Administración de Sistemas - Curso 2023 / 2024

# Laboratorio 5.- Docker

**Contenido:**



**Objetivos**: Poner en práctica los comandos y técnicas para crear y gestionar contenedores Docker.

Administración de Sistemas - Curso 2023 / 2024

# 1 Preparativos

Para realizar este laboratorio es necesario un entorno con Docker Engine (DE) instalado. Las instrucciones de instalación para un sistema Ubuntu Server se encuentran en la sección "Install using the repository" del siguiente sitio: https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository

Además, en este laboratorio se va a utilizar Redis como herramienta adicional de caso de uso. Redis es una base de datos en memoria que almacena información en formato clave/valor, con opciones de almacenamiento persistente. Es muy utilizada en entornos industriales y, según db-engines.com, la 6ª base de datos más popular del mercado a fecha de octubre de 2023.

Antes de comenzar con el laboratorio, se recomienda familiarizarse con Redis visitando su página web (https://redis.io) y buscando información sobre su uso en internet.

# 2 Docker Engine

En esta parte del laboratorio se trabaja con la herramienta CLI de Docker para manipular contenedores en un entorno local.

- Ejecutar 'docker --help' en el terminal y revisar las opciones disponibles para manipular contenedores.
- Utilizando la imagen de busybox, obtener la siguiente información. Para ello, sobre-escribe el comando de arranque en cada ocasión:
  - Mostrar los sistemas de ficheros que están montados.
  - Mostrar el contenido de la carpeta /etc. ¿Hay más o menos elementos que en la carpeta /etc de un Ubuntu Server?
  - Mostrar el cuántos binarios ejecutables hay en la carpeta /bin. ¿Hay más o menos elementos que en la carpeta /bin de un Ubuntu Server?
  - Mostrar cuantas interfaces de red tiene y qué IPs tienen asignadas.
- Lanzar un contenedor de busybox con el comando "ping www.ehu.eus". Sin interrumpir su ejecución, parar el contenedor desde otro terminal con el comando "docker stop". ¿Se realiza al instante?
- Repetir la tarea anterior, pero utilizando "docker kill". ¿Qué diferencia hay con utilizar "docker stop"?
- Abrir una Shell dentro de un contenedor busybox utilizando "docker run" y realizar lo siguiente:
  - Mostrar el número de procesos en ejecución.
  - Crear un fichero llamado miFichero dentro de /home.
  - Cerrar la sesión.
- Abrir una Shell de nuevo en un contenedor busybox. ¿El fichero en /home sigue estando? ¿Por qué?

A partir de aquí se trabaja con Redis. La forma de uso más básica de Redis es utilizar los siguientes comandos desde su consola:

- set ‹clave› ‹valor› Guardar un par clave-valor, por ejemplo: set miNumero 5
- get ‹clave› Recuperar el valor de una clave, por ejemplo: get miNumero

Más adelante se dan instrucciones de cómo acceder a la consola Redis.

Realizar las siguientes tareas:

- Lanzar un contenedor con el servidor redis. Para ello, buscar en DockerHub el nombre de la imagen oficial de Redis. Si se ha lanzado correctamente, debería mostrarse "Ready to accept connections" en la salida.
- Con el contenedor en ejecución, abrir una Shell y obtener la siguiente información:
    - Mostrar el contenido de la carpeta /etc. ¿Hay más o menos elementos que en la carpeta /etc de la imagen busybox?
    - Mostrar el cuántos binarios ejecutables hay en la carpeta /bin. ¿Hay más o menos elementos que en la carpeta /bin de busybox?
- Con el contenedor en ejecución, ejecutar el comando "redis-cli" dentro de él para acceder a una consola de Redis. Añadir una variable llamada miVar con valor 7. Salir de la consola Redis.
- Sin parar el contenedor, volver a entrar en la consola Redis. Leer el contenido de la variable miVar.
- Reiniciar el contenedor Redis y entrar a la consola. ¿Es posible leer el contenido de la variable miVar?
- Para acceder al contenedor mientras estaba en marcha habrás utilizado el comando "docker exec". ¿Qué diferencia hay al utilizar los siguientes parámetros?: -i, -t, -it, ninguno de ellos.

# 3 Imágenes Docker propias

En esta parte del laboratorio se plantean 3 tareas sobre crear imágenes propias utilizando ficheros Dockerfile.

La primera tarea es crear una imagen propia que ejecute un servidor Redis. El resultado será una imagen equivalente a la imagen oficial de Redis utilizada en la sección anterior del laboratorio, pero hecha por nosotros.

Seguir estos pasos:

- Crear un directorio nuevo en nuestro sistema y crear un fichero Dockerfile dentro. El Dockerfile deberá tener las siguientes instrucciones:
    - Utilizar como imagen base "ubuntu".
    - Instalar el paquete "redis" con apt.
    - Ejecutar como comando de arranque "redis-server".
- Utilizar los comandos de Docker para crear una imagen a partir del Dockerfile con el nombre ‹usuario›/‹redis-ubuntu› donde ‹usuario› es un nombre que vosotros elijáis, p.e. ulopez para Unai Lopez.
- En el proceso de creación de la imagen, ¿cuántos contenedores intermedios se han creado? ¿Por qué?
- Lanzar un contenedor con la imagen recién creada y abrir una Shell mientras esté en ejecución. Utilizar los comandos vistos en la sección anterior para crear una variable "miOtraVar" con valor 8. Salir de la consola y de la Shell.
- Abrir una Shell de nuevo en el contenedor y verificar que la variable "miOtraVar" mantiene su valor.

La segunda tarea es crear una imagen de las mismas características que la anterior, pero utilizando "alpine" como imagen base en lugar de "ubuntu":

- Crear un directorio nuevo en el sistema y crear un fichero Dockerfile dentro. El Dockerfile deberá tener las siguientes instrucciones:
    - Utilizar como imagen base "alpine".
    - Instalar el paquete "redis" con el sistema de paquetes de Alpine (Alpine no utiliza apt).
    - Ejecutar como comando de arranque "redis-server".
- Utilizar los comandos de Docker para crear una imagen a partir del Dockerfile con el nombre ‹usuario›/‹redis-alpine› donde ‹usuario› es el nombre elegido en la tarea anterior.
- Lanzar un contenedor con la imagen y verificar que funciona correctamente: abrir una Shell al contenedor y utilizar la consola de Redis para añadir una variable "miVar" con valor 9.

- Parar el contenedor y comparar los tamaños de las imágenes ‹redis-ubuntu› y ‹redis-alpine›. Teniendo en cuenta que ambas cumplen el mismo propósito, ¿cuál ocupa menos?

La tercera tarea es crear una imagen propia que incluya ficheros existentes en nuestro sistema. En esta tarea se creará una imagen que tendrá un servidor web Python simple para servir un único fichero HTML.

Seguir estos pasos:

- Crear un directorio nuevo en nuestro sistema.
- Dentro del nuevo directorio, crear un fichero index.html que contenga el código HTML que tenéis disponible al final de este enunciado.
- Dentro del nuevo directorio, crear un fichero Dockerfile con las siguientes instrucciones:
  - Utilizar como imagen base “ubuntu”.
  - Instalar el paquete “python3”.
  - Copiar el fichero index.html dentro de la imagen, en el directorio /miWeb. Para ello hay que utilizar la instrucción COPY de los Dockerfile. Documentación sobre COPY en: https://docs.docker.com/engine/reference/builder/#copy
  - Ubicarse dentro de /miWeb como directorio de trabajo. Utilizar la instrucción WORKDIR: https://docs.docker.com/engine/reference/builder/#workdir
  - Ejecutar como comando de arranque “python3 -m http.server 1080”
- Crear la imagen utilizando Docker con el nombre ‹usuario›/simple-web, donde ‹usuario› es un nombre que vosotros elijáis.
- Lanzar un contenedor con la imagen recién creada. Ya que el contenedor va a servir una web, es necesario redirigir uno de los puertos del contenedor al exterior para que sea usable. Para ello, es necesario utilizar el parámetro ‘-p’ de “docker run” o de “docker start”:
  https://docs.docker.com/engine/reference/commandline/run/#publish
  Utilizar ‘-p’ para redirigir el puerto 1080 del contenedor al puerto 80 del anfitrión.
- Abrir un navegador en el sistema en la dirección http://‹IP-DE-VUESTRA-MAQUINA› y comprobar que se muestra la web. Al usar Google Cloud, para que la web se muestre, el Firewall debe permitir tráfico al puerto 80 de vuestra instancia.

Código HTML para la web:

```
<!DOCTYPE html>
  <html>
    <head> <title>Web de prueba de Docker</title> </head>
    <body> <h1>Hola!</h1> <p>Esta es una web muy simple.</p> </body>
  </html>
```