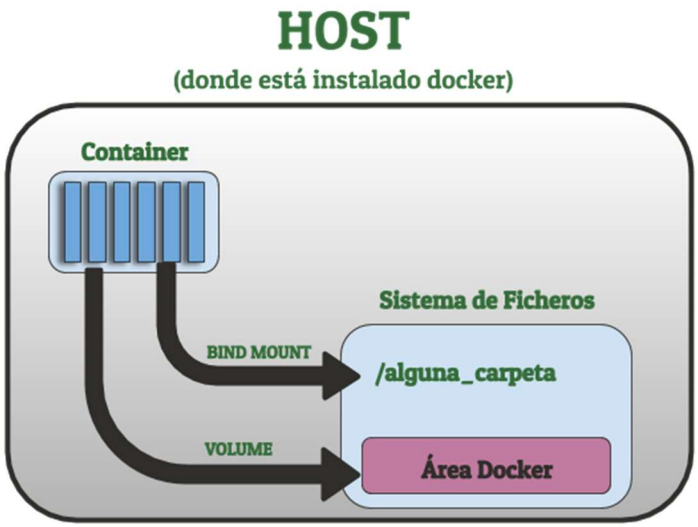
PERSISTENCIA DE DATOS EN DOCKER

- 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.



1. PERSISTENCIA USANDO VOLÚMENES

Los datos de los contenedores que nosotros decidamos se almacenen en una parte del sistema de ficheros que es gestionada por docker y a la que, debido a sus permisos, sólo docker tendrá acceso.

- */var/lib/docker/volumes* en Linux si lo hemos instalado desde paquetes estándar.
- */var/snap/docker/common/var-lib-docker/volumes* en Linux si hemos instalado docker mediante snap (no recomendado).
- *C:\ProgramData\docker\volumes* en Windows.
- */var/lib/docker/volumes* en Mac aunque se requiere que haya una conexión previa a la máquina virtual que se crea.

Se usan para:

- Compartir datos entre contenedores. Simplemente tendrán que usar el mismo volumen.
- Copias de seguridad ya sea para que sean usadas posteriormente por otros contenedores o para mover esos volúmenes a otros hosts.
- Almacenar los datos de mi contenedor no localmente si no en un proveedor cloud.
- En algunas situaciones donde usando Docker Desktop quiero más rendimiento. Esto se escapa al ámbito de este curso.

docker volume create [options] [volume_name] → **creación de volúmenes**
Si no se especifica el volume_name, se le da como nombre un ID.
[options]

- **--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.***
- **--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_name/volume_id] rm → eliminar un volumen en concreto
NO SE PUEDEN ELIMINAR VOLÚMENES EN USO POR CONTENEDORES, salvo que usemos el flag -f o --force y no es algo recomendado.

```
# 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
```

docker volume prune → eliminar los volúmenes que no están siendo usados por ningún contenedor.

```
# 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
```

docker volume ls → lista los volúmenes creados y algo de información adicional (driver usado, nombre o ID de cada volumen)

docker volume inspect [volume_name/volume_ID] → información mucho más detallada del volumen elegido (fecha de creación, tipo de driver, etiquetas asociadas, punto de montaje, opciones asociadas al driver y ámbito).

2. PERSISTENCIA USANDO BIND MOUNTS

"Mapeamos" una parte del sistema de ficheros, de la que normalmente tenemos el control, con una parte del sistema de ficheros del contenedor.
Se usa para compartir ficheros entre host y containers, para que otras aplicaciones, que no sean docker, tengan acceso a esos ficheros.
Es el sistema empleado en la fase de desarrollo porque:

- Las aplicaciones que podrán acceder a esos ficheros serán los IDEs o editores de código.
- Estaremos modificando con aplicaciones locales código que a la vez se encuentra en nuestro equipo y en el contenedor.
- Y desde mi propio equipo estaré probando ese código en el entorno elegido, o en varios entornos a la vez sin necesidad de tener que instalar absolutamente nada en mi sistema.

3. USAR VOLÚMENES Y BIND MOUNTS EN LOS CONTENEDORES

- Al usar tanto volúmenes como bind mount el contenido de lo que tenemos sobrescribirá 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 leer 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.

Para usar los volúmenes o los bind mounts, usaremos dos flags de la orden docker run:

- **--volume o -v**. Este flag lo utilizaremos para establecer bind mounts.
- **--mount**. Este flag nos servirá para establecer bind mounts y para usar volúmenes previamente definidos.

```
# 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
```