T4.2 – Gestión de Volúmenes

Las operaciones básicas son:

- **Creación** de los volúmenes: **docker volume create**
- **Eliminar de los volúmenes: docker volume rm /docker volume prune**
- **Obtener información de los volúmenes: docker volume ls /**

docker volume inspect nom-volume

Prune: elimina los volúmenes no enlazados a ningún contenedor.

El comando para crear un volumen puede incluir las siguientes opciones:

- **--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.
- **--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 debian 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.

Vamos a ilustrar este funcionamiento con varios ejemplos:

```
# 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
```

Ejemplos de borrado:

```
# 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
> 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
```

Cuando se crea un contenedor se puede establecer los Bind Mounts indicando en docker run los siguientes flag:

- --volume o -v:
- --mount: para volúmenes ya creados

Es importante que tengamos en cuenta dos cosas importantes a la hora de realizar estas operaciones:

- 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. Con esto hay que tener especial cuidado, sobre todo cuando estamos trabajando con carpetas que pueden contener datos y configuraciones varias.
- 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, y cuáles son las **carpetas importantes** en caso de ser imágenes que contengan ciertos servicios (web, base de datos etc...)

Como acostumbramos en este curso vamos a ilustrar todo esto mediante una serie de ejemplos a los que añadiremos varios opciones

```
# 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
```

ejemplos:

Capturas a entregar: (incluir capturas de todos los comandos necesario**s)**

- Comprueba los caso de volume y bind mount anteriormente indicados
- Crea un volumen llamado **volumen_web1** y volumen_http
- Comprueba la información del volumen creado.
- Elimina volumen_http y compruebalo.
- Crea un contenedor sobre la imagen **php**:7.4-apache a la vez que mapeas (type mount) un volumen que has creado anteriormente volumen_web con su directorio de trabajo y accesible por el puerto 8787 (consulta la documentación sobre dockerhub). Y lista los volúmenes y los contenedores creados.
- Inspecciona la diferencia entre los volúmenes y big mount creados creados. Comprueba el funcionamiento
- Elimina el contenedor.
- Crea un contenedor sobre la imagen **php**:7.4-apache a la vez que mapeas (type bind) un directorio de tu equipo con su directorio de trabajo y accesible por el puerto 8585 (consulta la documentación sobre dockerhub). Y lista los volúmenes y los contenedores creados.
- Comprueba el funcionamiento.