NOMBRE: Julian Bayona

TALLER: Bind mount

Ejercicio 1 — Bind mount en modo lectura con Nginx

1. Cree una carpeta y un archivo:

```
~/web  (0.061s)
echo "<h1>Hola desde bind mount </h1>" > ~/web/index.html

~/web  (0.059s)
cat index.html
<h1>Hola desde bind mount </h1>
```

2. Levante Nginx con bind mount de solo lectura

```
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
exialpine: 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
c76974956d516a5f1c0a68ee79d5e7b7c98a658d9413038cef88e79ba9e0846c
```

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

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

```
~/web  (0.063s)
echo "<h1>El contenido ha cambiado</h1>" > ~/web/index.html
```

```
curl -X GET http://localhost:8080/
<h1>El contenido ha cambiado</h1>
```

5. Intente crear un archivo dentro del contenedor:

```
~/web  (0.214s)
docker exec -it web-ro sh -c '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

2.1. Cree un volumen:

```
~/web  (0.099s)
docker volume create pgdata
pgdata
```

2.2. Ejecute PostgreSQL:

```
docker run -d --name pg -e POSTGRES_PASSWORD=postgres -p 5432:5432 -v pgdata:/var/lib/postgresql/data postgres:16-alp
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
57dc68ea5c8fe361085f22ac15e865bbd355adfdeddee8b7c6f6ace2bd07f9f4
```

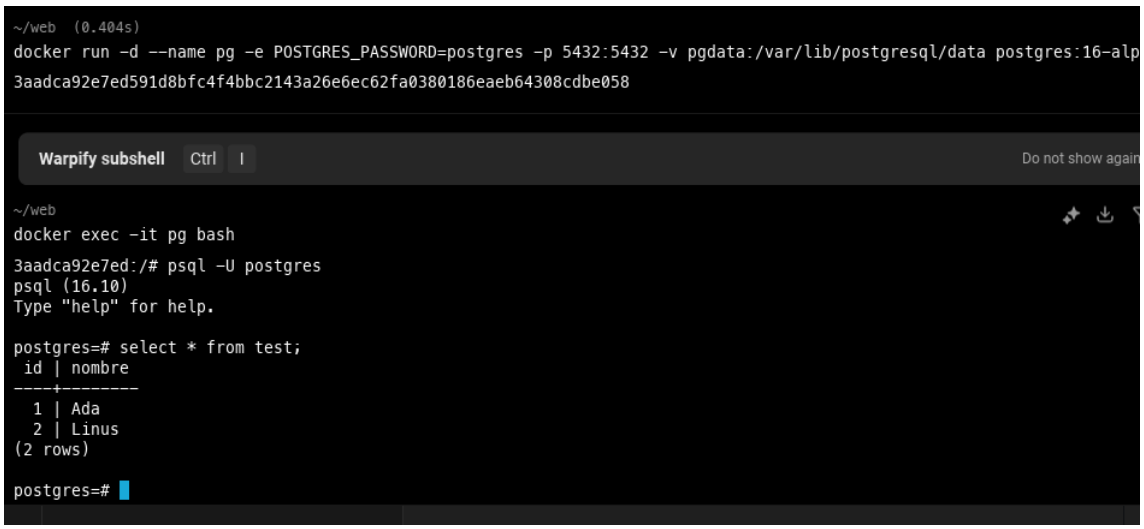
3. Cree una tabla y agregue datos

```
docker exec -it pg bash
57dc68ea5c8f:/# ^[[200~psql -U postgres
bash: $'\E[200~psql': command not found
57dc68ea5c8f:/# psql -U postgres
psql (16.10)
Type "help" for help.

postgres=# CREATE TABLE test(id serial, nombre text);
CREATE TABLE
```

```
postgres=# INSERT INTO
postgres-# test(nombre) VALUES ('Ada'),('Linus');
INSERT 0 2
postgres=# select * from test;
 id | nombre
----+--------
  1 | Ada
  2 | Linus
(2 rows)
```
\

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



3--- Volumen compartido entre dos contenedores

1. Cree un volumen:

```
docker volume create sharedlogs
sharedlogs
```

2. Productor (escribe timestamps cada segundo):

3. 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 19:47:27 UTC 2025
Wed Sep  3 19:47:28 UTC 2025
Wed Sep  3 19:47:29 UTC 2025
Wed Sep  3 19:47:30 UTC 2025
Wed Sep  3 19:47:31 UTC 2025
Wed Sep  3 19:47:32 UTC 2025
Wed Sep  3 19:47:33 UTC 2025
Wed Sep  3 19:47:34 UTC 2025
Wed Sep  3 19:47:35 UTC 2025
Wed Sep  3 19:47:36 UTC 2025
Wed Sep  3 19:47:37 UTC 2025
```

4. Reinicie el productor y revise que el archivo siga creciendo

```
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; don
d859c3f0c0289a813b619c664a8c3e26ad1307c9136d7dc30dfcd73702b51817
```

```
~
docker run -it --rm --name reader -v sharedlogs:/data alpine:3.20 tail -f /data/log.txt
Wed Sep  3 20:28:14 UTC 2025
Wed Sep  3 20:28:15 UTC 2025
Wed Sep  3 20:28:16 UTC 2025
Wed Sep  3 20:28:17 UTC 2025
Wed Sep  3 20:28:18 UTC 2025
Wed Sep  3 20:28:19 UTC 2025
Wed Sep  3 20:28:20 UTC 2025
Wed Sep  3 20:28:21 UTC 2025
Wed Sep  3 20:28:22 UTC 2025
```

```
docker exec -it writer sh
/ # ls
bin    dev    home   media  opt    root   sbin   sys    usr
data   etc    lib    mnt    proc   run    srv    tmp    var
/ # cd data/
/data # ls
log.txt
/data # nano log
sh: nano: not found
/data # nano log.txt
sh: nano: not found
/data # cat log.txt
```

Aqui podemos evidenciar como tras pausar el contenedor write y despues de activarlo pasado 8 minutos escribio en el mismo volumen pero a diferente hora

```
Wed Sep  3 20:19:56 UTC 2025
Wed Sep  3 20:19:57 UTC 2025
Wed Sep  3 20:19:58 UTC 2025
Wed Sep  3 20:27:51 UTC 2025
Wed Sep  3 20:27:52 UTC 2025
Wed Sep  3 20:27:53 UTC 2025
```

4 — Backup y restauración de un volumen

1. Cree un volumen y añade un archivo:

```
docker run --rm -v appdata:/data alpine:3.20 sh -c 'echo "datos importantes" > /data/info.txt'
```

Evidenciamos el contenido de nuestro volumen

```
~  (4.243s)
sudo ls -l /var/lib/docker/volumes/appdata/_data

[sudo] contraseña para julian:
total 4
-rw-r--r-- 1 root root 18 sep  3 16:05 info.txt
```

2. Haga backup a un tar en el host:

```
~  (0.059s)
mkdir -p ~/backups

~  (0.506s)
docker run --rm -v appdata:/data:ro -v ~/backups:/backup \
alpine:3.20 sh -c 'cd /data && tar czf /backup/appdata.tar.gz .'

~  (0.1s)
docker volume create appdata_restored
appdata_restored

~  (0.527s)
docker run --rm -v appdata_restored:/data -v ~/backups:/backup \
alpine:3.20 sh -c 'cd /data && tar xzf /backup/appdata.tar.gz'

~  (0.508s)
docker run --rm -v appdata_restored:/data alpine:3.20 cat /data/info.txt
datos importantes
```

Reflexión:
Con estos ejercicios puede entender de manera más concisa el uso y la diferencia entre un volumen y un bind mount y que usos puntuales se le pueden dar y la

manera practica de preservar nuestros datos independientemente de la subida y bajada de los contenedores.
Siento que el punto número 4 no lo comprendí a profundidad comparándolos con el resto de los puntos.