Building a Spring Boot Microservice with Gitpod Step by Step in 5 minutes

In today's fast-paced world of software development, microservices have become a popular architectural pattern. Microservices enable developers to break down large, monolithic applications into smaller, independent services that can be developed, deployed, and scaled independently. This approach has been widely adopted due to its ability to promote flexibility, scalability, and resilience in software systems.

One of the most popular frameworks for building microservices is Spring Boot, which is built on top of the widely-used Spring framework. Spring Boot simplifies the development of Spring applications by providing production-ready defaults for various configurations and reducing boilerplate code. With Spring Boot, developers can quickly create stand-alone, production-grade applications that can be easily deployed and managed.

In this article, we'll walk through the process of building a Spring Boot microservice using Gitpod, a cloud-based development environment. Gitpod allows developers to work on their code from any device, without having to set up a local development environment. By leveraging Gitpod's pre-built development environments, you can start working on your project immediately, without worrying about installing dependencies or configuring your local machine.

We'll begin by introducing Spring, Spring Boot, microservices, and Gitpod in an easy and intuitive manner. Then, we'll walk you through cloning a Spring Boot GitHub repository, setting it up in Gitpod, and running a complete microservice application. You'll learn how to work with Spring Boot, Gitpod, and Java extension packs to develop a functional application quickly and efficiently.

In this comprehensive guide, you will learn about:

1. **Spring Framework**: An open-source Java-based framework that simplifies the development of enterprise applications by providing a comprehensive infrastructure for building, deploying, and maintaining Java applications.
2. **Spring Boot**: A project built on top of the Spring framework that simplifies the development process by providing production-ready defaults and reducing boilerplate code. Spring Boot makes it easy to create stand-alone, production-grade applications with minimal effort.
3. **Microservices**: An architectural pattern that promotes breaking down large, monolithic applications into smaller, independent services. Microservices can be developed, deployed, and scaled independently, which makes them a popular choice for building flexible, scalable, and resilient software systems.

4. **Gitpod**: A cloud-based development environment that allows developers to work on their code from any device without having to set up a local development environment. Gitpod offers pre-built development environments, enabling you to start working on your project right away.

By the end of this article, you will have a solid understanding of these concepts and will be well-equipped to build a Spring Boot microservice using Gitpod. Let's get started!

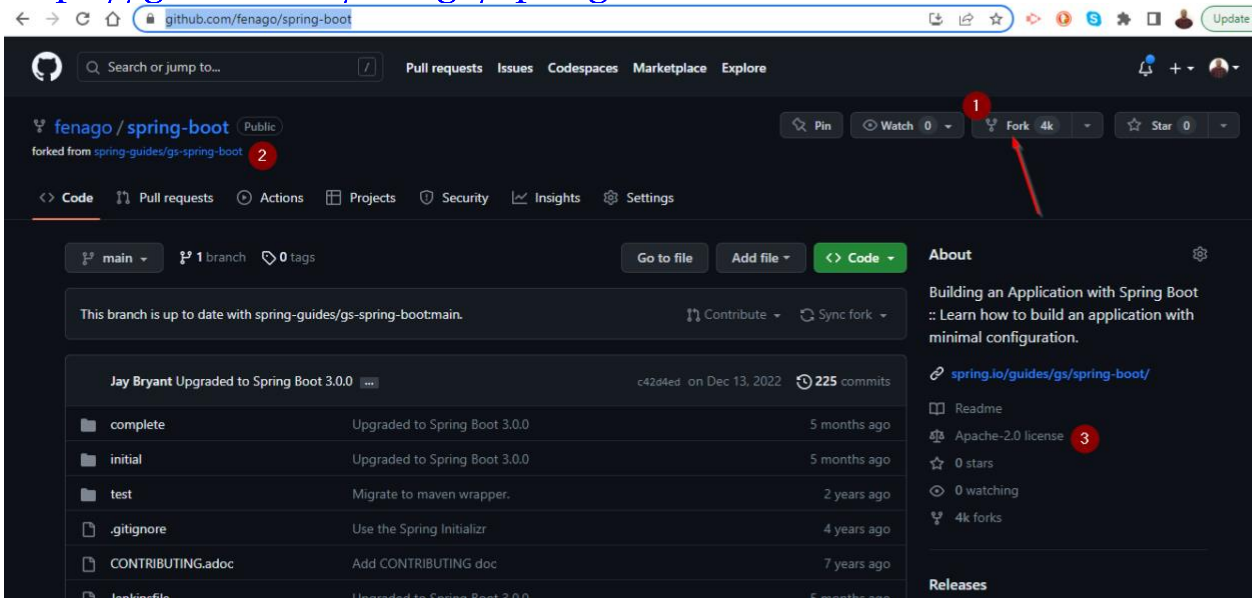
Prerequisites

- A GitHub account
- Basic understanding of Spring Boot and Java

Step 1: Clone the Repository

First, we'll clone the following repository:

https://github.com/fenago/spring-boot



This branch is up to date with spring-guides/gs-spring-boot:main.

Jay Bryant Upgraded to Spring Boot 3.0.0 ... c42d4ed on Dec 13, 2022 225 commits

complete	Upgraded to Spring Boot 3.0.0	5 months ago
initial	Upgraded to Spring Boot 3.0.0	5 months ago
test	Migrate to maven wrapper.	2 years ago
.gitignore	Use the Spring Initializr	4 years ago
CONTRIBUTING.adoc	Add CONTRIBUTING doc	7 years ago

About

Building an Application with Spring Boot :: Learn how to build an application with minimal configuration.

spring.io/guides/gs/spring-boot/

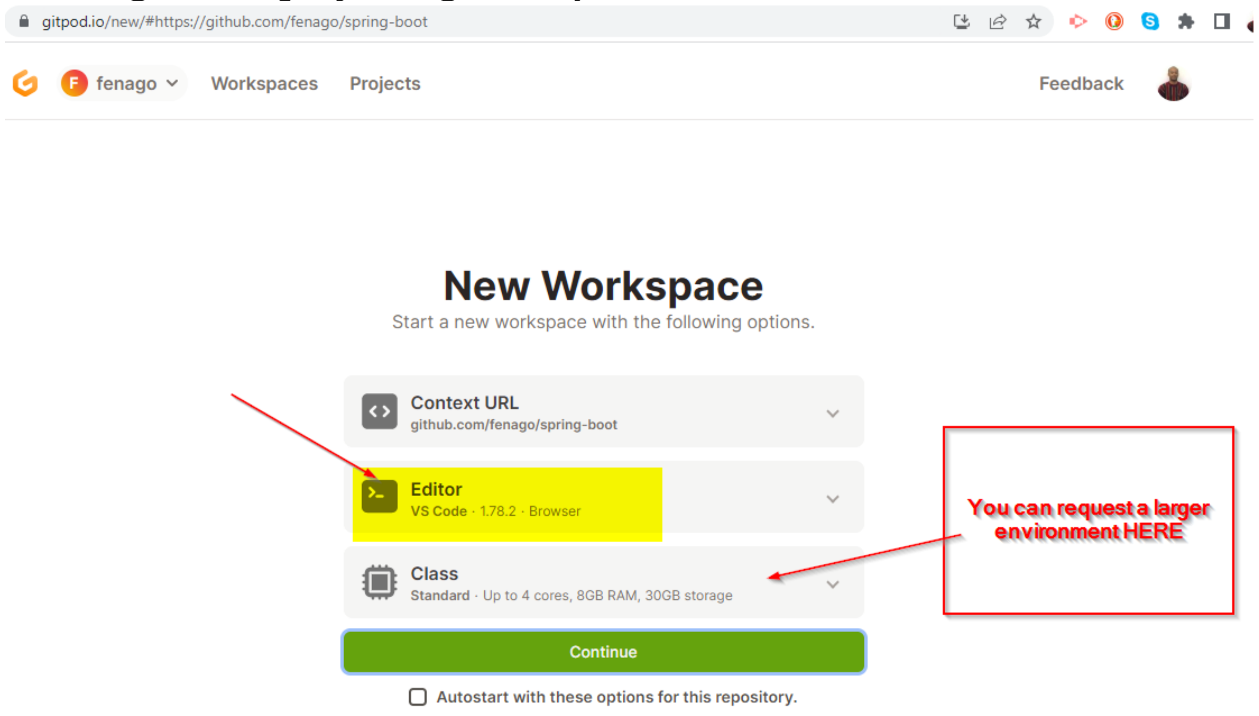
Step 2: Launch Gitpod

To launch Gitpod, use the following URL:

```
https://gitpod.io/#https://github.com/fenago/spring-boot
```

Note: Replace “fenago” with your GitHub username.

Gitpod will automatically set up your workspace, and you’ll be able to start working on the project right away.



gitpod.io/new/#https://github.com/fenago/spring-boot

fenago Workspaces Projects Feedback

New Workspace

Start a new workspace with the following options.

Context URL
github.com/fenago/spring-boot

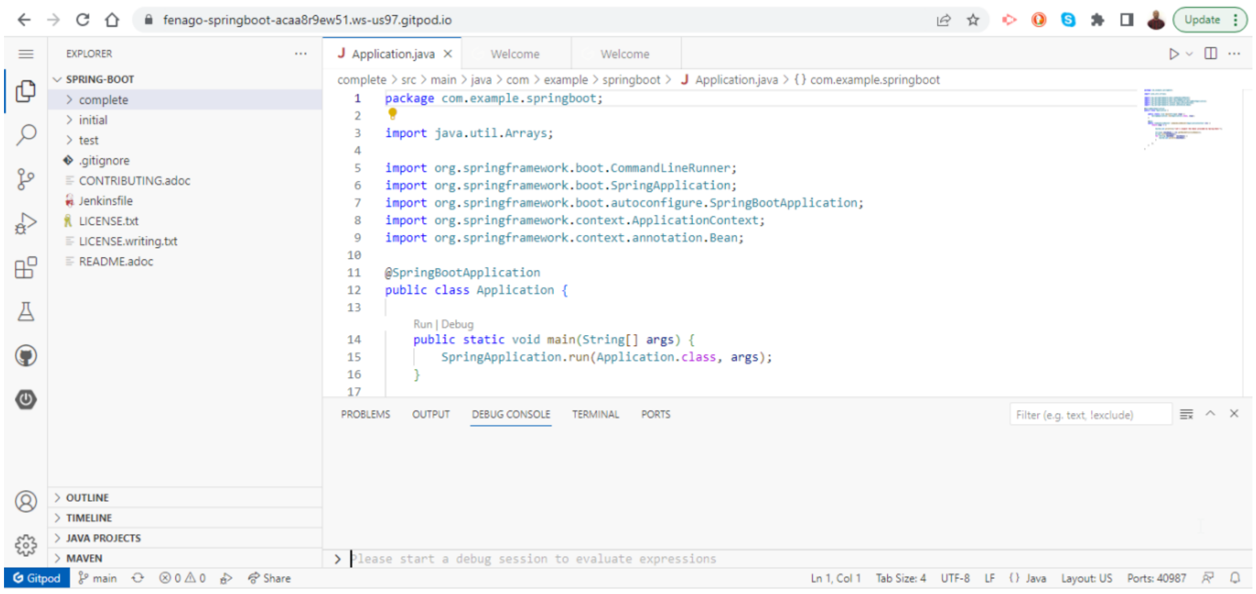
Editor
VS Code · 1.78.2 · Browser

Class
Standard · Up to 4 cores, 8GB RAM, 30GB storage

You can request a larger environment HERE

Continue

Autostart with these options for this repository.



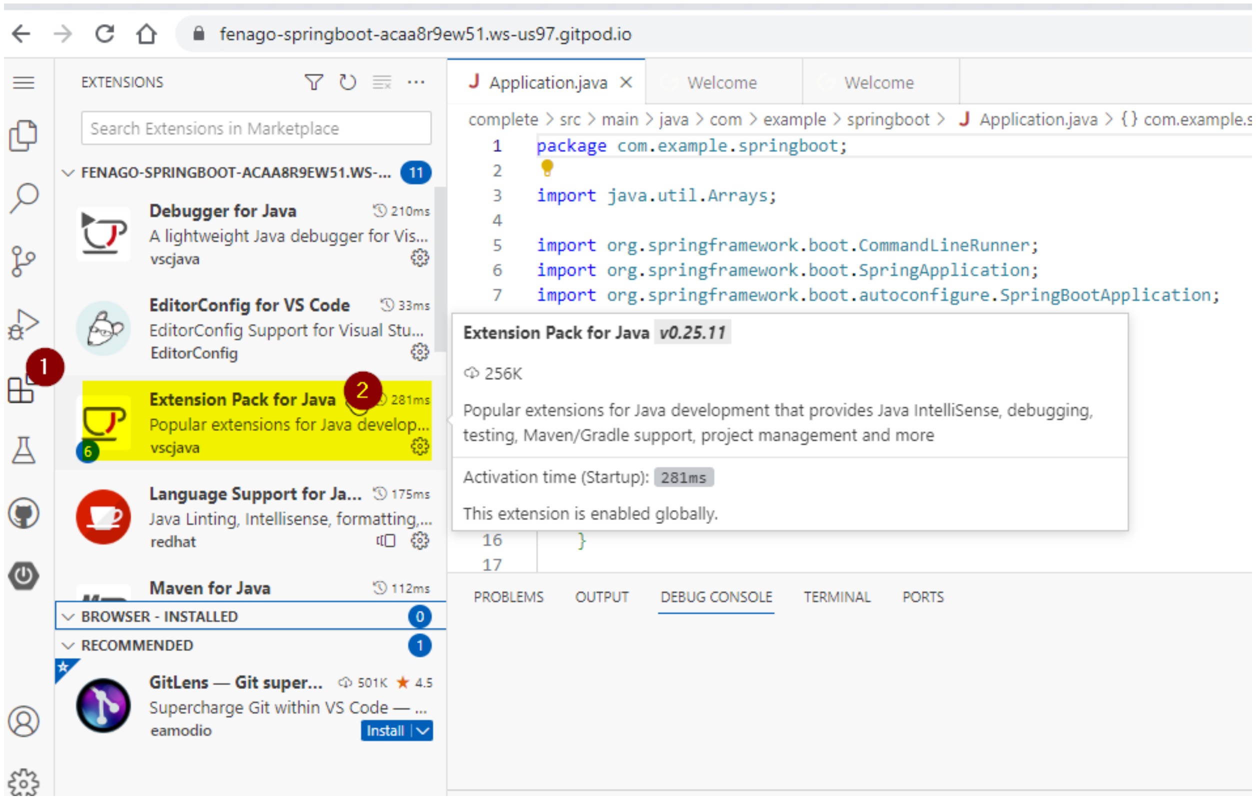
```
package com.example.springboot;

import java.util.Arrays;

import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
public class Application {

    Run | Debug
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
```

```
> Please start a debug session to evaluate expressions
```

Step 3: Install the Required Extensions

In Gitpod, you'll need to install the following extensions:

- Extension Pack for Java


fenago-springboot-acaa8r9ew51.ws-us97.gitpod.io
EXTENSIONS
Search Extensions in Marketplace
FENAGO-SPRINGBOOT-ACAA8R9EW51.WS-...
11
Debugger for Java
210ms
A lightweight Java debugger for Vis...
vscjava
EditorConfig for VS Code
33ms
EditorConfig Support for Visual Stu...
EditorConfig
1
Extension Pack for Java
2
281ms
Popular extensions for Java develop...
vscjava
6
Language Support for Ja...
175ms
Java Linting, Intellisense, formatting,...
redhat
Maven for Java
112ms
BROWSER - INSTALLED
0
RECOMMENDED
1
GitLens — Git super...
501K
4.5
Supercharge Git within VS Code — ...
eamodio
Install
Application.java
Welcome
Welcome
complete > src > main > java > com > example > springboot > Application.java > {} com.example.s
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS

```
package com.example.springboot;

import java.util.Arrays;

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

Extension Pack for Java v0.25.11

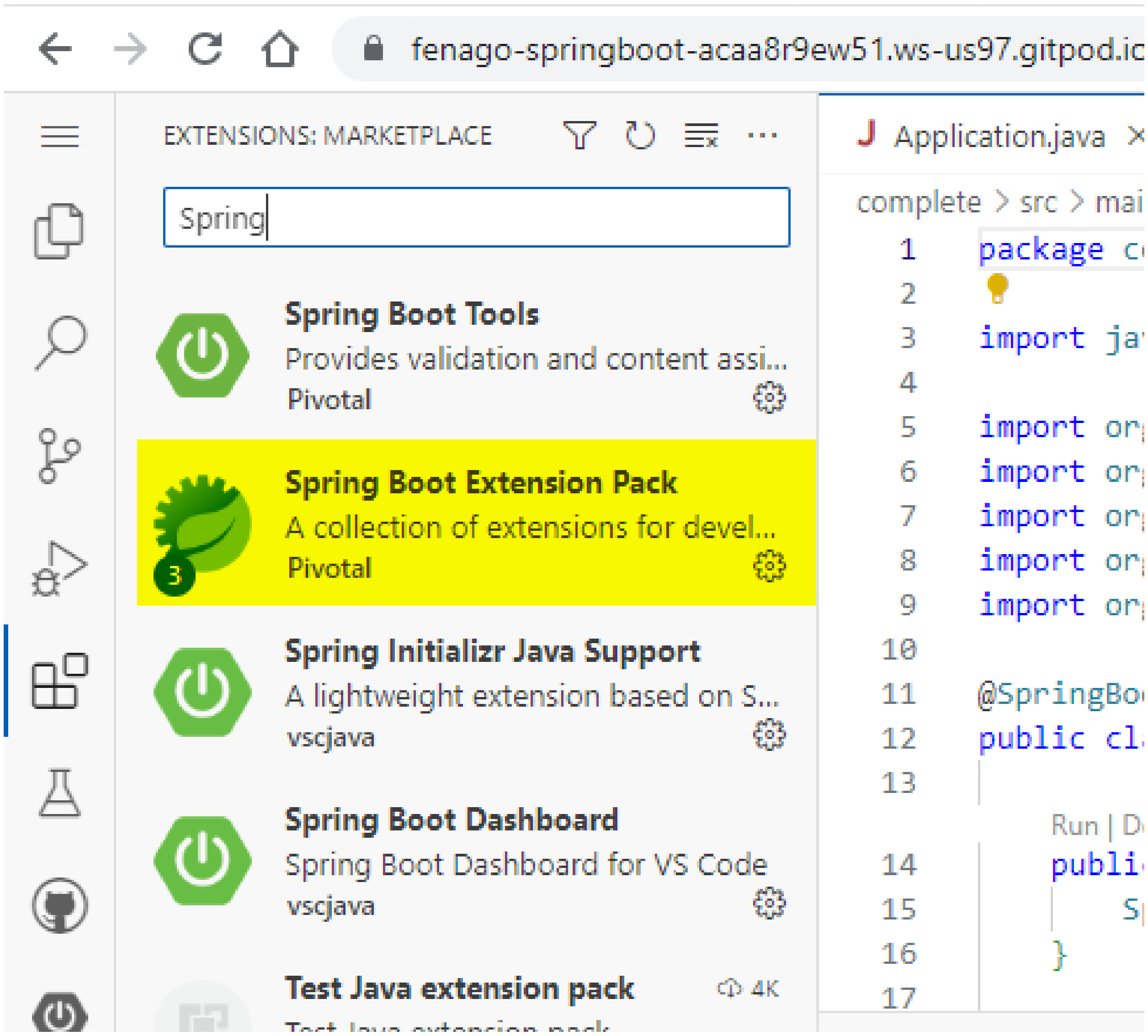
256K

Popular extensions for Java development that provides Java IntelliSense, debugging, testing, Maven/Gradle support, project management and more

Activation time (Startup): 281ms

This extension is enabled globally.

- Spring Boot Extension Pack


fenago-springboot-acaa8r9ew51.ws-us97.gitpod.ic
EXTENSIONS: MARKETPLACE
Spring
Spring Boot Tools
Provides validation and content assi...
Pivotal
Spring Boot Extension Pack
A collection of extensions for devel...
Pivotal
Spring Initializr Java Support
A lightweight extension based on S...
vscjava
Spring Boot Dashboard
Spring Boot Dashboard for VS Code
vscjava
Test Java extension pack
4K
Application.java
complete > src > mai

```
package c
import ja
import or
import or
import or
import or
import or
@SpringBo
public cl
    Run | D
    publi
        S
    }
```

To install the extensions, click on the Extensions icon on the left sidebar and search for each extension. Click on the "Install" button to add them to your workspace.

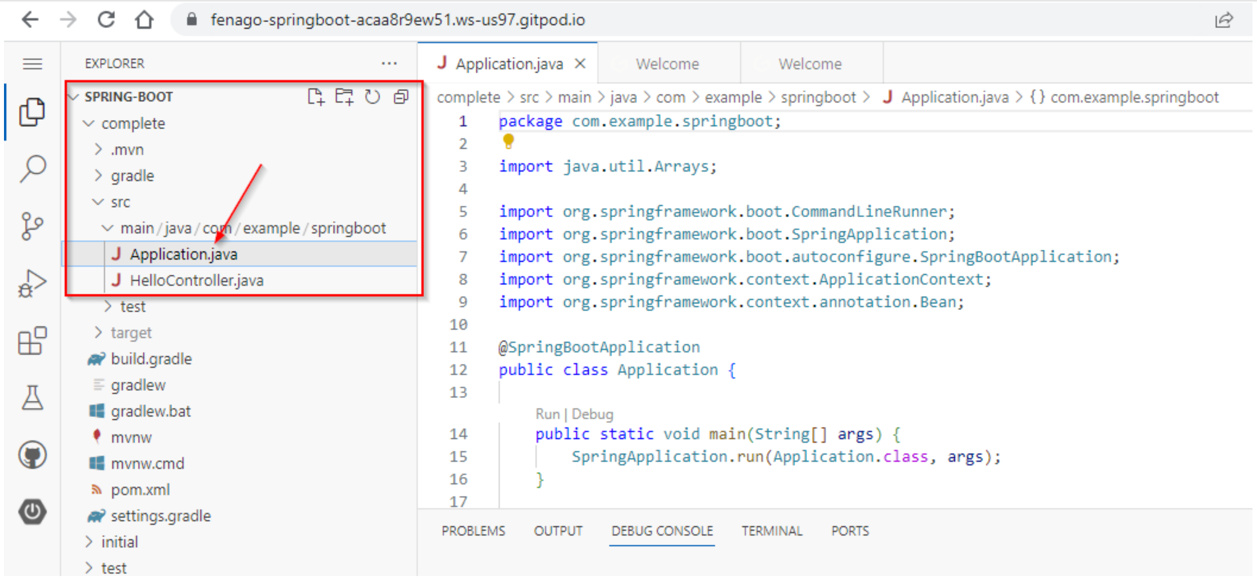
After installing the extensions, you may need to reload or restart Gitpod for the changes to take effect.

***note**: You will be asked to use either Gradle or Maven — all articles that I write will assume Maven.*

Step 4: Navigate to the Application File

In Gitpod, go to the following file:

```
/workspace/spring-boot/complete/src/main/java/com/example/springboot/Application.java
```

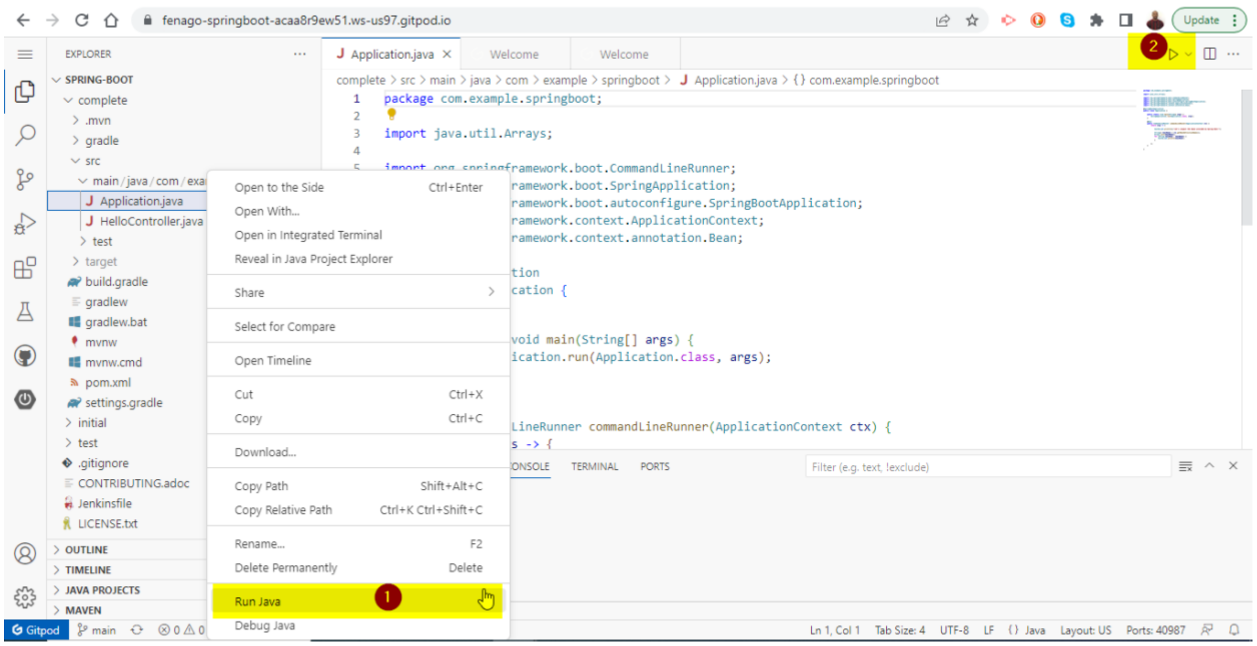


```java
package com.example.springboot;

import java.util.Arrays;

import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
public class Application {

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

```

This is the main Java class for our Spring Boot application.

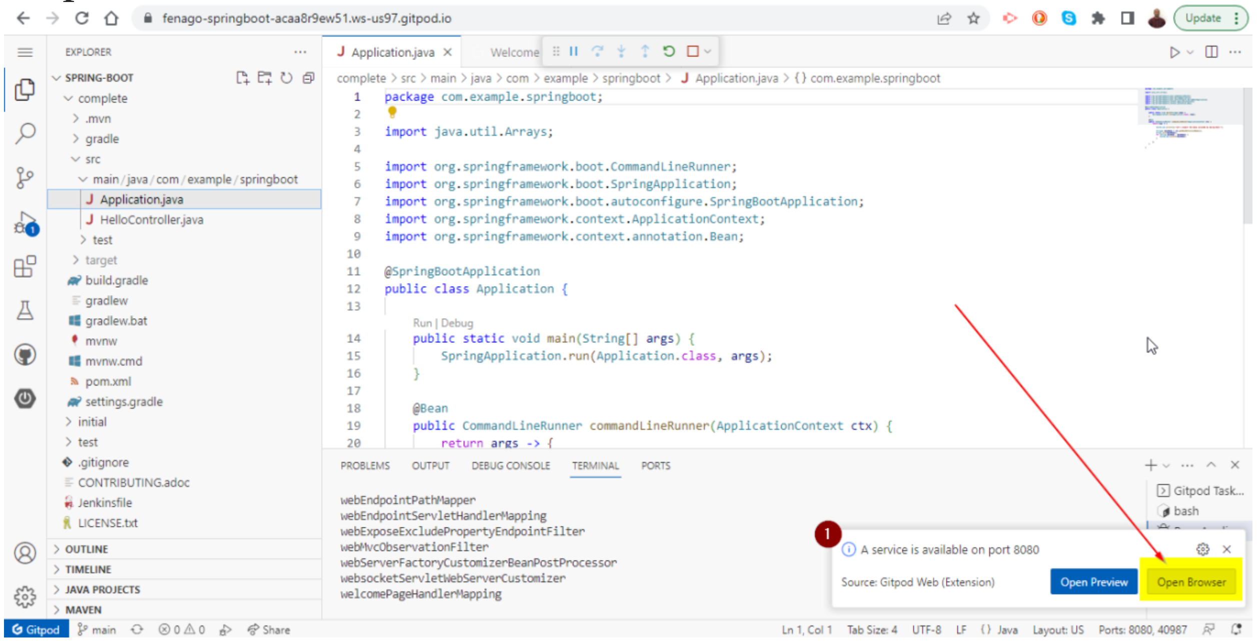
Step 5: Run the Application

To run the application, right-click inside the `Application.java` file and choose "Run". This will start the Spring Boot application.



```
package com.example.springboot;

import java.util.Arrays;

import org.springframework.boot.CommandLineRunner;
ramework.boot.SpringApplication;
ramework.boot.autoconfigure.SpringBootApplication;
ramework.context.ApplicationContext;
ramework.context.annotation.Bean;

tion
cation {

void main(String[] args) {
ication.run(Application.class, args);


LineRunner commandLineRunner(ApplicationContext ctx) {
s -> {
```

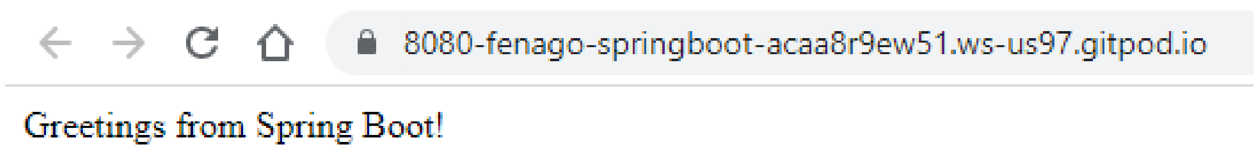
The first time you run the application, it will create a port (8080) for the application to listen on. Run the application again to open the browser and see the output.



```
package com.example.springboot;

import java.util.Arrays;

import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
public class Application {

    Run | Debug
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }

    @Bean
    public CommandLineRunner commandLineRunner(ApplicationContext ctx) {
        return args -> {
```

```
webEndpointPathMapper
webEndpointServletHandlerMapping
webExposeExcludePropertyEndpointFilter
webMvcObservationFilter
webServerFactoryCustomizerBeanPostProcessor
websocketServletWebServerCustomizer
welcomePageHandlerMapping
```

Now, you should see the output of your Spring Boot application in your browser. Congratulations! You have successfully built and run a Spring Boot microservice using Gitpod.



```
Greetings from Spring Boot!
```