Atypon

Docker Assignment report

By Rayyan Al-Hourani

First will start explaining about my projects then I will explain about docker and docker compose.

1. Enter Data App:

This application allows users to enter student names and their marks. Users need to log in before entering marks.

controllers:

A. Authentication Controller (Auth Controller):
   - GET method: Displays the login page.
   - POST method: Sends user credentials to the authentication server. If valid, the server stores the username in the session and redirects to the home page. If invalid, the user is redirected to the login page.

B. Data Controller:
   - GET method: Shows the home page.
   - POST method: Stores student names and marks in MySQL database.

Models:

- User class: Contains username and password.
- Student class: Contains student name and mark.

Services:

- StudentRepository and StudentService: Help store data in the MySQL database.

2. Analytics Service:

This application retrieves student marks from the MySQL database and performs various analytics.

Controllers:

A. Data Controller:

- GET method: Retrieves student marks, performs analytics using the Analytics class, and stores or updates the analytics in MongoDB.

Classes:

- Student class: Contains student name and mark.
- Analytics class: Calculates number of students, min and max marks, average, and median of marks.

Services:

- StudentRepository and StudentService: Retrieve student marks from MySQL database.
- AnalyticsRepository and AnalyticsService: Store or update analytics in MongoDB.

3. Show Results App:

This application displays analytics from MongoDB on the home page. Users log in similarly to the Enter Data App.

Controllers:

A. Authentication Controller (Auth Controller):

- GET method: Displays the login page.
- POST method: Sends user credentials to the authentication server to validate.

B. Data Controller:

- GET method: Sends a request to the Analytics Service to update data in MongoDB and then retrieves the updated analytics data.
- The updated analytics data is then displayed on the home page.

Models:

- User class: Contains username and password.
- Analytics class: Contains analytics data.

Services:

- AnalyticsService and AnalyticsRepository: Update analytics data in MongoDB and retrieve analytics data from MongoDB.

Note: The Data Controller now has a single GET method that first sends a request to the Analytics Service to update the data in MongoDB. After that, it retrieves the updated analytics data and displays it on the home page.

4. Authentication Service:

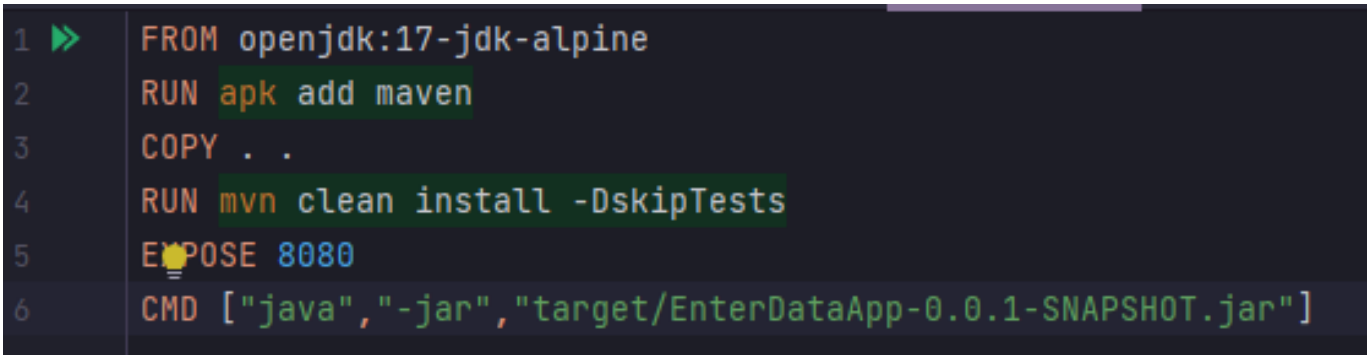
- A simple service with a single GET method that checks if a user is valid.

5. Databases:

- Two containers are used: one for MySQL server and another for MongoDB.

Docker and docker compose

So first for each project I create a docker file that contains the following:



```
1  FROM openjdk:17-jdk-alpine
2  RUN apk add maven
3  COPY . .
4  RUN mvn clean install -DskipTests
5  EXPOSE 8080
6  CMD ["java","-jar","target/EnterDataApp-0.0.1-SNAPSHOT.jar"]
```

Line 1 : import a docker image that contains alpine linux and OpenJDK to run java code inside the container.

Line 2: install maven inside the container.

Line 3: copy the files from the directory to the container.

line 4: This command runs Maven inside the container. It cleans the project, compiles and packages it.

line 5: the container listen on port 8080.

Line 6: this line used to run the jar file.

And this is the tamplete that we user for all docker files we just change the port and the name of jar file.

Docker compose

1-databases

To run the database containers we have to set the configuration for each database on the docker compose like that:

A. **mysqldb**:

1. Set the container name.
2. Specify the Docker image for the MySQL server.
3. Define the port on which the container will listen.
4. Set the database name, username, and password for accessing the database.
5. Configure the network through which the containers will communicate with each other.

B. **mongodb**:

1. Set the container name.
2. Choose the Docker image for the MongoDB database.
3. Define the database name.
4. Configure the network to facilitate communication between containers.

```yaml
services:
  mysqldb:
    container_name: mysqldb
    image: mysql:latest
    ports:
      - "3307:3306"
    environment:
      MYSQL_USER: admin
      MYSQL_PASSWORD: 123456
      MYSQL_ROOT_PASSWORD: 123456
      MYSQL_DATABASE: mydb
    networks:
      - net
```

```yaml
  mongodb:
    container_name: mongodb
    image: mongo:latest
    environment:
      MONGO_INITDB_DATABASE: Analytics
    ports:
      - "27017"
    networks:
      - net
```

to access the databases inside the applications we have to write that inside the application.proprieties

```
server.port=8081

spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://mysqldb:3306/mydb
spring.datasource.username=admin
spring.datasource.password=123456
spring.datasource.driver-class-name =com.mysql.cj.jdbc.Driver
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

spring.data.mongodb.host=mongodb
spring.data.mongodb.port=27017
spring.data.mongodb.database=Analytics
```

and now for each application we have to configure each microservice inside the docker compose like that :

1. Set the service name.
2. Set the container name.
3. In the build section, specify the microservice directory and the Dockerfile name.
4. Set the microservice's port.
5. Define the microservices that the current microservice depends on.
6. Set the network that the containers will connect to each other on.

- The entry data app runs on port 8080 and depends on authentication service and mysql data base,

```
dataentry:
  container_name: dataentry
  build:
    context: ./EnterDataApp
    dockerfile: Dockerfile
  ports:
    - "8080:8080"
  networks:
    - net
  depends_on:
    - authservice
    - mysqldb
```

- The analytics service runs on port 8081 and depends on Mysql and mongodb containers.

- The show data app runs on port 8082 and depends on mongodb and authentication service.

- The authentication service runs on port 8083 and its not depend on any container.