Certified Tech Developer
The Ultimate Degree

Infraestructura III

¡Trabajamos con contenedores con Docker!

Objetivo

Construir un ambiente de Docker que nos permitirá ver la tecnología en funcionamiento y entender su potencial.

Consigna

Sigamos los siguientes pasos para construir el ambiente.

Actualizar la distribución

Arrancar la máquina virtual y ejecutar los siguientes comandos siguientes en la terminal:

sudo apt update

sudo apt full-upgrade

Instalar utilidades

Instalar el editor nano:

sudo apt install nano

Añadir los repositorios de Docker y ejecutar los comandos siguientes en la terminal:

sudo apt update

sudo apt install apt-transport-https ca-certificates curl software-properties-common

Añadir la clave GPG oficial de Docker:

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

Comprobar que tenga la clave correcta:

sudo apt-key fingerprint 0EBFCD88

Ahora, el resultado debe ser:

pub rsa4096 2017-02-22 [SCEA]

9DC8 5822 9FC7 DD38 854A E2D8 8D81 803C 0EBF CD88

uid [desconocida] Docker Release (CE deb) <docker@docker.com>

sub rsa4096R 2017-02-22 [S]

Luego, añadir el repositorio correspondiente a la versión de Ubuntu instalada (escribir todo en una sola línea, sin las barras invertidas):

**sudo add-apt-repository **

**"deb [arch=amd64] https://download.docker.com/linux/ubuntu **

**$(lsb_release -cs) **

stable"

Actualizar APT:

sudo apt update

Comprobar que APT se conecta con el repositorio adecuado:

apt policy docker-ce

En la pantalla debe mostrarse el repositorio de la versión de Docker disponible para la versión de Ubuntu indicada (en el ejemplo se muestran la versión de Docker 18.09.1 y la versión de Ubuntu Bionic, es decir, Ubuntu 18.04):

docker-ce:

Instalados: (ninguno)

Candidato: 5:19.03.5~3-0~ubuntu-bionic

Tabla de versión:

5:19.03.5~3-0~ubuntu-bionic 500

500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages

...

Instalación de Docker

En la máquina virtual, ejecutar los siguientes comandos en la terminal:

sudo apt update

Instalar la última versión de Docker:

sudo apt install docker-ce

Arrancar el servicio Docker:

sudo systemctl start docker

Comprobar que el servicio está activo:

systemctl status docker

Se mostrará un mensaje similar a este (pulsar **Ctrl+C** para salir del texto):

docker.service - Docker Application Container Engine

Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)

Active: active (running) since ...

...

Imagen hello-world

Dos conceptos fundamentales de Docker son los contenedores y las imágenes. Docker encapsula las aplicaciones en contenedores. Un contenedor es el equivalente a una máquina virtual de la virtualización clásica, pero mucho más ligera porque utiliza recursos del sistema operativo del host. Las aplicaciones de cada contenedor **ven** un sistema operativo, que puede ser diferente en cada contenedor, pero quien realiza el trabajo es el sistema operativo común que hay por debajo.

Docker crea los contenedores a partir de imágenes. Las imágenes son una especie de plantillas que contienen —como mínimo— todo el software que necesita la aplicación para ponerse en marcha. Las imágenes se pueden crear a partir de otras imágenes más básicas, incluyendo software adicional en forma de capas. Todos los contenedores creados a partir de una imagen contienen el mismo software, aunque en el momento de su creación se pueden personalizar algunos detalles. En la máquina virtual, ejecutaremos los siguientes comandos en la terminal.

Primero, comprobar que no hay ningún contenedor creado. El comando **a** hace que se muestren también los contenedores detenidos. Sin ella, se muestran solo los contenedores que estén en marcha:

sudo docker ps -a

O también:

sudo docker container ls -a

La respuesta, en forma de tabla, será la siguiente. La cabecera de la tabla ocupa dos líneas, ya que es una tabla con muchos campos.

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

Comprobar que inicialmente tampoco disponemos de ninguna imagen:

sudo docker image ls

La respuesta, en forma de tabla, será la siguiente:

REPOSITORY TAG IMAGE ID CREATED SIZE

Docker crea los contenedores a partir de imágenes locales (ya descargadas), pero si —al crear el contenedor— no se dispone de la imagen local, Docker descarga la imagen de su repositorio. La orden más simple para crear un contenedor es:

sudo docker run IMAGEN

Crear un contenedor con la aplicación de ejemplo **hello-world**. La imagen de este contenedor se llama, justamente, **hello-world**:

sudo docker run hello-world

Como no tenemos todavía la imagen en nuestro ordenador, Docker descarga la imagen, crea el contenedor y lo pone en marcha. En este caso, la aplicación que contiene el contenedor **hello-world** simplemente escribe un mensaje de salida al arrancar e inmediatamente se detiene el contenedor. La respuesta será similar a esta:

Unable to find image 'hello-world:latest' locally

latest: Pulling from library/hello-world

1b930d010525: Pull complete

Digest: sha256:4fe721ccc2e8dc7362278a29dc660d833570ec2682f4e4194f4ee23e415e1064

Status: Download newer image from hello-wolrd:latest

Hello from Docker!

This message shows that your installation appears to be working correctly.

...

Si ahora listamos las imágenes existentes:

sudo docker image ls

Se mostrará información de la imagen creada:

REPOSITORY TAG IMAGE ID CREATED SIZE

hello-world latest fce209e99eb9 4 weeks ago 1.84 kB

Y si listamos los contenedores existentes:

sudo docker ps -a

Se mostrará información del contenedor creado:

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

614B4c431ffa hello-world "/hello" 2 minutes ago Exited (0) 2 minutes ago hopeful_elbakyan

Cada contenedor tiene un identificador (ID) y un nombre distinto. Docker **bautiza** los contenedores con un nombre peculiar, compuesto de un adjetivo y un apellido.

Podemos crear tantos contenedores como queramos a partir de una imagen. Una vez que la imagen está disponible localmente, Docker no necesita descargarla y el proceso de creación del contenedor es inmediato (aunque en el caso de **hello-world** la descarga es rápida, con imágenes más grandes la descarga inicial puede tardar un rato).

Normalmente, se aconseja usar siempre el comando **-d**, que arranca el contenedor en segundo plano (**detached**) y permite seguir teniendo acceso a la **shell** (aunque con **hello-world** no es estrictamente necesario porque dicho contenedor se detiene automáticamente tras mostrar el mensaje).

Al crear el contenedor **hello-world** con el comando **-d** no se muestra el mensaje, simplemente muestra el identificador completo del contenedor:

sudo docker run -d hello-world

1ae54736196021523a2b21c123fd671253e62150daccd882374

Si ahora listamos los contenedores existentes:

sudo docker ps -a

Se mostrarán los dos contenedores:

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

1ae547361960 hello-world "/hello" 4 seconds ago Exited (0) 3 seconds ago distracted_banach

614B4c431ffa hello-world "/hello" 5 minutes ago Exited (0) 5 minutes ago hopeful_elbakyan

Los contenedores se pueden destruir mediante el comando **rm**, haciendo referencia a ellos mediante su nombre o su ID. No es necesario indicar el ID completo, basta con escribir los

primeros caracteres (de manera que no haya ambigüedades). Podemos borrar los dos contenedores existentes de la siguiente manera:

sudo docker rm 1ae

1ae

sudo docker rm hopefukl_elbakyan

hopefukl_elbakyan

Comprobamos que ya no quedan contenedores:

sudo docker ps -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

También, podemos dar nombre a los contenedores al crearlos:

sudo docker run -d --name=hola-1 hello-world

Al haber utilizado el comando **-d** únicamente se mostrará el ID completo del contenedor:

54e9827bd10ab2825e1b3e4d3bf7a8cbdf778b472359c655d72d9c09e753500a

Si listamos los contenedores existentes:

sudo docker ps -a

Se mostrará el contenedor con el nombre que hemos indicado:

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

54e9827bd10a hello-world "/hello" 4 seconds ago Exited (0) 3 seconds ago hola-1

Si intentamos crear un segundo contenedor con un nombre ya utilizado:

sudo docker run -d --name=hola-1 hello-world

Docker nos avisará que no es posible:

docker: error response from daemon: Conflict. The container name "/hola-1" is already in use by con

tainer "54e9827bd10ab2825e1b3e4d3bf7a8cbdf778b472359c655d72d9c09e753500a". You have to remove (or re

name) that container to be able to reuse that name.

See 'docker run --help'.

Imagen apache

En este ejercicio vamos a crear un contenedor que incluye un servidor Apache en funcionamiento a partir de una imagen pública, modificar el contenido del contenedor y, finalmente, crear una nueva imagen para crear contenedores personalizados.

Primero debemos crear un contenedor que contenga un servidor Apache a partir de la imagen **bitnami/apache**.

El comando **-P** hace que Docker asigne de forma aleatoria un puerto de la máquina virtual al puerto asignado a Apache en el contenedor. La imagen **bitnami/apache** asigna a Apache el puerto 8080 del contenedor para conexiones HTTP y el puerto 8443 para conexiones HTTPS:

sudo docker run -d -P --name=apache-1 bitnami/apache

44c9e89bdfd24fa623774b3e774b0fb0efa107f752af5161b1d6b925330a82f6

Consultamos el puerto del host utilizado por el contenedor con:

sudo docker ps -a

Se mostrará el contenedor con el nombre que hemos indicado:

CONTAINER ID IMAGE COMMAND CREATED STATUS

PORTS NAMES

44c9e89bdfd2 bitnami/apache "/app-entrypoint.sh ..." About a minute ago Up About a min

ute 0.0.0.:32769->8080/tcp, 0.0.0.0:32768->8443/tcp apache-1

Luego, abrir en el navegador la página inicial del contenedor y comprobar que se muestra una página que dice "**It works!**".

Modificar la página inicial del contenedor Apache

Tengamos en cuenta que modificar el contenido de un contenedor tal y como vamos a hacer en este apartado solo es aconsejable en un entorno de desarrollo. Hacerlo en un entorno de producción va en contra de la **filosofía** de Docker. Los contenedores de Docker están pensados como objetos de **usar y tirar**, es decir, para ser creados, destruidos y creados de nuevo tantas veces como sea necesario. En el apartado siguiente realizaremos la misma tarea de una forma más conveniente, modificando no el contenedor sino la imagen a partir de la cual se crean los contenedores.

Crear un segundo contenedor que contenga un servidor Apache a partir de la imagen **bitnami/apache**:

sudo docker run -d -P --name=apache-2 bitnami/apache

c5041d12aabdc6f58219fec176036eb352ecaea7cf81b8f7fa2ad8af801c96a2

Consultar el puerto del host utilizado por el contenedor.

sudo docker ps -a

Se mostrarán los dos contenedores creados:

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

c5041d12aabd bitnami/apache "/app-entrypoint.sh ..." About a minute ago Up About a min

ute 0.0.0.:32771->8080/tcp, 0.0.0.0:32770->8443/tcp apache-2

44c9e89bdfd2 bitnami/apache "/app-entrypoint.sh ..." 5 minutes ago Up 5 min

utes 0.0.0.:32769->8080/tcp, 0.0.0.0:32768->8443/tcp apache-1

Crear la nueva página **index.html**:

```
sudo nano index.html
```

Por ejemplo, con el siguiente contenido: **<h1>Hola, mundo!</h1>** o mejor, una página válida HTML5 (para poder escribir caracteres no ingleses):

```
<!DOCTYPE html>
<html lang="es">
<head>
 <meta charset="utf-8">
 <title>Apache en Docker</title>
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
 <h1>¡Hola, mundo!</h1>
</body>
```

</html>

Entrar en el shell del contenedor para averiguar la ubicación de la página inicial:

sudo docker exec -it apache-2 /bin/bash

Se abrirá un shell en el contenedor. El **DocumentRoot** de Apache está en el directorio **/opt/bitnami/apache/htdocs**:

I have no name!@c5041d12aabd:/opt/bitnami/apache/htdocs$

En ese directorio se encuentra el fichero **index.html** que queremos modificar:

cat index.html

<html><body><h1>It works!</h1></body></html>

Salir del shell del contenedor:

exit

Copiar el fichero **index.html** en el contenedor:

sudo docker cp index.html apache-2:/opt/bitnami/apache/htdocs/index.html

Abrir en el navegador la página inicial del contenedor y comprobar que se ha modificado.

Crear una nueva imagen

Si queremos cambiar la página inicial, la forma correcta de hacerlo en Docker es crear una nueva imagen que incluya la página modificada, de manera que cada vez que se cree el contenedor, la página inicial sea modificada.

Las imágenes se crean a partir de **Dockerfiles**, ficheros que describen los elementos que forman la imagen. Estos pueden ser muy extensos, en este caso, se trata de un Dockerfile mínimo.

Crear un directorio que contendrá el Dockerfile:

sudo mkdir mi-apache

Copiar el fichero **index.html** creado anteriormente:

sudo mv index.html mi-apache

Entrar en el directorio y crear un fichero Dockerfile:

cd mi-apache

sudo nano Dockerfile

El contenido del Dockerfile puede ser el siguiente:

FROM bitnami/apache

COPY index.html /opt/bitnami/apache/htdocs/index.html

Generar la nueva imagen.

El último argumento (y el único imprescindible) es el nombre del archivo Dockerfile que debemos utilizar para generar la imagen. Como en este caso se encuentra en el mismo directorio y tiene el nombre predeterminado Dockerfile, se puede escribir simplemente un punto (**.**).

Para indicar el nombre de la imagen se debe añadir el comando **-t**. El nombre de la imagen debe seguir el patrón **nombre-de-usuario/nombre-de-imagen**. Si la imagen solo se va a utilizar localmente, el nombre de usuario y de la imagen pueden ser cualquier palabra:

sudo docker build -t barto/mi-apache.

Crear un contenedor a partir de la nueva imagen:

sudo docker run -d -P --name=mi-apache-1 barto/mi-apache

Abrir en el navegador la página inicial del contenedor y comprobar que se ha modificado.