Lesson 04 Demo 01

Spring Cloud Configuration

Objective: To understand how to manage an application through Spring Cloud configuration

Tool required: Eclipse IDE and Visual Studio Code

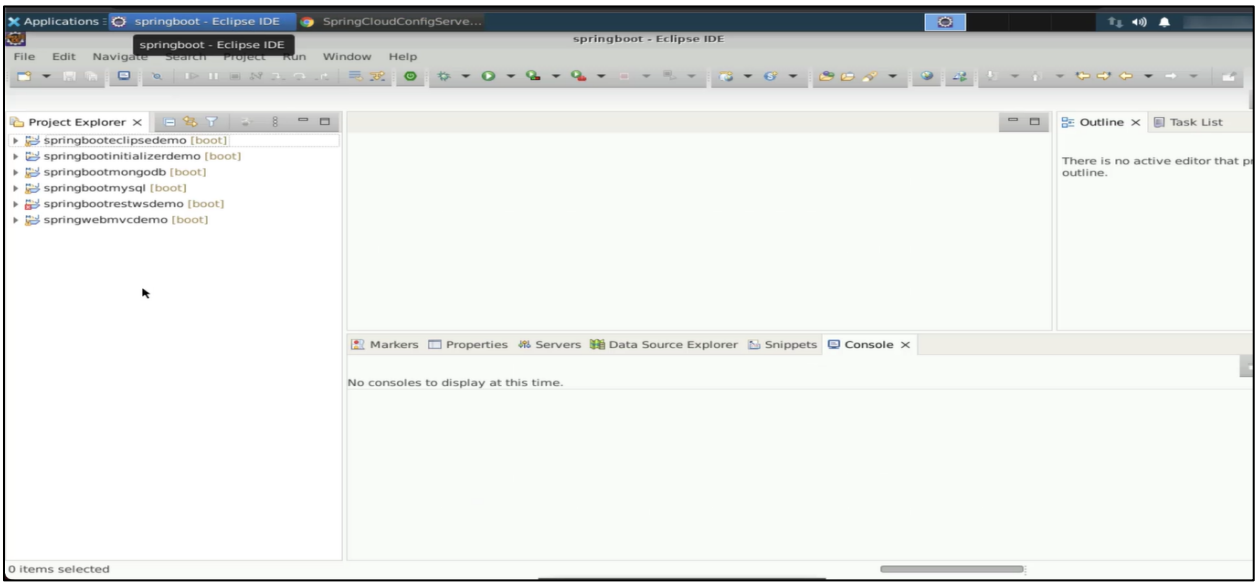
Prerequisites: None

Steps to be followed:

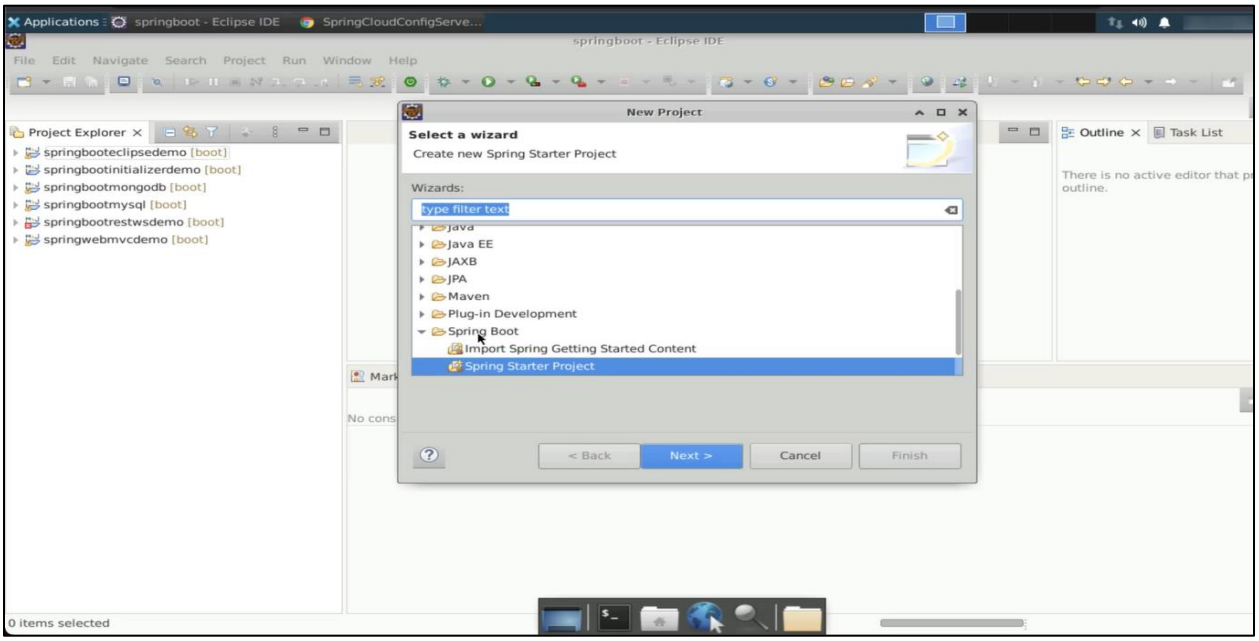
1. Creating a Spring Boot project with cloud configuration
2. Configuring the Spring Cloud in Eclipse IDE
3. Configuring the Git repository
4. Setting up the Git repository path for the Spring Cloud Configuration
5. Deploying the Spring Cloud Configuration project

Step 1: Creating a Spring Boot project with cloud configuration

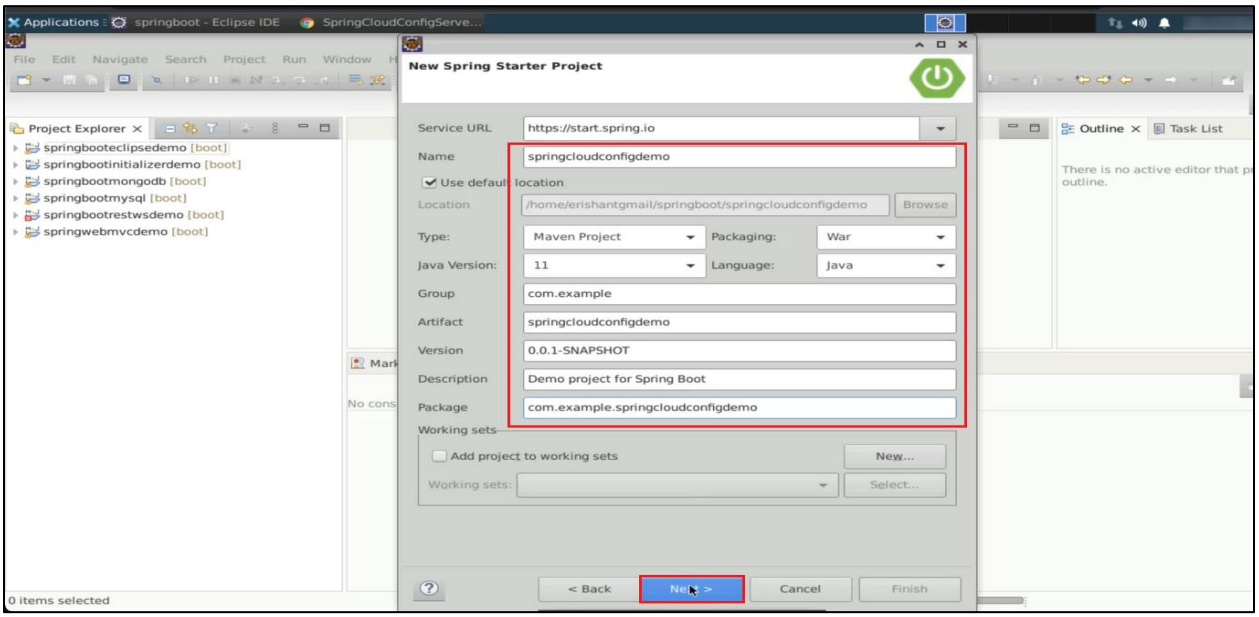
1.1 Open **Eclipse IDE** to create a new **Spring Boot** project



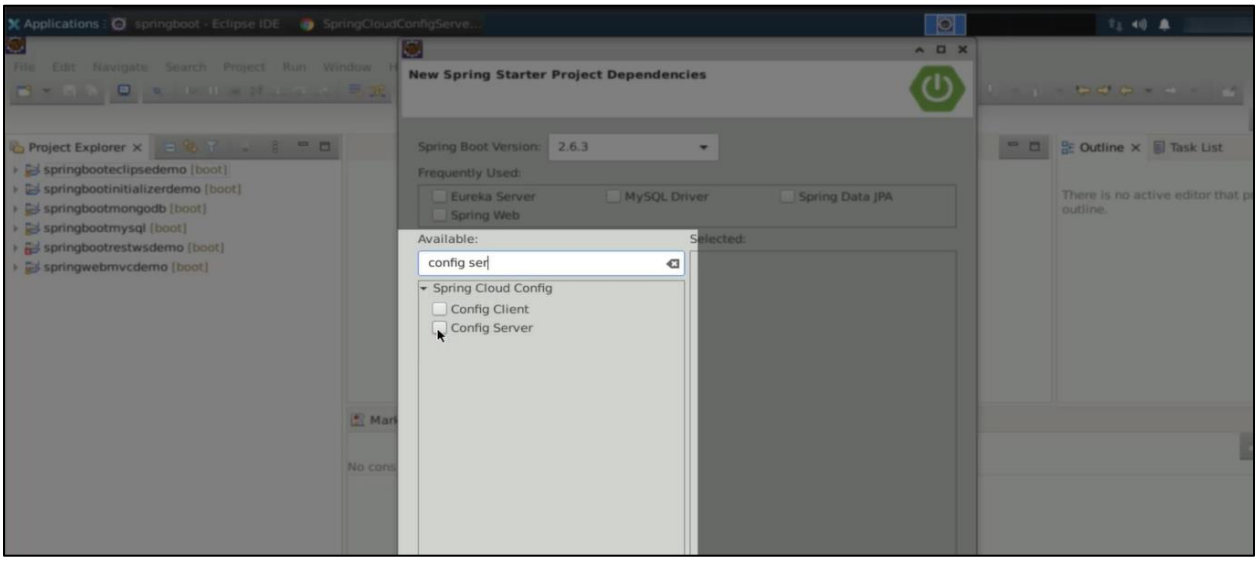
1.2 Right-click on **Eclipse IDE**, select **New > Project > Spring Boot > Spring Starter Project**. Now, click **Next**



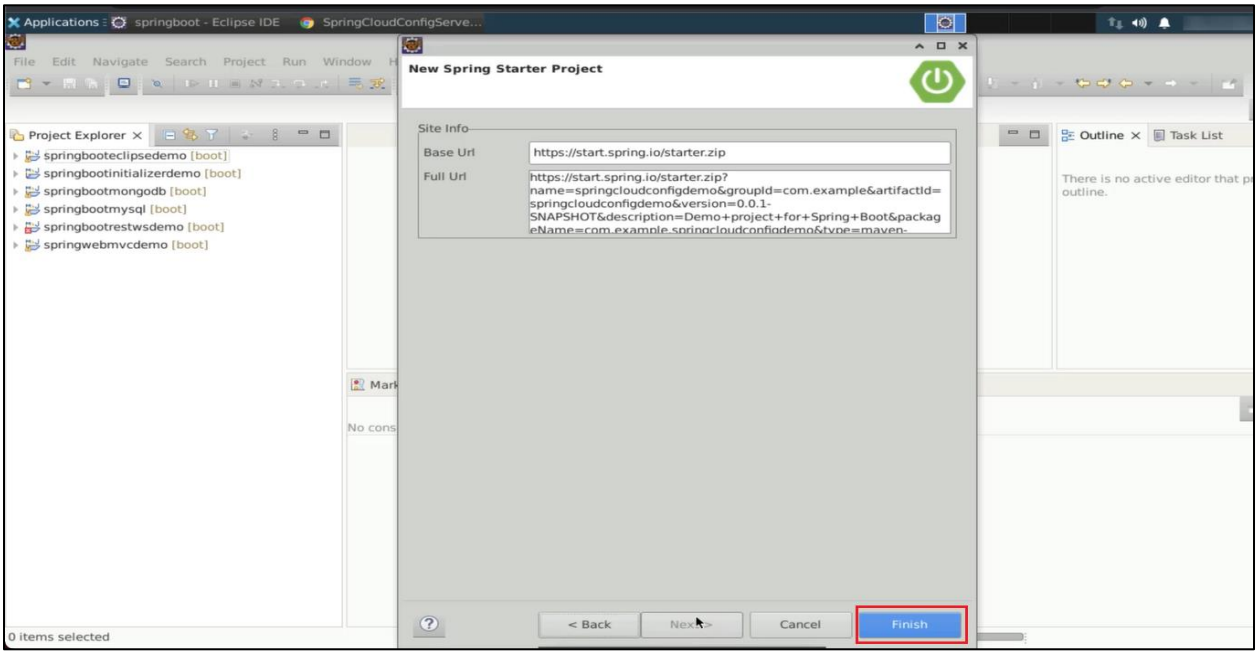
1.3 In the **New Spring Starter Project** wizard, provide a name for the Spring Boot project, like **springcloudconfigdemo**. Set the **Packaging** type as **War**, **Package Id** as **com.example.springcloudconfigdemo**, and click **Next**



1.4 To create the dependencies for the project, search for the **config server**. Under **Spring Cloud Config**, select **Config Server** and click **Next**

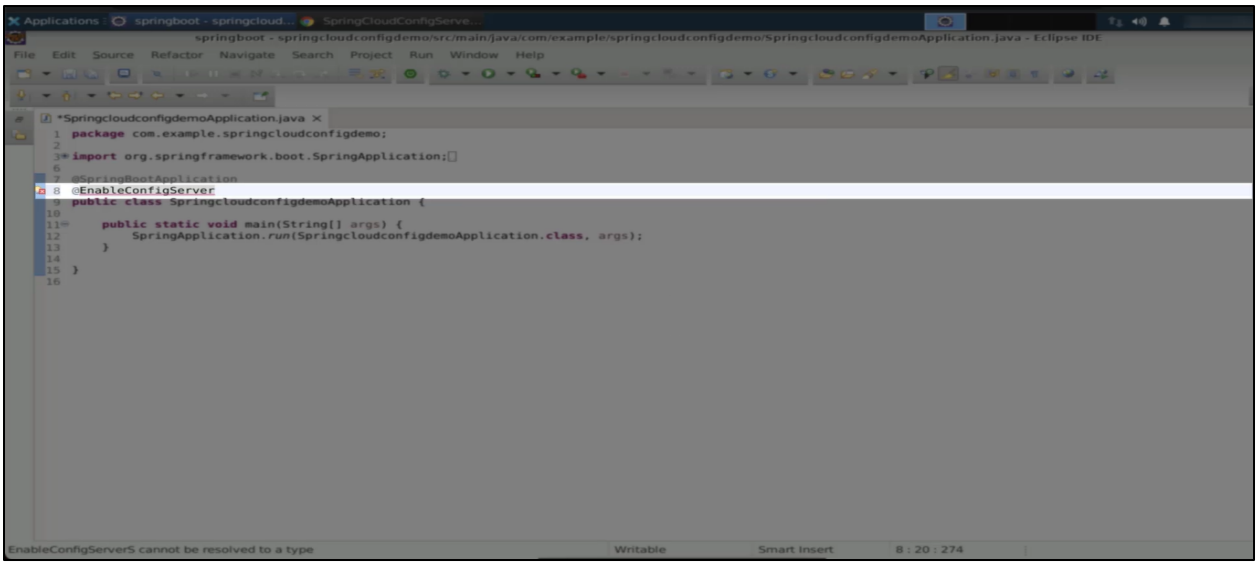


1.5 Now, click **Finish**

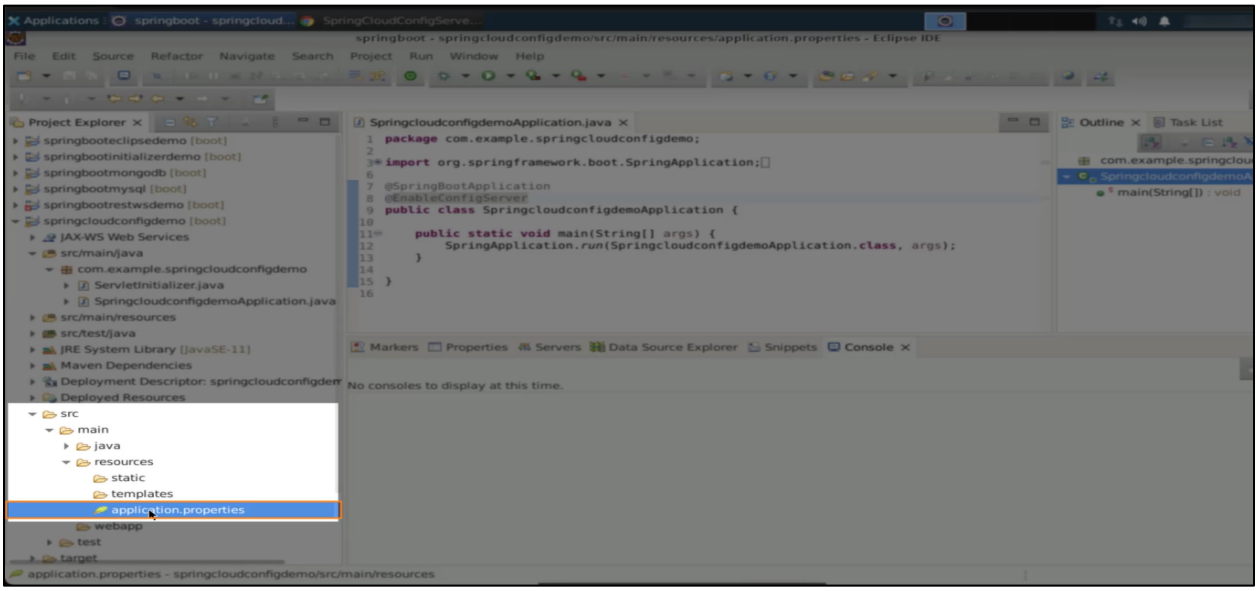


Step 2: Configuring the Spring Cloud in Eclipse IDE

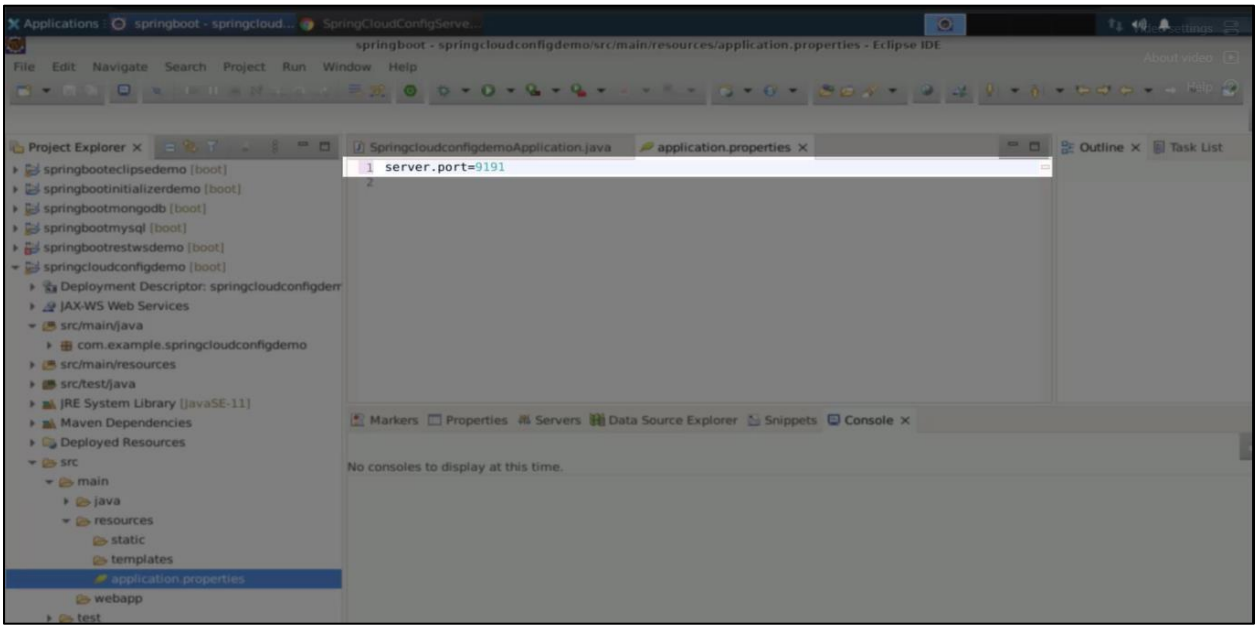
2.1 In **SpringcloudconfigdemoApplication.java**, add one of the annotations **@EnableConfigServer** as this will enable the cloud configuration server



2.2 Now, open **application.properties** file under **src > main > resources**

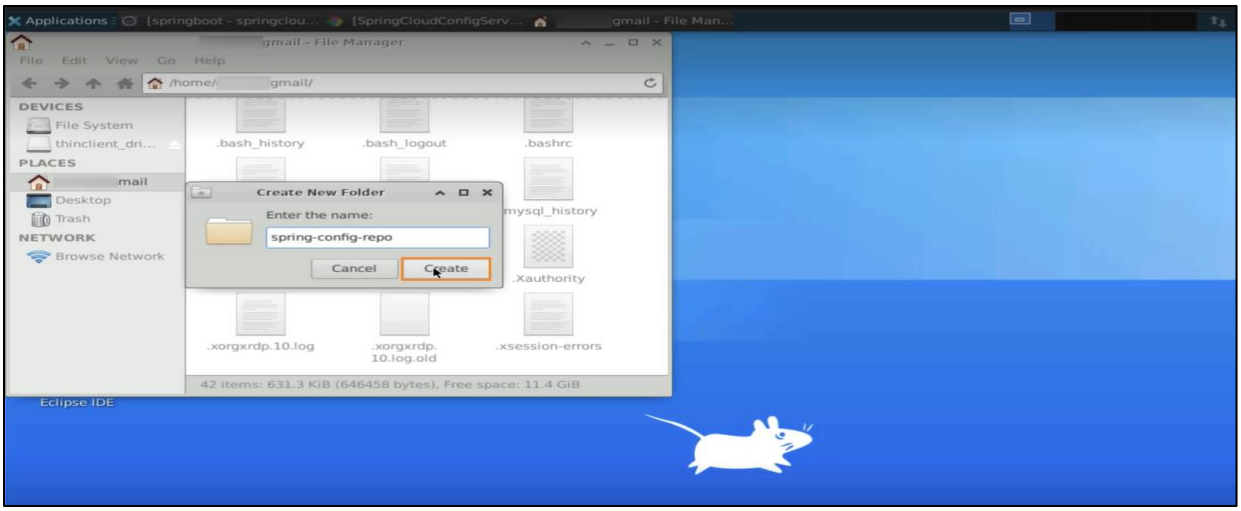


2.3 In **application.properties,** configure the server port as **9191**. Here, also configure the **Git URI**

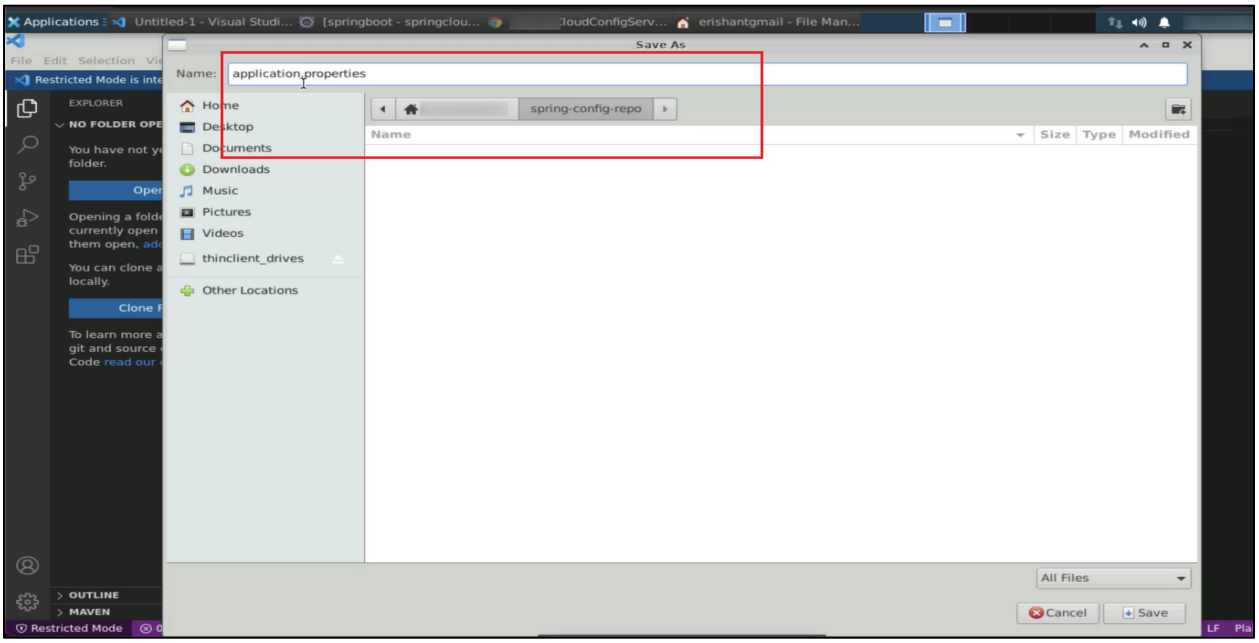


Step 3: Configuring the Git repository

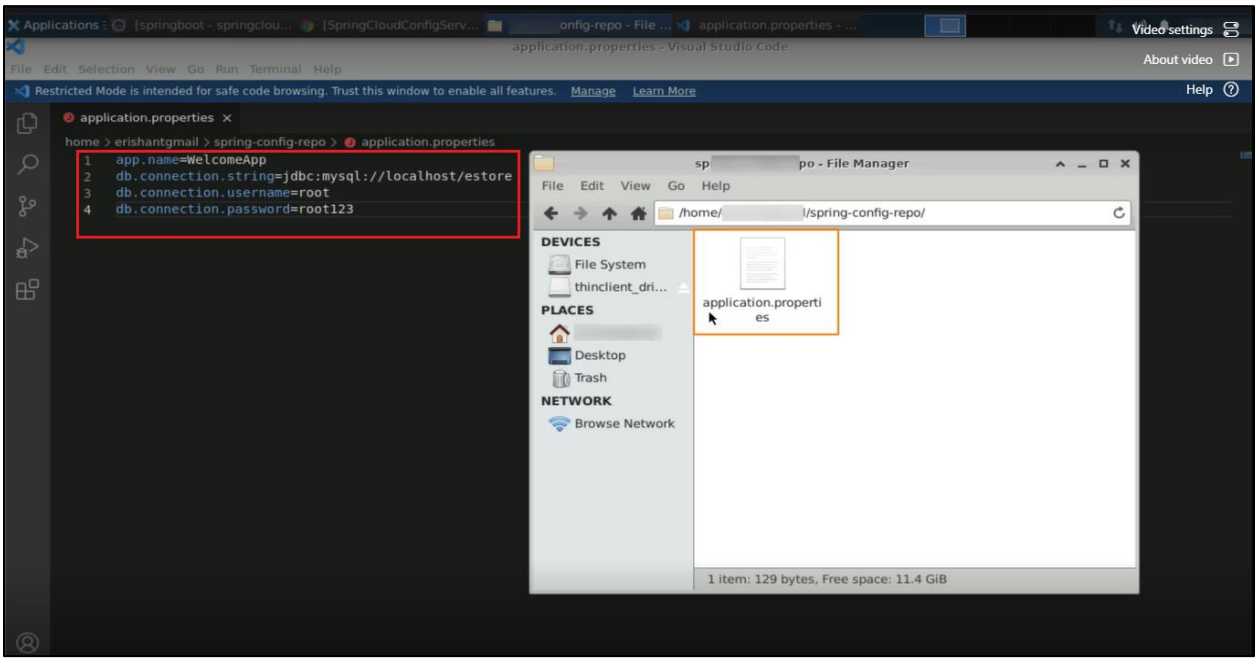
3.1 To configure the **Git URI**, provide a local location. For that, go to the system directory structure and create a folder named **spring-config-repo**



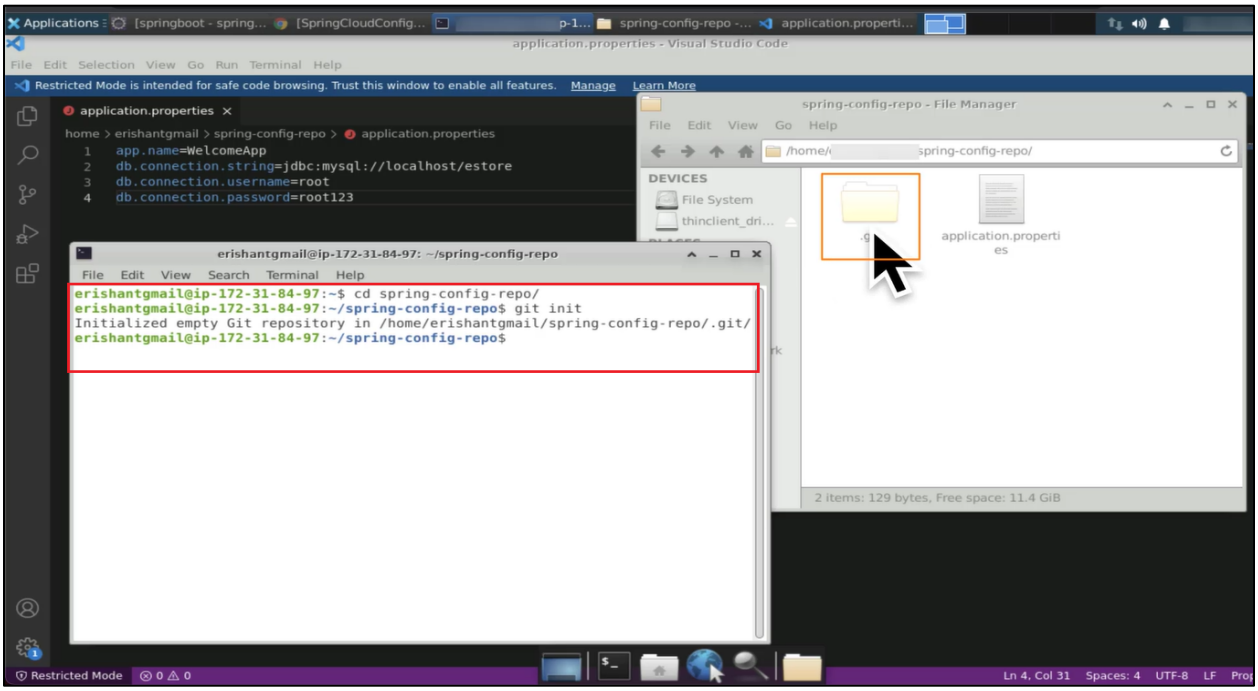
3.2 Open the **spring-config-repo** folder with **Visual Studio Code**. In this, create a new file named **application.properties** that specifies app properties



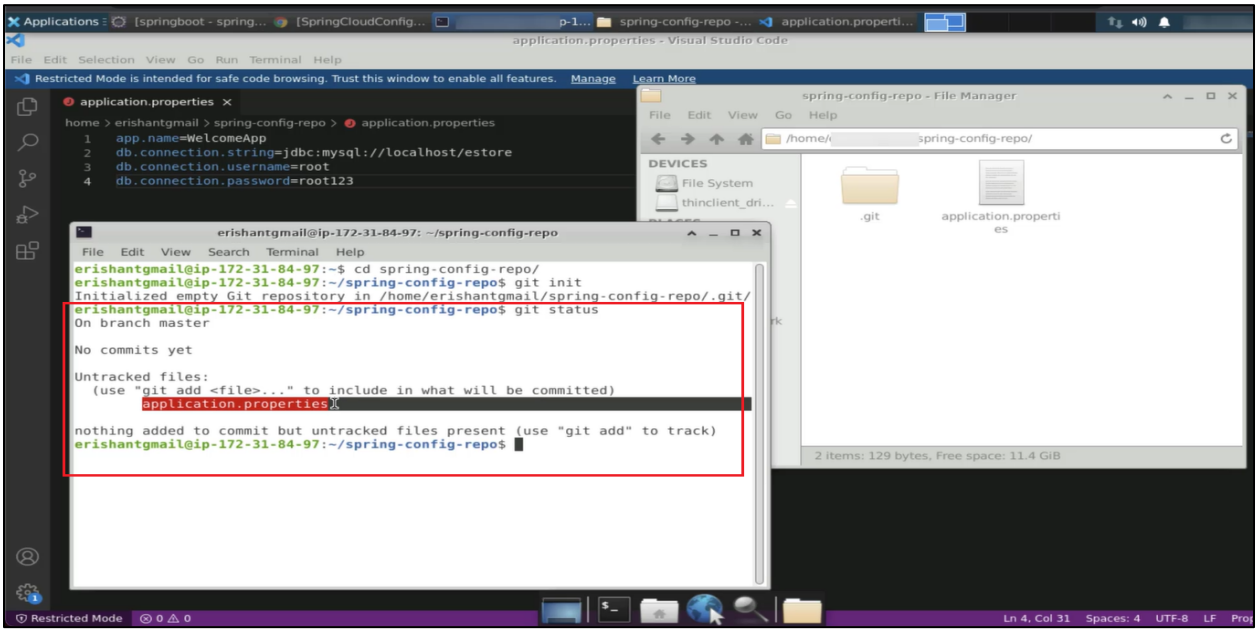
3.3 In **VS Code**, write the highlighted app properties in the **application.properties** file



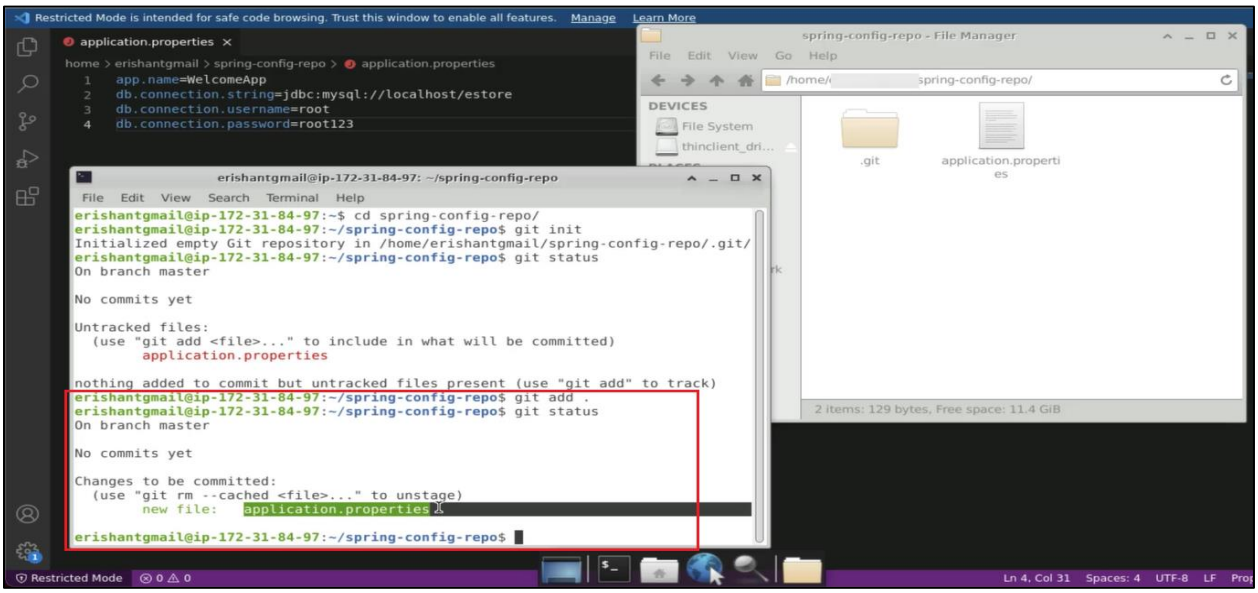
3.4 Now, open the terminal to initialize Git. Write the command **git init** to initiate Git. After initiating Git, a local Git repository will be created in the **spring-config-repo**



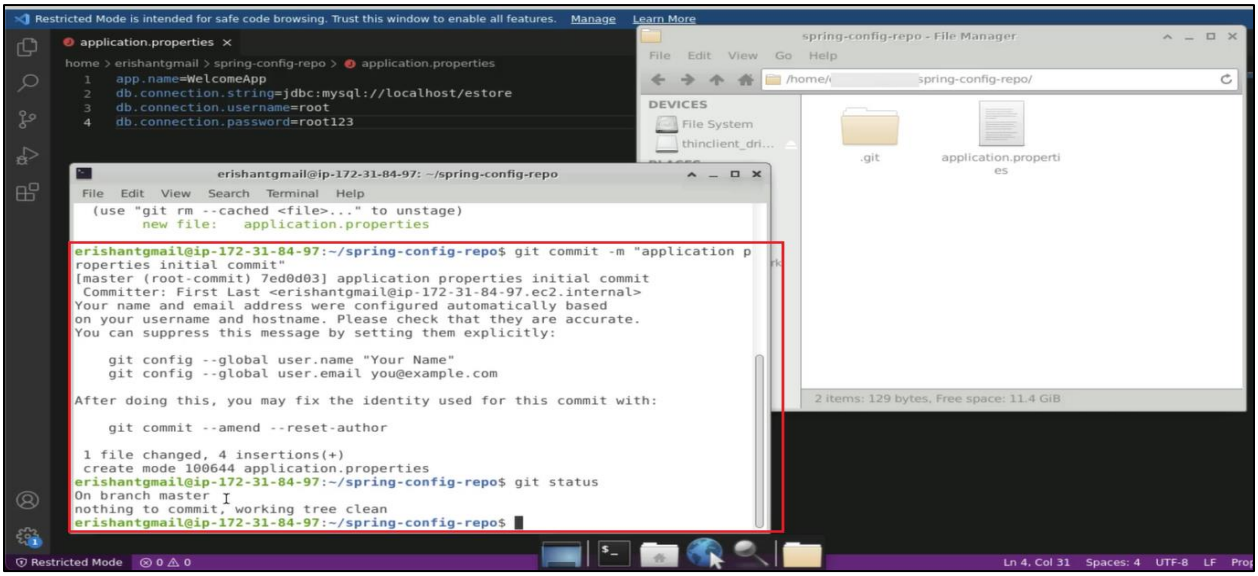
3.5 Check the status of the Git repository with the **git status** command. You'll see no files have been added to the local Git repository yet.



3.6 Now, use the **git add.** command to add the files and check the status again with **the git status** command. You can now see the file **application.properties,** which can be committed to the **git** repository.



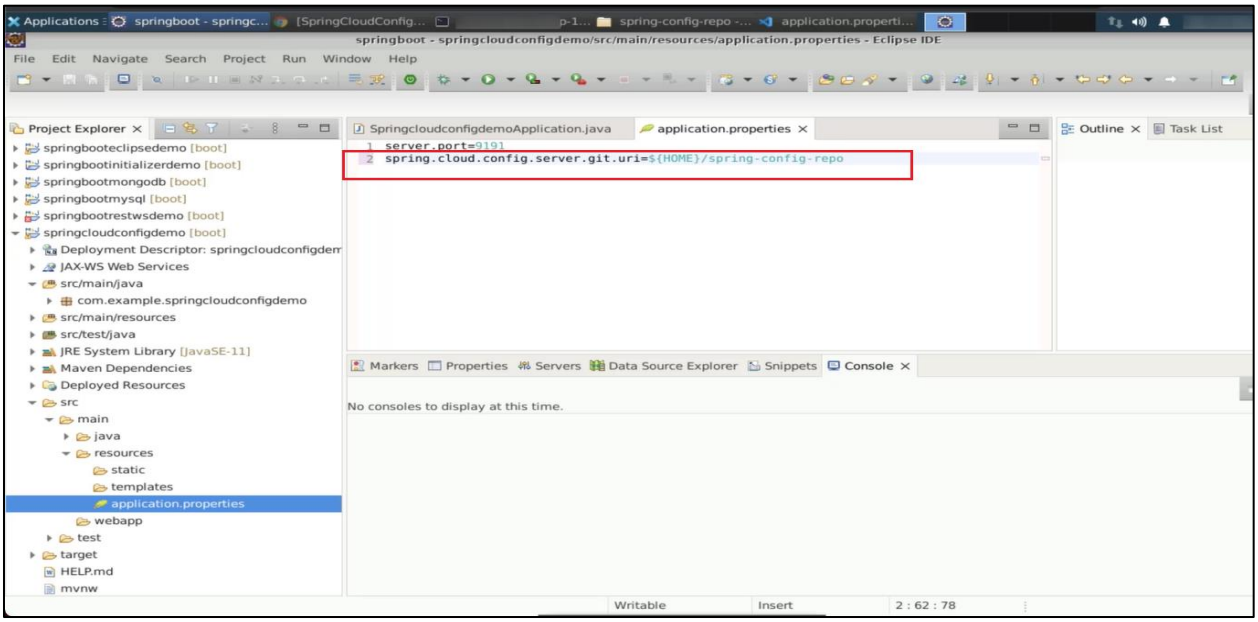
3.7 Use the **git commit -m "application properties initial commit"** command to commit changes to the Git repository. Now, you can check the status using the **git status** command.



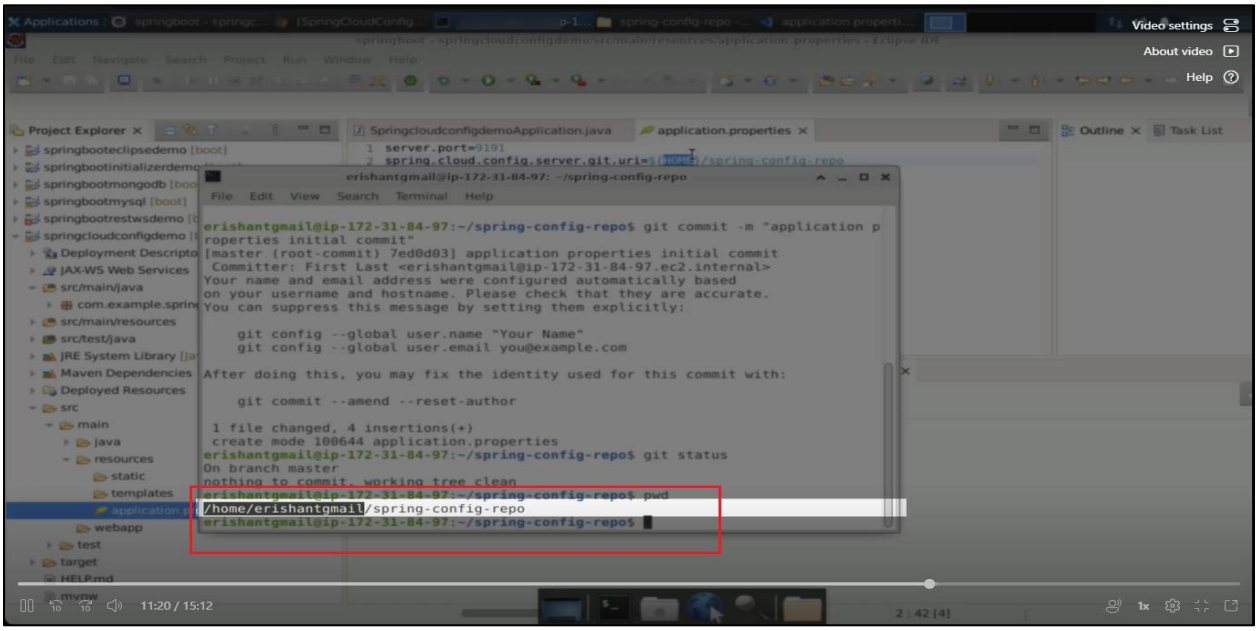
Now the Git repository has been set up for the Spring Cloud Configuration demo.

Step 4: Setting up the Git repository path for the Spring Cloud Configuration

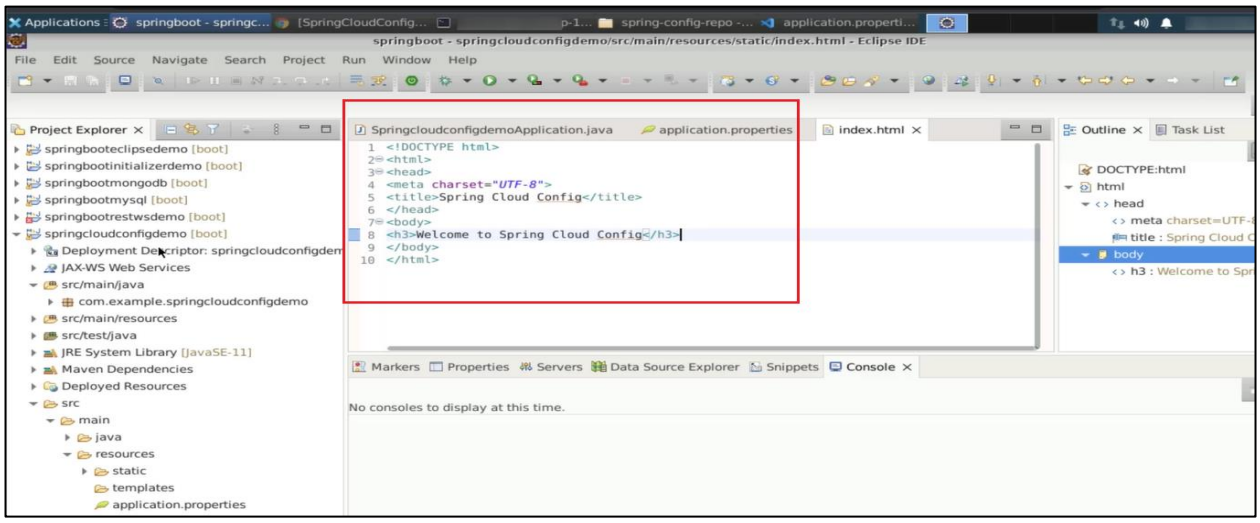
4.1 To set the path of the local Git repository to the **Spring Cloud Config** server, go to the **application.properties** in **Eclipse IDE** and write **spring.cloud.config.server.git.url=${HOME}/spring.config.repo** property



4.2 To check the working directory, write the **pwd** command in the terminal



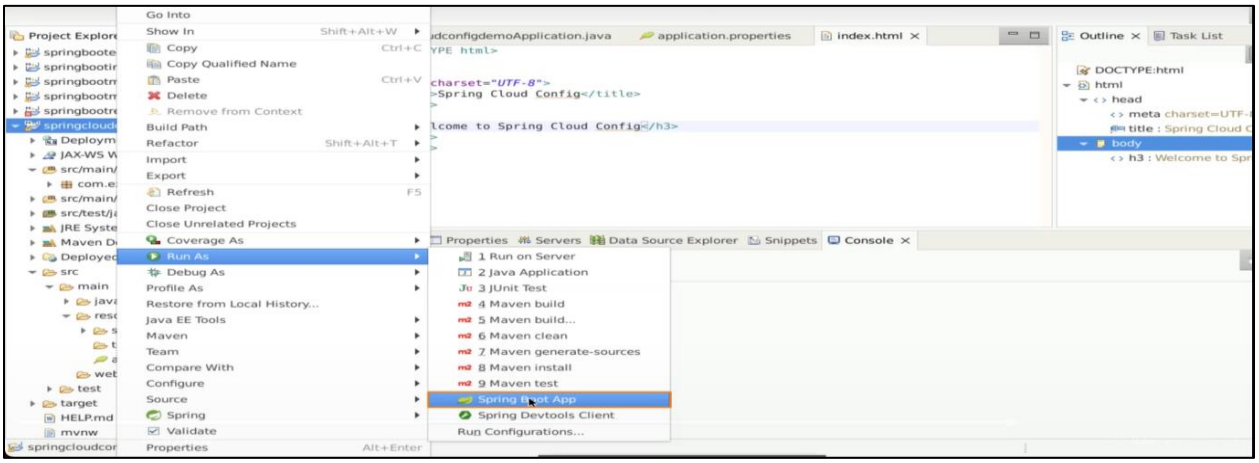
4.3 Before deploying the app, you need to make a welcome page. For this, go to the **static** folder under **resources** and **right-click** on it. Select **New > HTML File**. Create a file named **index.html** and add the highlighted code to that HTML file



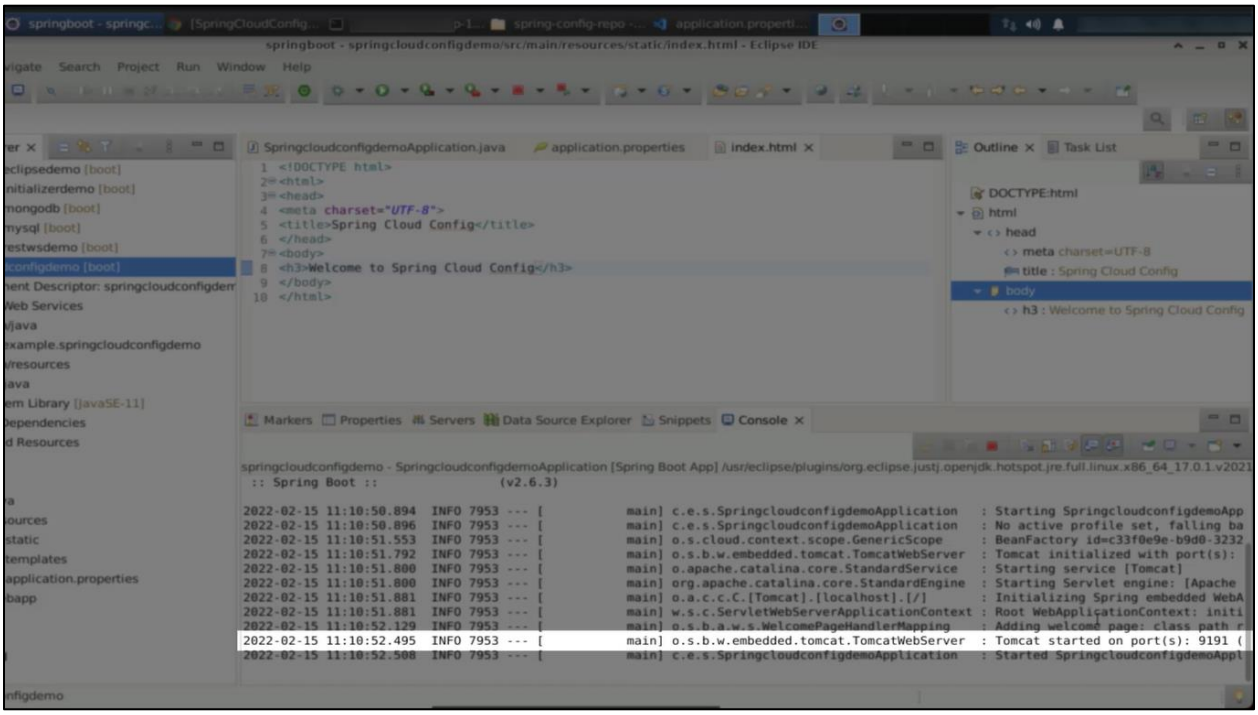
Note: The **index.html** file will be created under the **webapp** folder. Move this under the static folder

Step 5: Deploying the Spring Cloud Configuration project

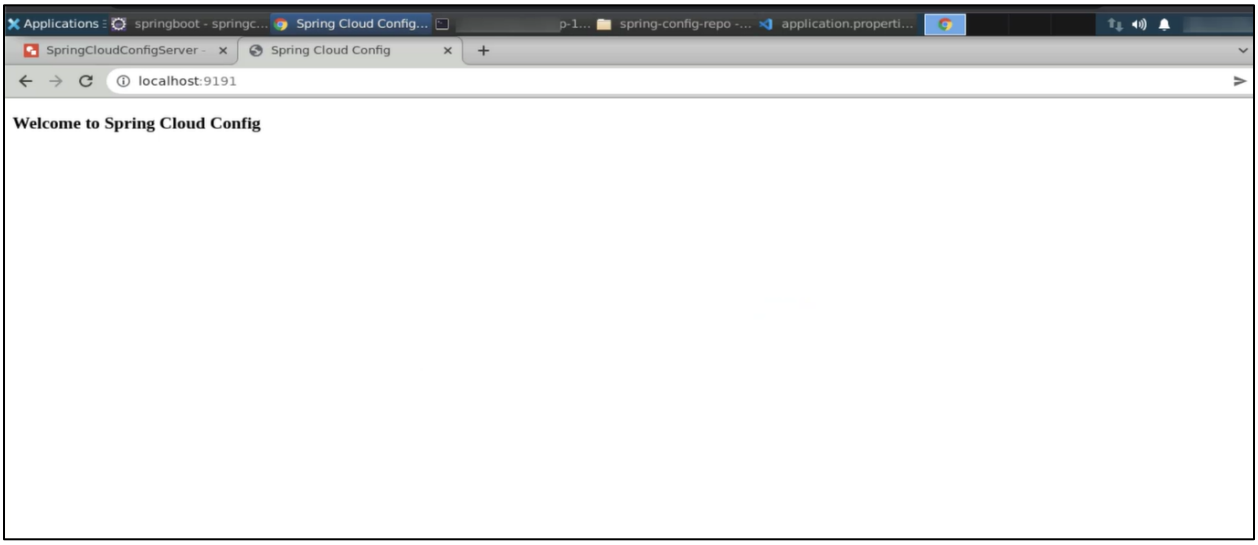
5.1 Run this Spring Cloud Config demo as a **Spring Boot App**. For this, **right-click** on the **springcloudconfigdemo** project and select **Run As > Spring Boot App**



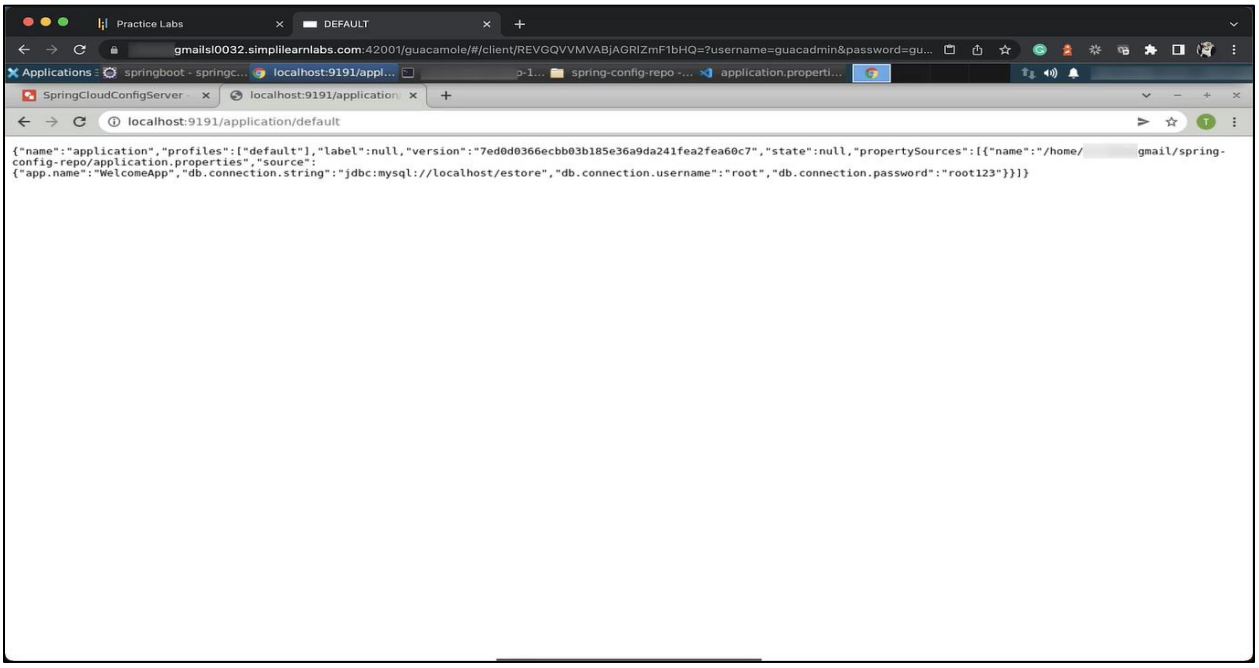
5.2 Notice that the **Tomcat server** starts on server port **9191**



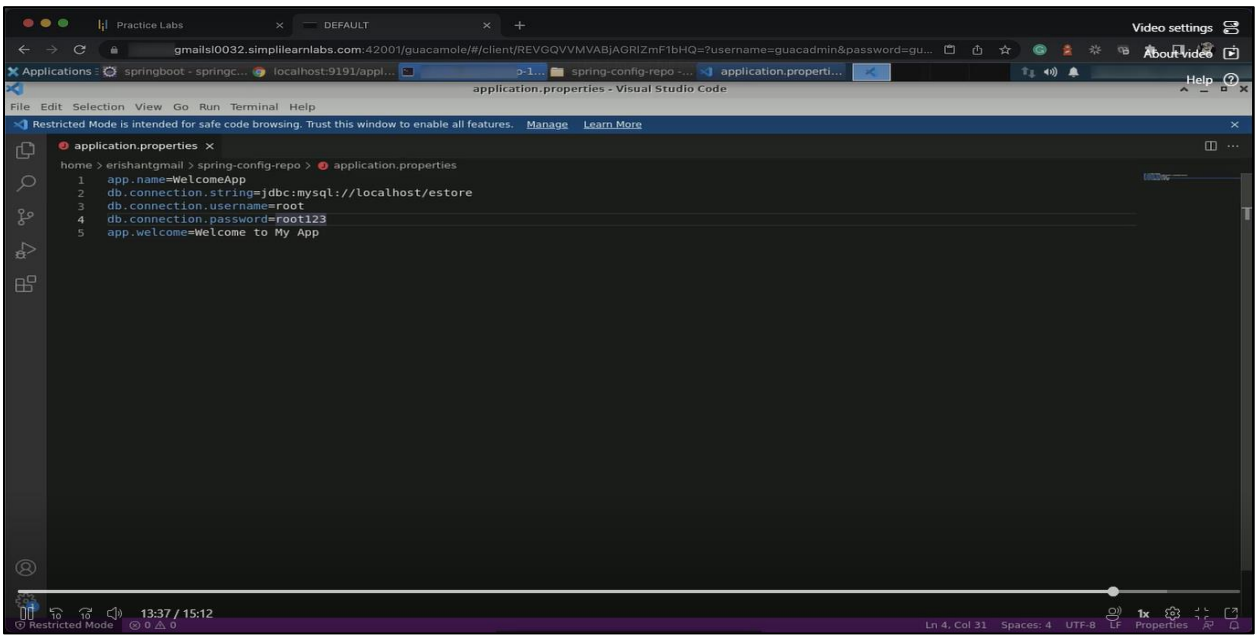
5.3 Now go to the browser and type **localhost:9191** to execute the web application



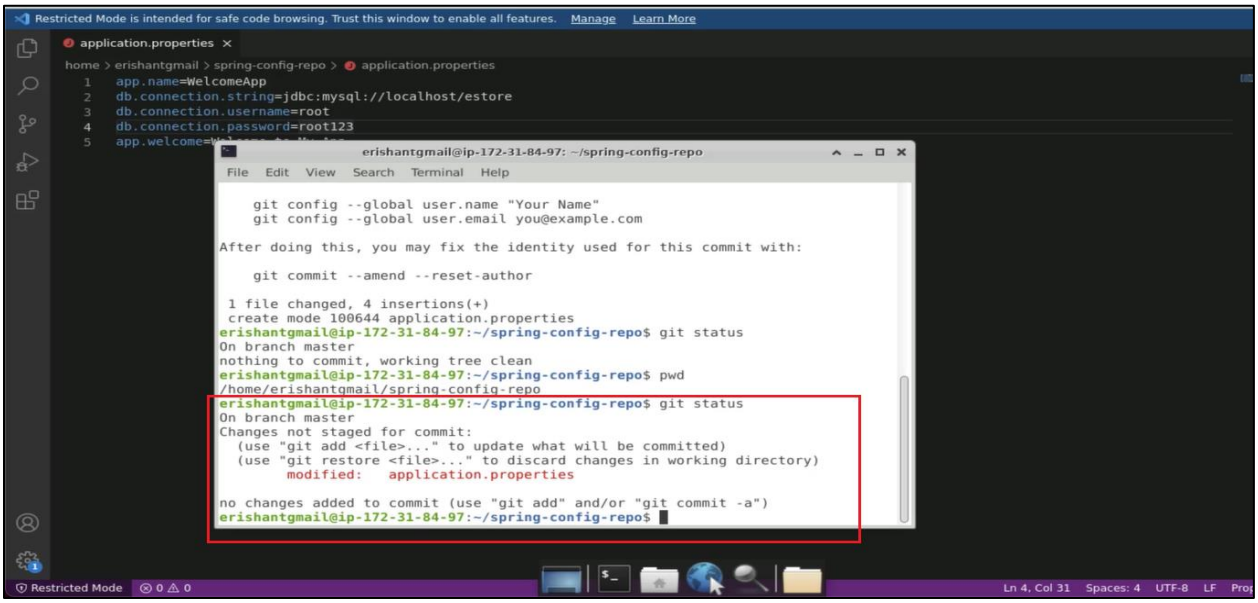
5.4 To check the properties of the application, type **localhost:9191/application/default** and click **Enter**



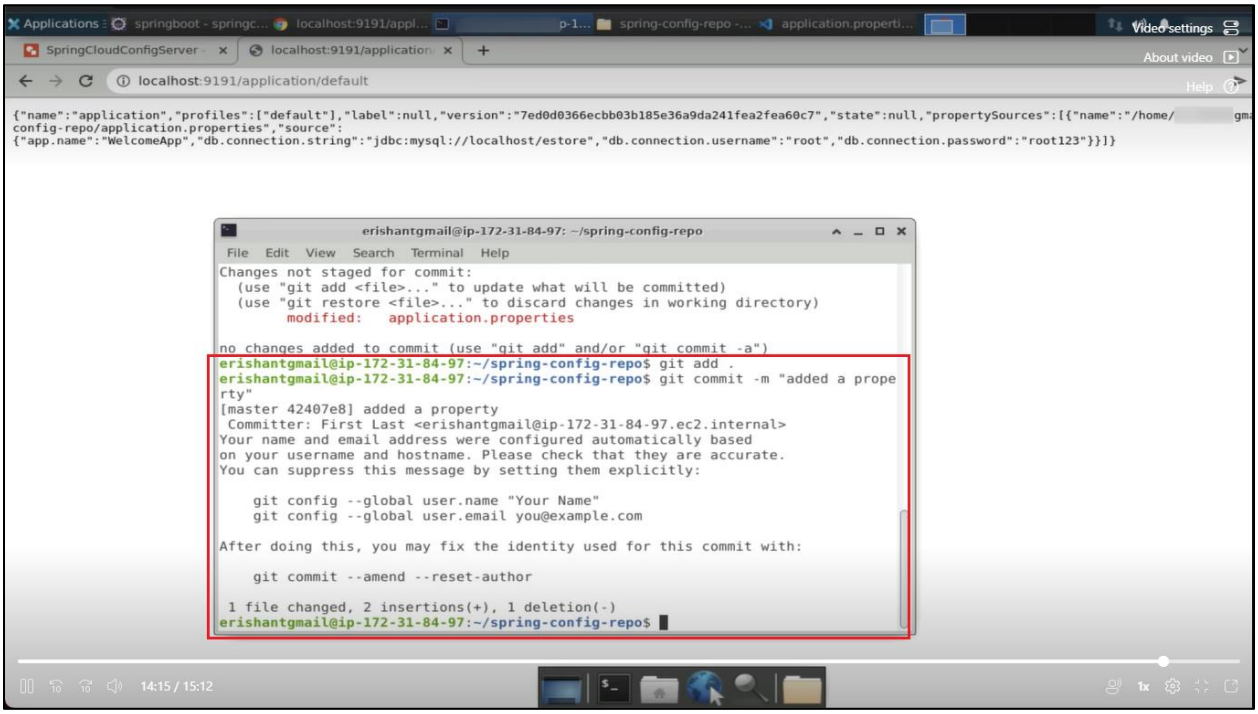
5.5 With **Spring Cloud Config**, any changes can be made to the application without deploying it again. Let's add another property, **app.welcome=Welcome to My App,** to the **application.properties** in Git repository



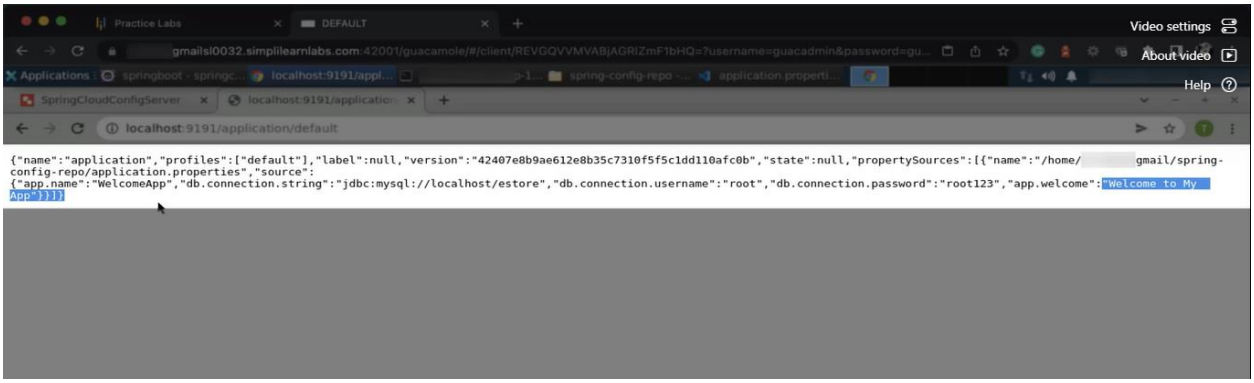
5.6 To check the modification status of your Git repository, use the **git status** command



5.7 This will not reflect any changes on the configuration server until we commit the changes to the Git repository. For this, use the **git add .** command to add changes and use the **git commit -m "added a property"** command to commit the changes



5.8 Now, hit the same URL **localhost:9191/application/default**. Now, the changes can be seen in the web application without deploying the microservices.



With this, you have successfully created a Spring Cloud Configuration that acts as a provider to the microservices of various configurations on the server side in a distributed environment.