Prácticas de Docker

Actividad 1

Una vez tenemos instalada la aplicación vamos a descargar una serie de imágenes que son de uso común y sobre las cuales realizaremos varias de las prácticas del sistema. En concreto, debes descargar las siguientes imágenes de Docker Hub:

- ubuntu:18.04
- centos:8
- debian:9
- mariadb:latest
- mysql:5.7
- httpd
- tomcat:9.0.39-jdk11
- jenkins/jenkins:lts
- php:7.3-apache

Una vez hayas descargado todas las imágenes, ejecuta la orden de **Docker cli** correspondiente para mostrar las imágenes que te acabas de descargar y realiza un pantallazo del resultado.

Arranca un contenedor de la imagen **ubuntu:18.04** al que llamaremos *ubuntu*. Una vez arrancado realizar las siguientes operaciones:

1. Salir del contenedor y capturar un pantallazo donde se pueda comprobar que ese contenedor se ha parado.
2. Rearrancar el contenedor desde tu equipo y capturar un pantallazo donde se pueda comprobar que está funcionando.
3. Sin entrar en el contenedor, mostrar por pantalla el fichero **/etc/os-release** y capturar un pantallazo del resultado.

Actividad 2

Para poder superar esta actividad debes realizar las siguientes acciones:

1. Arranca un contenedor que ejecute una instancia de la imagen **php:7.3-apache**, que se llame *web* y que sea accesible desde tu equipo en el puerto 8181.

2. Colocar en el directorio raíz del servicio web de dicho contenedor un fichero llamado **index.html** con el siguiente contenido:

   <h1>HOLA SOY XXXXXXXXXXXXXXX</h1>

   Deberás sustituir XXXXXXXXXXX por tu nombre y tus apellidos.

3. Colocar en ese mismo directorio raíz un archivo llamado **index.php** con el siguiente contenido:

   <?php phpinfo(); ?>

4. Arrancar un contenedor que se llame **bbdd** y que ejecute una instancia de la imagen *mariadb* para que sea accesible desde el puerto 3336.

   Antes de arrancarlo visitar la página del contenedor en Docker Hub (https://hub.docker.com/_/mariadb) y establecer las variables de entorno necesarias para que:

   - La contraseña de root sea root.
   - Crear una base de datos automáticamente al arrancar que se llame prueba.
   - Crear el usuario invitado con la contraseña invitado.

Deberás realizar los siguientes pantallazos:

- Pantallazo que desde el navegador muestre el fichero **index.html**
- Pantallazo que desde el navegador muestre el fichero **index.php**
- Pantallazo donde desde un cliente de base de datos se pueda observar que hemos podido conectarnos al servidor de base de datos con el usuario creado y que se ha creado la base de datos prueba (**show databases**)

Actividad 3

Partiendo de los contenedores en ejecución de la tarea anterior y ejecutando la orden de **docker cli** adecuada obtener la siguiente información realizando un pantallazo cada vez:

- Dirección IP del contenedor web
- Redirección de puertos del contenedor web
- Dirección IP del contenedor bbdd
- Redirección de puertos del contenedor bbdd

Para poder practicar la gestión de imágenes realizaremos las siguientes operaciones:

1. Descargar una nueva imagen desde Docker Hub, la imagen Ubuntu:20.04. Capturaremos un pantallazo donde usando el comando docker adecuado podamos comprobar que efectivamente esa imagen se ha descargado.
2. Obtener toda la información de esa imagen y volcarla el fichero **info.txt.**
3. Instanciar esa imagen creando un contenedor llamado **modulo3**. Captura un pantallazo donde usando el comando docker adecuado podemos comprobar que efectivamente ese contenedor se ha creado (en ejecución o no).
4. Intentar borrar la imagen Ubuntu:20.04. *¿Has podido borrar la imagen? Responde razonadamente.*
5. Realizar las operaciones necesarias para poder borrar la imagen. Captura un pantallazo de cada una de ellas.

Actividad 4

Para poder superar esta actividad debes realizar las siguientes tareas:

1.- Crear los siguientes volúmenes con la orden *docker volume*:

- *volumen_datos*
- *volumen_web*

2.- Una vez creados estos contenedores:

- Arrancar un contenedor llamado *c1* sobre la imagen **php:7.4-apache** que monte el *volumen_web* en la ruta **/var/www/html**
- Arrancar un contenedor llamado *c2* sobre la imagen **mariadb** que monte el *volumen_datos* en la ruta **/var/lib/mysql** y cuya contraseña de root sea admin.

3.- Parar y borrar el contenedor *c2* y tras ello borrar el volumen *volumen_datos.*

4.-Para superar la actividad deberás entregar en un fichero comprimido los siguientes pantallazos:

- Pantallazo donde se puedan ver los dos contenedores creados (docker volume ls).
- Pantallazo con la orden correspondiente para arrancar el contenedor *c1* usando el *volumen_web*.
- Pantallazo con la orden correspondiente para arrancar el contenedor *c2* usando el *volumen_datos.*
- Pantallazo con el resultado la orden para borrar el *volumen_datos*.
- Pantallazo donde se pueda apreciar que ya solo queda un contenedor.
- Pantallazo donde se pueda apreciar que el contenedor *c1* usa el *volumen_web* (docker inspect).

Actividad 5

En esta actividad vamos a crear dos redes de ese tipo (BRIDGE) con los siguientes datos:

Red1
Nombre: red1
Dirección de red: 172.28.0.0
Máscara de red: 255.255.0.0
Gateway: 172.28.0.1

Red2
Nombre: red2
Es resto de los datos será proporcionados automáticamente por Docker.

Una vez creadas estas dos redes deberás entregar en un archivo comprimido los siguientes pantallazos:

1. Resultado de la orden *docker network ls* donde se puedan ver las dos redes creadas correctamente.
2. Resultado de la orden *docker network inspect red1* donde se pueda ver que la configuración de la red ha sido la correcta.
3. Resultado de la orden *docker network inspect red2* donde se pueda ver la configuración que Docker ha dado a la red.

Deberemos realizar los siguientes pasos:

1. Poner en ejecución un contenedor de la imagen **ubuntu:20.04** que tenga como hostname host1, como IP 172.28.0.10 y que esté conectado a la red1. Lo llamaremos *u1.*
2. Entrar en ese contenedor e instalar la aplicación ping (*apt update && apt install inetutils-ping*).
3. Poner en ejecución un contenedor de la imagen **ubuntu:20.04** que tenga como hostname host2 y que esté conectado a la red2. En este caso será docker el que le de una IP correspondiente a esa red. Lo llamaremos *u2.*
4. Entrar en ese contenedor e instalar la aplicación ping (*apt update && apt install inetutils-ping*).

Una vez preparados estos contenedores debemos capturar los siguientes pantallazos:

- Pantallazo donde se vea la configuración de red del contenedor *u1*.
- Pantallazo donde se vea la configuración de red del contenedor *u2*.
- Pantallazo donde desde cualquiera de los dos contenedores se pueda ver que no podemos hacer ping al otro ni por ip ni por nombre.

5. Una vez hemos constatado que los dos contenedores están en redes diferentes y aisladas, conectar la red2 al contenedor *u1* mediante la orden *docker network connect.*

- Comprobar esta última conexión y capturar un último pantallazo donde se pueda comprobar que ahora, desde el contenedor *u1*, tenemos acceso al contenedor *u2* mediante ping, tanto por nombre como por ip.

Actividad 6

En esta práctica montaremos un entorno de desarrollo para el framework Ruby on Rails utilizando como base el ***docker-compose.yml*** de Bitnami que podemos encontrar en la siguiente URL: **https://hub.docker.com/r/bitnami/rails/**

Por lo tanto, para poder superar esta actividad debes realizar las siguientes tareas:

1.- Una vez hemos descargado el fichero *docker-compose.yml* asociado deberemos modificarlo de la siguiente manera:

- Modificar el puerto de la aplicación al que nos podremos conectar desde nuestro host para que sea el puerto 8001. Recordad que sólo debemos modificar el primer puerto de la redirección.

2.- Posteriormente deberemos iniciar la aplicación multicapa con *docker-compose up.*

3.- Para superar la actividad deberemos entregar:

- El fichero *docker-compose.yml* modificado
- Un pantallazo donde se pueda ver la ejecución de la orden *docker-compose up*.
- Un pantallazo donde se pueda apreciar que la aplicación base del framework está en ejecución a través de http://127.0.0.1:8001 en vuestro navegador.

Actividad 7

1.- Descargar la siguiente imagen no
firmada https://hub.docker.com/r/testinf/holamundohtml mediante la orden *docker pull*.
2.- Borrar dicha imagen.
3.- Habilitar correctamente la variable de entorno DOCKER_CONTENT_TRUST para descargar sólo imágenes "Trusted", es decir firmadas.

```
# Añado la siguiente línea al final del fichero .bashrc que sirve para
añadir esa variable de entorno. DOCKER_CONTENT_TRUST permite
habilitar/deshabilitar la realización de subcomandos PULL/RUN/BUILD sobre
imágenes no firmadas. Si está a 1 sólo permite trabajar con imágenes
firmadas, aunque sean de mi propiedad.

export DOCKER_CONTENT_TRUST=1

# Recargo el .bashrc

> source /home/miusuario/.bashrc
```

4.- Intentar descargar de nuevo la imagen y comprobar que nos da un mensaje de error.
5.- Descargar esa imagen de manera forzosa usando el flag *---disable-content-trust*

- Pantallazo donde se pueda verificar que la imagen se descarga sin problemas inicialmente.
- Pantallazo con la orden para borrar la imagen.
- Pantallazo con la habilitación de la variable de entorno.
- Pantallazo donde se pueda comprobar que ahora la imagen no se descarga y nos sale un mensaje de error.
- Pantallazo donde se pueda comprobar que usando el flag *–disable-content-trust* en la orden *docker pull* si que se descarga pese a no estar firmada.