SISTEMAS DE GESTIÓN EMPRESARIAL

TEMA 4: DOCKER

PRÁCTICA GUIADA SOBRE DOCKER (contenedores, volúmenes, variables de entorno, ...)

El trabajo de hoy consiste en realizar los siguientes ejercicios. Como producto, tendrás que entregar un fichero PDF con las capturas de pantalla de los diferentes comandos utilizados y su resultado.

Creación de un contendedor con Apache y PHP 7.2 (en segundo plano)

En este caso vamos a utilizar la imagen oficial php:7.2-apache.

Esta imagen está configurada para servir el contenido que se encuentre dentro del directorio `/var/www/html`.

```
$ docker run -d \
--rm \
--name apache_php \
-p 80:80 \
-v c:/una/ruta/en/tu/pc:/var/www/html \
php:7.2-apache
```

Cambia `C:/una/ruta/en/tu/pc` por la ruta que estimes más adecuada

Comprobamos que el contenedor está en ejecución:

```
$ docker ps
```

Creamos el archivo `info.php` en nuestro directorio de trabajo actual con el siguiente contenido:

```
<?php
 phpinfo();
```

```
?>
```

Ahora vamos a conectarnos a un terminal del contenedor para comprobar que el volumen se ha montado correctamente.

```
docker exec -it apache_php /bin/bash
```

Abrimos un navegador y accedemos a la URL http://localhost/info.php para comprobar que la página `info.php` se sirve correctamente.

Ejercicio propuesto (*)

1. Busca una **imagen oficial** que te permita servir una web **PHP** con el servidor **Nginx**. En caso de no encontrar ninguna, ¿qué podríamos hacer?

   Crear una

Creación de un contenedor con MySQL con persistencia de datos (en segundo plano)

Vamos a utilizar la imagen oficial mysql.

Existen dos formas de añadir persistencia de datos:

1. Crear un volumen interno gestionado por Docker.
2. Crear un volumen de tipo *bind mount* donde montamos un directorio de nuestra máquina local en un directorio dentro del contenedor.

Solución 1. Crear un volumen interno gestionado por Docker

```
docker volume create mysql_data
```

```
docker run -d \
--rm \
--name mysqlc \
-e MYSQL_ROOT_PASSWORD=root \
-p 3306:3306 \
-v mysql_data:/var/lib/mysql \
mysql:5.7.28
```

Abrimos un terminal en el contenedor para interaccionar con él.

```
docker exec -it mysqlc /bin/bash
```

Una vez que estamos dentro del contenedor nos conectamos desde la consola de MySQL.

```
# mysql -u root -p
```

Creamos una nueva base de datos con los siguientes datos.

```
DROP DATABASE IF EXISTS tienda;
CREATE DATABASE tienda CHARSET utf8mb4;
USE tienda;

CREATE TABLE fabricante (
  codigo INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL
);

CREATE TABLE producto (
  codigo INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  precio DOUBLE NOT NULL,
  codigo_fabricante INT UNSIGNED NOT NULL,
  FOREIGN KEY (codigo_fabricante) REFERENCES fabricante(codigo)
);

INSERT INTO fabricante VALUES(1, 'Asus');
INSERT INTO fabricante VALUES(2, 'Lenovo');
INSERT INTO fabricante VALUES(3, 'Hewlett-Packard');
INSERT INTO fabricante VALUES(4, 'Samsung');
INSERT INTO fabricante VALUES(5, 'Seagate');
INSERT INTO fabricante VALUES(6, 'Crucial');
INSERT INTO fabricante VALUES(7, 'Gigabyte');
INSERT INTO fabricante VALUES(8, 'Huawei');
INSERT INTO fabricante VALUES(9, 'Xiaomi');

INSERT INTO producto VALUES(1, 'Disco duro SATA3 1TB', 86.99, 5);
INSERT INTO producto VALUES(2, 'Memoria RAM DDR4 8GB', 120, 6);
INSERT INTO producto VALUES(3, 'Disco SSD 1 TB', 150.99, 4);
INSERT INTO producto VALUES(4, 'GeForce GTX 1050Ti', 185, 7);
INSERT INTO producto VALUES(5, 'GeForce GTX 1080 Xtreme', 755, 6);
INSERT INTO producto VALUES(6, 'Monitor 24 LED Full HD', 202, 1);
INSERT INTO producto VALUES(7, 'Monitor 27 LED Full HD', 245.99, 1);
INSERT INTO producto VALUES(8, 'Portátil Yoga 520', 559, 2);
INSERT INTO producto VALUES(9, 'Portátil Ideapd 320', 444, 2);
INSERT INTO producto VALUES(10, 'Impresora HP Deskjet 3720', 59.99, 3);
INSERT INTO producto VALUES(11, 'Impresora HP Laserjet Pro M26nw', 180, 3);
```

Comprobamos que la base de datos se ha creado correctamente.

```
mysql> SHOW TABLES;
+------------------+
```

```
| Tables_in_tienda |
+------------------+
| fabricante       |
| producto         |
+------------------+
2 rows in set (0.00 sec)
```

Comprobamos que las tablas tienen datos.

```
mysql> SELECT * FROM fabricante;
+--------+-----------------+
| codigo | nombre          |
+--------+-----------------+
|      1 | Asus            |
|      2 | Lenovo          |
|      3 | Hewlett-Packard |
|      4 | Samsung         |
|      5 | Seagate         |
|      6 | Crucial         |
|      7 | Gigabyte        |
|      8 | Huawei          |
|      9 | Xiaomi          |
+--------+-----------------+
9 rows in set (0.00 sec)
mysql> SELECT * FROM producto;
+--------+----------------------------------+--------+-------------------+
| codigo | nombre                           | precio | codigo_fabricante |
+--------+----------------------------------+--------+-------------------+
|      1 | Disco duro SATA3 1TB             |  86.99 |                 5 |
|      2 | Memoria RAM DDR4 8GB             |    120 |                 6 |
|      3 | Disco SSD 1 TB                   | 150.99 |                 4 |
|      4 | GeForce GTX 1050Ti               |    185 |                 7 |
|      5 | GeForce GTX 1080 Xtreme          |    755 |                 6 |
|      6 | Monitor 24 LED Full HD           |    202 |                 1 |
|      7 | Monitor 27 LED Full HD           | 245.99 |                 1 |
|      8 | Porttil Yoga 520                 |    559 |                 2 |
|      9 | Porttil Ideapd 320               |    444 |                 2 |
|     10 | Impresora HP Deskjet 3720        |  59.99 |                 3 |
|     11 | Impresora HP Laserjet Pro M26nw  |    180 |                 3 |
+--------+----------------------------------+--------+-------------------+
11 rows in set (0.00 sec)
```

Una vez que hemos llegado a este punto tendríamos **la base de datos almacenada en el volumen** `mysql_data`, de modo que si elimimanos el contenedor y volvemos a crear uno nuevo que haga uso del mismo volumen, tendríamos acceso a la misma base de datos. Vamos a comprobarlo.

Comprobamos que el contenedor está en ejecución.

```
docker ps
```

Detenemos el contenedor. Como hemos iniciado el contenedor con la opción `--rm` al detenerlo se eliminará automáticamente.

```
$ docker stop mysqlc
```

Comprobamos que el contenedor se ha detenido y se ha eliminado correctamente.

```
$ docker ps -a
```

Ejercicios

1. Instancia un nuevo contenedor con el nombre `mysql_container_2`, que haga uso del volumen `mysql_data` donde está almacenada la base de datos `tienda`.

```
PS C:\> docker run  -d --rm --name mysql_container_2 -e MYSQL_ROOT_PASSW
ORD=root -p 3306:3306 -v mysql_data:/var/lib/mysql mysql:5.7.28
2eb59a0a41ee9c833cbbfb0184645d21763c8b9c9eea0311b0c81d0da7f08c67
PS C:\>
```

2. Abre un terminal en el contenedor que acabas de crear (`mysql_container_2`) para interaccionar con él.

```
2eb59a0a41ee9c833cbbfb0184645d21763c8b9c9eea0311b0c81d0da7f08c67
PS C:\> docker exec -it mysql_container_2 /bin/bash
root@2eb59a0a41ee:/#
```

3. Una vez dentro del contenedor inicia una conexión a la consola de MySQL.

```
ORD=root -p 3306:3306 -v mysql_data:/var/lib/mysql mysql:5.7.28
2eb59a0a41ee9c833cbbfb0184645d21763c8b9c9eea0311b0c81d0da7f08c67
PS C:\> docker exec -it mysql_container_2 /bin/bash
root@2eb59a0a41ee:/# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.28 MySQL Community Server (GPL)
```

4. Comprueba que la base de datos `tienda` existe y tiene datos.

```
mysql> use tienda
Reading table information for co
You can turn off this feature to

Database changed
mysql> SHOW TABLES;
+------------------+
| Tables_in_tienda |
+------------------+
| fabricante       |
| producto         |
+------------------+
2 rows in set (0.00 sec)

mysql>
```

5. ¿Qué comando tendríamos que ejecutar si quisiéramos eliminar el volumen `mysql_data`?

```
PS C:\> docker stop mysql_container_2
mysql_container_2
PS C:\> docker volume rm mysql_data
mysql_data
PS C:\>
```

Conectar un contenedor phpMyAdmin con MySQL

Para este ejemplo usaremos la imagen oficial de phpmyadmin.

Para conectar dos contenedores podemos hacerlo de dos formas:

1. Utilizando `legacy container links` con el flag `--link`, en la `bridge network`.
2. Utilizando una `user-defined bridge network`.

Solución 1. `Legacy container links` **con el flag** `--link`**, en la** `bridge network`

```
docker run -d \
--rm \
--name mysqlc \
-p 3306:3306 \
-e MYSQL_ROOT_PASSWORD=root \
-v mysql_data:/var/lib/mysql \
mysql:5.7.28
```

Le pasamos la variable de entorno `-e PMA_ARBITRARY=1` para que en la página principal de phpMyAdmin me aparezca un campo en el formulario donde pueda indicar el servidor al que quiero conectarme.

```
docker run -d \
--rm \
--link mysqlc \
-e PMA_ARBITRARY=1 \
-p 8080:80 \
 phpmyadmin
```

Solución 2. Utilizando una `user-defined bridge network`

En primer lugar creamos una `user-defined bridge network`.

```
$ docker network create my-net
```

Creamos un contenedor con MySQL indicando que queremos que esté en la red `--network my-net`.

```
$ docker run -d \
--rm \
--name mysqlc \
--network my-net \
-p 3306:3306 \
-e MYSQL_ROOT_PASSWORD=root \
-v mysql_data:/var/lib/mysql \
 mysql:5.7.28
```

Creamos un contenedor con phpMyAdmin indicando que queremos que esté en la red `--network my-net`.

```
$ docker run -d \
--rm \
--network my-net \
-e PMA_ARBITRARY=1 \
-p 8080:80 \
 phpmyadmin
```

Comprobamos que el contendedor `phpMyAdmin` puede conectar con el contenedor `mysql` abriendo un navegador web y accediendo a la URL: http://localhost:8080.

Para eliminar la red que hemos creado ejecutamos lo siguiente.

```
docker network rm my-net
```

Ejercicio

1. Busca en Docker Hub las imágenes del sistema gestor de bases de datos **PostgreSQL** y **phpPgAdmin**.
2. Crea una instancia de PostreSQL y phpPgAdmin, de modo que desde phpPgAdmin pueda conectarme a PostgreSQL.

> Necesitarás consultar en Docker Hub cuáles son las variables de entorno que necesitas para ambos casos.

La entrega debe realizarse en formato PDF.