Desplegando una aplicación spring mysql en docker remoto

Nota:
Diremos: **en local** cuando queremos hacer referencia a la máquina donde hemos desarrollado nuestra aplicación y tenemos instalada nuestra base de datos

Diremos: **en remoto** cuando queramos hacer referencia a la máquina donde vamos a hacer el despliegue de nuestra aplicación

Nos ubicamos en local:
En local, Tomamos la base de datos que queremos y la guardamos en un sql:

mysqldump seguimientomonedas -u root -p > seguimientomonedas.sql

para copiar desde local a remoto el sql (se presupone un usuario ubuntu en remoto):

scp seguimientomonedas.sql ubuntu@ipremota:/home/ubuntu

Nos ubicamos en remoto:
Vamos a tomar una imagen docker de mysql. Para ello es posible que tengamos que instalar docker:

sudo apt update
sudo apt install docker.io

Ahora agregamos nuestro usuario al grupo docker para que pueda usarlo sin sudo
sudo usermod -aG docker ubuntu

es posible que tengamos que reiniciar (o cerrar la sesión y volver a iniciarla) para conseguir los permisos en el usuario ubuntu

Queremos que cuando se reinicie el docker de la base de datos persistan los cambios. Para ello creamos un volumen de almacenamiento al que pueda conectarse nuestro docker. Le buscamos un nombre descriptivo. Se propone: mysql-db-data

docker volume create mysql-db-data

De cara al exterior, únicamente queremos que esté disponible nuestra api, no la base de datos. Vamos a crear una red en docker para que se puedan comunicar internamente nuestros servicios (bases de datos, api, etc) pero no sea accesible desde el exterior

docker network create redmonedas

Descargamos una imagen docker desde repositorio de mysql:

```
docker pull mysql/mysql-server:latest
```

Creamos el contenedor, por ejemplo con el nombre: mysqlcontainer y le decimos que queremos que se active como un servicio (un demonio de linux: opción -d) en la red: redmonedas queremos que nos persista en un volumen llamado mysql-db-data la información almacenada de la base de datos (carpeta /var/lib/mysql) y finalmente le decimos el nombre de la imagen que hemos descargado: mysql/mysql-server:latest

```
docker container run -d --network redmonedas --name mysqlcontainer -v mysql-db-data:/var/lib/mysql mysql/mysql-server:latest
```

Necesitamos conocer la contraseña de mysql que se ha generado al crear el contenedor. Ejecutamos:
docker logs mysqlcontainer

y buscamos la línea que diga:
GENERATED ROOT PASSWORD

necesitamos esa contraseña para acceder. La copiamos.

Para acceder a nuestro mysql y configurarlo necesitamos una shell interactiva.

Para entrar con una shell interactiva en el contenedor que hemos creado:
docker exec -it mysqlcontainer bash

docker exec es la forma de ejecutar un comando en un contenedor. observar que le decimos interactivo: -it y el comando que le pedimos es que nos genere una shell bash

una vez dentro accedemos a mysql con la pass que hemos copiado:

mysql -u root -p

Nos pedirá que cambiemos la contraseña. En el siguiente ejemplo se pone la clave: 1q2w3e4r
ALTER USER 'root'@'localhost' IDENTIFIED BY '1q2w3e4r';

Por defecto mysql no permite acceso remoto con usuario root. Como estamos usando servicios docker nos puede impedir el acceso. Se recomienda crear otro usuario:
Si queremos crear ahora otro usuario admistrador con nombre admin:
CREATE USER 'admin'@'localhost' IDENTIFIED BY '1q2w3e4r';
GRANT ALL PRIVILEGES ON *.* TO 'admin'@'localhost' WITH GRANT OPTION;
CREATE USER 'admin'@'%' IDENTIFIED BY '1q2w3e4r';
GRANT ALL PRIVILEGES ON *.* TO 'admin'@'%' WITH GRANT OPTION;
CREATE DATABASE seguimientomonedas;

Después exit del container

Si la password creada es: 1q2w3e4r podemos copiar la base de datos así:
docker exec -i mysqlcontainer mysql -uroot seguimientomonedas -p1q2w3e4r < seguimientomonedas.sql

Con lo anterior ya tenemos la base de datos en funcionamiento
Ahora vamos a crear nuestro propio docker de spring

Nos ubicamos en local:

En application.properties de nuestro proyecto spring cambiamos la ruta de localhost a nuestro contenedor docker:

```
spring.datasource.url=jdbc:mysql://mysqlcontainer:3306/seguimientomonedas?allowPublicKeyRetrieval=true&useSSL=false&serverTimezone=UTC
```

Recordar también poner nuestro usuario creado y la password en el mysqlcontainer

En nuestro proyecto spring debemos ejecutar el comando maven: clean package
Una opción es desde eclipse sobre el proyecto → botón derecho → run as → maven build → Goals: clean package

y pulsamos en el botón: Run

Podemos comprobar que generó correctamente el .jar de la siguiente forma:
botón derecho sobre el proyecto → show in → terminal

y ejecutamos (atención a las rutas. Por ejemplo, se supone que se ha descargado el jdk17 en el home de dam2)
/home/dam2/jdk-17/bin/java -jar target/Monedas-0.0.1-SNAPSHOT.jar

si arranca correctamente vamos bien

Construimos ahora en la ruta que estamos nuestro: Dockerfile que contendrá:

```
FROM openjdk:17-alpine
ADD target/Monedas-0.0.1-SNAPSHOT.jar /usr/share/app.jar
ENTRYPOINT ["/opt/ojenjdk-17/bin/java", "-jar", "/usr/share/app.jar"]
```

Observar que agregamos nuestra app compilada .jar también tomamos una imagen ya creada del jdk17 y copiamos nuestro .jar en *usr*share/app.jar

finalmente le decimos que ejecute: java -jar /usr/share/app.jar Así que nos estamos haciendo una imagen docker con el jdk17, nuestro .jar y le decimos que lo ejecute. Así de simple

Generamos la imagen docker mediante:
docker build -t monedas .

Esto es: la imagen se llamará monedas y toma la información contextual de donde estamos (y por tanto el fichero Dockerfile que creamos antes)

Ahora exportamos la imagen a un fichero tar:
docker save monedas -o monedas.tar

copiamos el fichero tar a remoto: (reemplazar ipremota por la que corresponda)
 scp monedas.tar ubuntu@ipremota:/home/ubuntu

Nos ubicamos en remoto:

docker load -i monedas.tar

docker container run --network redmonedas --name monedascontainer -p 8080:8080 -d monedas:latest

También se puede usar docker-compose:

sudo apt install docker-compose

Generamos el fichero de configuración: docker-compose.yml

fichero: docker-compose.yml

```
version: "2"
services:
  monedascontainer:
    image: monedas
    ports:
      - "8080:8080"
    networks:
      - redmonedas
    depends_on:
      - mysqlcontainer

  mysqlcontainer:
    image: mysql/mysql-server
    networks:
      - redmonedas
    volumes:
      - mysql-db-data:/var/lib/mysql

networks:
  redmonedas:

volumes:
  mysql-db-data:
```

Para arrancar el servicio:
docker-compose up -d

y para pararlo:
docker-compose stop