**DOMAIN TRAINING PROJECT**

**Name: Aayushi Pandit**

**Department: SmartOps Fusion**

**Technologies Used:** HTML, CSS, Java

**Environments Used:** Spring boot, Eclipse, GitHub, Jenkin, Docker

* **Spring Boot:**

**What it is:** A framework for building Java-based web applications with minimal configuration. It simplifies the process of creating stand-alone, production-ready applications by providing built-in features such as embedded servers, dependency injection, and various out-of-the-box integrations.

**Why we use it:** It speeds up development, reduces boilerplate code, and is highly configurable. It's ideal for microservices, REST APIs, and scalable enterprise applications.

* **Eclipse:**

**What it is:** A popular Integrated Development Environment (IDE) primarily used for Java development, but it also supports other languages and technologies through plugins.

**Why we use it:** It offers powerful tools for code editing, debugging, and managing complex projects, along with a large ecosystem of plugins for additional functionality.

* **GitHub:**

**What it is:** A cloud-based platform for version control and collaborative software development, using Git. It hosts code repositories, allows for code collaboration, issue tracking, and pull requests.

**Why we use it:** It enables distributed version control, collaboration among developers, code sharing, and tracking changes over time. It also integrates well with Continuous Integration/Continuous Deployment (CI/CD) pipelines.

* **Jenkins:**

**What it is:** An open-source automation server that supports building, testing, and deploying code in a CI/CD pipeline. It integrates with various tools and allows automated workflows.

**Why we use it:** It automates repetitive tasks such as code building, testing, and deployment, which improves software quality, reduces manual errors, and speeds up the development lifecycle.

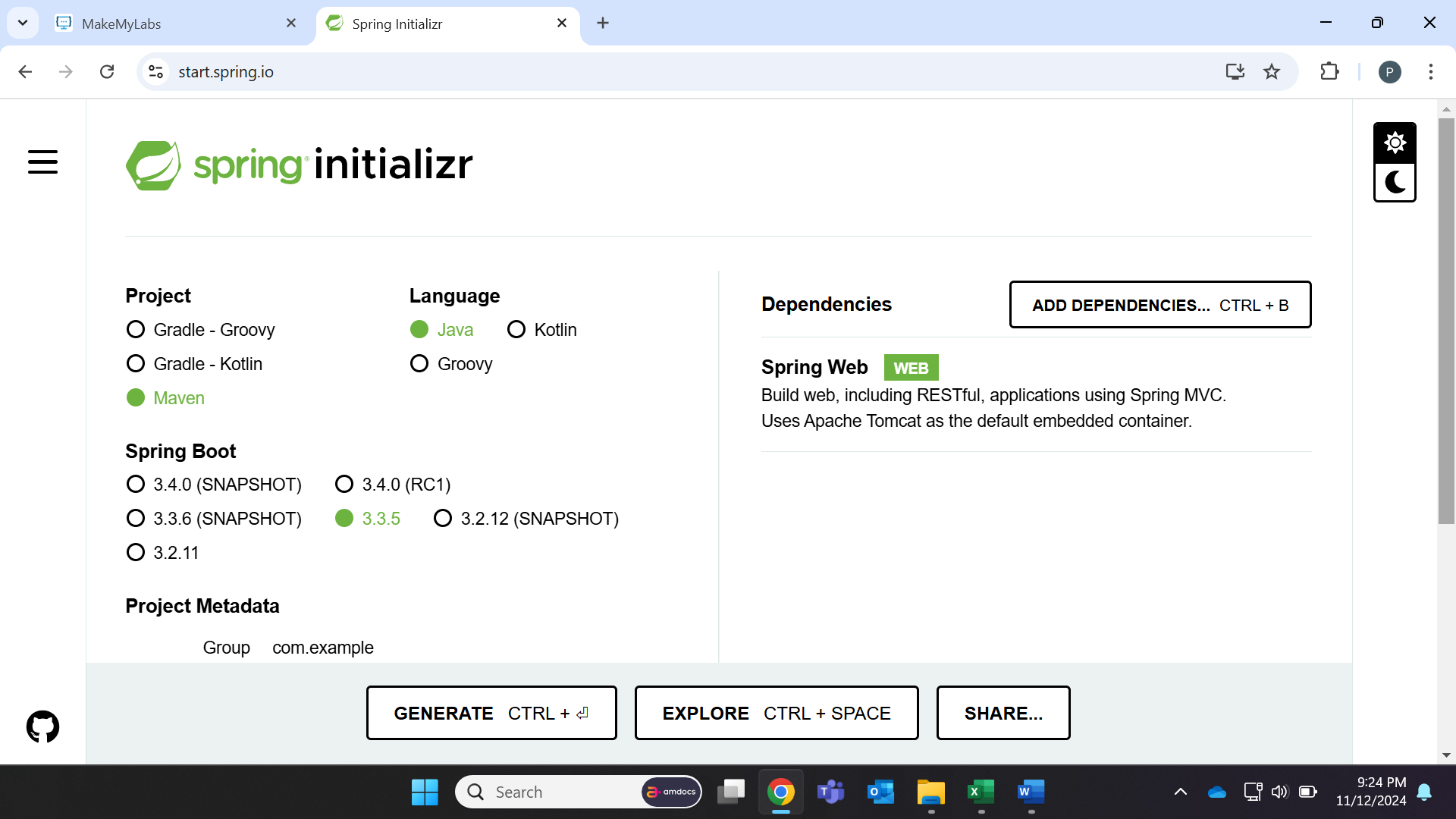
* **Docker:**

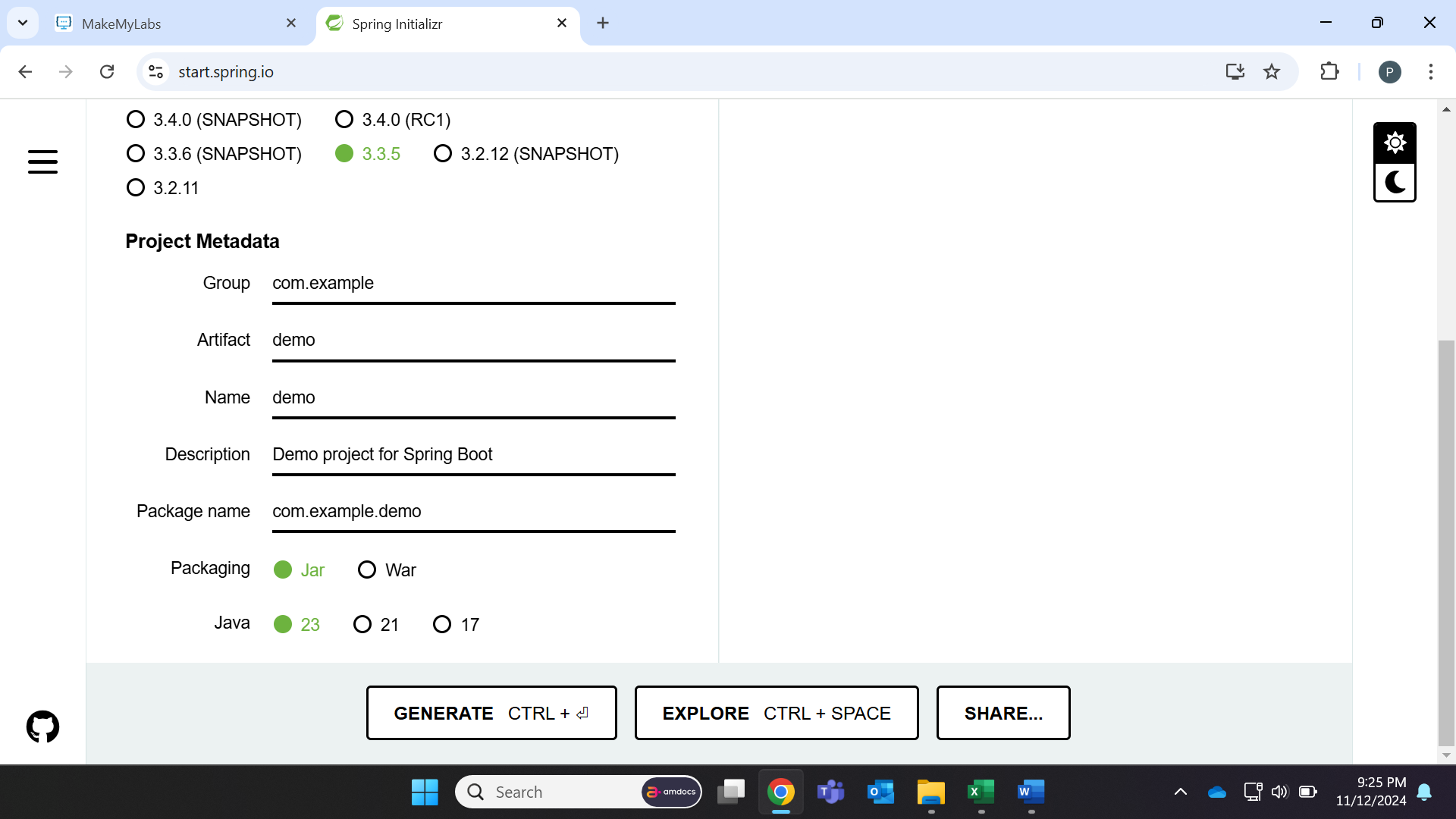
**What it is:** A platform for developing, shipping, and running applications in containers. Containers are lightweight, portable, and ensure that the application runs consistently across different environments.

**Why we use it:** Docker simplifies deployment, ensures consistency between development and production environments, and allows for scalability and isolation of services, which is especially useful in microservices architectures.

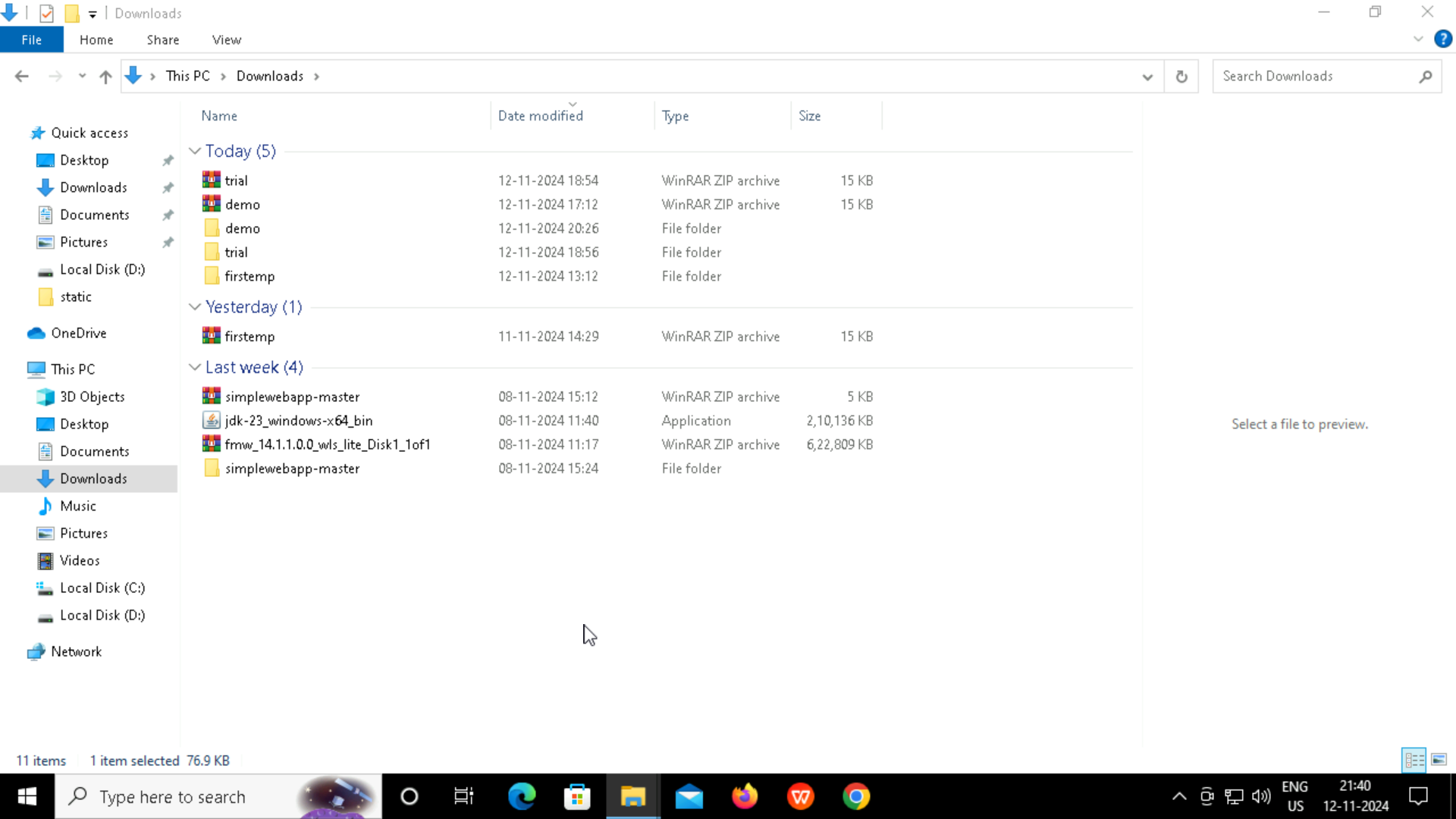
**PROJECT IMPLEMENTATION**

**Step-1: Open Spring boot Initializer:** [**https://start.spring.io/**](https://start.spring.io/) **then select below details and click on generate.**

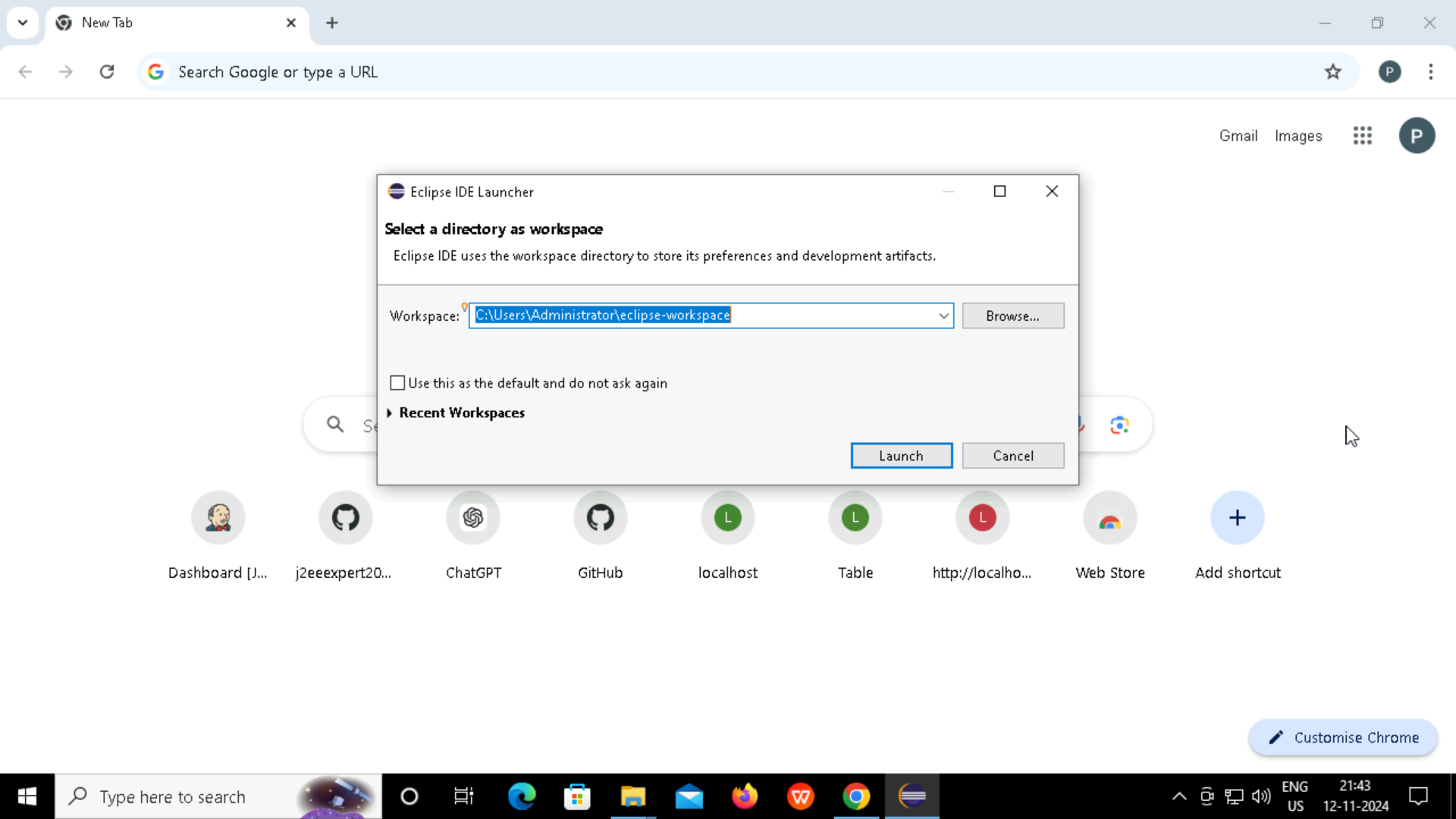




**Step-2: Open the downloaded file and extract it.**

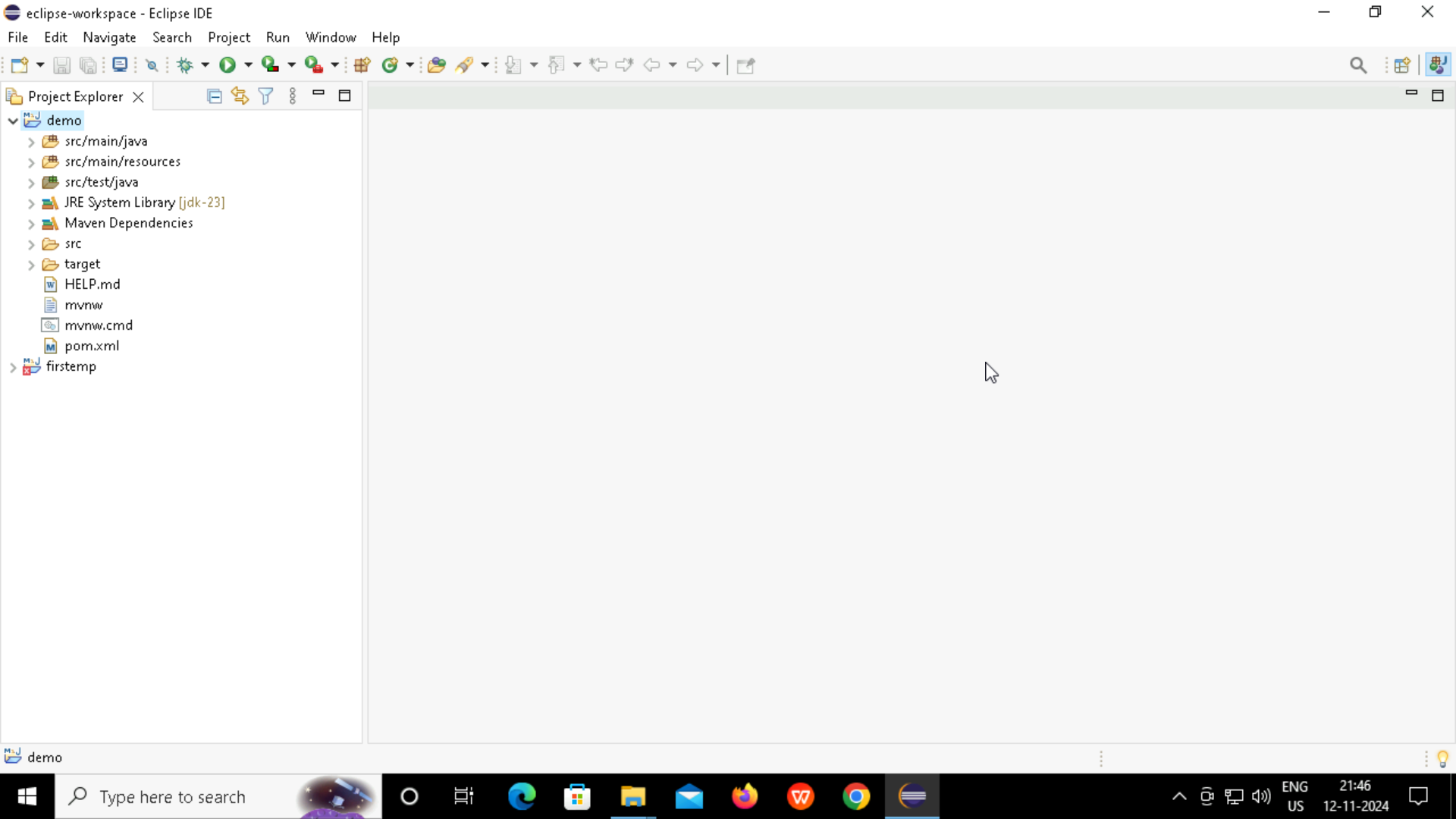


**Step-3: Open Eclipse IDE, select suitable directory and then click on Launch**

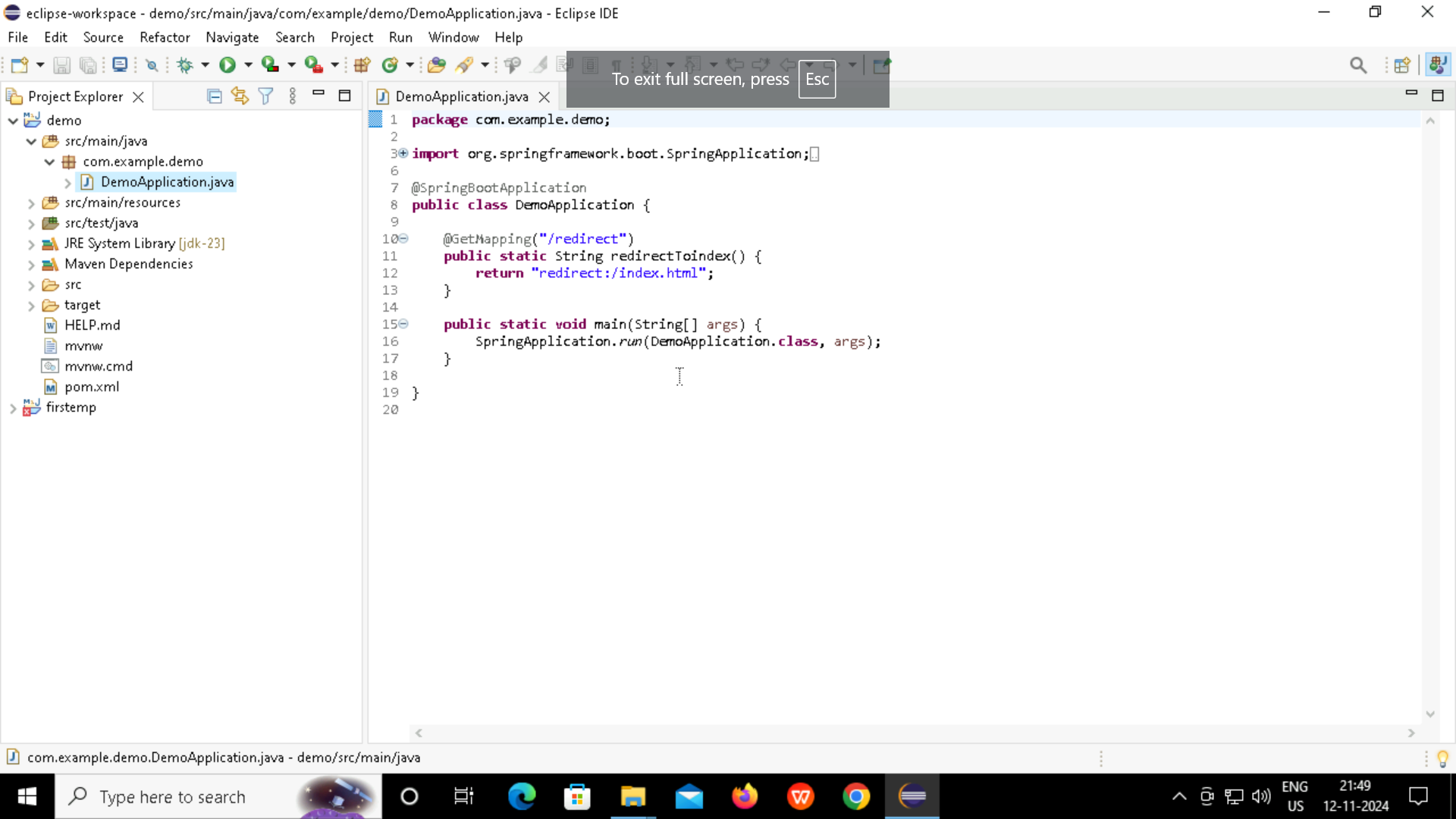


**Step-4: Go to File -> Import - > Existing Marvin Projects -> Select Folder -> Finish.**

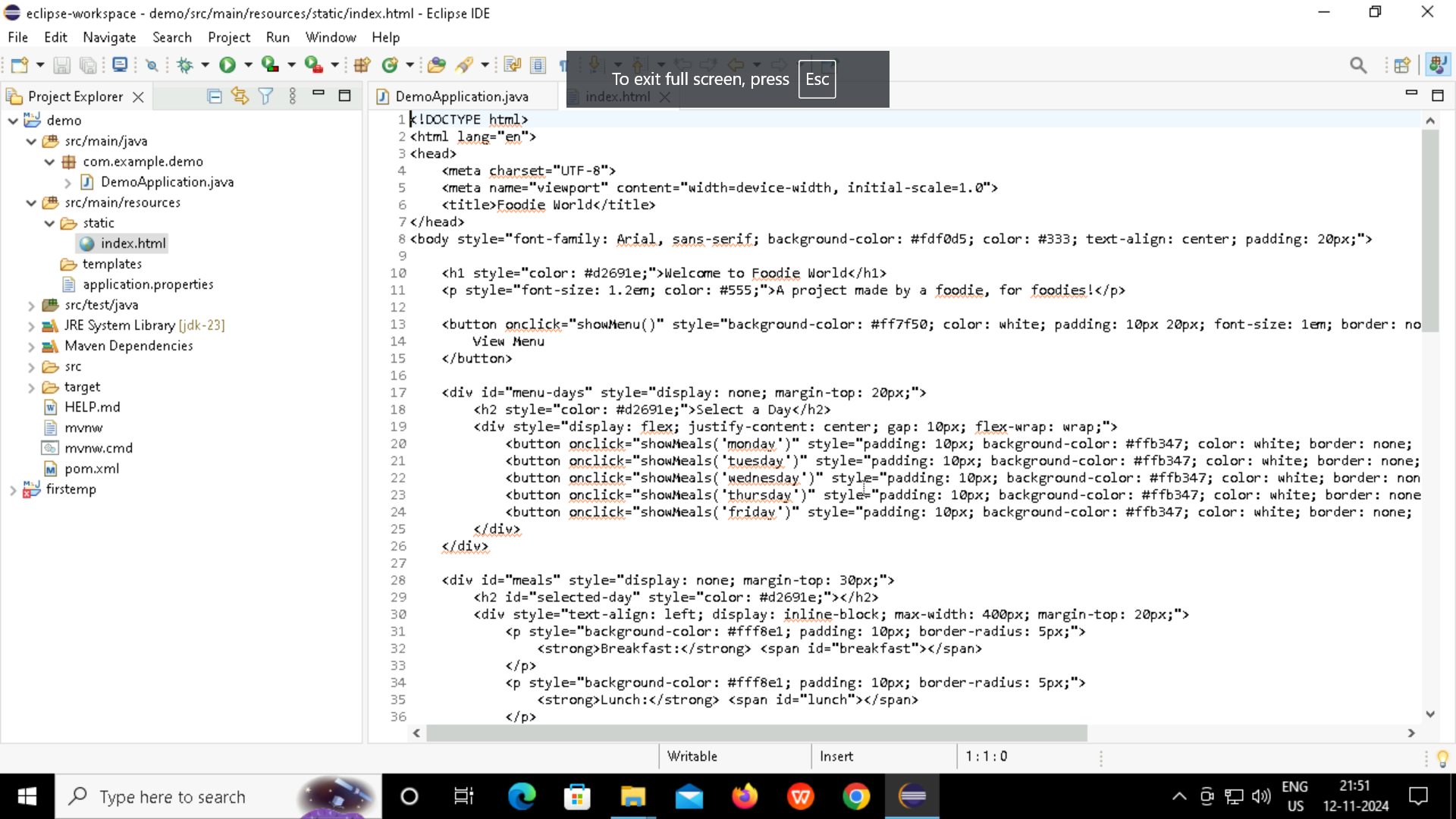
**This page should appear ones imported successfully...**



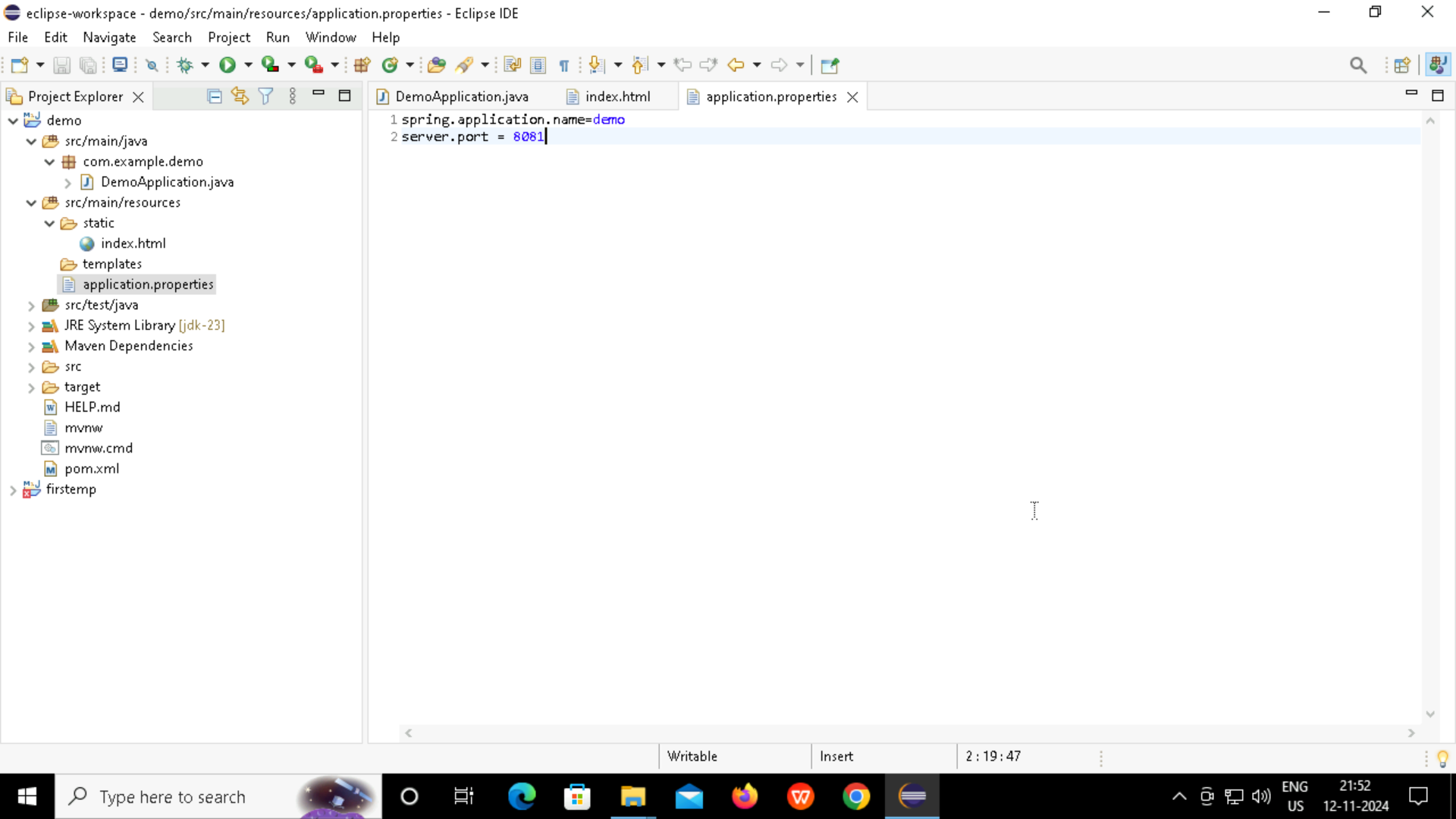
**Step-5: Go to folder: src/main/java -> com.example.demo -> DemoApplication.java and add code following code for mapping of link.**



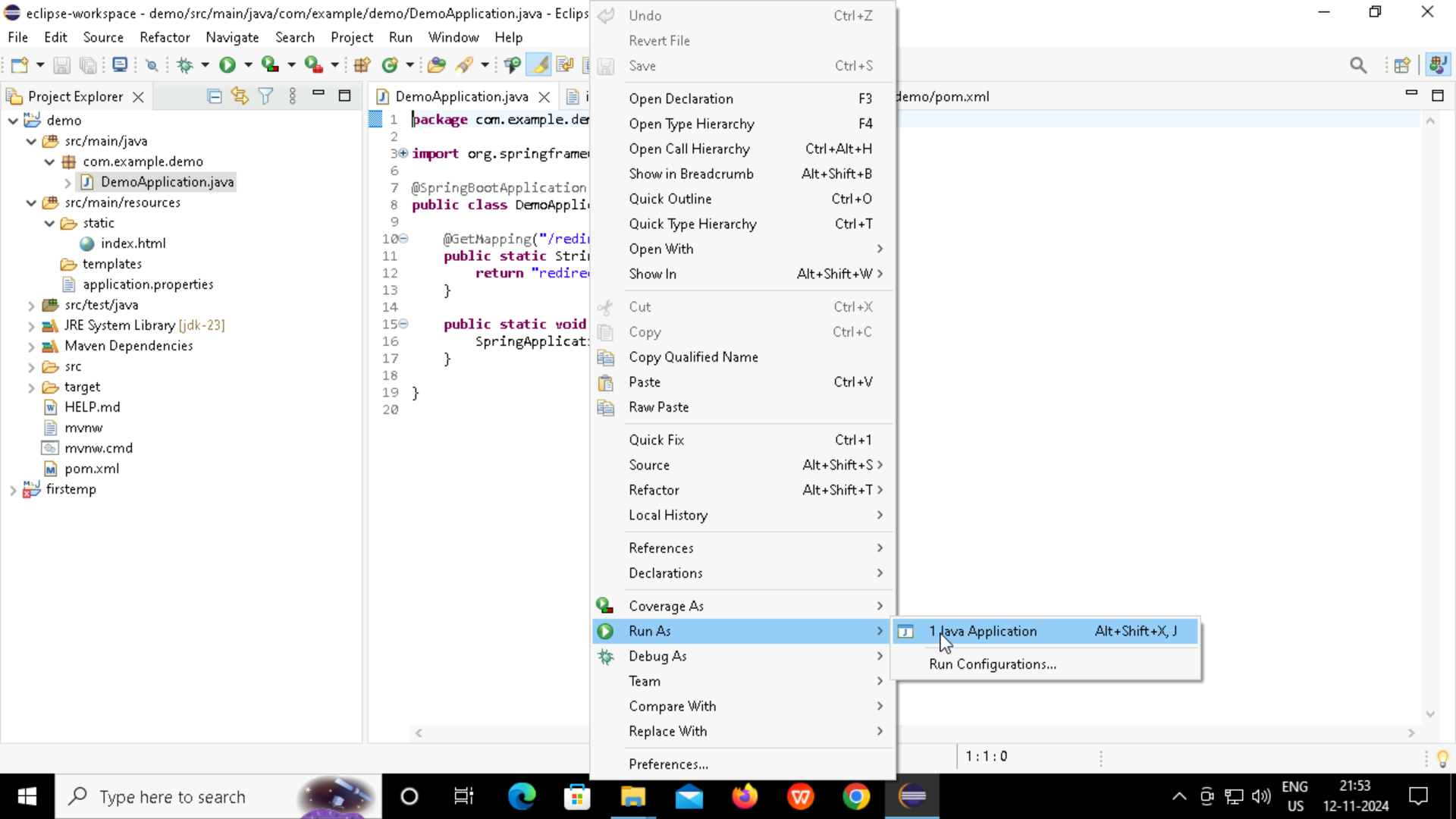
**Step-6: Go to folder: src/main/resources -> static. Add new file index.html and write code in HTML for webpage you like to view.**



**Step-7: Under same folder go to application.properties and add code for port number.**



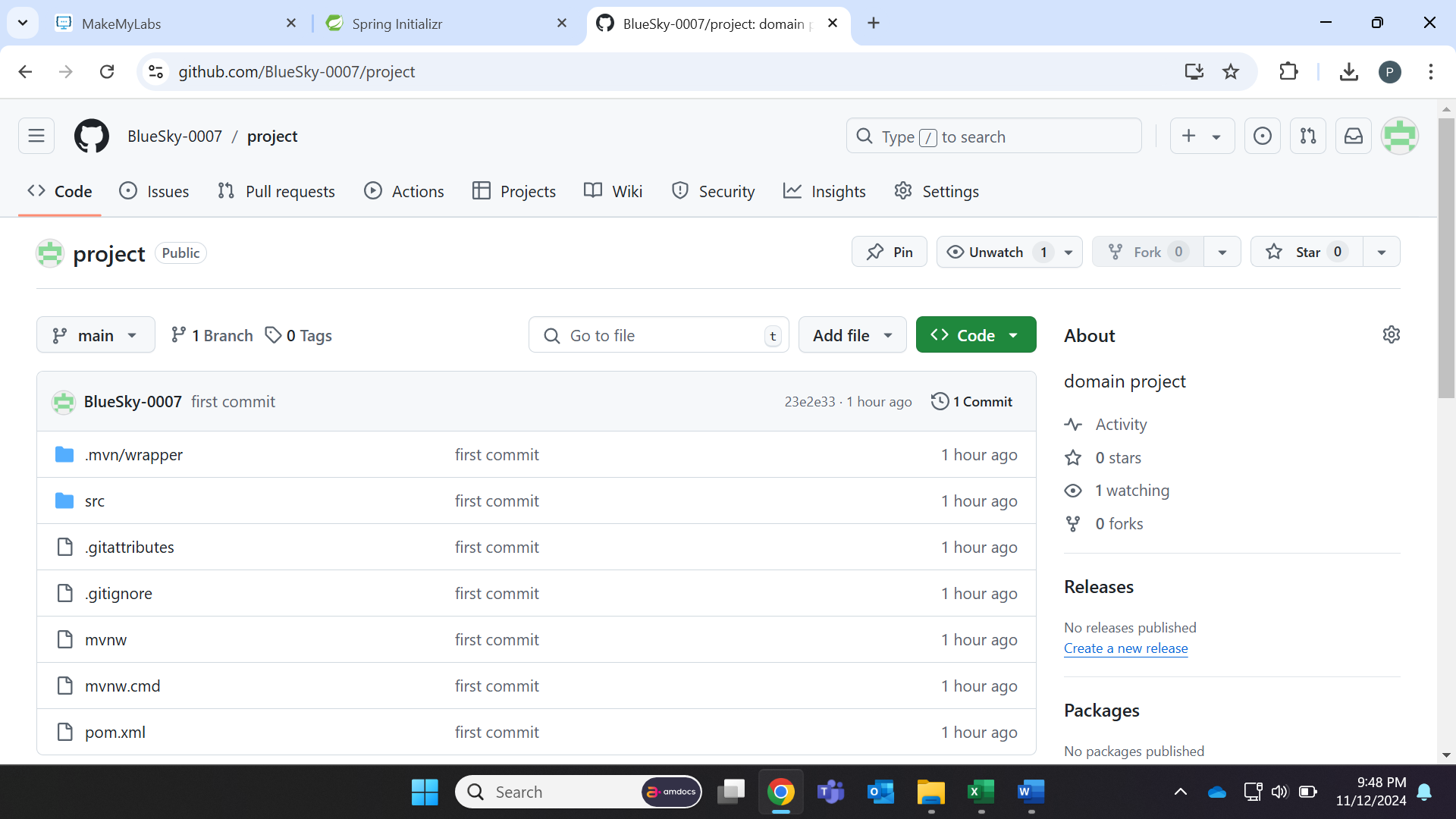
**Step-8: Go to DemoApplication.java, right click: then go to Run as -> Java Application and wait for it to load.**



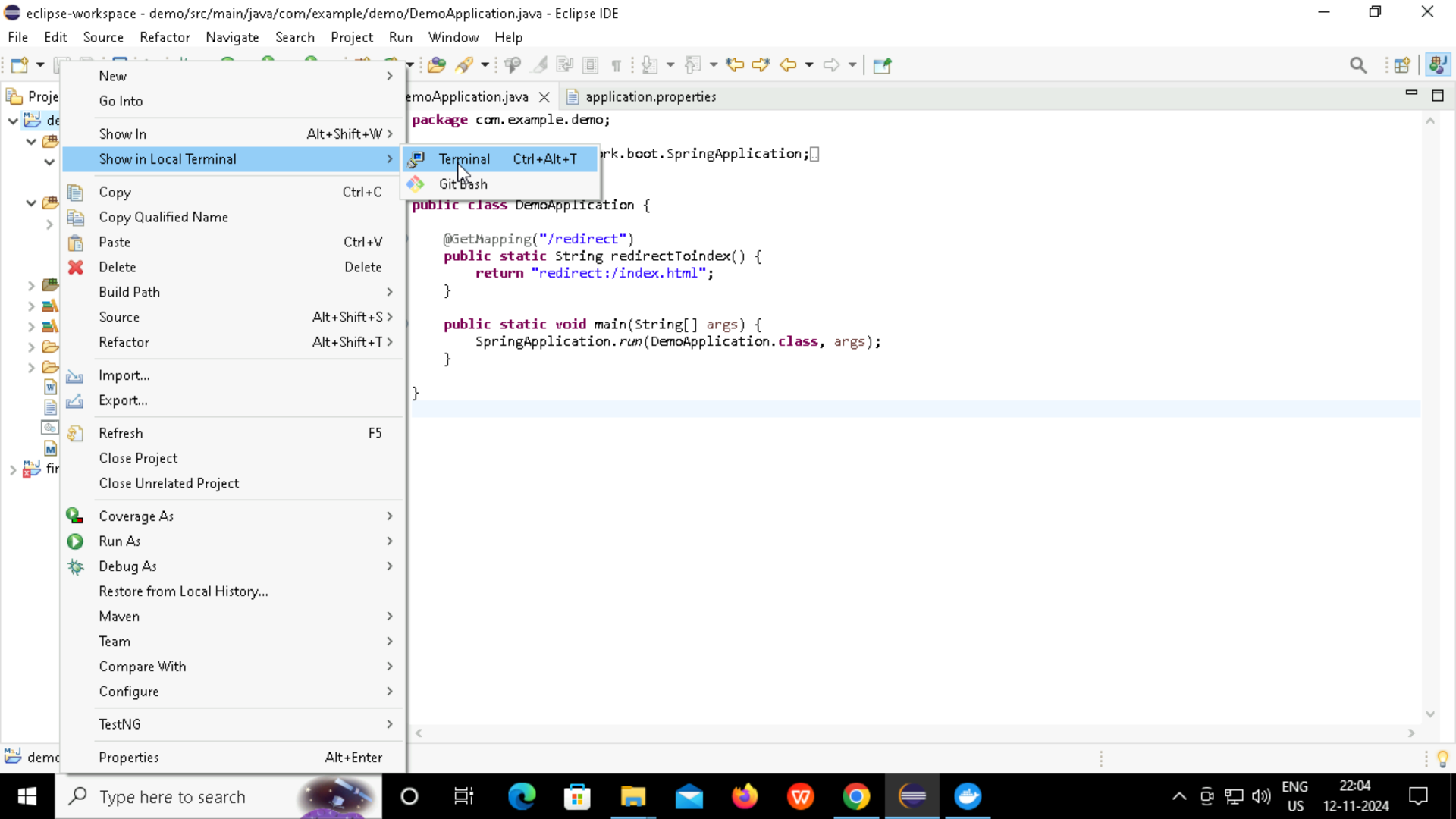
**Step-9: On Google Chrome type localhost:portnumber In this case, localhost:8082**



**Step-10: Go to GitHub and create a repository. Make sure to keep it public.**



**Step-11: Go to Eclipse then right click on folder -> Show in Local Terminal -> Terminal**



**Step-12: Type git commands to push files from Eclipse to created Git repository.**

**Commands:**

git init

git config –global user.name “ ”

git config –global user.email “ “

git status

git add .

git status

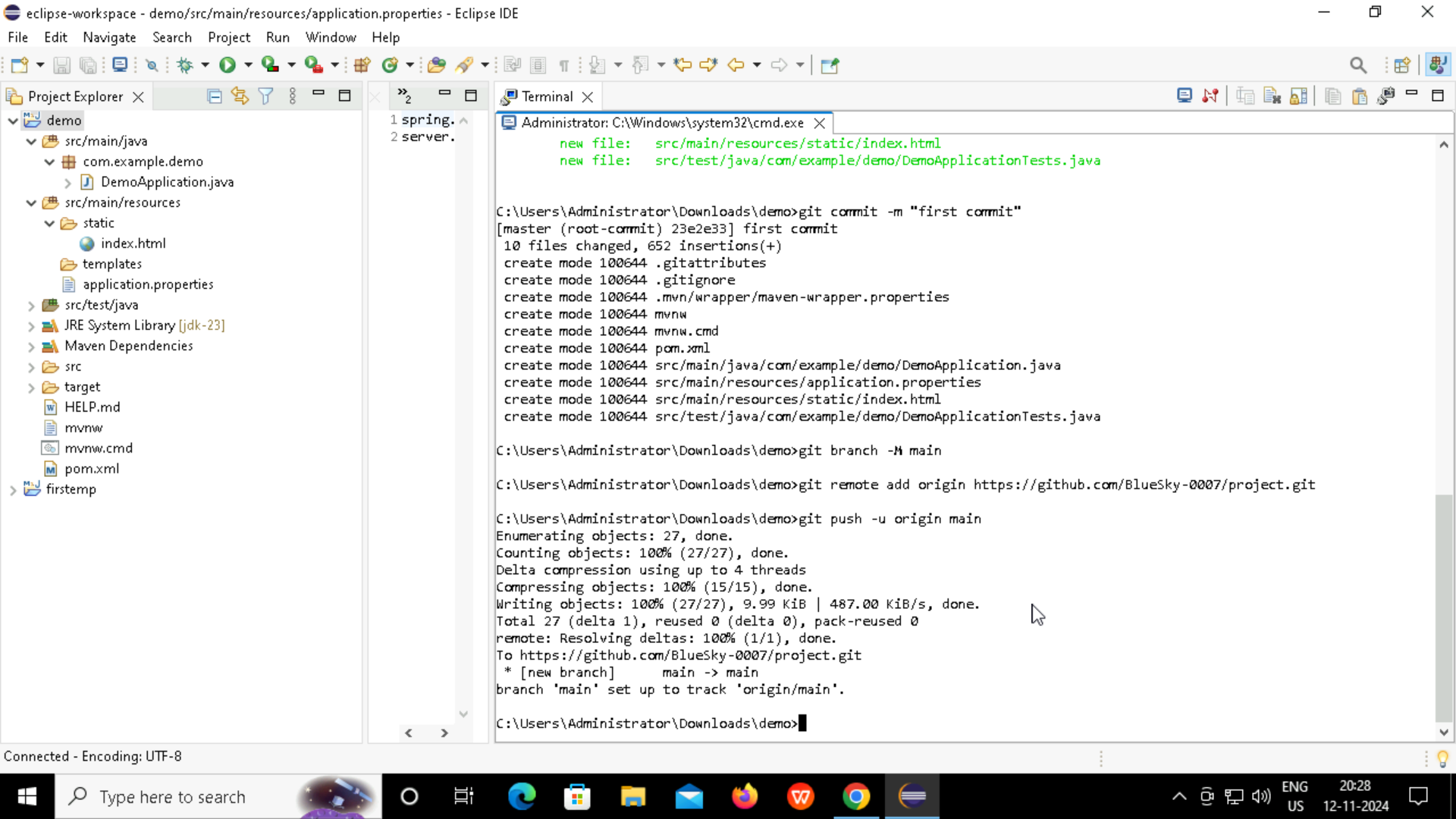
git commit -m “first commit”

git commit -M main

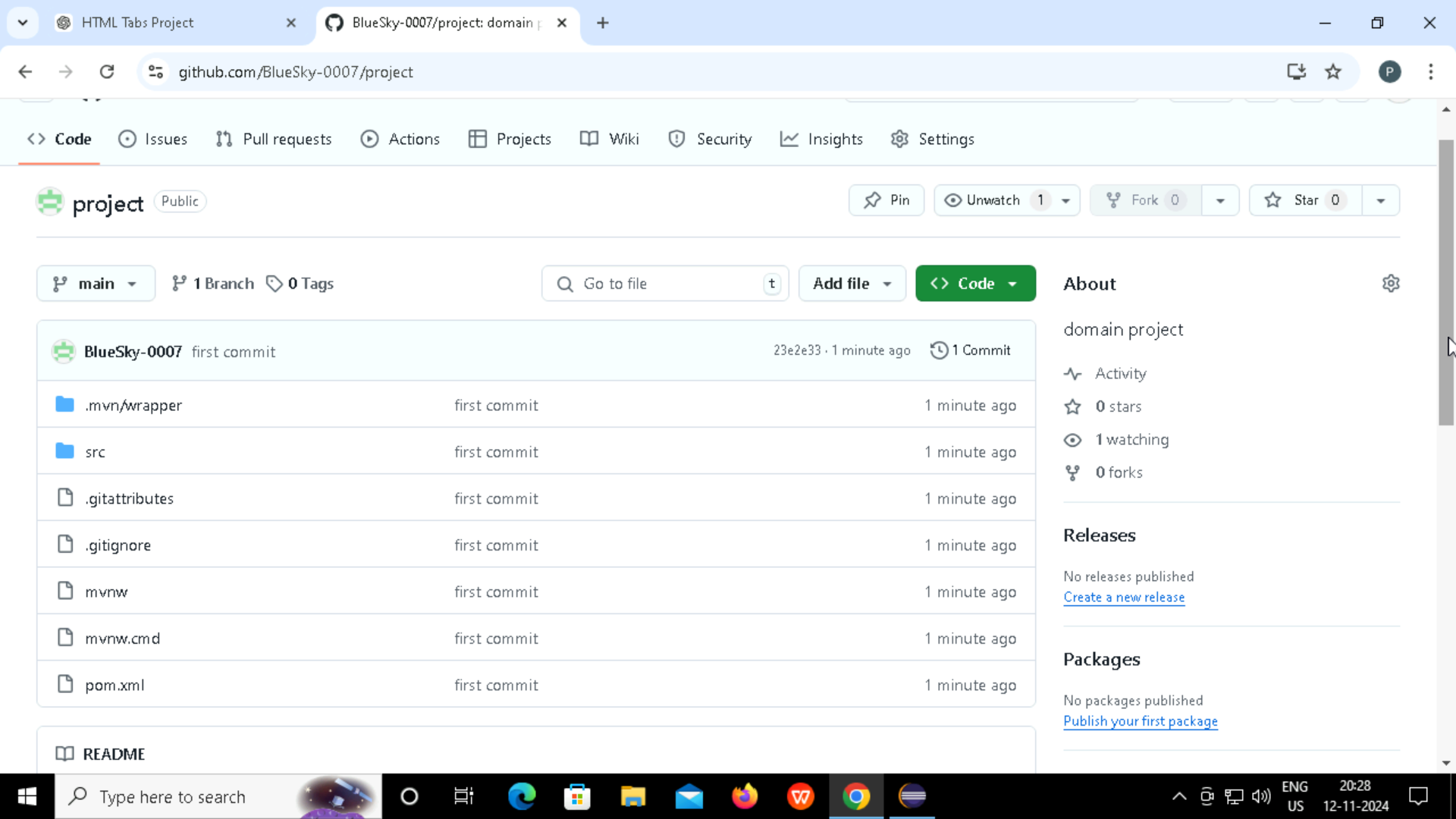
git remote add origin git\_repository\_url

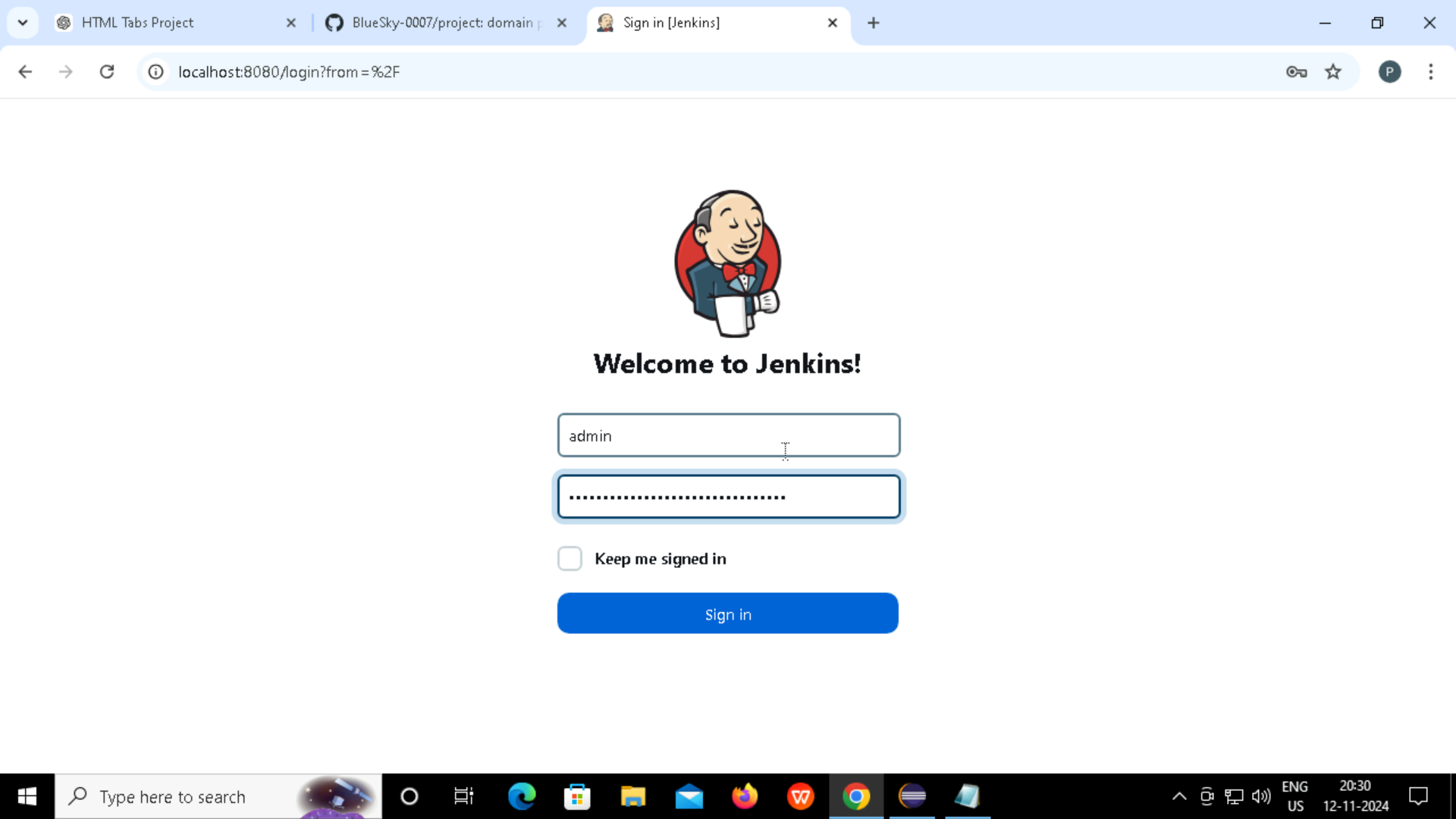
git push -u origin main

**Below page appears when pushed successfully…**

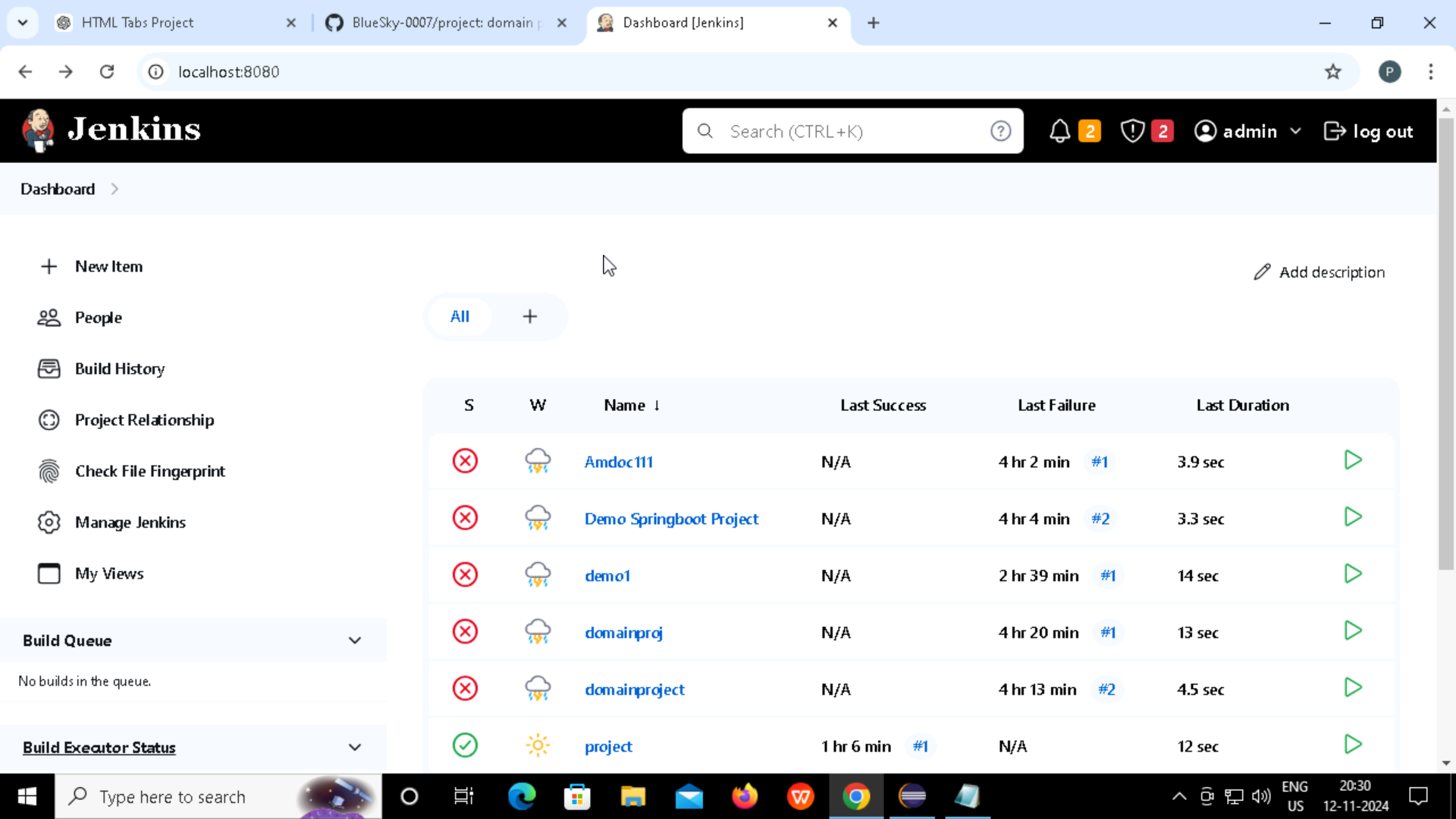


**Step-13: Go to GitHub account and check if you can see the files.**

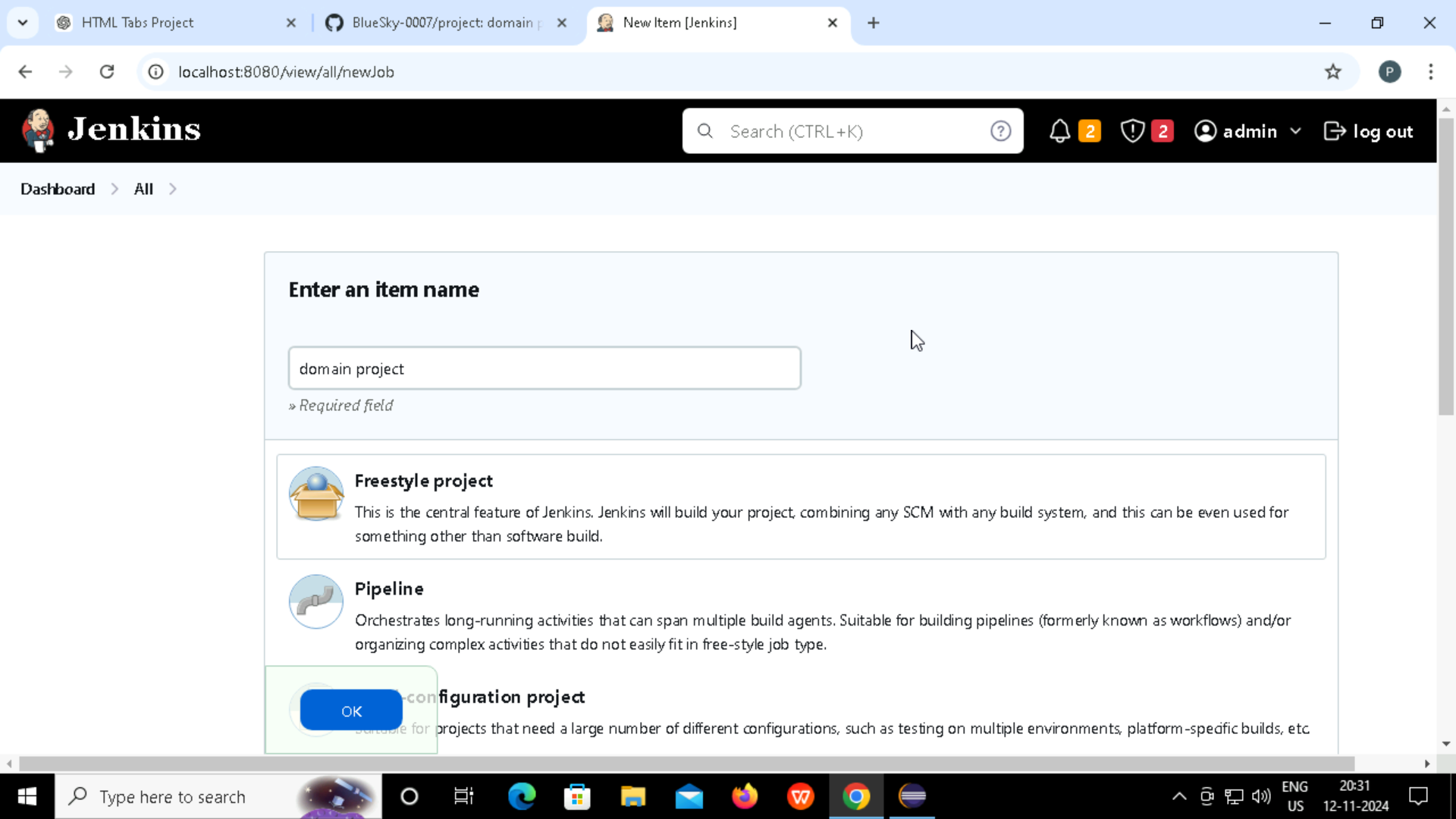


**Step-14: Once done, open Jenkins and enter username and password.**

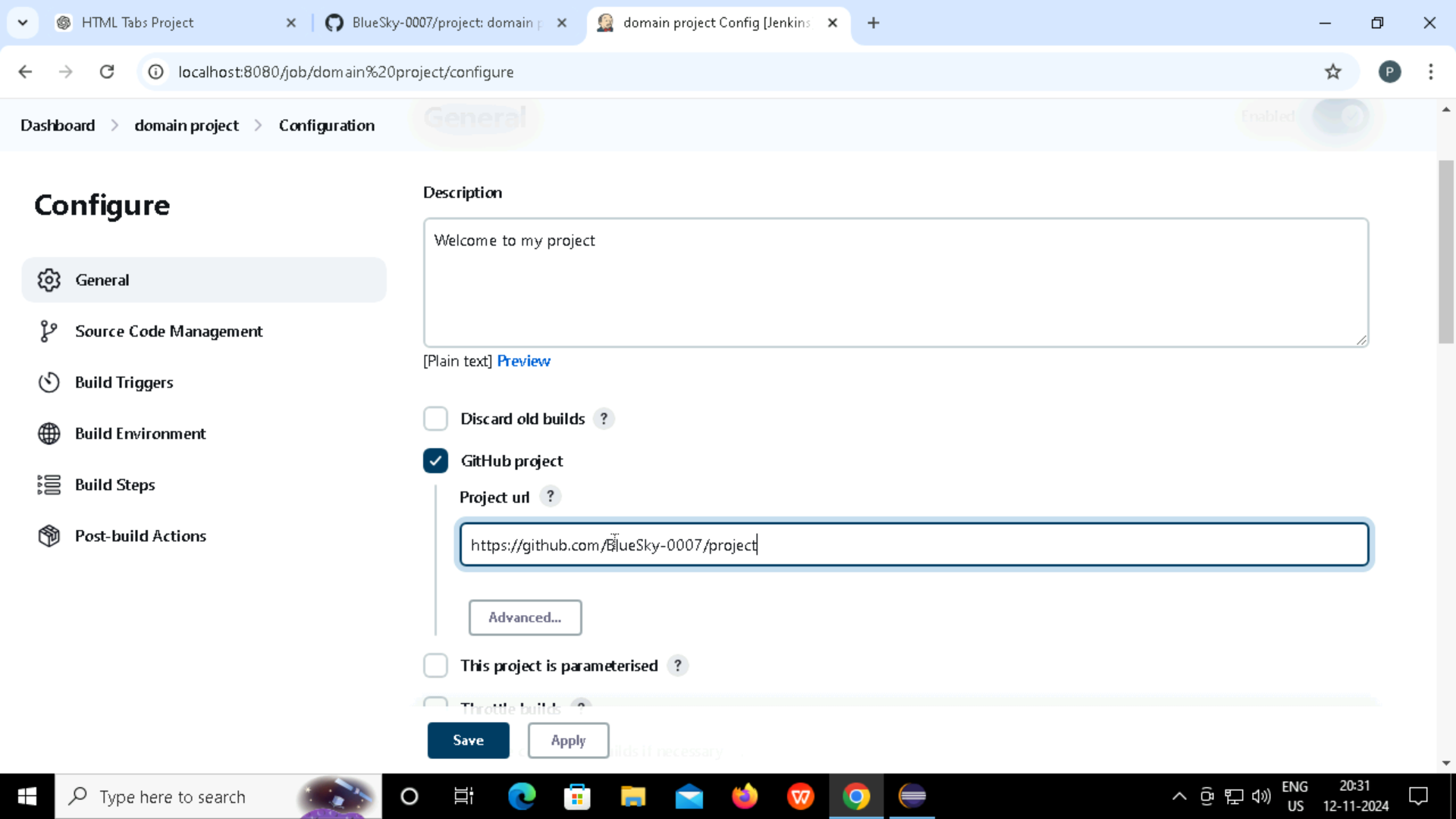
**Step-15: Go to New Items**

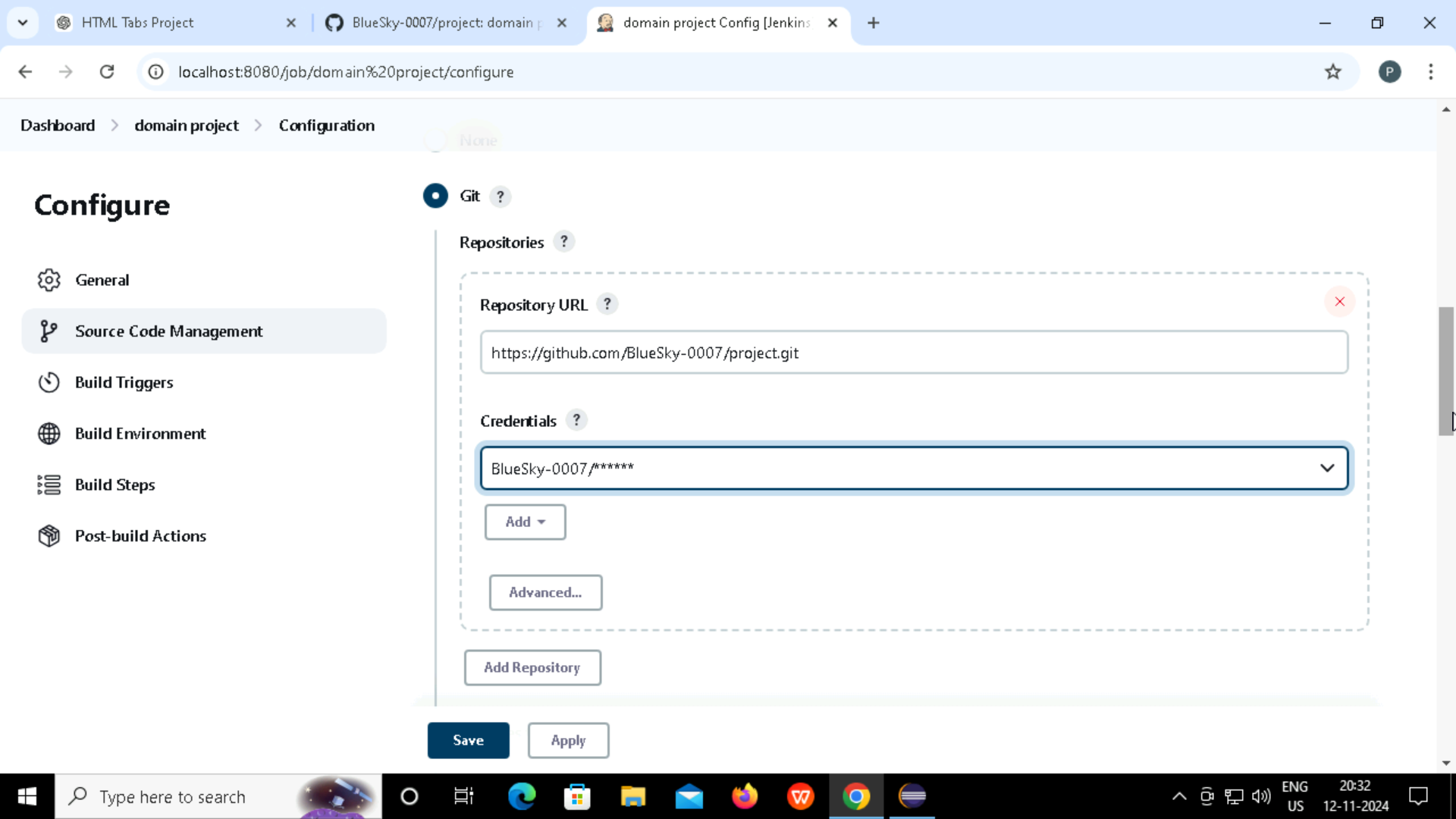


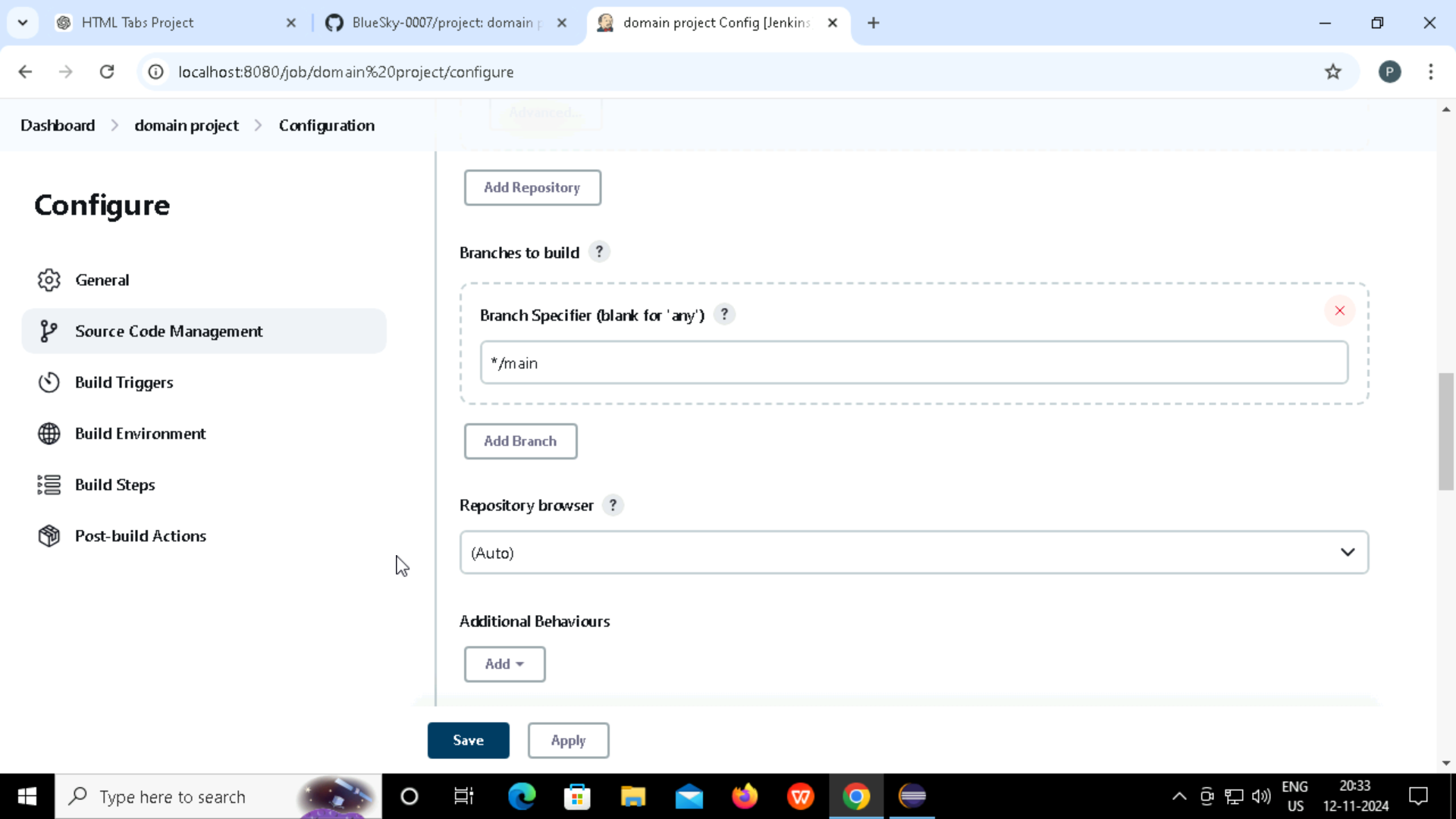
**Step-16: Enter domain name and select Freestyle Project**

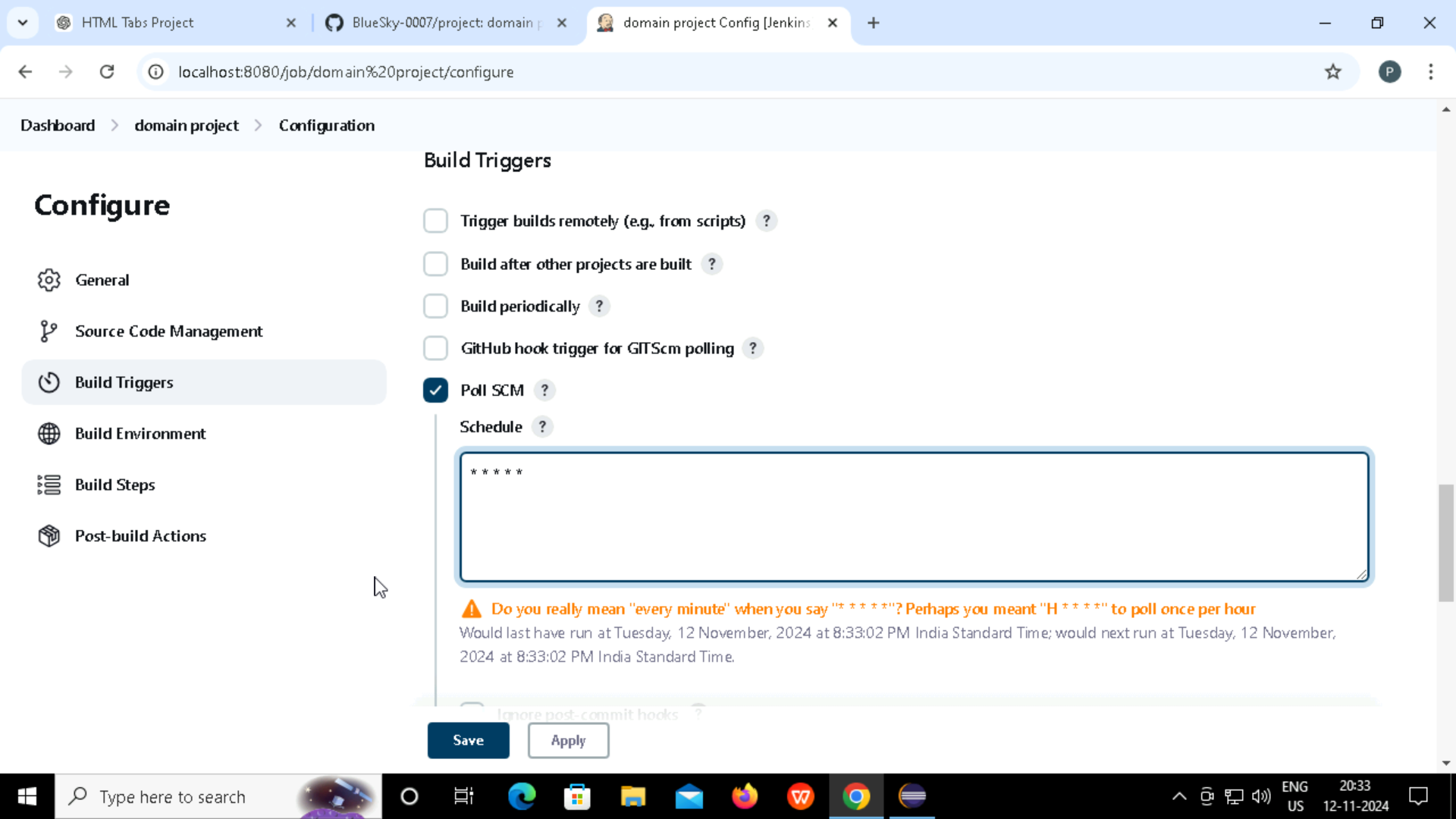


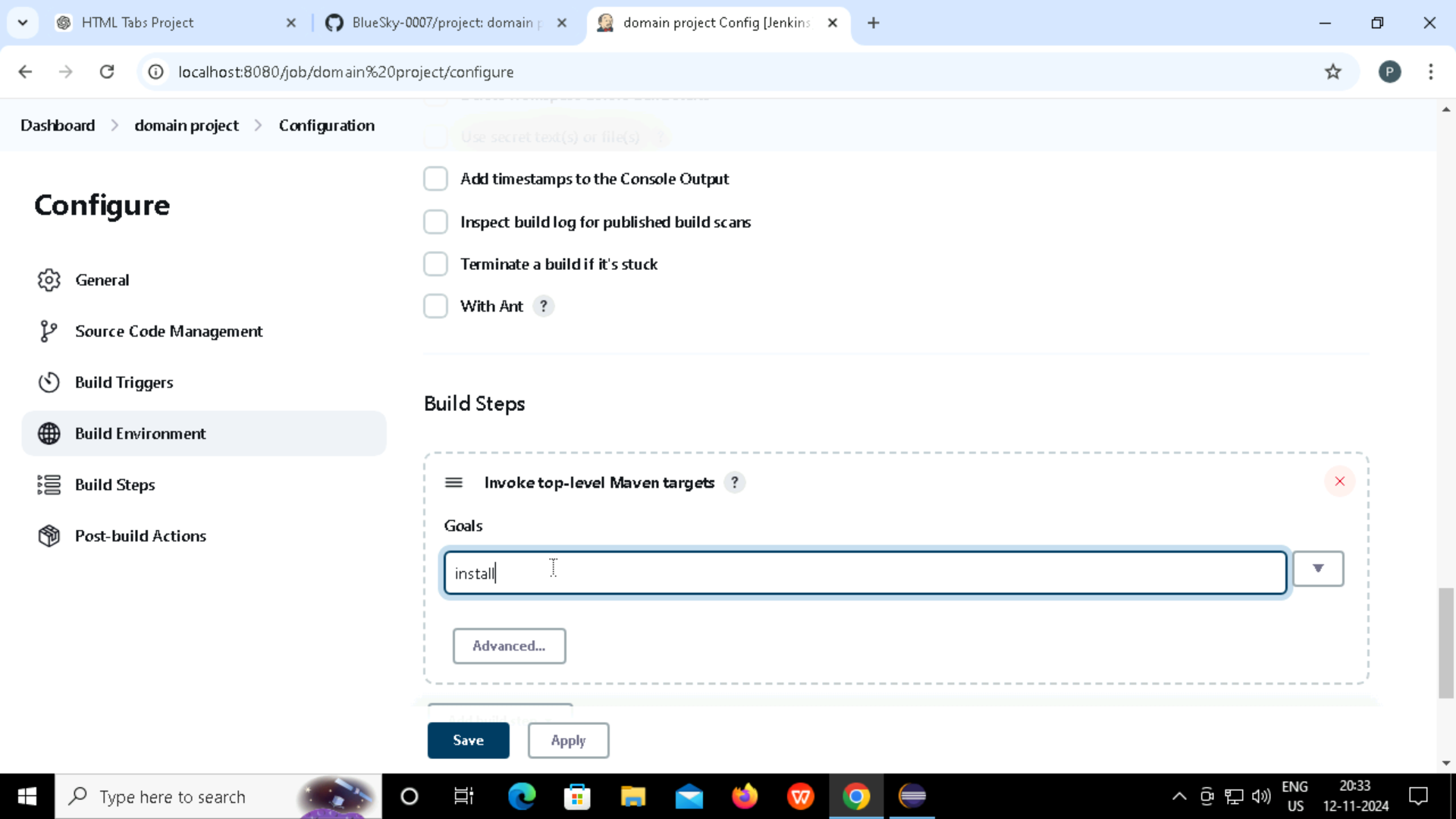
**Step-17: Follow and enter below details:**



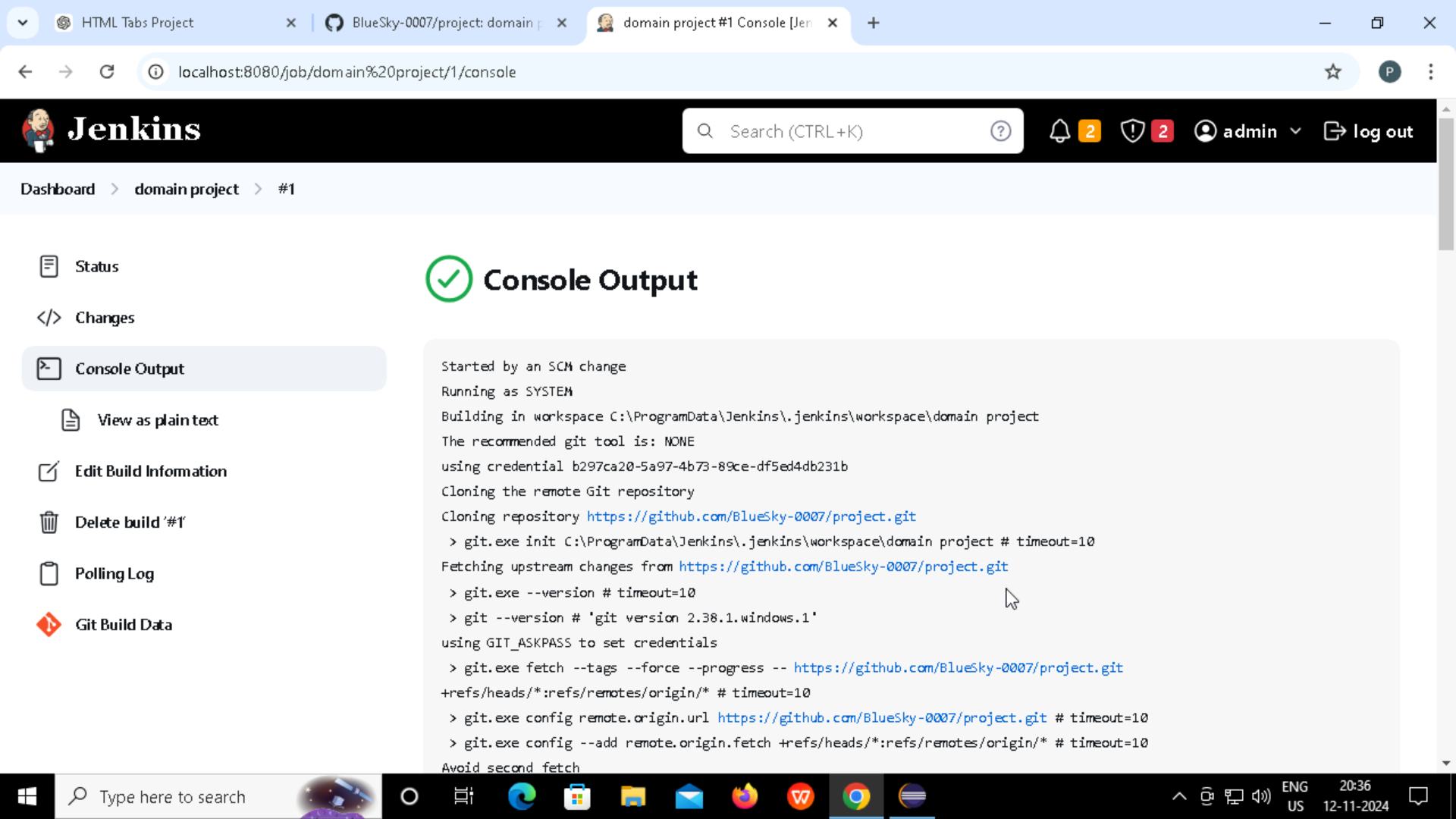


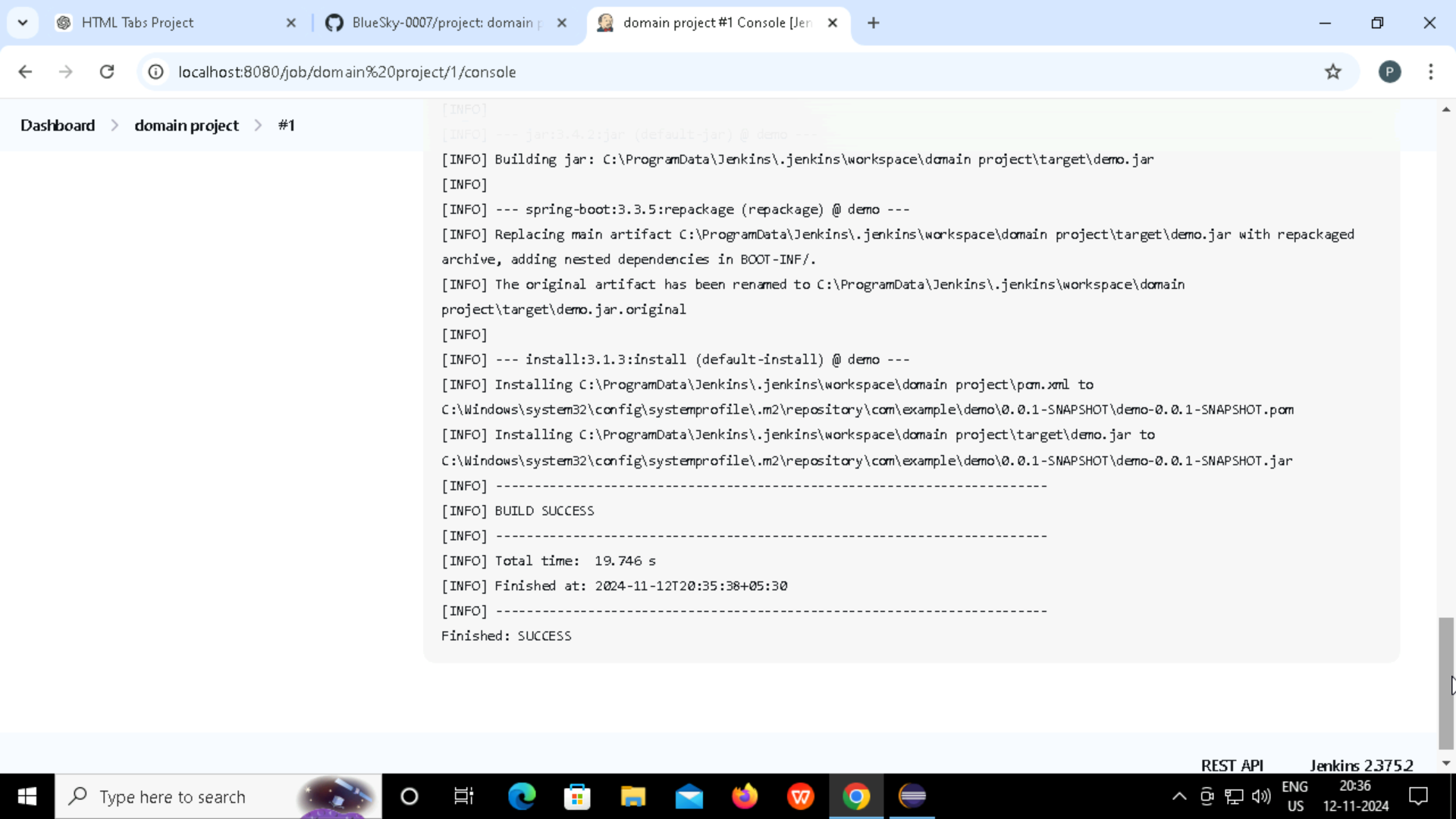




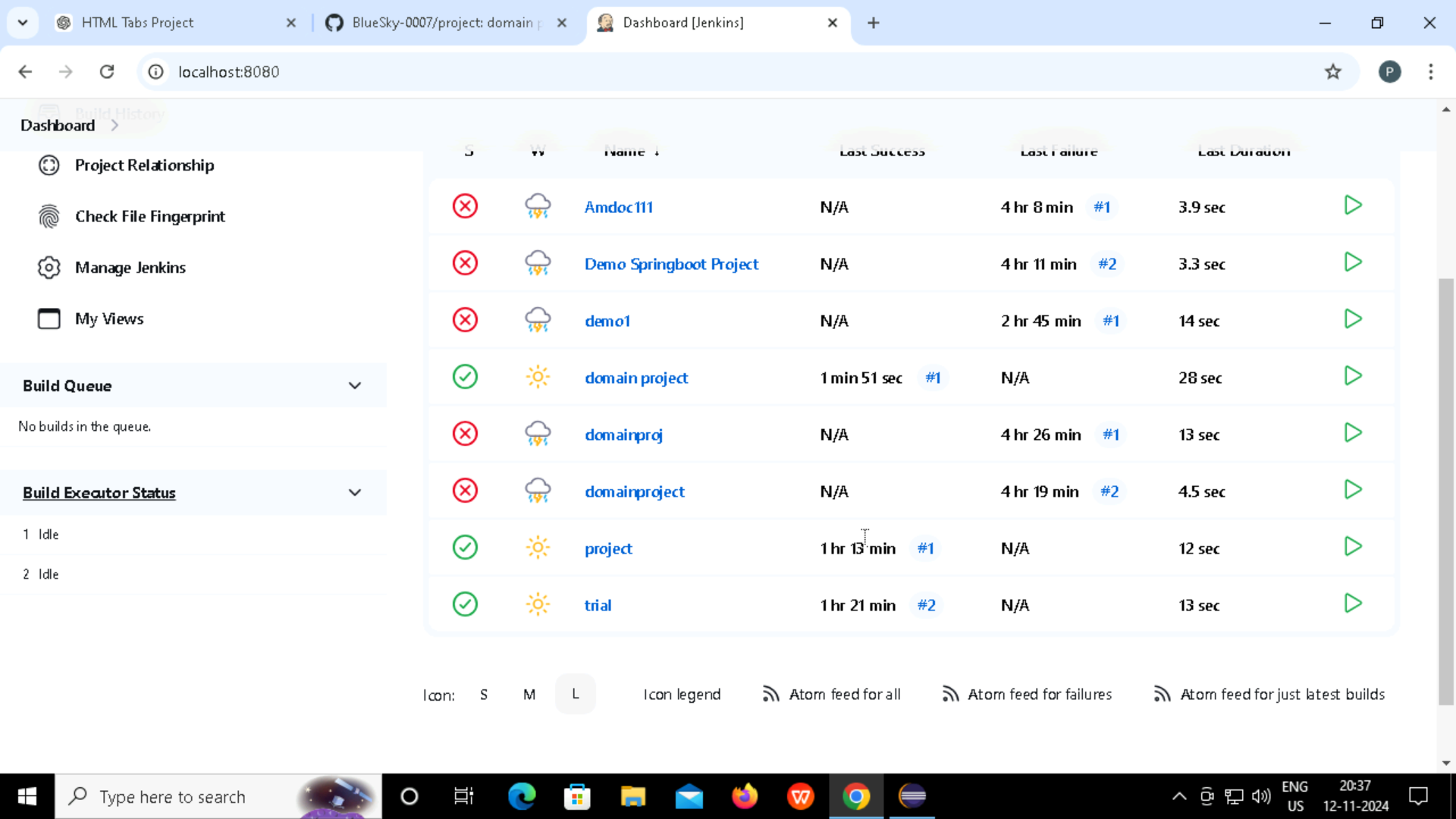


**Step-18: After clicking on Save, console output appears.**

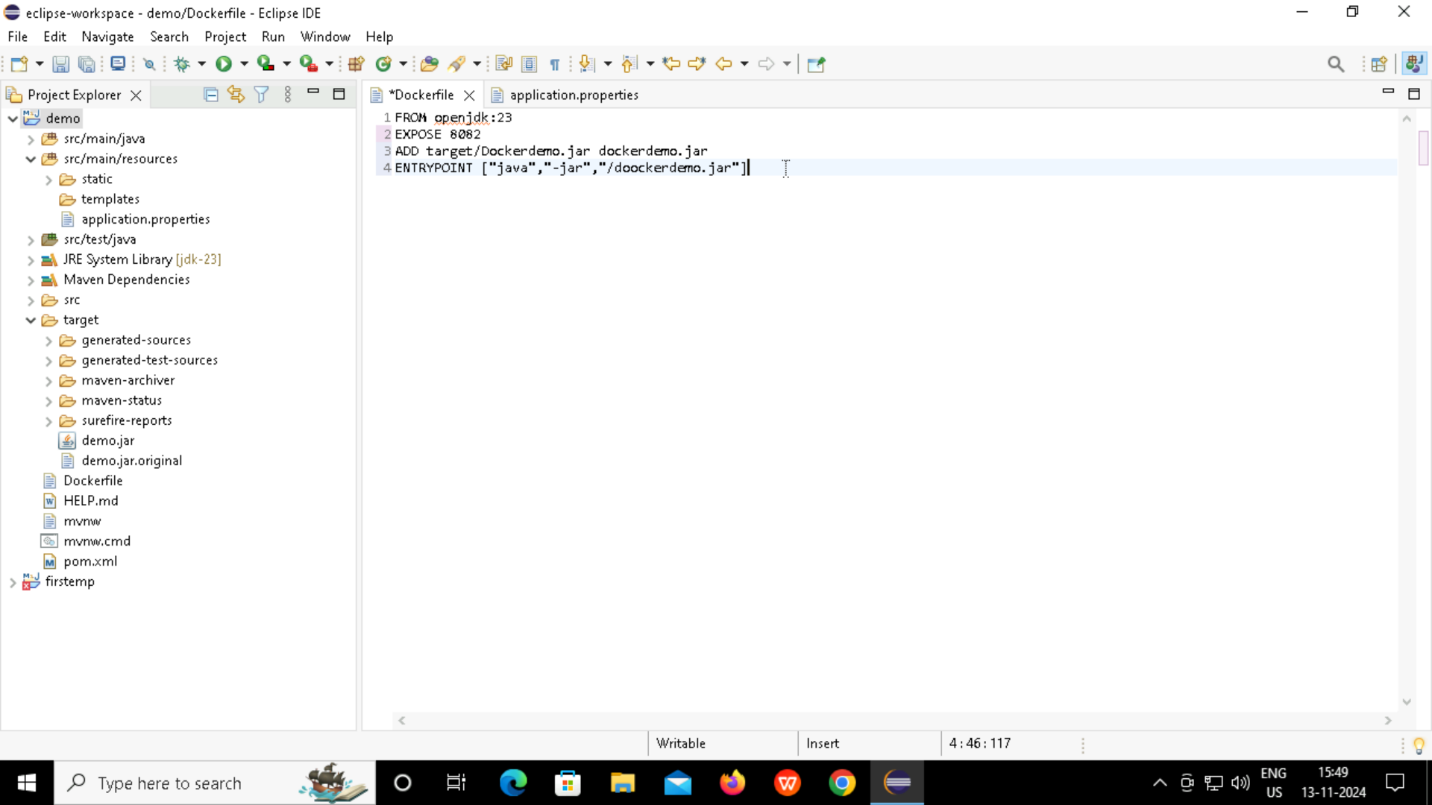




**Step-19: Go to Dashboard and re-check if project is deployed successfully…**

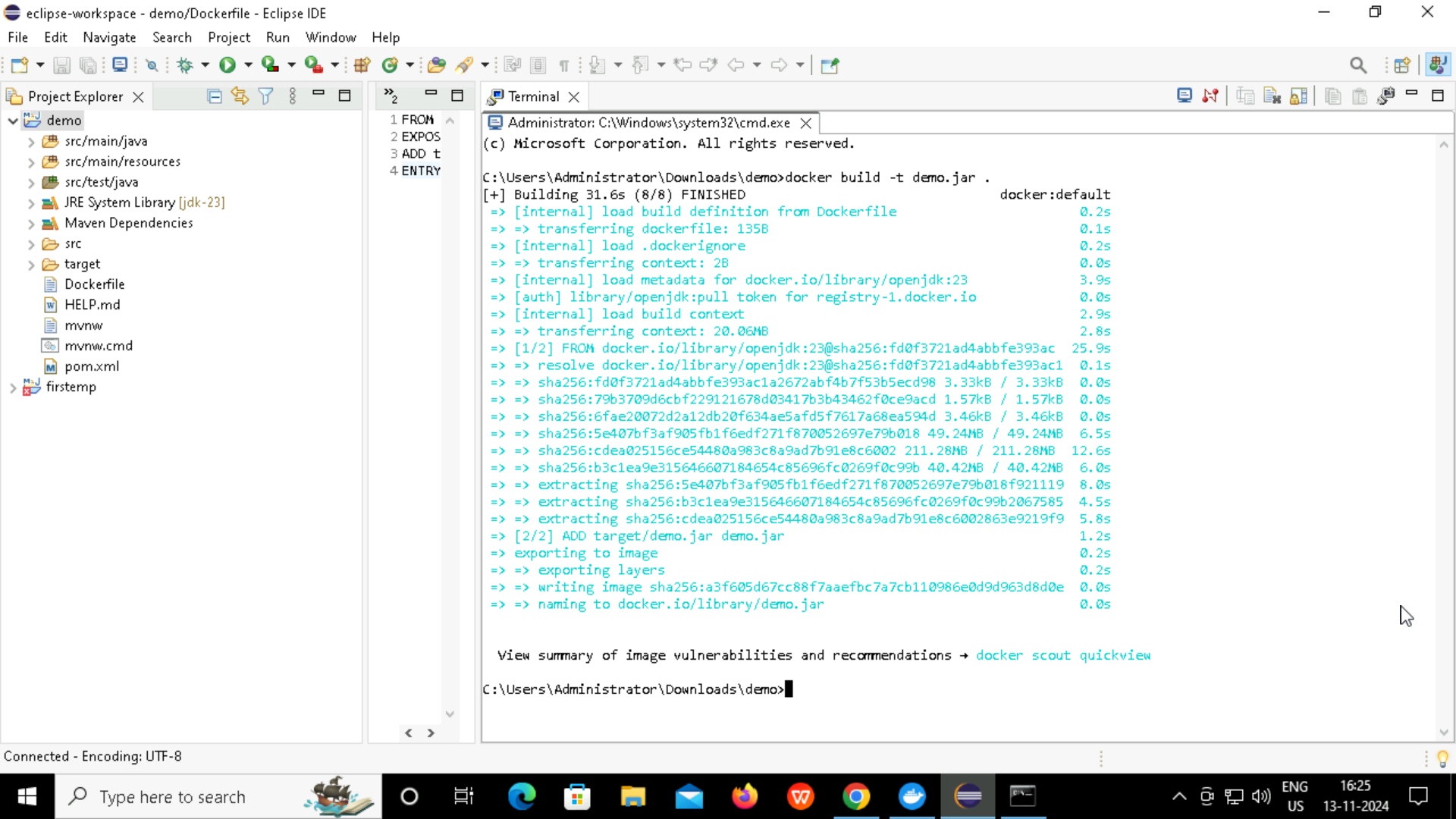


**Step-20: In Eclipse, go to demo -> new file -> Dockerfile. Then add below code.**

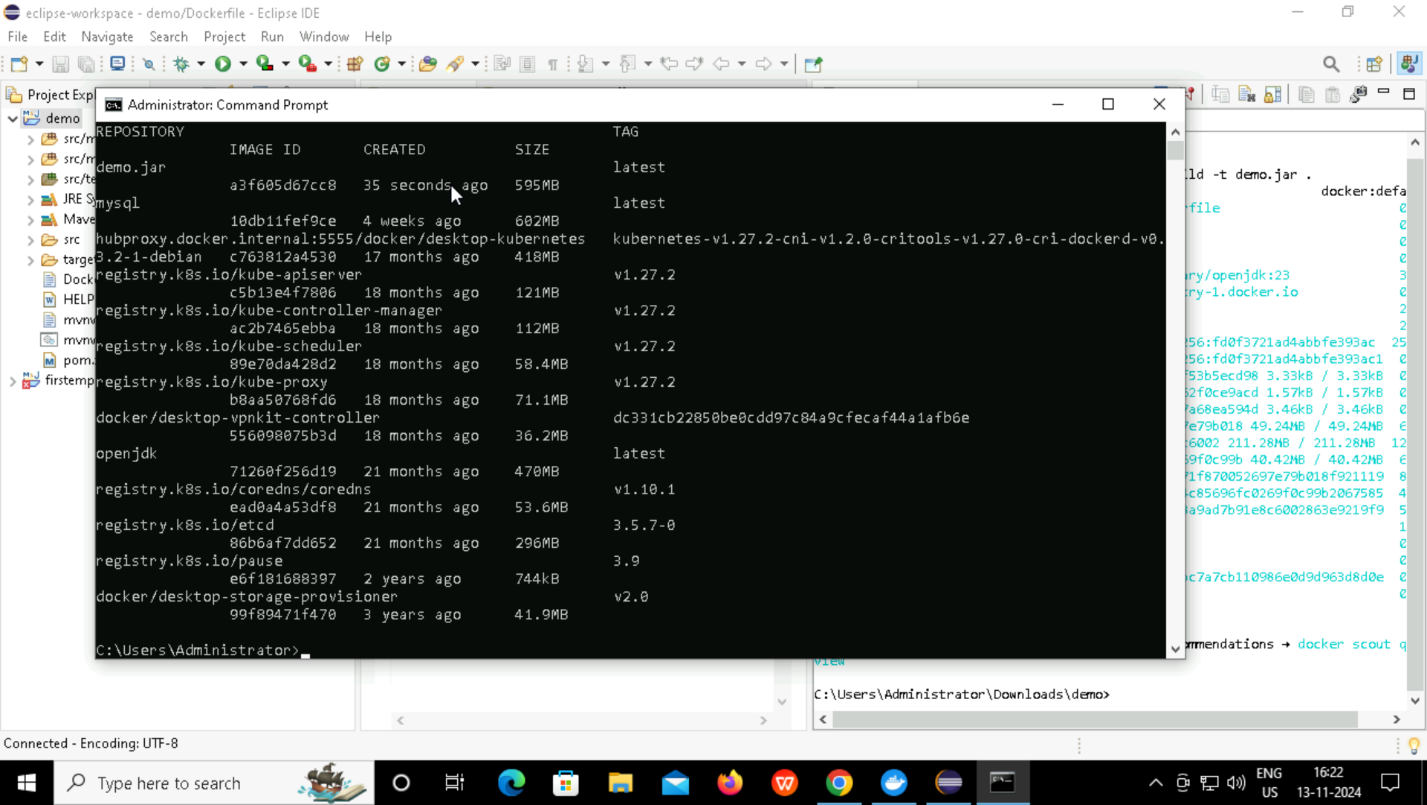


**Step-21: Run in terminal: docker build -t demo.jar .**

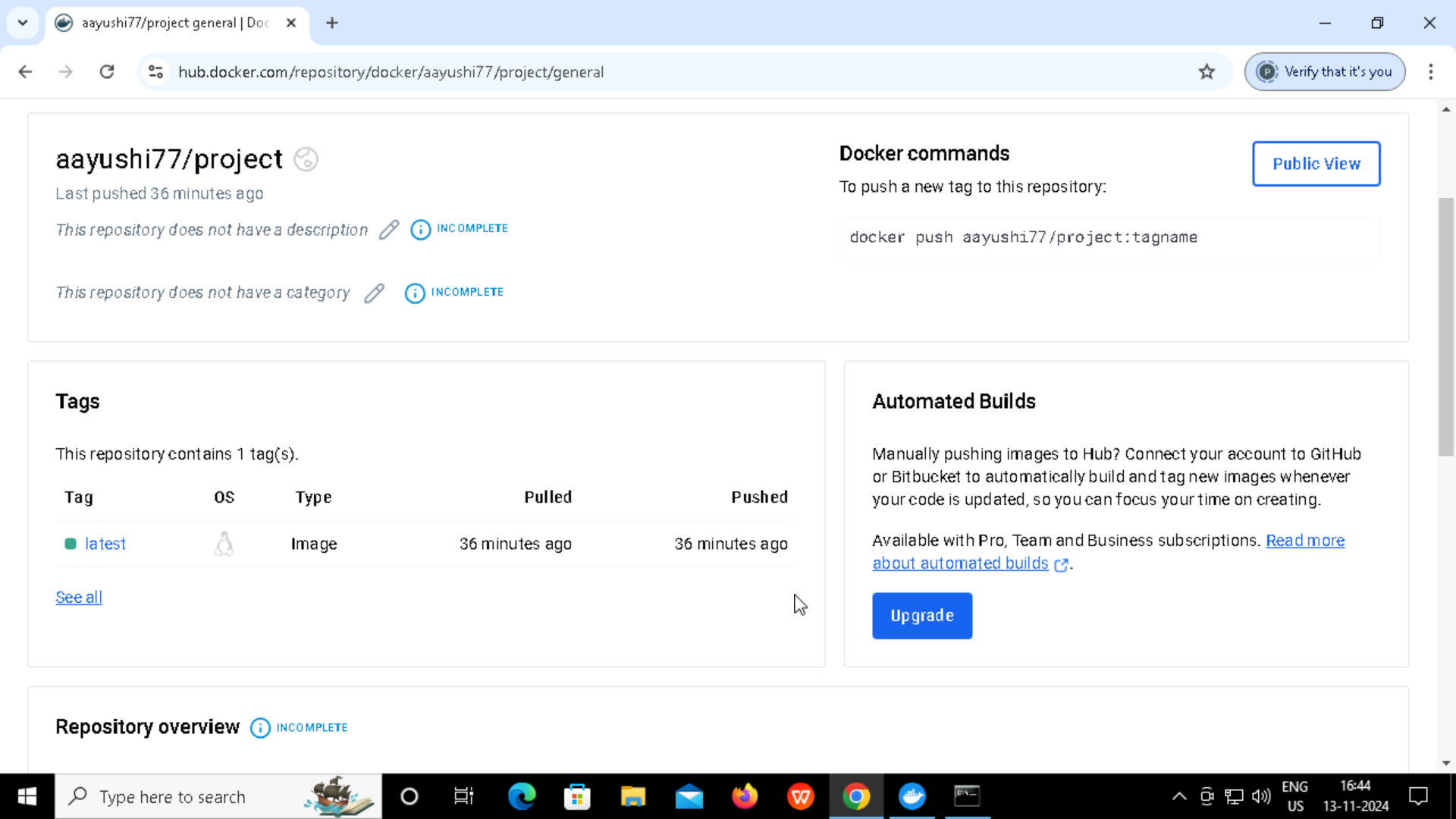
**This command will create docker image once build successfully.**



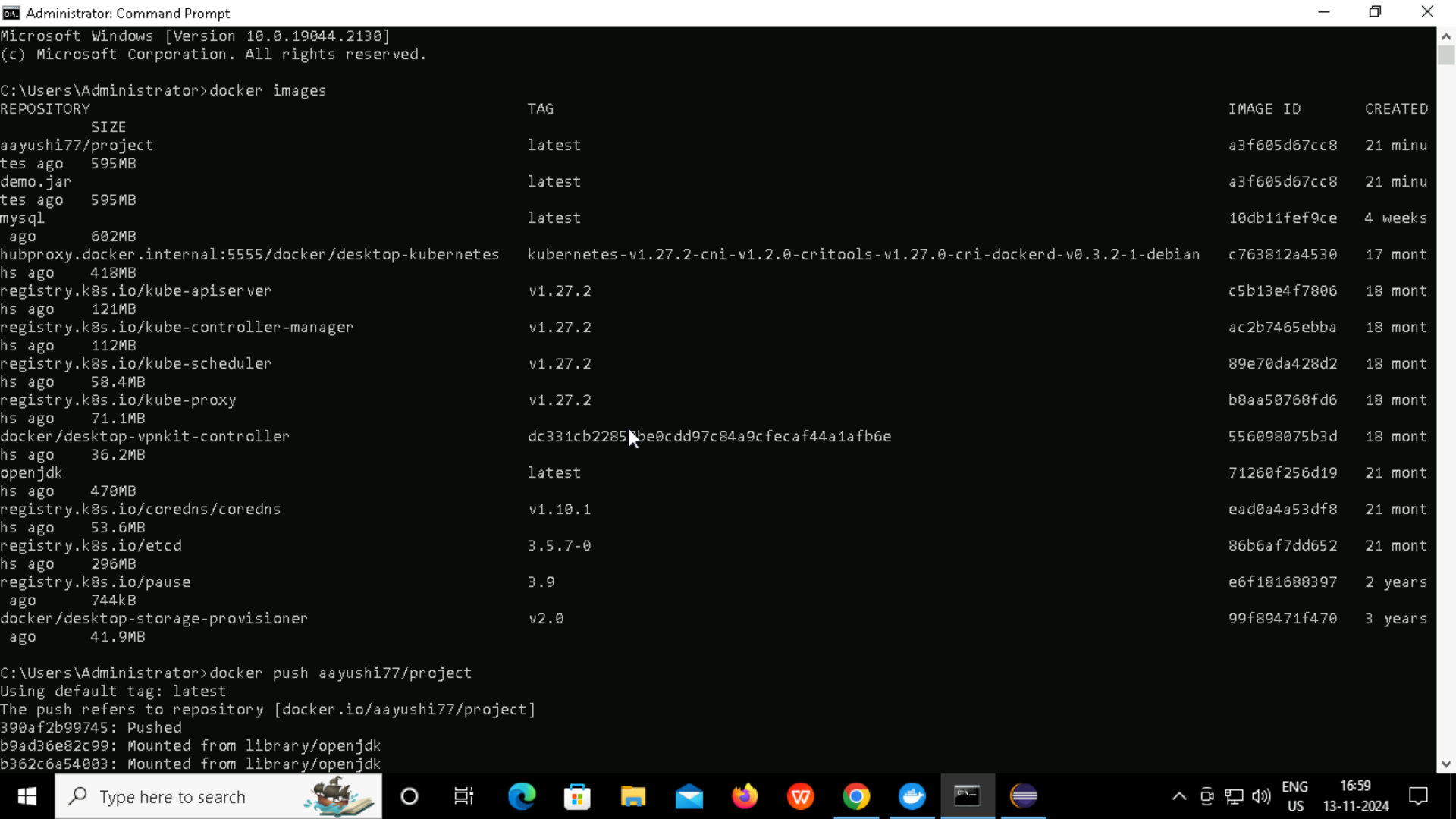
**Step-22: Go to command prompt, then type docker images and check if image is created.**



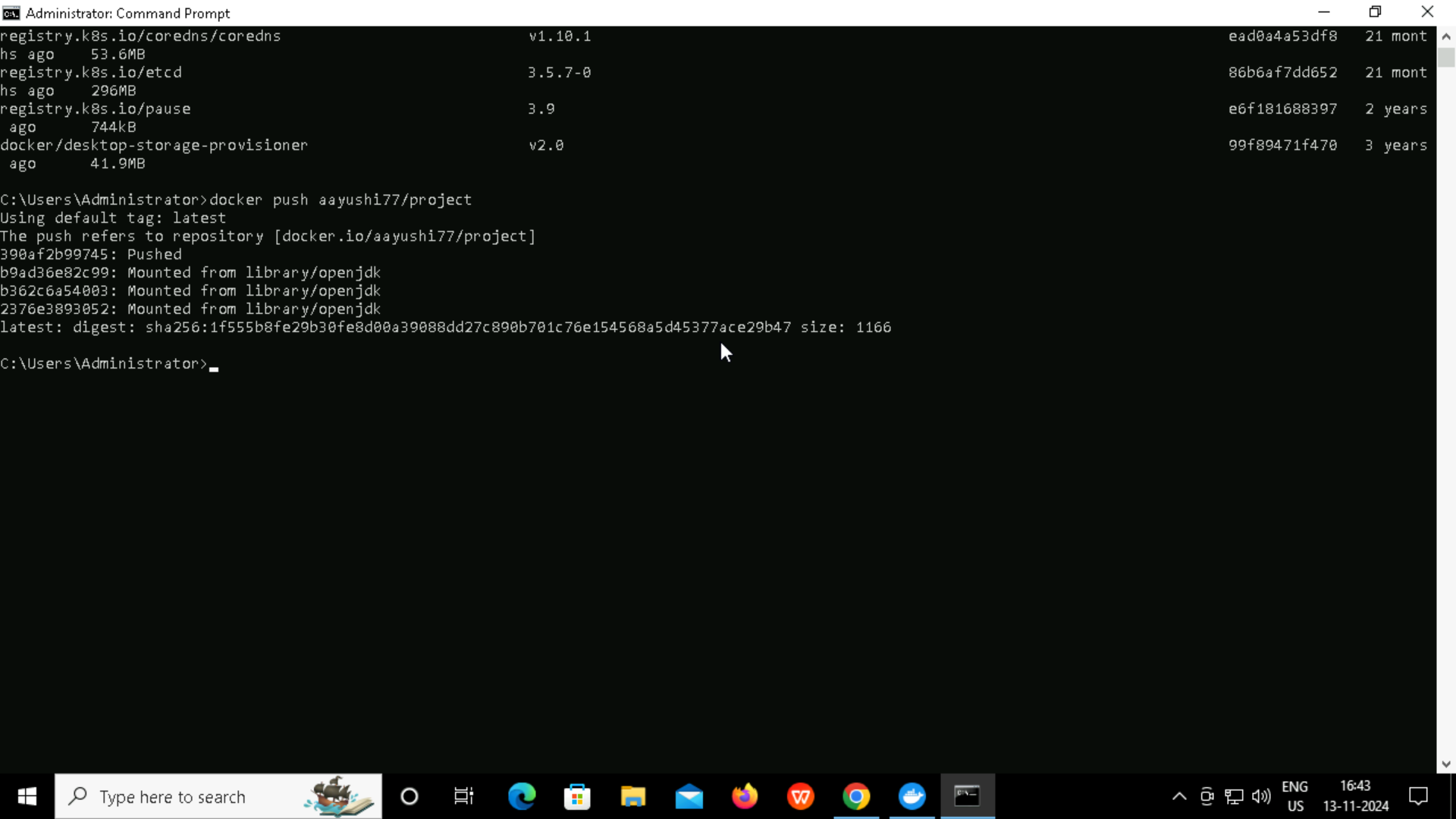
**Step-23: Go to Docker Hub and login. Once logged in, create repository**



**Step-24: Go to command prompt -> enter docker tag demo.jar aayushi77/project and then check docker images to check if the tag is present.**



**Step-25: In command prompt, write docker push aayushi77/project to push the project to docker hub repository.**



**Step-26: Go to Docker Hub and check if files are pushed. Once pushed, the project is completed successfully.**

