Dockerizar Spring Boot MySQL

Me voy a basar en estos tutoriales:

**Dockerize Spring Boot Application with MySQL**

<https://youtu.be/S2s28PCg4M4>

May 24, 2022, 16 minutos.

**Dockerize Spring Boot Application with MySQL using Docker compose**

<https://youtu.be/6hMHziv0T2Y>

May 27, 2022, 11 minutos.

Voy a hacer muchos cambios. Realmente lo que quiero es tener un apunte para las materias.

# Dos opciones para dockerizar

Vamos a ver dos opciones para dockerizar una aplicación de Spring Boot con MySQL:

* Crear una network de containers Docker.
* Usando docker-compose.

# Crear una network de containers Docker

En la extensión para Docker de VS Code, borrar todos los containers, imágenes, volúmenes y networks. Esto no es estrictamente necesario, pero trabajar con un espacio limpio ayuda a pensar con claridad.

## PASO 1 Descargar la imagen de mysql

Descargar la imagen de mysql desde docker hub:

docker pull mysql

Al terminar la descarga, comprobamos que tenemos la imagen mysql:latest.

## PASO 2 Crear una network

Crear una network para que los containers se puedan comunicar entre sí. Es un bridge entre dos servicios:

docker network create springboot-mysql-net

Comprobamos que tenemos la network de nombre springboot-mysql-net, pero todavía no tiene ningún container asociado.

## PASO 3 Correr un container para MySQL

Correr un container para MySQL en la network:

docker run -d --name mysqldb -p 3306:3306 --net springboot-mysql-net -e MYSQL\_ROOT\_PASSWORD=1234 -e MYSQL\_DATABASE=anime -e MYSQL\_USER=cruduser -e MYSQL\_PASSWORD=1234 mysql

Comprobamos que ahora tenemos corriendo el container llamado mysqldb. Vemos que tiene asociado un volumen, aunque no le pusimos nada explícitamente. Y vemos que está asociado a la network correspondiente.

Si vamos comprobar los volúmenes, vemos que ahora tenemos uno asociado con nuetro container.

Si vamos a las networks, comprobamos que tenemos la network que creamos. Si nos dice que no tiene containers asociados, refrescamos la vista, y aparecerá el correspondiente container.

## PASO 4 Verificar la base de datos

Verificar que se ha creado la base de datos. En la terminal de Windows ejecutamos el comando:

docker ps

Nos fijamos en los primeros tres caracteres de la ID de nuestro container. En este caso eran f76:

docker exec -it f76 bash

Aparece una ventana de bash:

mysql -ucruduser -p1234

Estamos en la consola de MySQL:

show databases;

Vemos que tenemos anime más las otras propias de la instalación de MySQL:

use anime;

Estamos en la base de datos:

show tables;

Vemos que la base de datos está vacía.

Para salir de la consola de MySQL:

exit

Estamos en bash. Para salir de bash:

exit

Estamos de nuevo en la terminal de Windows.

## PASO 5 application.properties

Actualizar el archivo application.properties:

spring.datasource.url=jdbc:mysql://mysqldb:3306/anime

spring.datasource.username=cruduser

spring.datasource.password=1234

## PASO 6 Imagen de la aplicación Spring Boot

Construir la imagen de la aplicación de Spring Boot. Antes de compilar, ir al paquete test y comentar todos los tests. Esto es porque si algún test falla la compilación no será exitosa. Recompilar la aplicación. En la terminal de VS Code:

mvn clean install

Deberíamos tener un jar recién creado en target.

En el directorio crud crear un archivo Dockerfile y pegar:

# eclipse-temurin soporta versiones recientes del JDK.

# En muchos tutoriales usan openjdk, pero la documentación

# oficial dice que está deprecado.

FROM eclipse-temurin

COPY target/\*.jar app.jar

ENTRYPOINT ["java", "-jar", "app.jar"]

Construir la imagen de la aplicación Spring Boot:

docker build -t springbootmysql .

Comprobamos en la extensión de Docker que apareció esta nueva imagen. No hay nuevos containers, ni tampoco volúmenes, ni se agregó nada a la network. Solo tenemos la imagen nueva.

## PASO 7 Container de Spring Boot

Arrancar el container de Spring Boot en la misma network:

docker run --network springboot-mysql-net --name springboot-container -p 8080:8080 -d springbootmysql

Vemos que apareció un nuevo container llamado springboot-container, instancia de la imagen springbootmysql, que está en la network springboot-mysql-net.

Vamos a las networks, refrescamos, y vemos que nuestra red ahora tiene dos containers.

No hay volúmenes nuevos.

Repetimos los pasos de antes para ir a la consola de MySQL y vemos que ahora tenemos la tabla anime creada, pero está vacía.

Pegamos en la consola de MySQL el contenido del archivo anime.sql.

Vemos que ahora se han creado las dos tablas, y que tienen registros.

Si ahora vamos a la URL <http://localhost:8080/all> tenemos la lista de los registros.

## PASO 8 Se pierden los datos

Detener ambos containers. Repetir los pasos 3 y 7 para arrancarlos ambos nuevamente.

Si ahora vamos a la URL <http://localhost:8080/all> veremos que en lugar de la lista de los registros solo obtenemos una lista vacía. La aplicación funciona bien, pero los datos que habíamos cargado en la aplicación se perdieron.

## PASO 9 Correr un container con volumen para MySQL

Detener y borrar ambos containers y el volumen asociado. Correr un container con volumen para MySQL en la network:

docker run -d --name mysqldb -p 3306:3306 --net springboot-mysql-net -e MYSQL\_ROOT\_PASSWORD=1234 -e MYSQL\_DATABASE=anime -e MYSQL\_USER=cruduser -e MYSQL\_PASSWORD=1234 -v db:/var/lib/mysql mysql

Comprobamos que ahora tenemos corriendo el container llamado mysqldb. Vemos que tiene asociado un volumen, aunque no le pusimos nada explícitamente. Y vemos que está asociado a la network correspondiente.

Si vamos comprobar los volúmenes, vemos que ahora tenemos uno asociado con nuetro container.

Si vamos a las networks, comprobamos que tenemos la network que creamos. Si nos dice que no tiene containers asociados, refrescamos la vista, y aparecerá el correspondiente container.

## PASO 10 Verificar la base de datos

Verificar que se ha creado la base de datos. En la terminal de Windows ejecutamos el comando:

docker ps

Nos fijamos en los primeros tres caracteres de la ID de nuestro container. En este caso eran f76:

docker exec -it f76 bash

Aparece una ventana de bash:

mysql -ucruduser -p1234

Estamos en la consola de MySQL:

show databases;

Vemos que tenemos anime más las otras propias de la instalación de MySQL:

use anime;

Estamos en la base de datos:

show tables;

Vemos que la base de datos no tiene tablas. Pegar en la consola de MySQL el contenido del archivo anime.sql.

Para ver cómo quedó:

show tables;

Para salir de la consola de MySQL:

exit

Estamos en bash. Para salir de bash:

exit

Estamos de nuevo en la terminal de Windows.

## PASO 11 Container de Spring Boot

Arrancar el container de Spring Boot en la misma network:

docker run --network springboot-mysql-net --name springboot-container -p 8080:8080 -d springbootmysql

Vemos que apareció un nuevo container llamado springboot-container, instancia de la imagen springbootmysql, que está en la network springboot-mysql-net.

Vamos a las networks, refrescamos, y vemos que nuestra red ahora tiene dos containers.

No hay volúmenes nuevos.

Repetimos los pasos de antes para ir a la consola de MySQL y vemos que ahora tenemos la tabla anime creada, pero está vacía.

Pegamos en la consola de MySQL el contenido del archivo anime.sql.

Vemos que ahora se han creado las dos tablas, y que tienen registros.

Si ahora vamos a la URL <http://localhost:8080/all> tenemos la lista de los registros.

## PASO 12 Se conservan los datos

Detener ambos containers. No borrar el volumen db. Repetir los pasos 3 y 7 para arrancarlos ambos nuevamente.

Si ahora vamos a la URL <http://localhost:8080/all> veremos la lista de los registros. La aplicación funciona bien, y los datos que habíamos cargado en la aplicación se conservaron.

# Usando Docker Compose

Ahora vamos a dockerizar la misma aplicación, pero usando Docker Compose.

## PASO 1 application.properties

Actualizar el archivo application.properties:

spring.datasource.url=jdbc:mysql://mysqldb:3306/anime

spring.datasource.username=cruduser

spring.datasource.password=1234

Notar que el puerto 3306 es el default. Este jar va a intentar conectarse con un servidor de MySQL que esté listening en el puerto 3306. Pero esto va a correr en una network de containers. Por lo tanto, lo que tengo que poner es el puerto interno del container de MySQL, no el puerto local publicado por ese container.

## PASO 2 Imagen de la aplicación Spring Boot

Para no dejar cabos sueltos, vamos a borrar los containers, los volúmenes, las networks y las imágenes. Vamos al directorio target y borramos los jar. Arrancamos de nuevo con todo limpio. Recompilar la aplicación. En la terminal de VS Code:

mvn clean install

Verificamos que tenemos un jar recién creado en target, con el puerto 3307.

## PASO 3 compose.yaml

Muchos tutoriales usan docker-compose.yml, pero la documentación oficial dice que se debe preferir compose.yaml. De modo que esto es lo que yo voy a usar.

Para ver la versión de Docker que tengo instalada, en la consola de Windows:

docker -v

El tutorial usa la propiedad version en la primera línea, pero la documentación oficial[[1]](#footnote-1) dice que está deprecada, de modo que yo no la uso.

Pegar el siguiente texto:

services:

  server:

    build: .

    restart: always

    ports:

      - 8080:8080

    depends\_on:

      - mysqldb

  mysqldb:

    image: mysql

    restart: always

    ports:

      - 3307:3306

    environment:

      MYSQL\_DATABASE: anime

      MYSQL\_USER: cruduser

      MYSQL\_PASSWORD: 1234

      MYSQL\_ROOT\_PASSWORD: 4321

El container en el que va a correr la aplicación Spring Boot se llama server. Podemos poner cualquier nombre, pero esta es una elección razonable, ya que estamos hablando de una API que es como el backend.

La opción build está pidiendo que construyamos la imagen Docker, a partir del jar que acabamos de compilar en el paso anterior. Eso lo haremos siguiendo las instrucciones de Dockerfile.

Después exponemos los ports. Lo que estamos haciendo es mapear el puerto local 8080 al puerto interno del container 8080.

Finalmente, decimos que este containder depends on mysqlbd, que es el servicio de MySQL. Todavía no existe, de modo que el comando docker-compose primero lo levanta, y una vez que está corriendo levanta el container del server.

Notar que el puerto local de MySQL es 3306, o sea el default. Es el que va a usar el jar, porque están en una network.

Ejecutamos el comando:

docker-compose up

Si vamos a la URL <http://localhost:8080/all> veremos que la aplicación está funcionando. Esto es porque el jar usa el puerto 3306 y el container mysqldb usa el puerto interno 3306.

### El puerto publicado por el container

Si intentamos conectarnos a este container mysqkdb que está corriento con una aplicación que esté fuera de la network de Docker, tendremos que usar el puerto 3307, que es el que publica el container. No el interno, 3306, que es el usamos dentro de la network.

1. <https://docs.docker.com/compose/compose-file/#compose-file> [↑](#footnote-ref-1)