TALLER DE REPASO VOLÚMENES

1. **Bind mount en modo lectura con Nginx**

   1.1. Se ejecuto el comando

   mkdir -p ~/web & echo “<h1>Hola desde bind mount /h1>" > ~/web/index.html

   El cual me crea el cual me crea el archivo index.html com el cuerpo que le indique dentro del echo.

   1.2. Se ejecuto el comando

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

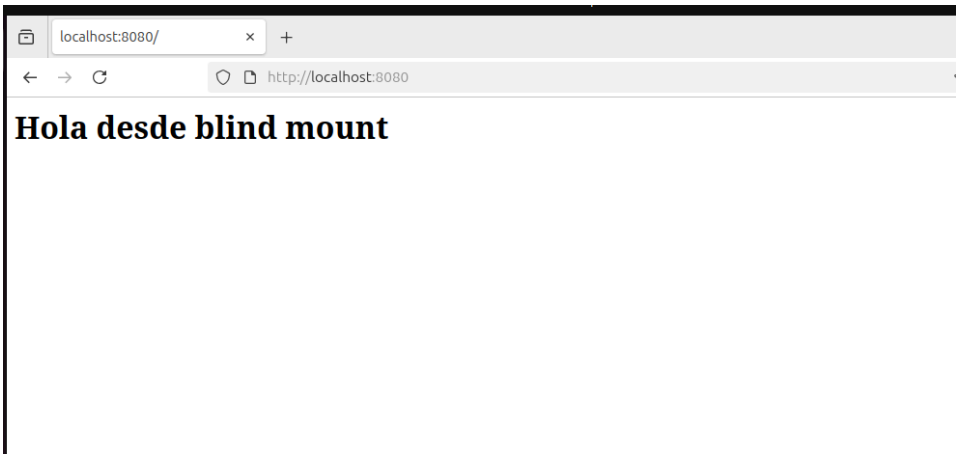
   El cual le indica a docker que cree un contenedor de nombre web-ro en el con puerto el 8080 para mi maquina y el 80 para mi contenedor y que además de esto lo ejecute con la versión alpine nginx

   1.3. Se verifica el id de contenedor y se procede a la prueba poniendo en el navegador la URL http://llocalhost:8080

   1.4.Cómo adicional se comprueba si se puede editar el archivo index.html y volver a ejecutar el contenedor lo cual no es posible ya que arroja el error Read-only file system detectando de nginx esta en modo lectura

```
ce is denied

Run 'docker run --help' for more information
daniel-reyes@daniel-reyes-VirtualBox:~$ 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
7190af6e400daa3cea0bd3301a2b08562b55a641ff2a02057943d67fd422343e
daniel-reyes@daniel-reyes-VirtualBox:~$ 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
daniel-reyes@daniel-reyes-VirtualBox:~$
```



2. Named volume con PostgreSQL

2.1. Se ejecuta el comando

docker volume create pgdata

El cual crea un volumen llamado pgdata

2.2. Ejecuta un contenedor de PostgreSQL y conéctalo al volumen pgdata. El comando es

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

2.3. Crea una tabla y agrega datos dentro de la base de datos usando docker exex

docker exec -it pg psql -U postgres -c "CREATE TABLE test(id serial, nombre text);" docker exec -it pg psql -U postgres -c "INSERT INTO test(nombre) VALUES ('Ada'),('Linus');" docker exec -it pg psql -U postgres -c "SELECT * FROM test;"

f
v
m

2.4. Elimina el contenedor con docker rm -f pg. Los datos persisten en el volumen pgdata aunque el contenedor ya no exista.

2.5. Vuelve a iniciar el contenedor con el mismo comando del paso 2 y verifica que los datos siguen.

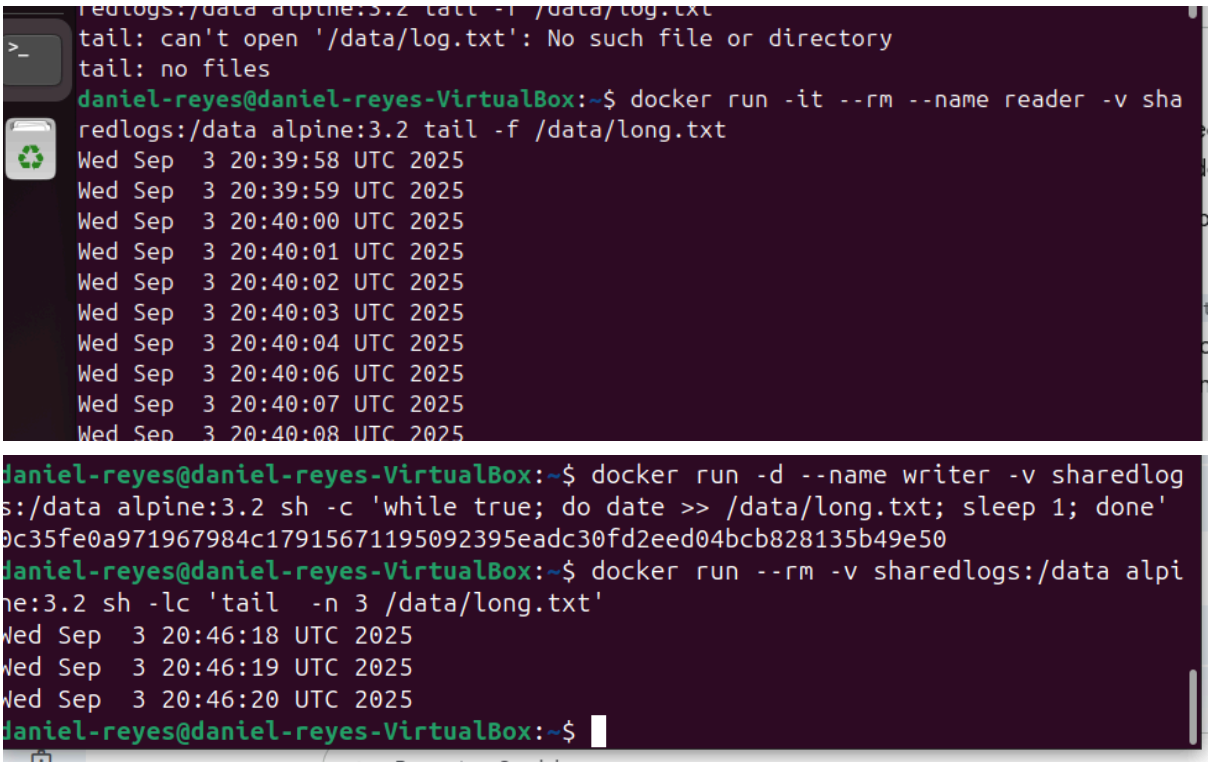
```
daniel-reyes@daniel-reyes-VirtualBox:~$ docker volume create pgdata
pgdata
```

```
Digest: sha256:8ffca822c1933bdc8be7dbbe9c2330974bdb43f5027f47717772fa35925412b0
Status: Downloaded newer image for postgres:16-alpine
2b1882ec88a6ec5056c8053ddd97422945af1660b351227c9e4e990f496be098
daniel-reyes@daniel-reyes-VirtualBox:~$ docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED          ST
ATUS         PORTS                                       NAMES
2b1882ec88a6   postgres:16-alpine   "docker-entrypoint.s…"   10 seconds ago   Up
 6 seconds   0.0.0.0:5432->5432/tcp, [::]:5432->5432/tcp   pg
daniel-reyes@daniel-reyes-VirtualBox:~$ docker exec -it pg psql -U postgres -c "
CREATE TABLE test(id serial, nombre text);"
CREATE TABLE
daniel-reyes@daniel-reyes-VirtualBox:~$ docker exec -it pg psql -U postgres -c "
INSERT INTO test(nombre) VALUES ('Ada'), ('Linus');"
INSERT 0 2
daniel-reyes@daniel-reyes-VirtualBox:~$ docker exec -it pg psql -U postgres -c "
SELECT * FROM test;"
 id | nombre
----+--------
  1 | Ada
  2 | Linus
(2 rows)

daniel-reyes@daniel-reyes-VirtualBox:~$
```

```
daniel-reyes@daniel-reyes-VirtualBox:~$ docker rm -f pg
pg
daniel-reyes@daniel-reyes-VirtualBox:~$ docker run -d --name pg -e POSTGRES_PASS
WORD=postgres -p 5432:5432 -v pgdata:/var/lib/postgresql/data postgres:16-alpine
aa9e9a6698bf13577b6371c3308d2707fae501867a7c8df8e2dfe4798e923278
daniel-reyes@daniel-reyes-VirtualBox:~$ docker exec -it pg psql -U postgres -c "
SELECT * FROM test;"
 id | nombre
----+--------
  1 | Ada
  2 | Linus
(2 rows)

daniel-reyes@daniel-reyes-VirtualBox:~$
```



3. **Volumen compartido entre dos contenedores**

3.1. Crea un volumen llamado sharedlogs con docker volume create sharedlogs

3.2. Inicia el contenedor "productor" (writer) que escribe la fecha actual en un archivo de registro cada segundo. El comando es

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

Pero el archivo no se encuentra aún y según búsquedas este demora un poco en aparecer

3.3.Inicia el contenedor "consumidor" (reader) que lee el archivo de registro en tiempo real con el comando tail -f. El comando es:

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

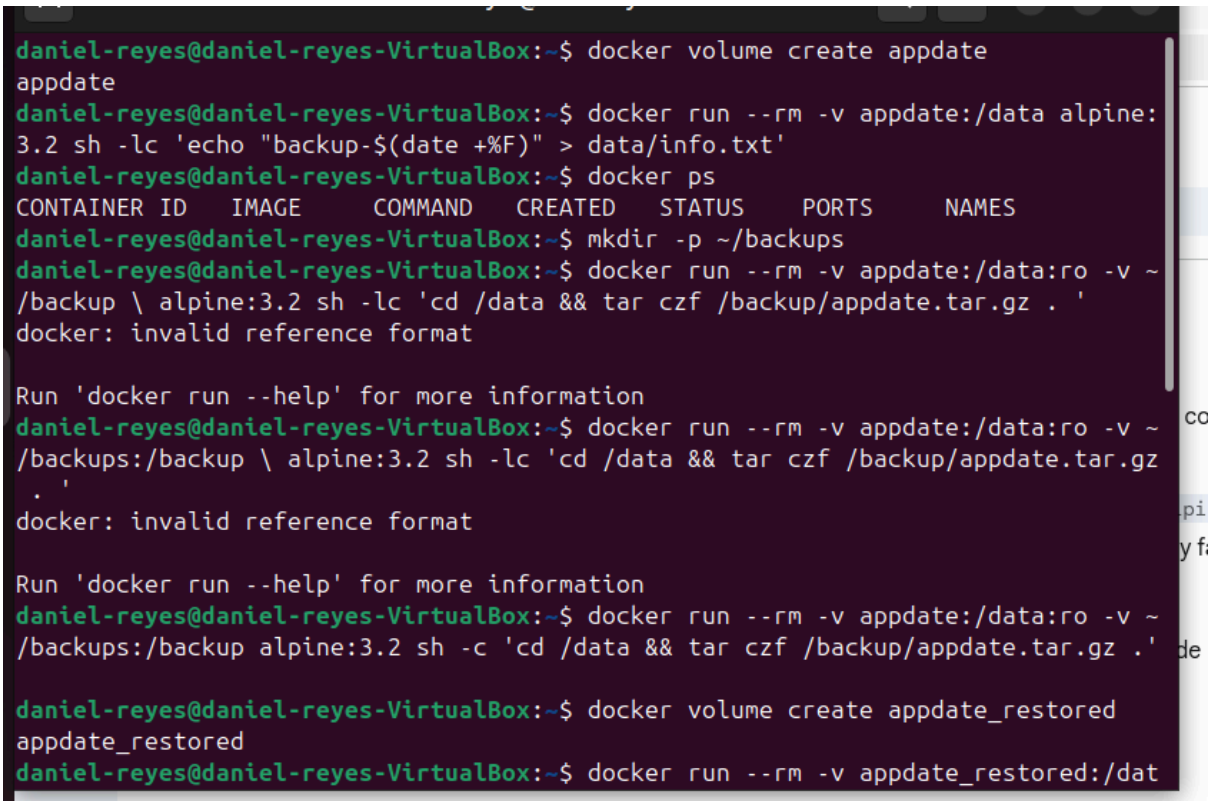
3.4.Reinicie el productor y revise que el archivo siga creciendo:

docker rm -f writer docker run -d -name writer -v sharedlogs:/data \ alpine:3.20 sh -c 'while true; do date > /data/log.txt; sleep 1; done' docker run -rm -v sharedlogs:/data alpine:3.20 sh -lc 'tail -n 3 /data/log.txt'

```
daniel-reyes@daniel-reyes-VirtualBox:~$ docker run -d --name writer -v sharedlog
s:/data \alpine:3.2 sh -c 'while true; do date >> /data/long.txt; sleep 1; done'
Unable to find image 'alpine:3.2' locally
3.2: Pulling from library/alpine
95f5ecd24e43: Pull complete
Digest: sha256:e9a2035f9d0d7cee1cdd445f5bfa0c5c646455ee26f14565dce23cf2d2de7570
Status: Downloaded newer image for alpine:3.2
11e855fefedb02e8935013c30542708126e668739e326bdb7c1860c237bdb1ba
daniel-reyes@daniel-reyes-VirtualBox:~$ docker ps
CONTAINER ID   IMAGE        COMMAND                  CREATED         STATUS
    PORTS     NAMES
11e855fefedb   alpine:3.2   "sh -c 'while true; …"   5 seconds ago   Up 4 second
s             writer
daniel-reyes@daniel-reyes-VirtualBox:~$
```

```
daniel-reyes@daniel-reyes-VirtualBox:~$ docker run -it --rm --name reader -v sh
redlogs:/data \alpine:3.2 tail -f /data/log.txt
tail: can't open '/data/log.txt': No such file or directory
tail: no files
daniel-reyes@daniel-reyes-VirtualBox:~$
```

4. **Volumen compartido entre dos contenedores**
   4.1. Crear el volumen llamado appdata con: docker volume create appdata
   4.2. Crear un archivo con el comando docker run -rm -v appdata:/data alpine:3.20 sh -lc 'echo "backup-$(date +%F)" > data/info.txt'
   4.3. Crear un respaldo(backup) creando un archivo .tar del volumen en tu host. Esto se hace con un contenedor temporal que monta ambos directorios y comprime los datos. con el comando: 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 .'
   4.4. Restaura los datos en un nuevo volumen. Primero, crea el volumen appdata_restored y luego usa un contenedor temporal para descomprimir el archivo de respaldo en este nuevo volumen. Con el comando docker volume create 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'
   4.5. Verifica el contenido restaurado ejecutando el comando docker run --rm -v appdata_restored:/data alpine:3.20 cat /data/info.txt. El contenido del archivo debe ser el mismo que el original

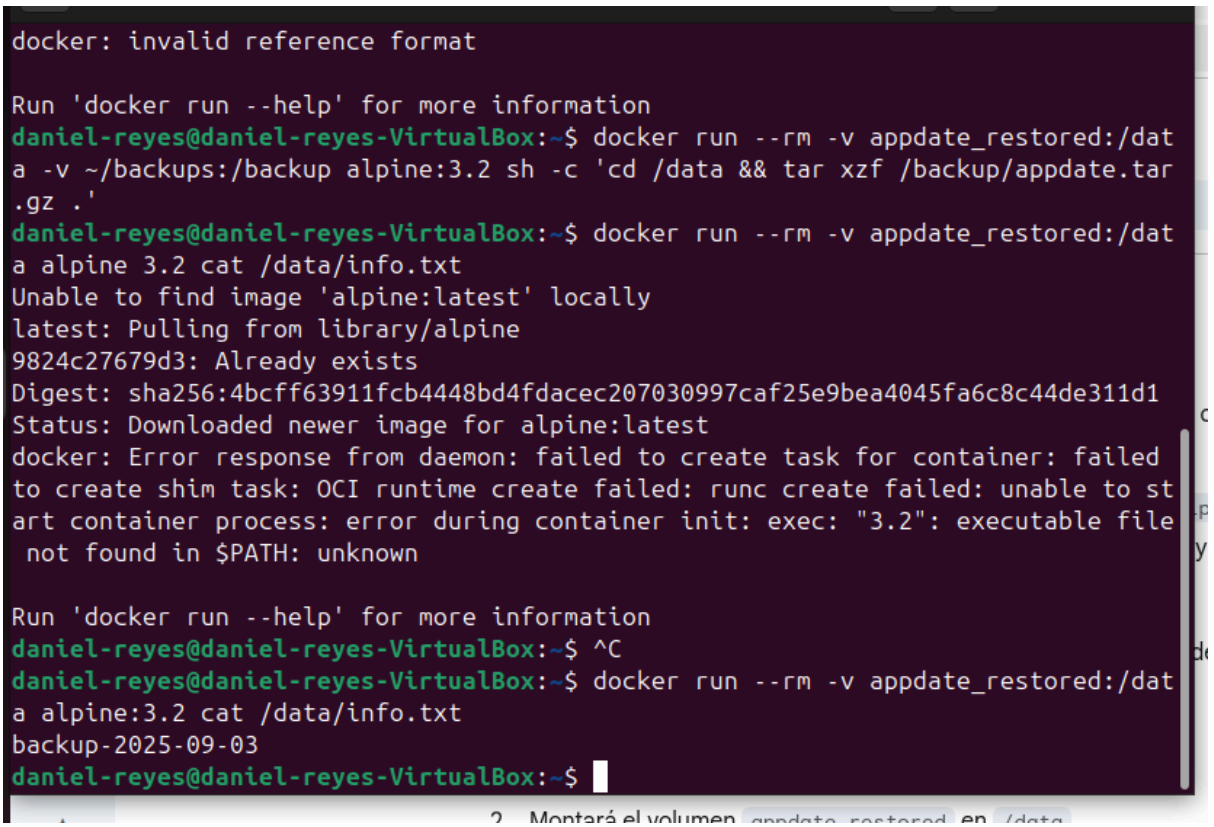


```
daniel-reyes@daniel-reyes-VirtualBox:~$ docker volume create appdate
appdate
daniel-reyes@daniel-reyes-VirtualBox:~$ docker run --rm -v appdate:/data alpine:
3.2 sh -lc 'echo "backup-$(date +%F)" > data/info.txt'
daniel-reyes@daniel-reyes-VirtualBox:~$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
daniel-reyes@daniel-reyes-VirtualBox:~$ mkdir -p ~/backups
daniel-reyes@daniel-reyes-VirtualBox:~$ docker run --rm -v appdate:/data:ro -v ~
/backup \ alpine:3.2 sh -lc 'cd /data && tar czf /backup/appdate.tar.gz . '
docker: invalid reference format

Run 'docker run --help' for more information
daniel-reyes@daniel-reyes-VirtualBox:~$ docker run --rm -v appdate:/data:ro -v ~
/backups:/backup \ alpine:3.2 sh -lc 'cd /data && tar czf /backup/appdate.tar.gz
 . '
docker: invalid reference format

Run 'docker run --help' for more information
daniel-reyes@daniel-reyes-VirtualBox:~$ docker run --rm -v appdate:/data:ro -v ~
/backups:/backup alpine:3.2 sh -c 'cd /data && tar czf /backup/appdate.tar.gz .'

daniel-reyes@daniel-reyes-VirtualBox:~$ docker volume create appdate_restored
appdate_restored
daniel-reyes@daniel-reyes-VirtualBox:~$ docker run --rm -v appdate_restored:/dat
```



```
docker: invalid reference format

Run 'docker run --help' for more information
daniel-reyes@daniel-reyes-VirtualBox:~$ docker run --rm -v appdate_restored:/dat
a -v ~/backups:/backup alpine:3.2 sh -c 'cd /data && tar xzf /backup/appdate.tar
.gz .'
daniel-reyes@daniel-reyes-VirtualBox:~$ docker run --rm -v appdate_restored:/dat
a alpine 3.2 cat /data/info.txt
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
9824c27679d3: Already exists
Digest: sha256:4bcff63911fcb4448bd4fdacec207030997caf25e9bea4045fa6c8c44de311d1
Status: Downloaded newer image for alpine:latest
docker: Error response from daemon: failed to create task for container: failed
to create shim task: OCI runtime create failed: runc create failed: unable to st
art container process: error during container init: exec: "3.2": executable file
 not found in $PATH: unknown

Run 'docker run --help' for more information
daniel-reyes@daniel-reyes-VirtualBox:~$ ^C
daniel-reyes@daniel-reyes-VirtualBox:~$ docker run --rm -v appdate_restored:/dat
a alpine:3.2 cat /data/info.txt
backup-2025-09-03
daniel-reyes@daniel-reyes-VirtualBox:~$
```

2. Montará el volumen appdate_restored en /data

REFLEXIÓN

Este taller me permitió entender a profundidad la gestión de datos en Docker, un aspecto crucial para cualquier aplicación. En cada ejercicio, aprendí un concepto diferente: con los bind mounts en el Ejercicio 1, comprendí cómo el host puede controlar el contenido de un contenedor en modo de solo lectura, lo que es útil para archivos de configuración o sitios estáticos. El Ejercicio 2 me enseñó la importancia de los volúmenes nombrados para asegurar la persistencia de datos de una base de datos, ya que estos persisten incluso después de que el contenedor se elimina. El Ejercicio 3 fue muy instructivo al mostrar un patrón de productor-consumidor, donde dos contenedores pueden escribir y leer del mismo volumen simultáneamente. Finalmente, el Ejercicio 4 me dio una habilidad práctica: cómo hacer un respaldo y restauración de un volumen, un proceso esencial para la gestión de datos y la recuperación ante desastres.

Los principales problemas que enfrenté fueron los errores de sintaxis en comandos largos, especialmente con las comillas y las barras invertidas, lo cual causaba errores como "invalid reference format". Resolví estos problemas revisando cuidadosamente cada línea y carácter, y en el caso del "tail", intentando el comando varias veces hasta que el archivo fuera creado por el otro contenedor. En general, este proceso de depuración me enseñó la importancia de la precisión y el pensamiento lógico al trabajar con Docker.