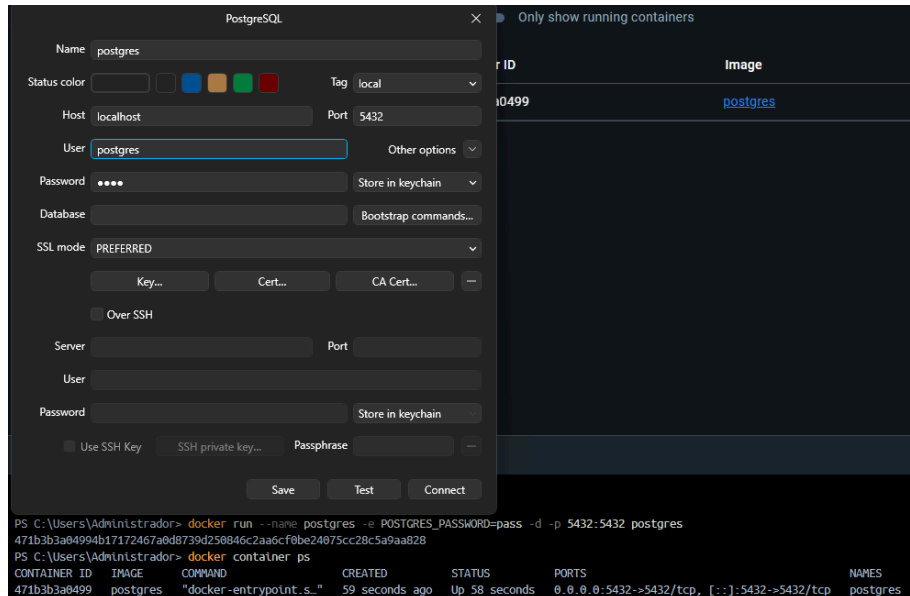


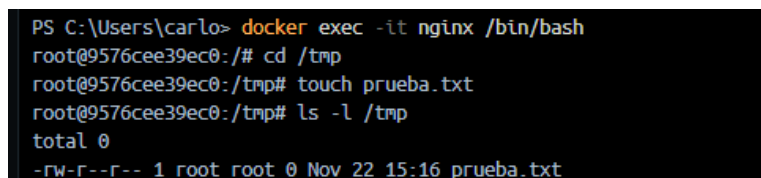
1. Usa una base de datos diferente

Crea un contenedor de MySQL o PostgreSQL. Configura usuario, contraseña y nombre de base de datos con variables de entorno. Comprueba que el contenedor arranca correctamente y que puedes conectarte desde una herramienta externa como TablePlus.



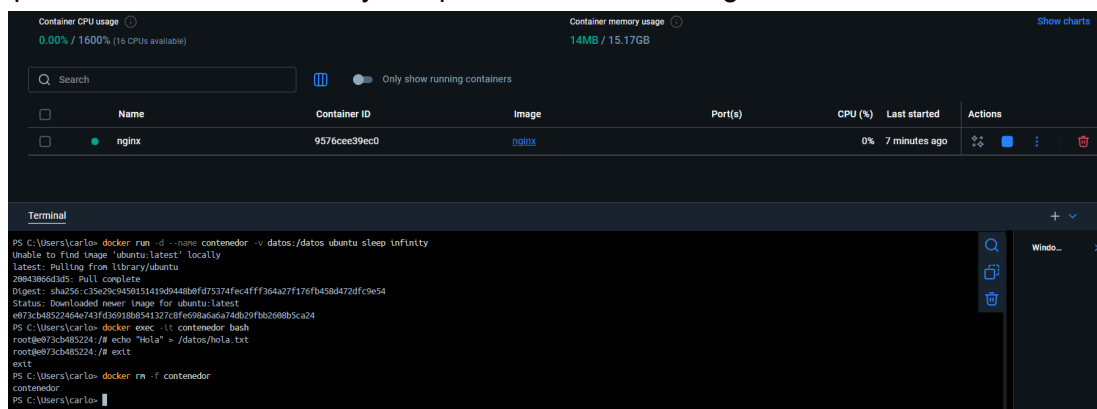
2. Explora el contenido de un contenedor

Ejecuta un contenedor de nginx o python, entra dentro de él con `docker exec -it` y mira qué hay en su sistema de archivos. Localiza los archivos principales y comprueba si hay alguna carpeta donde puedas crear un archivo temporal.



3. Prueba a modificar un volumen

Crea un contenedor con un volumen montado. Abre el contenedor, modifica o añade un archivo dentro del volumen y luego elimina el contenedor. Después, crea otro contenedor que use el mismo volumen y comprueba si el archivo sigue estando ahí.



```
PS C:\Users\carlo> docker run -d --name nuevo_contenedor -v datos:/datos ubuntu sleep infinity
9c4bb4e1c5057a9569ce6c94528c12e724a607b531c6950e63fcb62357dd07b0
PS C:\Users\carlo> docker exec -it nuevo_contenedor cat /datos/hola.txt
Hola
PS C:\Users\carlo>
```

4. Red entre dos contenedores

Crea una red con docker network create. Lanza dos contenedores dentro de esa red (por ejemplo, uno de nginx y otro de python). Comprueba si pueden comunicarse entre sí. Si necesitas hacer pruebas de conexión, instala el comando ping dentro de uno de ellos.

```
PS C:\Users\carlo> docker network create red
ab9cd5692324f96bd55ffd471511233ad2d8cb228e240c1b87c6795d06b259ce
PS C:\Users\carlo> docker run -d --name red_nginx --network red nginx
51ad53ac251796262204a64d796cc657b865a702944f0ab39cce18b331456071
PS C:\Users\carlo> docker run -d --name red_python --network red python:3-slim sleep infinity
74e7547abcefc646dc56981cc229eddd5ad499f511993d666a9403b9c9085af2
PS C:\Users\carlo> docker exec -it red_python bash
root@74e7547abcef:/# apt-get update && apt-get install -y inetutils-ping
Hit:1 http://deb.debian.org/debian trixie InRelease
Get:2 http://deb.debian.org/debian trixie-updates InRelease [47.3 kB]
Get:3 http://deb.debian.org/debian-security trixie-security InRelease [43.4 kB]
Get:4 http://deb.debian.org/debian trixie/main amd64 Packages [9670 kB]
```

```
root@74e7547abcef:/# ping red_nginx
PING red_nginx (172.18.0.2): 56 data bytes
64 bytes from 172.18.0.2: icmp_seq=0 ttl=64 time=0.099 ms
64 bytes from 172.18.0.2: icmp_seq=1 ttl=64 time=0.096 ms
64 bytes from 172.18.0.2: icmp_seq=2 ttl=64 time=0.222 ms
64 bytes from 172.18.0.2: icmp_seq=3 ttl=64 time=0.142 ms
64 bytes from 172.18.0.2: icmp_seq=4 ttl=64 time=0.053 ms
```

5. Página web simple con volumen local

Crea una carpeta local con un archivo index.html. Ejecuta un contenedor de nginx montando esa carpeta como volumen. Modifica el HTML desde tu ordenador y comprueba en el navegador cómo se actualiza automáticamente sin reiniciar el contenedor.



A screenshot of a web browser window. The address bar shows 'http://localhost:2020/'. The page content displays 'Hola mundo' in a simple black font. The browser's developer tools and other interface elements are visible but not the focus.

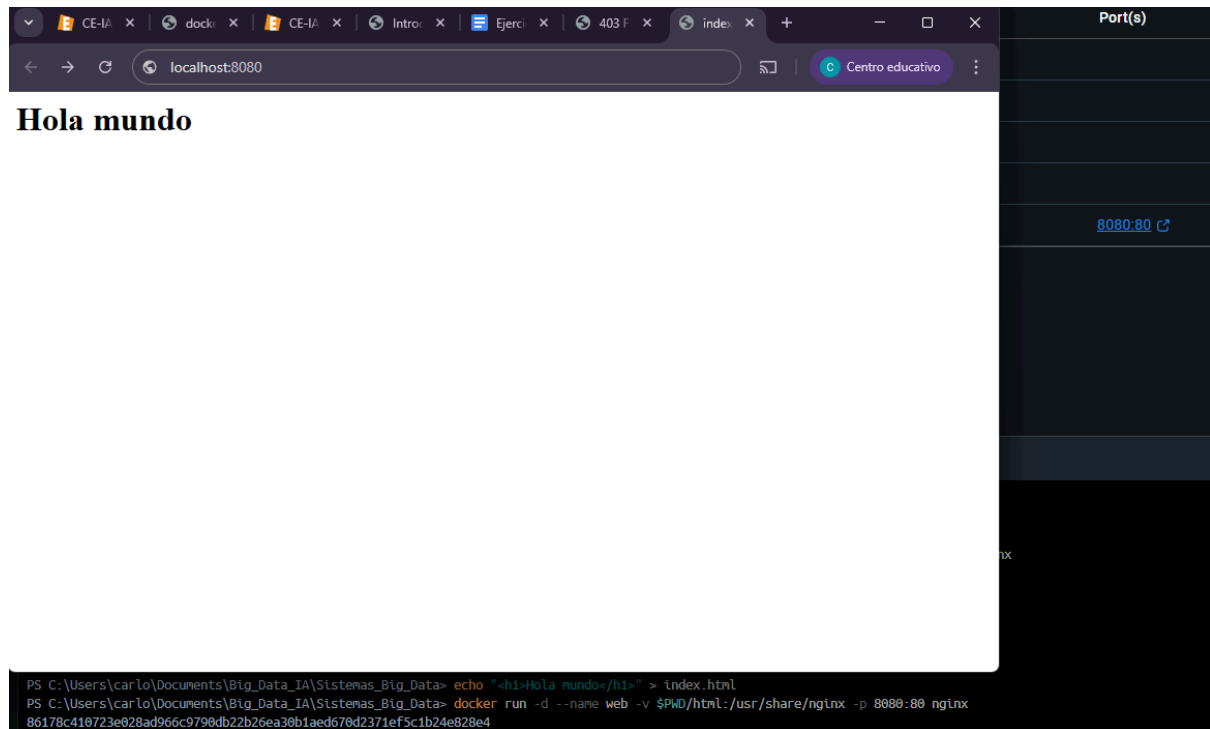
Hola mundo

```
PS C:\Users\carlo\Documents\Big_Data_IA\Sistemas_Big_Data> echo "¡Hola mundo!" > index.html
PS C:\Users\carlo\Documents\Big_Data_IA\Sistemas_Big_Data> docker run -d --name web -v $PWD/html:/usr/share/nginx/html -p 2020:20 nginx
39c4bb4e1c5057a9569ce6c94528c12e724a607b531c6950e63fcb62357dd07b0
```

6. Crea una pequeña API con FastAPI

Haz una aplicación sencilla en Python con FastAPI que tenga dos rutas. Empaquétala con un Dockerfile y ejecuta la imagen para acceder a la API desde el navegador. No hace falta que tenga base de datos, solo que responda correctamente a las peticiones.

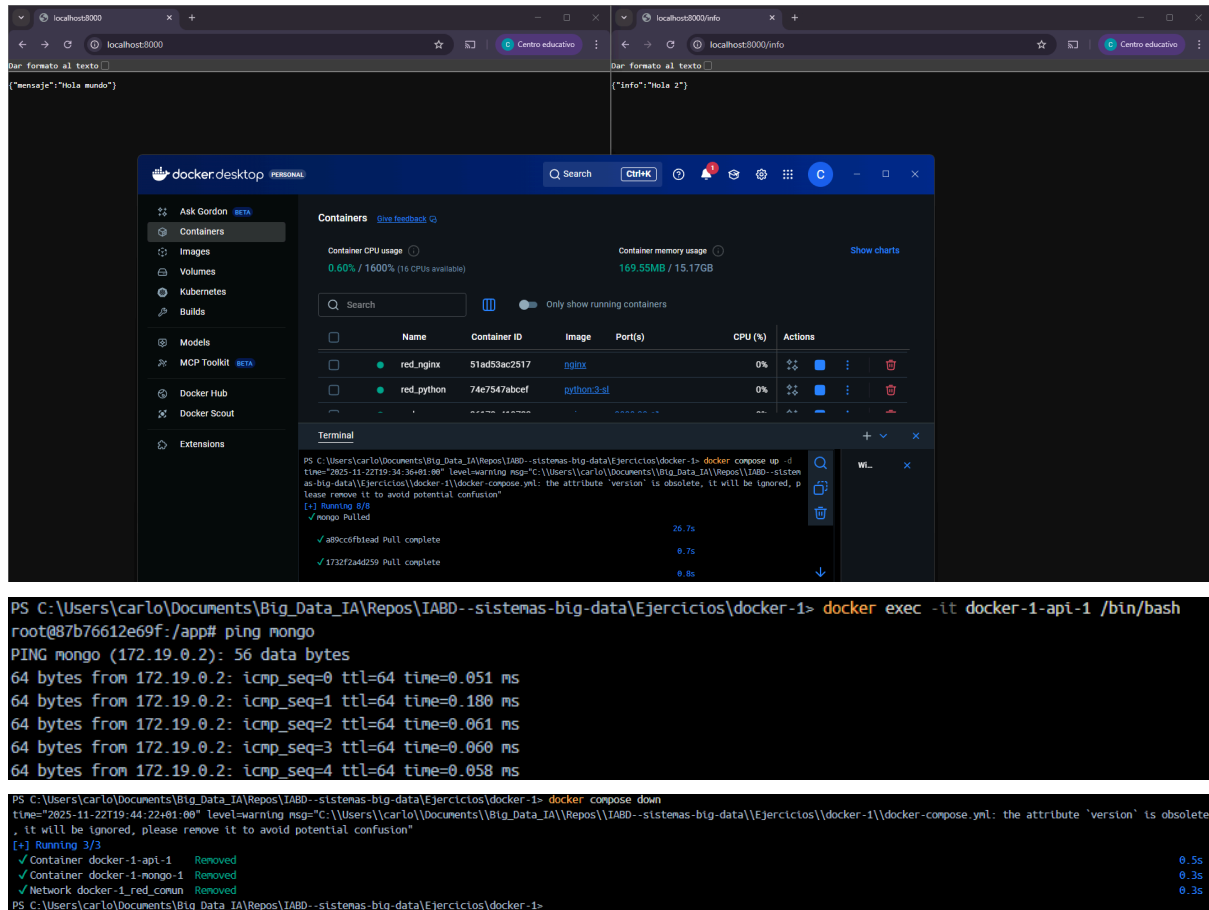
Para que funcione, hay que añadir los permisos `chmod o+rx /` y `chmod o+rx /`



(sigue en la siguiente página)

7. Usa Docker Compose para juntar dos servicios

Crea un archivo `docker-compose.yml` que levante una API y una base de datos (por ejemplo, FastAPI + MongoDB). Comprueba que ambos servicios arrancan y que están conectados a la misma red. Luego deténlos y elimínalos correctamente.



8. Limpieza del entorno

Has creado varios contenedores, redes y volúmenes, por lo tanto una vez finalizada la práctica limpia todo con los comandos de eliminación. Comprueba con `docker ps-a`, `docker images` y `docker volume ls` qué elementos quedan después de cada limpieza.

Al intentar eliminar los datos, solo se eliminan los que no se han creado manualmente ni tienen contenedores asociados.

```
PS C:\Users\carlo\Documents\Big_Data_IA\Repos\IABD--sistemas-big-data\Ejercicios\docker-1> docker rm -f $(docker ps -aq)
86178c410723
74e7547abcfe
51ad53ac2517
9c4bb4e1c505
9576cee39ec0
PS C:\Users\carlo\Documents\Big_Data_IA\Repos\IABD--sistemas-big-data\Ejercicios\docker-1> docker rmi -f $(docker images -q)
Untagged: docker-1-ap1:latest
Deleted: sha256:e332397cdfc4a852b87291e9cd185fb64bf84adea23329aaaca9be9f6855bce2
Untagged: mongo:latest
Deleted: sha256:7245ffb851d149dbfac67397caf91bae494d899972f9fd1db985f6cee31c13d
Untagged: python:latest
Deleted: sha256:6942ebef735aad5f708ef9c5e750cbe373bdc7751cee35c140e33764e34843ab9
Untagged: python:3-slim
Deleted: sha256:0aeca02dc3d4c5dbb624b753af084cafe41f5416e02193f1ce345d671ec966e
Untagged: nginx:latest
Deleted: sha256:553f64aecdc31b5bf944521731cd70e35da4faed96b2b7548a3d8e2598c52a42
Untagged: ubuntu:latest
Deleted: sha256:c35e29c9450151419d9448b0fd75374fec4fff364a27f16fb458d472dfc9e54
PS C:\Users\carlo\Documents\Big_Data_IA\Repos\IABD--sistemas-big-data\Ejercicios\docker-1> docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS          NAMES
PS C:\Users\carlo\Documents\Big_Data_IA\Repos\IABD--sistemas-big-data\Ejercicios\docker-1>
```

Docker 1, Carlos Robledo serradilla

```
PS C:\Users\carlo\Documents\Big_Data_IA\Repos\IABD--sistemas-big-data\Ejercicios\docker-1> docker network prune -f
Untagged: nginx:latest
Deleted: sha256:553f64aecdc31b5bf944521731cd70e35da4faed96b2b7548a3d8e2598c52a42
Untagged: ubuntu:latest
Deleted: sha256:c35e29c9450151419d9448b0fd75374fec4fff364a27f176fb458d472dfc9e54
PS C:\Users\carlo\Documents\Big_Data_IA\Repos\IABD--sistemas-big-data\Ejercicios\docker-1> docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
PS C:\Users\carlo\Documents\Big_Data_IA\Repos\IABD--sistemas-big-data\Ejercicios\docker-1> docker volume prune -f
Total reclaimed space: 0B
PS C:\Users\carlo\Documents\Big_Data_IA\Repos\IABD--sistemas-big-data\Ejercicios\docker-1> docker volume ls
DRIVER      VOLUME NAME
local       datos
local       docker-1_mongo_data
local       volumen
```