SJSU SAN JOSÉ STATE UNIVERSITY

Assignment 1

Hello World Microservices Application

Name: Yash Bharatbhai Savani

Student ID: 017581122

Subject Code: CMPE272

Subject Name: Enterprise Software Platforms

Guidance by: Prof. Rakesh Ranjan

Objective: The goal of this assignment is to get you familiar with the basic concepts of microservices architecture. You will create a simple "Hello World" microservices application using Spring Boot (Java) or Express.js (Node.js), Docker, and Kubernetes.

Tasks:

1) Setting Up the Development Environment
2) Creating the Microservices
3) Containerizing the Microservices with Docker
4) Deploying the Application on Kubernetes
5) Testing and Integration
6) Documentation

Date: 09/04/2024

- ## Setting Up the Development Environment

Programming language: Java

- To setup java version, download and install latest version of JDK in your system.
- Set JDK bin path in environment variable.
- After that check its completely installed or not, for that write “java –version” command in terminal and check the version of it.

Framework: Spring Boot

- Go to https://start.spring.io/ website to initialize spring project
- Select
    - Project-Maven
    - Spring Boot-3.3.3
    - Project Metadata
        - Group:
        - Artifact:
        - Name:
        - Packaging:
        - Java:
    - Dependencies:
- Generate the file

IDE: IntelliJ IDEA CE

- Download IntelliJ IDEA CE from its official website.
- Install it and select default language as Java.
- Open project which you generate through Spring initializer.
- Develop microservices in it.

Containerization: Docker

- Download Docker from its official website.
- Create Docker Hub account online.
- Login to that account in local machine.

- **Creating the Microservices**

1) Hello service

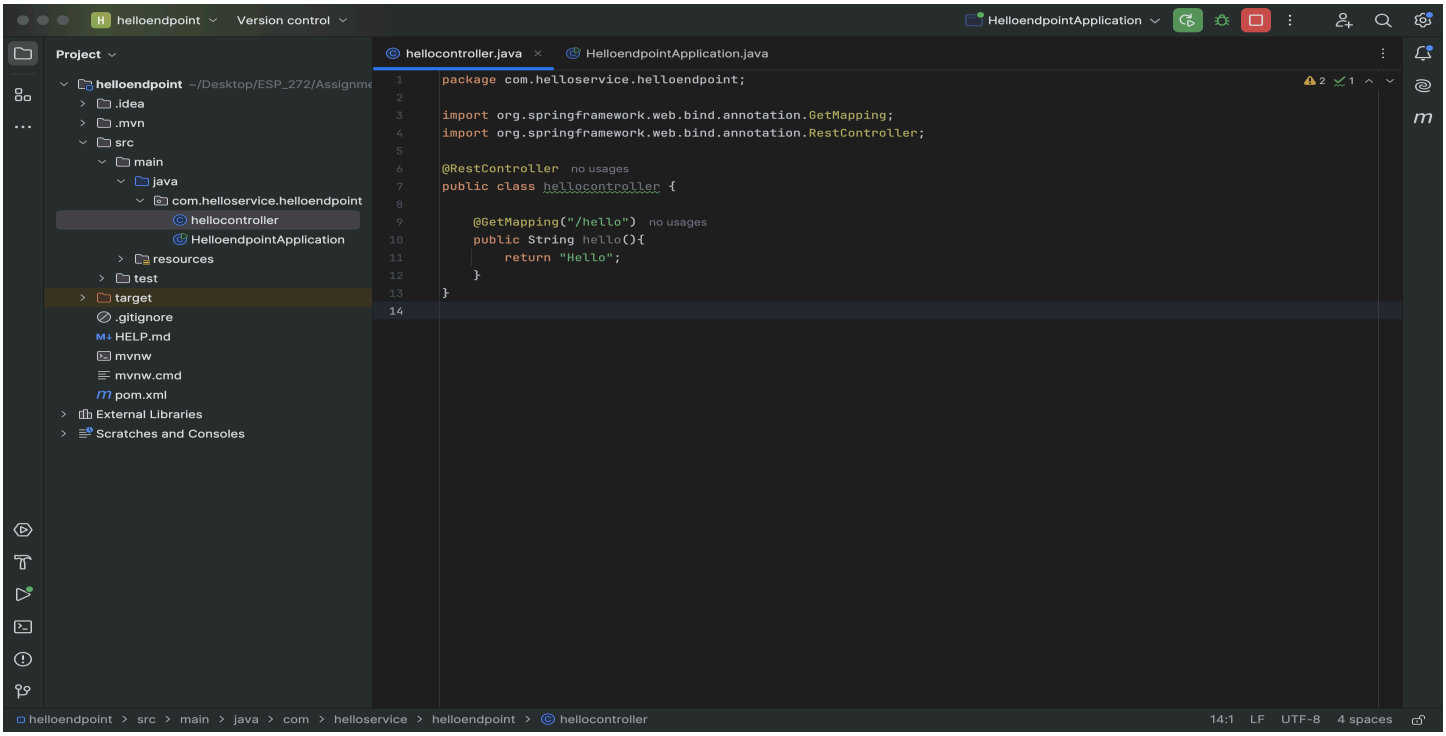
Create an endpoint `/hello` that returns the string `"Hello"`.

To generate spring project,

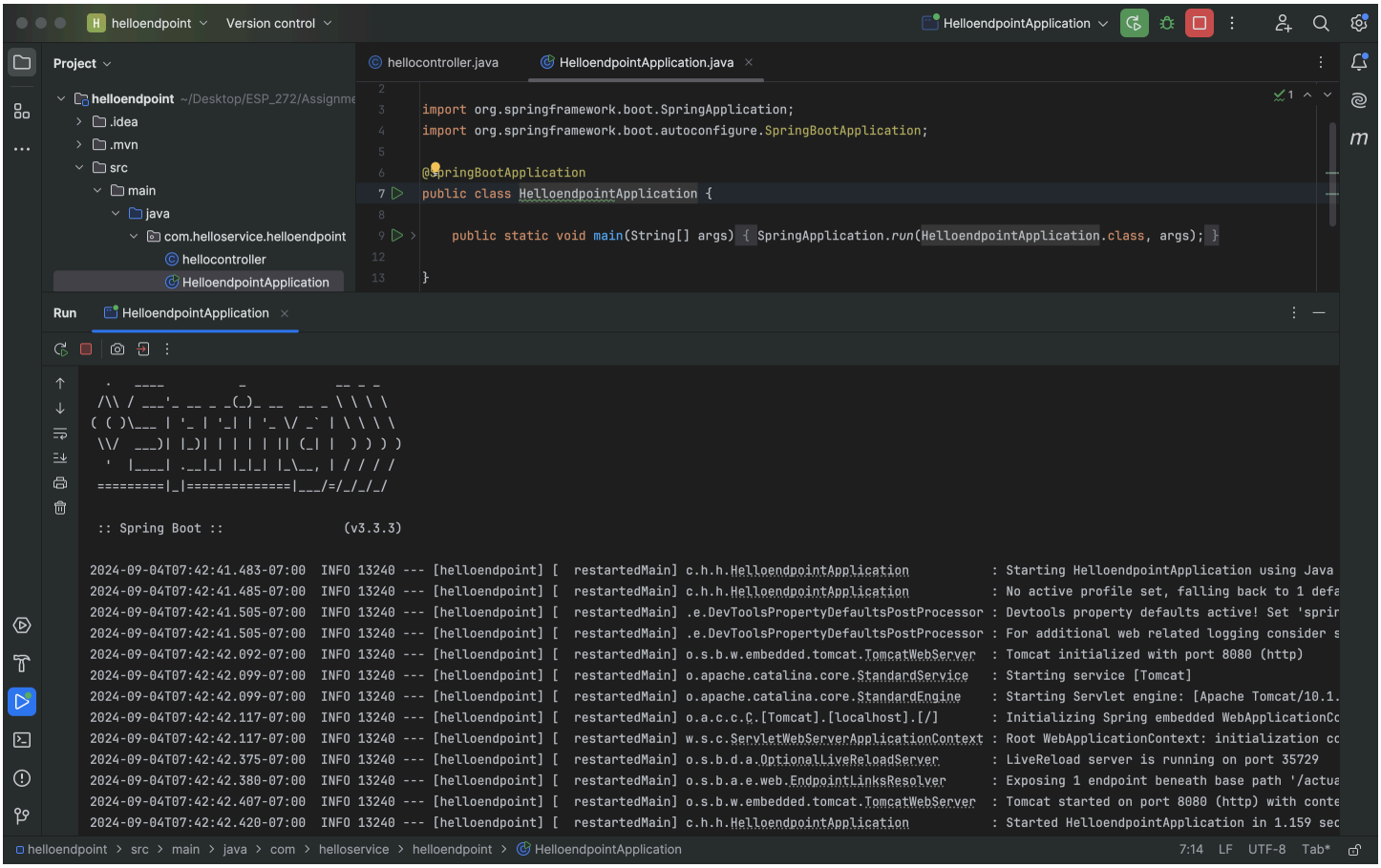
- Go to https://start.spring.io/ website to initialize spring project
- Select
    - Project-Maven
    - Spring Boot-3.3.3
    - Project Metadata
        - Group: com.helloservice
        - Artifact: helloendpoint
        - Name: helloendpoint
        - Packaging: Jar
        - Java: 22
    - Dependencies:
        - Spring Web (To call restAPI)
        - Spring Boot Actuator (To check health of service)
        - Spring Boot DevTools (To load things at runtime)
- Generate the file

Open this generated file as a project in IntelliJ IDEA CE.

Create a controller which end-point is “/hello” and return a string “Hello”.



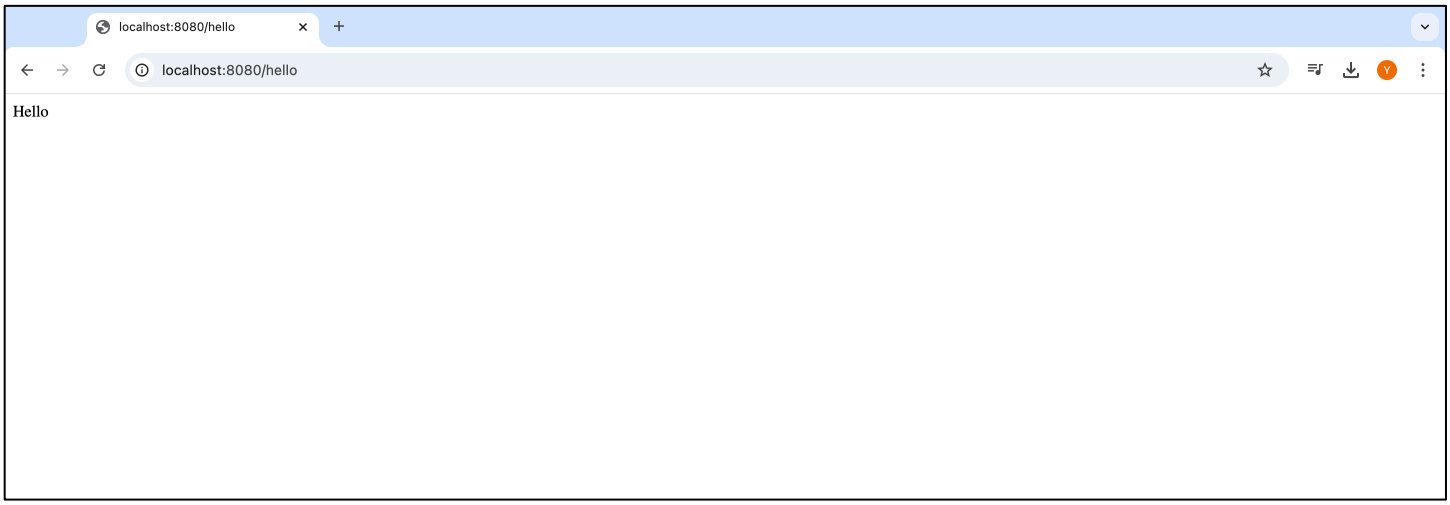
Now, run helloendpointApplication in IDE.



This project is run on port 8080.

Write localhost:8080/hello in Chrome web browser and see the output.

Output:



2) World service

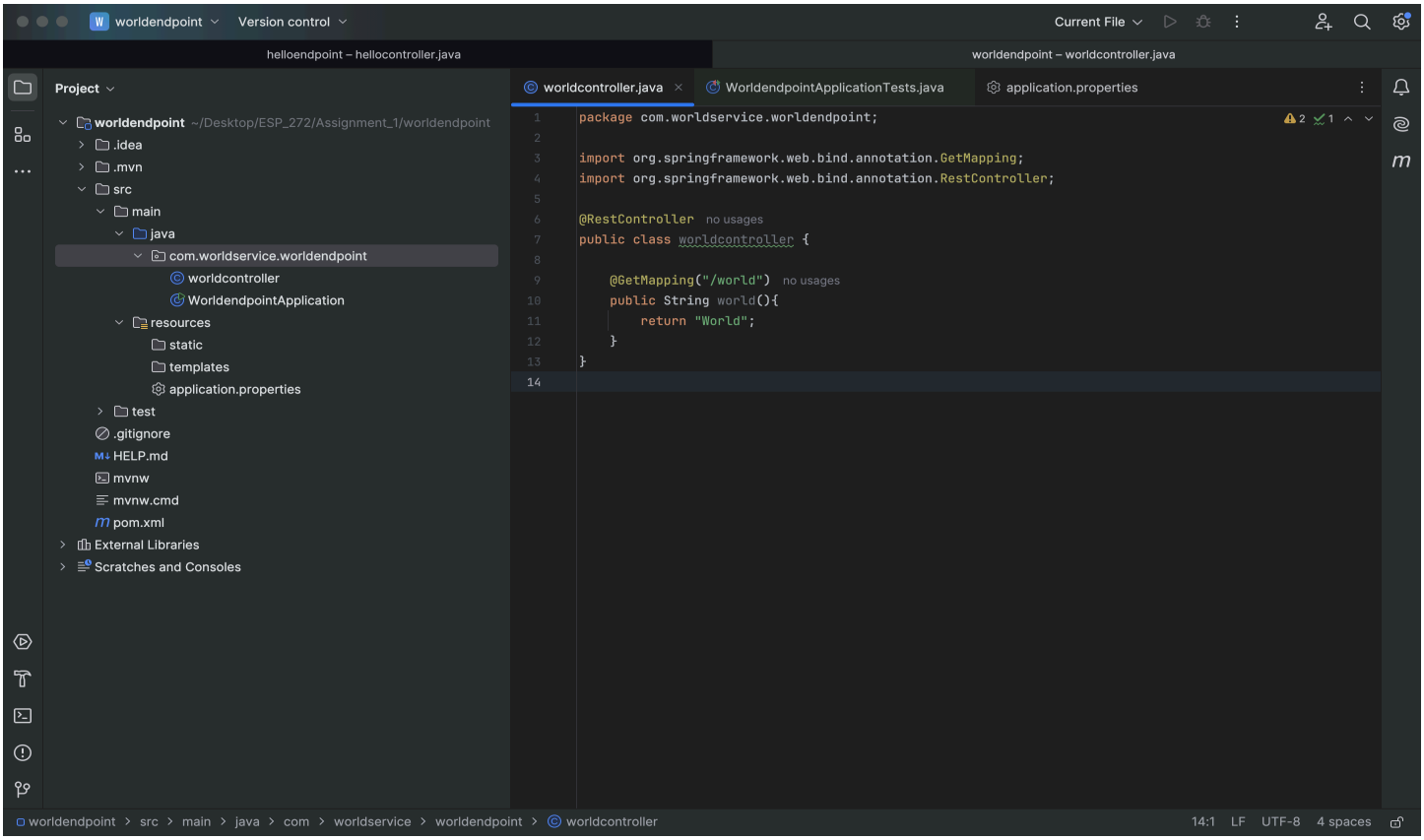
Create an endpoint `/world` that returns the string `"World"`.

To generate spring project,

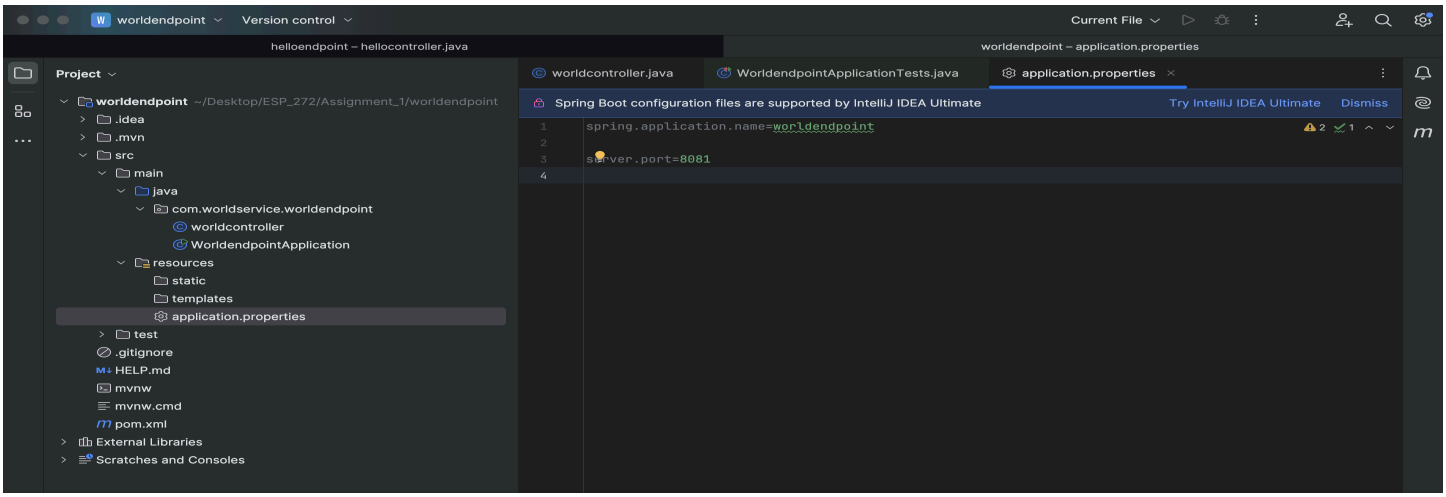
- Go to https://start.spring.io/ website to initialize spring project
- Select
    - Project-Maven
    - Spring Boot-3.3.3
    - Project Metadata
        - Group: com.worldservice
        - Artifact: worldendpoint
        - Name: worldendpoint
        - Packaging: Jar
        - Java: 22
    - Dependencies:
        - Spring Web (To call restAPI)
        - Spring Boot Actuator (To check health of service)
        - Spring Boot DevTools (To load things at runtime)
- Generate the file

Open this generated file as a project in IntelliJ IDEA CE.

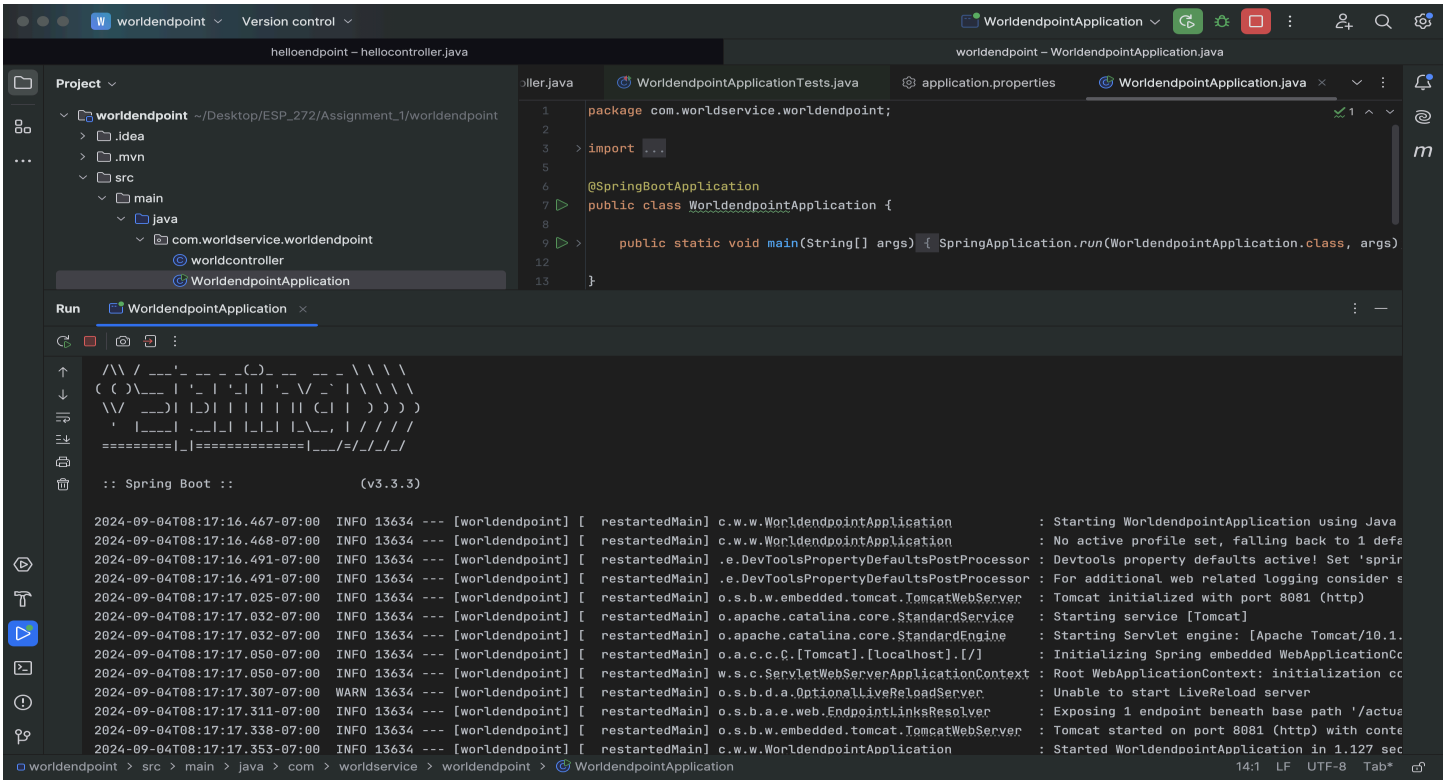
Create a controller which end-point is “/world” and return a string “World”.



Set server port 8081 in application.properties

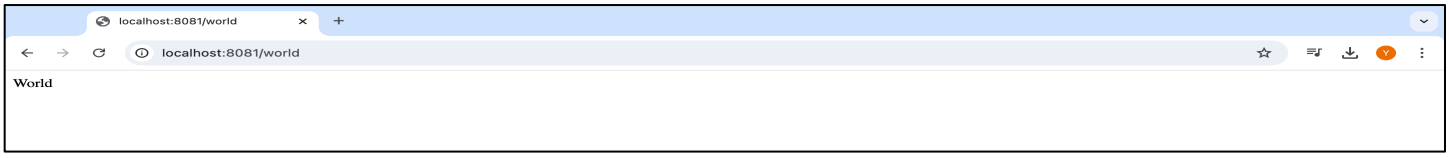


Noe, run worldendpointApplication in IDE.



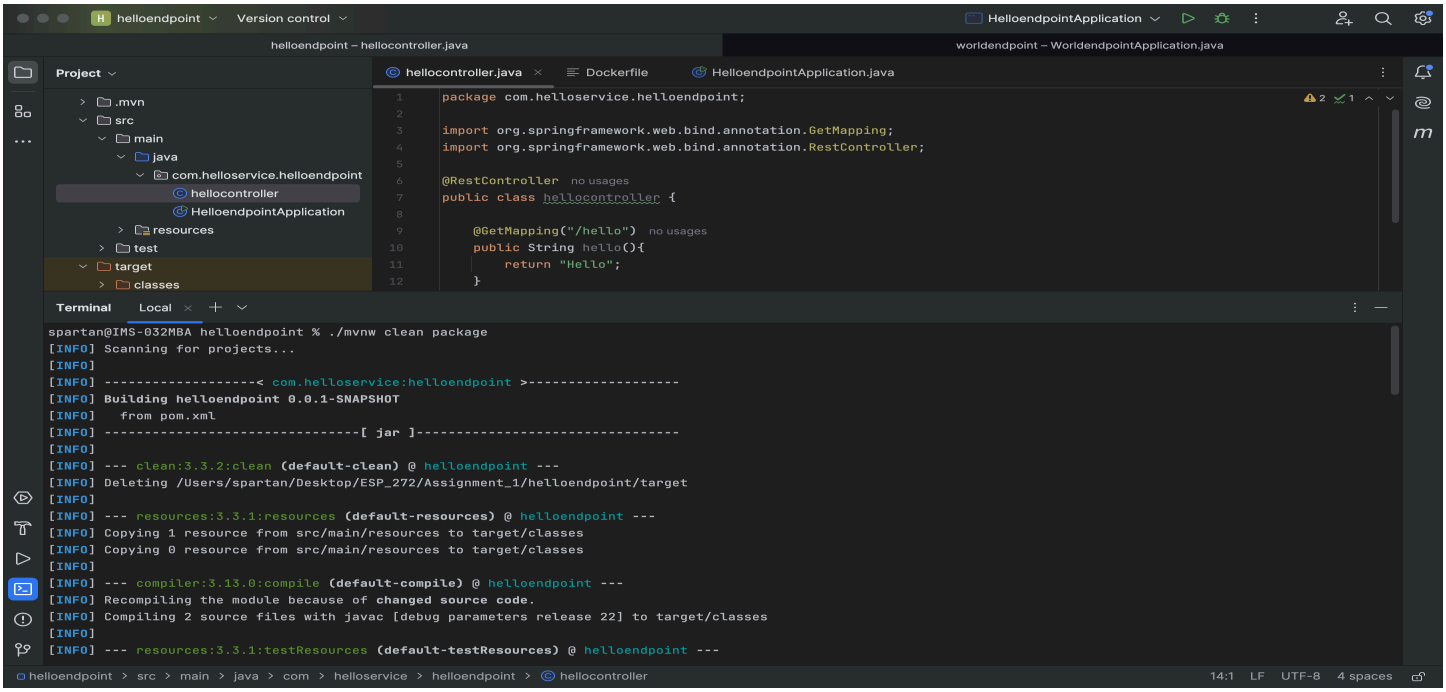
This project is run on port 8081.
Write localhost:8081/world in Chrome web browser and see the output.

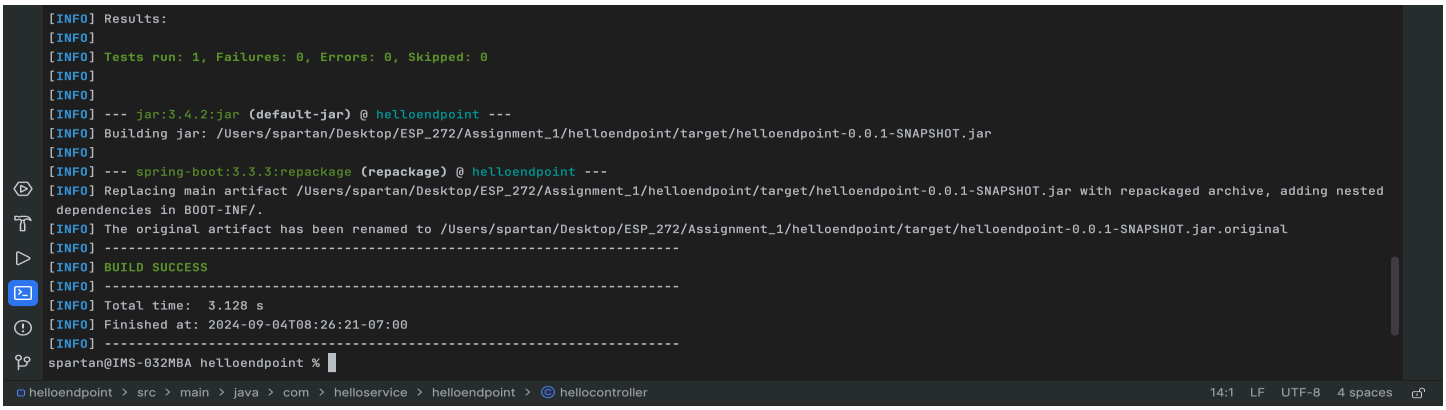
Output:

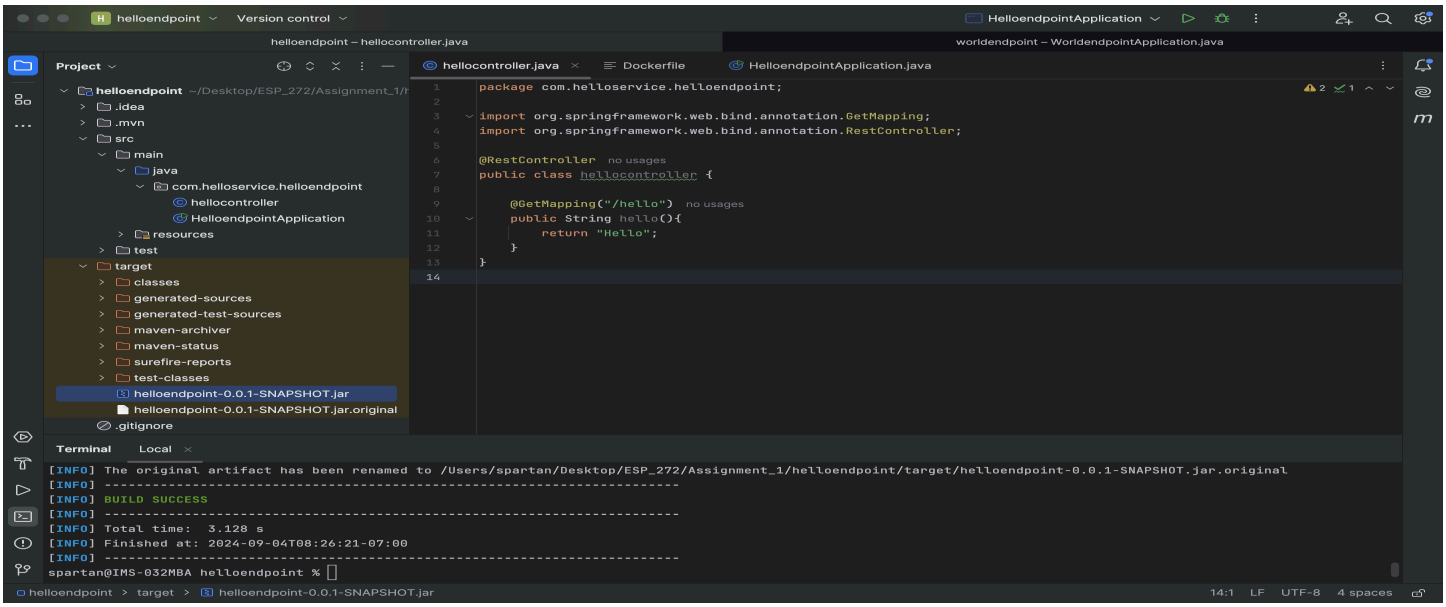


3) Generate .jar file for helloendpoint

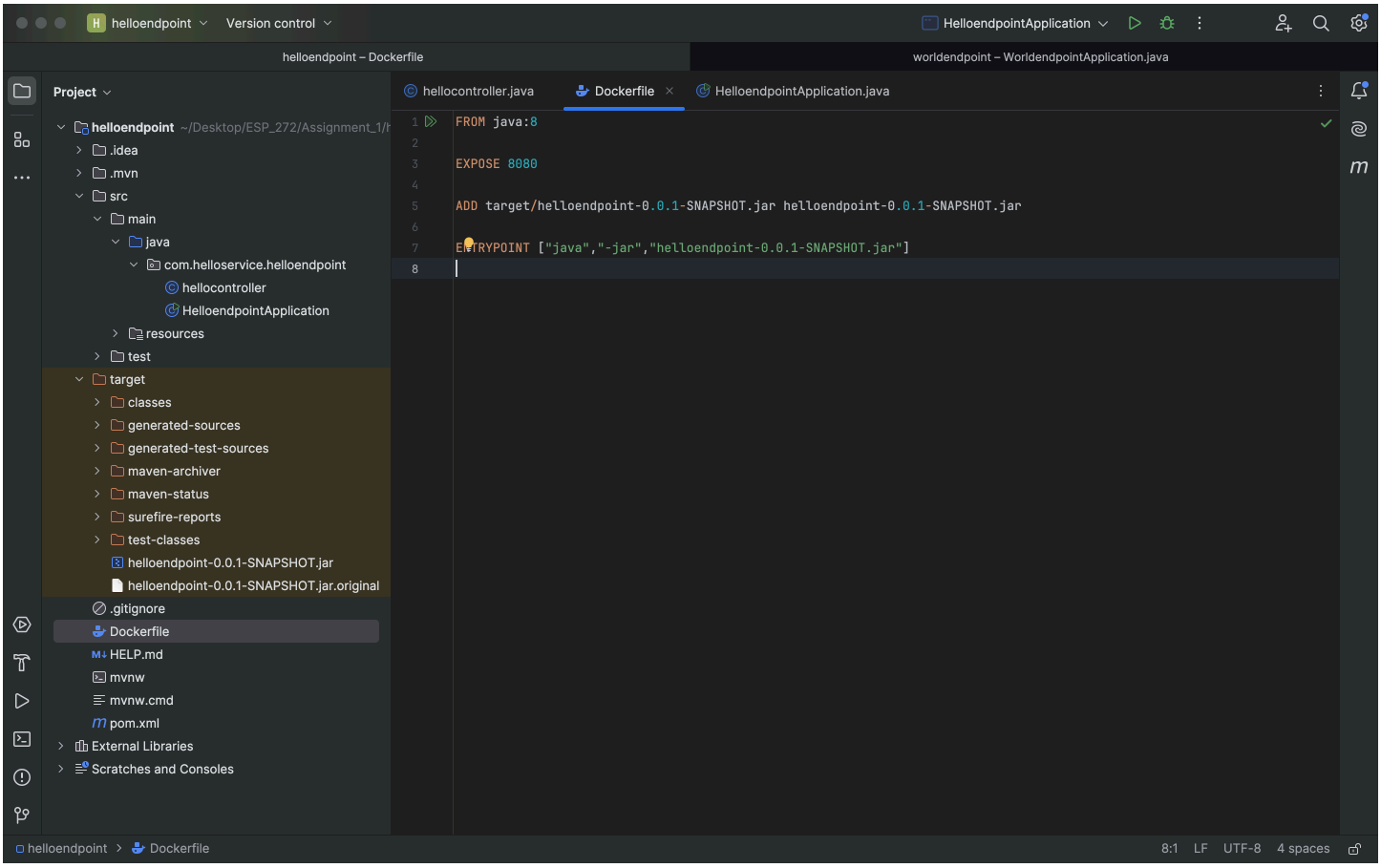
To generate .jar file write command “./mvnw clean package” in IDE terminal.







4) Create Docker File

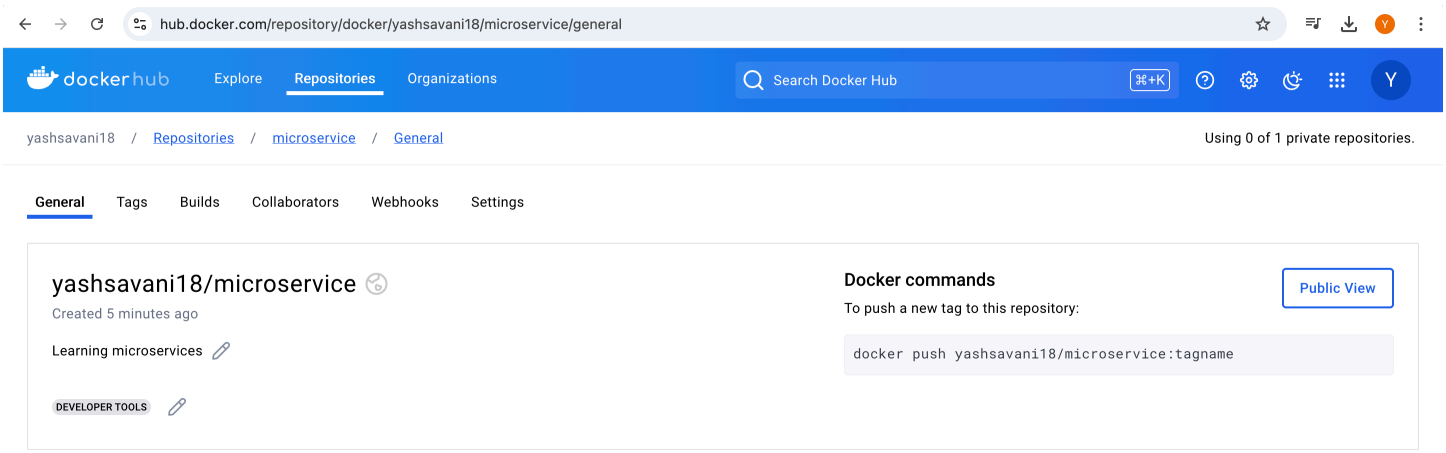


Go through same process in World service as well.

- **Containerizing the Microservices with Docker**

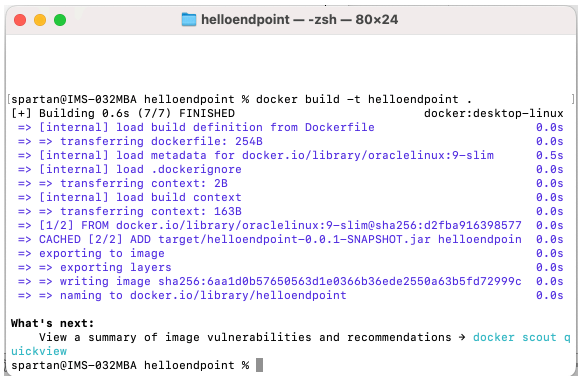
1) Create a repository in Docker HUB with name as microservice

Username: yashsavani18
Repository: yashsavani18/microservice

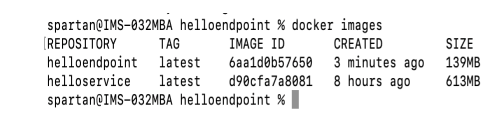


2) Create Docker image

To create docker image write “docker build -t helloendpoint .”in terminal.



To check docker image is created or not, write “docker images” in terminal.



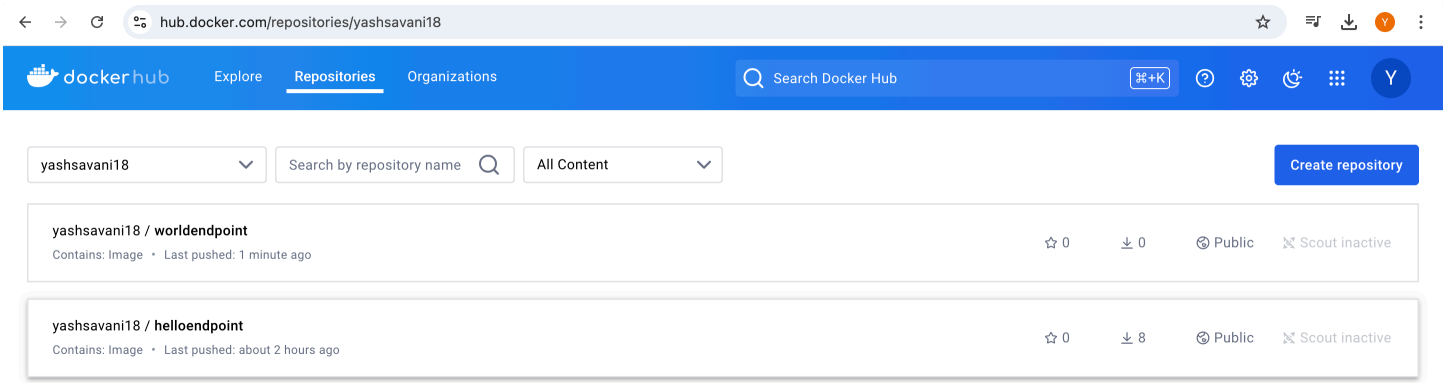
Here, you can see helloendpoint image is created in docker.

3) Push image to Docker Hub

To push image into docker hub write “docker tag helloendpoint yashsavani18/helloendpoint” and “docker push yashsavani18/helloendpoint” in terminal.

```
spartan@IMS-032MBA helloendpoint %
spartan@IMS-032MBA helloendpoint % docker tag helloendpoint yashsavani18/helloendpoint
spartan@IMS-032MBA helloendpoint % docker push yashsavani18/helloendpoint
Using default tag: latest
The push refers to repository [docker.io/yashsavani18/helloendpoint]
ade12c9346d7: Mounted from yashsavani18/microservice
6223193d8f8c: Mounted from yashsavani18/microservice
latest: digest: sha256:606ce8575df3dc482717ec76369140e18e150f9397d9b8c66e7f32f5c8de1ec5 size: 741
spartan@IMS-032MBA helloendpoint %
```

In Docker Hub, we can also see pushed image.



4) Remove images from locally

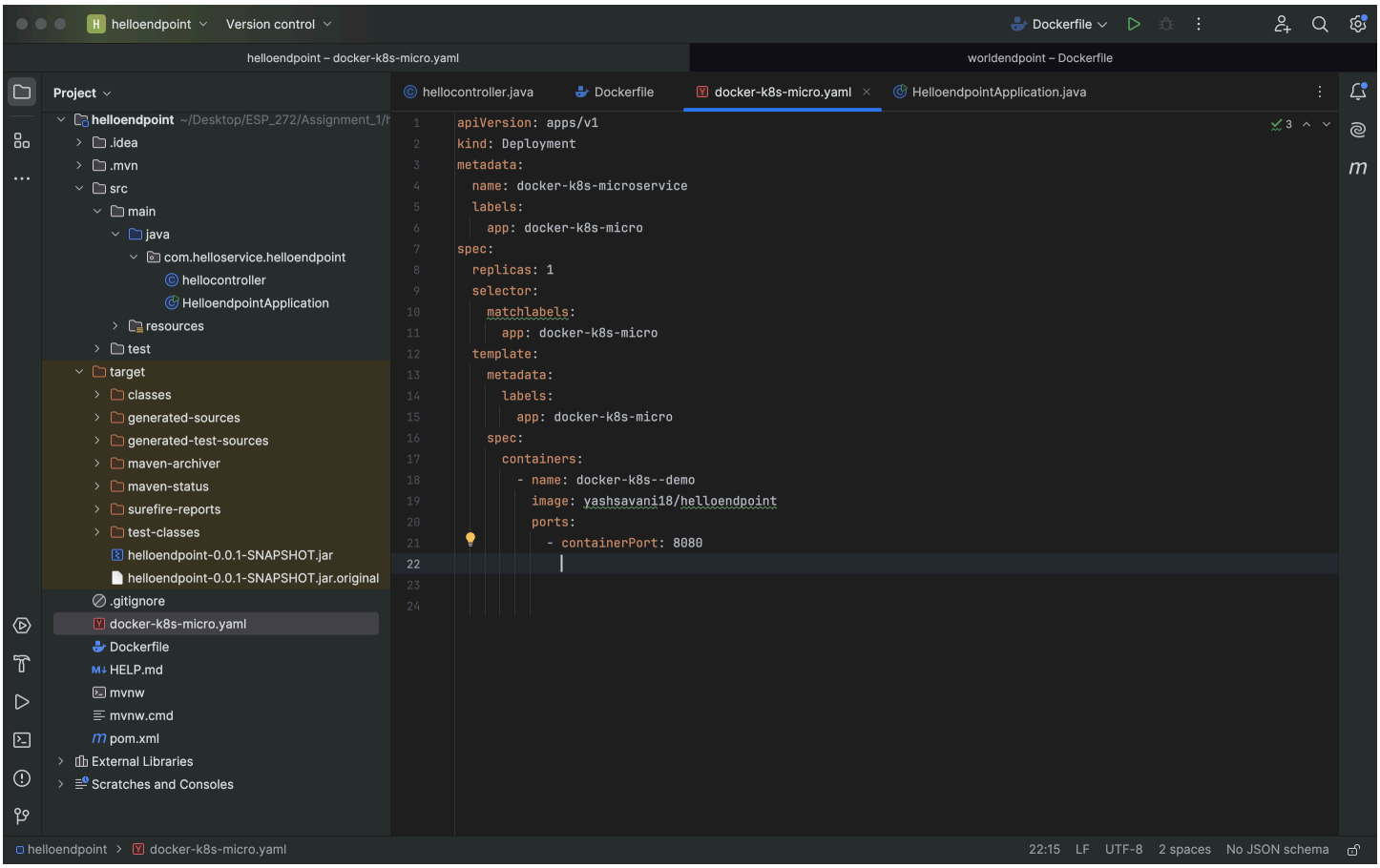
To remove image, write code “docker rmi helloendpoint yashsavani18/helloendpoint

5) Pull and Run image using Docker Hub

To pull image, write “docker run -p 8080:8080 yashsavani18/helloendpoint”

- **Deploying the Application on Kubernetes**

- Go to Google Cloud Platform.
- To create cluster, click on Kubernetes Engine -> Cluster -> Create Cluster.
- Provide name of cluster “k8s-docker-microservice”
- Create deployment file docker-k8s-micro.yaml



- Now, deploy this file in Kubernetes engine.
- Click on connect button and then click on run in cloud shell to connect the cluster.
- Hit enter and upload .yaml file.
- Write a comman “kubectl apply -f docker-k8s-micro”. During this command it will create container which creates replica/pods.
- When it status become in “ok” state then click on expose button to expose service.
- Enter port: 8080
- Click on external endpoint and provide endpoint=”/hello” it shows output “Hello”
- Follow same steps for worldservice as well.

- **Documentation**

This is the end of documentation and how to run the application is in above this page.

Links:

GitHub: https://github.com/yashsavani16/cmpe272_assignment1/tree/master

Docker HUB: https://hub.docker.com/repositories/yashsavani18