

Task:

Get the application Up and running in the docker compose .

Soln:

Docker Full Stack application

Frontend

1. Create a react app and install axios in it
2. Create docker file for react app

```
1 FROM node:latest
2 WORKDIR /app
3 COPY . .
4 RUN npm install
5 EXPOSE 3000
6 CMD [ "npm", "start" ]
7
```

- 3.
4. Create dockerignore to ignore the node_modules and .gitignore

```
Dockerfile docker-frontent 1,U .dockerignore U X
docker-frontent > .dockerignore
1 node_modules
```

- 5.

Backend

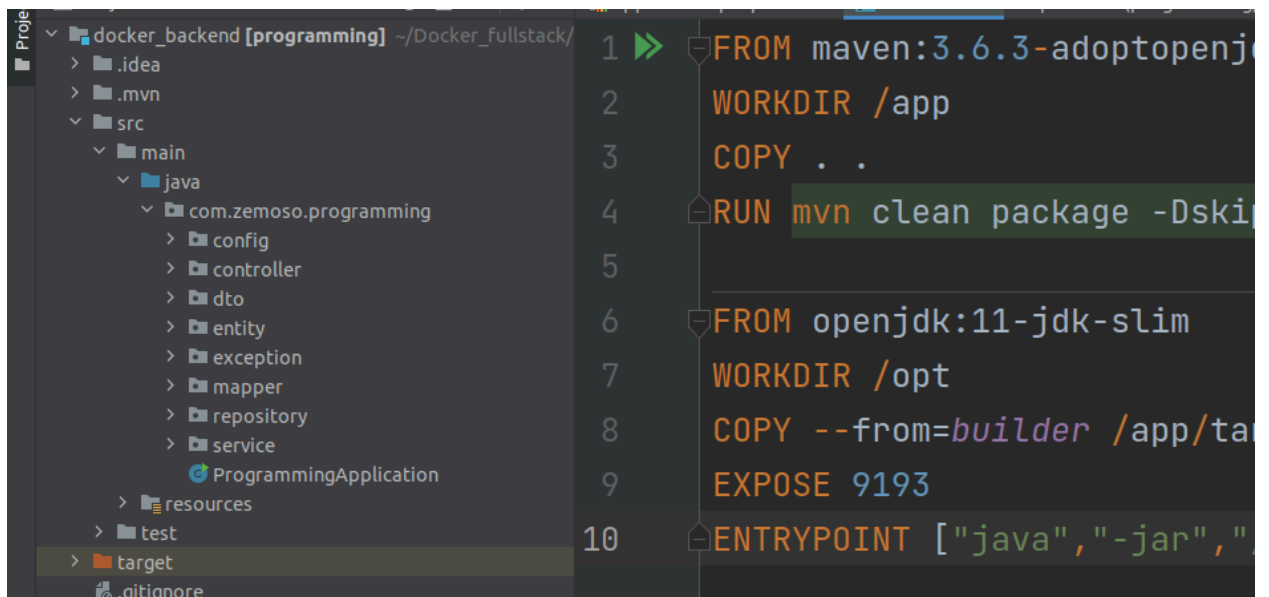
1. Create a spring boot project

2. Create dockerfile for backend

```
FROM maven:3.6.3-adoptopenjdk-11 AS builder
WORKDIR /app
COPY . .
RUN mvn clean package -DskipTests

FROM openjdk:11-jdk-slim
WORKDIR /opt
COPY --from=builder /app/target/*.jar /opt/app.jar
EXPOSE 9193
ENTRYPOINT ["java", "-jar", "/opt/app.jar"]
```

- 3.
4. We have to create multistage docker file because first stage we have to download dependencies and create jar file and second step we need to run the jar file. It will also reduce the image size as only the jar file will be present on image (no dependencies , build tools etc)
5. Create at one controller endpoint to see the api result



The screenshot shows an IDE with a project structure on the left and a Dockerfile on the right. The project structure is for a Spring Boot application named 'docker_backend [programming]'. It includes a 'src' directory with 'main' and 'test' subdirectories. The 'main' directory contains a 'java' package with a 'com.zemoso.programming' package, which includes 'config', 'controller', 'dto', 'entity', 'exception', 'mapper', 'repository', and 'service' sub-packages, and a 'ProgrammingApplication' class. There are also 'resources' and 'target' directories. The Dockerfile on the right is a multistage Dockerfile with 10 lines, matching the code in the first image. It uses 'maven:3.6.3-adoptopenjdk-11' as the builder stage and 'openjdk:11-jdk-slim' as the final stage. The 'ENTRYPOINT' is set to run the jar file.

- 6.

Database:

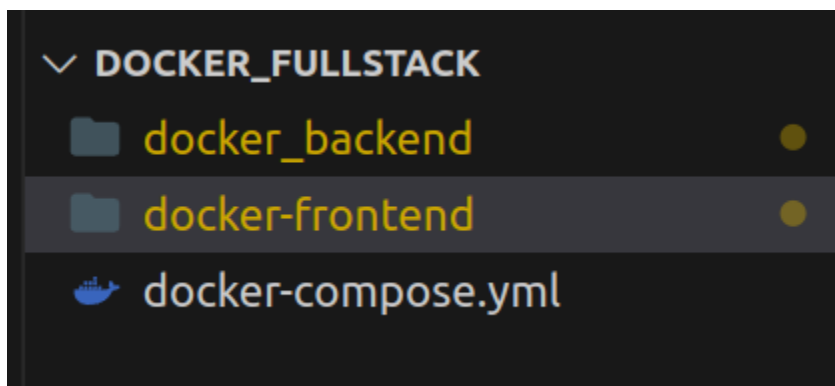
1. Create normal data base in root connection
2. Write the credential in src/java/resources in spring

```
application.properties x Dockerfile x m pom.xml (programming) x
1 server.port=9193
2 spring.datasource.url=jdbc:mysql://database:3306/todo
3 spring.datasource.username=root
4 spring.datasource.password=Test@1234
5
6 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
7 #spring.jpa.hibernate.ddl-auto=update
8 spring.jpa.hibernate.ddl-auto=create
9
```

- 3.
4. Here localhost is replaced by database because database is a name of service which will run prior to backend and backend is dependent on it

Docker Compose File Creation


1. Create the docker compose file outside the frontend/backend folder




- 2.

3.

```
docker-compose.yml
1  version: '3'
2  services:
3    database:
4      image: mysql:latest
5      environment:
6        - MYSQL_ROOT_PASSWORD=Test@1234
7        - MYSQL_DATABASE=todo
8      ports:
9        - 3307:3306
10     volumes:
11       - todo-list-mysql-data:/var/lib/mysql
12     networks:
13       - main_network
14
15     backend:
16       build:
17         dockerfile: Dockerfile
18         context: ./docker_backend
19       depends_on:
20         - database
21       restart: always
22       environment:
23         - SPRING_DATASOURCE_URL=jdbc:mysql://database:3306/todo
24         - SPRING_DATASOURCE_USERNAME=root
```

4.  docker-compose.yml

```
22     environment:
23       - SPRING_DATASOURCE_URL=jdbc:mysql://database:3306/todo
24       - SPRING_DATASOURCE_USERNAME=root
25       - SPRING_DATASOURCE_PASSWORD=Test@1234
26     ports:
27       - 9193:9193
28     volumes:
29       - backend_volume:/opt
30     networks:
31       - main_network
32 frontend:
33   build:
34     dockerfile: Dockerfile
35     context: ./docker-frontend
36   depends_on:
37     - backend
38   ports:
39     - 3000:3000
40   networks:
41     - main_network
42   volumes:
43     - ./docker-frontend:/app
44 volumes:
45   todo-list-mysql-data:
```

5.  docker-compose.yml

```
volumes:
  todo-list-mysql-data:
  backend_volume:
networks:
  main_network:
```

First we have to start the database service with password and database name

Then start the backend service which depends on database and place the env variables defined on database (spring boot)

6. Then we have to start the frontend end service which depends on backend