Almacenamiento en docker

José Domingo Muñoz

IES Gonzalo Nazareno

Febrero 2021

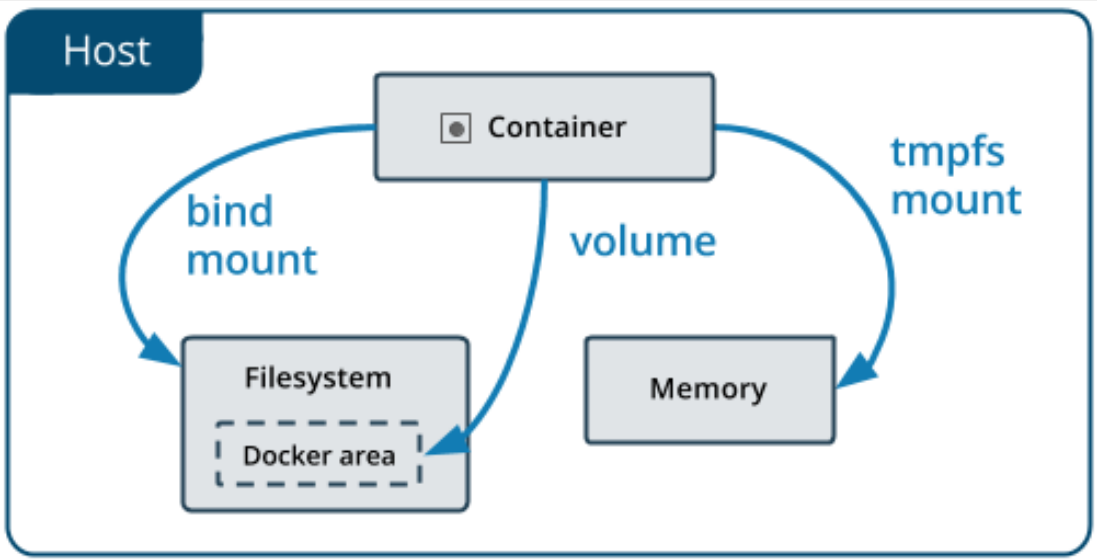
Los contenedores son efímeros

Los contenedores son efímeros, es decir, los ficheros, datos y configuraciones que creamos en los contenedores sobreviven a las paradas de los mismos pero, sin embargo, son destruidos si el contenedor es destruido.

Los contenedores son efímeros

```
# Creo un contenedor con un servidor web
$ docker run -d --name my-apache-app -p 8080:80 httpd:2.4
# Creo un fichero index.html en el document root
$ docker exec my-apache-app bash -c 'echo "<h1>Hola</h1>" > htdocs/index.html'
# Accedo al contenedor y veo el contenido del fichero
$ curl http://localhost:8080
<h1>Hola</h1>
# Elimino el contenedor
$ docker rm -f my-apache-app
my-apache-app
# Creo uno nuevo
$ docker run -d --name my-apache-app -p 8080:80 httpd:2.4
# Se ha perdido el contendio del fichero index.html
$ curl http://localhost:8080
<html><body><h1>It works!</h1></body></html>
```

Los datos en los contenedores



Volúmenes docker y bind mount

- **Volúmenes docker**: 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. En linux se guardan en `/var/lib/docker/volumes`.
- **Bind mounts**: Si elegimos conseguir la persistencia de los datos de los contenedores usando bind mount lo que estamos haciendo es “mapear” (montar) una parte de mi sistema de ficheros, de la que yo normalmente tengo el control, con una parte del sistema de ficheros del contenedor.

Gestionando volúmenes

- **docker volumen create**: Crea un volumen con el nombre indicado.
- **docker volume rm**: Elimina el volumen indicado.
- **docker volumen prune**: Para eliminar los volúmenes que no están siendo usados por ningún contenedor.
- **docker volume ls**: 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.

Asociando almacenamiento a los contenedores

- 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.
- 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, y cuáles son las carpetas importantes en caso de ser imágenes que contengan ciertos servicios (web, base de datos etc…)

Ejemplo usando volúmenes docker

```
# Creo un volumen
$ docker volume create miweb
miweb
# Monto el volumen en el directorio /usr/local/apache2/htdocs usando --mount
$ docker run -d --name my-apache-app --mount type=volume,src=miweb,dst=/usr/local/apache2/htdocs -p 8080:80 httpd:2.4
# Creo un fichero index.html en el document root
$ docker exec my-apache-app bash -c 'echo "<h1>Hola</h1>" > htdocs/index.html'
# Accedo al contenedor y veo el contenido del fichero
$ curl http://localhost:8080
<h1>Hola</h1>
# Borro el contenedor
$ docker rm -f my-apache-app
# Monto el volumen en el directorio /usr/local/apache2/htdocs usando -v
$ docker run -d --name my-apache-app -v miweb:/usr/local/apache2/htdocs -p 8080:80 httpd:2.4
# Accedo al contenedor y veo el contenido del fichero. no se ha perdido el contenido.
$ curl http://localhost:8080
<h1>Hola</h1>
```

Ejemplo usando volúmenes docker

```
$ docker run -d --name my-apache-app --mount type=volume,dst=/usr/local/apache2/htdocs -p 8080:80 httpd:2.4
$ docker run -d --name my-apache-app -v wwwroot:/usr/local/apache2/htdocs -p 8080:80 httpd:2.4
```

Estos dos ejemplos crean nuevos volúmenes docker.

Ejemplo usando bind mount

```
# Creamos un directorio con un fichero index.html
$ mkdir web
$ cd web
/web$ echo "<h1>Hola</h1>" > index.html
# Monto el volumen en el directorio /usr/local/apache2/htdocs usando -v
$ docker run -d --name my-apache-app -v /home/usuario/web:/usr/local/apache2/htdocs -p 8080:80 httpd:2.4
# Accedo al contenedor y veo el contenido del fichero
$ curl http://localhost:8080
<h1>Hola</h1>
# Borro el contenedor
$ docker rm -f my-apache-app
# Monto el volumen en el directorio /usr/local/apache2/htdocs usando --mount
$ docker run -d --name my-apache-app --mount type=bind,src=/home/usuario/web,dst=/usr/local/apache2/htdocs -p 8080:80 htt
1751b04b0548217d7faa628fd69c10e84c695b0e5cc33b482df2c04a6af83292
#Accedo al contenedor y veo el contenido del fichero. no se ha perdido el contenido.
$ curl http://localhost:8080
<h1>Hola</h1>
```

Ejemplo usando bind mount

Podemos modificar el contenido del fichero aunque este montado en el contenedor:

```
$ echo "<h1>Adios</h1>" > web/index.html
$ curl http://localhost:8080
<h1>Adios</h1>
```

MARIADB CON ALMACENAMIENTO PERSISTENTE

```
# creamos un contenedor desde la imagen mariadb usando un bind mount
$ docker run --name some-mariadb -v /home/usuario/datadir:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=my-secret-pw -d mariadb
# Accedemos  y creamos una base ded datos
$ docker exec -it some-mariadb bash -c 'mysql -uroot -p$MYSQL_ROOT_PASSWORD'
...
MariaDB [(none)]> create database prueba;
# Borro el contenedor
$ docker rm -f some-mariadb
# vuelvo a crear un nuevo contenedor con el mismo almacenamiento
$ docker run --name some-mariadb -v /home/vagrant/datadir:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=my-secret-pw -d mariadb
# Vuelvo acceder y compruebo que sigue existiendo la base de datos
$ docker exec -it some-mariadb bash -c 'mysql -uroot -p$MYSQL_ROOT_PASSWORD'
...
MariaDB [(none)]> show databases;
...
| prueba             |
...
```

¿Qué información tenemos que guardar?

- Los datos de la aplicación
- Los logs del servicio
- La configuración del servicio: En este caso podemos añadirla a la imagen, pero será necesaria la creación de una nueva imagen si cambiamos la configuración. Si la guardamos en un volumen hay que tener en cuanta que ese fichero lo tenemos que tener en el entorno de producción (puede ser bueno, porque las configuraciones de los distintos entornos puede variar).

Ejercicios

1. Vamos a trabajar con volúmenes docker:
   - Crea un volumen docker que se llame `miweb`.
   - Crea un contenedor desde la imagen `php:7.4-apache` donde montes en el directorio `/var/www/html` (que sabemos que es el docuemntroot del servidor que nos ofrece esa imagen) el volumen docker que has creado.
   - Utiliza el comando `docker cp` para copiar un fichero `info.php` en el directorio `/var/www/html`.
   - Accede al contenedor desde el navegador para ver la información ofrecida por el fichero `info.php`.
   - Borra el contenedor
   - Crea un nuevo contenedor y monta el mismo volumen como en el ejercicio anterior.
   - Accede al contenedor desde el navegador para ver la información ofrecida por el fichero `info.php`. ¿Seguía existiendo ese fichero?

Ejercicios

2. Vamos a trabajar con bind mount:
   - Crea un directorio en tu host y dentro crea un fichero `index.html`.
   - Crea un contenedor desde la imagen `php:7.4-apache` donde montes en el directorio `/var/www/html` el directorio que has creado por medio de `bind mount`.
   - Accede al contenedor desde el navegador para ver la información ofrecida por el fichero `index.html`.
   - Modifica el contenido del fichero `index.html` en tu host y comprueba que al refrescar la página ofrecida por el contenedor, el contenido ha cambiado.
   - Borra el contenedor
   - Crea un nuevo contenedor y monta el mismo directorio como en el ejercicio anterior.
   - Accede al contenedor desde el navegador para ver la información ofrecida por el fichero `index.html`. ¿Se sigue viendo el mismo contenido?

Ejercicios

3. Contenedores con almacenamiento persistente
   - Crea un contenedor desde la imagen `nextcloud` (usando sqlite) configurando el almacenamiento como nos muestra la documentación de la imagen en Docker Hub (pero utilizando bind mount). Sube algún fichero.
   - Elimina el contenedor.
   - Crea un contenedor nuevo con la misma configuración de volúmenes. Comprueba que la información que teníamos (ficheros, usuaurio, …), sigue existiendo.
   - Comprueba el contenido de directorio que se ha creado en el host.