Práctica 7. Creación de un contenedor Docker con phpMyAdmin

Apuntes de BD para DAW, DAM y ASIR

José Juan Sánchez Hernández

Curso 2023/2024

Índice

Índice de figuras

Índice de cuadros

1 Creación de un contenedor Docker con phpMyAdmin

1.1 Requisitos

Para poder ejecutar contenedores Docker es necesario tener instalado Docker Community Edition (CE) en nuestro equipo.

En la web oficial encontrará la información necesaria para realizar la instalación de Docker CE sobre Windows, macOS, Ubuntu, Debian, Fedora y CentOS.

1.2 Cómo enlazar phpMyAdmin con un contenedor Docker con MySQL o MariaDB

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`.

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

Los enlaces permiten que los contenedores se descubran entre sí y transfieran de manera segura información sobre un contenedor a otro contenedor. Para crear un enlace se utiliza el flag *–link*.

En primer lugar debe existir un contenedor con **MySQL Server**.

```
docker run -d \
--rm \
--name mysqlc \
-p 3306:3306 \
-e MYSQL_ROOT_PASSWORD=root \
-v mysql_data:/var/lib/mysql \
mysql:8.0
```

Una vez que la instancia de **MySQL** está en ejecución podemos crear el contenedor con phpMyAdmin.

```
docker run -d \
--rm \
--name phpmyadminc \
--link mysqlc \
-e PMA_HOST=mysqlc \
-p 8080:80 \
phpmyadmin
```

La variable de entorno `PMA_HOST` nos permite indicar el nombre del contenedor con el que quiero conectarme desde phpMyAdmin.

En lugar de la variable de entorno `PMA_HOST` podía haber utilizado la variable `PMA_ARBITRARY=1`.

La única diferencia es que con `PMA_ARBITRARY=1` nos aparece un campo de texto en la página de login de phpMyAdmin donde tenemos que indicar el nombre del contenedor al que queremos conectarme y con `PMA_HOST` no hay que escribir nada porque se configura automáticamente.

Con el flag **`--link mysqlc`** hemos creado un enlace entre los contenedores `mysql` y `phpmyadminc`.

En el archivo **/etc/hosts** del contenedor `phpmyadminc` se ha añadido una nueva línea que permite resolver la dirección IP del contenedor de MySQL a partir de su nombre (`mysqlc`) o su ID.

Por ejemplo si el ID del contenedor de **MySQL** (`mysqlc`) fuese `8411f6064e44` el archivo `/etc/hosts` tendría un contenido similar a este:

```
127.0.0.1 localhost
::1 localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
172.17.0.3  mysqlc 8411f6064e44
172.17.0.4  c25ca9a48fb3
```

Podemos comprobar que el contendedor `phpmyadminc` puede conectar con el contenedor `mysqlc` abriendo un navegador web y accediendo a la URL: http://localhost:8080.

1.2.2 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:8.0
```

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

```
docker run -d \
--rm \
--name phpmyadminc \
--network my-net \
-e PMA_HOST=mysqlc \
-p 8080:80 \
phpmyadmin
```

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

Para eliminar la red que hemos creado ejecutamos lo siguiente.

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

1.3 Ejemplo de uso con MySQL y phpMyAdmin

Creamos un contenedor con MySQL con persistencia de datos.

```
docker run -d \
--rm \
--name mysqlc \
-p 3306:3306 \
-e MYSQL_ROOT_PASSWORD=root \
-v mysql_data:/var/lib/mysql \
mysql:8.0
```

Creamos un contenedor con phpMyAdmin que esté enlazado con el contenedor anterior.

```
docker run -d \
--rm \
--name phpmyadminc \
--link mysqlc \
-e PMA_HOST=mysqlc \
-p 8080:80 \
phpmyadmin
```

1.4 Ejemplo de uso con MariaDB y phpMyAdmin

Creamos un contenedor con MariaDB con persistencia de datos.

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

```
mariadb:10
```

Creamos un contenedor con phpMyAdmin que esté enlazado con el contenedor anterior.

```
docker run -d \
--rm \
--name phpmyadminc \
--link mariadbc \
-e PMA_HOST=mariadbc \
-p 8080:80 \
phpmyadmin
```

1.5 Cómo comprobar que los contenedores están en ejecución

Una vez que hemos iniciado los contenedores podemos comprobar que se están ejecutando con el siguiente comando:

```
docker ps
```

Deberíamos obtener una salida similar a esta.

```
CONTAINER ID        IMAGE                  COMMAND                  CREATED
                STATUS              PORTS                                      NAMES
19a928aac23c        phpmyadmin/phpmyadmin  "/run.sh …supervisord"   3 seconds
    ago       Up 3 seconds        9000/tcp, 0.0.0.0:8080->80/tcp   phpmyadminc
acd81669c572        mariadb                "docker-entrypoint.…s"   19 seconds
     ago      Up 18 seconds       0.0.0.0:3306->3306/tcp           mariadbc
```

1.6 Cómo detener los contenedores

Para detener un contenedor en primer lugar tenemos que conocer cuál es su ID. Para obtenerlo podemos hacer uso del comando `docker ps`.

```
docker ps

CONTAINER ID        IMAGE                  COMMAND                  CREATED
                STATUS              PORTS                                      NAMES
19a928aac23c        phpmyadmin/phpmyadmin  "/run.sh …supervisord"   3 seconds
    ago       Up 3 seconds        9000/tcp, 0.0.0.0:8080->80/tcp   phpmyadminc
acd81669c572        mariadb                "docker-entrypoint.…s"   19 seconds
     ago      Up 18 seconds       0.0.0.0:3306->3306/tcp           mariadbc
```

En la primera columna podemos ver cuál es el `CONTAINER ID`. Una vez localizado el identificador ejecutamos el comando `docker stop` y le pasamos como parámetro el identificador del contenedor que queremos detener.

Para el caso anterior deberíamos ejecutar:

```
docker stop 19a928aac23c
```

```
docker stop acd81669c572
```

1.7 Ejemplo de `docker-compose.yml` con MySQL y phpMyAdmin

Docker Compose es una utilidad que nos permite gestionar varios contenedores Docker de una forma sencilla. Para ello, utiliza un archivo YAML donde se definen y configuran los servicios, los volúmenes y las redes que queremos crear.

En este caso vamos a crear un archivo `docker-compose.yml` para definir dos servicios: `mysql` y `phpmyadmin`.

```yaml
version: '3'

services:
  mysql:
    image: mysql:8.0
    ports:
      - 3306:3306
    environment:
      - MYSQL_ROOT_PASSWORD=root
      - MYSQL_DATABASE=database
      - MYSQL_USER=user
      - MYSQL_PASSWORD=password
    volumes:
      - mysql_data:/var/lib/mysql

  phpmyadmin:
    image: phpmyadmin
    ports:
      - 8080:80
    environment:
      - PMA_HOST=mysql
    depends_on:
      - mysql

volumes:
  mysql_data:
```

Puede descargar el archivo `docker-compose.yml` de GitHub.

Con la opción `depends_on` indicamos que el servicio `phpmyadmin` depende del servicio `mysql` y que no podrá iniciarse hasta que el servicio de `mysql` se haya iniciado.

Para iniciar los servicios en segundo plano con la utilidad `docker-compose` tenemos que ejecutar el siguiente comando.

```
docker-compose up -d
```

Para detener los servicios y mantener el volumen que hemos creado podemos ejecutar:

```
docker-compose down
```

Si además de detener los servicios queremos eliminar el volumen que hemos creado tenemos que añadir el parámetro `-v`.

```
docker-compose down -v
```

Para comprobar el estado de los servicios podemos ejecutar:

```
docker-compose ps
```

1.8 Referencias

- Web oficial de phpMyAdmin
- Imagen de phpMyAdmin en Docker Hub
- Web oficial de Docker
- Instalación de Docker Community Edition (CE)
- Instalación de Docker Community Edition (CE) en Windows
- Instalación de Docker Community Edition (CE) en macOS
- Instalación de Docker Community Edition (CE) en Ubuntu
- Instalación de Docker Community Edition (CE) en Debian
- Instalación de Docker Community Edition (CE) en Fedora
- Instalación de Docker Community Edition (CE) en CentOS
- Docker Hub
- Docker Compose
- YAML

2 Licencia