**Module 8: Assignment**

**Docker with microservices using Springboot -II**

## **Step 1: Build the JAR File**

1. Navigate to the project directory:

|  |
| --- |
| cd /path/to/studentms |

1. Build the Spring Boot JAR file using Maven:

|  |
| --- |
| mvn clean package |

* + The generated JAR file will be in the target/ directory.

## **Step 2: Create a Dockerfile**

Create a file named Dockerfile in the root of your project and add the following content:

|  |
| --- |
| # Use an official OpenJDK runtime as base image  FROM openjdk:17-jdk-slim  # Set the working directory inside the container  WORKDIR /app  # Copy the JAR file into the container  COPY target/studentms-0.0.1-SNAPSHOT.jar app.jar  # Expose the application port  EXPOSE 8080  # Run the application  ENTRYPOINT ["java", "-jar", "app.jar"] |

Modify the COPY command to match the actual JAR filename generated in Step 1.

## **Step 3: Build the Docker Image**

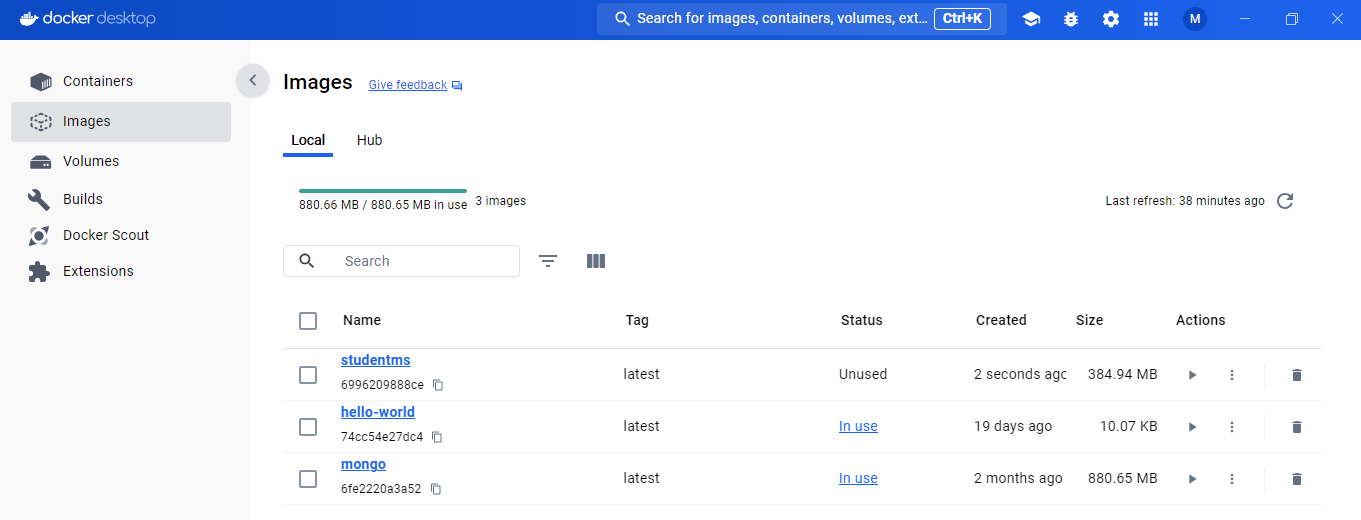
Run the following command to build the Docker image:

|  |
| --- |
| docker image build -t studentms . |

* -t studentms assigns a name (studentms) to the Docker image.
* The . indicates the current directory contains the Dockerfile.

Verify the image was created:

|  |
| --- |
| docker images |



## **Step 4: Run the Docker Container**

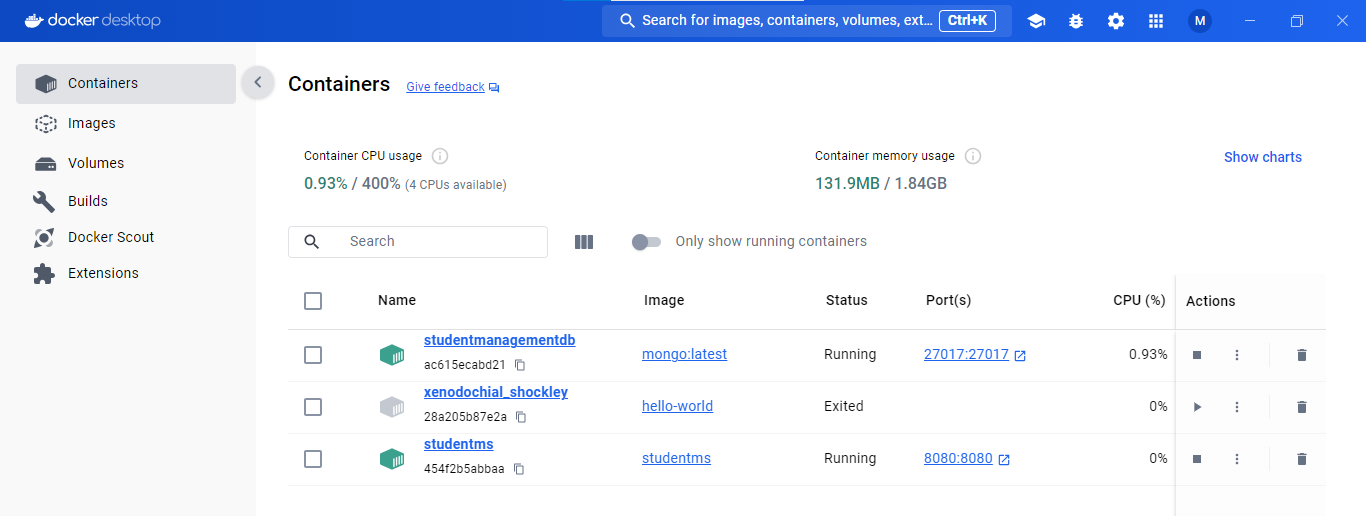
Run the following command to start the container:

|  |
| --- |
| docker run -p 8080:8080 --name studentms --link studentmanagementdb:mongo -d studentms |

Verify that the container is running:

|  |
| --- |
| docker ps |

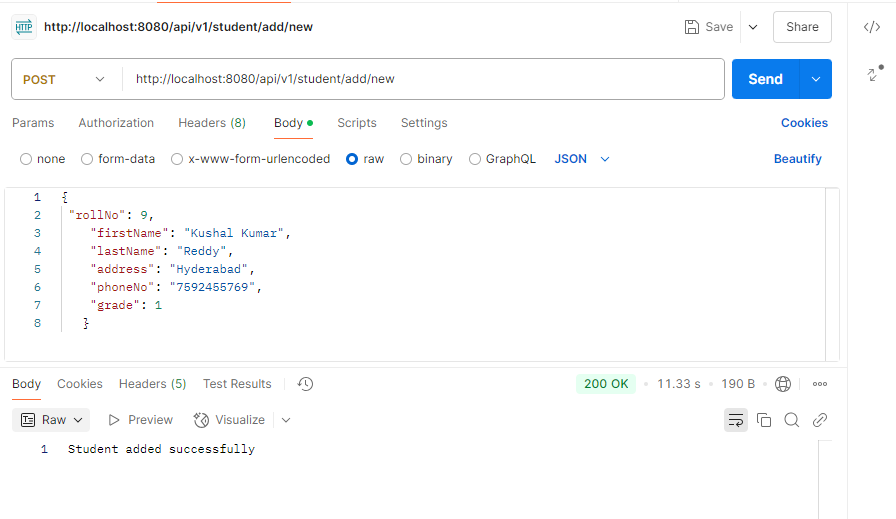
## 



## **Step 5: Test the API**

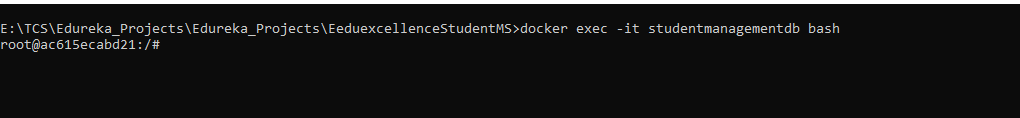
Check if the application is running by executing:

|  |
| --- |
| curl <http://localhost:8080/> |

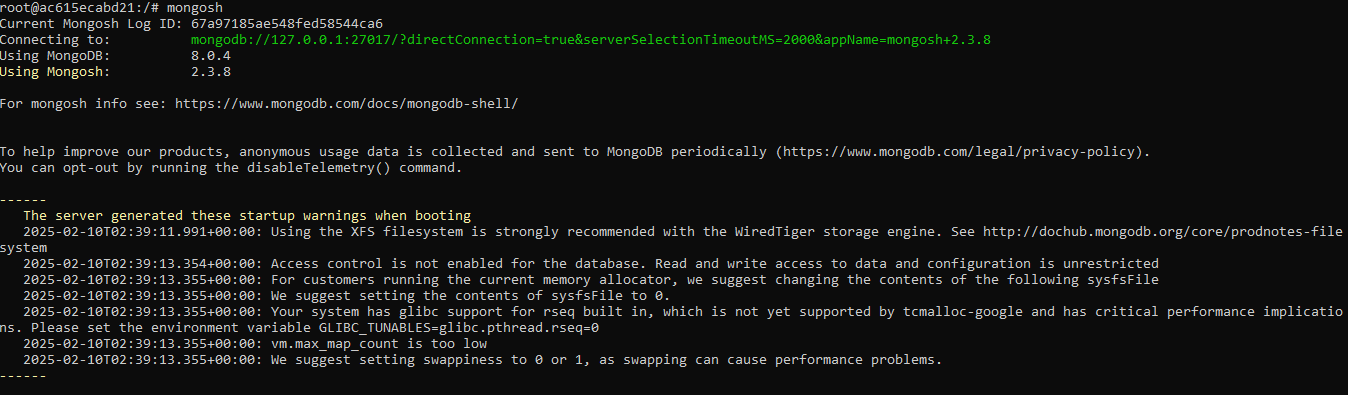


Check the collection in Mongo db

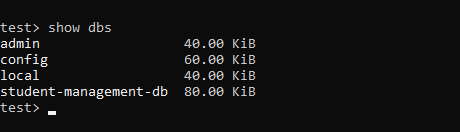
docker exec -it studentmanagementdb bash



mongosh



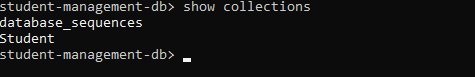
show dbs



use student-management-db



show collections



db.student.find()