Integrar contenedores y volúmenes

Ya tenemos suficientes conocimientos para montar un servicio completo usando Docker. Para ello vamos a necesitar levantar contenedores que trabajen juntos y accedan a volúmenes para guardar los datos.

En este caso vamos a montar un *blog* con *WordPress*.

> **Atención**
>
> Lo importante de este capítulo no es saber montar un Wordpress. Eso es solo un ejemplo y no necesitas saber cómo se monta un Wordpress. Lo importante es saber crear los contenedores, enlazarlos entre ellos, usar un volumen. No te preocupes si no entiendes los detalles de los parámetros que se pasan a Wordpress. No es relevante.

Wordpress es un servicio de blogs y páginas web que necesita para funcionar:

- PHP
- Una base de datos MySQL o MariaDB.
- Soporte HTTPS

En una instalación tradicional bastaría un servidor donde instalaríamos PHP, una base de datos, un servidor web Apache o Nginx con soporte https y el propio servidor worpress. La base de datos se almacenaría en el sistema de archivos del propio servidor. Los ficheros de configuración y las páginas que se fueran creando se almacenarían también en el propio servidor.

Pero en el despliegue con contenedores usaremos:

- Un contenedor, que llamaremos **wordpress**, conendrá el servidor Wordpress.
- Este contenedor almacenará los ficheros de configuración y páginas que creemos en una carpeta de la máquina host
- Otro contenedor, que llamaremos **wordpress-db**, contendrá el sistema gestor de base de datos MariaDB.
- Dicho gestor MariaDB guardará la base de datos en un volumen docker, no en el propio contenedor. Usaremos el volumen creado en el capítulo anterior y que llamamos **vol-wordpress-db**.
- Enlazaremos ambos contenedores **wordpress** y **wordpres-db**

Importante

Como vemos al hacer un despliegue en docker independizamos los servidores de los datos. De esta forma podremos actualizar el sistema a otra versión, o hacer pruebas de plugins o modificaciones simplemente creando o sustituyendo el contenedor del servidor por uno nuevo que acceda a los mismos datos.

Crear el volumen

Empezaremos creando un volumen que se llame **vol-wordpress-db**

```
    $ docker volume create vol-wordpress-db
```

Crear un contenedor con *MariaDB*.

WordPress soporta los motores relaciones *MySQL* y *MariaDB*. Usaremos este último. Vamos a crear nuestra base de datos usando volumen el volumen que acabamos de crear para que guarde los datos.

```
docker run -d --name wordpress-db \
    --mount source=vol-wordpress-db,target=/var/lib/mysql \
    -e MYSQL_ROOT_PASSWORD=secret \
    -e MYSQL_DATABASE=wordpress \
    -e MYSQL_USER=manager \
    -e MYSQL_PASSWORD=secret \
    mariadb:10.3.9
```

La imagen se descargará, si no lo estaba ya, y se iniciará nuestro contenedor de *MariaDB*:

```
docker run -d --name wordpress-db \
    --mount source=vol-wordpress-db,target=/var/lib/mysql \
    -e MYSQL_ROOT_PASSWORD=secret \
    -e MYSQL_DATABASE=wordpress \
    -e MYSQL_USER=manager \
    -e MYSQL_PASSWORD=secret \
    mariadb:10.3.9
Unable to find image 'mariadb:10.3.9' locally
10.3.9: Pulling from library/mariadb
124c757242f8: Pull complete
9d866f8bde2a: Pull complete
fa3f2f277e67: Pull complete
398d32b153e8: Pull complete
afde35469481: Pull complete
31f2ae82b3e3: Pull complete
3eeaf7e45ea6: Pull complete
716982328e17: Pull complete
34ce605c9036: Pull complete
```

```
4502ed9073c0: Pull complete
2afafbdf5a96: Pull complete
43d52b11dd31: Pull complete
30c7b70556f3: Pull complete
8b1b39f2f89a: Pull complete
41480b9319d7: Pull complete
Digest:
sha256:b7894bd08e5752acdd41fea654cb89467c99e67b8293975bb5d787b27e66ce1a
Status: Downloaded newer image for mariadb:10.3.9
30634831d17108aa553a5774e27f398760bdbdf32debc3179843e73aa5957956

$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
STATUS              PORTS               NAMES
30634831d171        mariadb:10.3.9      "docker-entrypoint.s…"   20 seconds
ago      Up 16 seconds       3306/tcp            wordpress-db
```

Analicemos las distintas partes del comando.

- Lo primero que habremos notado es que el contenedor ya no se queda en primer plano. El parámetro `-d` indica que debe ejecutarse como un proceso en segundo plano. Así no podremos pararlo por accidente con `Control+C`.
- Lo segundo es que vemos que el contenedor usa un puerto, el `3306/tcp`, pero no está enlazado a ningún puerto de la máquina anfitrión. Nosotros no hemos usado el parámetro `-p PUERTOANFITRION:PUERTOCONTENEDOR` como en otros casos ni hemos dicho en la configuración nada respecto al puerto a exponer. Esa información ya está en la imagen que usamos. Con esta configuración no tenemos forma de acceder a la base de datos directamente. Nuestra intención es que solo el contenedor de *WordPress* (que crearemos a continuación) pueda acceder. En este caso el puerto 3306/tcp, por el que se accede al servidor de MariaDB se está *exponiendo*, es decir, haciéndolo disponible **solo** a otros contenedores. Ni siquiera se podrá acceder desde la máquina anfitrión.
- Luego una serie de parámetros `-e` que nos permite configurar nuestra base de datos pasándole variables de entorno al contenedor. Como dijimos anteriormente no nos preocupan estos parámetros, aunque son bastante autoexplicativos.

  > **Info**
  >
  > Los contenedores se configuran a través de variables de entorno, que podemos configurar con el parámetro `-e` que vemos en la orden anterior. Gracias a ellos hemos creado una base de datos, un usuario y configurado las contraseñas.
  >
  > Se recomienda buscar en el registro de *Docker* la imagen oficial de MariaDB para entender el uso de los parámetros.

- Por último, el parámetro `--mount` nos permite enlazar el volumen **vol-wordpress-db** que creamos en el paso anterior con el directorio `/var/lib/mysql` del contenedor. Ese directorio es donde se guardan los datos de *MariaDB*. Eso significa que si borramos el

contenedor, o actualizamos el contenedor a una nueva versión, no perderemos los datos porque ya no se encuentran en él, sino en el volumen. Solo lo perderíamos si borramos explícitamente el volumen.

> **Warning**
>
> Cada contendor que usemos tendrá uno o varios directorios donde se deben guardar los datos no volátiles. Nos corresponde a nosotros conocer la herramienta y saber de qué directorios se tratan. Usualmente están en la documentación del contenedor, pero no siempre.

> **Info**
>
> El parámetro `--mount` se empezó a utilizar desde la versión `17.06` para contenedores independientes (los que no pertenecen a un enjambre o *swarm*). Los que conozcan *Docker* de versiones más antiguas estarán más acostumbrados a usar el parámetro `--volume` que hace algo similar. Sin embargo la documentación aconseja usar ya `--mount`, sobre todo para nuevos usuarios.
>
> Nosotros somos muy obedientes así que en este taller usaremos `--mount`.

- Finalmente indicamos la imagen que vamos a usar: `mariadb:10.3.9`.

Creando nuestro blog

Vamos ahora a crear nuestro contenedor de *WordPress*, y a conectarlo con nuestra base de datos. Además, queremos poder editar los ficheros de las plantillas, por si tenemos que modificar algo, así que necesitaremos montar el directorio del contenedor donde está instalado *WordPress* con nuestra cuenta de usuario en la máquina anfitrión.

Vamos a crear el espacio de trabajo en la máquina anfitrión donde estamos usando docker:

```
mkdir -p ~/Sites/wordpress/target && cd ~/Sites/wordpress
```

Y dentro de este directorio arrancamos el contenedor:

```
docker run -d --name wordpress \
    --link wordpress-db:mysql \
    --mount type=bind,source="$(pwd)"/target,target=/var/www/html \
    -e WORDPRESS_DB_USER=manager \
    -e WORDPRESS_DB_PASSWORD=secret \
    -p 8080:80 \
    wordpress:4.9.8
```

Revisa bien los distintos parámetros, la mayoría ya los hemos visto antes:

- `-d` para lanzar en 2º plano
- --mount para enlazar un directorio en el contenedor con un directorio en la máquina anfitrión
- -e para definir parámetros en el contenedor
- -p 8080:80 para enlazar el puerto 8080 en el anfitrión con el 80 en el contenedor.

Pero encontramos un parámetro nuevo:

- --link wordpress-db:mysql: indica que este contenedor se vincula con otro llamado "wordpress-db", y permite que este contenedor lo referencie utilizando el alias "mysql".

> **Info**
>
> Puedes encontrar toda la información referente a esta imagen en DockerHub buscando la imagen wordpress.
>
> Pero insisto, no se trata de saber montar un Wordpress, sino de aprender cómo usar contenedores que se enlazan, usan un volumen, etc.

Cuando termine la ejecución, si accedemos a la dirección http://localhost:8080/, ahora sí podremos acabar el proceso de instalación de nuestro WordPress. Recuerda que si accedes desde un navegador en un equipo distinto al que estás lanzando los contenedores deberás usar `http://IPSERVER:8080` sustituyendo IPSERVER por la IP de tu máquina anfitrión.

Si listamos el directorio target comprobaremos que tenemos todos los archivos de instalación accesibles desde el directorio anfitrión.

Ejercicios

Ejercicios:

1. Para los contenedores, tanto **wordpress** como **wordpress-db**.
2. Borra ambos.
3. Comprueba que sigue existiendo el volumen **vol-wordpress-db** que contiene la base de datos.
4. Comprueba que sigue existiendo el directorio ~/Sites/worpress/target en la máquina anfitrión con los ficheros de configuración.
5. Vuelve a crear el contenedor **wordpress-db** pero actualizando la versión de MariaDb. Usa la imagen mariadb:11.
6. Vuelve a crear el contenedor **wordpress** pero actualizando la versión de Wordpress. Usa la imagen wordpress:5.0.0-apache.
7. Vuelve a acceder a tu wordpress y comprueba lo que ha costado actualizar la versión y cómo se han mantenido los datos de la base de datos y de los ficheros guardados en el anfitrión.
8. Vuelve a borrarlos y borra también el volumen y el contenido de ~/Sites/worpress/target
9. Vuelve a crear el volumen y los contenedores y comprueba que ahora sí hay que volver a instalar *WordPress*. Ahora puedes hacerlo con la versión latest de ambos servidores.

Nota

Si intentas actualizar a las versiones latest de ambos servidores sin eliminar el volumen y los ficheros en ~/Sites/worpress/target no te funcionará. Probablemente en algún cambio de versión, bien de MariaDB o de Wordpress, cambió la forma de almacenar la información y se perdió la compatibilidad hacia atrás, con lo que no son capaces de leer los datos existentes

Para saber más

Para los efectos de este curso es suficiente con lo visto hasta aquí. Pero si quieres saber más y estudiaste el documento de "Para saber más" de la sección anterior donde profundizaba en volúmenes y redes docker, puedes hacer la práctica siguiente.

En esta práctica hace lo mismo que hemos visto anteriormente pero no usa el parámetro --link para enlazar los contenedores. A partir de Docker 1.13, el uso de --link ha sido desaconsejado en favor de la creación de redes para conectar contenedores.

Caso práctico 01 - Wordpress + MySQL