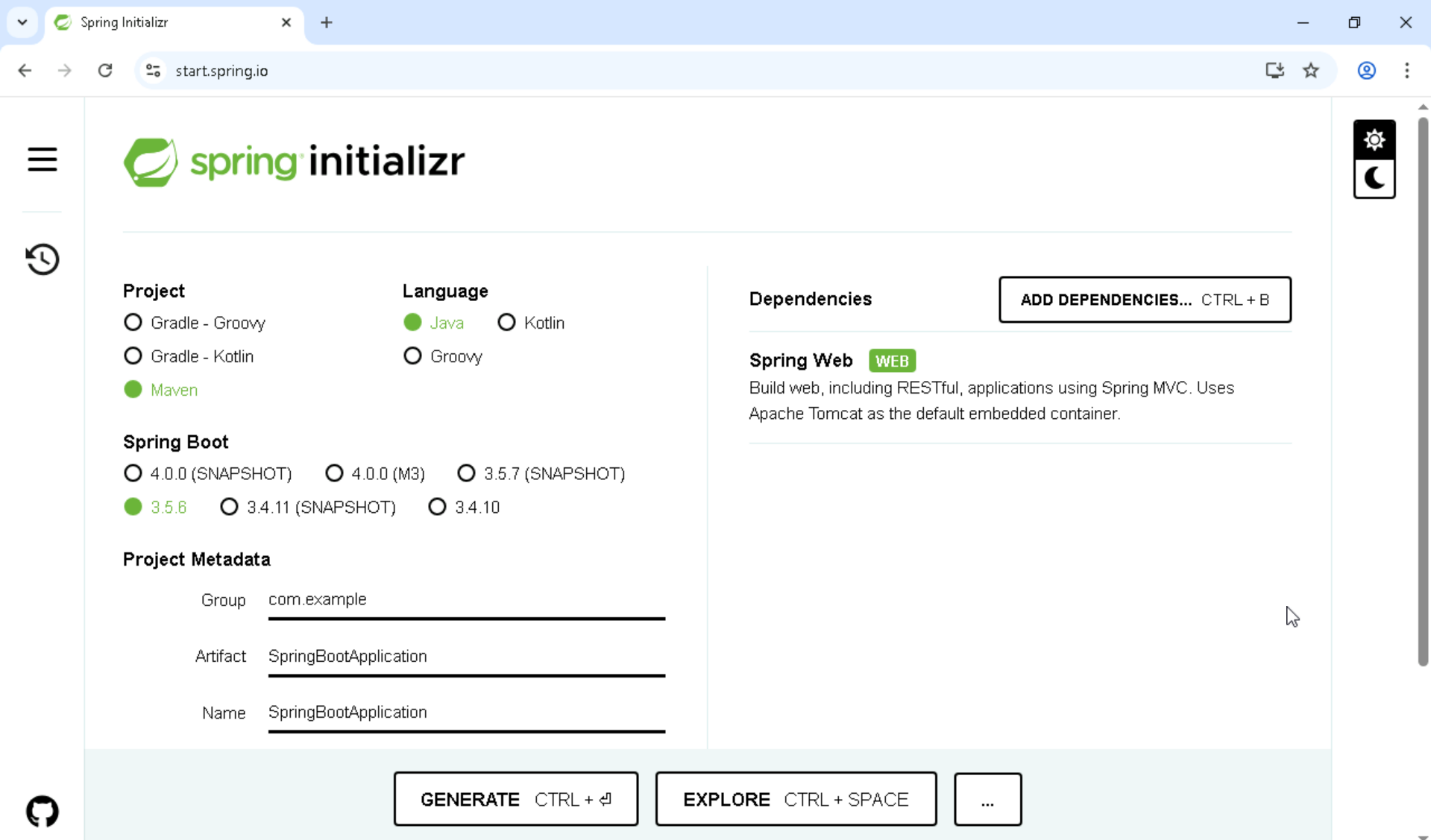
**Spring Boot Application Setup and Configuration**

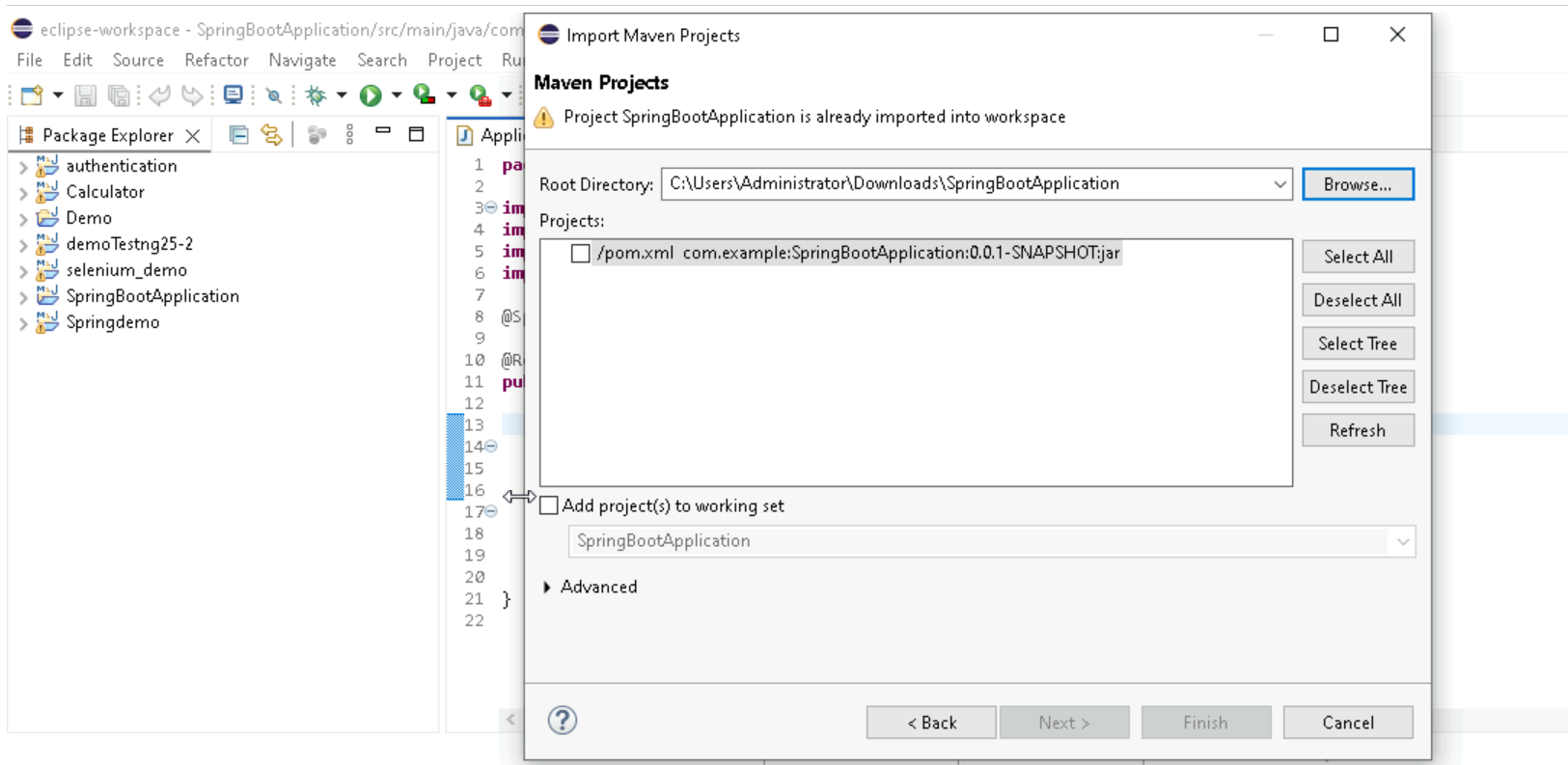
1. **Create Project Using Spring Initializr**

* Go to [https://start.spring.io](https://start.spring.io/)
* Choose:
  + Project: Maven
  + Language: Java
  + Spring Boot version (e.g., 3.x.x)
  + Group and Artifact names
  + Dependencies (e.g., Spring Web)
* Click **Generate** to download the .zip file



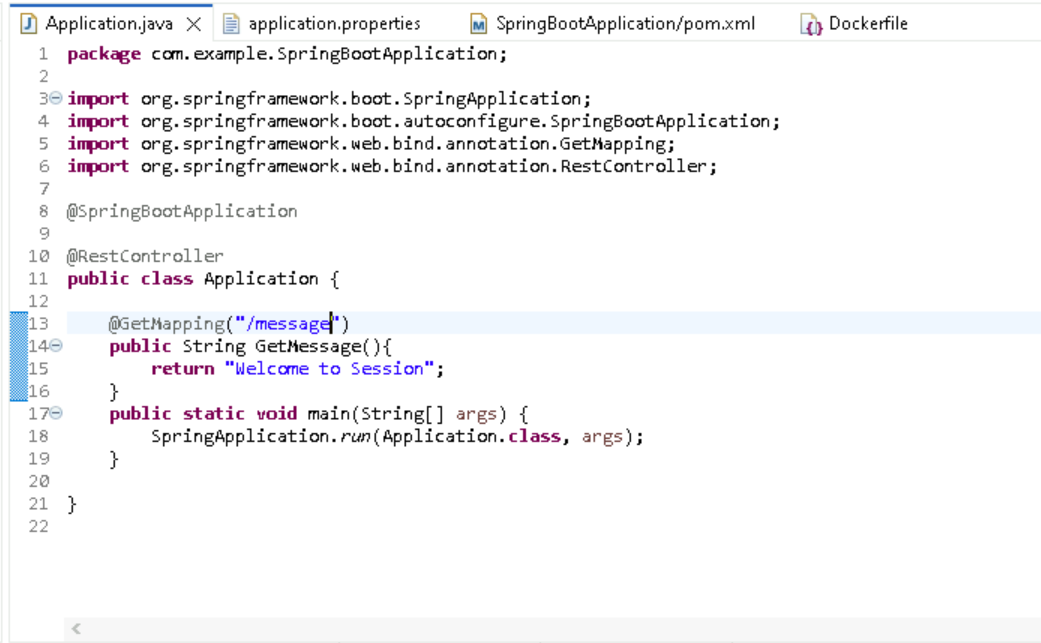
2. **Extract and Open in Eclipse**

* Extract the downloaded .zip file
* Open Eclipse IDE
* Go to **File → Import → Existing Maven Projects**
* Select the extracted folder and finish the import



3. **Modify Java Files to Add Endpoints**

* Navigate to src/main/java/com/example/demo
* Open the main application class or create a new controller class
* Add REST endpoints using @RestController and @RequestMapping or @GetMapping, etc.



4. **Change Server Port in application.properties**

* Go to src/main/resources/application.properties
* Add or modify the following line to change the default port:

server.port=8081

A screenshot of a computer

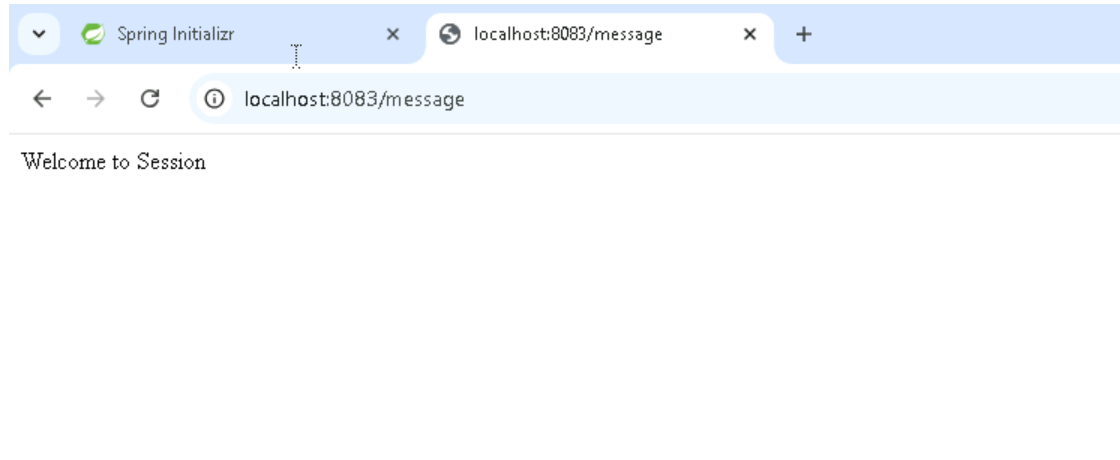
AI-generated content may be incorrect.

5. **Update pom.xml as Needed**

* Open pom.xml file
* Add or modify dependencies, plugins, or properties as required

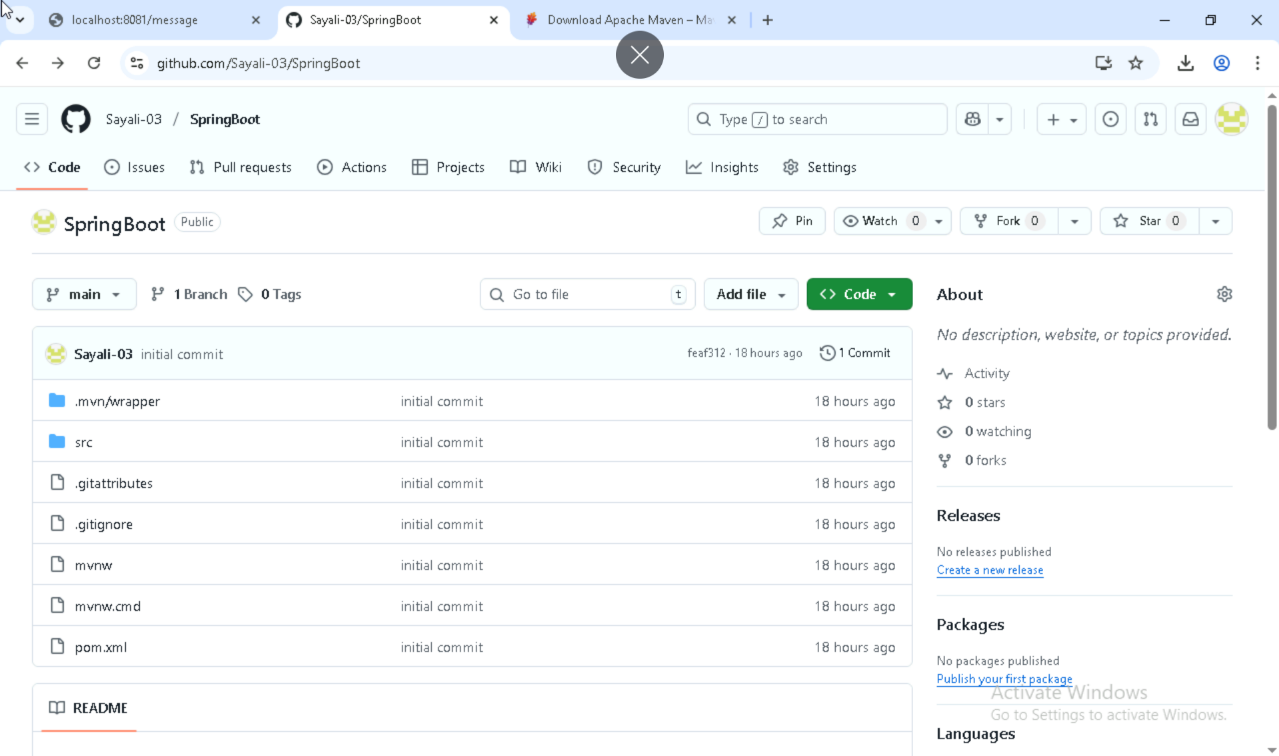
6. **Run and Test the Application**

* Right-click the main class → Run As → Java Application
* Open browser and test endpoint: <http://localhost:8083/message>



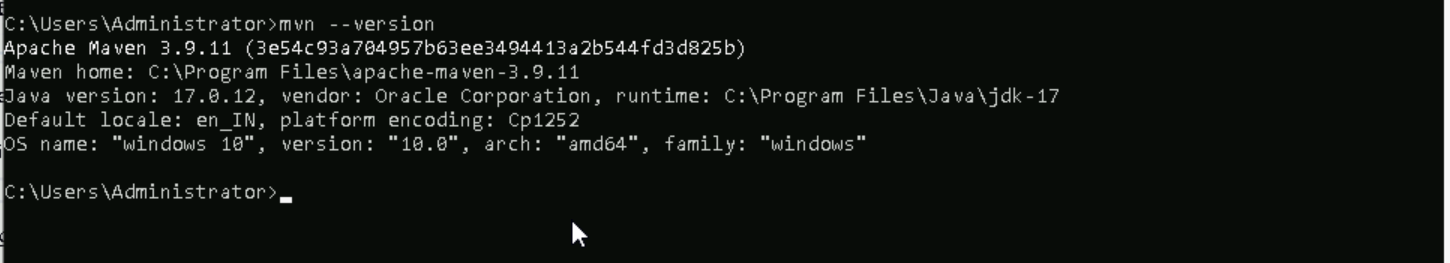
**8. Push Spring Boot Project to GitHub**

* Initialize the git in project.
* Create one repository in github.
* Add the files, commit the changes and push the project to github repository.



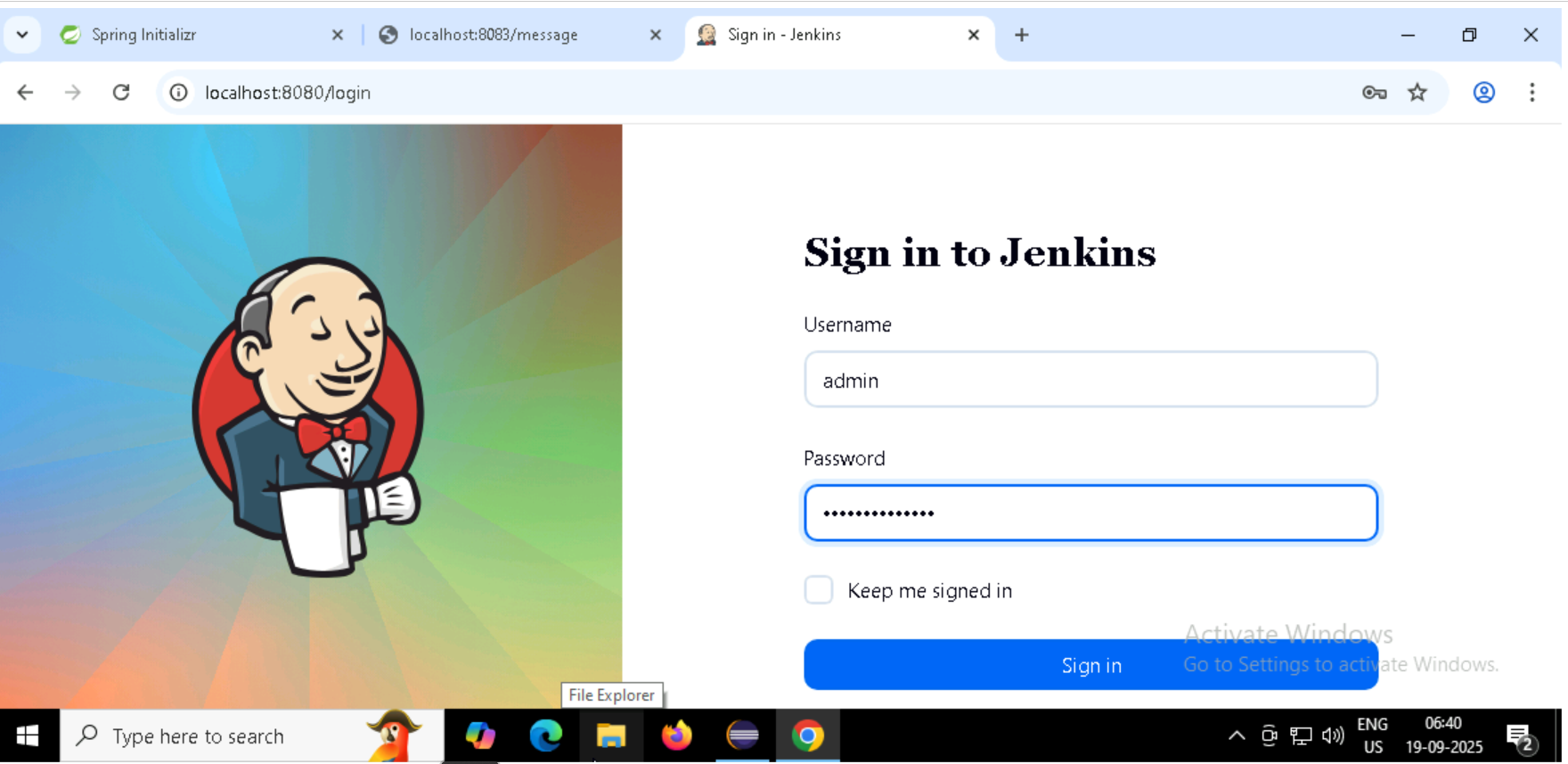
**Jenkins Integration for Spring Boot Project**

First Install the Maven in your system and update the environment variables –



Steps to Configure Jenkins

1. Login to Jenkins
   * Open Jenkins in your browser and log in using your credentials.



1. Create a New Project
   * Click on “New Item”.
   * Enter a project name and select “Freestyle project”.
   * Click OK to proceed

A screenshot of a computer

AI-generated content may be incorrect.

1. Configure Project Details
   * Add a project description.
   * Under Source Code Management, choose Git.
   * Enter your GitHub repository URL (Spring Boot project).
   * Add GitHub credentials (username and personal access token).

A screenshot of a computer

AI-generated content may be incorrect.

1. Configure Build Tools
   * Go to Build Environment → Tool Configuration.
   * Add and configure Maven (select installed Maven version).

A screenshot of a computer

AI-generated content may be incorrect.

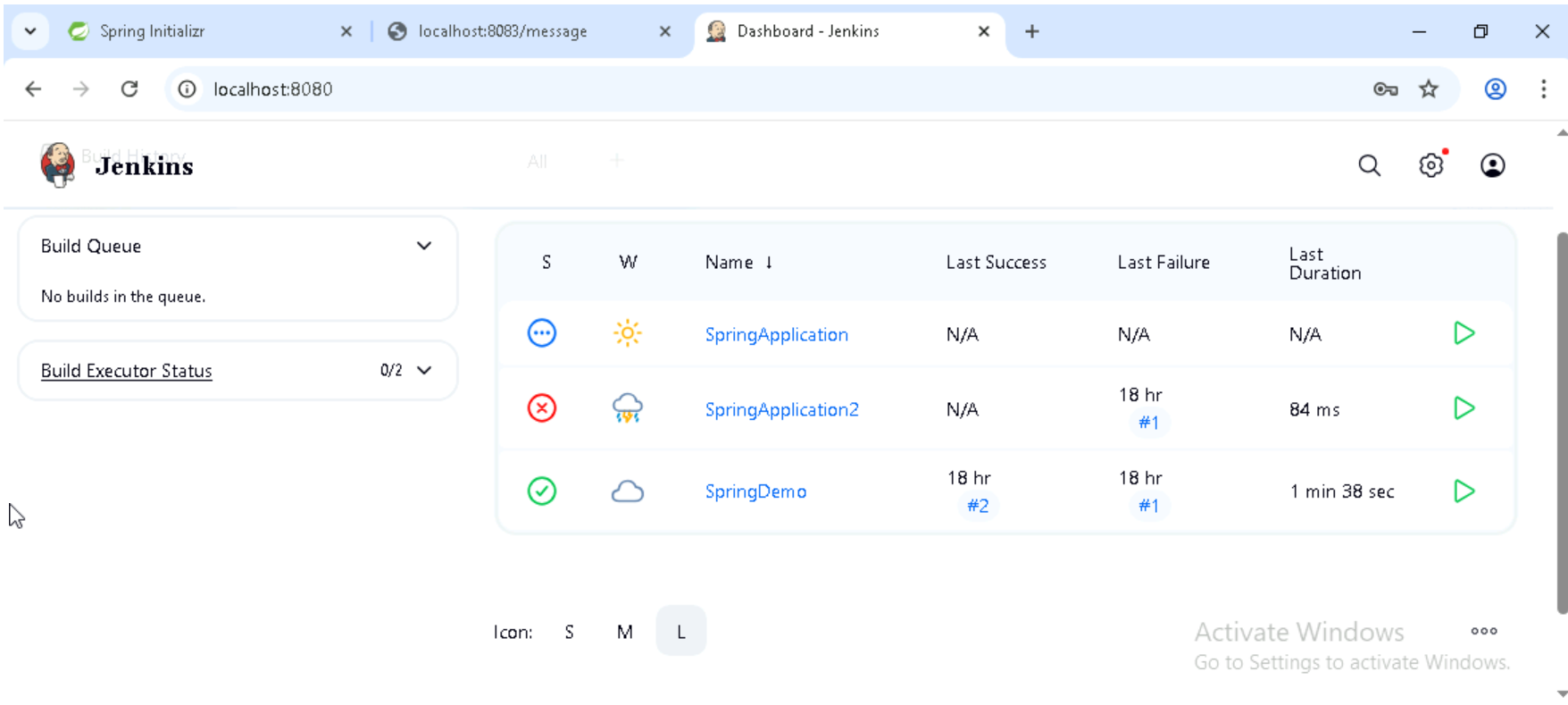
1. Build the Project
   * In the Build section, add a Maven command like clean install.
   * Save the configuration and click Build Now.
   * Check the Console Output to confirm build success.

A screenshot of a computer

AI-generated content may be incorrect.

1. Verify Build Success

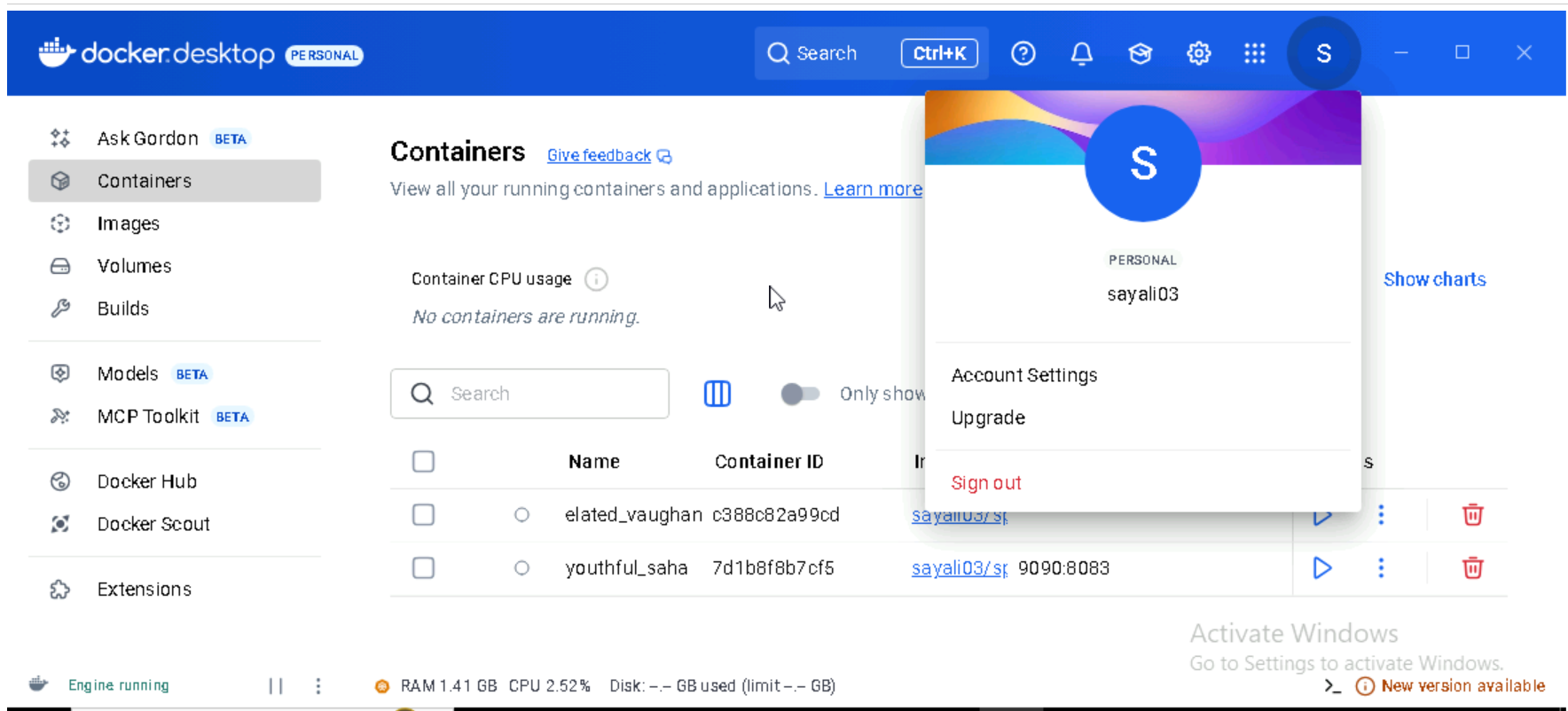
* A successful build will show a green checkmark.
* Artifacts (like JAR files) will be generated and stored.



**Docker Integration with Spring Boot -**

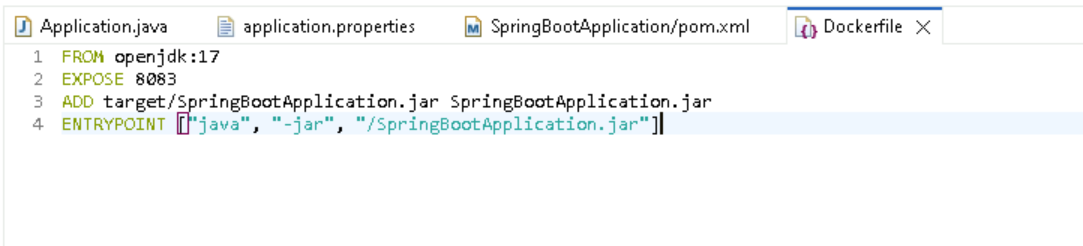
**Step 1: Create a DockerHub Account and Login via Docker Desktop**

* Go to <https://hub.docker.com> and create an account.
* DockerHub is a cloud-based registry where you can store and share container images.
* Open Docker Desktop and log in using your DockerHub credentials.



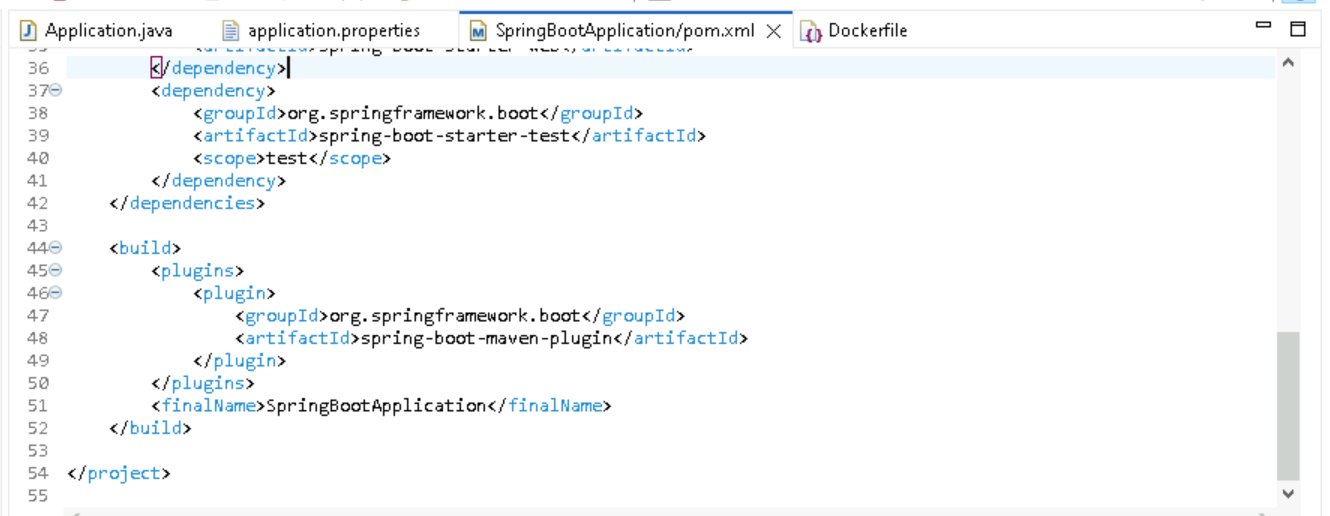
**Step 2: Create a Dockerfile in Your Spring Boot Project**

* Place the Dockerfile in the root directory of your Spring Boot project.
* This file defines how your Spring Boot app will be packaged into a Docker image.



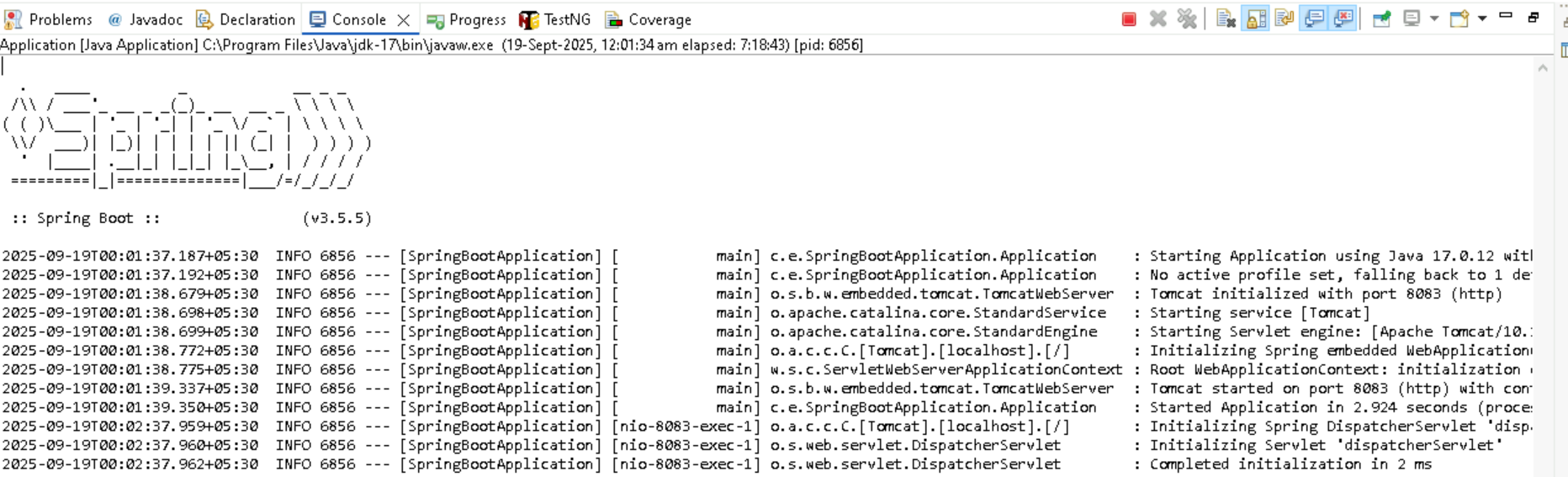
**Step 3: Update pom.xml to Package the Application**

* Update the finalName in pom.xml file. Ensures your project can be compiled into a runnable JAR for Docker.



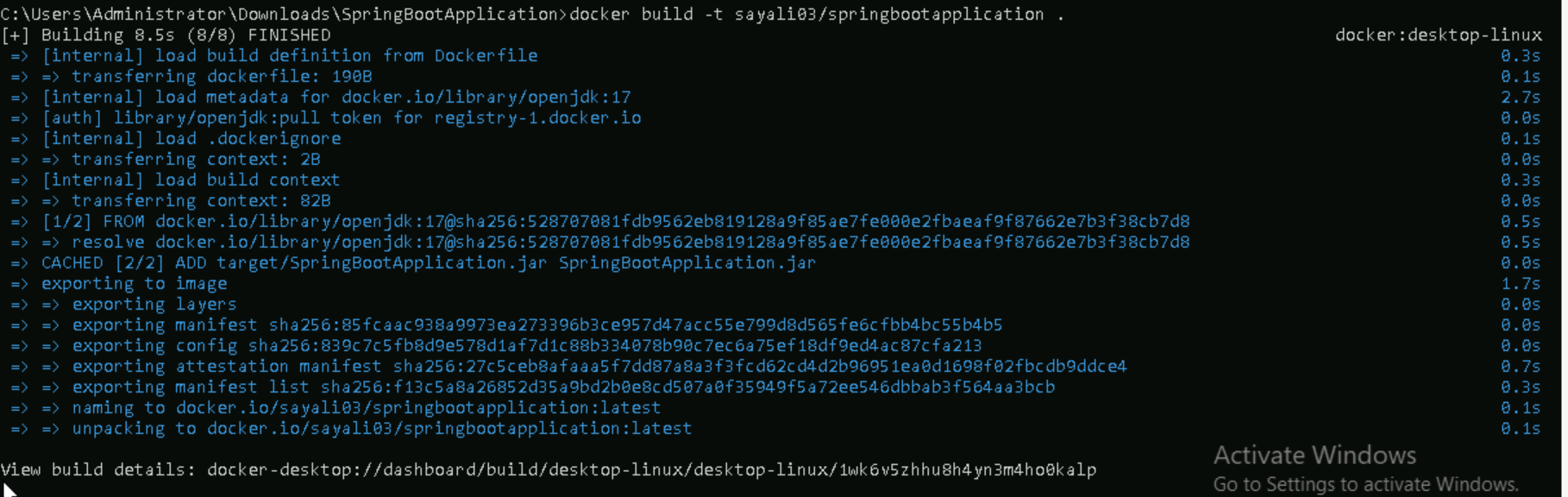
**Step 4: Run Maven Install to Build the Project**

* Compiles the code and packages it into a JAR file inside the target/ folder.
* Run as maven 6 install.



**Step 5: Build the Docker Image**

* **Command**: docker build -t sayali03/springbootapplication .
* Creates a Docker image from your project using the Dockerfile. The -t flag tags the image with your DockerHub username and image name.



**Step 6: List Docker Images**

* **Command**: docker images
* Displays all locally available Docker images. Confirms your image was built successfully.



**Step 7: Run the Docker Container**

* **Command**: docker run -p 9090:8083 sayali03/springbootapplication
* Starts a container from your image. Maps port 8083 inside the container to 9090 on your machine.

A black screen with white text

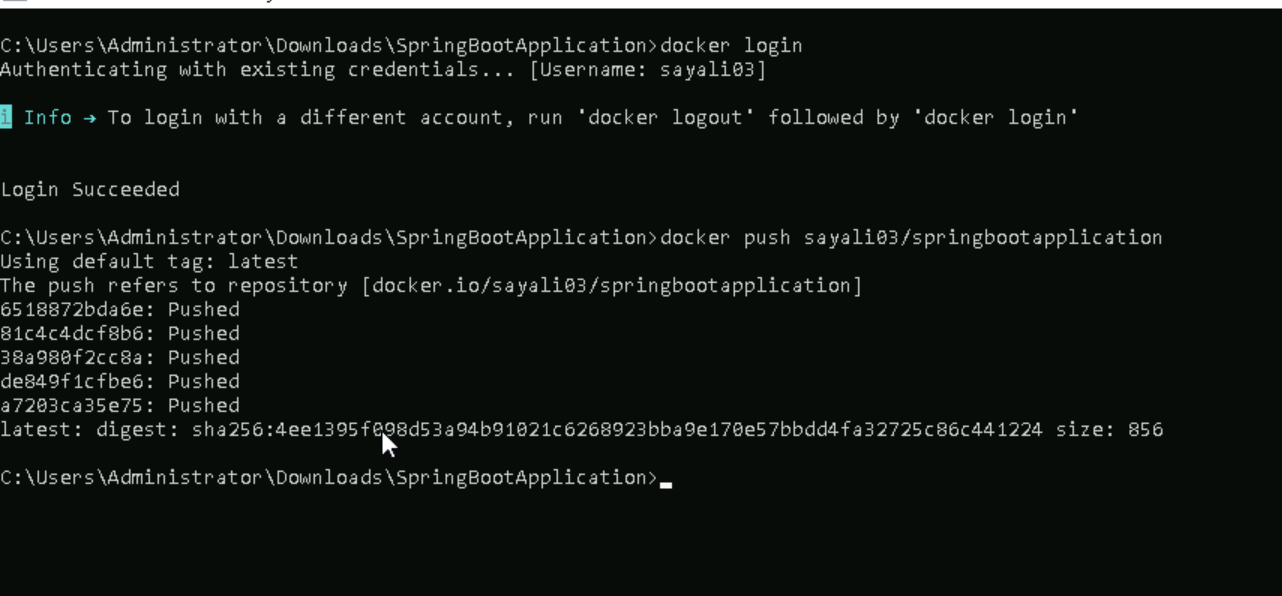
AI-generated content may be incorrect.

**Step 8: Login to DockerHub**

* **Command**: docker login
* Authenticates your Docker CLI with DockerHub so you can push images.

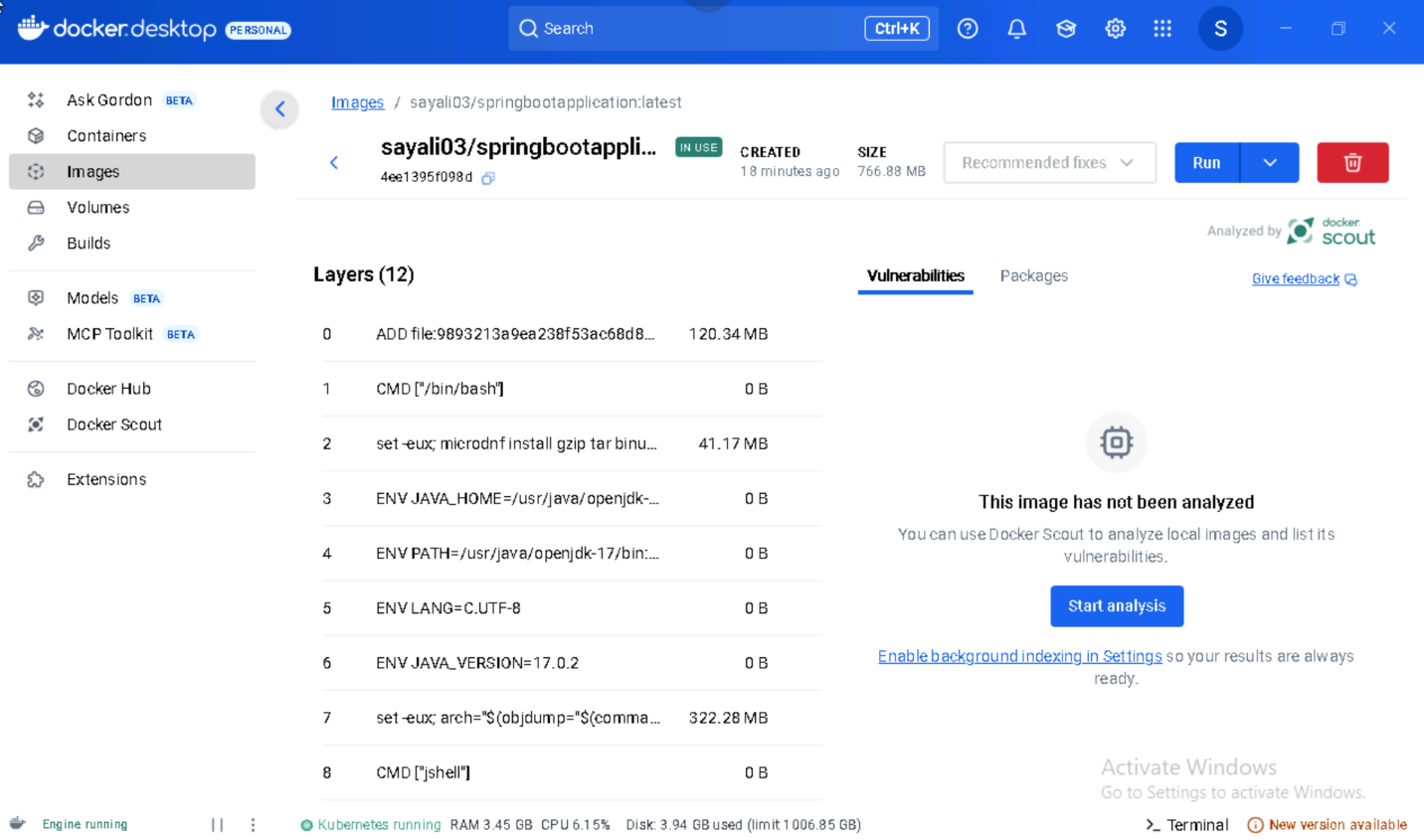
**Step 9: Push the Image to DockerHub**

* **Command**: docker push sayali03/springbootapplication
* Uploads your image to DockerHub so others can pull and use it.



**Step 10: Verify Image in Docker Desktop**

* Open Docker Desktop → Go to "Images" tab.
* Confirms your image is available locally and/or synced with DockerHub.



A screenshot of a computer

AI-generated content may be incorrect.