In this article, we are going to discuss two ways of dockerizing a Spring Boot app with MySQL database. Those are;

1. By creating a docker network
2. Using docker-compose

Here is the application we intend to deploy to docker.

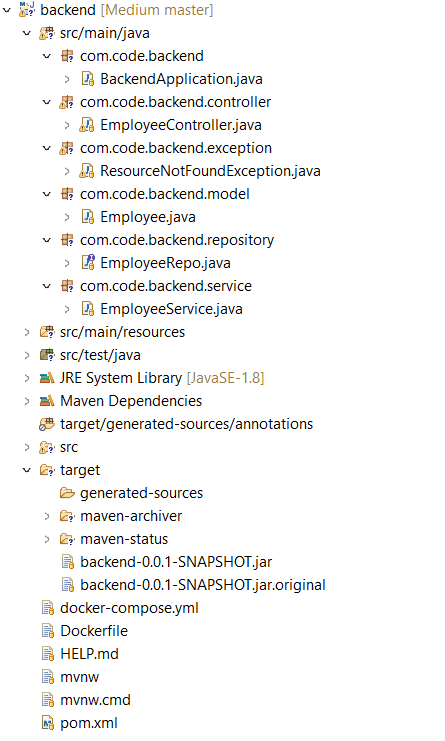
**[GitHub - Kanchana46/spring-boot-backend](https://github.com/Kanchana46/spring-boot-backend.git" \t "_blank)**

[You can't perform that action at this time. You signed in with another tab or window. You signed out in another tab or…](https://github.com/Kanchana46/spring-boot-backend.git" \t "_blank)

[github.com](https://github.com/Kanchana46/spring-boot-backend.git" \t "_blank)

This application contains CRUD operations for an employee-management system such as adding employees, getting employees, updating employees, and so on.

**Project Structure**



Project structure

One of the important files is **application.properties** which we provide database configurations.

**application.properties**

And we need to have a jar of our spring boot app. To build a jar file do the following.

***Right-click on pom.xml →Run As → Maven build***

It will create the jar ***backend-0.0.1-SNAPSHOT.jar*** file inside the ***target***folder, as we can see in the project structure.

We need to create the Dockerfile which includes the instruction to building the image of the spring-boot app as well.

Now we are good to go for dockerizing the application.

**Deploying the application to Docker**

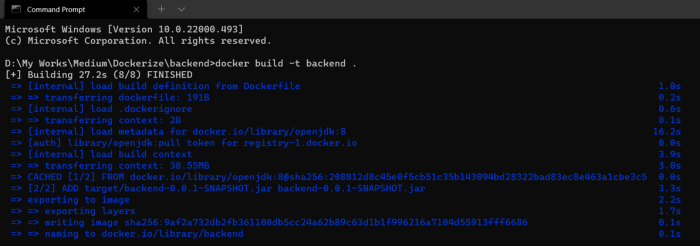
* **Using a Docker network**

1. We need to pull the MySQL image from the docker hub with the following command. Here we will use version 5.7.

**docker pull mysql:5.7**

2. Create a docker image using the spring-boot app. Open the command terminal inside the project folder and do the following command.

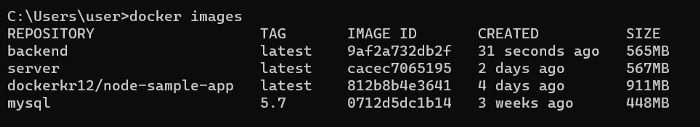
**docker build -t backend .**



Docker build

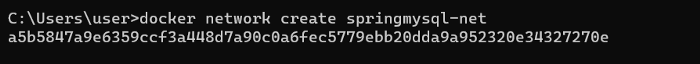
We can verify whether images are created correctly using the following command.

**docker images**



Images

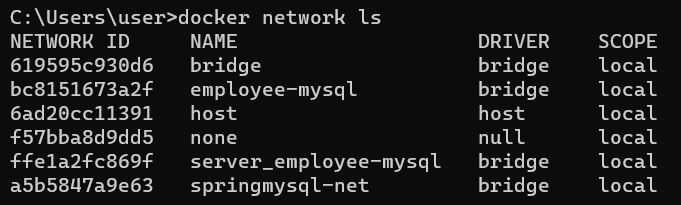
3. We will have a docker network that contains both Spring-boot and MySQL containers and these containers will communicate with each other. So the first thing we need to do is to create a docker network. Let’s create a docker network as **springmysql-net.**



Network

And also we can verify it with the following command.

**docker network ls**



Networks

4. Run the MySQL container in the network using the following.

https://miro.medium.com/max/700/1*XP87KcaYIF-HaXI1PnBqqQ.png

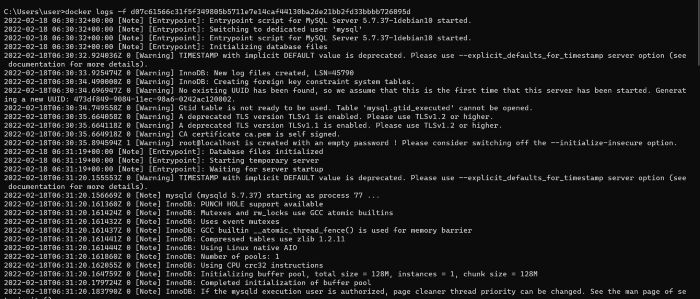
Run MySQL container

**docker run --name mysqldb --network springmysql-net -e MYSQL\_ROOT\_PASSWORD=1234 -e MYSQL\_DATABASE=employeedb -e MYSQL\_USER=sa -e MYSQL\_PASSWORD=1234 -d mysql:5.7**

* Container name is mysqldb. Note that we have given this in the connection URL in the application.properties file.
* Network is springmysql-net.
* -e stands for environment variables. These values also have mentioned in the application.properties file.
* -d instructs to run in detached mode.

To verify everything went fine, we can see the logs with the command below.

**docker logs -f <container\_name>**

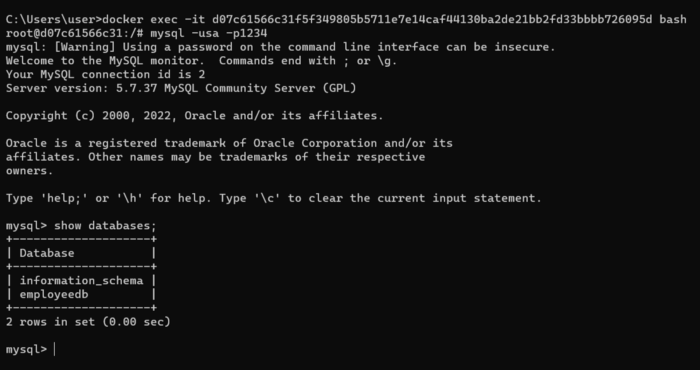


MySQL logs

If needed we can check if the database **employeedb**has been created correctly.

We can do the following commands.

**- docker exec -it <container\_id> bash  
- mysql -u<username> -p<password>  
- show databases;**



If the database has been created successfully, we can see it like above.

5. Run the Spring-boot container in the same network with this command.

**docker run --network springmysql-net --name backend-container -p 8080:8080 -d backend**

https://miro.medium.com/max/700/1*5JEuOMYAauhRJ9cMGWp1ag.png

Run Spring-boot container

You can if these containers are running correctly using the command below.

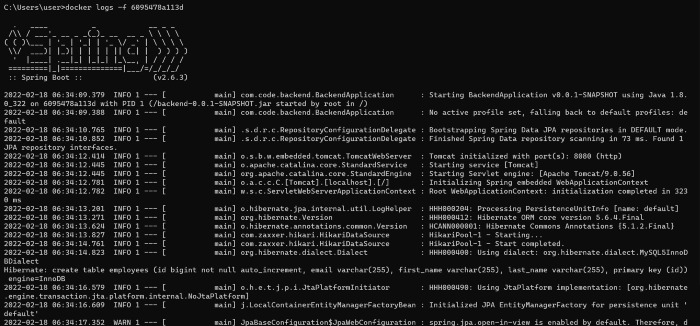
**docker ps**

https://miro.medium.com/max/700/1*FK2WabBoXD4hLev1gcyVSQ.png

Running containers

6. Let’s see the logs of the Spring-boot container to confirm everything is fine.

**docker logs -f <container\_id>**



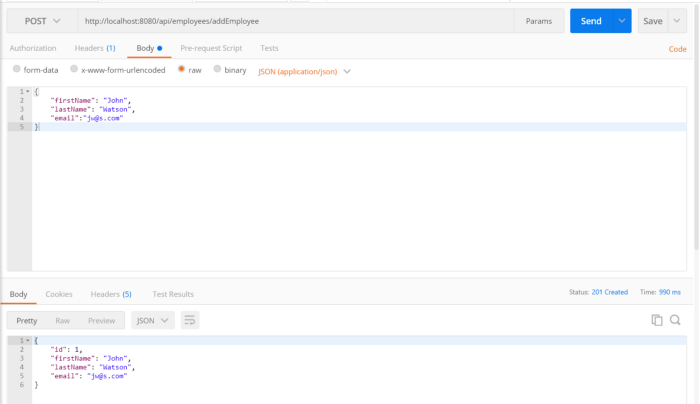
Spring boot app logs

That’s all we need to do!.

Let’s check if our app is working as expected by doing a POST and GET request.

**POST:**

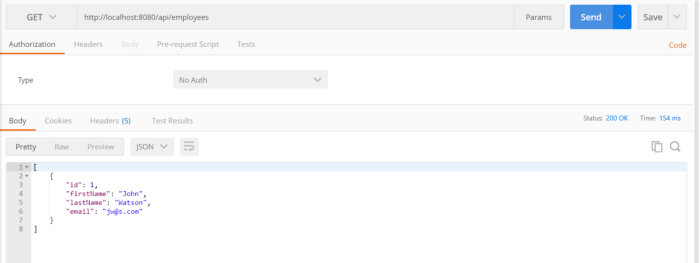
Endpoint →[***http://localhost:8080/api/employees/addEmployee***](http://localhost:8080/api/addEmployee)



Add employee

**GET:**

Endpoint →[***http://localhost:8080/api/employees***](http://localhost:8080/api/addEmployee)



Get employees

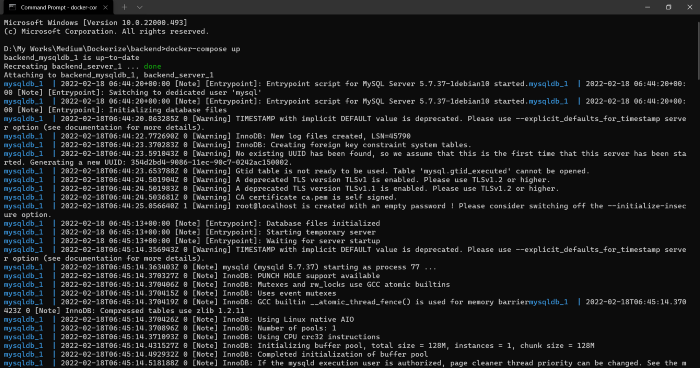
* **Using docker-compose**

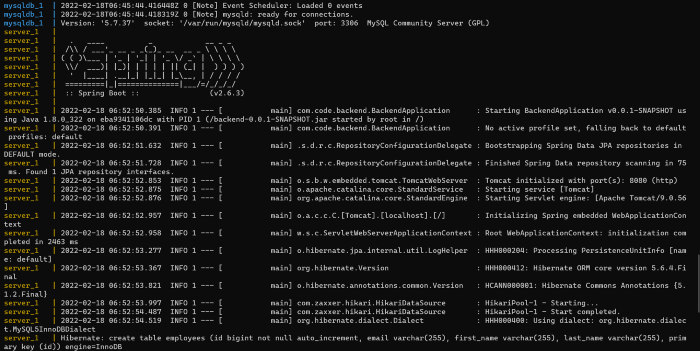
Without going through these many steps we can do the same thing with one command docker-compose.

To do that, we need to create docker-compose.yml file which includes the following.

Then Open the command terminal inside the project folder and do the following command.

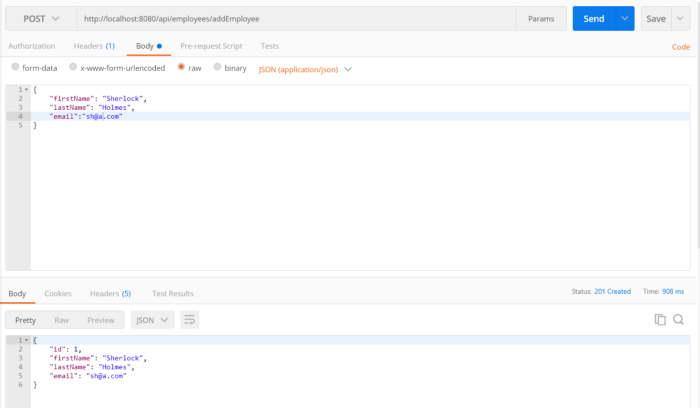
**docker-compose up**





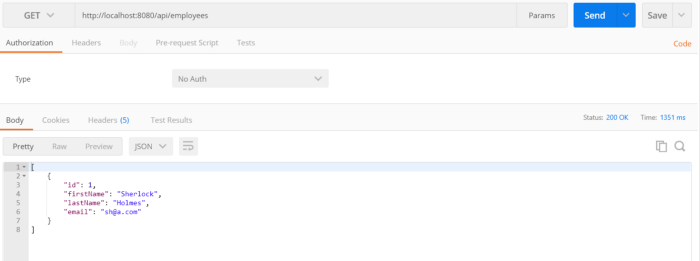
And let’s verify this one also by creating POST and GET requests.

**POST:**



Add employee

**GET:**



Get employees

So these are the methods you can use to dockerize a Spring boot application with MySQL. Even if you use different programming languages, databases the concept remains the same.