Introducción a Docker

José Domingo Muñoz

IES Gonzalo Nazareno

Febrero 2021

Docker

Docker es una tecnología de virtualización “ligera” cuyo elemento básico es la utilización de contenedores en vez de máquinas virtuales y cuyo objetivo principal es el despliegue de aplicaciones encapsuladas en dichos contenedores.

Componentes:

- **Docker Engine**
- **Docker Client**
- **Docker Registry**: (Docker Hub)

Instalación de docker

```
apt install docker.io

usermod -aG docker usuario

$ docker --version
Docker version 18.09.1, build 4c52b90
```

El "Hola Mundo" de docker

```
$ docker run hello-world
```

- Al ser la primera vez que ejecuto un contenedor basado en esa imagen, la imagen `hello-word` se descarga desde el repositorio que se encuentra en el registro que vayamos a utilizar, en nuestro caso DockerHub.
- Muestra el mensaje de bienvenida que es la consecuencia de crear y arrancar un contenedor basado en esa imagen.
- **Un contenedor ejecuta un proceso y cuando termina la ejecución, el contenedor se para.**

El "Hola Mundo" de docker

- **docker run**: Crea un contenedor a partir de una imagen.
- **docker images**: Muestra imágenes descargadas.
- **docker ps**: Muestra contenedores en ejecución.
- **docker ps -a**: Muestra todos los contenedores.
- **docker rm**: Elimina un contenedor parado.
- **docker rm -f**: Eliminar un contenedor en ejecución.

Otro “Hola Mundo”

```
$ docker run ubuntu /bin/echo 'Hello world'
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
8387d9ff0016: Pull complete
...
Status: Downloaded newer image for ubuntu:latest
Hello world

$ docker ps -a
CONTAINER ID    IMAGE     COMMAND                  CREATED          STATUS       NAMES
3bbf39d0ec26    ubuntu    "/bin/echo 'Hello wo..."  31 seconds ago   Exited...    wizardly_edison

$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              latest              f63181f19b2f        7 days ago          72.9MB
hello-world         latest              bf756fb1ae65        13 months ago       13.3kB
```

Ejecutando un contenedor interactivo

```
$  docker run -it --name contenedor1 ubuntu /bin/bash
root@2bfa404bace0:/#
```

Todas las imágenes tienen definidas un proceso que se ejecuta, en concreto la imagen `ubuntu` tiene definida por defecto el proceso `bash`, por lo que podríamos haber ejecutado:

```
$  docker run -it --name contenedor1 ubuntu
root@2bfa404bace0:/#
```

El comando docker

- **docker start**: Arranca un contenedor parado.
- **docker stop**: Para un contenedor.
- **docker restart**
- **docker attach**: En un contenedor interactivo se vuelve a conectar.
- **docker exec**: Ejecuta una instrucción en un contenedor en ejecución.
- **docker inspect**: Nos devuelve información del contenedor.

Creando un contenedor demonio

Utilizamos la opción `-d` del comando `run`

```
$ docker run -d --name contenedor2 ubuntu /bin/sh -c "while true; do echo hello world; sleep 1; done"
7b6c3b1c0d650445b35a1107ac54610b65a03eda7e4b730ae33bf240982bba08
```

- **docker logs**: Nos devuelve el log del contenedor.
- **docker logs -f**: Seguimos visualizando los logs en tiempo real.

Creando un contenedor con un servidor web

```
$ docker run -d --name my-apache-app -p 8080:80 httpd:2.4
```

- Vemos que el contenedor se está ejecutando, además con la opción `-p` mapeamos un puerto del equipo donde tenemos instalado el docker, con un puerto del contenedor.
- Para probarlo accede desde un navegador a la ip del servidor con docker y al puerto 8080.

Configuración de contenedores con variables de entorno

- Para crear una variable de entorno al crear un contenedor usamos el flag `-e` o `--env`:

```
$ docker run -it --name prueba -e USUARIO=prueba ubuntu bash
root@91e81200c633:/# echo $USUARIO
prueba
```

Configuración de contenedores con variables de entorno

- Para crear un contenedor desde la imagen de mariadb, hay que inicializar la variable MYSQL_ROOT_PASSWORD).

```
$ docker run --name some-mariadb -e MYSQL_ROOT_PASSWORD=my-secret-pw -d mariadb

$ docker exec -it some-mariadb bash
root@9c3effd891e3:/# mysql -u root -p"$MYSQL_ROOT_PASSWORD"
...
MariaDB [(none)]>
```

Ejercicios

1. Instala docker en una máquina y configuralo para que se pueda usar con un usuario sin privilegios.
2. Crea un contenedor interactivo desde una imagen debian. Instala un paquete (por ejemplo nano). Sal de la terminal, ¿sigue el contenedor corriendo? ¿Por qué?. Vuelve a iniciar el contenedor y accede de nuevo a él de forma interactiva. ¿Sigue instalado el nano?. Sal del contenedor, y borralo. Crea un nuevo contenedor interactivo desde la misma imagen. ¿Tiene el nano instalado?
3. Crea un contenedor demonio con un servidor nginx, usando la imagen oficial de nginx. Al crear el contenedor, ¿has tenido que indicar algún comando para que lo ejecute? Accede al navegador web y comprueba que el servidor esta funcionando. Muestra los logs del contenedor.
4. Crea un contenedor con la aplicación Nextcloud, mirando la documentación en docker Hub, para personalizar el nombre de la base de datos sqlite que va a utilizar.