Práctica 6. Creación de un contenedor Docker con MariaDB

Apuntes de BD para DAW, DAM y ASIR

José Juan Sánchez Hernández

Curso 2023/2024

Índice

1 Creación de un contenedor Docker con MariaDB **1**

1.1 Requisitos 1
1.2 Cómo crear un contenedor sin persistencia de datos 1
1.3 Cómo crear un contenedor con persistencia de datos 2
1.4 Cómo comprobar que el contenedor está en ejecución 3
1.5 Cómo conectar con MariaDB 4
1.6 Cómo detener el contenedor 4
1.7 Referencias 4

2 Licencia **6**

Índice de figuras

Índice de cuadros

1 Creación de un contenedor Docker con MariaDB

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 crear un contenedor sin persistencia de datos

Un contenedor Docker que no tiene persistencia de datos quiere decir que cuando finalice la ejecución perderá todo el contenido que hayamos creado durante la ejecución. Es decir, si durante la ejecución del contenedor hemos creado varias bases de datos en MariaDB, éstas se perderán cuando el contenedor se detenga.

El comando que podríamos usar para lanzar nuestro contenedor Docker con MariaDB sin persistencia de datos podría ser el siguiente:

```
1  docker run -d --rm --name mariadb -e MYSQL_ROOT_PASSWORD=root -p 3306:3306
       mariadb
```

- `docker run` es el comando que nos permite crear un contenedor a partir de una imagen Docker.
- El parámetro `-d` nos permite ejecutar el contenedor en modo *detached*, es decir, ejecutándose en segundo plano.
- El parámetro `--rm` hace que cuando salgamos del contenedor, éste se elimine y no ocupe espacio en nuestro disco.
- El parámetro `--name` nos permite asignarle un nombre a nuestro contenedor. Si no le asignamos un nombre Docker nos asignará un nombre automáticamente.
- El parámetro `e` es para pasarle al contenedor una variable de entorno. En este caso le estamos pasando la variable de entorno `MYSQL_ROOT_PASSWORD` con el valor de la contraseña que tendrá el usuario `root` para MySQL Server.
- El parámetro `-p` nos permite mapear los puertos entre nuestra máquina local y el contenedor. En este caso, estamos mapeando el puerto `3306` de nuestra máquina local con el puerto `3306` del contenedor.
- `mariadb` es el nombre de la imagen. Si no indicamos la versión utilizará la última disponible. Si no se indica lo contrario buscará las imágenes en el repositorio oficial Docker Hub.

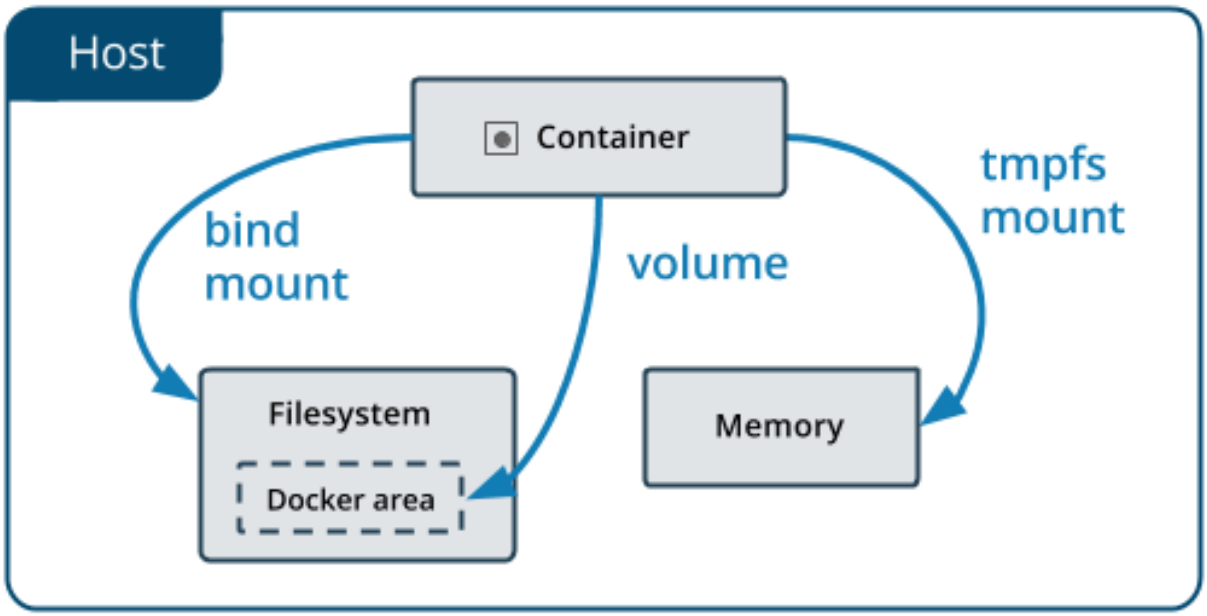
1.3 Cómo crear un contenedor con persistencia de datos

Si queremos que los datos del contenedor sean persistentes tenemos que crear un **volumen** donde vamos a indicar el directorio de nuestra máquina local vamos a almacenar el directorio `/var/lib/mysql`, que es el directorio que utiliza MariaDB para almacenar las bases de datos.

Para crear un volumen utilizamos el parámetro `-v`.

Docker nos ofrece dos posibilidades para implementar persistencia de datos en los contenedores:

- `bind mount`: pueden estar almacenados en cualquier directorio del sistema de archivos de la máquina host. Estos archivos pueden ser consultados o modificados por otros procesos de la máquina host o incluso por otros contenedores Docker.
- `volume`: se almacenan en la máquina host dentro del área del sistema de archivos que gestiona Docker. Otros procesos de la máquina host no deberían modificar estos archivos, sólo deberían ser modificados por contenedores Docker.



En la documentación oficial podemos encontrar más información sobre el uso de **volumenes** en Docker.

Ejemplo de uso del parámetro `-v` para crear un volumen de tipo `bind_mount`:

```
-v /home/josejuan/data:/var/lib/mysql
```

En este caso el directorio `/home/josejuan/data` de nuestra máquina local estará sincronizado con el directorio `/var/lib/mysql` del contenedor con MariaDB.

Podemos hacer uso de la variable de entorno `$PWD` para indicar que queremos crear el volumen en nuestro directorio actual.

Ejemplo de uso del parámetro `-v` para crear un volumen de tipo `bind_mount` con la variable de entorno `$PWD`:

```
-v "$PWD":/var/lib/mysql
```

Ejemplo de uso del parámetro `-v` con un volumen de tipo `volume`:

```
-v mariadb_data:/var/lib/mysql
```

El comando que podríamos usar para lanzar nuestro contenedor Docker con MariaDB con persistencia de datos en un volumen podría ser el siguiente:

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

- `docker run` es el comando que nos permite crear un contenedor a partir de una imagen Docker.
- El parámetro `-d` nos permite ejecutar el contenedor en modo *detached*, es decir, ejecutándose en segundo plano.
- El parámetro `--rm` hace que cuando salgamos del contenedor, éste se elimine y no ocupe espacio en nuestro disco.
- El parámetro `--name` nos permite asignarle un nombre a nuestro contenedor. Si no le asignamos un nombre Docker nos asignará un nombre automáticamente.
- El parámetro `e` es para pasarle al contenedor una variable de entorno. En este caso le estamos pasando la variable de entorno `MYSQL_ROOT_PASSWORD` con el valor de la contraseña que tendrá el usuario `root` para MySQL Server.
- El parámetro `-p` nos permite mapear los puertos entre nuestra máquina local y el contenedor. En este caso, estamos mapeando el puerto `3306` de nuestra máquina local con el puerto `3306` del contenedor.
- El parámetro `-v` nos permite crear un volumen para tener persistencia de datos al finalizar el contenedor.
- `mariadb` es el nombre de la imagen. Si no indicamos la versión utilizará la última versión disponible que está etiquetada como `latest`. Si no se indica lo contrario buscará las imágenes en el repositorio oficial Docker Hub.

1.4 Cómo comprobar que el contenedor está en ejecución

Una vez que hemos iniciado el contenedor podemos comprobar que se está ejecutando con el siguiente comando:

```
docker ps
```

Deberíamos obtener una salida similar a esta.

```
CONTAINER ID        IMAGE               COMMAND                  CREATED
                 STATUS              PORTS                    NAMES
52bcaee9a157        mariadb             "docker-entrypoint.…s"   4 seconds ago
          Up 2 seconds        0.0.0.0:3306->3306/tcp   mariadb
```

1.5 Cómo conectar con MariaDB

Una vez que MariaDB está en ejecución podemos conectarnos con cualquier cliente: MySQL Workbench, PHPMyAdmin, Adminer, etc.

Los datos de conexión serán:

- Host: 127.0.0.1
- Puerto: 3306
- Usuario: root
- Password: root

1.6 Cómo detener el contenedor

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

```
1 docker ps
2
3 CONTAINER ID        IMAGE               COMMAND                  CREATED
                 STATUS              PORTS                     NAMES
4 52bcaee9a157        mariadb             "docker-entrypoint.…s"   4 seconds ago
          Up 2 seconds        0.0.0.0:3306->3306/tcp   mariadb
```

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:

```
1 docker stop 52bcaee9a157
```

1.7 Referencias

- Web oficial de MariaDB
- Imagen de MariaDB 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

- Uso de *volumes* en Docker

2 Licencia