



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
 - o Project-Maven
 - o Spring Boot-3.3.3
 - o Project Metadata
 - Group: com.helloservice
 - Artifact: helloendpoint
 - Name: helloendpoint
 - 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 “/hello” and return a string “Hello”.

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

- Project View:** Shows the project structure under "helloendpoint". The "src/main/java/com.helloservice.helloendpoint" package contains two files: "HelloController.java" and "HelloendpointApplication.java". "HelloController.java" is currently selected and displayed in the editor.
- HelloController.java Content:**

```
1 package com.helloservice.helloendpoint;
2
3 import org.springframework.web.bind.annotation.GetMapping;
4 import org.springframework.web.bind.annotation.RestController;
5
6 @RestController
7 public class HelloController {
8
9     @GetMapping("/hello")
10    public String hello(){
11        return "Hello";
12    }
13}
```
- Editor Status:** The status bar at the bottom right indicates the file is 14:1 lines long, uses LF line endings, is in UTF-8 encoding, and has 4 spaces per indentation.

Now, run helloendpointApplication in IDE.

The screenshot shows a Java Spring Boot application named "Helloendpoint" in a code editor. The project structure includes a .idea folder, .mvn folder, and a src directory containing main/java/com.helloservice.helloendpoint/hellocontroller and HelloendpointApplication.java. The HelloendpointApplication.java file contains the following code:

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class HelloendpointApplication {

    public static void main(String[] args) {
        SpringApplication.run(HelloendpointApplication.class, args);
    }
}
```

The Run tab is active, showing the output of the application's execution. The logs indicate the application is starting up, using Spring Boot version v3.3.0, and that no active profile is set, falling back to the default. It also shows the initialization of Tomcat and the Spring embedded WebApplicationContext.

```
2024-09-04T07:42:41.483-07:00 INFO 13240 --- [helloendpoint] [ restartedMain] c.h.h.HelloendpointApplication : Starting HelloendpointApplication using Java
2024-09-04T07:42:41.485-07:00 INFO 13240 --- [helloendpoint] [ restartedMain] c.h.h.HelloendpointApplication : No active profile set, falling back to 1 defa
2024-09-04T07:42:41.505-07:00 INFO 13240 --- [helloendpoint] [ restartedMain] e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'sprin
2024-09-04T07:42:41.505-07:00 INFO 13240 --- [helloendpoint] [ restartedMain] e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider s
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.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] [ restartedMain] 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”.

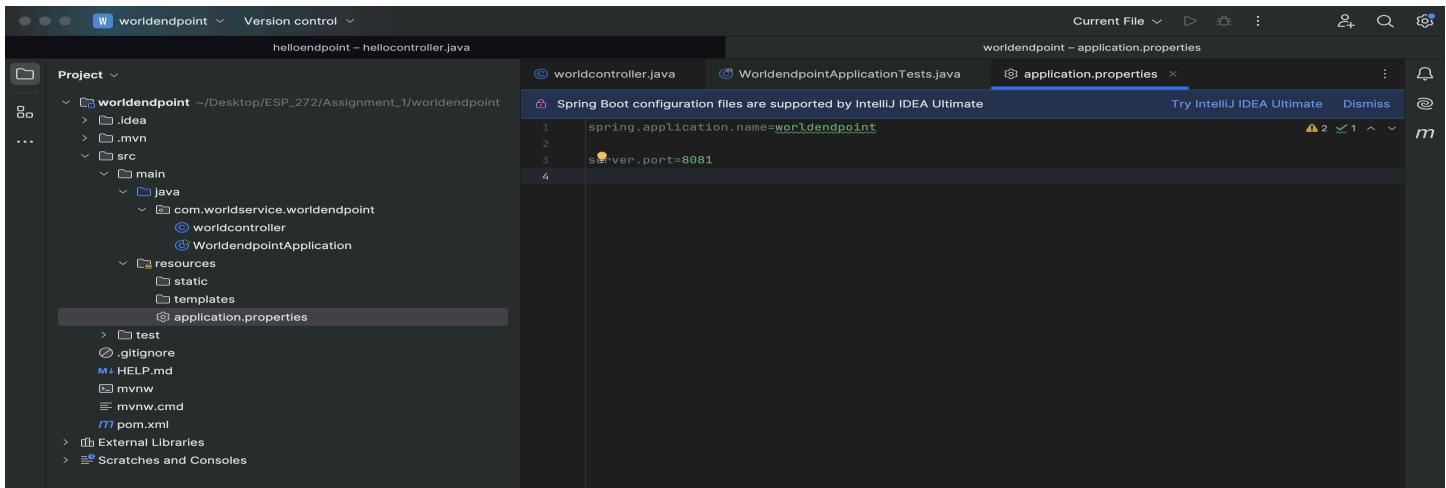
The screenshot shows the IntelliJ IDEA CE interface with the following details:

- Project View:** On the left, the project structure is displayed under "worldendpoint". It includes a .idea folder, .mvn, src (containing main/java/com/worldservice/worldendpoint/worldcontroller.java and WorldendpointApplication.java), resources (static and templates), application.properties, test, .gitignore, HELP.md, mvnw, mvnw.cmd, pom.xml, External Libraries, and Scratches and Consoles.
- Code Editor:** The right side shows the code editor for `worldcontroller.java`. The code is as follows:

```
1 package com.worldservice.worldendpoint;
2
3 import org.springframework.web.bind.annotation.GetMapping;
4 import org.springframework.web.bind.annotation.RestController;
5
6 @RestController no usages
7 public class worldcontroller {
8
9     @GetMapping("/world") no usages
10    public String world(){
11        return "World";
12    }
13}
14
```

The code editor also shows `WorldendpointApplicationTests.java` and `application.properties` in the tabs above the code area. The status bar at the bottom indicates the file is 14:1, LF, UTF-8, 4 spaces, and has 2 errors (indicated by a red exclamation mark icon).

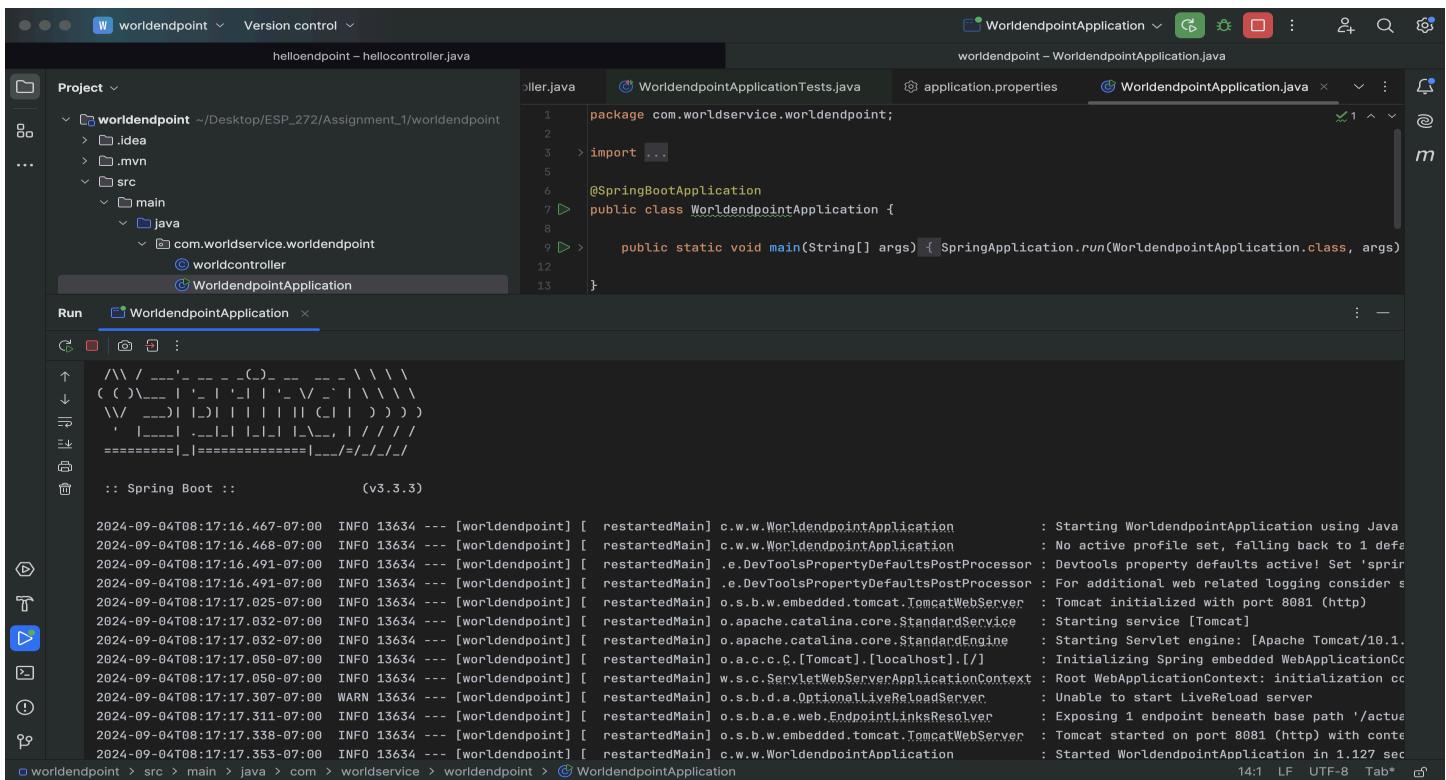
Set server port 8081 in application.properties



The screenshot shows the IntelliJ IDEA interface with the project 'worldendpoint' open. The 'application.properties' file is selected in the right-hand editor pane. The content of the file is as follows:

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

Noe, run worldendpointApplication in IDE.



The screenshot shows the IntelliJ IDEA interface with the 'WorldendpointApplication' run configuration selected in the bottom-left toolbar. The right-hand editor pane displays the 'WorldendpointApplication.java' file, which contains the main method:

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

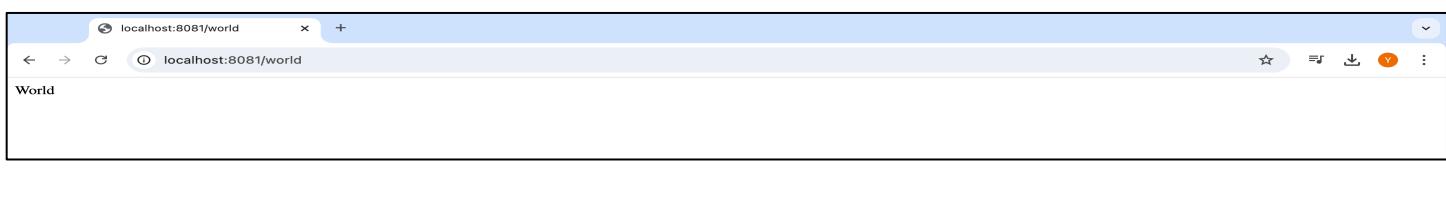
Below the editor, the 'Run' tool window shows the output of the application's execution. The log output includes:

```
2024-09-04T08:17:16.467-07:00 INFO 13634 --- [worldendpoint] [ restartedMain] c.w.w.WorldendpointApplication : Starting WorldendpointApplication using Java
2024-09-04T08:17:16.468-07:00 INFO 13634 --- [worldendpoint] [ restartedMain] c.w.w.WorldendpointApplication : No active profile set, falling back to 1 default
2024-09-04T08:17:16.491-07:00 INFO 13634 --- [worldendpoint] [ restartedMain] e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devtools.restart.enabled=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=DEBUG'
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 WebApplicationContext
2024-09-04T08:17:17.050-07:00 INFO 13634 --- [worldendpoint] [ restartedMain] w.s.c.WebServerApplicationContext : 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 context path ''
2024-09-04T08:17:17.353-07:00 INFO 13634 --- [worldendpoint] [ restartedMain] c.w.w.WorldendpointApplication : Started WorldendpointApplication in 1.127 seconds (JVM running for 1.132)
```

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.

The screenshot shows the IntelliJ IDEA interface with the project 'helloendpoint' open. The terminal window at the bottom displays the command 'mvnw clean package' being run, along with the Maven build logs. The logs show the project is being scanned, resources are being copied, the compiler is running, and test resources are being processed. The build completes successfully with a total time of 3.128 seconds.

```
spartan@IM-S-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 ---
[INFO]

o helloendpoint > src > main > java > com > helloservice > helloendpoint > hellocontroller
14:1 LF UTF-8 4 spaces
```

This screenshot shows the terminal output after the Maven build has completed. It includes the results summary, test execution details, and the final message indicating the artifact was repackaged and renamed. The build took 3.128 seconds and finished at 2024-09-04T08:26:21-07:00.

```
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[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@IM-S-032MBA helloendpoint %
14:1 LF UTF-8 4 spaces
```

This screenshot shows the terminal output again, but with the 'target' directory highlighted in the project tree. The output is identical to the previous one, showing the repackaging of the artifact and its renaming.

```
[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@IM-S-032MBA helloendpoint %
14:1 LF UTF-8 4 spaces
```

4) Create Docker File

The screenshot shows a Java development environment with a project named "helloendpoint". The project structure includes a "src" directory containing "main" and "java" sub-directories, which contain "HelloController" and "HelloEndpointApplication" classes respectively. A "target" directory contains generated JAR files, including "helloendpoint-0.0.1-SNAPSHOT.jar" and "helloendpoint-0.0.1-SNAPSHOT.jar.original". The "Dockerfile" tab is selected in the editor, displaying the following content:

```
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 "Dockerfile" file is highlighted in the sidebar.

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 selected), and 'Organizations'. A search bar says 'Search Docker Hub'. Below the navigation, it shows 'yashsavani18 / Repositories / microservice / General'. It indicates 'Using 0 of 1 private repositories.' On the left, there are tabs for 'General', 'Tags', 'Builds', 'Collaborators', 'Webhooks', and 'Settings'. The 'General' tab is selected. It displays the repository name 'yashsavani18/microservice' with a circular badge showing '0'. It was 'Created 5 minutes ago'. Below this, there's a 'Learning microservices' section with a link and a 'DEVELOPER TOOLS' section with a link. On the right, under 'Docker commands', it says 'To push a new tag to this repository:' followed by the command 'docker push yashsavani18/microservice:tagname'. There's also a 'Public View' button.

2) Create Docker image

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

A terminal window titled 'helloendpoint -- zsh -- 80x24' is shown. The command 'spartan@IMS-032MBA helloendpoint % docker build -t helloendpoint .' is run. The output shows the build process: '[+] Building 0.6s (7/7) FINISHED', '=> [internal] load build definition from Dockerfile', '=> transferring dockerfile: 254B', etc. It also shows the download of an Oracle Linux image and the creation of the 'helloendpoint' image. At the end, it says 'What's next: View a summary of image vulnerabilities and recommendations > docker scout q' and '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
spartan@IMS-032MBA helloendpoint %
```

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]
ade12c934d07: Mounted from yashsavani18/microservice
6223193d8f8c: Mounted from yashsavani18/microservice
latest: digest: sha256:686ce8575df3dc482717ec76369140e18e150f9397d9b8c66e7f32f5c8de1ec5 size: 741
spartan@IMS-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 "yashsavani18" and a "Create repository" button. Below the search bar, there are two repository cards:

- yashsavani18 / worldendpoint**: Contains: Image • Last pushed: 1 minute ago. It has 0 stars, 0 forks, and is marked as Public. It also has a "Scout inactive" badge.
- yashsavani18 / helloendpoint**: Contains: Image • Last pushed: about 2 hours ago. It has 0 stars, 8 forks, and is marked as Public. It also has a "Scout inactive" badge.

4) Remove images from locally

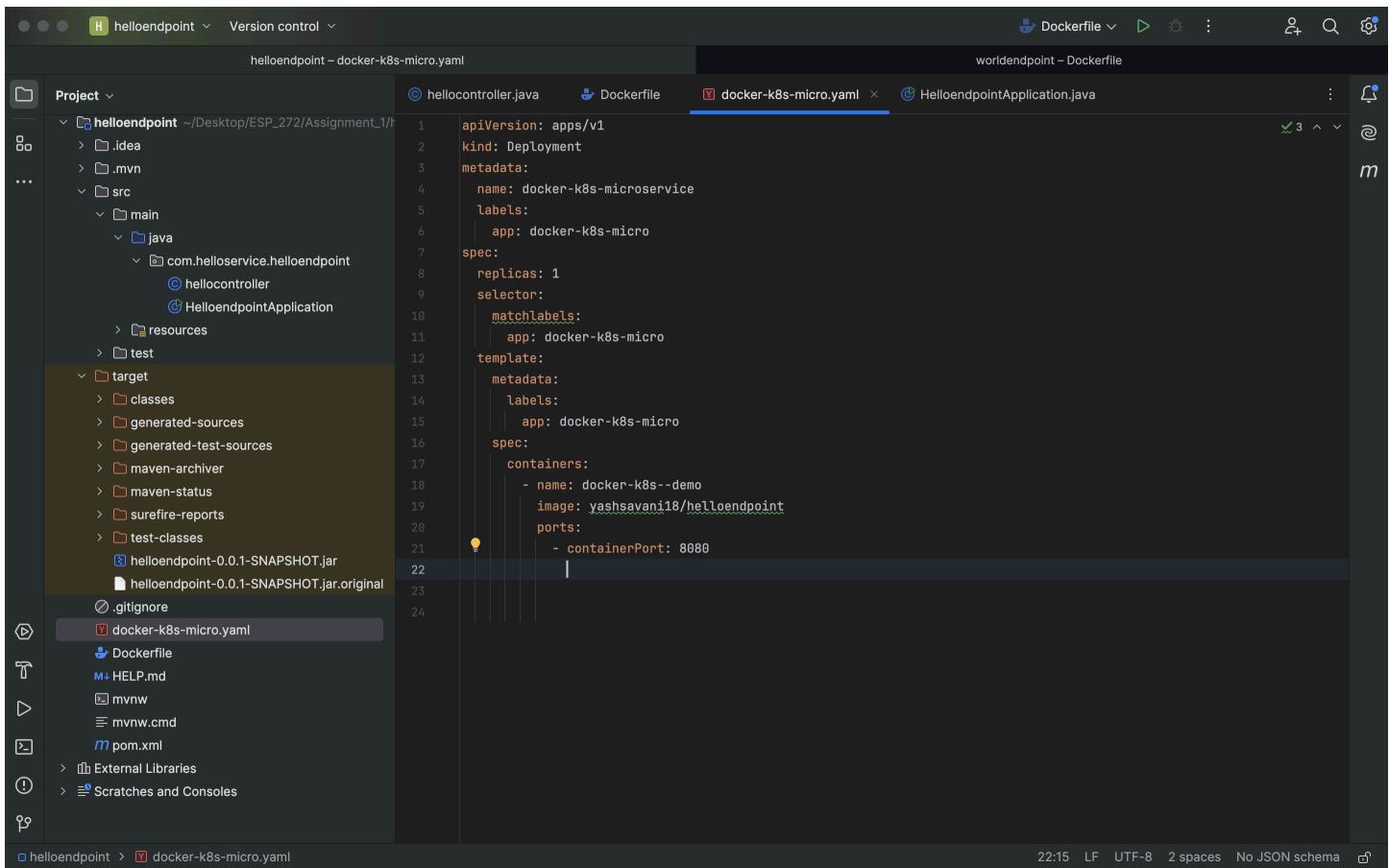
To remove image, write code “docker rmi helloendpoint yashsavani18/helloendpoint”

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



```
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
```

- 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 becomes “ok” 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>