Persistencia de datos

Volúmenes:

Si elegimos conseguir la persistencia usando volúmenes estamos haciendo que 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.

Esa "ZONA RESERVADA" de docker cambia de un sistema operativo a otro y también puede cambiar dependiendo de la forma de instalación, pero de manera general podemos decir que es:

- */var/lib/docker/volumes* en las distribuciones de 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 lo recomiendo).
- *C:\ProgramData\docker\volumes* en las instalaciones de Windows.
- */var/lib/docker/volumes* también en Mac aunque se requiere que haya una conexión previa a la máquina virtual que se crea.

Este tipo de volúmenes se suele usar en los siguiente casos:

- Para compartir datos entre contenedores. Simplemente tendrán que usar el mismo volumen.
- Para copias de seguridad ya sea para que sean usadas posteriormente por otros contenedores o para mover esos volúmenes a otros hosts.
- Cuando quiero 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.

Bind Mounts:

Si elegimos conseguir la persistencia de los datos de los contenedores usando bind mount lo que estamos haciendo es *"mapear"* una parte de mi sistema de ficheros, de la que yo normalmente tengo el control, con una parte del sistema de ficheros del contenedor.

Este mapeado de partes de mi sistema de ficheros con el sistema de ficheros del contenedor me va a permitir:

- Compartir ficheros entre el host y los containers.
- Que otras aplicaciones que no sean docker tengan acceso a esos ficheros, ya sean código, ficheros etc...

Puede parecer que el hecho de que otras aplicaciones accedan a esos datos es algo negativo pero precisamente los bind mounts es el mecanismo que vamos a preferir para la fase de DESARROLLO ya que:

- 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.

Comandos:

$ docker volume create [options] [volume] creación de volúmenes

- **--driver o -d** para especificar el driver elegido para el volumen.
- **--label** para especificar los metadatos del volumen mediante parejas clave-valor.
- **--opt o -o** para especificar opciones relativas al driver elegido.
- **-e NOMBRE_VARIABLE=VALOR** para definir variables de entorno.
- **--name** para especificar un nombre para el volumen.

Ejemplos:

```
docker volume create -d local data
```

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

$ docker volume rm borrar un volumen

$ docker volumen prune para eliminar los volúmenes que no están siendo usados por ningún contenedor.

$ docker volume ls listar volúmenes creados

$ docker volume inspect información detallada de un volumen

Flags para 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