Practicum 1

CSCI-P 466/566 Software Engineering II

Indiana University Bloomington - Spring 2025

Spring APIs

Spring Framework

- Spring makes it easy to create Java enterprise applications.
- The Spring Framework is divided into modules. Applications can choose which modules they need.
- The core container, messaging, transactional data and persistence, web, …

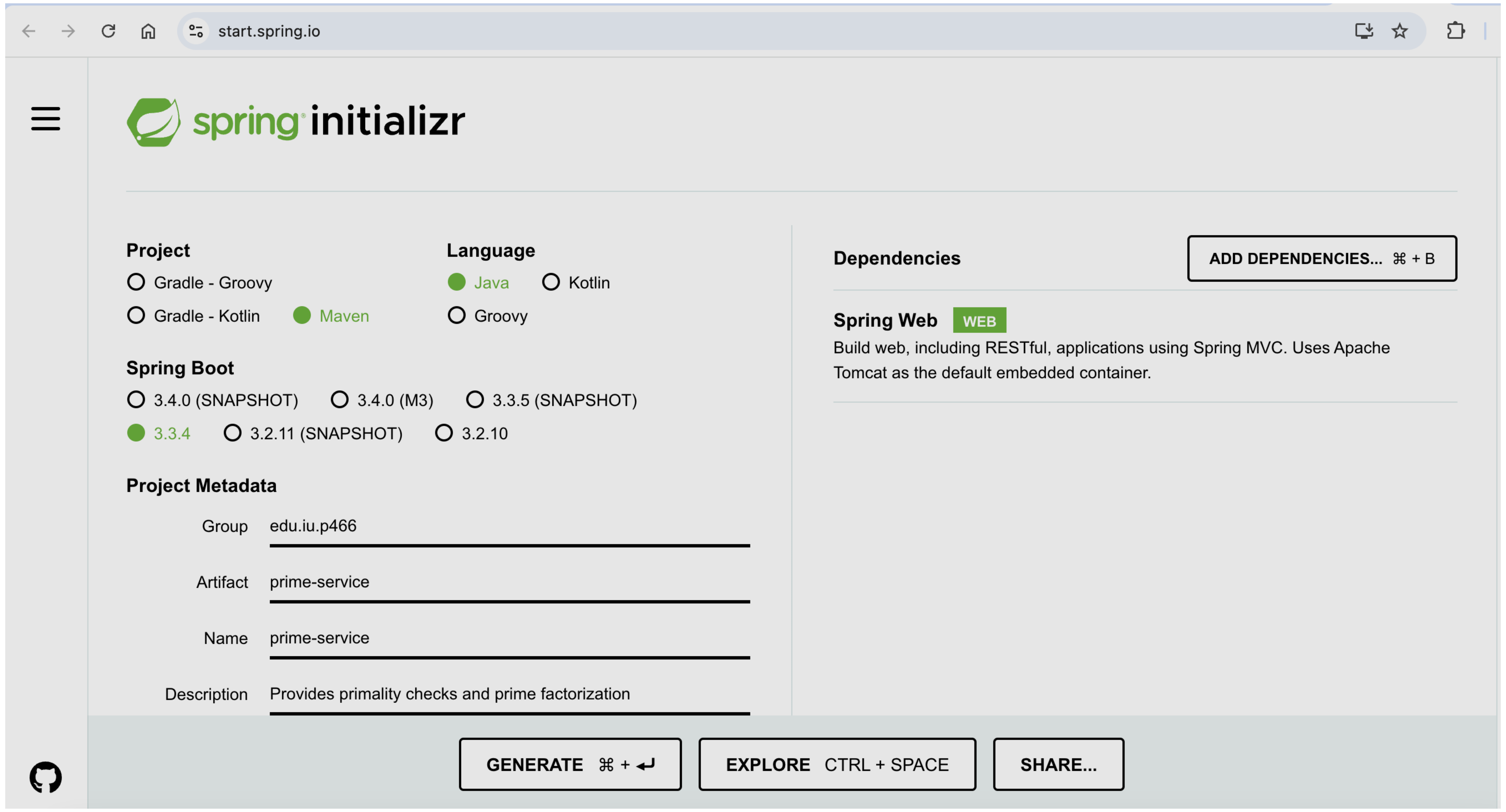
Spring Core Technologies

- Inversion of Control (IoC) container
- Resources
- Validation, Binding and Type Conversion
- Aspect-oriented Programming (AOP)
- Data Buffers
- Logging

Prime Service

Steps:

- Use start.spring.io to create the skeleton of the API.
- Open the folder in Intellij IDEA.



Project
- ○ Gradle - Groovy
- ○ Gradle - Kotlin
- ● Maven

Language
- ● Java
- ○ Kotlin
- ○ Groovy

Spring Boot
- ○ 3.4.0 (SNAPSHOT)
- ○ 3.4.0 (M3)
- ○ 3.3.5 (SNAPSHOT)
- ● 3.3.4
- ○ 3.2.11 (SNAPSHOT)
- ○ 3.2.10

Project Metadata

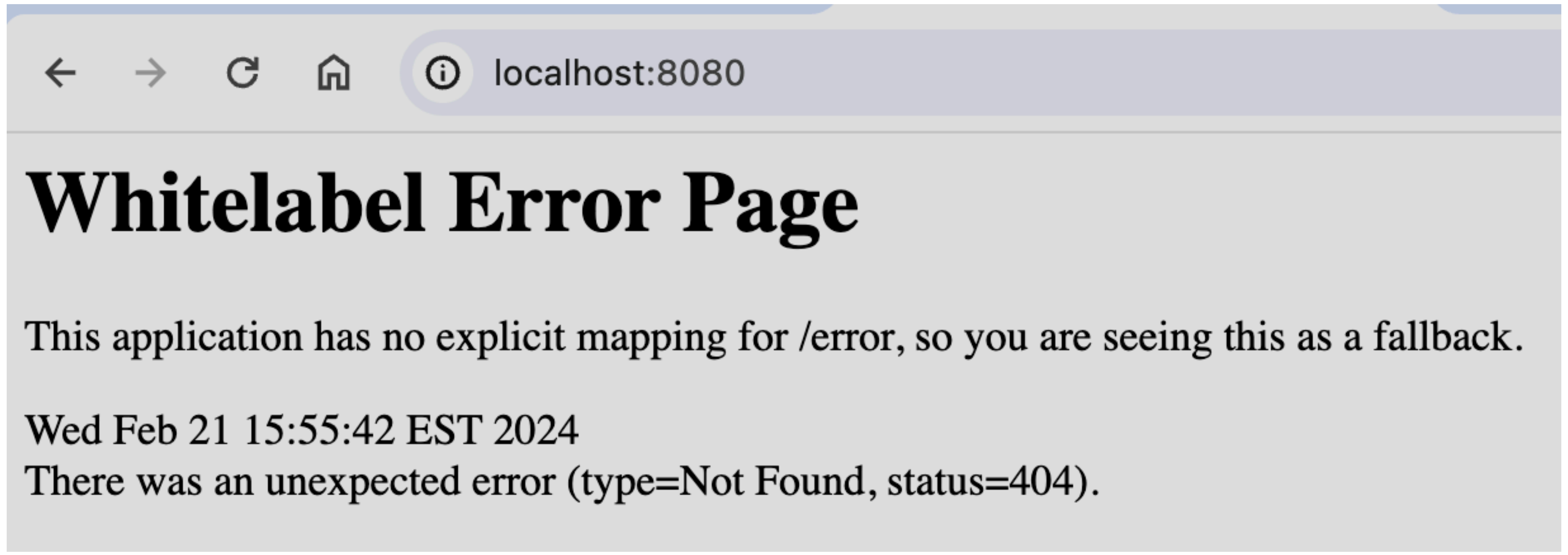
Group	edu.iu.p466
Artifact	prime-service
Name	prime-service
Description	Provides primality checks and prime factorization

Dependencies

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Steps:

- Run your service: **./mvnw spring-boot:run**
- Test your service using a browser: http://localhost:8080



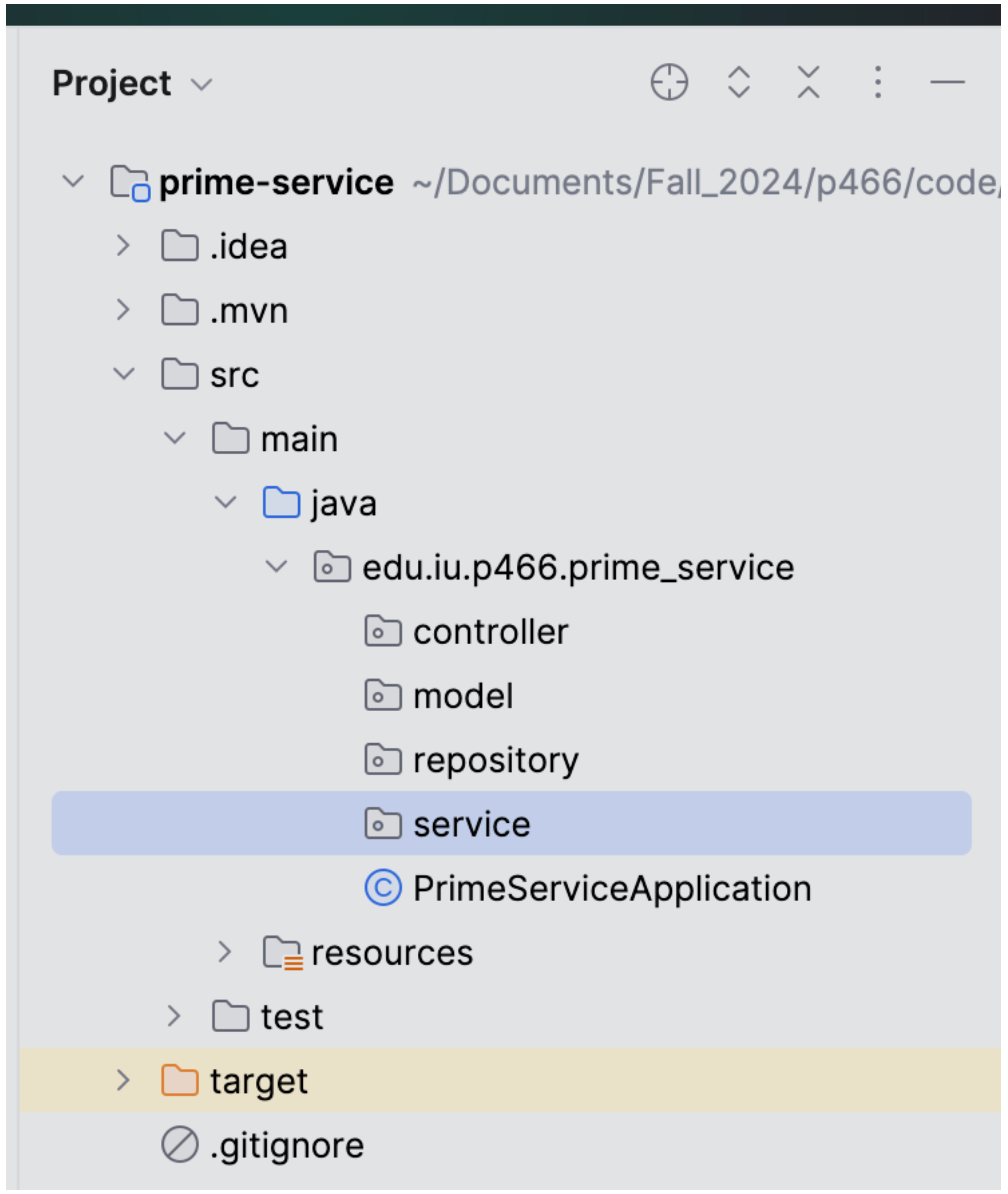
Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Wed Feb 21 15:55:42 EST 2024
There was an unexpected error (type=Not Found, status=404).

Steps:

- Create the structure of the API: controller, service, model, repository:

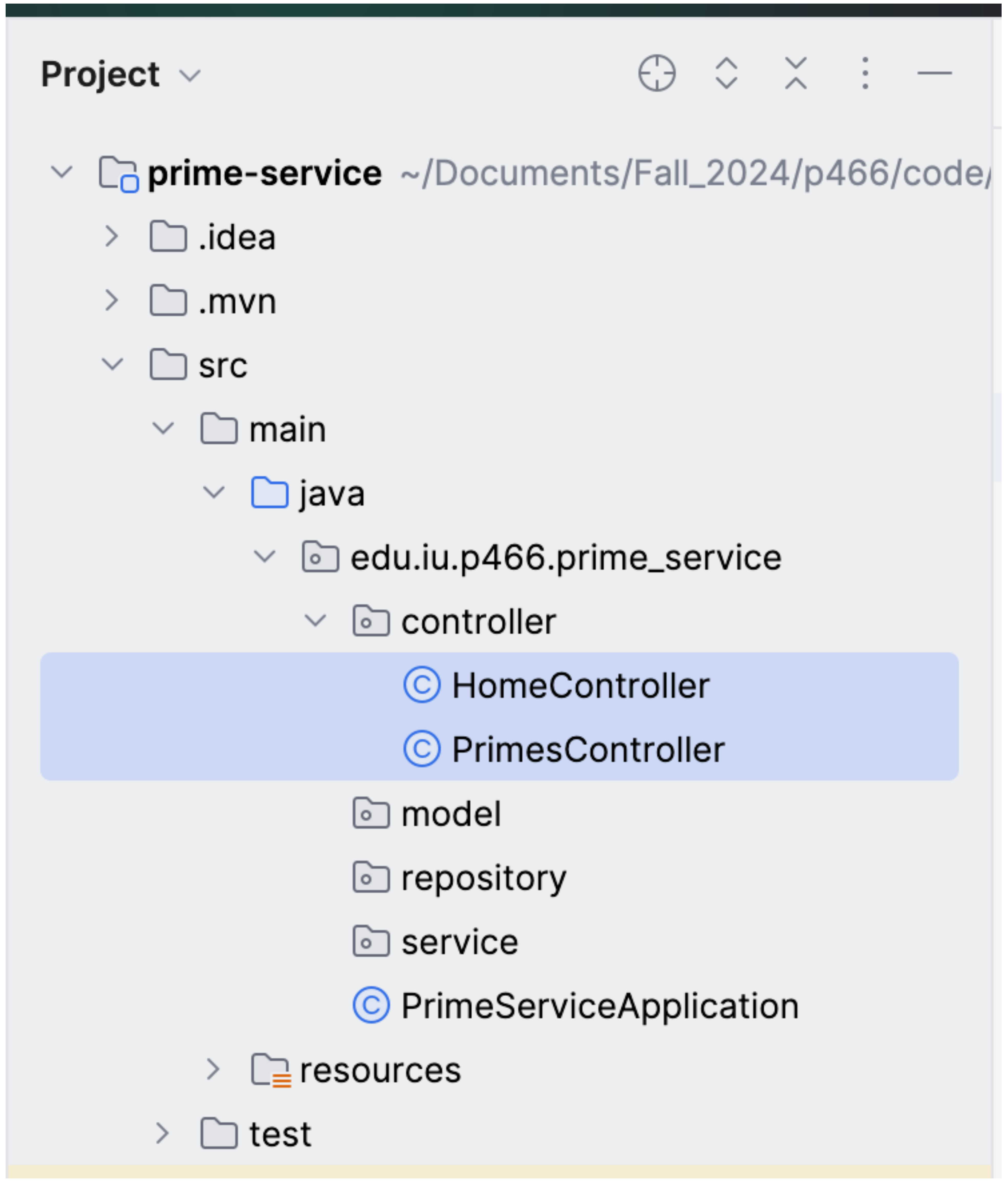


Project

```
prime-service ~/Documents/Fall_2024/p466/code
  .idea
  .mvn
  src
    main
      java
        edu.iu.p466.prime_service
          controller
          model
          repository
          service
          PrimeServiceApplication
      resources
    test
  target
  .gitignore
```

Steps:

- Add the controllers:

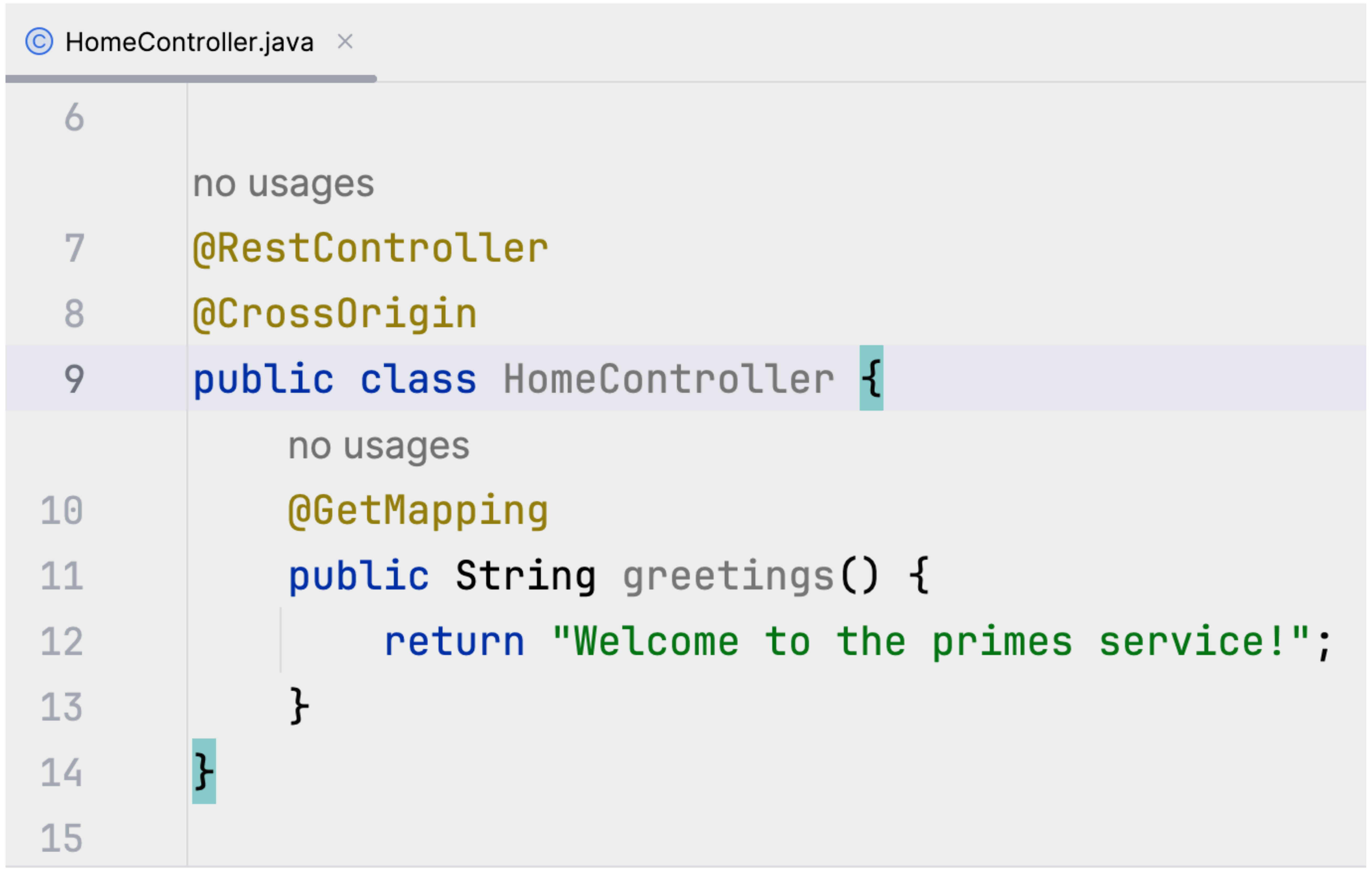


Project

```
prime-service ~/Documents/Fall_2024/p466/code/
  .idea
  .mvn
  src
    main
      java
        edu.iu.p466.prime_service
          controller
            HomeController
            PrimesController
          model
          repository
          service
          PrimeServiceApplication
      resources
    test
```

Steps:

- Add the greetings endpoint to the HomeController:


HomeController.java

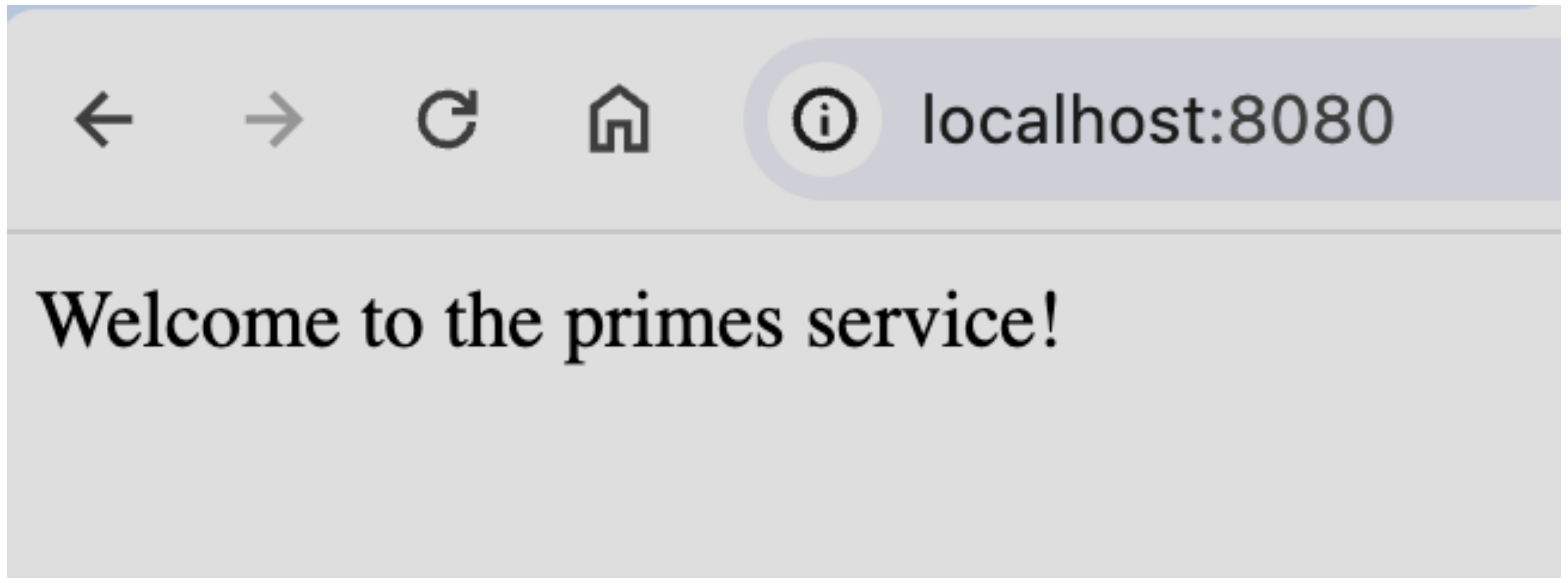
```
no usages
@RestController
@CrossOrigin
public class HomeController {
    no usages
    @GetMapping
    public String greetings() {
        return "Welcome to the primes service!";
    }
}
```

Steps:

- Rerun the application: stop using CTRL+C and run using

  ./mvnw spring-boot:run

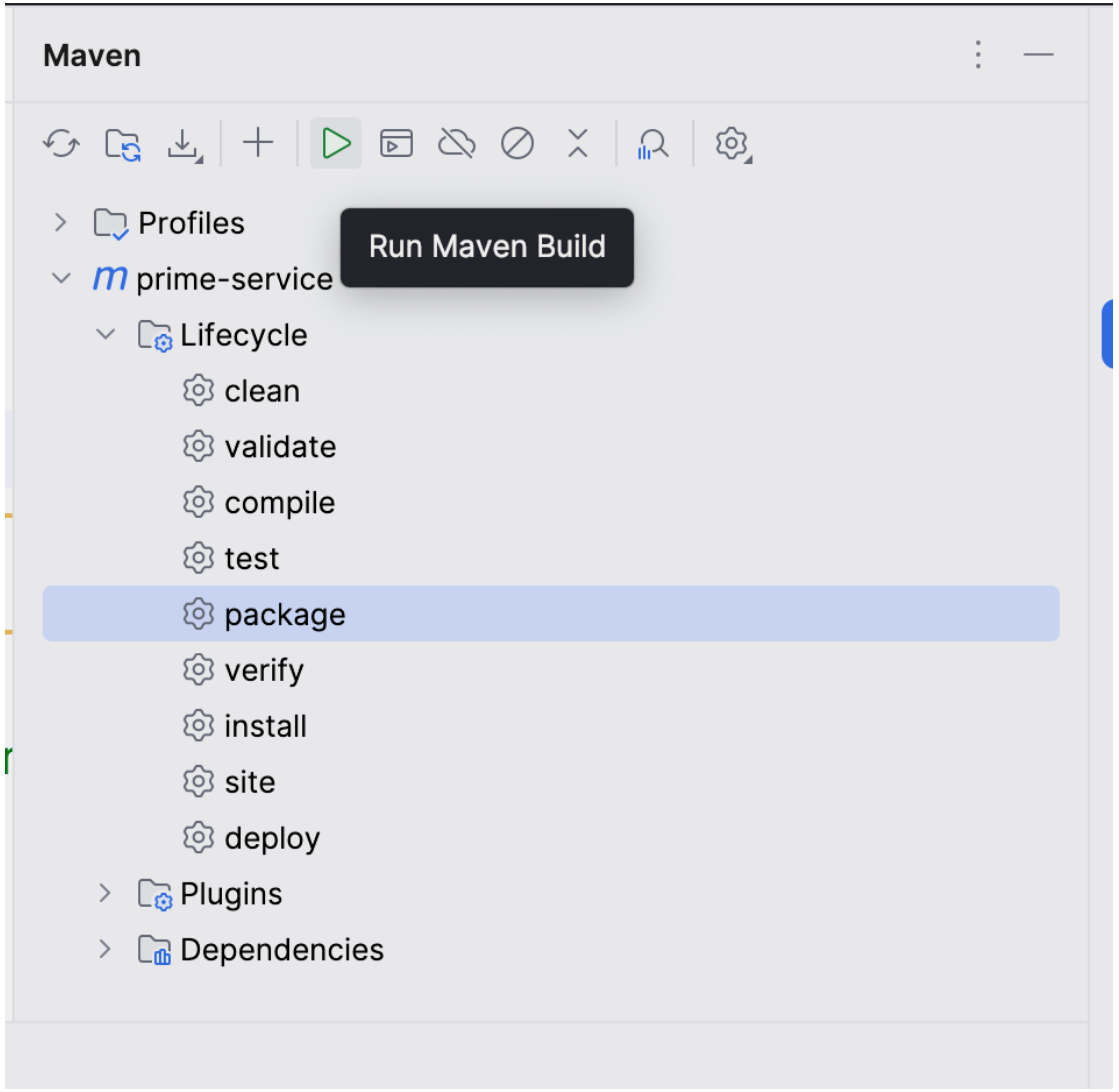
- Test the greetings endpoint.



Welcome to the primes service!

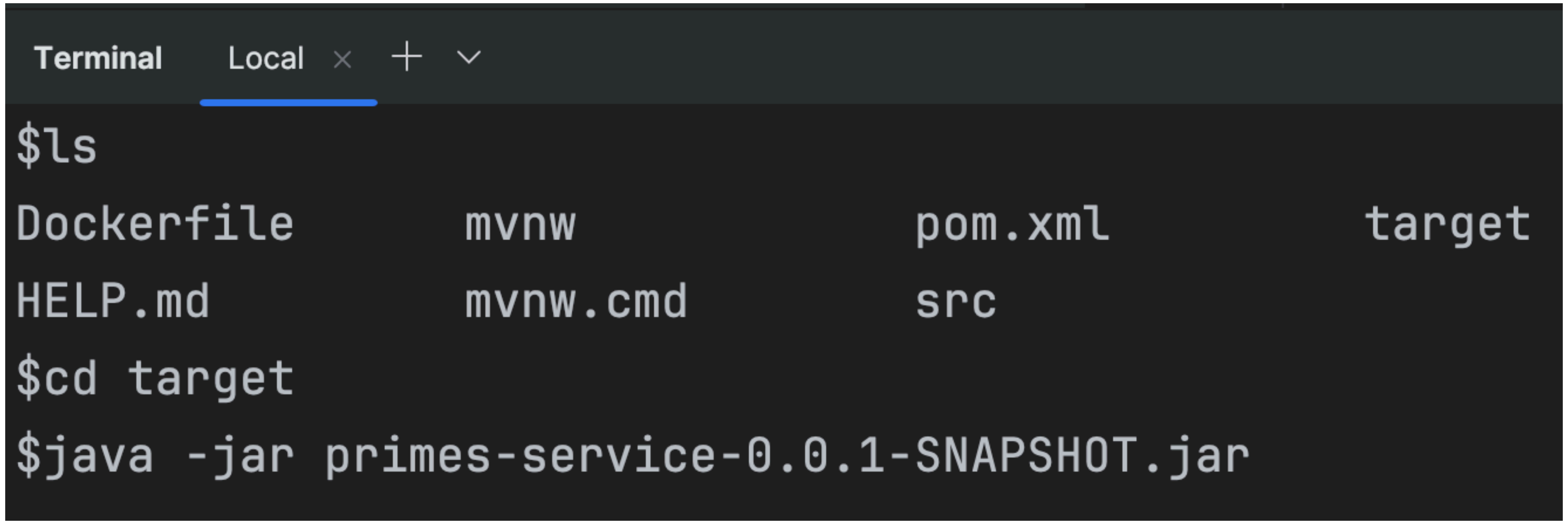
Steps:

- Now let’s run the application directly using java commands and not maven commands.
- Stop the application using CTRL+C
- First we need to create the .jar file. To do so go to the maven panel on the right and run the package step. This should create a jar file under your “target” folder.

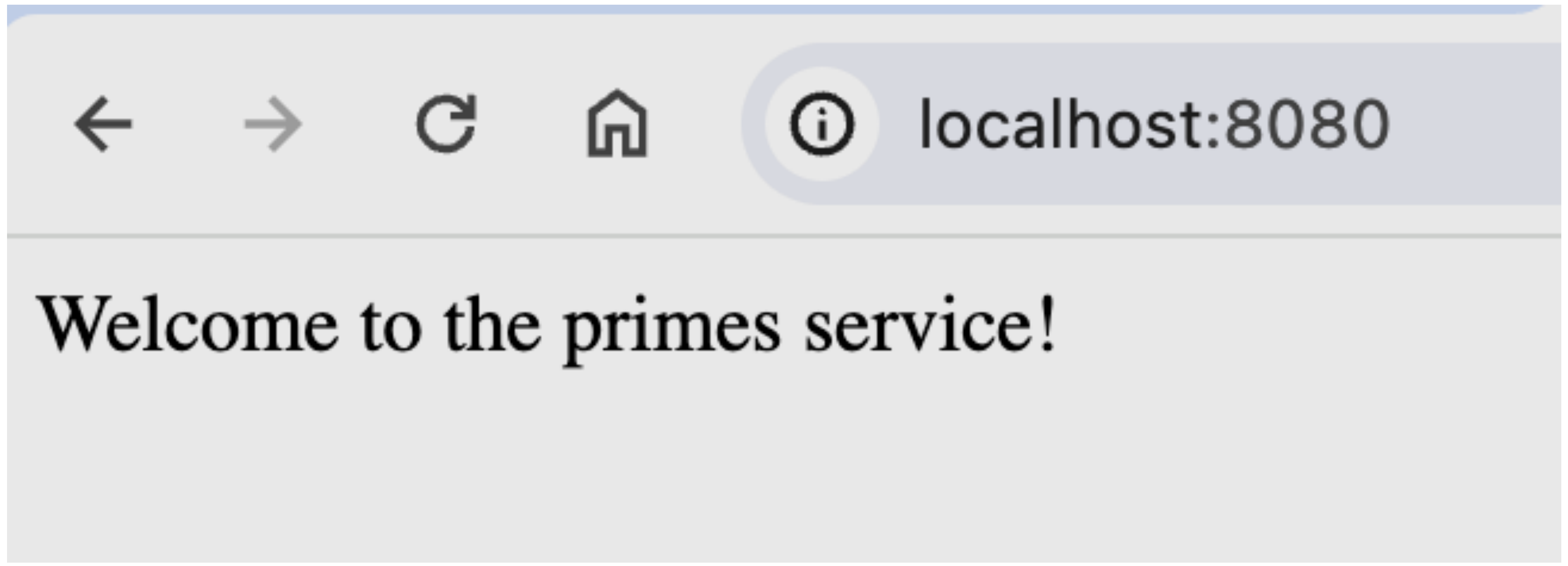


Steps:

- **cd target**
- Run the jar file.
- Verify that it is working.



```
$ls
Dockerfile      mvnw            pom.xml         target
HELP.md         mvnw.cmd        src
$cd target
$java -jar primes-service-0.0.1-SNAPSHOT.jar
```



localhost:8080

Welcome to the primes service!

Steps:

- In your P566-466 organization on github.com, create a private repository on Github call it **practicum1**.
- Run the following commands to initialize a local git repository and connect it to your remote repository on Github (**first make sure you are in the root directory**):
- **git init**
- **git remote add origin https://github.com/YourOrganizationname/practicum1.git**
- Then add, commit and push the changes:
- **git add .**
- **git commit -m “initial setup”**
- **git push --set-upstream origin master**

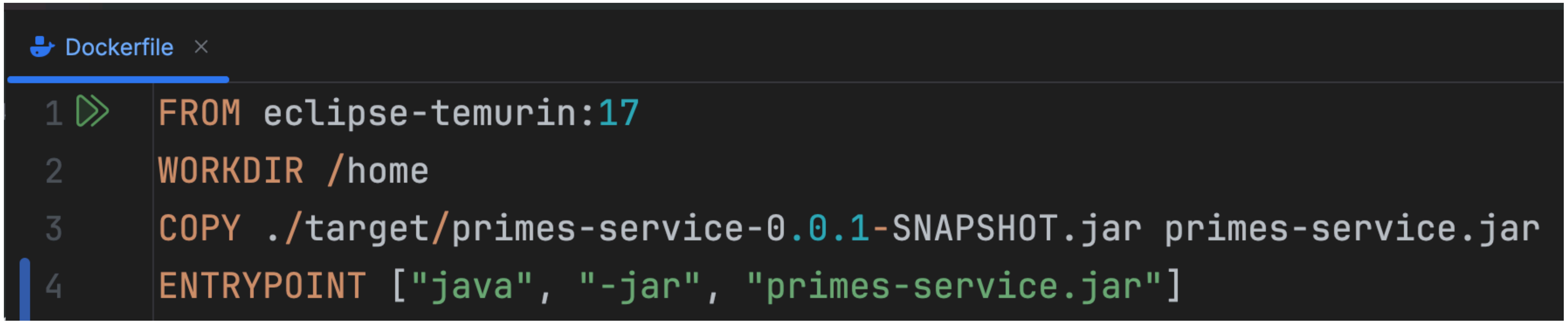
Steps:

- Now the setup of the local and the remote repository is complete.
- Every now and then, when you have some changes completed and tested, push your changes to the remote repository.
- You can do so by running the following commands:
- **git add .**
- **git commit -m “initial setup”**
- **git push --set-upstream origin master**

Containerize the service

Steps

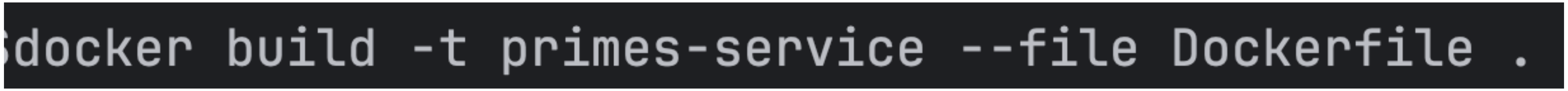
- Create a Dockerfile with the following content:



```
FROM eclipse-temurin:17
WORKDIR /home
COPY ./target/primes-service-0.0.1-SNAPSHOT.jar primes-service.jar
ENTRYPOINT ["java", "-jar", "primes-service.jar"]
```

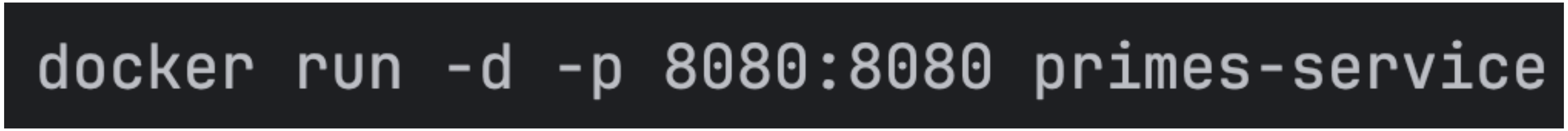
Steps

- Build the docker image:



```
$docker build -t primes-service --file Dockerfile .
```

- Run the docker image:



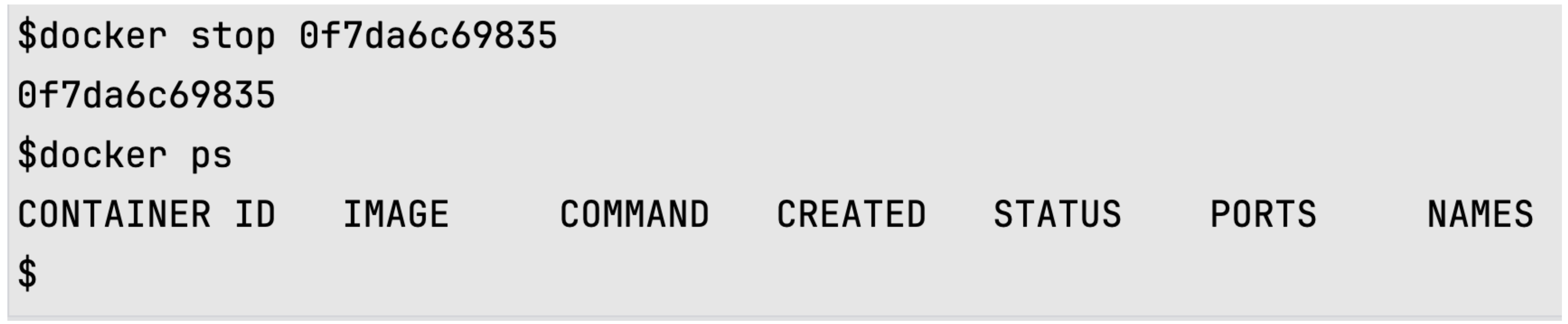
```
docker run -d -p 8080:8080 primes-service
```

- To see the list of the running containers run:
- **docker ps**

```
$docker ps
CONTAINER ID   IMAGE           COMMAND                  CREATED              STATUS              PORTS
            NAMES
0f7da6c69835   primes-service   "java -jar primes-se…"   About a minute ago   Up About a minute   0.0.0.0:8080-
```

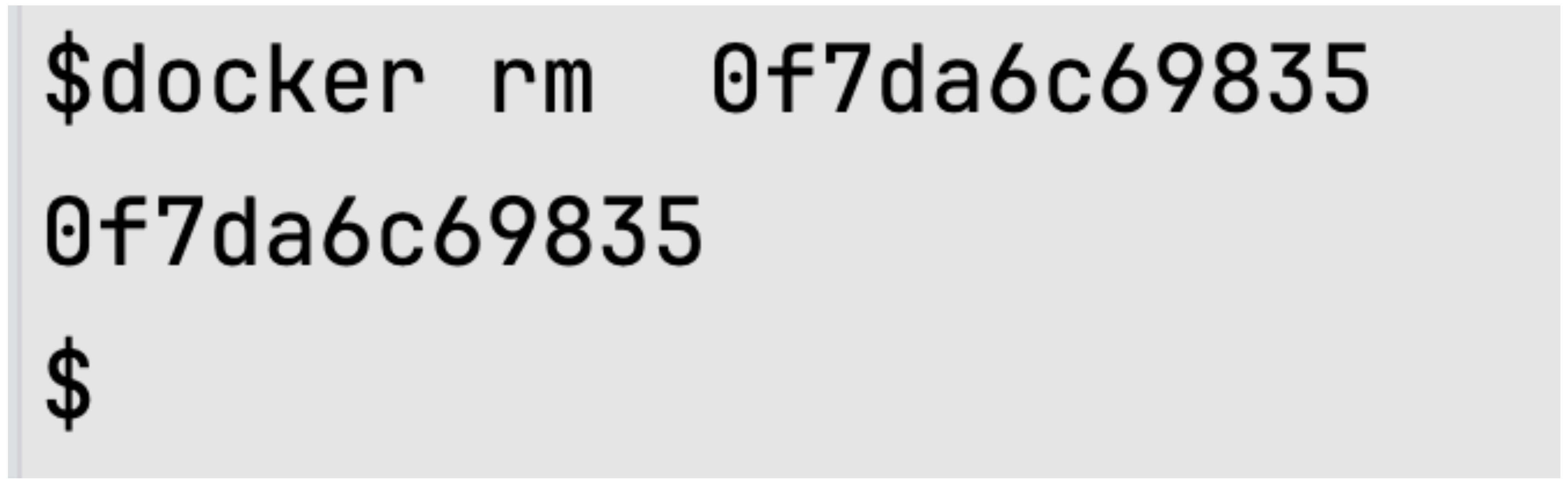
- To see the list of all containers (both running and not running containers):
- **docker ps -a**
- To see the list of the images:
- **docker images**

- To stop a running container, you can use the following command and provide either the container name or container ID.
- **docker stop CNAME-OR-ID**



```
$docker stop 0f7da6c69835
0f7da6c69835
$docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
$
```

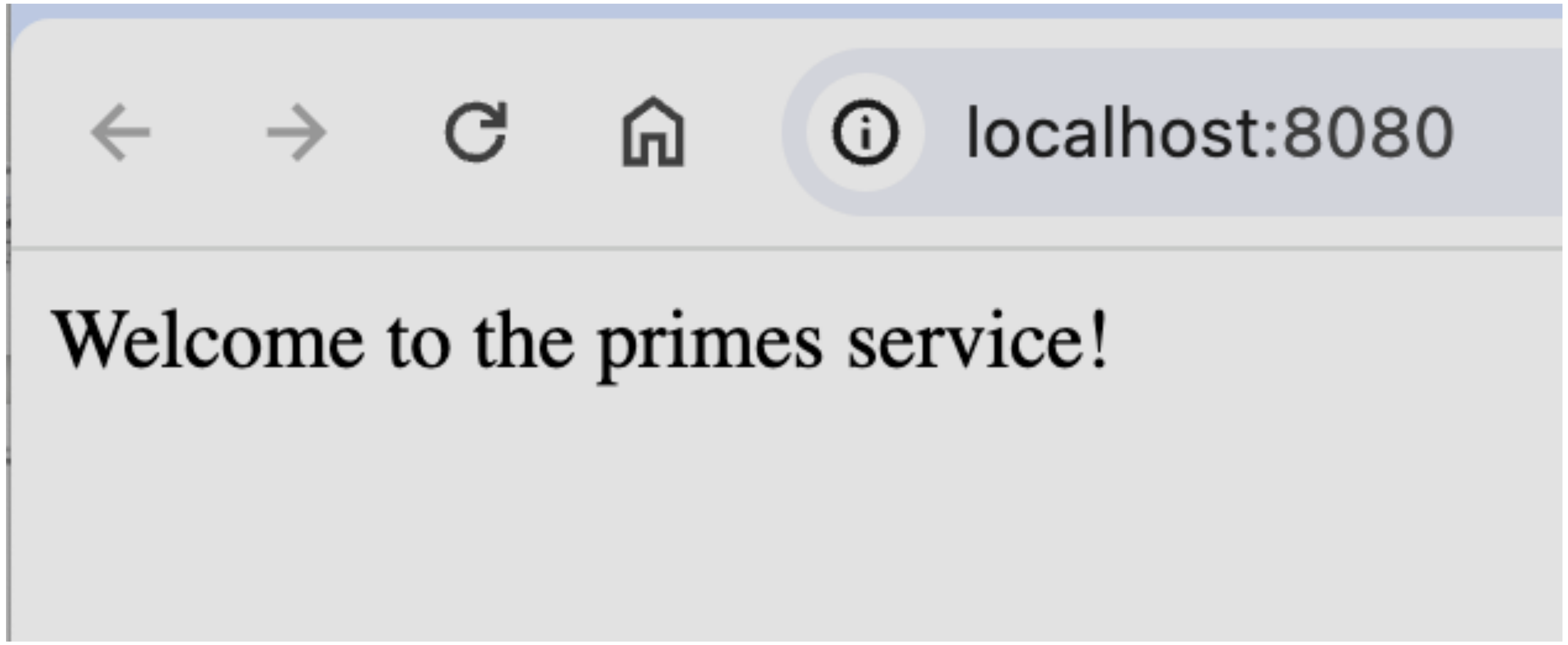
- To remove a running container:
- **docker rm CNAME-OR-ID**



```
$docker rm  0f7da6c69835
0f7da6c69835
$
```

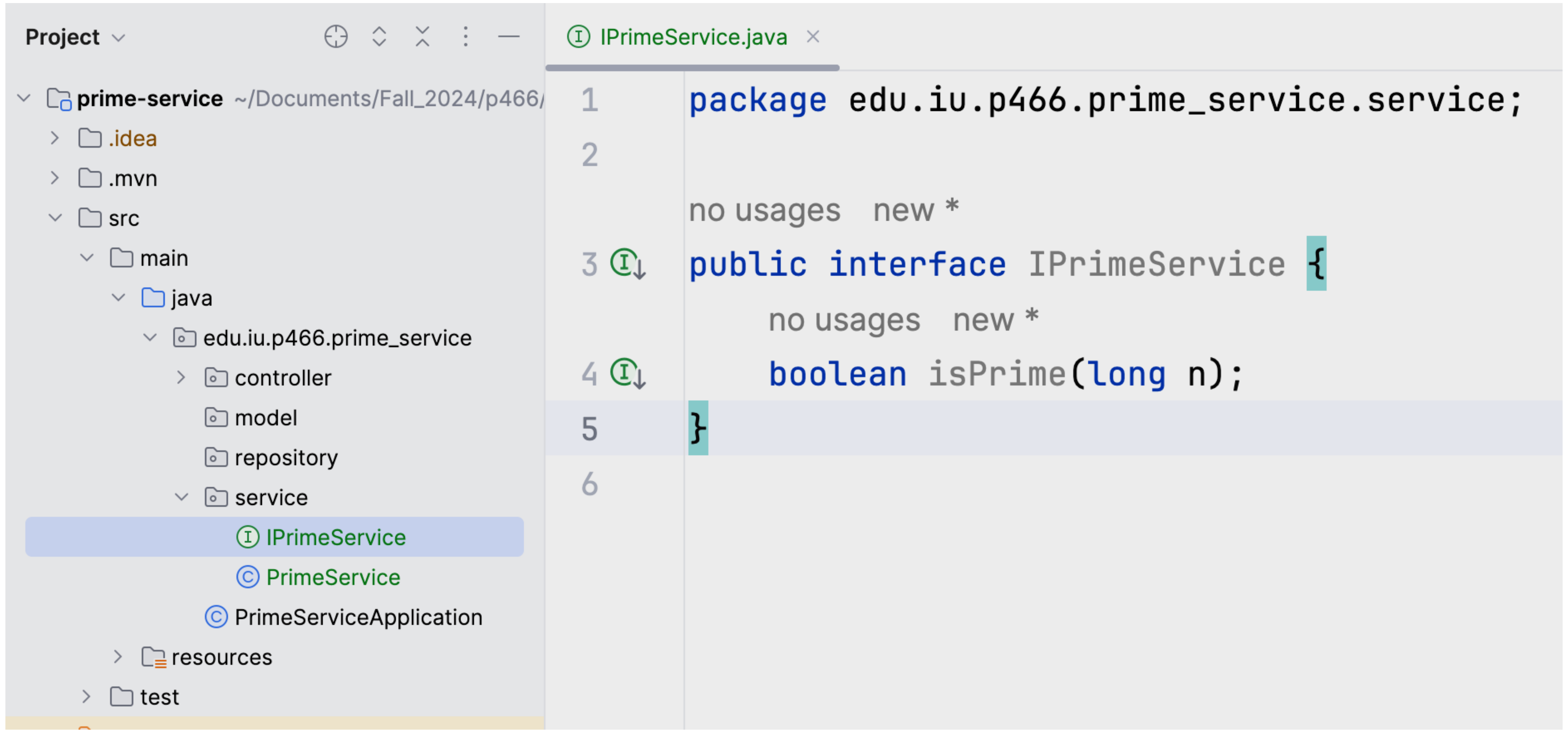
- To remove all containers on your development machine:
- **docker rm -v -f $(docker ps -qa)**
- To remove all the images on your development machine:
- **docker image remove -f $(docker images -a -q)**

- Verify that it is working.



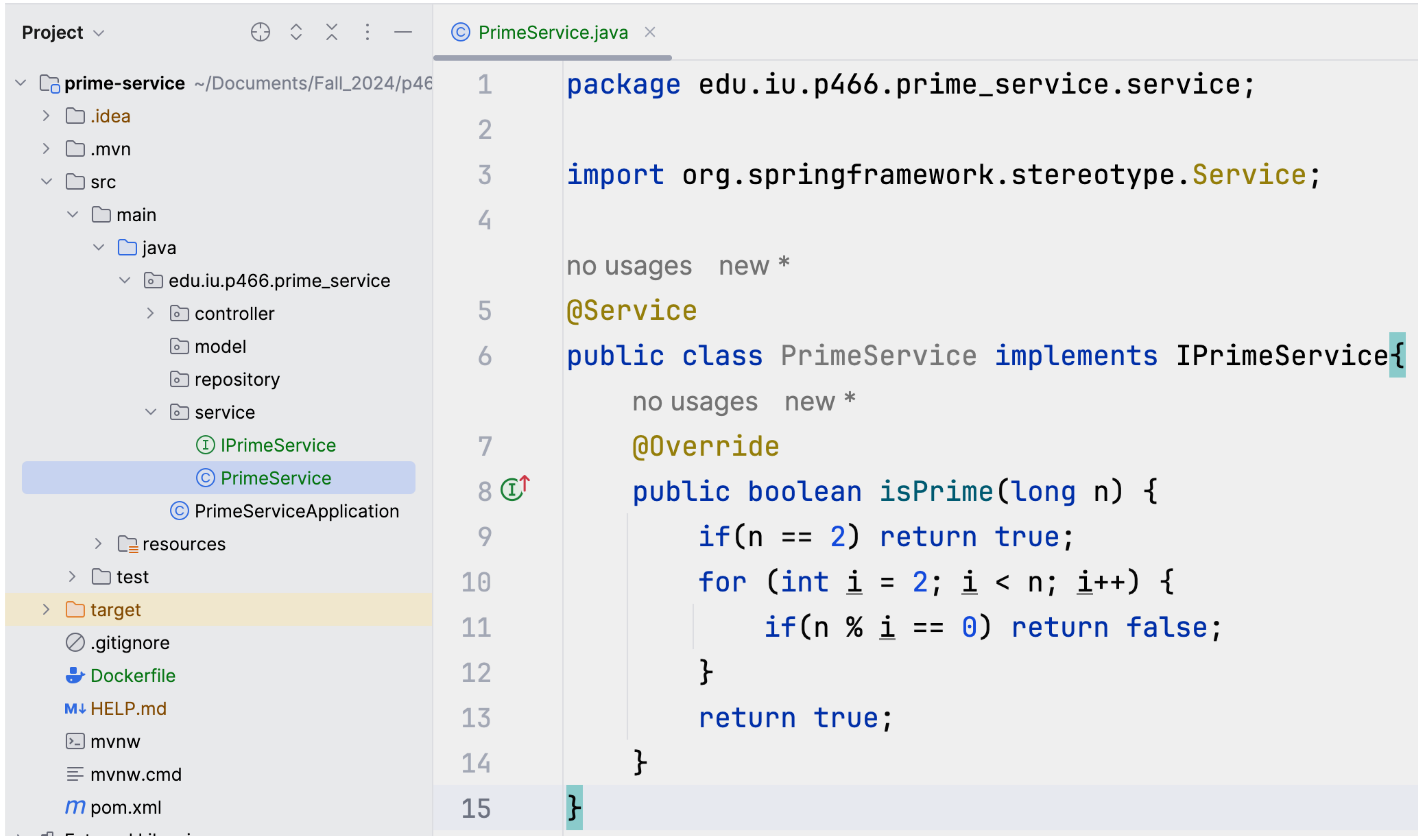
Welcome to the primes service!

- Add the service class with a method determining if a given number is prime or not.



```java
package edu.iu.p466.prime_service.service;

no usages   new *
public interface IPrimeService {
    no usages   new *
    boolean isPrime(long n);
}

```

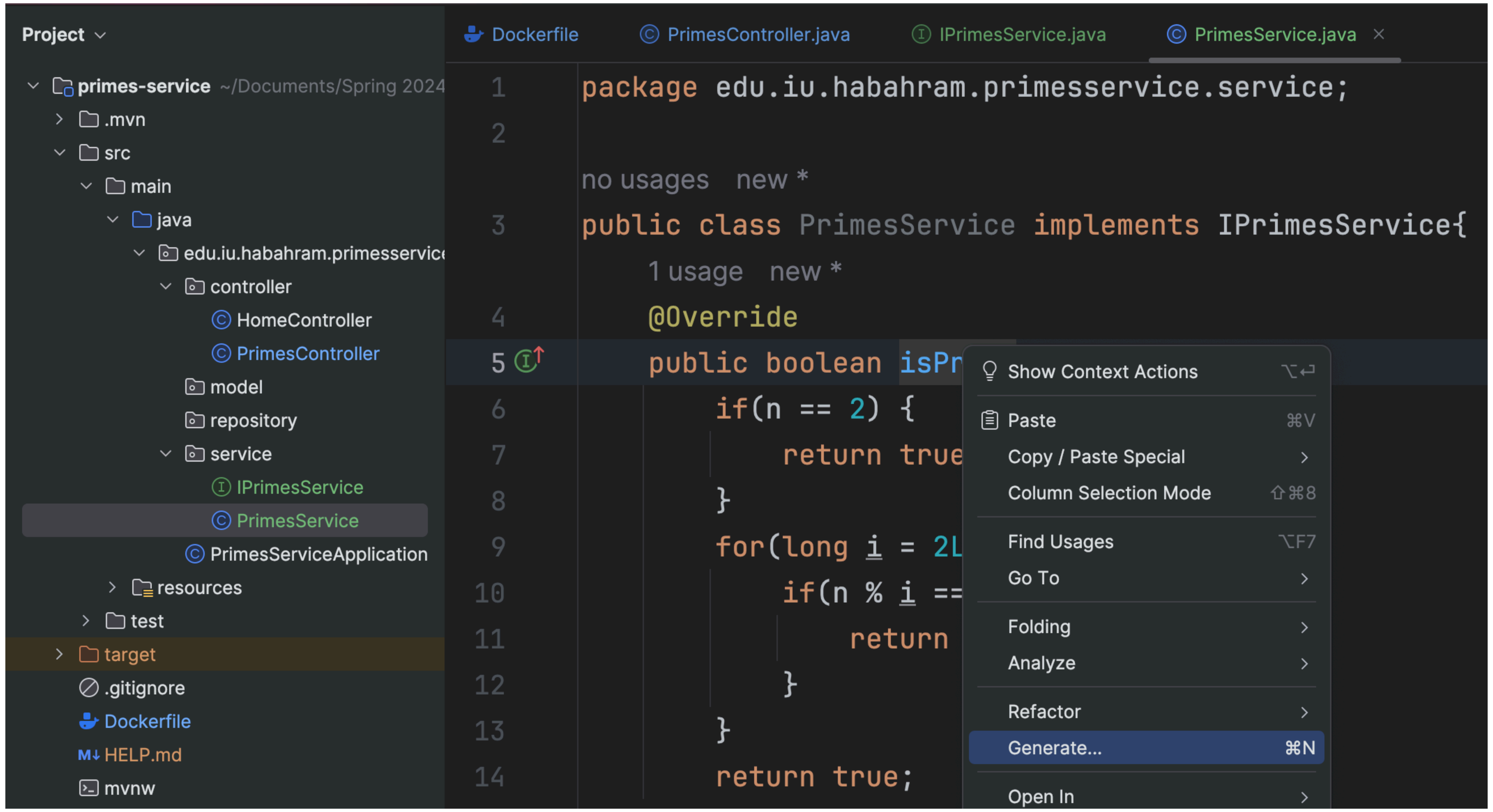
- Add the service class with a method determining if a given number is prime or not.



```java
package edu.iu.p466.prime_service.service;

import org.springframework.stereotype.Service;

no usages   new *
@Service
public class PrimeService implements IPrimeService{
    no usages   new *
    @Override
    public boolean isPrime(long n) {
        if(n == 2) return true;
        for (int i = 2; i < n; i++) {
            if(n % i == 0) return false;
        }
        return true;
    }
}
```

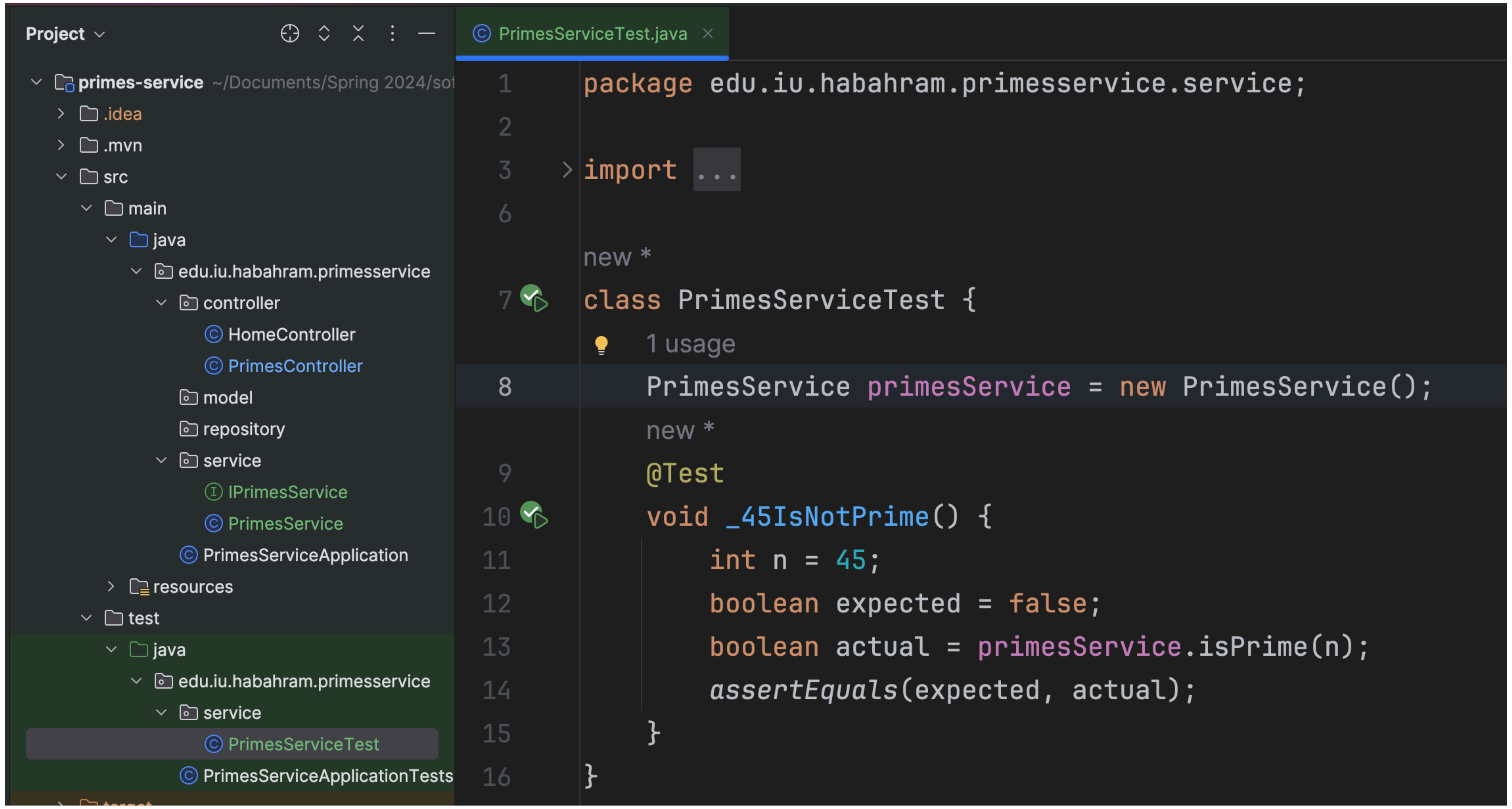
Unit testing using junit

- Unit test the isPrime function using junit.
- Right click inside the function, select "Generate" and then select "Test" and then select the isPrime function from the list of the functions. Make sure the testing library is "junit5".



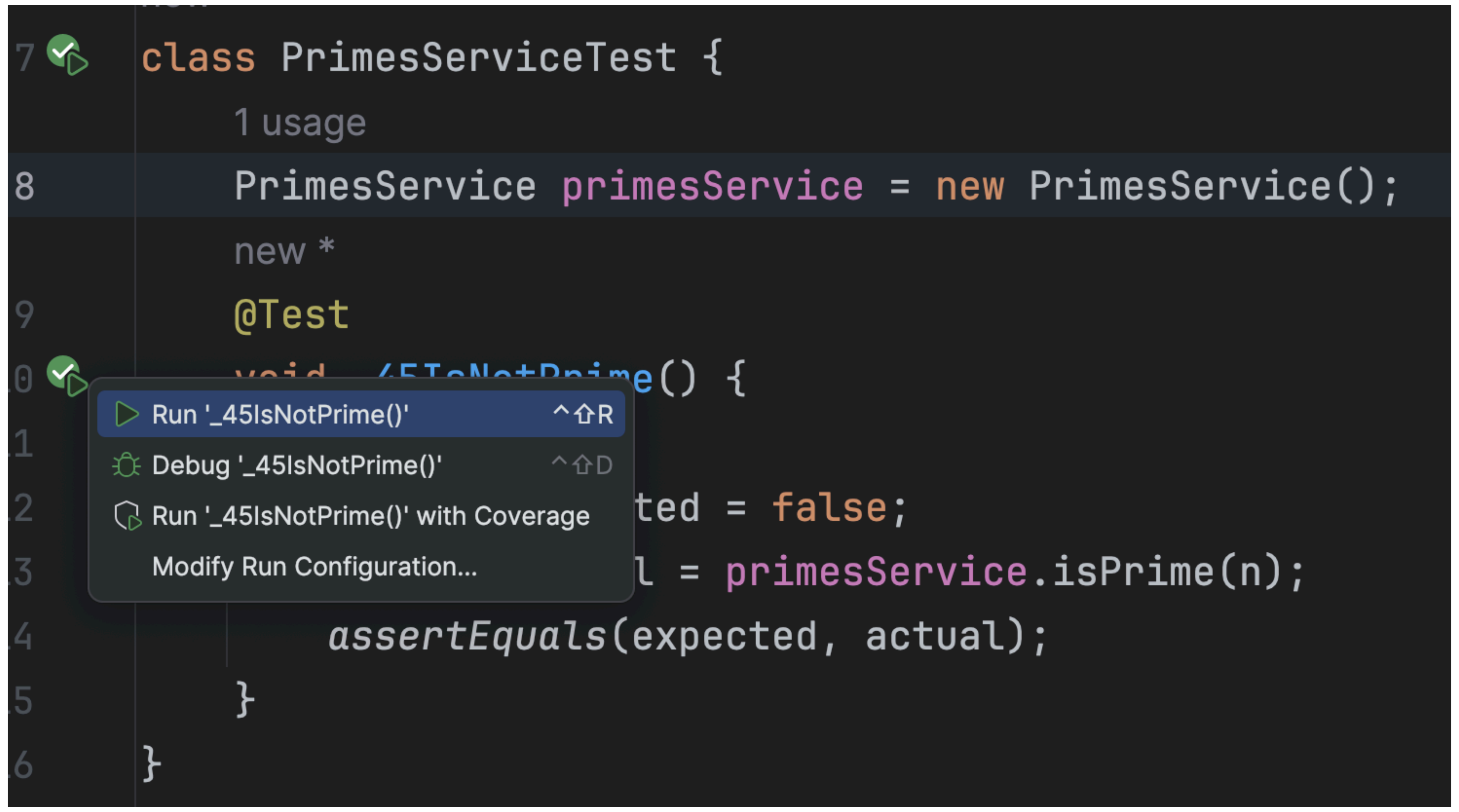
```
package edu.iu.habahram.primesservice.service;

no usages   new *
public class PrimesService implements IPrimesService{
    1 usage   new *
    @Override
    public boolean isPr
        if(n == 2) {
            return true
        }
        for(long i = 2L
            if(n % i ==
                return
            }
        }
        return true;
```

- Add a unit test:



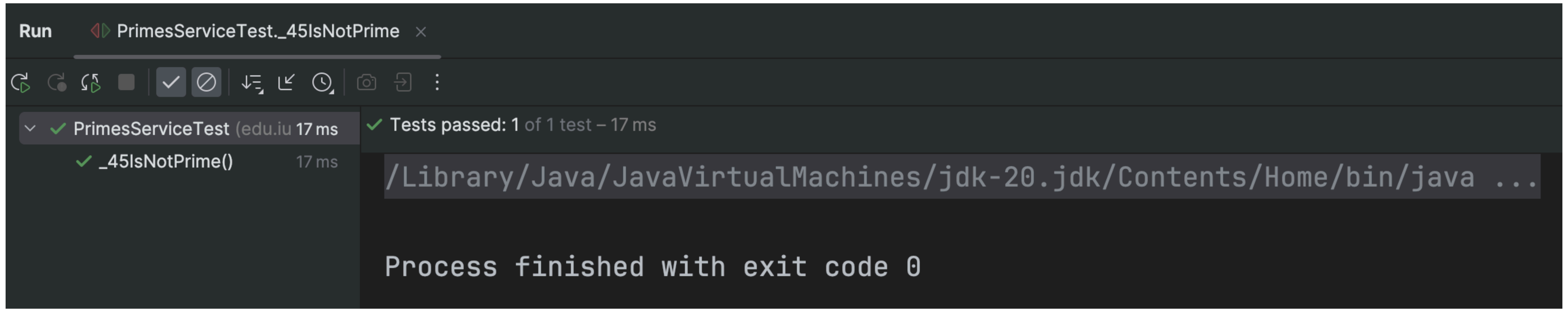
```java
package edu.iu.habahram.primesservice.service;

import ...

class PrimesServiceTest {
    PrimesService primesService = new PrimesService();
    @Test
    void _45IsNotPrime() {
        int n = 45;
        boolean expected = false;
        boolean actual = primesService.isPrime(n);
        assertEquals(expected, actual);
    }
}
```

- Run the unit test:



```
class PrimesServiceTest {
    1 usage
    PrimesService primesService = new PrimesService();
    new *
    @Test
    void _45IsNotPrime() {
                          ted = false;
                          l = primesService.isPrime(n);
        assertEquals(expected, actual);
    }
}
```

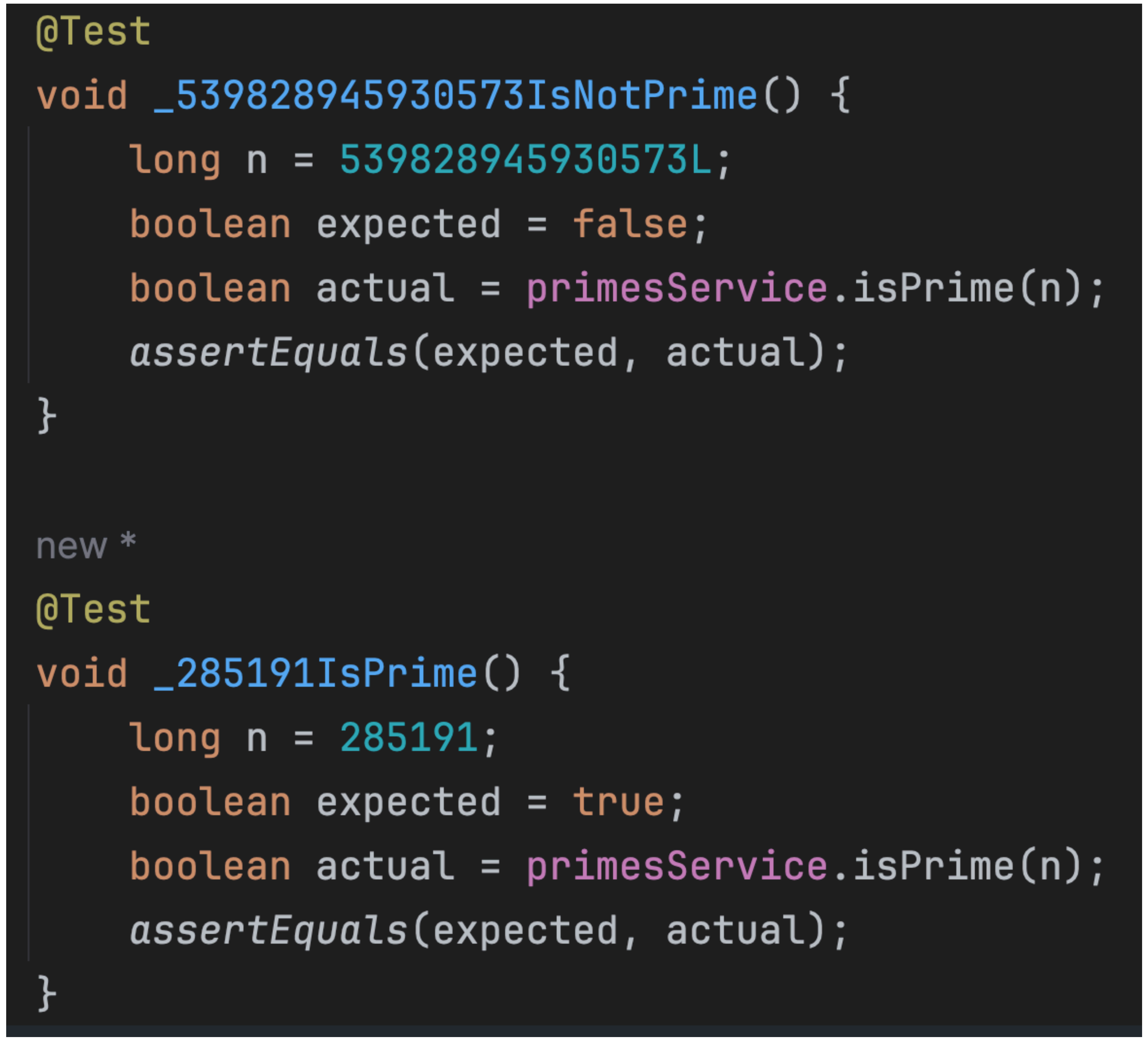
Run '_45IsNotPrime()' ^⇧R
Debug '_45IsNotPrime()' ^⇧D
Run '_45IsNotPrime()' with Coverage
Modify Run Configuration...



Run
PrimesServiceTest._45IsNotPrime
PrimesServiceTest (edu.iu 17 ms
_45IsNotPrime() 17 ms

```
✓ Tests passed: 1 of 1 test – 17 ms
/Library/Java/JavaVirtualMachines/jdk-20.jdk/Contents/Home/bin/java ...

Process finished with exit code 0
```

- Add some more unit tests:

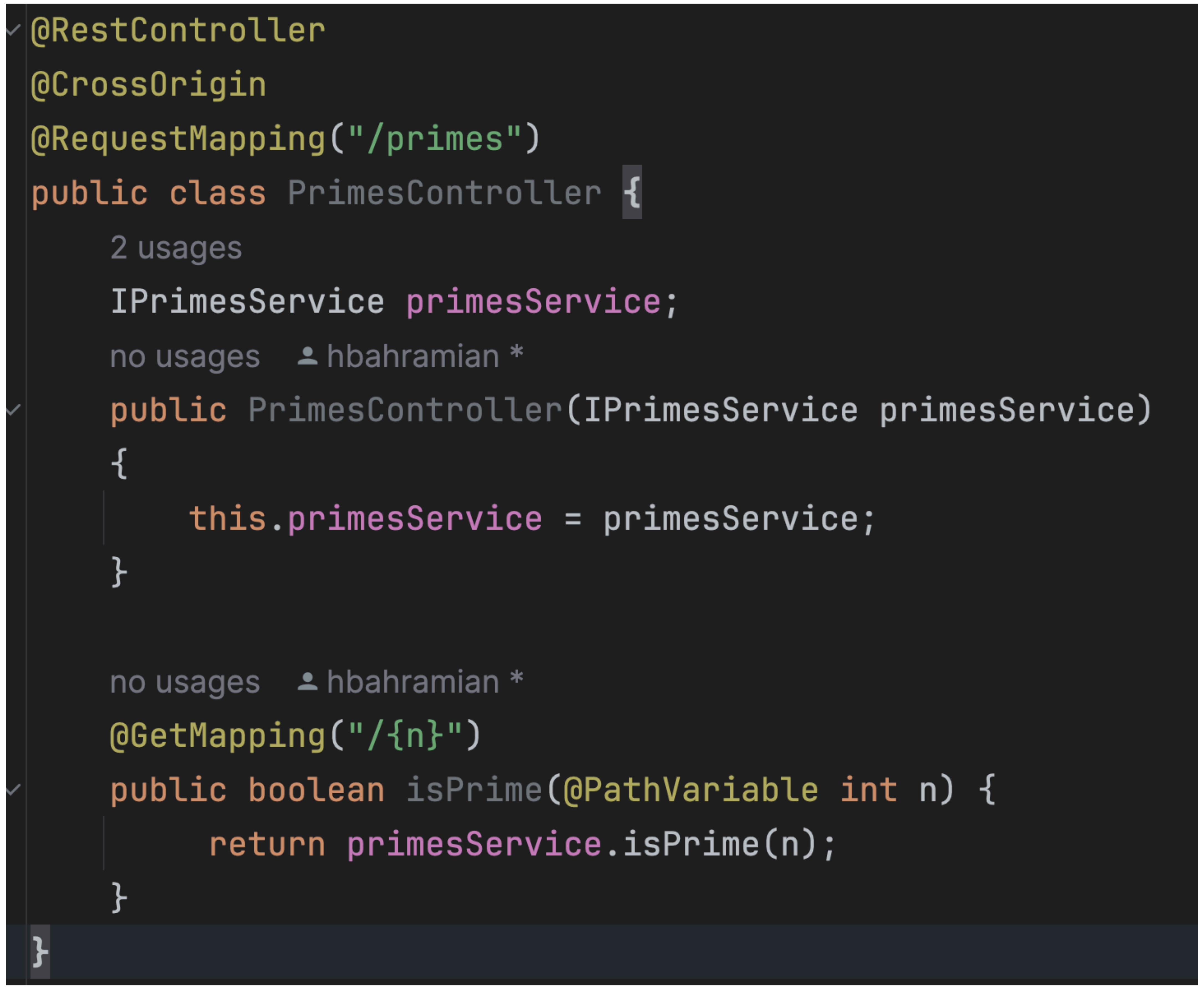


```
@Test
void _539828945930573IsNotPrime() {
    long n = 539828945930573L;
    boolean expected = false;
    boolean actual = primesService.isPrime(n);
    assertEquals(expected, actual);
}

new *
@Test
void _285191IsPrime() {
    long n = 285191;
    boolean expected = true;
    boolean actual = primesService.isPrime(n);
    assertEquals(expected, actual);
}
```

Add the primality check endpoint

Steps

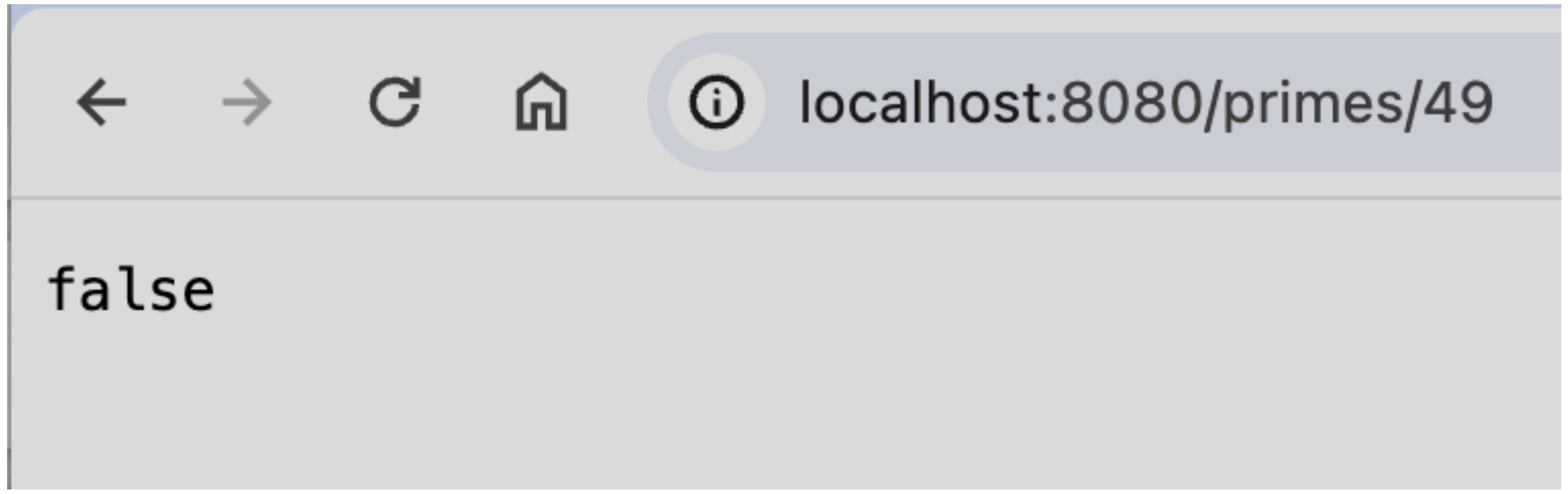
- Add an endpoint that exposes the prime check functionality:



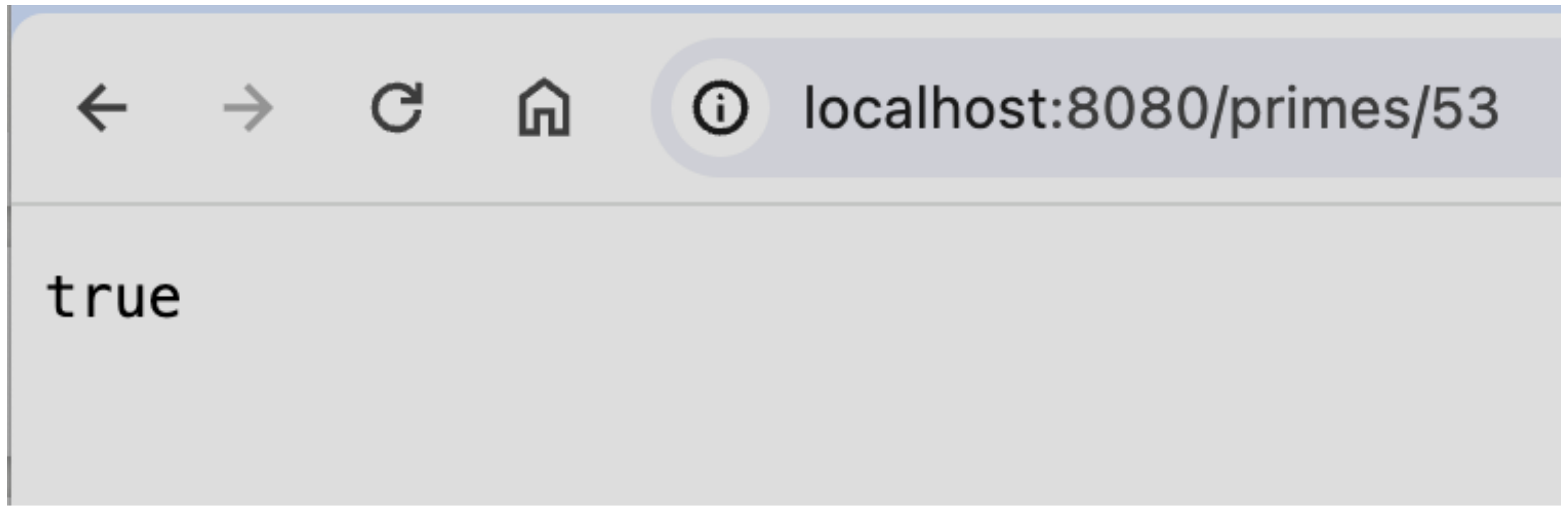
```
@RestController
@CrossOrigin
@RequestMapping("/primes")
public class PrimesController {
    2 usages
    IPrimesService primesService;
    no usages  hbahramian *
    public PrimesController(IPrimesService primesService)
    {
        this.primesService = primesService;
    }

    no usages  hbahramian *
    @GetMapping("/{n}")
    public boolean isPrime(@PathVariable int n) {
        return primesService.isPrime(n);
    }
}
```

Steps

- Test the endpoint:



```
false
```



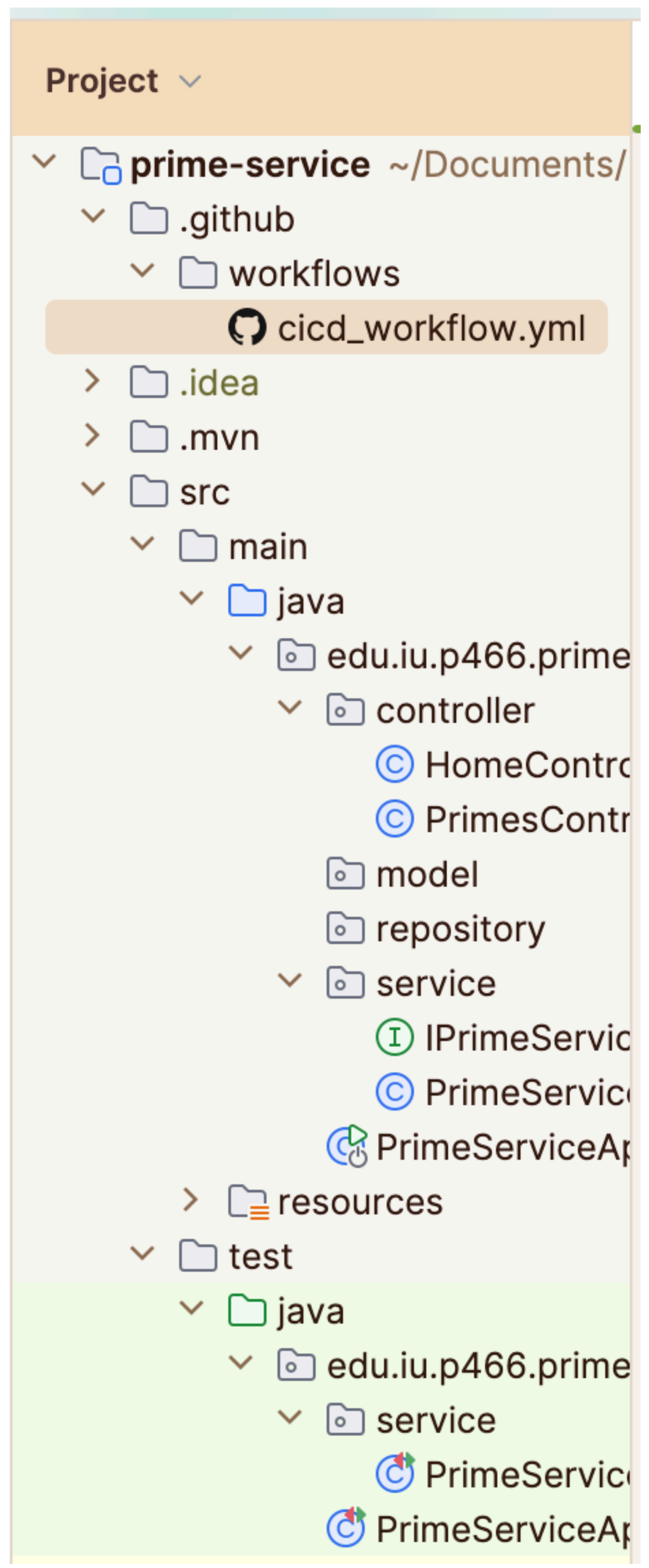
```
true
```

CI/CD pipeline

Steps

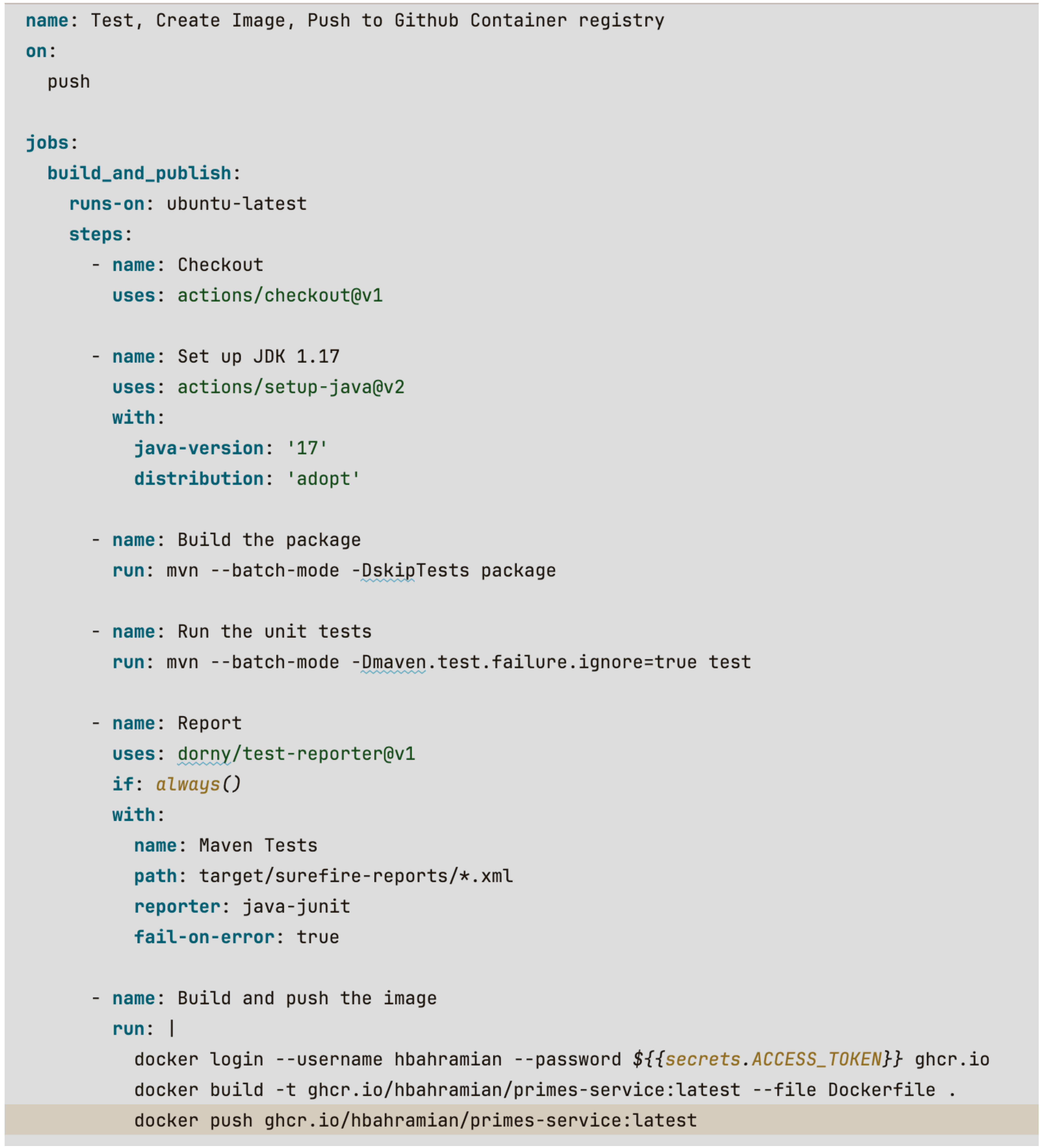
- Create a repository secret called ACCESS_TOKEN in your primes-service repository.
- Also give write permissions to workflows: Settings -> Actions -> General -> Workflow permissions -> “Read and Write Permissions”.

- Add a CI/CD pipeline with steps to automate unit testing and docker image creation.
- https://gist.github.com/hbahramian/006c3fded1c4d0b551b02a1350c54c9f



- Add a CI/CD pipeline with steps to automate unit testing and docker image creation.

- https://gist.github.com/hbahramian/006c3fded1c4d0b551b02a1350c54c9f

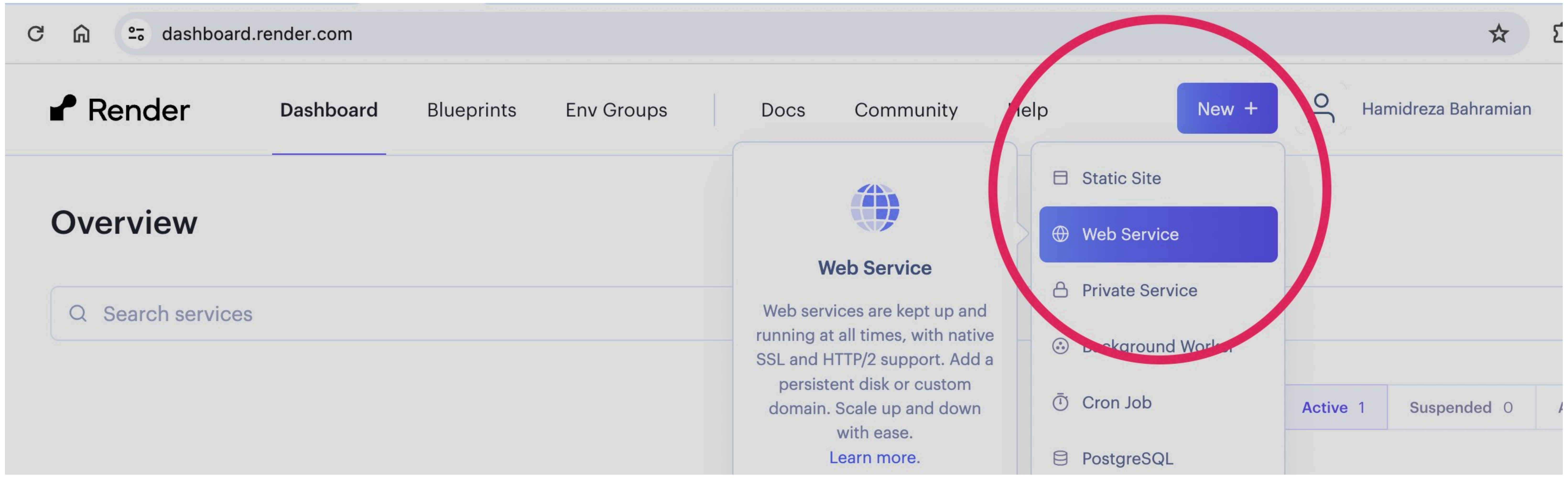


```yaml
name: Test, Create Image, Push to Github Container registry
on:
  push

jobs:
  build_and_publish:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v1

      - name: Set up JDK 1.17
        uses: actions/setup-java@v2
        with:
          java-version: '17'
          distribution: 'adopt'

      - name: Build the package
        run: mvn --batch-mode -DskipTests package

      - name: Run the unit tests
        run: mvn --batch-mode -Dmaven.test.failure.ignore=true test

      - name: Report
        uses: dorny/test-reporter@v1
        if: always()
        with:
          name: Maven Tests
          path: target/surefire-reports/*.xml
          reporter: java-junit
          fail-on-error: true

      - name: Build and push the image
        run: |
          docker login --username hbahramian --password ${{secrets.ACCESS_TOKEN}} ghcr.io
          docker build -t ghcr.io/hbahramian/primes-service:latest --file Dockerfile .
          docker push ghcr.io/hbahramian/primes-service:latest
```

Deploy the container on [render.com]

Steps

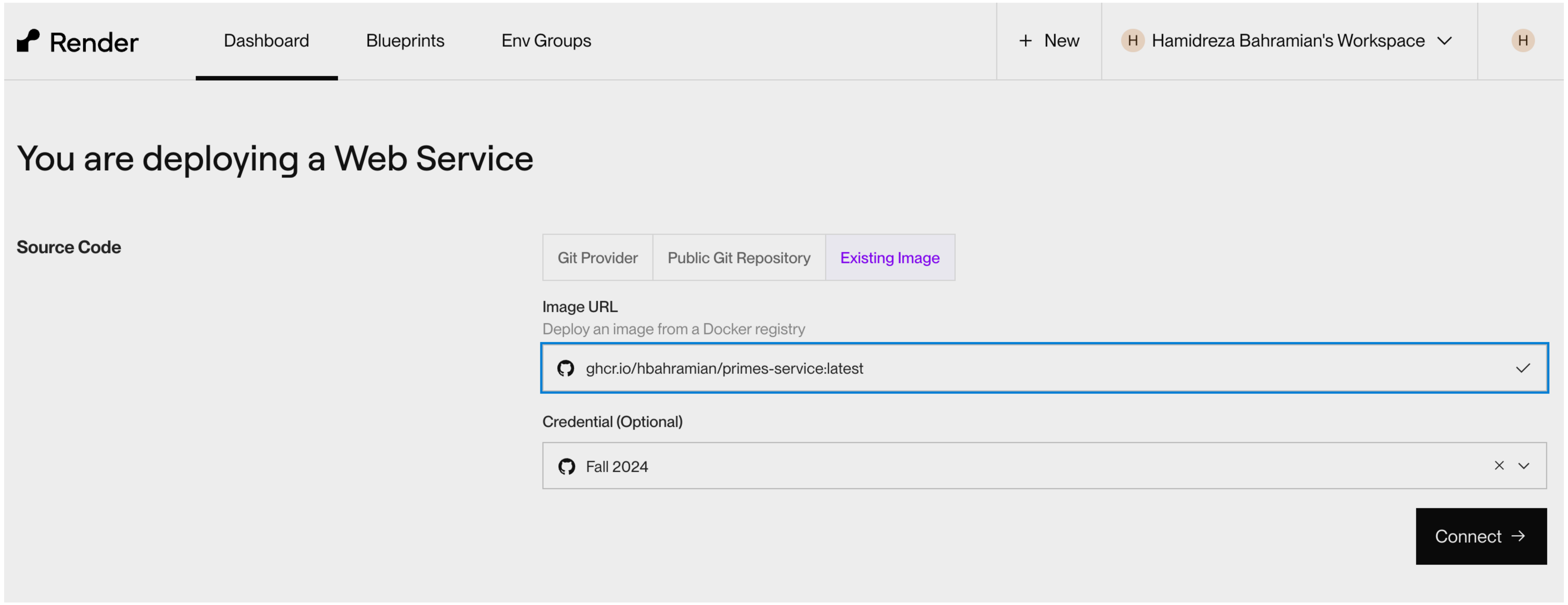
- Create a free account on render.com
- In the dashboard page create a new web service.



Web services are kept up and running at all times, with native SSL and HTTP/2 support. Add a persistent disk or custom domain. Scale up and down with ease.

Steps

- Select “Existing image” tab and fill in the form.



You are deploying a Web Service

Source Code

Image URL

Deploy an image from a Docker registry

ghcr.io/hbahramian/primes-service:latest

Credential (Optional)

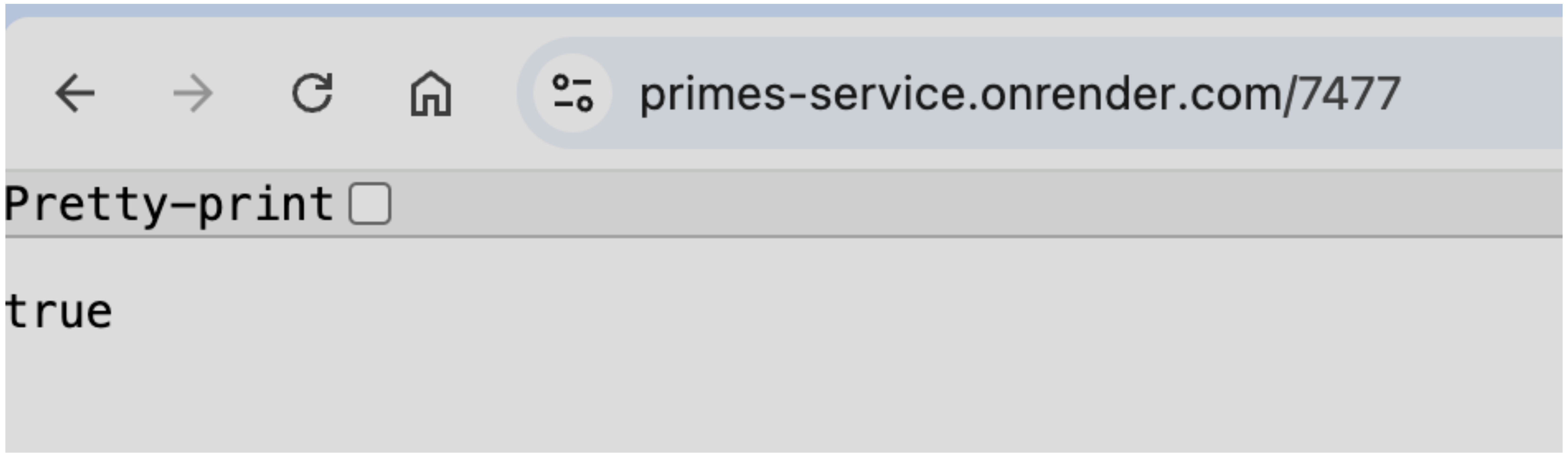
Fall 2024

Steps

- Click next, give the service a name (primes-service-YOURUSERNAME), select the FREE plan (hobby projects) and create the service.

Steps

- Verify that the service is working: **https://primes-service-YOURUSERNAME.onrender.com/**
- Verify that the add function is working: https://primes-service-YOURUSERNAME.onrender.com/primes/49



```
true
```

The End!

- Submit the url of your primes service repository.
- Submit the url of your primes service deployed on render.com.