

### 1. ¿Para qué sirven los volúmenes?

Los volúmenes son una de las formas de persistencia de datos de contenedores. Estos nos permiten guardar los datos que queramos antes de eliminar un contenedor en una parte del sistema de ficheros gestionada por docker.

### 2. ¿Qué diferencias hay entre los volúmenes y los Bind Mount?

Los volúmenes los gestiona docker, de manera que solo él tendrá acceso y la zona reservada para los archivos cambiará según el sistema operativo. Para compartir datos entre contenedores o en una nube o hacer copias de seguridad será la mejor opción.

Los bind mount los gestiona el usuario, mapeando una parte de su sistema de ficheros con una parte del sistema de ficheros del contenedor. Esto permite compartir ficheros entre el host y los contenedores y que otras aplicaciones que no sean docker tengan acceso a esos ficheros.

Esta opción es la preferida para la fase de desarrollo porque los IDEs podrán acceder a los diferentes ficheros, se modificará con aplicaciones locales código que a la vez se encuentra en nuestro equipo y en el contenedor y se podrá probar en varios entornos a la vez sin tener que hacer instalaciones adicionales.

### 3. Crea un volumen sin nombre, ¿cómo lo ha llamado docker?. Muestra la información detallada del volumen y explica qué significa cada línea. Haz lo mismo con un volumen creado por el sistema (lístalos primero).

Si no se especifica un nombre, docker crea uno, poco apropiado para el uso frecuente, de manera automática.

```
estudiante@DAW1:~$ docker volume create
70778055ecb8b36e04e12d28b689d2d42c89a6c3096daa56f980d966207beadc
estudiante@DAW1:~$ docker volume ls
DRIVER      VOLUME NAME
local       70778055ecb8b36e04e12d28b689d2d42c89a6c3096daa56f980d966207beadc
local       a2f845397be0f3928ac72774aa5b7a9c3cc879953dbac175a617b54a6144c400
local       cdbdcb80c5425afd4b99700da524514fb78f6938b2f307b437749b48969dca32
local       d49e601d71e622f4be214b1b06171492a7e9e1db0c91a439d8ea431f459c237b
local       da3035de58f1146a3c1479c84eb65469f8a654e22d248b756f7be62a999cdae1
estudiante@DAW1:~$ docker volume inspect 70778055ecb8b36e04e12d28b689d2d42c89a6c3096daa56f980d966207beadc
[
  {
    "CreatedAt": "2022-10-04T10:32:37+02:00",
    "Driver": "local",
    "Labels": {},
    "Mountpoint": "/var/lib/docker/volumes/70778055ecb8b36e04e12d28b689d2d42c89a6c3096daa56f980d966207beadc/_data",
    "Name": "70778055ecb8b36e04e12d28b689d2d42c89a6c3096daa56f980d966207beadc",
    "Options": {},
    "Scope": "local"
  }
]
estudiante@DAW1:~$ docker volume inspect a2f845397be0f3928ac72774aa5b7a9c3cc879953dbac175a617b54a6144c400
[
  {
    "CreatedAt": "2022-09-27T08:47:29+02:00",
    "Driver": "local",
    "Labels": null,
    "Mountpoint": "/var/lib/docker/volumes/a2f845397be0f3928ac72774aa5b7a9c3cc879953dbac175a617b54a6144c400/_data",
    "Name": "a2f845397be0f3928ac72774aa5b7a9c3cc879953dbac175a617b54a6144c400",
    "Options": null,
    "Scope": "local"
  }
]
```

La información detallada nos muestra la fecha de creación, el tipo de driver, las etiquetas asociadas, el punto de montaje, el nombre del volumen, las opciones asociadas y el ámbito del volumen.

### 4. Elimina todos los volúmenes que hayas creado.

Utilizo docker volume prune y se borran los volúmenes que no están siendo utilizados por ningún contenedor.

5. **Arranca un Bind Mount usando la carpeta “web” del usuario como directorio raíz del servidor apache (Haz lo mismo con un volumen). Después obtén información del volumen y el bind mount y explica lo que se te muestra.**

Para arrancar con un bind mount:

```
docker run --name apache -v /home/usuario/web:/usr/local/apache2/htdocs -p 80:80 httpd
```

Para arrancar con un volumen “Data” creado anteriormente, ubicado en la carpeta raíz:

```
docker run --name apache -p 80:80 --mount type=volume,src=Data,dst=/usr/local/apache2/htdocs httpd
```

Con el comando `docker inspect idcontenedor` en la etiqueta `Mounts` encontramos la información del volumen sobre el tipo (volumen o bind mount), la ruta, el modo... con el siguiente formato:

```
{
  "Type": "bind",
  "Source": "/home/usuario/web",
  "Destination": "/usr/local/apache2/htdocs",
  "Mode": "",
  "RW": true,
  "Propagation": "rprivate"
}
```

6. **Arranca la versión más reciente del contenedor de ubuntu y comprueba que está “up”. Después páralo, comprueba que está parado. Por último, elimina el contenedor de ubuntu.**

Primero se ejecuta `docker run ubuntu` y como no la tengo se descarga la versión más reciente. Con `docker ps` comprobamos que está encendido, y para pararlo utilizamos `docker stop` con el nombre o id del contenedor (jovial\_newton, 896e7f082594). Comprobamos con `docker ps` de nuevo que está parado y lo eliminamos con `docker rm` (no se podrá eliminar si no está parado).

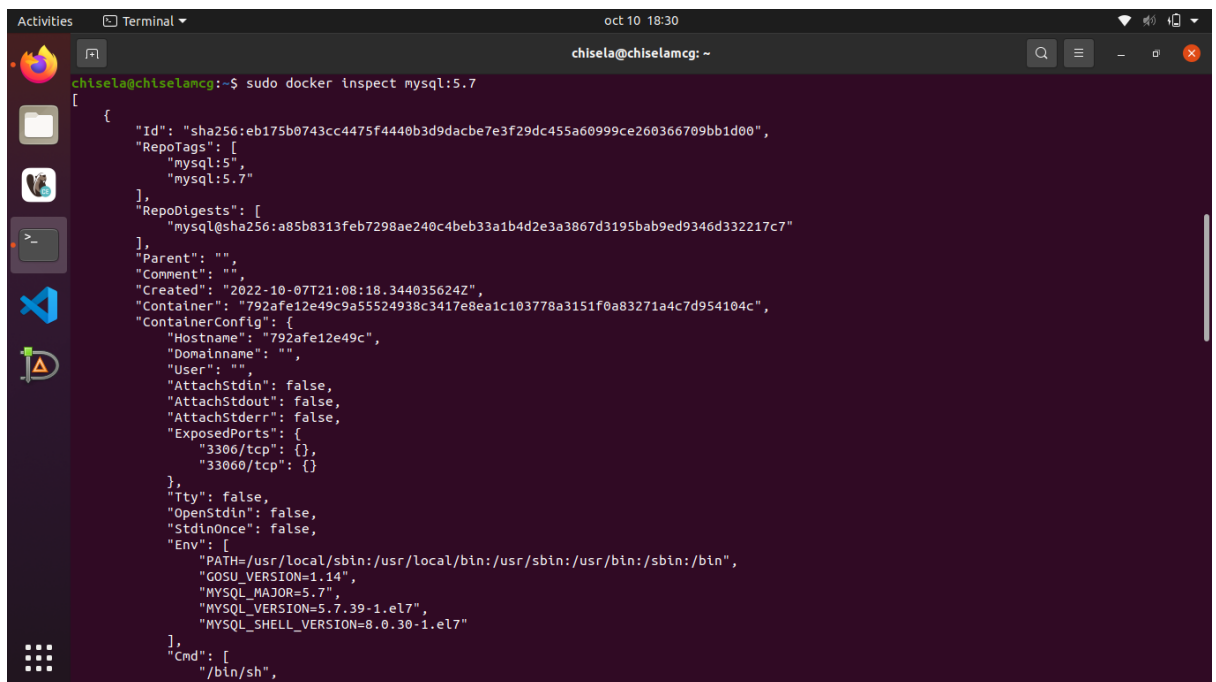
7. **Ejecuta el contenedor de apache (busca en Docker Hub) poniéndole nombre “web” ¿qué IP le ha asignado?. Compruébalo.**

Ejecutamos el contenedor poniéndole nombre con `docker run --name web httpd`

Para comprobar la IP usamos `docker inspect web` y buscamos el elemento `IPAddress`.

```
Ports: {
  "80/tcp": null
},
"SandboxKey": "/var/run/docker/netns/7c61621ee118",
"SecondaryIPAddresses": null,
"SecondaryIPv6Addresses": null,
"EndpointID": "9fc9a0bf418d09681f06fa464b003a88a8d594874a21284e731a294748fb0197",
"Gateway": "172.17.0.1",
"GlobalIPv6Address": "",
"GlobalIPv6PrefixLen": 0,
"IPAddress": "172.17.0.2",
"IPPrefixLen": 16,
"IPv6Gateway": "",
"MacAddress": "02:42:ac:11:00:02",
"Networks": {
  "bridge": {
    "IPAMConfig": null,
    "Links": null,
    "Aliases": null,
    "MacAddress": "02:42:ac:11:00:02",
    "IPv4Address": "172.17.0.2",
    "IPv6Address": ""
  }
}
```

8. **Arranca un contenedor del servicio Tomcat versión jdk 11. , llamándolo “Tomcat”, redirigiendolo al puerto 9999 (tomcat usa el puerto 8080). Comprueba que está funcionando.**  
docker run -d --name Tomcat -p 99:99 httpd:10.1.0-jdk11-temurin-jammy  
docker ps
9. **Para todos los contenedores que estén funcionando y bórralos.**  
docker stop \$(docker ps -a -q)  
docker rm \$(docker ps -a -q)
10. **Descarga la imagen mariadb (base de datos) y crea un volumen llamado DATA donde vayamos a guardar datos de mariadb. Comprueba que está creado.**  
docker pull mariadb  
docker volume create DATA  
docker ps -a
11. **Arranca un contenedor con el servicio mariadb funcionando... llamado “db1” con redirección de puerto 3336:3306 y haz el montaje en el volúmen DATA y destino /var/lib/mysql (carpeta del servidor)... decirle una variable de entorno -e MYSQL\_ROOT\_PASSWORD=root -e MYSQL\_DATABASE=test mariadb**  
docker run -d -it --name db1 --mount type=volume,src=DATA,dst=/var/lib/mysql -p 3306:3306 -e MYSQL\_ROOT\_PASSWORD=root mariadb
12. **Comprueba que el servidor está funcionando**  
docker ps
13. **Busca en el repositorio de docker hub y descarga la imagen mysql con una versión no actualizada y obtén información de la misma (explícala) . Muestra las imágenes descargadas hasta ese momento.**  
docker pull mysql:5.7



```
chisela@chiselamcg:~$ sudo docker inspect mysql:5.7
[
  {
    "Id": "sha256:eb175b0743cc4475f4440b3d9dacbe7e3f29dc455a60999ce260366709bb1d00",
    "RepoTags": [
      "mysql:5",
      "mysql:5.7"
    ],
    "RepoDigests": [
      "mysql@sha256:a85b8313feb7298ae240c4beb33a1b4d2e3a3867d3195bab9ed9346d332217c7"
    ],
    "Parent": "",
    "Comment": "",
    "Created": "2022-10-07T21:08:18.344035624Z",
    "Container": "792afe12e49c9a55524938c3417e8ea1c103778a3151f0a83271a4c7d954104c",
    "ContainerConfig": {
      "Hostname": "792afe12e49c",
      "Domainname": "",
      "User": "",
      "AttachStdin": false,
      "AttachStdout": false,
      "AttachStderr": false,
      "ExposedPorts": {
        "3306/tcp": {},
        "33060/tcp": {}
      },
      "Tty": false,
      "OpenStdin": false,
      "StdinOnce": false,
      "Env": [
        "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",
        "COSU_VERSION=1.14",
        "MYSQL_MAJOR=5.7",
        "MYSQL_VERSION=5.7.39-1.el7",
        "MYSQL_SHELL_VERSION=8.0.30-1.el7"
      ],
      "Cmd": [
        "/bin/sh",
        "-c"
      ]
    }
  ]
}
```

```
chisela@chiselamcg:~$ sudo docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
mysql	5	eb175b0743cc	2 days ago	433MB
mysql	5.7	eb175b0743cc	2 days ago	433MB
store/oracle/database-enterprise	12.2.0.1	12a359cd0528	5 years ago	3.44GB

**14. Borra dos imágenes a la vez en caso de que existan.**

```
docker rmi mysql:5.7 mysql:5
```