Deploy Spring Boot MySQL Application to Docker

1. Deploy MySQL Image in a Docker Container

// ----------

Step1: Pull MySQL Image

Here is the docker command to pull the latest MySQL docker image:

=> docker pull mysql

// ----------

Step 2: Create a docker network to communicate Spring boot application and MySQL database

Here is the docker command to create a new network:

=> docker network create springboot-mysql-net

Here springboot-mysql-net is the network name.

Use the below command to list the networks:

=> docker network ls

// ---

Step 3: Run MySQL image in a docker container in the same network (it means on springboot-mysql-net network which is above created)

Here is the docker command to run MySQL image in a container in the same network:

=> docker run --name mysqldb --network springboot-mysql-net -e MYSQL_ROOT_PASSWORD=root -e MYSQL_DATABASE=employeedb -d mysql

// just change MYSQL_DATABASE name if you want (so on running mysql image converting mysql container having name mysqldb)

// ---

// not important (optional)

Step 4: Access the MySQL database in a container

Here is the command to access the MySQL database in a container:

=> docker exec -it mysqldb bash

=> mysql -u root -p

=> root

=> show databases;

That's it. Once the MySQL image is deployed in a docker container. Next, we will deploy the Spring boot application in a docker container.

// ----------

2. Deploy Spring Boot Application in a Docker Container

// step 1.

spring.application.name=student

spring.datasource.url=jdbc:mysql://localhost:3306/test

spring.datasource.username=root

spring.datasource.password=Piyush@l1

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.generate-ddl=true

=> change to docker

spring.application.name=student

spring.profiles.active=docker

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

```
spring.datasource.username=root

spring.datasource.password=root

spring.jpa.hibernate.ddl-auto=update


// optional

spring.jpa.show-sql=true

spring.jpa.generate-ddl=true
```

and because we are changing url from localhost to mysqldb (container name) so we have to add dependency in pom.xml

```
    <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <configuration>
            <skipTests>true</skipTests>
        </configuration>
    </plugin>
```

after that do Maven install => it will create [name].jar file under target file

// step 2.----------

create Dockerfile seperately just below Target file

Type below text in Dockerfile as text

=> student-0.0.1-SNAPSHOT.jar recently created jar file

=> student-doc.jar this is the custom name of created jar file

```
FROM openjdk:17
ADD target/student-0.0.1-SNAPSHOT.jar student-doc.jar
ENTRYPOINT ["java", "-jar", "/student-doc.jar"]
```

// ---

now come in command prompt or power shell

now find the springboot project path where docker and .jar inside target is created . like

"C:\Users\piyush_kumar\Downloads\student (2)\student\"

PS C:\Users\piyush_kumar> cd "C:\Users\piyush_kumar\Downloads\student (2)\student\"

now after that

// PS C:\Users\piyush_kumar\Downloads\student (2)\student> docker build -t [image_name] .

PS C:\Users\piyush_kumar\Downloads\student (2)\student> docker build -t student-doc .

this will create image of our spring boot project

// for seeing the created image

docker images

// step 3. ----------

Run a docker image in a docker container in the same network

Once you have a docker image, you can run it using the docker run command like so:

// connect this springboot image with same network (springboot-mysql-net) on which mysql image is already running

// now when we run springboot image whose name is student-doc then docker will create container of that whose name is springboot-mysql-container

=> docker run --network springboot-mysql-net --name springboot-mysql-container -p 8080:8080 student-doc

//
==========
==========

Test CRUD RESTful WebServices using Postman Client

Create User REST API:

Request URL: http://localhost:8080/api/...

---

now we have to upload our springboot image on docker hub so that in future we can pull and use it with mysql image on a common network

// step 1. PS C:\Users\piyush_kumar> docker login

it will give

// Authenticating with existing credentials...

// Login Succeeded

// so to push on docker hub we have to create image with name piyush12singh/student-doc (which is earlier only student-doc) because username of docker hub is piyush12singh so overall image is piyush12singh/image_name

// step 2. PS C:\Users\piyush_kumar> cd "C:\Users\piyush_kumar\Downloads\student (2)\student\"

//step3 PS C:\Users\piyush_kumar\Downloads\student (2)\student> docker build -t piyush12singh/student-doc .

// step 4. PS C:\Users\piyush_kumar> docker push piyush12singh/student-doc:latest

// step 5. PS C:\Users\piyush_kumar> docker pull piyush12singh/student-doc:latest

// step 6. PS C:\Users\piyush_kumar> docker run --network springboot-mysql-net12 --name springboot-mysql-container -p 8080:8080 piyush12singh/student-doc

// PS C:\Users\piyush_kumar> docker run --name mysqldb --network springboot-mysql-net12 -e MYSQL_ROOT_PASSWORD=root -e MYSQL_DATABASE=test -d mysql