3 00:42:07 UTC 2025
Wed Sep  3 00:42:08 UTC 2025
Wed Sep  3 00:42:09 UTC 2025
Wed Sep  3 00:42:10 UTC 2025
Wed Sep  3 00:42:11 UTC 2025
Wed Sep  3 00:42:12 UTC 2025
Wed Sep  3 00:42:13 UTC 2025
Wed Sep  3 00:42:14 UTC 2025
Wed Sep  3 00:42:15 UTC 2025
Wed Sep  3 00:42:16 UTC 2025
Wed Sep  3 00:42:17 UTC 2025
Wed Sep  3 00:42:18 UTC 2025
```

4) Reinicie el productor y revise que el archivo siga creciendo:

   a) Eliminar y recrear el productor

```
➜ ~ docker rm -f writer
writer
➜ ~ docker run -d --name writer -v sharedlogs:/data \
  alpine:3.20 sh -c 'while true; do date >> /data/log.txt; sleep 1; done'

1972e9681bc56637a0d459c9f707471cc3c1388987132e8e61a0139d7480eae0
```

   b) Revisar que el archivo sigue creciendo:

```
➜ ~ docker run --rm -v sharedlogs:/data alpine:3.20 sh -lc 'tail -n 3 /data/log.txt'
Wed Sep  3 00:44:48 UTC 2025
Wed Sep  3 00:44:49 UTC 2025
Wed Sep  3 00:44:50 UTC 2025
```

Ejercicio 4 — Backup y restauración de un volumen

1. Crear un volumen y añadir un archivo:

```
➜ ~ docker volume create appdata
appdata
➜ ~ docker run --rm -v appdata:/data alpine:3.20 \
  sh -lc 'echo "backup-$(date +%F)" > /data/info.txt'
```

2. Haga backup a un tar en el host:

```
➜ ~ mkdir -p ~/backups
➜ ~ docker run --rm -v appdata:/data:ro -v ~/backups:/backup \
  alpine:3.20 sh -lc 'cd /data && tar czf /backup/appdata.tar.gz .'
```

3. Restaure en un nuevo volumen:

```
➜ ~ docker volume create appdata_restored
appdata_restored
➜ ~ docker run --rm -v appdata_restored:/data -v ~/backups:/backup \
  alpine:3.20 sh -lc 'cd /data && tar xzf /backup/appdata.tar.gz'
```

4. Verifique el contenido restaurado:

```
➜ ~ docker run --rm -v appdata_restored:/data alpine:3.20 cat /data/info.txt
backup-2025-09-03
```

Pequeña reflexión:

En este taller aprendí que un bind mount el contenido es controlado por el host y como con el :ro podemos evitar modificaciones desde el contenedor, además aprendí que un volumen nombrado puede guardar los datos fuera del ciclo de vida del contenedor, así se garantiza la persistencia después de eliminar el contenedor. En el tercer ejercicio pude ver como varios contenedores pueden compartir el mismo volumen, uno produciendo datos y el otro consumiendolos, y ese volumen mantiene la información incluso cuando se reinician los contenedores, y finalmente se hizo el respaldo y restauración de un volumen usando “tar”. Los principales problemas fueron la instalación de los alpines que no tenía instalados, y la aparición de nuevos conceptos en este taller, sin embargo con una pequeña investigación quedaron claros estos nuevos conceptos.