



## **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

## ● 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".

The screenshot shows a Java project named "helloendpoint" in an IDE. The "Project" view on the left lists files like ".gitignore", "HELP.md", "mvnw", "mvnw.cmd", "pom.xml", and "External Libraries". The "src/main/java/com.helloservice.helloendpoint" package is expanded, showing "HelloendpointApplication.java" and "hellocontroller.java". The "hellocontroller.java" file contains the following code:

```
package com.helloservice.helloendpoint;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class hellocontroller {
    @GetMapping("/hello")
    public String hello(){
        return "Hello";
    }
}
```

Now, run helloendpointApplication in IDE.

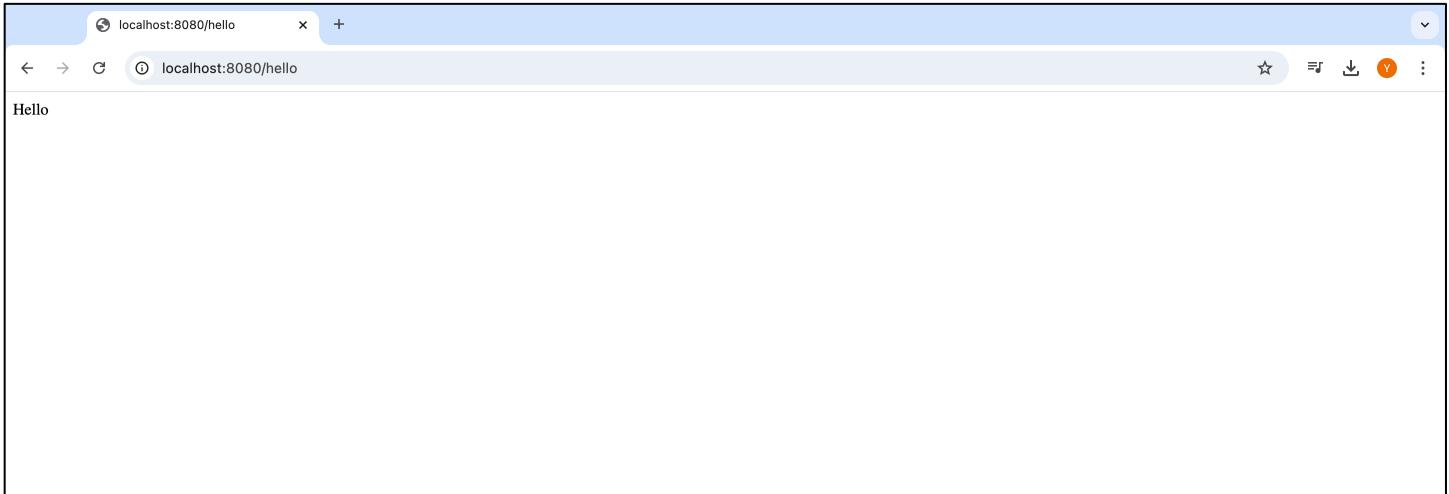
The screenshot shows the IDE running the "HelloendpointApplication" task. The terminal output shows the application starting up and listening on port 8080:

```
2024-09-04T07:42:41.483-07:00 INFO 13240 --- [helloendpoint] c.h.h.HelloendpointApplication : Starting HelloendpointApplication using Java
2024-09-04T07:42:41.485-07:00 INFO 13240 --- [helloendpoint] c.h.h.HelloendpointApplication : No active profile set, falling back to 1 defa
2024-09-04T07:42:41.505-07:00 INFO 13240 --- [helloendpoint] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'sprin
2024-09-04T07:42:41.505-07:00 INFO 13240 --- [helloendpoint] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider
2024-09-04T07:42:42.092-07:00 INFO 13240 --- [helloendpoint] [restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8080 (http)
2024-09-04T07:42:42.099-07:00 INFO 13240 --- [helloendpoint] [restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2024-09-04T07:42:42.099-07:00 INFO 13240 --- [helloendpoint] [restartedMain] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.
2024-09-04T07:42:42.117-07:00 INFO 13240 --- [helloendpoint] [restartedMain] o.a.c.c.C.[Tomcat].[localhost].[] : Initializing Spring embedded WebApplicationCo
2024-09-04T07:42:42.117-07:00 INFO 13240 --- [helloendpoint] [restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization co
2024-09-04T07:42:42.375-07:00 INFO 13240 --- [helloendpoint] [restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2024-09-04T07:42:42.380-07:00 INFO 13240 --- [helloendpoint] [restartedMain] o.s.b.a.e.web.EndpointLinksResolver : Exposing 1 endpoint beneath base path '/actua
2024-09-04T07:42:42.407-07:00 INFO 13240 --- [helloendpoint] [restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with conte
2024-09-04T07:42:42.420-07:00 INFO 13240 --- [helloendpoint] c.h.h.HelloendpointApplication : Started HelloendpointApplication in 1.159 sec
```

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
  - o Project-Maven
  - o Spring Boot-3.3.3
  - o Project Metadata
    - Group: com.worldservice
    - Artifact: worldendpoint
    - Name: worldendpoint
    - Packaging: Jar
    - Java: 22
  - o 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".

The screenshot shows the IntelliJ IDEA interface. On the left is the Project tool window displaying the file structure of the 'worldendpoint' project. The 'src' directory contains 'main' and 'java' folders. Inside 'java', there's a package 'com.worldservice.worldendpoint' with classes 'worldcontroller' and 'WorldendpointApplication'. The 'resources' folder contains 'static' and 'templates' subfolders, along with 'application.properties'. Other files visible include '.idea', '.mvn', '.gitignore', 'HELP.md', 'mvnw', 'mvnw.cmd', 'pom.xml', 'External Libraries', and 'Scratches and Consoles'. The right side of the screen has three tabs open: 'worldcontroller.java', 'WorldendpointApplicationTests.java', and 'application.properties'. The 'application.properties' tab shows the configuration file with the line 'server.port=8081' highlighted. The status bar at the bottom right indicates the time is 14:1, the line separator is LF, the encoding is UTF-8, and there are 4 spaces.

```
package com.worldservice.worldendpoint;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController no usages
public class worldcontroller {
    @GetMapping("/world") no usages
    public String world(){
        return "World";
    }
}
```

Set server port 8081 in application.properties

This screenshot shows the same IntelliJ IDEA environment as the previous one, but with a message overlay: 'Spring Boot configuration files are supported by IntelliJ IDEA Ultimate'. The 'application.properties' tab is active, showing the contents of the file: 'spring.application.name=worldendpoint' and 'server.port=8081'. The 'server.port' line is highlighted with a yellow dot. The status bar at the bottom right shows the time is 14:1, the line separator is LF, the encoding is UTF-8, and there are 4 spaces.

```
spring.application.name=worldendpoint
server.port=8081
```

Noe, run worldendpointApplication in IDE.

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Tree:** Shows the project structure under "worldendpoint".
- Editor:** Displays the code for `WorldendpointApplication.java` and `application.properties`.
- Run Tab:** Set to "WorldendpointApplication".
- Output Tab:** Shows the application's log output.

```
package com.worldservice.worldendpoint;
import ...
@SpringBootApplication
public class WorldendpointApplication {
    public static void main(String[] args) { SpringApplication.run(WorldendpointApplication.class, args)}
}
```

```
2024-09-04T08:17:16.467-07:00 INFO 13634 --- [worldendpoint] c.w.w.WorldendpointApplication : Starting WorldendpointApplication using Java
2024-09-04T08:17:16.468-07:00 INFO 13634 --- [worldendpoint] c.w.w.WorldendpointApplication : No active profile set, falling back to 1 defa
2024-09-04T08:17:16.491-07:00 INFO 13634 --- [worldendpoint] [ restartedMain] c.w.w.WorldendpointApplication : Starting WorldendpointApplication using Java
2024-09-04T08:17:16.491-07:00 INFO 13634 --- [worldendpoint] [ restartedMain] c.w.w.WorldendpointApplication : No active profile set, falling back to 1 defa
2024-09-04T08:17:16.491-07:00 INFO 13634 --- [worldendpoint] [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring-boot-devtools' to false to disable
2024-09-04T08:17:16.491-07:00 INFO 13634 --- [worldendpoint] [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting 'logging.level.web'
2024-09-04T08:17:17.025-07:00 INFO 13634 --- [worldendpoint] [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8081 (http)
2024-09-04T08:17:17.032-07:00 INFO 13634 --- [worldendpoint] [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2024-09-04T08:17:17.032-07:00 INFO 13634 --- [worldendpoint] [ restartedMain] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.14]
2024-09-04T08:17:17.050-07:00 INFO 13634 --- [worldendpoint] [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContainer
2024-09-04T08:17:17.050-07:00 INFO 13634 --- [worldendpoint] [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1 ms
2024-09-04T08:17:17.307-07:00 WARN 13634 --- [worldendpoint] [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : Unable to start LiveReload server
2024-09-04T08:17:17.311-07:00 INFO 13634 --- [worldendpoint] [ restartedMain] o.s.b.a.e.web.EndpointLinksResolver : Exposing 1 endpoint beneath base path '/actuator'
2024-09-04T08:17:17.338-07:00 INFO 13634 --- [worldendpoint] [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8081 (http) with conte
2024-09-04T08:17:17.353-07:00 INFO 13634 --- [worldendpoint] [ restartedMain] c.w.w.WorldendpointApplication : Started WorldendpointApplication in 1.127 sec
```

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.

Project: helloendpoint

Terminal:

```

spartan@IMSA-032MBA helloendpoint % ./mvnw clean package
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.helloservice:helloendpoint >-----
[INFO] Building helloendpoint 0.0.1-SNAPSHOT
[INFO]   from pom.xml
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- clean:3.3.2:clean (default-clean) @ helloendpoint ---
[INFO] Deleting /Users/spartan/Desktop/ESP_272/Assignment_1/helloendpoint/target
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ helloendpoint ---
[INFO] Copying 1 resource from src/main/resources to target/classes
[INFO] Copying 0 resource from src/main/resources to target/classes
[INFO]
[INFO] --- compiler:3.13.0:compile (default-compile) @ helloendpoint ---
[INFO] Recompiling the module because of changed source code.
[INFO] Compiling 2 source files with javac [debug parameters release 22] to target/classes
[INFO]
[INFO] --- resources:3.3.1:testResources (default-testResources) @ helloendpoint ---

```

14:1 LF UTF-8 4 spaces

Terminal:

```

[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] --- jar:3.4.2:jar (default-jar) @ helloendpoint ---
[INFO] Building jar: /Users/spartan/Desktop/ESP_272/Assignment_1/helloendpoint/target/helloendpoint-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] --- spring-boot:3.3.3:repackage (repackage) @ helloendpoint ---
[INFO] Replacing main artifact /Users/spartan/Desktop/ESP_272/Assignment_1/helloendpoint/target/helloendpoint-0.0.1-SNAPSHOT.jar with repackaged archive, adding nested dependencies in BOOT-INF/.
[INFO] The original artifact has been renamed to /Users/spartan/Desktop/ESP_272/Assignment_1/helloendpoint/target/helloendpoint-0.0.1-SNAPSHOT.jar.original
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 3.128 s
[INFO] Finished at: 2024-09-04T08:26:21-07:00
[INFO]
[INFO] spartan@IMSA-032MBA helloendpoint %

```

14:1 LF UTF-8 4 spaces

Project: helloendpoint

Terminal:

```

[INFO] The original artifact has been renamed to /Users/spartan/Desktop/ESP_272/Assignment_1/helloendpoint/target/helloendpoint-0.0.1-SNAPSHOT.jar.original
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 3.128 s
[INFO] Finished at: 2024-09-04T08:26:21-07:00
[INFO]
[INFO] spartan@IMSA-032MBA helloendpoint %

```

14:1 LF UTF-8 4 spaces

## 4) Create Docker File

The screenshot shows a Java IDE interface with two tabs open: "Helloendpoint – Dockerfile" and "HelloendpointApplication.java". The Dockerfile tab contains the following code:

```
1 FROM java:8
2
3 EXPOSE 8080
4
5 ADD target/helloendpoint-0.0.1-SNAPSHOT.jar helloendpoint-0.0.1-SNAPSHOT.jar
6
7 ENTRYPOINT ["java", "-jar", "helloendpoint-0.0.1-SNAPSHOT.jar"]
8
```

The project structure on the left shows a directory named "helloendpoint" containing ".idea", ".mvn", "src" (with "main" and "java" subfolders), "test", and "target" (with "classes", "generated-sources", "generated-test-sources", "maven-archiver", "maven-status", "surefire-reports", "test-classes", "helloendpoint-0.0.1-SNAPSHOT.jar", and "helloendpoint-0.0.1-SNAPSHOT.jar.original"). The ".gitignore" file is also listed.

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

The screenshot shows the Docker Hub interface. At the top, the URL is hub.docker.com/repository/docker/yashsavani18/microservice/general. The navigation bar includes 'Explore', 'Repositories' (which is underlined), and 'Organizations'. A search bar says 'Search Docker Hub'. Below the bar, it shows 'yashsavani18 / Repositories / microservice / General'. It also indicates 'Using 0 of 1 private repositories.' On the left, there's a sidebar with tabs for 'General', 'Tags', 'Builds', 'Collaborators', 'Webhooks', and 'Settings'. The 'General' tab is selected. The main area shows the repository details: 'yashsavani18/microservice' was created 5 minutes ago, and it's described as 'Learning microservices'. There's a 'DEVELOPER TOOLS' section with a link. On the right, there's a 'Docker commands' section with a button for 'Public View' and a command box containing 'docker push yashsavani18/microservice:tagname'.

## 2) Create Docker image

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

A terminal window titled 'helloendpoint -- zsh - 80x24' displays the output of a 'docker build' command. The output shows the progress of building an image from a Dockerfile, including steps like loading the Dockerfile, transferring files, and creating layers. It ends with a success message: '[+] Building 0.6s (7/7) FINISHED'. Below the command, there's a 'What's next?' section with a 'View a summary of image vulnerabilities and recommendations' link and a 'quickview' link. The prompt ends with 'spartan@IMS-032MBA helloendpoint %'.

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

```
spartan@IMS-032MBA helloendpoint % docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
helloendpoint        latest   6aa1d0b57650   3 minutes ago  139MB
helloservice         latest   d90cfaf7a8081   8 hours ago   613MB
```

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@IM-032MBA helloendpoint %]
[spartan@IM-032MBA helloendpoint %] docker tag helloendpoint yashsavani18/helloendpoint
[spartan@IM-032MBA helloendpoint %] docker push yashsavani18/helloendpoint
Using default tag: latest
The push refers to repository [docker.io/yashsavani18/helloendpoint]
ade12c9346d7: Mounted from yashsavani18/microservice
6223193d8fb8: Mounted from yashsavani18/microservice
latest: digest: sha256:686ce8575df3dc482717ec76369140e18e150f9397d9b8c66e7f32f5c8de1ec5 size: 741
spartan@IM-032MBA helloendpoint %
```

In Docker Hub, we can also see pushed image.

The screenshot shows the Docker Hub interface. At the top, there's a search bar with "Search Docker Hub" and a "Create repository" button. Below the search bar, there are tabs for "Explore" and "Repositories". The "Repositories" tab is selected, and the search results show two entries:

- yashsavani18 / worldendpoint**: Contains: Image • Last pushed: 1 minute ago. It has 0 stars, 0 forks, and is public. The status is "Scout inactive".
- yashsavani18 / helloendpoint**: Contains: Image • Last pushed: about 2 hours ago. It has 0 stars, 8 forks, and is public. The status is "Scout inactive".

#### 4) Remove images from locally

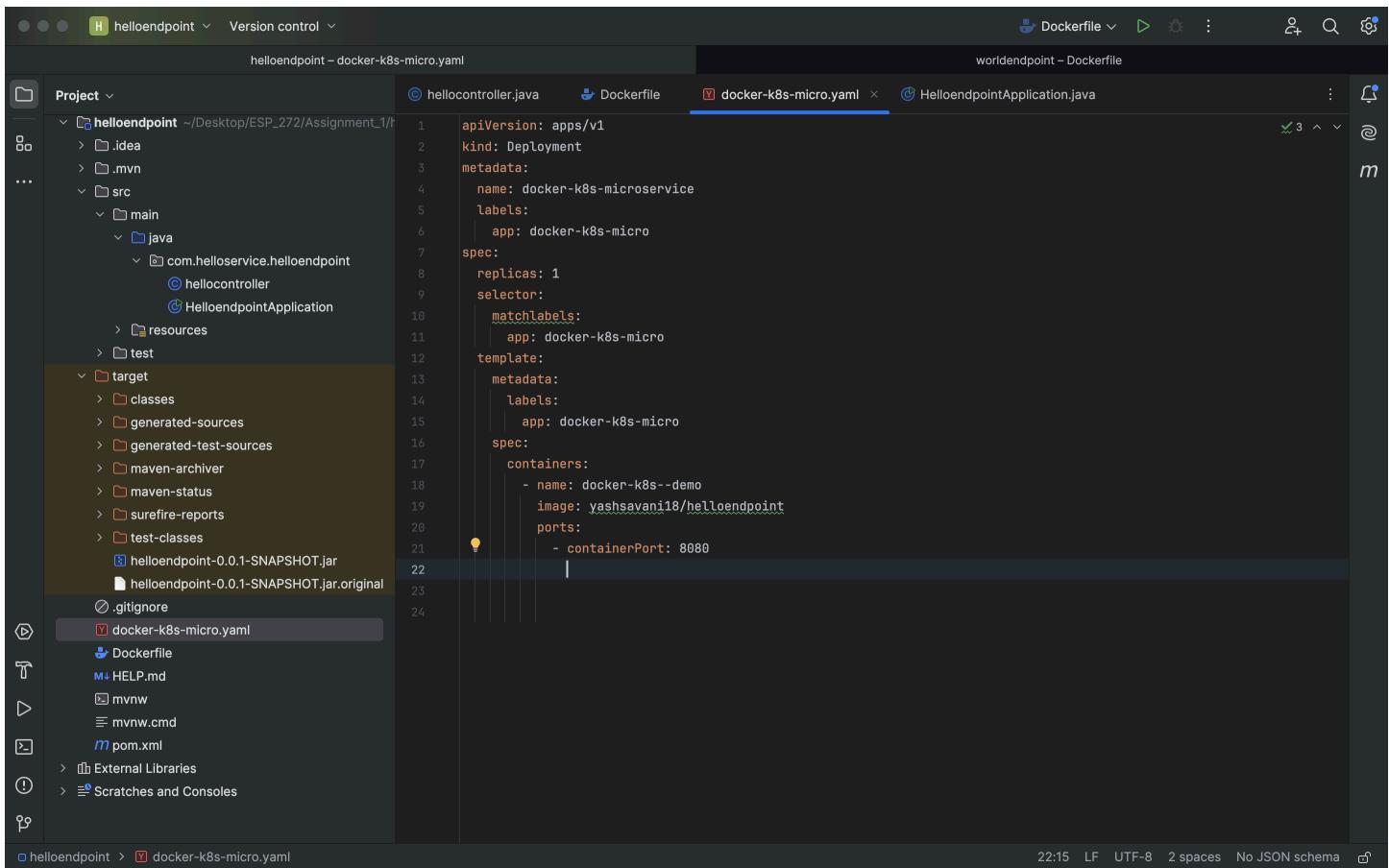
To remove image, write code “docker rmi helloendpoint yashsavani18/helloendpoint”

#### 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



The screenshot shows the Eclipse IDE interface with the following details:

- Project View:** Shows the project structure for "helloendpoint". It includes a .idea folder, a .mvn folder, a src folder containing main and test subfolders, a target folder, and generated files like classes, generated-sources, generated-test-sources, maven-archiver, maven-status, surefire-reports, test-classes, and a JAR file helloendpoint-0.0.1-SNAPSHOT.jar.
- Editor Area:** The "docker-k8s-micro.yaml" file is open in the editor. The code is a Kubernetes Deployment manifest:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: docker-k8s-microservice
  labels:
    app: docker-k8s-micro
spec:
  replicas: 1
  selector:
    matchLabels:
      app: docker-k8s-micro
  template:
    metadata:
      labels:
        app: docker-k8s-micro
    spec:
      containers:
        - name: docker-k8s--demo
          image: yashsavani18/helloendpoint
          ports:
            - containerPort: 8080
```

- Toolbars and Status Bar:** The top bar shows tabs for Dockerfile, Hellocontroller.java, Dockerfile, docker-k8s-micro.yaml (which is selected), and HelloendpointApplication.java. The status bar at the bottom right shows the time as 22:15, LF encoding, UTF-8 encoding, 2 spaces, and No JSON schema.

- 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 command “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](https://github.com/yashsavani16/cmpe272_assignment1/tree/master)

**Docker HUB:** <https://hub.docker.com/repositories/yashsavani18>