GLOSARIO DOCKER 2

- Los datos de un contenedor mueren con él.

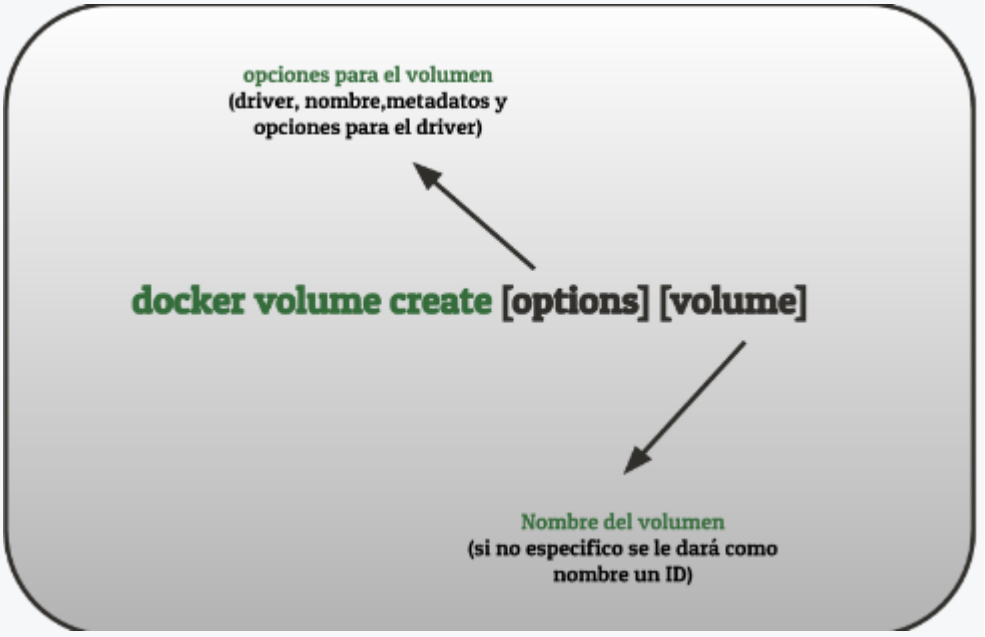
- Los datos de los contenedores no se mueven fácilmente ya que están fuertemente acoplados con el host en el que el contenedor está ejecutándose.

- Escribir en los contenedores es más lento que escribir en el host ya que tenemos una capa adicional.

Existen **VARIAS SOLUCIONES PARA PERSISTIR DATOS** de contenedores.

- **Los VOLÚMENES docker.**
- **Los BIND MOUNT.**

docker volume	● **Creación de los volúmenes.** ● **Eliminación de los volúmenes.** ● **Obtención de información de los volúmenes.**
docker volume create	Utilizada para la creación de volúmenes.



- **--driver o -d** para especificar el driver elegido para el volumen. Si no especificamos nada el driver utilizado es el ***local*** que es el que nos interesa desde el punto de vista de desarrollo porque desarrollamos en nuestra máquina. Al ser Linux en mi caso ese driver local es **overlay2** pero existen otras posibilidades como ***aufs, btrfs, zfs, devicemapper o vfs.*** Si estamos interesados en conocer al detalle cada uno de ellos aquí tenemos más información.
- **--label** para especificar los metadatos del volumen mediante parejas clave-valor.
- **--opt o -o** para especificar opciones relativas al driver elegido. Si son opciones relativas al sistema de ficheros puedo usar una sintaxis similar a las opciones de la orden mount.
- **--name** para especificar un nombre para el volumen. Es una alternativa a especificarlo al final que es la forma que está descrita en la imagen superior.

```
# Creación de un volumen llamado datos (driver local sin opciones)

> docker volume create data

# Creación de un volumen data especificando el driver local

> docker volume create -d local data

# Creación de un volumen llamando web añadiendo varios metadatos

> docker volume create --label servicio=http --label server=apache Web
```

docker volume rm	para eliminar un volumen en concreto (por nombre o por id).
docker volumen prune	ara eliminar los volúmenes que no están siendo usados por ningún contenedor.

```
# Borrar un volumen por nombre

> docker volume rm nombre_volumen

# Borrar un volumen por ID

> docker volume rm
a5175dc955cfcf7f118f72dd37291592a69915f82a49f62f83666ddc81f67441

# Borrar dos volúmenes de una sola vez

> docker volume rm nombre_volumen1 nombre_volumen2

# Forzar el borrado de un volumen -f o --force

> docker volume rm -f nombre_volumen

# Borrar todos los volúmenes que no tengan contenedores  asociados
```

```
# Borrar todos los volúmenes que no tengan contenedores  asociados

> docker volume prune

# Borrar todos los volúmenes que no tengan contenedores  asociados sin pedir
confirmación (-f o --force)

> docker volume prune -f

# Borrar todos los volúmenes sin usar que contengan cierto valor de etiqueta (--filter)

> docker volume prune --filter label=valor
```

NOTA: NO SE PUEDEN ELIMINAR VOLÚMENES EN USO POR CONTENEDORES, salvo que usemos el flag -f o --force y no es algo recomendado.

docker volume ls	que nos proporciona una lista de los volúmenes creados y algo de información adicional.

docker volume inspect	nos dará una información mucho más detallada de el volumen que hayamos elegido.

LISTA DE VOLÚMENES DEL SISTEMA

Si ejecutamos la siguiente orden:

```
# Listar los volúmenes creados en el sistema

> docker volume ls
```

Obtendremos una salida similar a la siguiente:

```
pekechis ~ docker volume ls
DRIVER              VOLUME NAME
local               a5175dc955cfcf7f118f72dd37291592a69915f82a49f62f83666ddc81f67441
local               jenkins_home
```

INFORMACIÓN DETALLADA DE UN VOLUMEN

Si queremos información más detallada de un volumen tenemos que ejecutar la siguiente orde

```
# Información detallada de un volumen por nombre

> docker volume inspect nombre_volumen

# Información detallada de un volumen por ID

> docker volume inspect
a5175dc955cfcf7f118f72dd37291592a69915f82a49f62f83666ddc81f67441
```

Obtendremos una salida similar a la siguiente:

```
[
    {
        "CreatedAt": "2020-11-05T00:37:51+01:00",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/jenkins_home/_data",
        "Name": "jenkins_home",
        "Options": null,
        "Scope": "local"
    }
]
```

Y la información que nos muestra es:

- La fecha de creación del volúmen.
- El tipo del driver.
- Etiquetas asociadas.
- El punto de montaje.
- El nombre del volumen.
- Las opciones asociadas al driver.
- Y el ámbito del volumen.

docker run --volume o -v.	Como puedo usar los volúmenes y los bind mounts en los contenedores. Este flag lo utilizaremos para **establecer bind mounts.**
docker run --mount	Este flag nos servirá para **establecer bind mounts** y para **usar volúmenes previamente definidos** (entre otras cosas).

Al usar tanto volúmenes como bind mount el contenido de lo que tenemos **sobreescribirá la carpeta destino en el sistema de ficheros del contenedor en caso de que exista.** Y si nuestra carpeta origen no existe y hacemos un bind mount esa carpeta se creará pero lo que tendremos en el contenedor es una carpeta vacía.

Si usamos imágenes de DockerHub, debemos l**eer la información que cada imagen nos proporciona en su página** ya que esa información suele indicar **cómo persistir los datos** de esa imagen, ya sea con volúmenes o bind mounts.

Ejemplos:

```
# BIND MOUNT (flag -v): La carpeta web del usuario será el directorio raíz del servidor apache. Se crea si no existe

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

# BIND MOUNT (flag --mount): La carpeta web del usuario será el directorio raíz del servidor apache. Se crea si no existe

> docker run --name apache -p 80:80 --mount type=bind,src=/home/usuario/web,dst=/usr/local/apache2/htdocs httpd

# VOLUME (flag --mount). Mapear el volumen previamente creado y que se llama Data en la carpeta raíz del servidor apache

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

# VOLUME (flag --mount). Igual que el anterior pero al no poner nombre de volumen se crea uno automáticamente (con un ID como nombre)

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