Manual de uso de Docker

Alejandro Charte Luque, Alejandro Munuera Fuentes, Francisco Porcuna Martínez

Deditec

Índice

1 Introducción

1.1 ¿Qué es Docker?

Docker es una tecnología de virtualización basada en contenedores. Estos contenedores son más ligeros que máquinas virtuales tradicionales, ya que no ejecutan un sistema operativo completo, solamente las herramientas necesarias para realizar la tarea que nosotros deseemos.

Por esta razón, los contenedores de Docker consumen menos recursos que las máquinas virtuales tradicionales, lo que permite tener más contenedores ejecutándose simultáneamente en un mismo host.

1.2 ¿Qué diferencia hay entre imágenes y contenedores?

Una imagen es una "plantilla" para crear un contenedor. Una imagen no se ejecuta directamente, se utiliza para generar un contenedor a partir de ella con la configuración y paquetes que vienen predefinidos en dicha imagen.

Una vez creado el contenedor, éste puede ser iniciado, parado, eliminado, etc. usando los comandos que veremos más adelante.

Posteriormente veremos también como a partir de un contenedor que hayamos configurado nosotros, podemos crear una imagen la cual pueden utilizar otros usuarios o nosotros mismos para crear nuevos contenedores con la misma configuración.

2 Comandos para imágenes

2.1 `docker image ls`

Nos permite ver una lista de las imágenes que tenemos descargadas.

2.2 `docker pull`

Sirve para descargar una imagen que aun no tengamos localmente.

> Nota: No le daremos gran uso a este comando ya que al crear un contenedor, la imagen se descarga automáticamente si no la tenemos.

2.3 `docker image rm`

Para eliminar una imagen que no queremos tener localmente o que ya no estemos utilizando.

3 Comandos para contenedores

3.1 `docker run`

Este comando nos permite generar un nuevo contenedor a partir de una imagen. Será uno de los comandos que más utilicemos ya que con él crearemos el contenedor y especificaremos distintas configuraciones, como puertos que queremos exponer, nombre y hostname o variables de entorno que queramos asignar.

3.2 Estructura del comando

```
docker run [opciones] <nombre_de_la_imagen>
```

3.3 Opciones

- Los parámetros `--name` y `--hostname` nos permiten especificar el nombre del contenedor (por defecto usa uno generado aleatoriamente por Docker) y un hostname. El `--name` es importante a la hora de buscar el contenedor cuando ya tenemos una gran cantidad de ellos.
- El parámentro `-i` y `-t` activan el modo interactivo y asignan una terminal (en la que estemos actualmente) al contenedor. Estos dos comandos suelen utilizarse en conjunto para crear un contenedor e inmediatamente conectarnos a él y empezar a ejecutar comandos dentro del mismo.
- El parámetro `-d` permite iniciar el contenedor y dejarlo en segundo plano, de forma que no veremos su output en el terminal.
- Con el parámetro `-e` podemos asignar variables de entorno del contenedor. De esta forma si creamos un contenedor usando la opción `-eMENSAJE=Bienvenido`, si dentro del contenedor ejecutamos `echo $MENSAJE` veremos que nos devuelve `Bienvenido`. Ésta opción puede usarse múltiples veces para asignar más de una variable de entorno.
- Un parámetro utilizado a menudo es `--rm`. Con él podemos hacer que el contenedor se elimine automáticamente cuando nos salgamos de él (útil para realizar alguna prueba sin que quede rastro).

3.4 Ejemplos

Si queremos ejecutar un contenedor simple de Ubuntu podemos usar el siguiente comando:

```
docker run -it --name miubuntu --hostname miubuntu ubuntu
```

3.5 docker start/stop

Con estos comandos podemos iniciar y detener un contenedor.

3.6 docker rm

Permite eliminar un contenedor, el cual debe haberse detenido anteriormente.

3.7 docker exec

Nos permite ejecutar un comando dentro del contenedor desde el host. También sirve para conectarnos al contenedor y poder interactuar con él usando el siguiente comando: `docker exec -it <nombre_contenedor> bash`.

3.8 docker ps

Nos muestra los contenedores que están ejecutándose actualmente. Si añadimos la opción `-a` nos muestra también los contenedores que no están iniciados.

3.9 docker logs

Nos muestra los logs del contenedor que le indiquemos. Es muy útil a la hora de diagnosticar problemas con el contenedor sin necesidad de conectarnos a él.

Podemos añadir la opción `-f` y nos irá mostrando en tiempo real los logs que se van generando, pero nos bloqueará el terminal.

4 Otros comandos de Docker

4.1 docker commit

Una vez tenemos un contenedor configurado a nuestro gusto, es probable que queramos crear más instancias o compartirlo con otras personas. Éste comando genera una imagen a partir del contenedor que le digamos, siguiendo el comando la siguiente estructura: `docker commit -m "Mensaje de cambios" <nombre_contenedor> <nombre_imagen>`

4.2 docker login

Nos permite iniciar sesión en Docker Hub para poder subir imágenes que generemos con `docker commit` o para descargar imágenes que no tengamos públicas en nuestro repositorio.

4.3 docker push

Permite subir una imagen que hayamos creado a Docker Hub una vez hayamos iniciado sesión con `docker login`. Lo haremos así: `docker push <nombre_imagen>`

5 Docker Compose

La herramienta **docker-compose** nos permite crear múltiples contenedores de Docker desde un único archivo.

5.1 docker-compose up

Nos generará los contenedores y redes que haya especificados en el archivo `docker-compose.yml`.

5.2 docker-compose down

Nos generará eliminará contenedores y redes que haya especificados en el archivo `docker-compose.yml`.

5.3 Otros comandos

Con `docker-compose` tambiés es posible realizar operaciones como las vistas anteriormente (start, logs, etc). Para ello hay que ejecutar `docker-compose <operacion> <nombre_contenedor>`.

> Nota: El nombre del contenedor debe ser el que aparece en el archivo `docker-compose.yml`, no el nombre de contenedor que aparece en `docker ps`, ya que le añade el nombre de la carpeta en la que se encuentra nuestro archivo.