Project 2: Catalog-Service

Project overview:

> This project is focusing on building a catog-service application using Java files and using spring boot with gradle to build and kubernetes to deploy the application.

The first step is to create our Java files and build them using spring intilizer and gradle, for this project I have chosen to use IntelliJ as my IDE. *Java and Gradle must be downloaded and installed locally. (I am using Java 17 and Gadle 8.3)

1.) CatalogServiceApplication.java

```java
package com.polarbookshop.catalogservice;

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

@SpringBootApplication
public class CatalogServiceApplication {

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

}
```

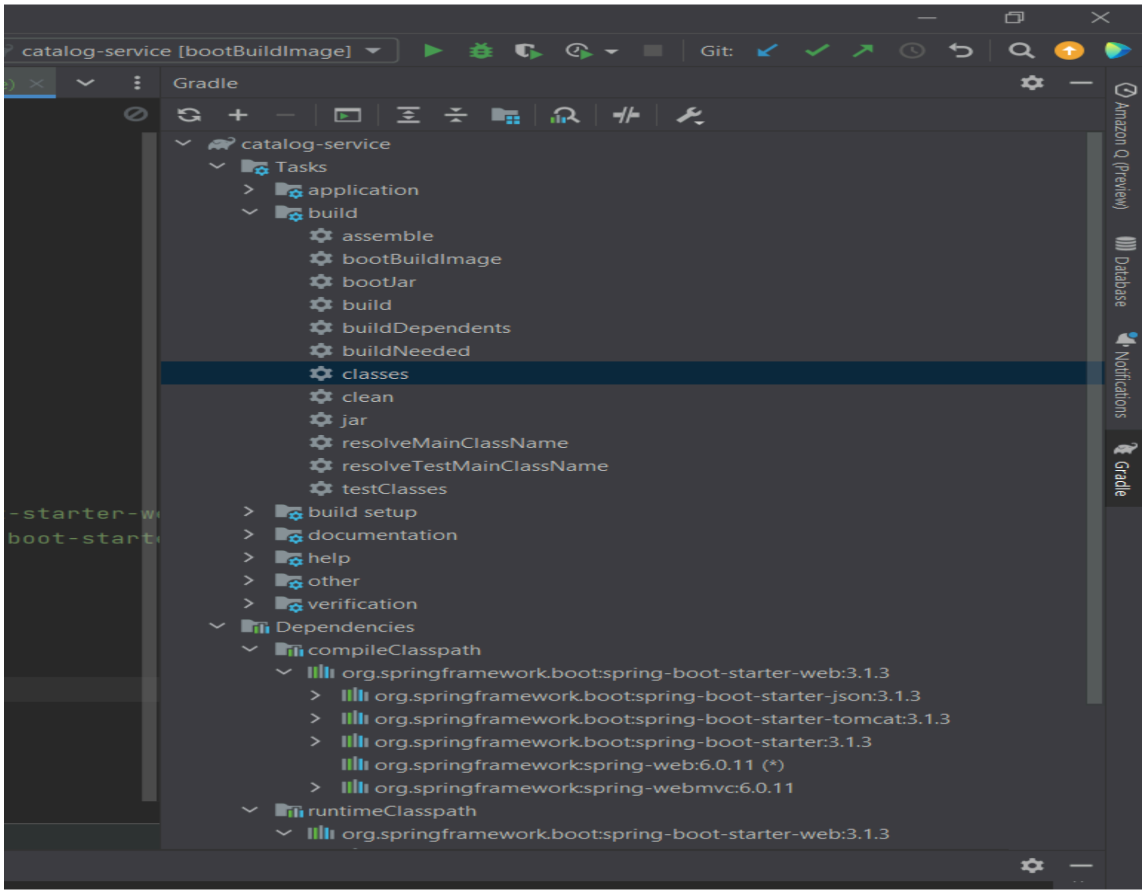
> This is the application that will be deployed.

2.) HomeController.java

```java
package com.polarbookshop.catalogservice;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HomeController {

   @GetMapping("/")
   public String getGreeting() {

       return "Welcome to the book catalog!";
   }
}
```

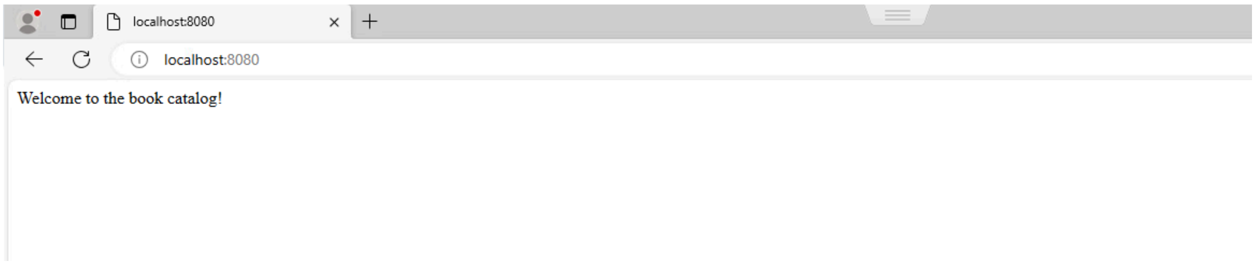
This Rest controller will determine a endpoint and will return the greeting "Welcome to the book catalog!" when the root URL is accessed.

3.) Open Docker Desktop (v4.24.2) now we can run the bootBuildImage command either within the CLI or on the GUI Via the catalog-service>build>bootBuildImage



```
catalog-service
  Tasks
    application
    build
      assemble
      bootBuildImage
      bootJar
      build
      buildDependents
      buildNeeded
      classes
      clean
      jar
      resolveMainClassName
      resolveTestMainClassName
      testClasses
    build setup
    documentation
    help
    other
    verification
  Dependencies
    compileClasspath
      org.springframework.boot:spring-boot-starter-web:3.1.3
        org.springframework.boot:spring-boot-starter-json:3.1.3
        org.springframework.boot:spring-boot-starter-tomcat:3.1.3
        org.springframework.boot:spring-boot-starter:3.1.3
        org.springframework:spring-web:6.0.11 (*)
        org.springframework:spring-webmvc:6.0.11
    runtimeClasspath
      org.springframework.boot:spring-boot-starter-web:3.1.3
```

4.) Verify the container is running and you can access the URL

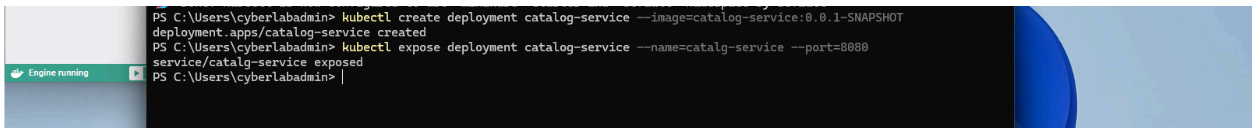


Welcome to the book catalog!

5.) Load the docker image into minikube using the command:
$ minikube image load catalog-service:0.0.1-SNAPSHOT

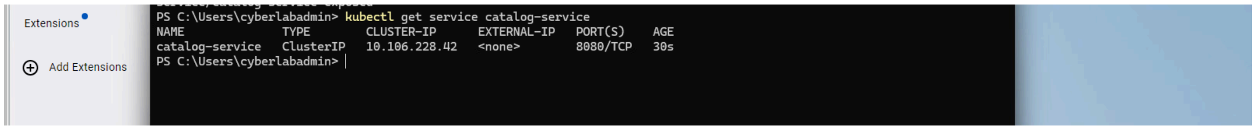
6.) Create deployment using the command:
$ kubectl create deployment catalog-service --image=catalog-service:0.0.1-SNAPSHOT

7.) Expose Deployment using the command:
$ kubectl expose deployment catalog-service --name=catalog-service --port=8080



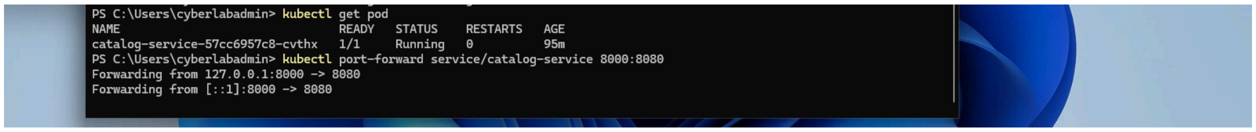
```
PS C:\Users\cyberlabadmin> kubectl create deployment catalog-service --image=catalog-service:0.0.1-SNAPSHOT
deployment.apps/catalog-service created
PS C:\Users\cyberlabadmin> kubectl expose deployment catalog-service --name=catalg-service --port=8080
service/catalg-service exposed
PS C:\Users\cyberlabadmin> |
```

8.) Verify the creation using the command:
$ kubectl get service catalog-service



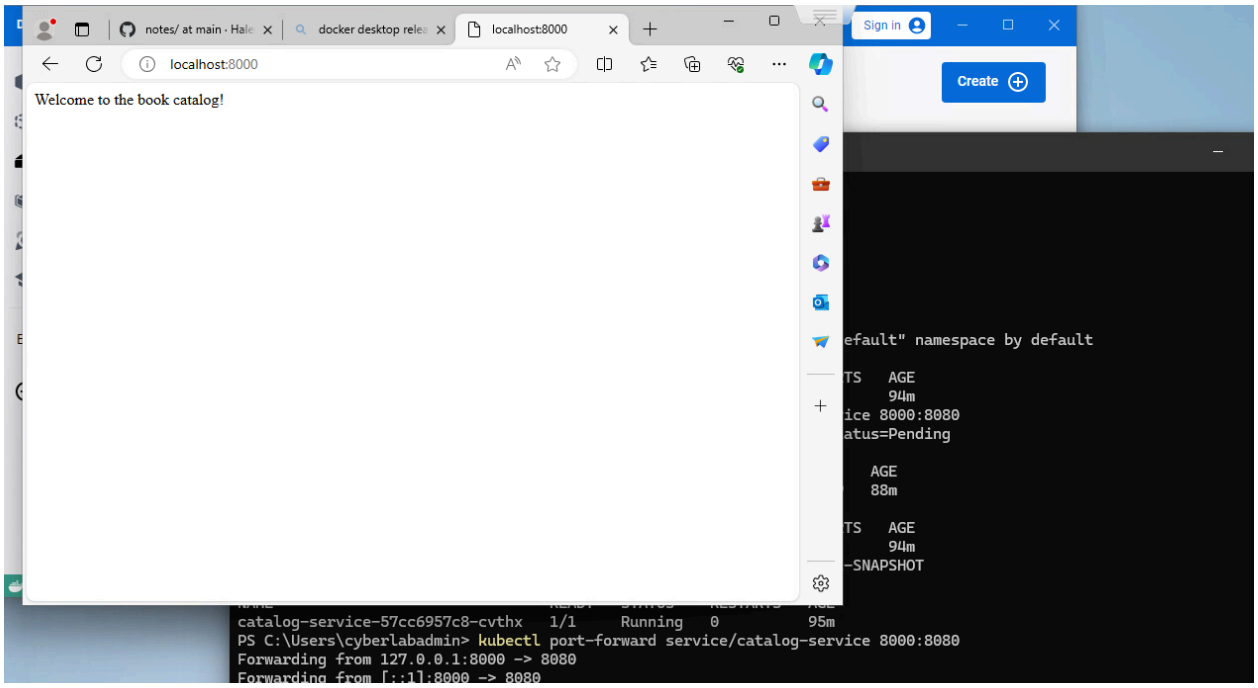
```
PS C:\Users\cyberlabadmin> kubectl get service catalog-service
NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
catalog-service   ClusterIP   10.106.228.42   <none>        8080/TCP   30s
PS C:\Users\cyberlabadmin> |
```

9.) Now we can forward the traffic from a local port on our computer (8000) to the port being used by the cluster (8080) using the following command:
$ kubectl port-forward service/catalog-service 8000:8080



```
PS C:\Users\cyberlabadmin> kubectl get pod
NAME                               READY   STATUS    RESTARTS   AGE
catalog-service-57cc6957c8-cvthx   1/1     Running   0          95m
PS C:\Users\cyberlabadmin> kubectl port-forward service/catalog-service 8000:8080
Forwarding from 127.0.0.1:8000 -> 8080
Forwarding from [::1]:8000 -> 8080
```

10.) Verify that you can access the URL with the port 8000 and you get the expected results



Welcome to the book catalog!

```
efault" namespace by default
TS   AGE
     94m
ice 8000:8080
atus=Pending
   AGE
   88m
TS   AGE
     94m
-SNAPSHOT
catalog-service-57cc6957c8-cvthx   1/1     Running   0          95m
PS C:\Users\cyberlabadmin> kubectl port-forward service/catalog-service 8000:8080
Forwarding from 127.0.0.1:8000 -> 8080
Forwarding from [::1]:8000 -> 8080
```

We have a working Java application that is deployed kubernetes, we are able to access the URL: http://localhost:8000 and get the intended response.

The code for this project can be found at the following URL -

GitHub: https://github.com/Halen0Jacobsen/catalog-service