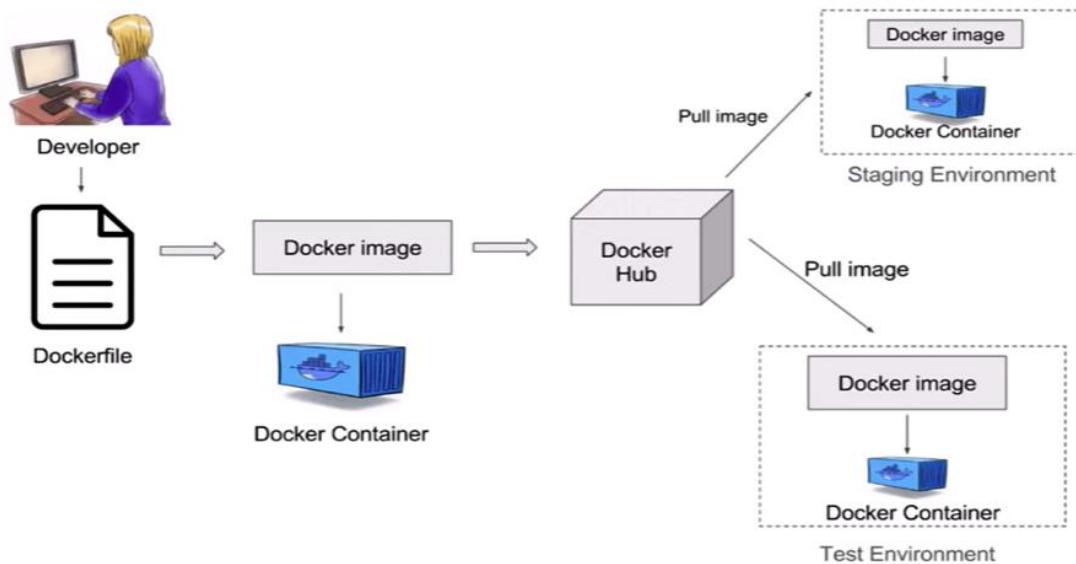


Deploying a Spring Boot Application using Docker on VSCode IDE

In this tutorial, you will learn how to use Docker to deploy the Spring Boot application in a container. This Spring Boot Docker tutorial is designed for beginners to learn step-by-step how to use Docker with the Spring Boot application. In this tutorial we will use the VSCode IDE.

Docker Workflow



The developer will create a **Dockerfile**, where all the instructions and commands to build the docker image. Dockerfile is a text file, it contains all instructions or commands to build the docker image. Once you create a Dockerfile, you can use the Docker build command to build the docker image. Docker image is an executable package and we can run the docker image in a container. Docker container is just a running instance of Docker image. The process of writing the Dockerfile and building the docker image is called **Dockerization**.

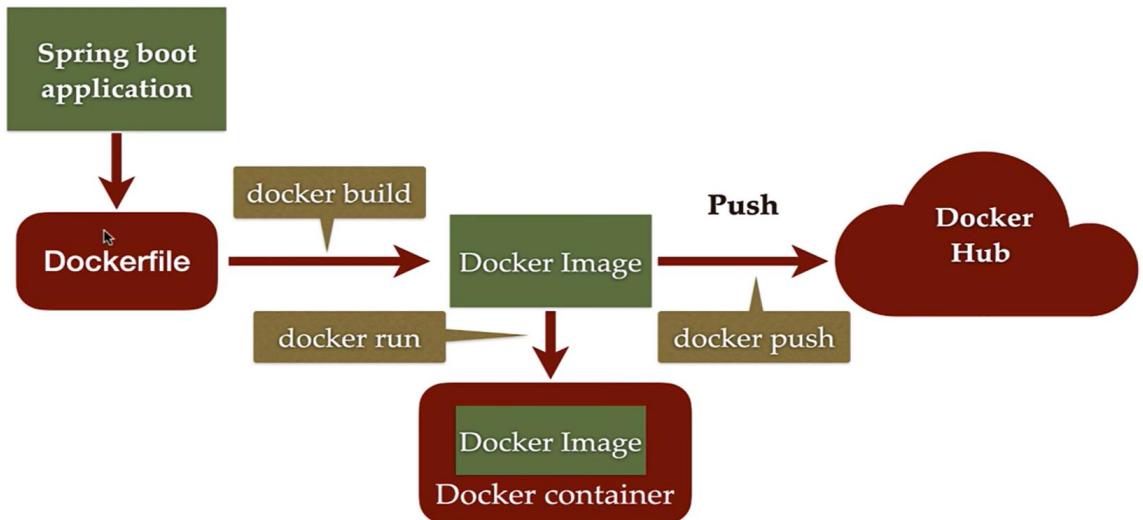
Once we have the Docker image in our local machine, we can push the image to an online Cloud repository called Docker hub. Once we push the image to the Docker hub, we can pull that image on any environment and we can start deploying or running this Docker image in a Docker container. For example, let us say we have different environments like staging, test, production or UAT environment. This is how a typical Docker Workflow looks like.

Architecture

Consider we want to Dockerize the Spring Boot application. We have to first create a Dockerfile where we will define all the instructions or commands to build the Docker Image. Once the Dockerfile is ready, we can use the Docker Build Command to build the Docker image from the Dockerfile. The Docker Image is just an executable package.

Next, we can use the Docker Run Command to run this Docker Image in a container. Docker container is just a running instance or runtime instance of the Docker Image. This complete process is called **Dockerization**.

Once we have a Docker Image on our local machine, then we can go ahead and push this Docker Image on a Docker Hub using Docker Push Command. Once we push the Docker Image to the Docker hub, and if we make this Docker Image as a public on Docker Hub. Then anybody can go ahead and pull this Docker Image from the Docker hub and they can run that Docker image in a Docker Image.



IMPLEMENTATION

Now, let us see how to Dockerize Spring Boot application step by step. We have to implement these steps now.

Dockerizing Spring Boot Application:

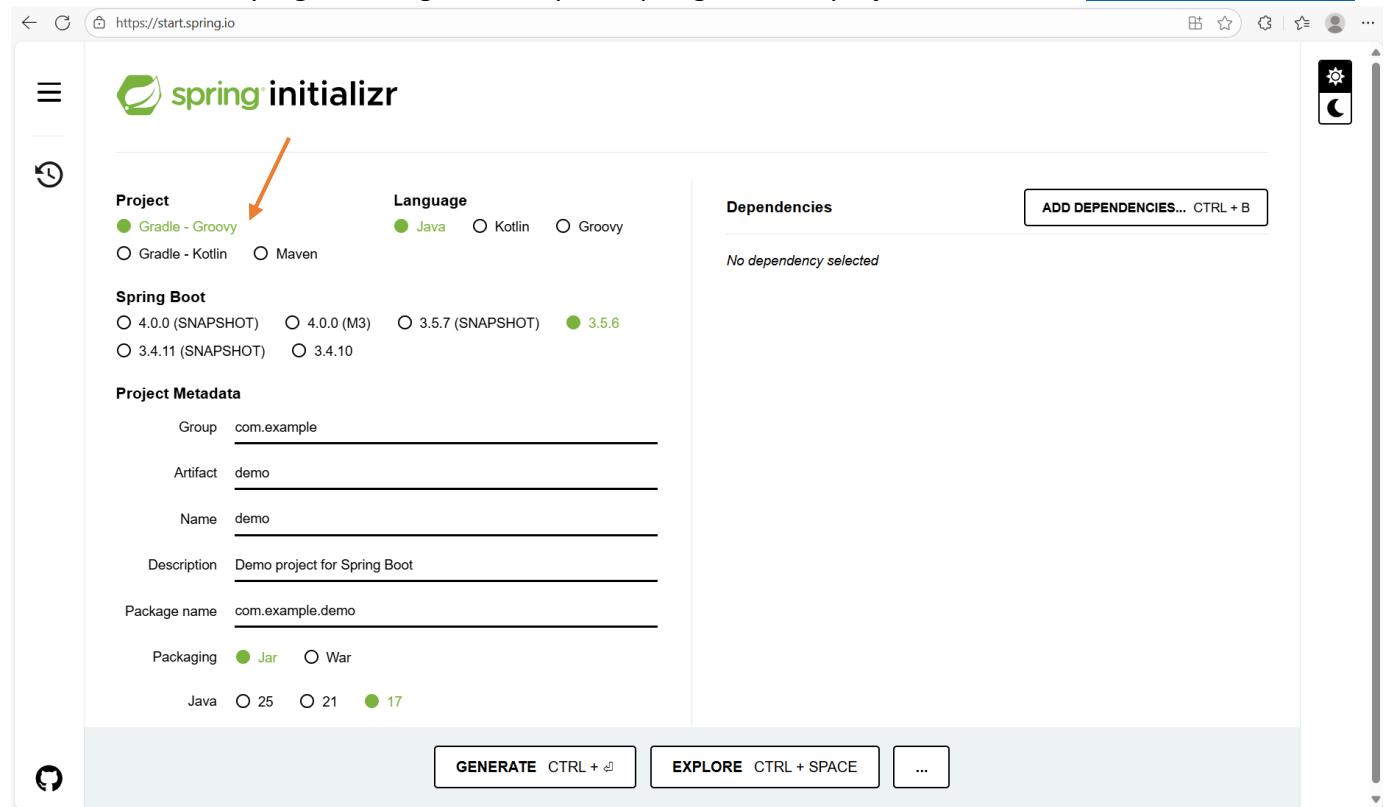
1. Create a Spring Boot Project
2. Build a Simple REST API
3. Build and Package Maven application
4. Create a Dockerfile to Build a Docker Image
5. Build Docker Image from Dockerfile
6. Run Docker Image in a Docker Container
7. Run Docker Image in a Docker Container in Detached Mode
8. Push Docker Image from Local Machine to DockerHub
9. Pull the Docker Image from a Docker Hub

Pull Docker Image from DockerHub

STEP 1: Create a Spring Boot Project

First, we will create a Spring Boot application and we will build some REST API. Let us create a Spring Boot application and we will import that Spring Boot application.

Let us start by generating a simple Spring boot project. Go to <http://start.spring.io>



The screenshot shows the Spring Initializr web interface at <https://start.spring.io>. The interface is used to generate a Spring Boot project. It has several sections:

- Project**: Options for "Gradle - Groovy" (selected), "Gradle - Kotlin", and "Maven".
- Language**: Options for "Java" (selected), "Kotlin", and "Groovy".
- Dependencies**: A section where users can add dependencies. It currently says "No dependency selected" and has a button "ADD DEPENDENCIES... CTRL + B".
- Spring Boot**: Version selection for "4.0.0 (SNAPSHOT)", "4.0.0 (M3)", "3.5.7 (SNAPSHOT)" (selected), "3.5.6", "3.4.11 (SNAPSHOT)", and "3.4.10".
- Project Metadata**: Fields for "Group" (com.example), "Artifact" (demo), "Name" (demo), "Description" (Demo project for Spring Boot), "Package name" (com.example.demo), and "Packaging" (Jar selected, War, War).
- Java**: Version selection for "25", "21" (selected), and "17".
- Buttons**: "GENERATE" (CTRL + D) and "EXPLORE" (CTRL + SPACE) buttons, and a "...".

An orange arrow points to the "Maven" option under the "Project" section.

On "Project", we will choose "Maven"

The screenshot shows the Spring Initializr web application at <https://start.spring.io>. The 'Project' section is set to Maven. The 'Language' section has Java selected. In the 'Spring Boot' section, the version 3.5.6 is highlighted with a green dot and an orange arrow points to it from the text above. The 'Project Metadata' section includes fields for Group (com.example), Artifact (demo), Name (demo), Description (Demo project for Spring Boot), Package name (com.example.demo), and Packaging (Jar). Below these, Java version counts are shown: 25, 21, and 17. At the bottom are 'GENERATE' and 'EXPLORE' buttons.

On “Spring Boot”, we will use version “3.5.6”

This screenshot shows the same Spring Initializr interface as the previous one, but with a change in the 'Project Metadata' section. The 'Group' field now contains 'com.example' instead of its previous value. An orange arrow points from the text above to this modified field. All other settings remain the same as in the first screenshot.

Let us change the Group to “net.javaguides”

The screenshot shows the Spring Initializr interface. In the 'Project' section, 'Gradle - Groovy' and 'Gradle - Kotlin' are listed, with 'Gradle - Groovy' selected. Under 'Language', 'Java' is selected. In the 'Spring Boot' section, '3.5.6' is selected. The 'Project Metadata' section includes fields for Group (net.javaguides), Artifact (demo), Name (demo), Description (Demo project for Spring Boot), Package name (net.javaguides.springboot), and Packaging (Jar). Below these, Java version 17 is selected. On the right, there's a 'Dependencies' section with a button to 'ADD DEPENDENCIES...'. At the bottom, there are 'GENERATE' and 'EXPLORE' buttons.

We have to give the project a name, I will call it “**springboot-docker-demo**”

This screenshot is identical to the previous one, but the 'Package name' field has been changed to 'net.javaguides.springboot-docker-demo'. An orange arrow points from the text 'For “Package Name”, we will use “net.javaguides.springboot”' to this field.

For “Package Name”, we will use “**net.javaguides.springboot**”

The screenshot shows the Spring Initializr interface. In the 'Project' section, 'Maven' is selected. Under 'Spring Boot', '3.5.6' is selected. In the 'Project Metadata' section, the group is set to 'net.javaguides', artifact to 'springboot-docker-demo', name to 'springboot-docker-demo', description to 'Demo project for Spring Boot', and package name to 'net.javaguides.springboot'. The 'Packaging' section shows 'Jar' selected. An orange arrow points from the text 'On “Packaging”, we will use “Jar”' to the 'Jar' button. Below the packaging options, Java versions 25, 21, and 17 are listed, with 17 highlighted. At the bottom, there are 'GENERATE' and 'EXPLORE' buttons.

On “Packaging”, we will use “Jar”

This screenshot is identical to the one above, but the Java version has changed. Now, Java version 21 is selected, indicated by an orange arrow pointing to the '21' button. The other project details remain the same.

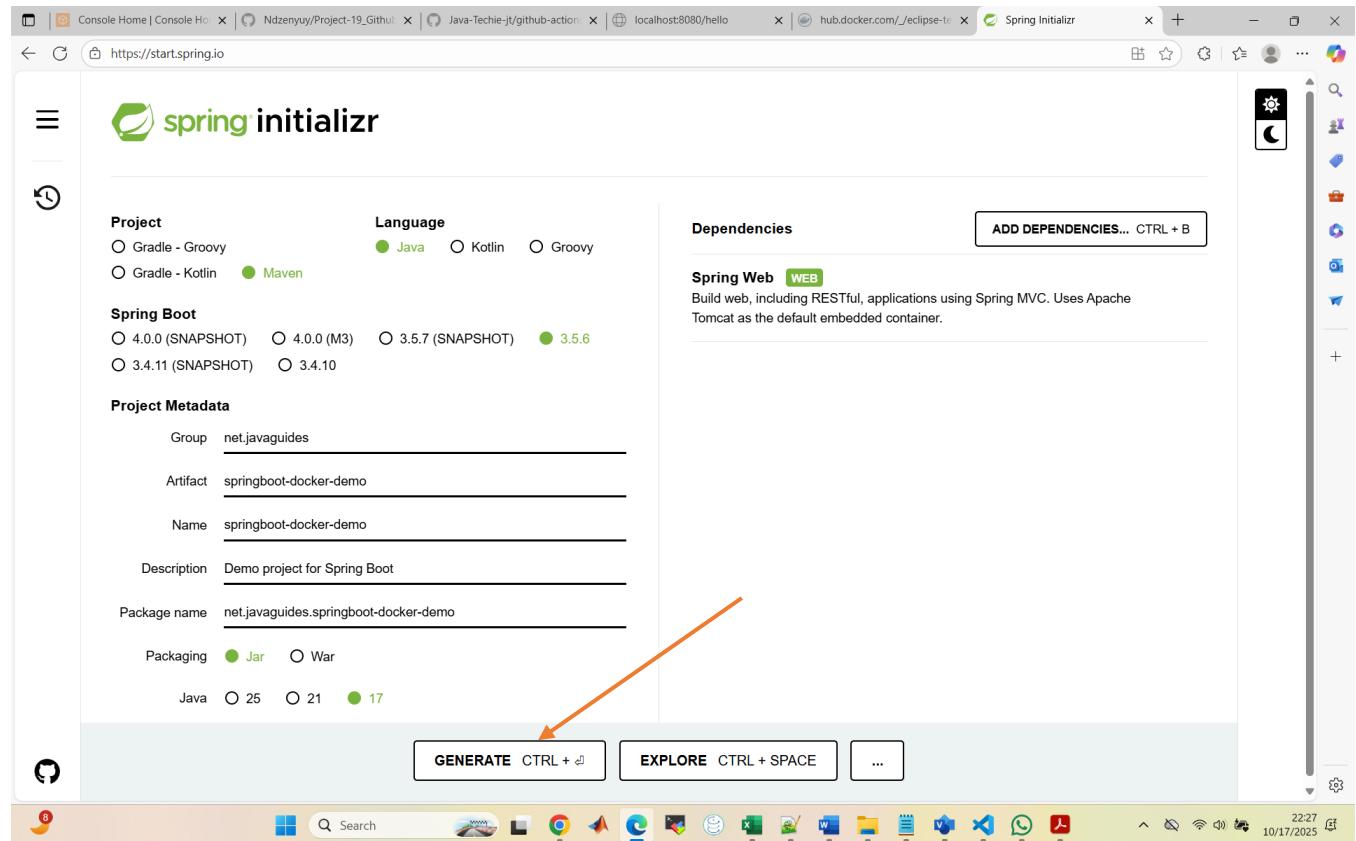
We will use Java version 17 or above. So, we will just use Java Version 17

The screenshot shows the Spring Initializr web application at <https://start.spring.io>. The interface includes sections for Project (Gradle - Groovy, Gradle - Kotlin, Maven), Language (Java, Kotlin, Groovy), and Spring Boot (4.0.0 (SNAPSHOT), 4.0.0 (M3), 3.5.7 (SNAPSHOT), 3.5.6, 3.4.11 (SNAPSHOT), 3.4.10). On the right, there's a 'Dependencies' section with a button labeled 'ADD DEPENDENCIES... CTRL + B'. An orange arrow points to this button. Below it, a message says 'No dependency selected'. At the bottom, there are 'GENERATE' and 'EXPLORE' buttons.

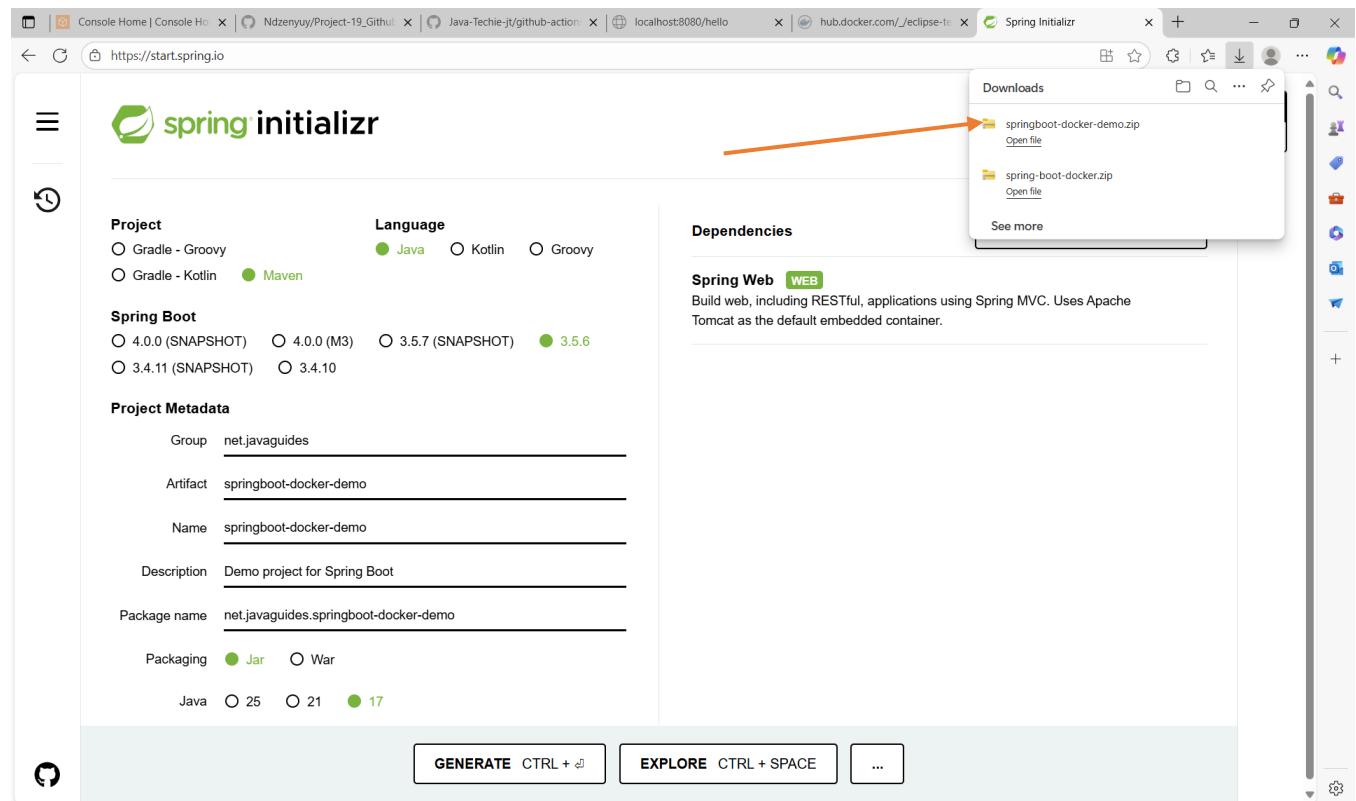
Click on “Add Dependencies”

The screenshot shows the Spring Initializr web application at <https://start.spring.io>. It features a 'DEVELOPER TOOLS' section with options like 'Web, Security, JPA, Actuator, Devtools...', 'Spring Configuration Processor', 'Docker Compose Support', and 'Spring Modulith'. The 'WEB' tab is selected. Under 'WEB', the 'Spring Web' section is highlighted with an orange arrow. It describes building web applications using Spring MVC with Apache Tomcat as the default embedded container. The 'Spring Reactive Web' section is also visible below it. Other tabs include 'HTTP Client' and 'Reactive HTTP Client'.

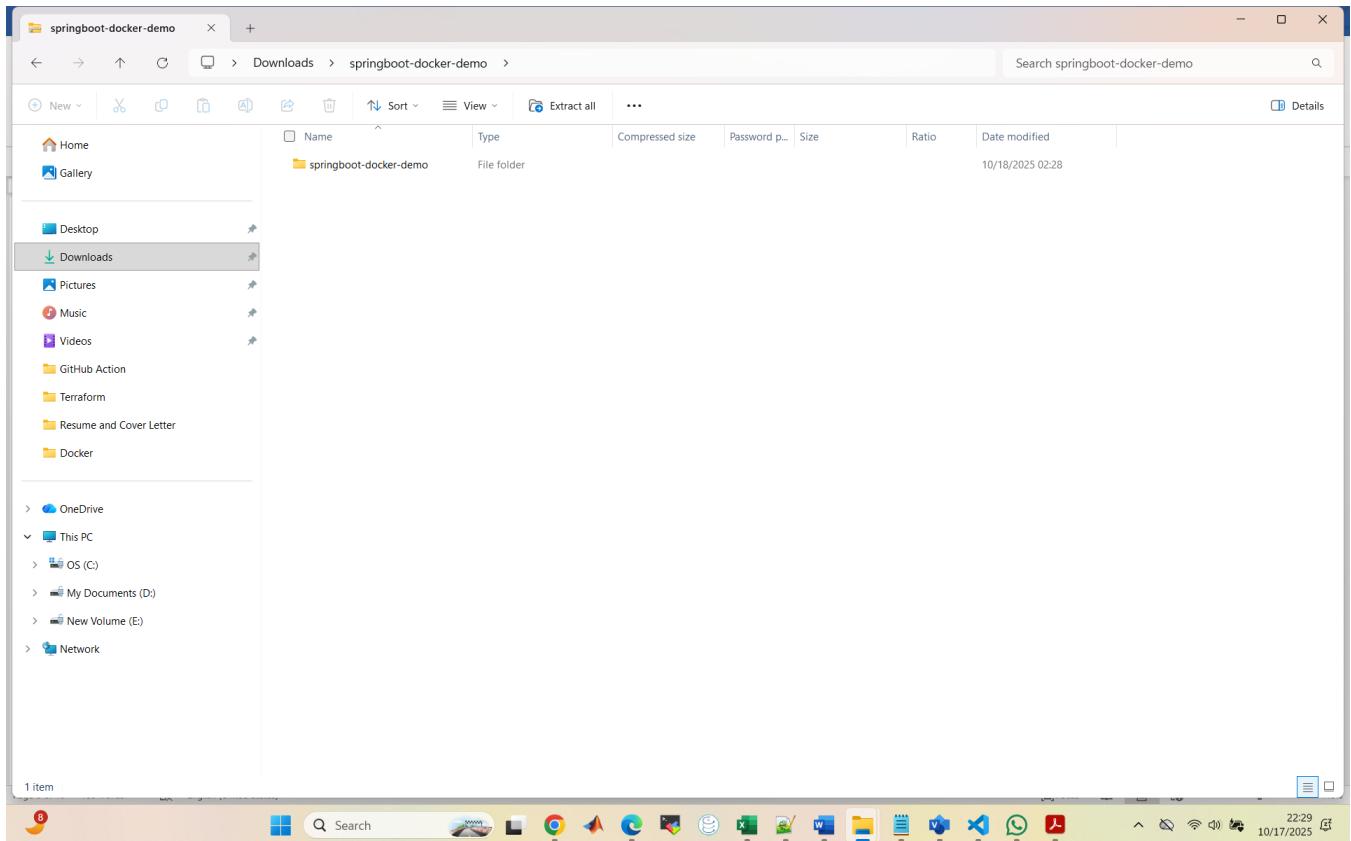
Click on “Spring Web”



Then click on “Generate CTRL”

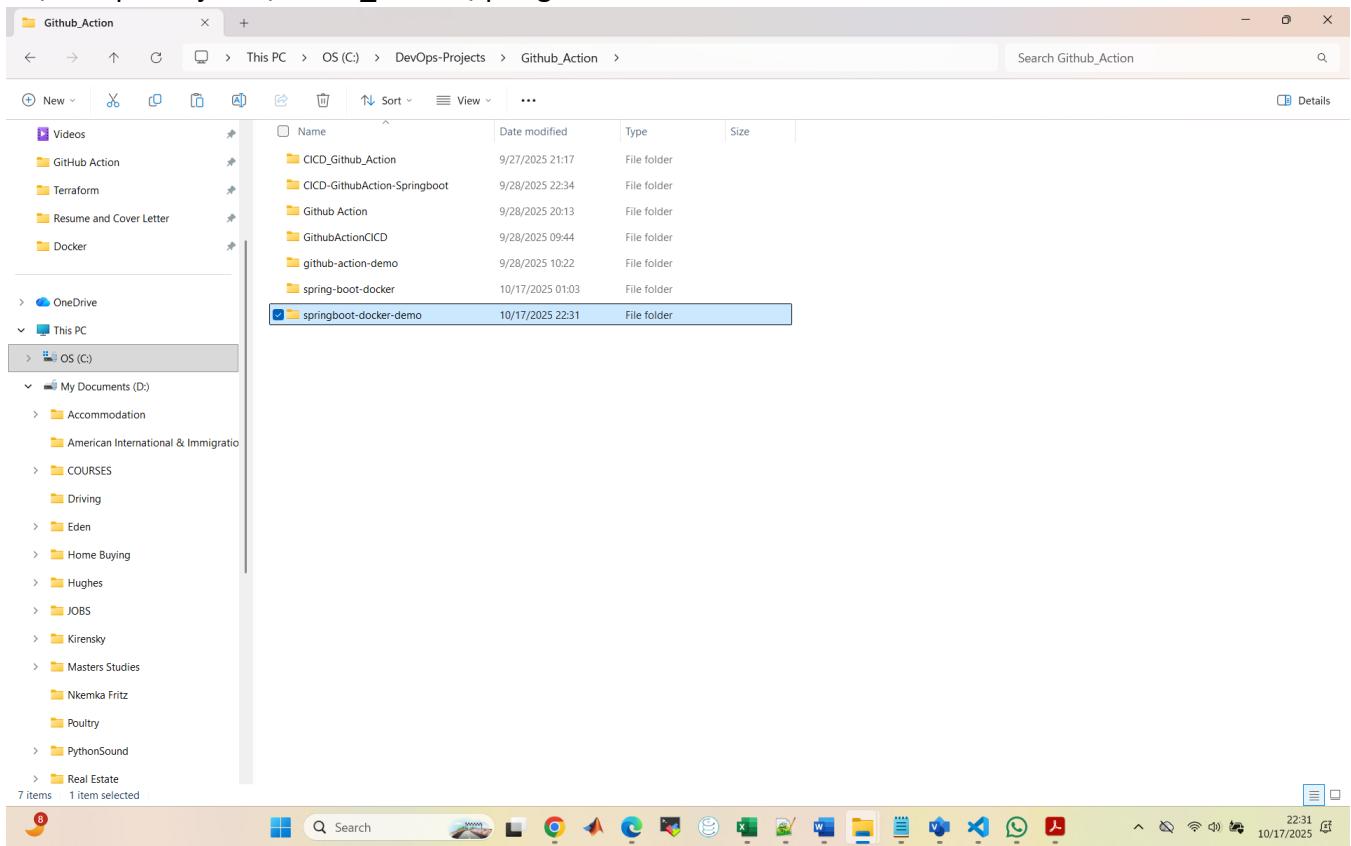


The project has been created and the project folder has been downloaded to the “Downloads” folder.



Copy the project folder and save in a suitable location. I will save it in this location:

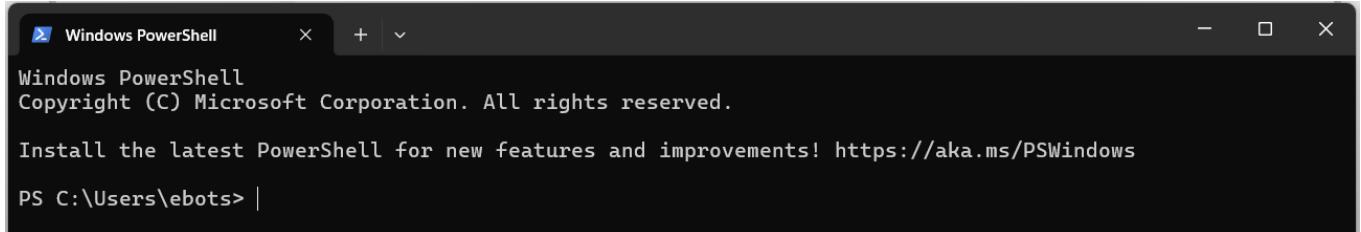
C:\DevOps-Projects\Github_Action\springboot-docker-demo



STEP 3: Build a Simple REST API and Test it

We will build a simple REST API and test it. We will do this by opening our project folder in VSCode. This is will be done on PowerShell.

Part 1: Open PowerShell and Navigate to Project Folder

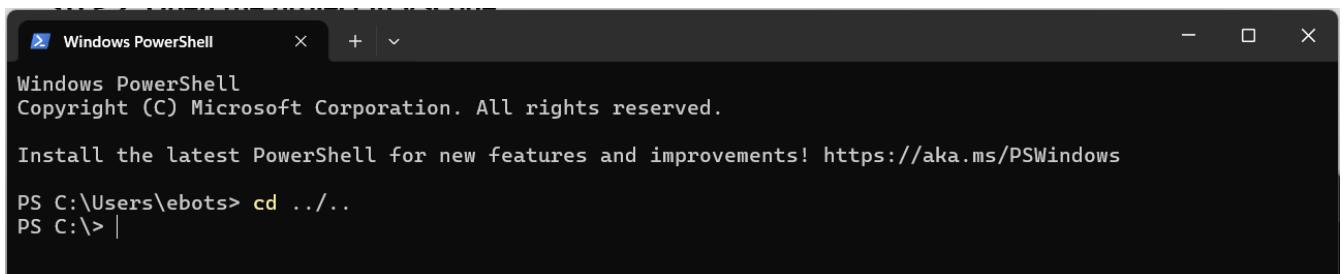


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\ebots> |
```

Navigate to the project folder. We have to first go to the C drive



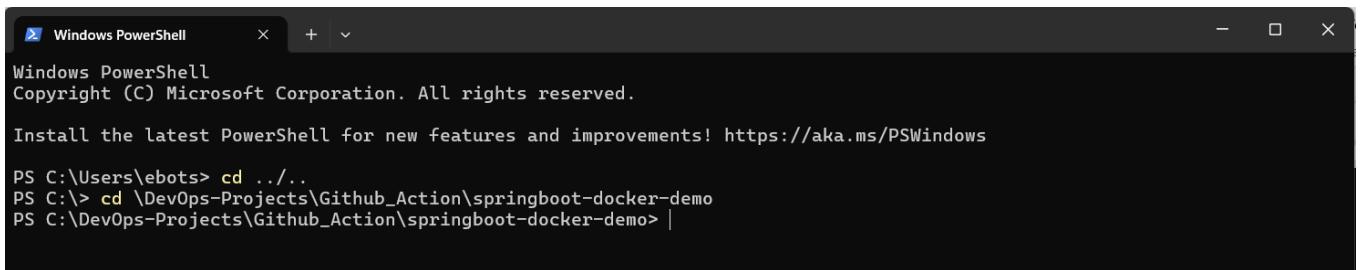
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\ebots> cd ../..
PS C:> |
```

Then go to the project folder using the command:

```
cd \DevOps-Projects\Github_Action\springboot-docker-demo
```

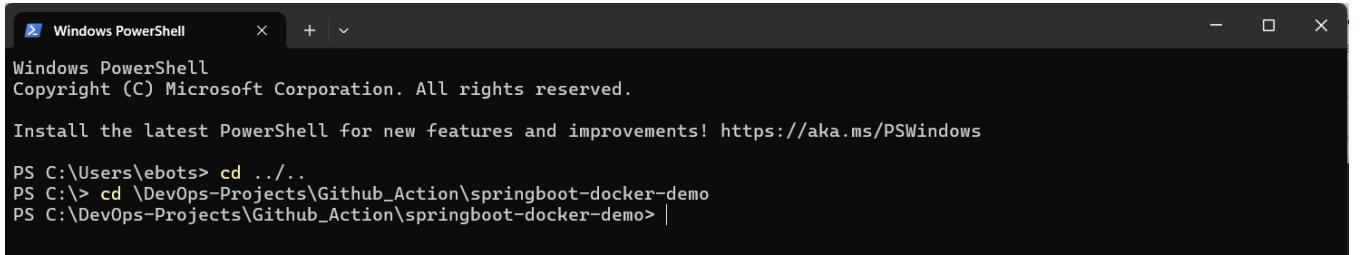


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\ebots> cd ../..
PS C:> cd \DevOps-Projects\Github_Action\springboot-docker-demo
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> |
```

Part 2: Open the project folder in VSCode



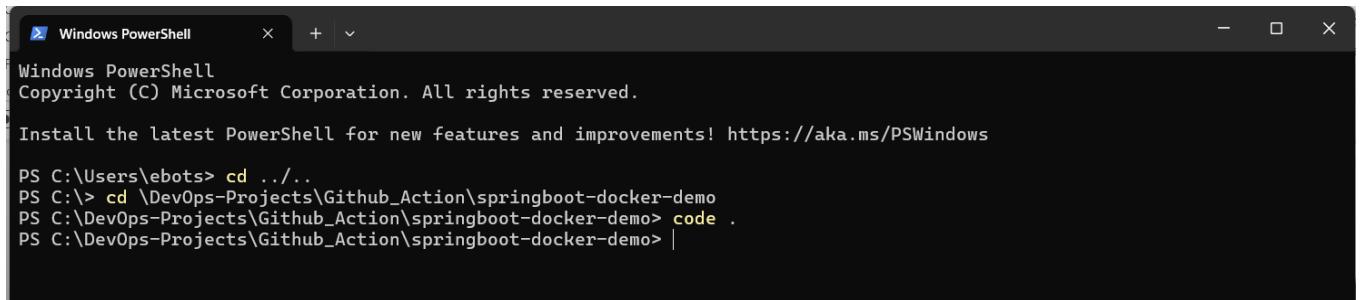
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\ebots> cd ../..
PS C:> cd \DevOps-Projects\Github_Action\springboot-docker-demo
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> |
```

We will now open the project folder in VSCode using the command:

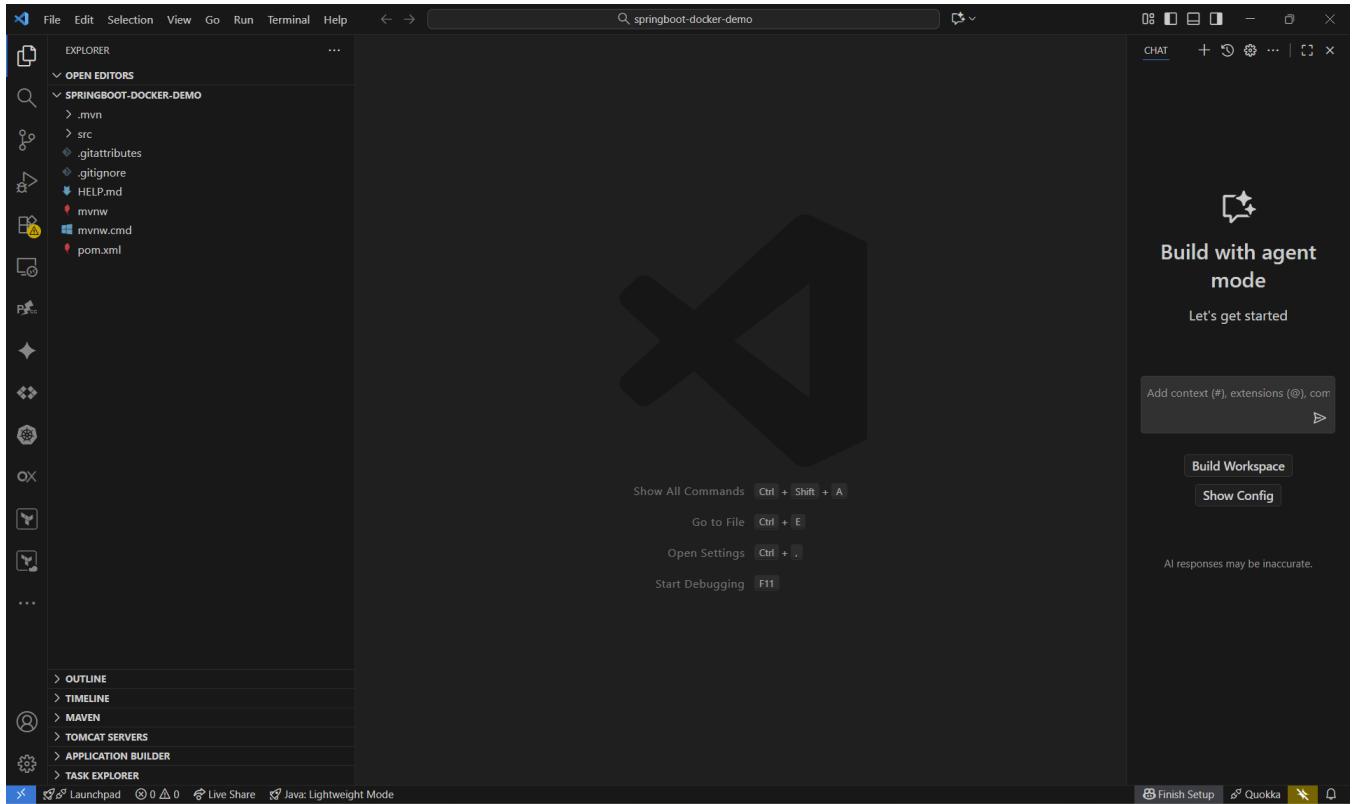
```
code .
```



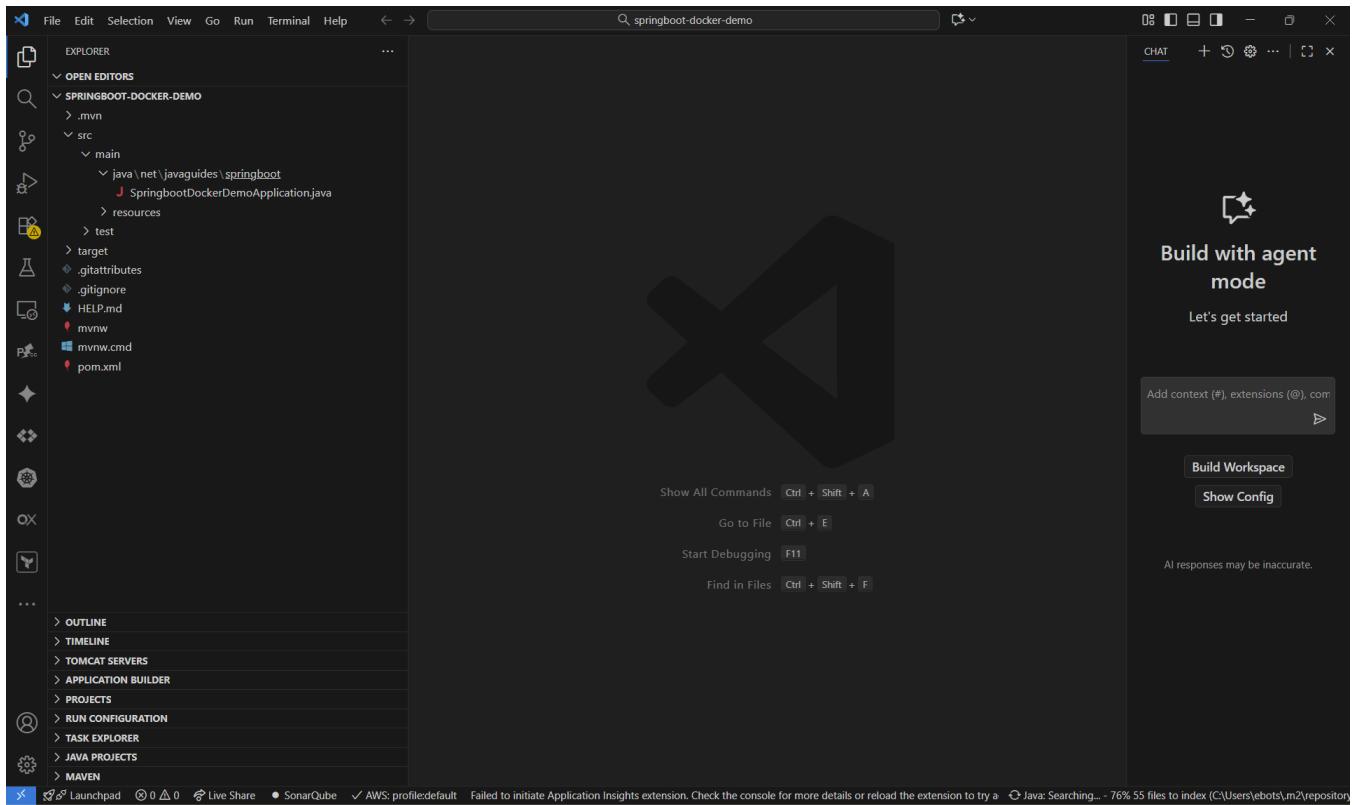
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

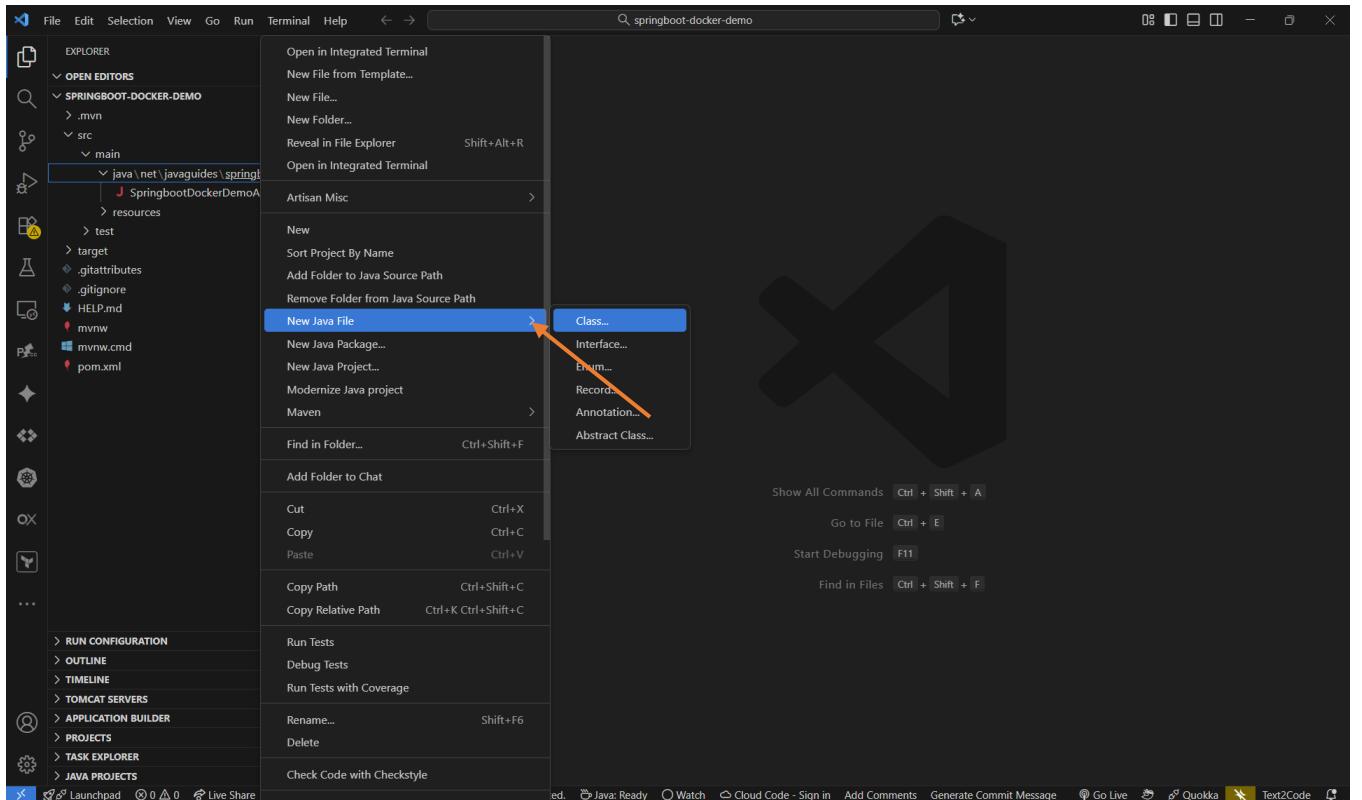
PS C:\Users\ebots> cd ../..
PS C:\> cd \DevOps-Projects\Github_Action\springboot-docker-demo
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> code .
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo>
```



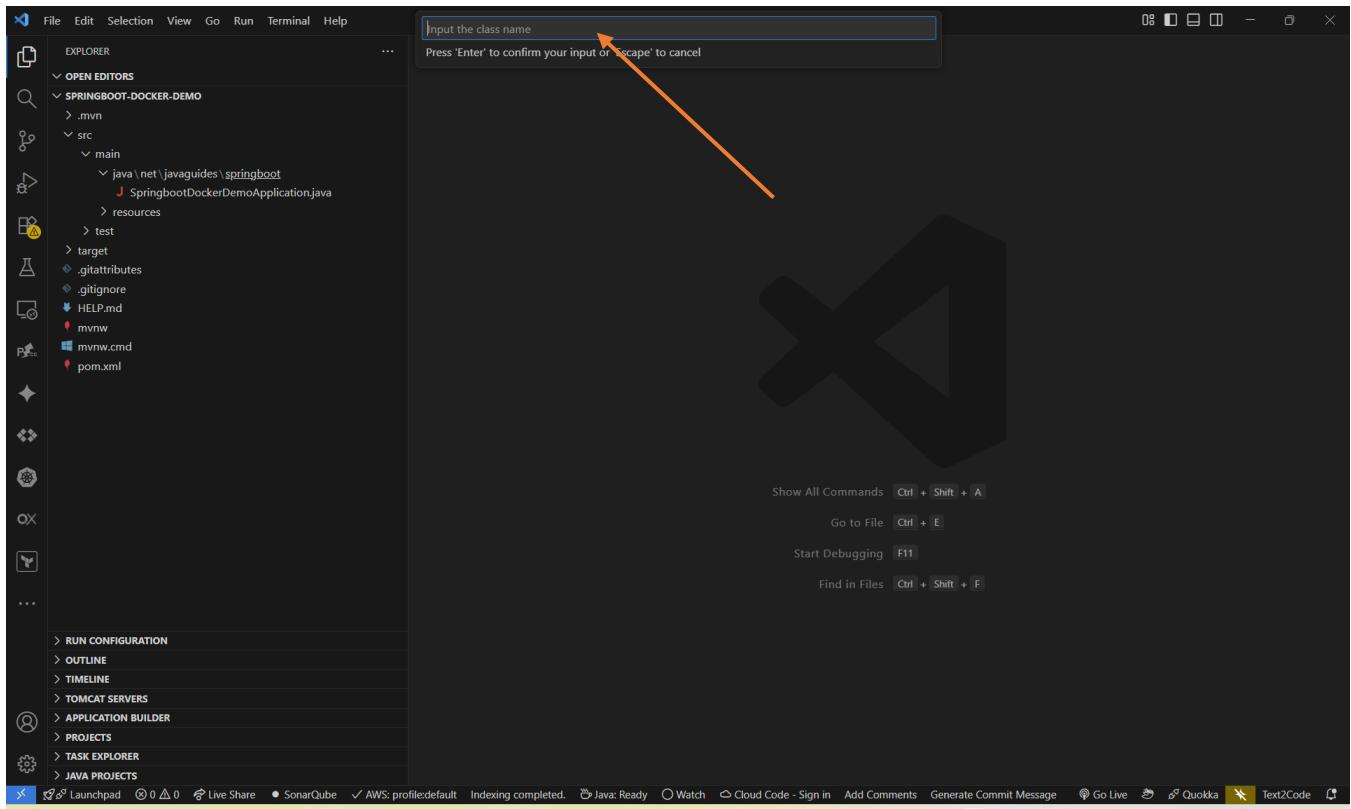
Let us create a package in the folder “src” -> “main” -> “Java”



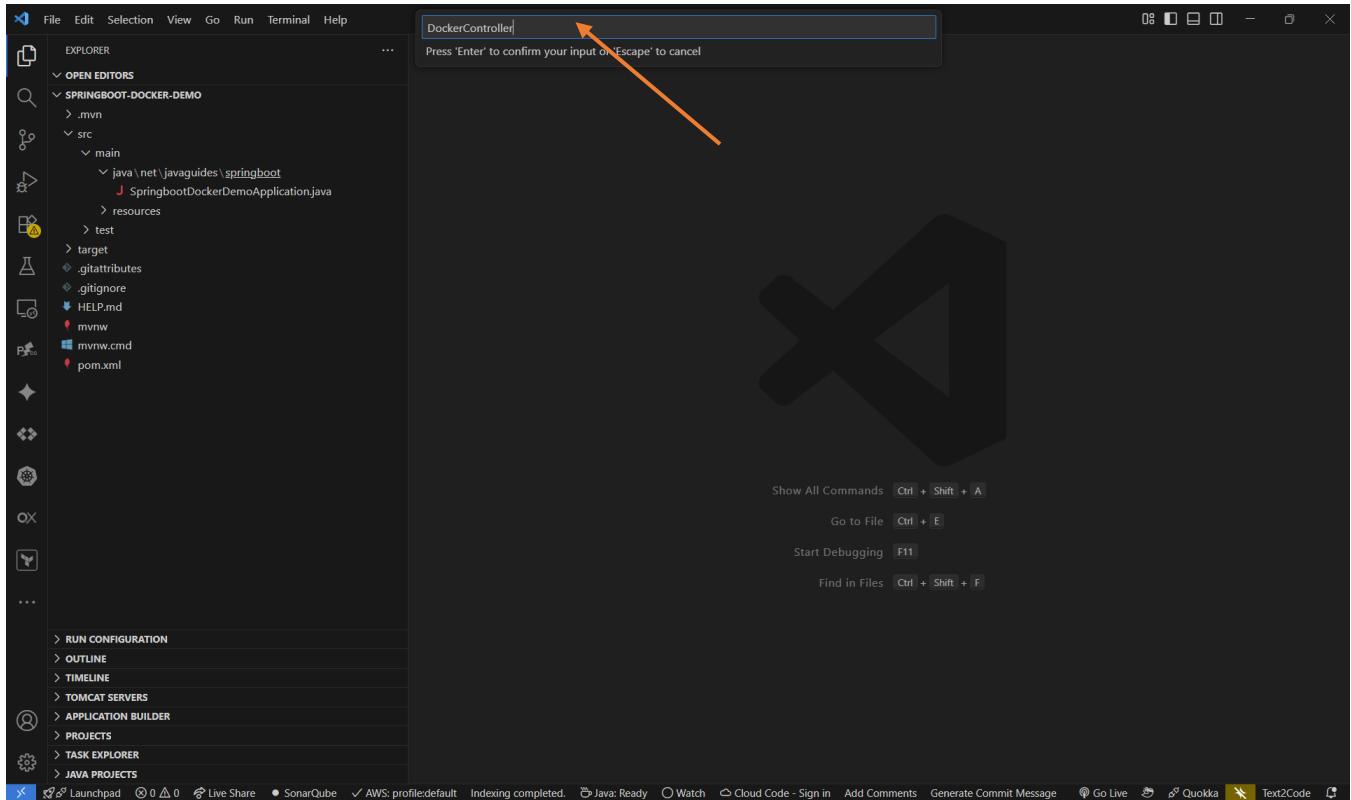
We have to create the endpoint called "DockerController". To do this, right-click on "java\net\javaguides\springboot-docker-demo"



Select "Class"



We will create a simple endpoint called “**docker**”. So, type “**DockerController**”



and press Enter

```

1 package net.javaguides.springboot;
2
3 public class DockerController {
4
5 }

```

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows the project structure for "SPRINGBOOT-DOCKER-DEMO". It includes a ".mvn" folder, a "src" folder containing "main" and "java\javaguides\springboot" subfolders, and files like ".gitattributes", ".gitignore", "HELP.md", "mvnw", "mvnw.cmd", and "pom.xml".
- Editor:** The current file is "DockerController.java" in the "src/main/java/net/javaguides/springboot" directory. The code is as follows:

```

1 package net.javaguides.springboot;
2
3 public class DockerController {
4
5 }

```
- Bottom Status Bar:** Shows various icons and status messages, including "AWS: profiledefault", "Indexing completed.", "Java: Ready", "SonarQube focus: overall code", and "Prettier".

Let us go ahead and annotate this controller class to make this class a REST API controller. We will add this line of code.

@RestController

```

1 package net.javaguides.springboot;
2
3 import org.springframework.web.bind.annotation.RestController;
4
5 @RestController
6 public class DockerController {
7
8 }

```

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows the project structure for "SPRINGBOOT-DOCKER-DEMO". It includes a ".mvn" folder, a "src" folder containing "main" and "java\javaguides\springboot" subfolders, and files like ".gitattributes", ".gitignore", "HELP.md", "mvnw", "mvnw.cmd", and "pom.xml".
- Editor:** The current file is "DockerController.java" in the "src/main/java/net/javaguides/springboot" directory. The code now includes the "@RestController" annotation:

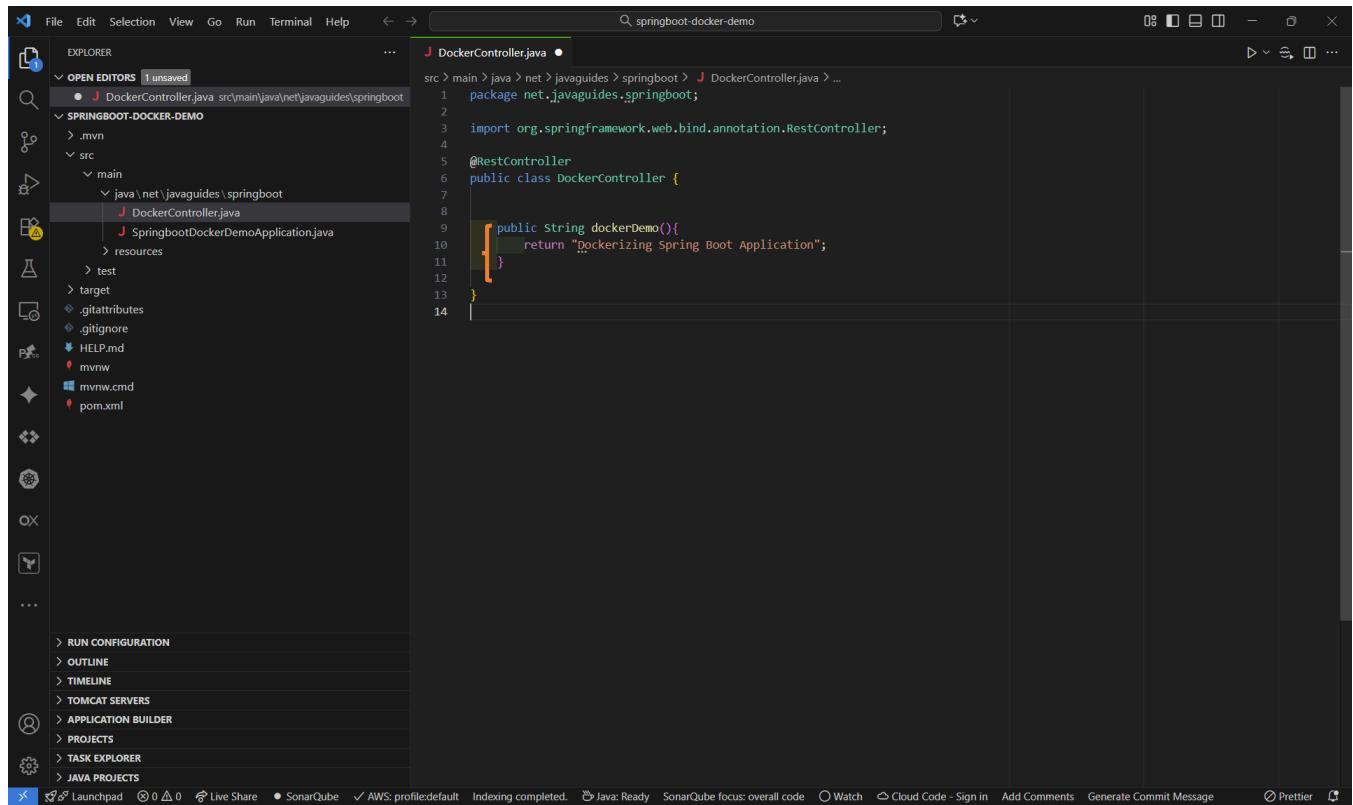
```

1 package net.javaguides.springboot;
2
3 import org.springframework.web.bind.annotation.RestController;
4
5 @RestController
6 public class DockerController {
7
8 }

```
- Bottom Status Bar:** Shows various icons and status messages, including "AWS: profiledefault", "Indexing completed.", "Java: Ready", "SonarQube focus: overall code", and "Prettier".

Within this controller, we can define a REST API by adding a method called “dockerDemo” that will return a string “Dockerizing Spring Boot Application”

```
public String dockerDemo() {
    return "Dockerizing Spring Boot Application";
}
```



Let us annotate this method with by adding the line

```
@GetMapping("/docker")
```

In this line, we have added an annotation called “GetMapping” with API URL called “docker”.

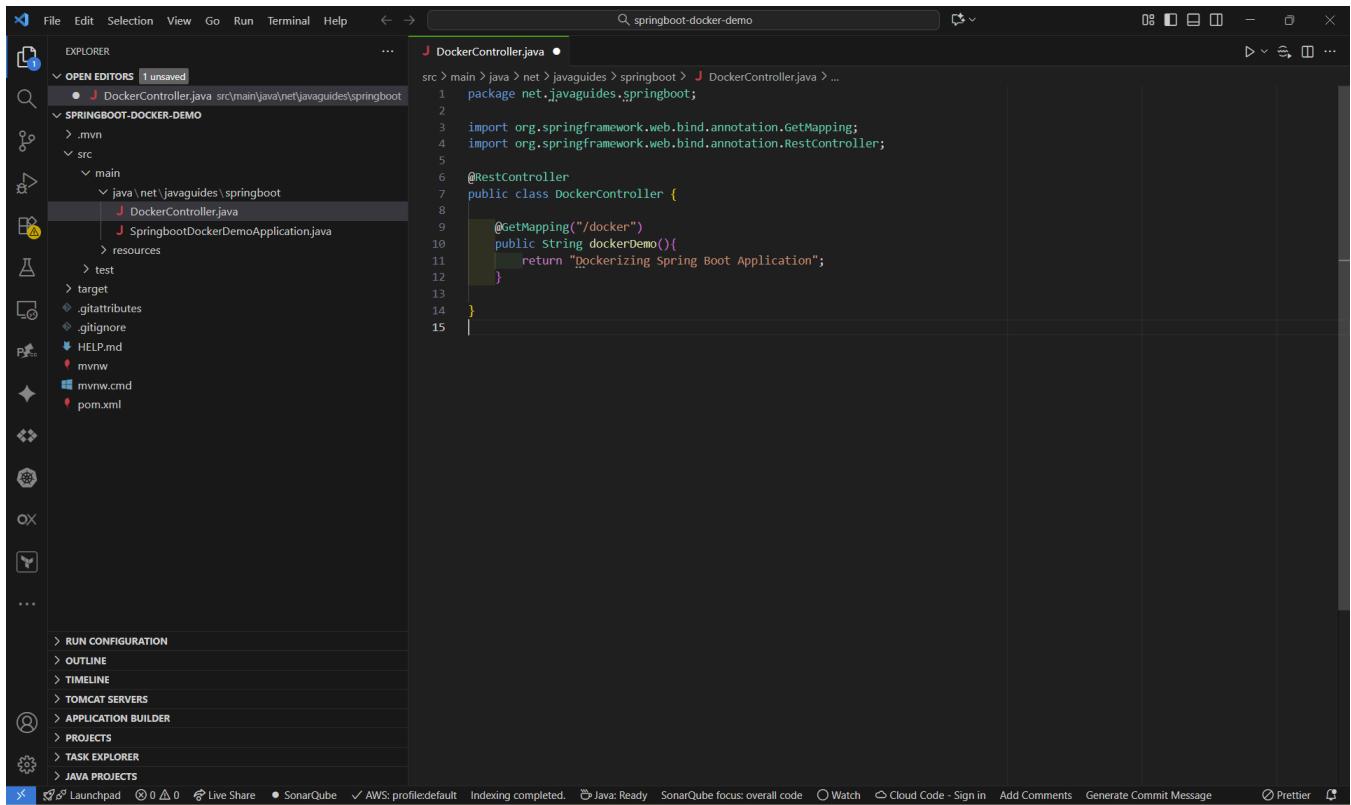
```
package net.javaguides.controller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class DockerController {

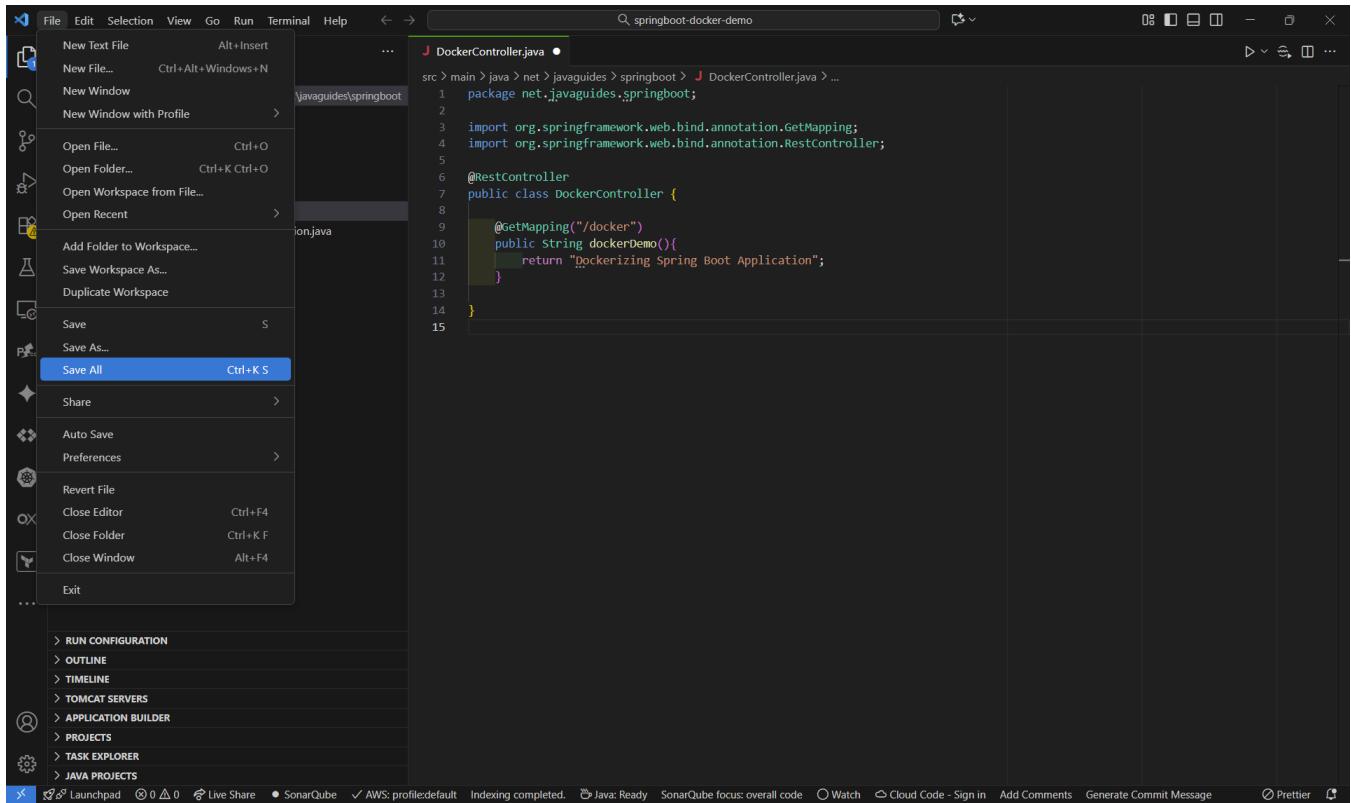
    @GetMapping("/docker")
    public String dockerDemo() {
        return "Dockerizing Spring Boot Application";
    }
}
```

Prepared by Sidney Smith Ebot



```
src > main > java > net > javaguides > springboot > DockerController.java > ...
1 package net.javaguides.springboot;
2
3 import org.springframework.web.bind.annotation.GetMapping;
4 import org.springframework.web.bind.annotation.RestController;
5
6 @RestController
7 public class DockerController {
8
9     @GetMapping("/docker")
10    public String dockerDemo(){
11        return "Dockerizing Spring Boot Application";
12    }
13
14 }
```

Now, we have built a simple REST API. Save the project by clicking on “File”



Then select “Save All”

```

src > main > java > net > javaguides > springboot > DockerController.java ...
1 package net.javaguides.springboot;
2
3 import org.springframework.web.bind.annotation.GetMapping;
4 import org.springframework.web.bind.annotation.RestController;
5
6 @RestController
7 public class DockerController {
8
9     @GetMapping("/docker")
10    public String dockerDemo(){
11        return "Dockerizing Spring Boot Application";
12    }
13
14 }
15

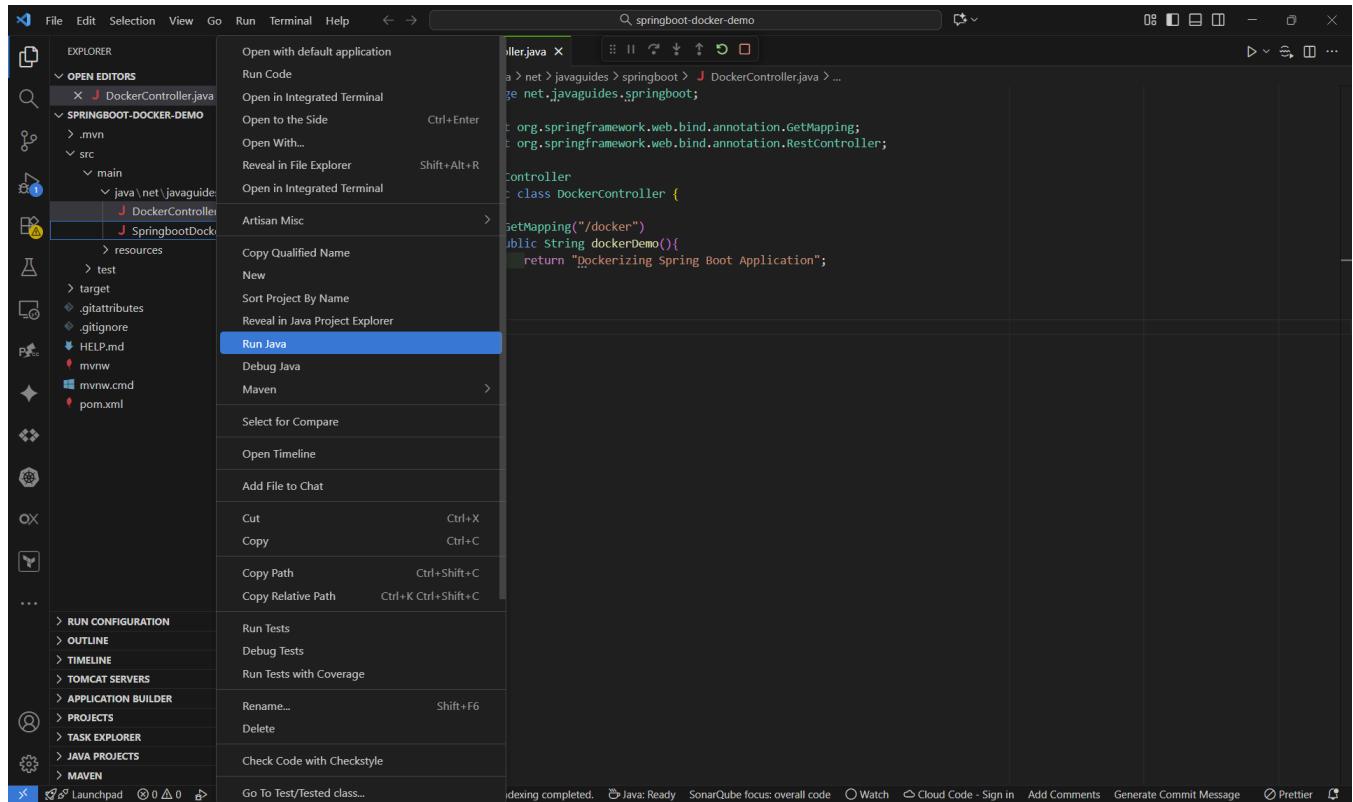
```

The screenshot shows the VS Code interface with the DockerController.java file open in the editor. The code defines a REST controller with a single endpoint mapping to '/docker'.

Part 3: Test the REST API

Now, let us test the application by running it.

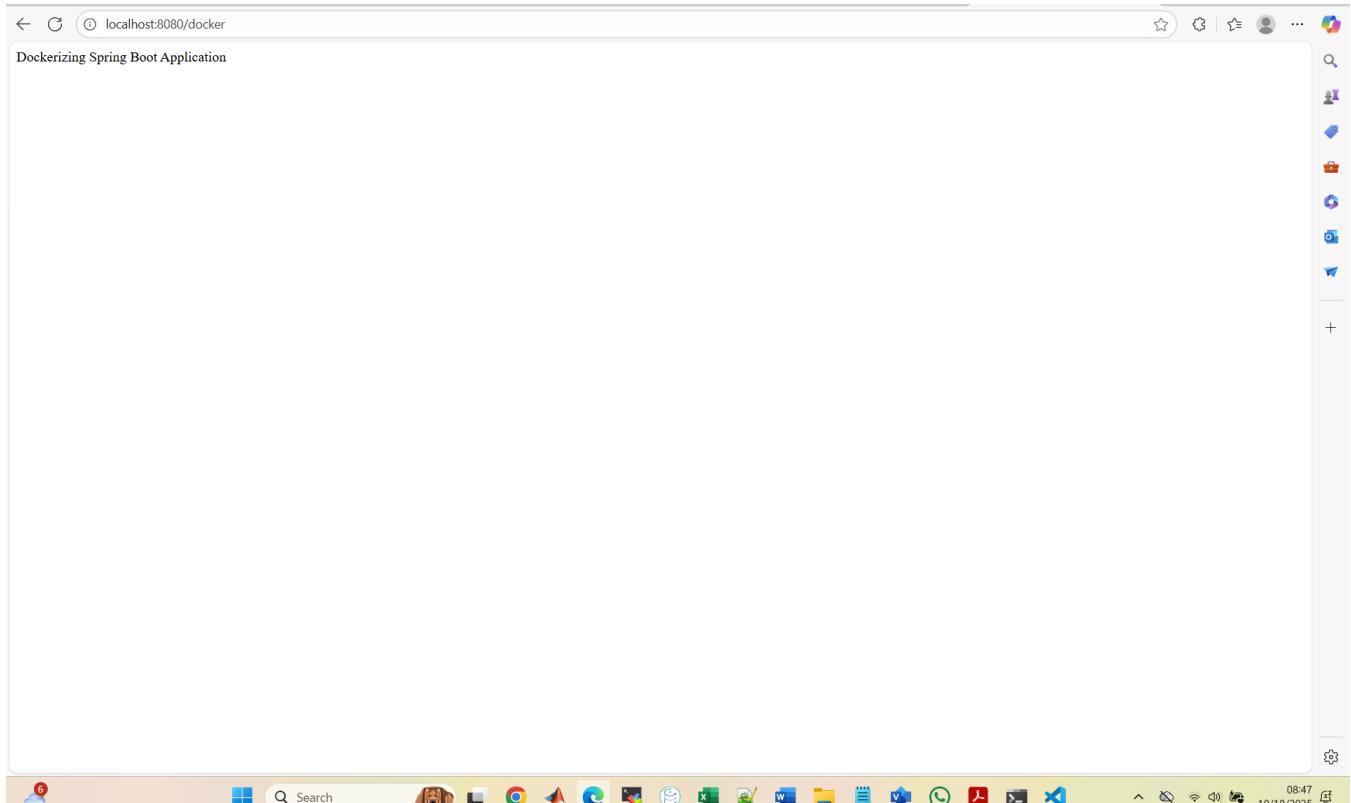
To do this, right-click on “**SpringbootDockerDemoApplication.java**”



Select “Run Java”

```
src > main > java > net > javaguides > springboot > DockerController.java ...  
1 package net.javaguides.springboot;  
2  
3 import org.springframework.web.bind.annotation.GetMapping;  
4 import org.springframework.web.bind.annotation.RestController;  
5  
6 @RestController  
7 public class DockerController {  
8  
9     @GetMapping("/docker")  
10    public String dockerDemo(){  
11        return "Dockering Spring Boot Application";  
12    }  
13  
14 }  
  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE SPELL CHECKER 3 QUERY RESULTS ... Run:SpringbootDockerDemoApplication + ⌂ X  
2025-10-19T11:09:38.246-04:00 INFO 20952 --- [springboot-docker-demo] [ng Servlet engine: [Apache Tomcat/10.1.46]  
2025-10-19T11:09:38.438-04:00 INFO 20952 --- [springboot-docker-demo] [lizing Spring embedded webApplicationContext  
2025-10-19T11:09:38.444-04:00 INFO 20952 --- [springboot-docker-demo] [ebApplicationContext: initialization completed in 1757 ms  
2025-10-19T11:09:39.051-04:00 INFO 20952 --- [springboot-docker-demo] [tarted on port 8080 (http) with context path '/'  
2025-10-19T11:09:39.076-04:00 INFO 20952 --- [springboot-docker-demo] [pringbootDockerDemoApplication in 3.199 seconds (process running for 3.583)  
2025-10-19T11:09:44.372-04:00 INFO 20952 --- [springboot-docker-demo] [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initia  
lizing Spring DispatcherServlet 'dispatcherServlet'  
2025-10-19T11:09:44.374-04:00 INFO 20952 --- [springboot-docker-demo] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initia  
lizing Servlet 'dispatcherServlet'  
2025-10-19T11:09:44.376-04:00 INFO 20952 --- [springboot-docker-demo] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Compl  
eted initialization in 2 ms  
  
main] o.apache.catalina.core.StandardEngine : Starti  
main] o.a.c.c.C.[Tomcat].[localhost].[] : Initia  
main] w.s.c.ServletWebServerApplicationContext : Root w  
main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat  
started on port 8080 (http) with context path '/'  
main] n.j.s.SpringbootDockerDemoApplication : Starte  
d SpringbootDockerDemoApplication in 3.199 seconds (process running for 3.583)  
main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initia  
lizing Spring DispatcherServlet 'dispatcherServlet'  
main] n.j.s.SpringbootDockerDemoApplication : Initia  
lizing Servlet 'dispatcherServlet'  
main] o.s.web.servlet.DispatcherServlet : Comple  
ted initialization in 2 ms  
  
Live Share • SonarQube ✓ AWS: profiledefault Indexing completed. ⚡ Java: Ready SonarQube focus: overall code ⚡ Watch ⚡ Cloud Code - Sign in Add Comments Generate Commit Message ⚡ Prettier ⚡
```

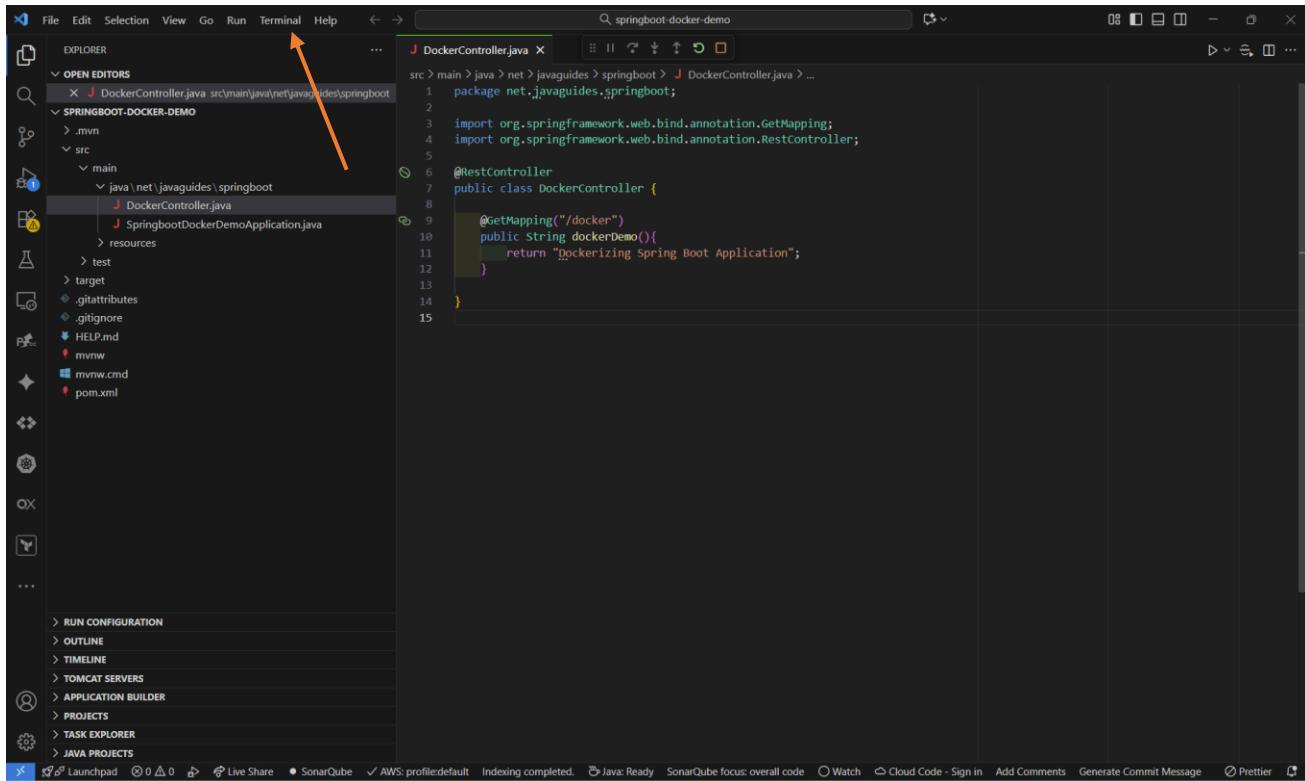
Now, verify on your browser by typing: **localhost:8080/docker**



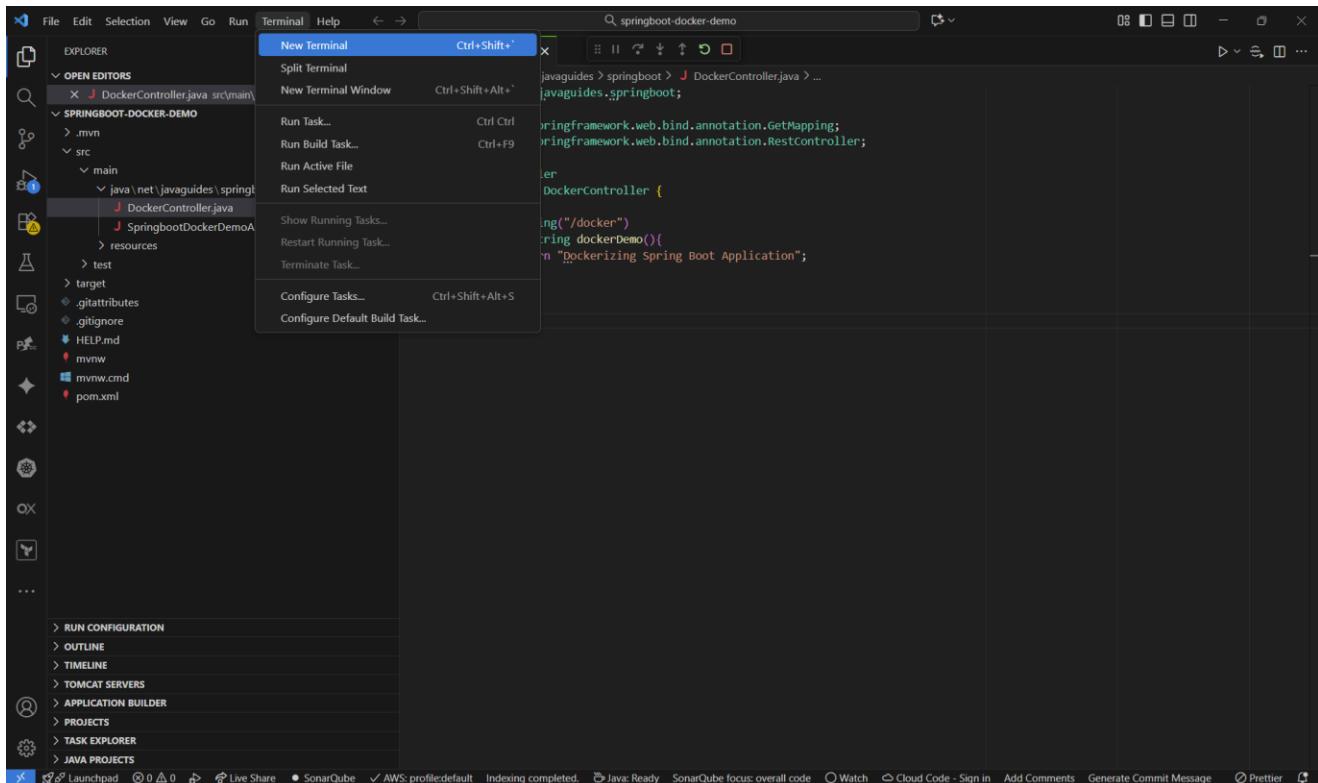
The application is working. We can now proceed to Dockerize the application.

STEP 3: Build and Package Maven Application

We have to open a new terminal.

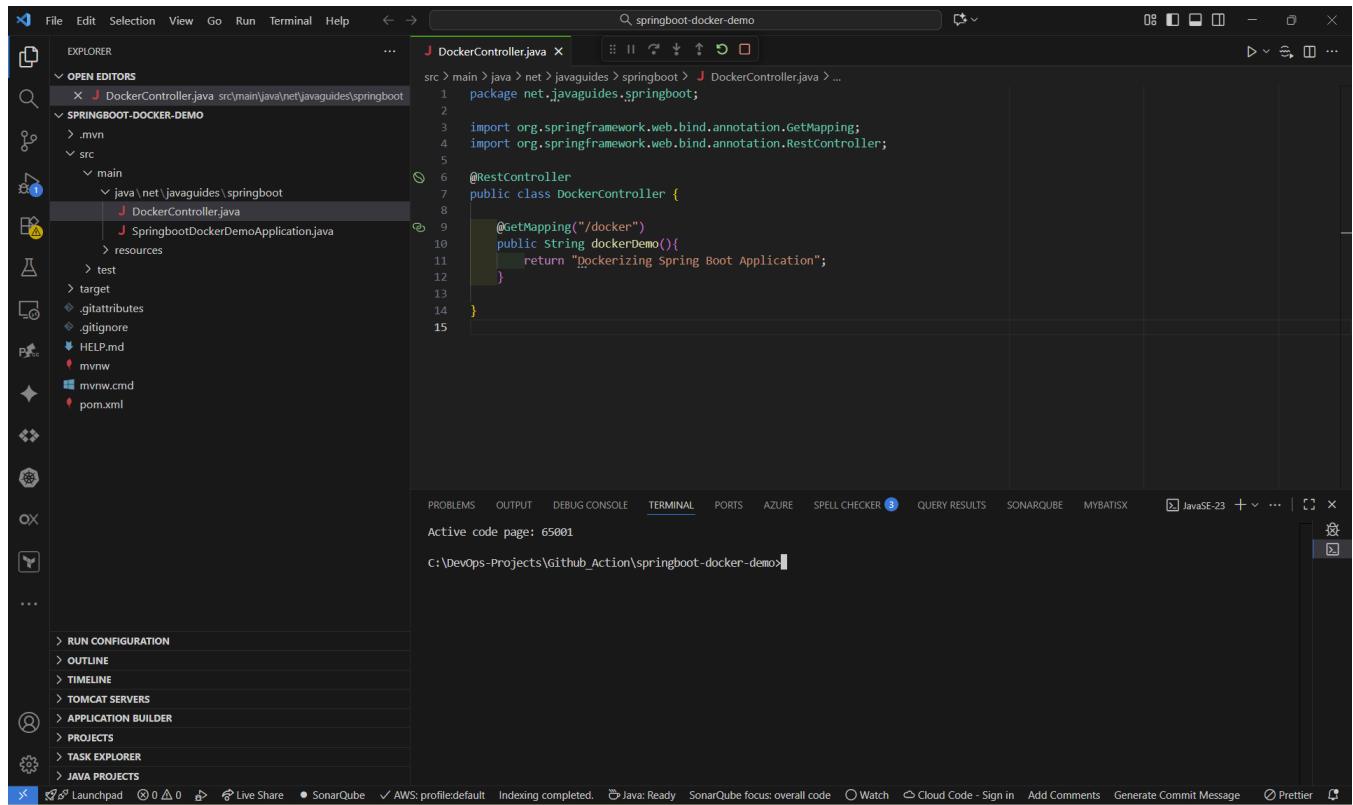


Click on “Terminal”



Select “New Terminal”

Prepared by Sidney Smith Ebot



```
src > main > java > net > javaguides > springboot > DockerController.java ...
1 package net.javaguides.springboot;
2
3 import org.springframework.web.bind.annotation.GetMapping;
4 import org.springframework.web.bind.annotation.RestController;
5
6 @RestController
7 public class DockerController {
8
9     @GetMapping("/docker")
10    public String dockerDemo(){
11        return "Dockerizing Spring Boot Application";
12    }
13
14 }
```

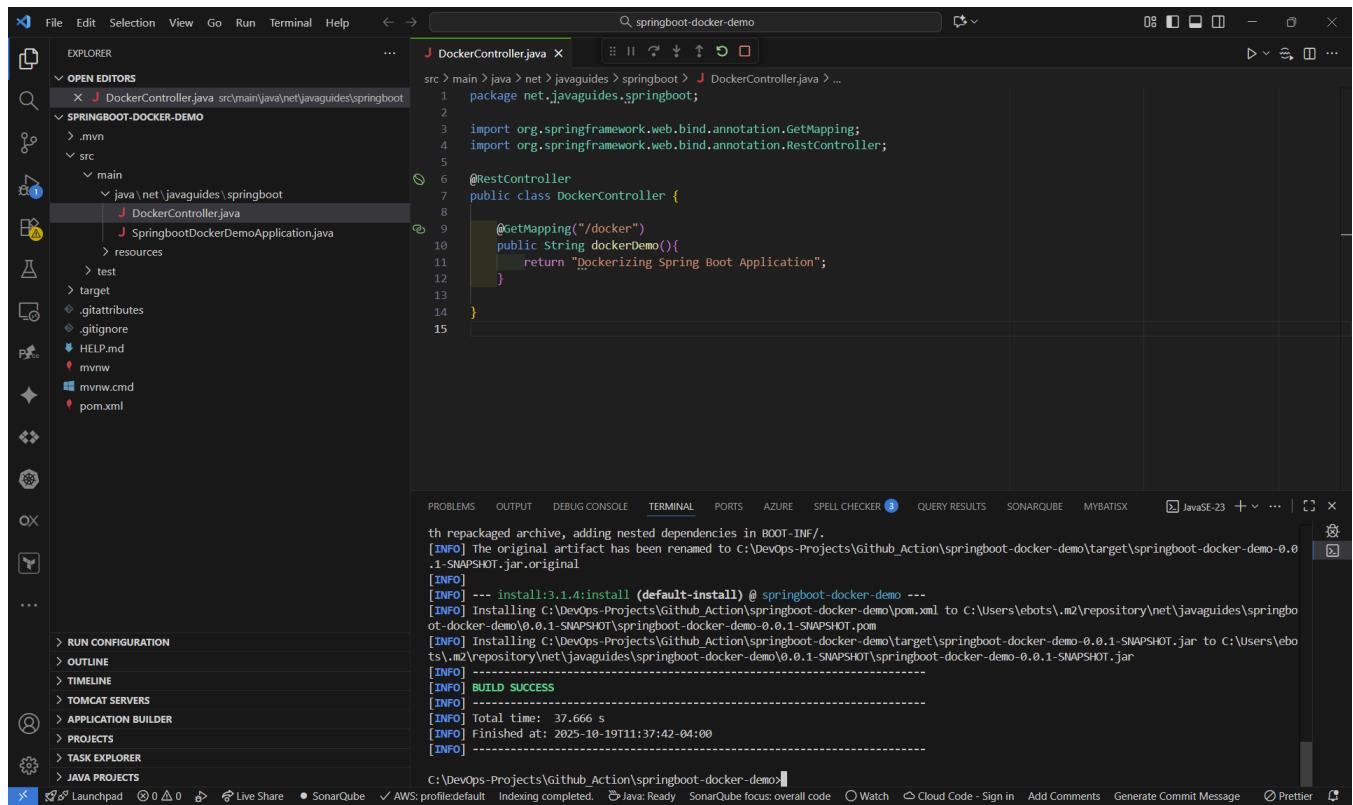
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE SPELL CHECKER 3 QUERY RESULTS SONARQUBE MYBATISX JavaSE-23 + ...

Active code page: 65001

C:\DevOps-Projects\Github_Action\springboot-docker-demo>

Run the command to install Maven:

```
mvn clean install
```



```
src > main > java > net > javaguides > springboot > DockerController.java ...
1 package net.javaguides.springboot;
2
3 import org.springframework.web.bind.annotation.GetMapping;
4 import org.springframework.web.bind.annotation.RestController;
5
6 @RestController
7 public class DockerController {
8
9     @GetMapping("/docker")
10    public String dockerDemo(){
11        return "Dockerizing Spring Boot Application";
12    }
13
14 }
```

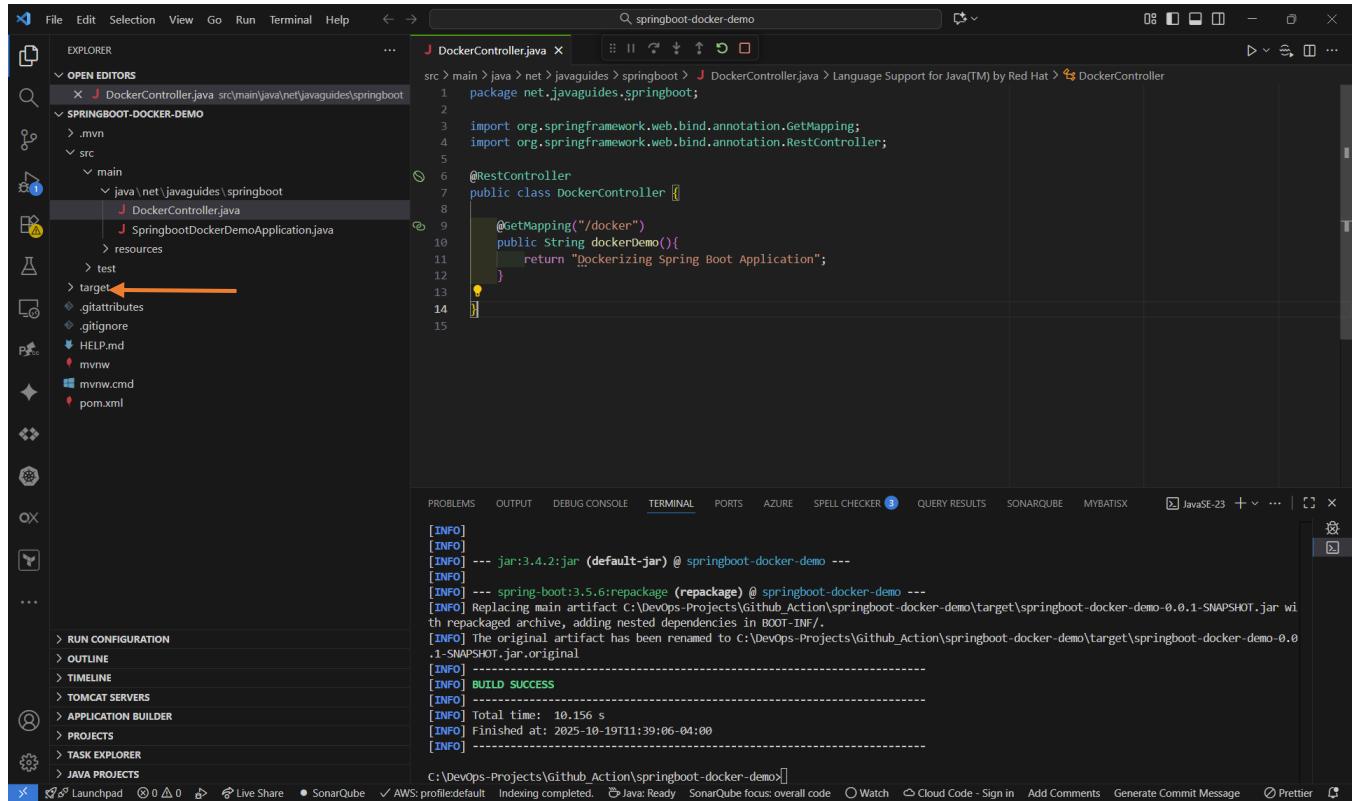
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE SPELL CHECKER 3 QUERY RESULTS SONARQUBE MYBATISX JavaSE-23 + ...

```
th repackaged archive, adding nested dependencies in BOOT-INF/
[INFO] The original artifact has been renamed to C:\DevOps-Projects\Github_Action\springboot-docker-demo\target\springboot-docker-demo-0.0.1-SNAPSHOT.jar.original
[INFO]
[INFO] --- install:3.1.4:install (default-install) @ springboot-docker-demo ---
[INFO] Installing C:\DevOps-Projects\Github_Action\springboot-docker-demo\pom.xml to C:\Users\ebots\.m2\repository\net\javaguides\springboot-docker-demo\0.0.1-SNAPSHOT\springboot-docker-demo-0.0.1-SNAPSHOT.pom
[INFO] Installing C:\DevOps-Projects\Github_Action\springboot-docker-demo\target\springboot-docker-demo-0.0.1-SNAPSHOT.jar to C:\Users\ebots\.m2\repository\net\javaguides\springboot-docker-demo\0.0.1-SNAPSHOT\springboot-docker-demo-0.0.1-SNAPSHOT.jar
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] total time: 37.666 s
[INFO] Finished at: 2025-10-19T11:37:42+04:00
[INFO] -----
```

C:\DevOps-Projects\Github_Action\springboot-docker-demo>

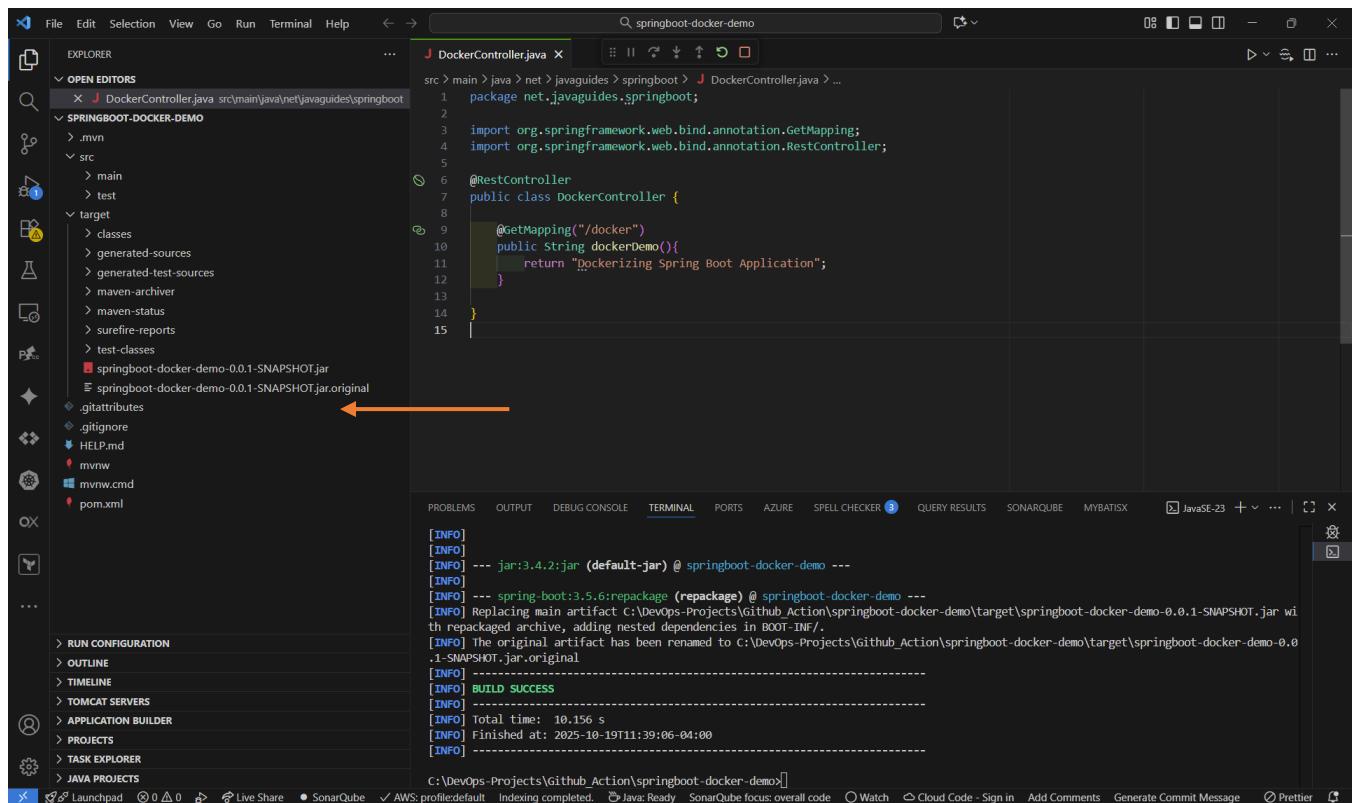
Then run the command to package Maven:

mvn package



```
[INFO]
[INFO] --- jar:3.4.2:jar (default-jar) @ springboot-docker-demo ---
[INFO]
[INFO] --- spring-boot:3.5.6:repackage (repackage) @ springboot-docker-demo ---
[INFO] Replacing main artifact C:\DevOps-Projects\Github_Action\springboot-docker-demo\target\springboot-docker-demo-0.0.1-SNAPSHOT.jar with repackaged archive, adding nested dependencies in BOOT-INF.
[INFO] The original artifact has been renamed to C:\DevOps-Projects\Github_Action\springboot-docker-demo\target\springboot-docker-demo-0.0.1-SNAPSHOT.jar.original
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 10.156 s
[INFO] Finished at: 2025-10-19T11:39:06-04:00
[INFO] -----
```

To verify the Maven build is successful. Click on the “target” folder

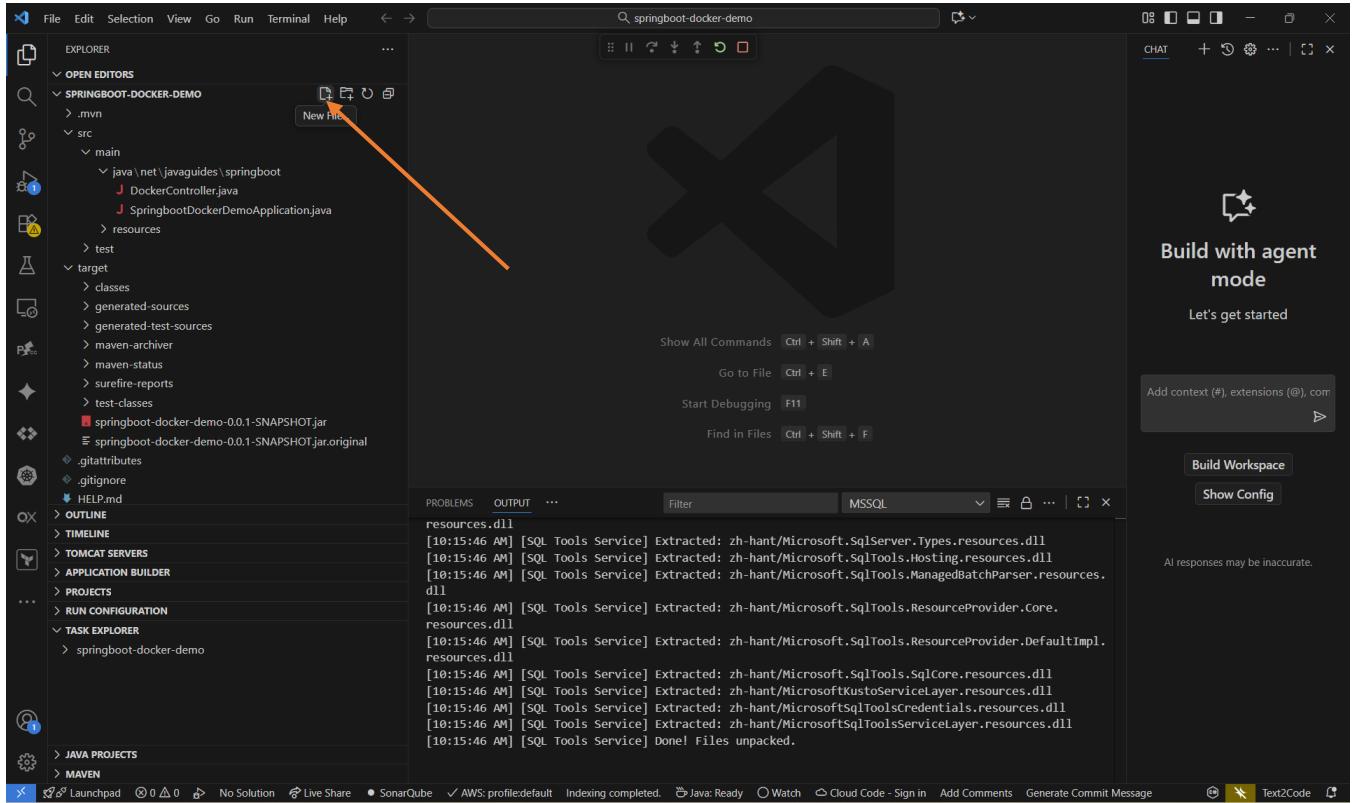


You can see that “Springboot-docker-demo-0.0.1-SNAPSHOT.jar” has been created.

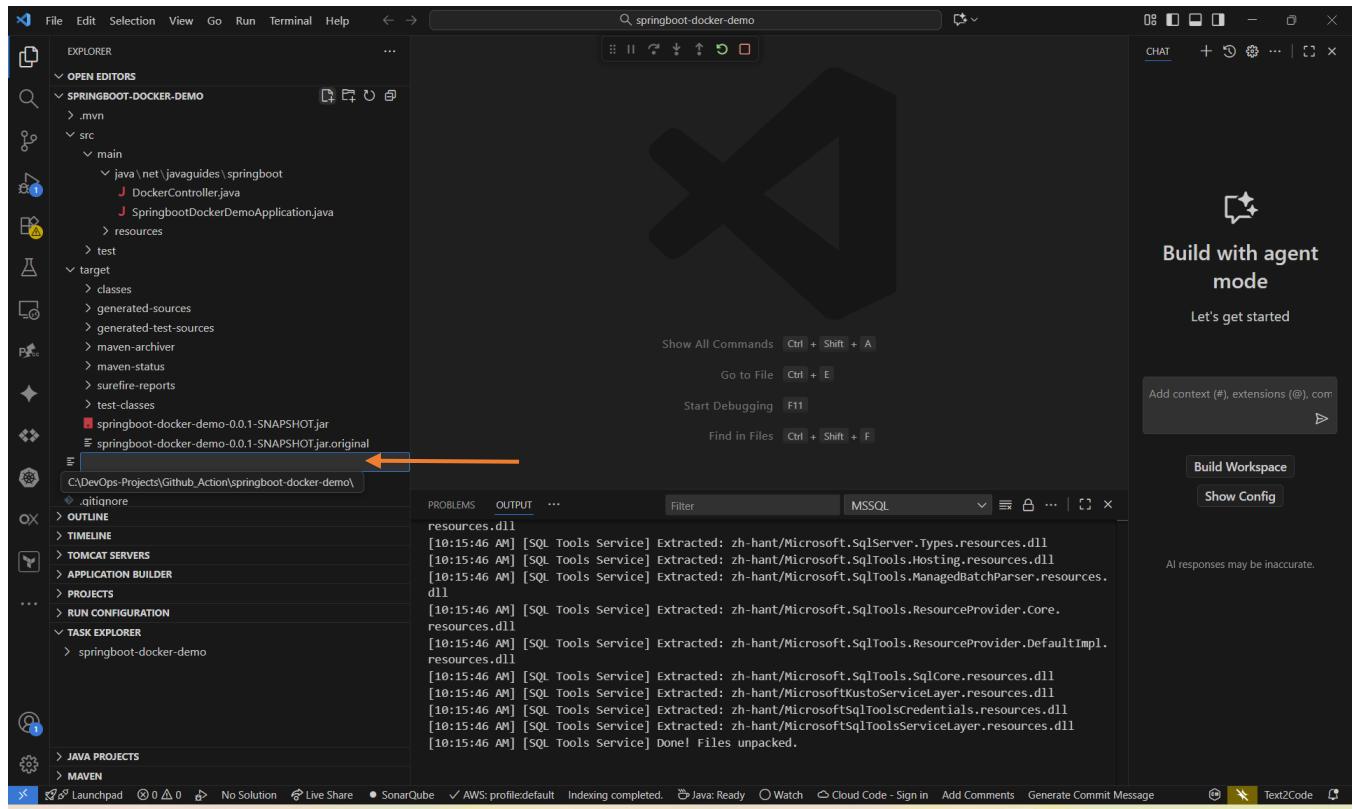
STEP 4: Create a Dockerfile

Docker can build images automatically by reading the instructions from a Dockerfile. A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image. This page describes the commands you can use in a Dockerfile.

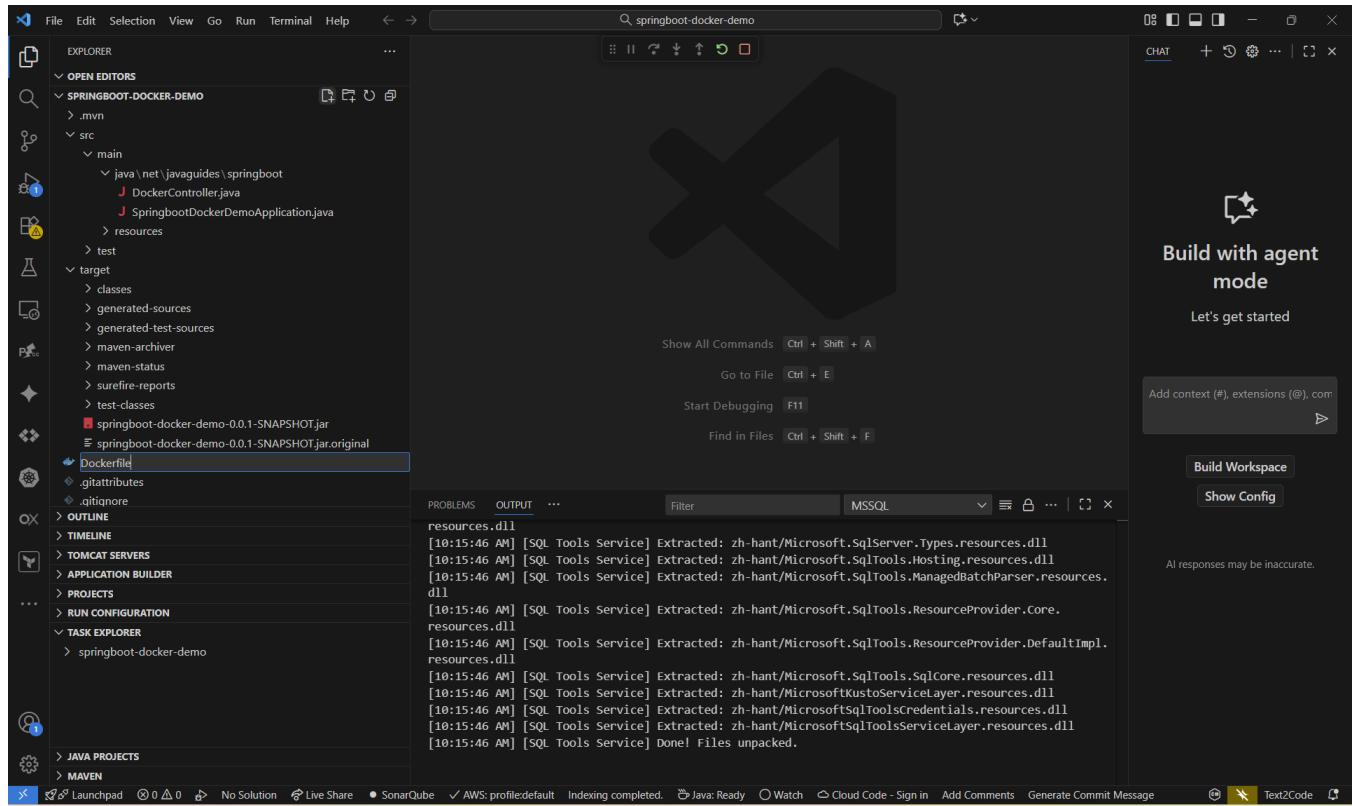
We will create a Dockerfile where we will define all the instructions and commands to build the Docker Image.



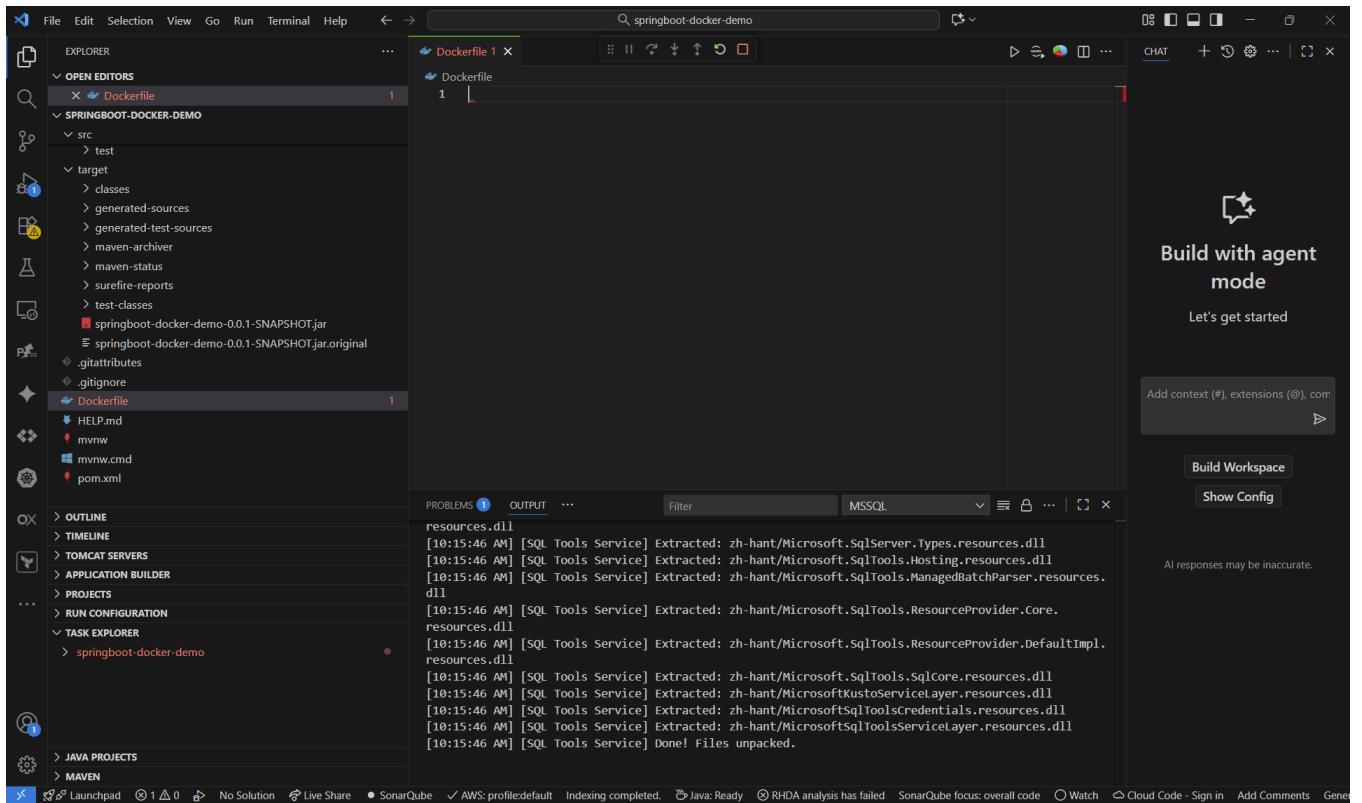
Let us create a new file in our project. Click on “New File”



Enter the name of the file as “**Dockerfile**”



Press Enter



Let us now write the commands on our Dockerfile

```
# Use an official Eclipse Temurin base image
FROM eclipse-temurin:17

LABEL maintainer="ebotsmith@gmail.com"

# Set the working directory inside the container
WORKDIR /app

# Copy the application's JAR file into the container
COPY target/springboot-docker-demo-0.0.1-SNAPSHOT.jar /springboot-docker-demo.jar

# Define the ENTRYPOINT for the container
# This command will be executed when the container starts
# Arguments provided to 'docker run' will be appended to this ENTRYPOINT
ENTRYPOINT ["java", "-jar", "springboot-docker-demo.jar"]
```

The screenshot shows the Eclipse IDE interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Left Sidebar (EXPLORER):**
 - OPEN EDITORS: 1 unsaved
 - SPRINGBOOT-DOCKER-DEMO
 - .mvn
 - src
 - target
 - classes
 - generated-sources
 - generated-test-sources
 - maven-archiver
 - maven-status
 - surefire-reports
 - test-classes
 - springboot-docker-demo-0.0.1-SNAPSHOT.jar
 - springboot-docker-demo-0.0.1-SNAPSHOT.jar.original
 - .gitattributes
 - .gitignore
 - Dockerfile
 - HELP.md
 - mvnw
 - mvnw.cmd
 - pom.xml
 - RUN CONFIGURATION
 - OUTLINE
 - TIMELINE
 - TOMCAT SERVERS
 - APPLICATION BUILDER
 - PROJECTS
 - TASK EXPLORER
 - JAVA PROJECTS
- Center Area:**
 - Create a class file
 - Please input a rule-compliant file name! (Press 'Enter' to confirm or 'Escape' to cancel)
 - Dockerfile > ...


```
1 FROM eclipse-temurin:17
2
3 # Maintainer or author
4 LABEL maintainer="ebotsmith@gmail.com"
5
6 # Set the working directory inside the container
7 WORKDIR /app
8
9 # Copy the application's JAR file into the container
10 COPY ./target/springboot-docker-demo-0.0.1-SNAPSHOT.jar ./app/springboot-docker-demo.jar
11
12 # Define the ENTRYPOINT for the container. This command will be executed when the container starts
13 # Arguments provided to 'docker run' will be appended to this ENTRYPOINT
14 ENTRYPOINT ["java", "-jar", "./app/springboot-docker-demo.jar"]
```
 - PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, AZURE, SPELL CHECKER, QUERY RESULTS, SONARQUBE, MYBATISX, JavaSE-23 tabs.
 - TERMINAL tab shows build logs:


```
[INFO]
[INFO] --- jar:3.4.2:jar (default-jar) @ springboot-docker-demo ---
[INFO] --- spring-boot:3.5.0:repackage (repackage) @ springboot-docker-demo ---
[INFO] Replacing main artifact C:\DevOps-Projects\Github_Action\springboot-docker-demo\target\springboot-docker-demo-0.0.1-SNAPSHOT.jar with repackaged archive, adding nested dependencies in BOOT-INF/
[INFO] The original artifact has been renamed to C:\DevOps-Projects\Github_Action\springboot-docker-demo\target\springboot-docker-demo-0.0.1-SNAPSHOT.jar.original
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 10.156 s
[INFO] Finished at: 2025-10-19T11:39:06-04:00
[INFO] -----
```
- Bottom Status Bar:** C:\DevOps-Projects\Github_Action\springboot-docker-demo, Live Share, SonarQube, AWS: profile=default, Indexing completed, Java: Ready, RHDA analysis has failed, SonarQube focus: overall code, Watch, Cloud Code - Sign in, Add Comments, Generate Commit M

Now, let us save the file.

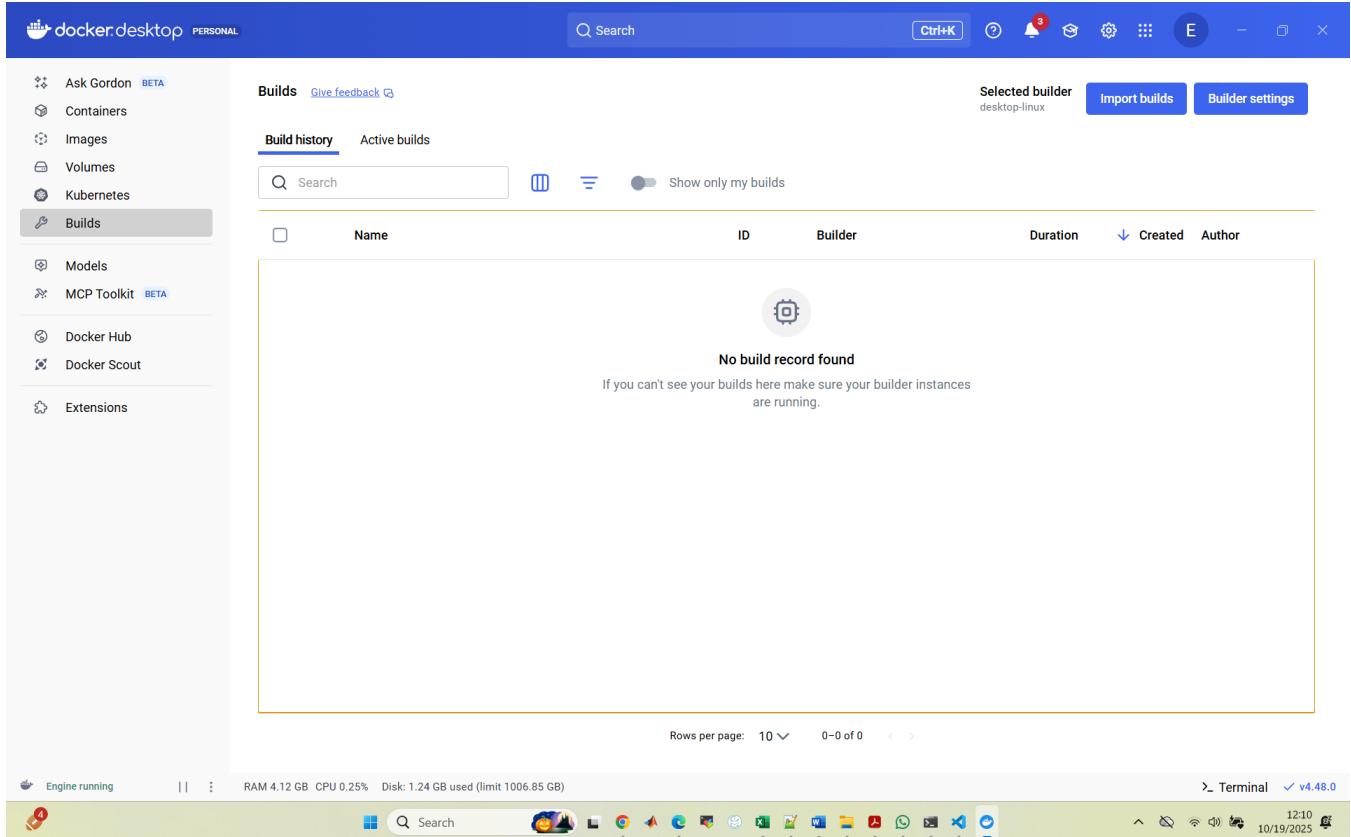
The screenshot shows the Eclipse IDE interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Left Sidebar (EXPLORER):**
 - OPEN EDITORS: Dockerfile
 - SPRINGBOOT-DOCKER-DEMO
 - src
 - target
 - classes
 - generated-sources
 - generated-test-sources
 - maven-archiver
 - maven-status
 - surefire-reports
 - test-classes
 - springboot-docker-demo-0.0.1-SNAPSHOT.jar
 - springboot-docker-demo-0.0.1-SNAPSHOT.jar.original
 - .gitattributes
 - .gitignore
 - Dockerfile
 - HELP.md
 - mvnw
 - mvnw.cmd
 - pom.xml
 - OUTLINE
 - TIMELINE
 - TOMCAT SERVERS
 - APPLICATION BUILDER
 - PROJECTS
 - RUN CONFIGURATION
 - TASK EXPLORER
 - JAVA PROJECTS
- Center Area:**
 - Dockerfile X
 - Dockerfile > ...

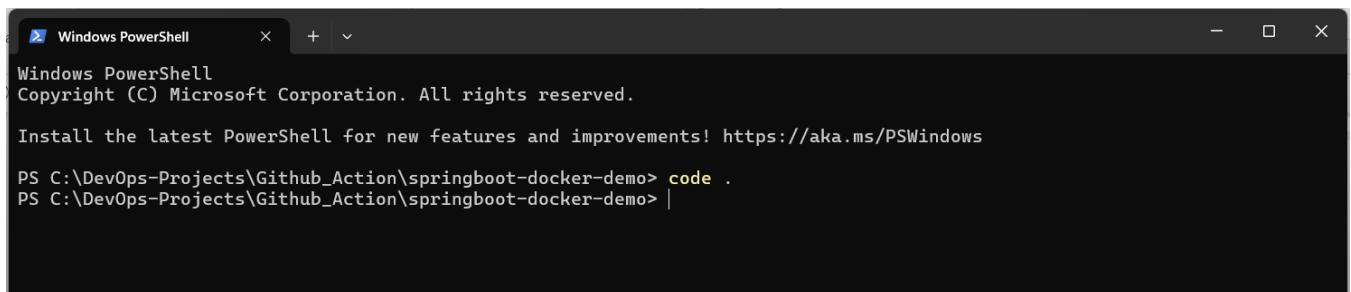

```
1 # Use an official Eclipse Temurin base image
2 FROM eclipse-temurin:17
3
4 LABEL maintainer="ebotsmith@gmail.com"
5
6 # Set the working directory inside the container
7 WORKDIR /app
8
9 # Copy the application's JAR file into the container
10 COPY ./target/springboot-docker-demo-0.0.1-SNAPSHOT.jar ./app/springboot-docker-demo.jar
11
12 # Define the ENTRYPOINT for the container
13 # This command will be executed when the container starts
14 # Arguments provided to 'docker run' will be appended to this ENTRYPOINT
15 ENTRYPOINT ["java", "-jar", "springboot-docker-demo.jar"]
```
 - CHAT, Build with agent mode, Let's get started, Build Workspace, Show Config buttons.
 - AI responses may be inaccurate.
 - A warning message: RHDA error while analyzing c:\DevOps-Projects\Github_Action\springboot-docker-demo...
- Bottom Status Bar:** Launchpad, No Solution, Live Share, SonarQube, AWS: profile=default, Indexing completed, Java: Ready, RHDA analysis has failed, SonarQube focus: overall code, Watch, Cloud Code - Sign in, Add Comments, Generate Commit M

STEP 5: Build Docker Image from Dockerfile

We will now build a Docker Image from the docker file. Before doing this, make sure your Docker desktop is up and running on your local machine.



Head back to the PowerShell terminal



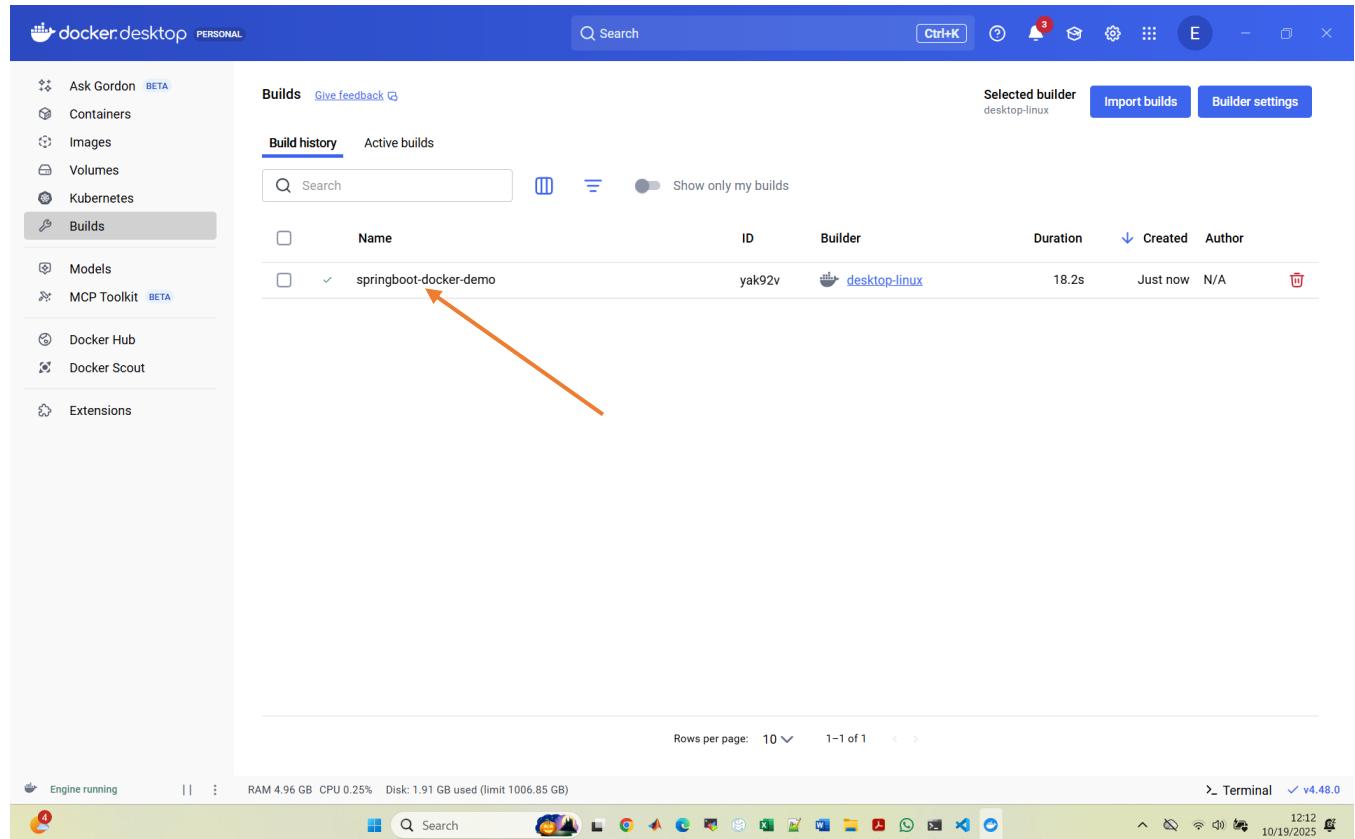
Run the command to build the Docker image with the name “**springboot-docker-demo**”

```
docker build -t springboot-docker-demo .
```

```
[+] Windows PowerShell x + v - □ ×
=> [auth] library/eclipse-temurin:pull token for registry-1.docker.io 0.0s
=> [internal] load .dockercfgignore 0.0s
=> => transferring context: 2B 0.0s
=> [1/3] FROM docker.io/library/eclipse-temurin:17@sha256:54a16ef91e00c11ac9b05029faa5efd02c37296844204a6561d24 14.0s
=> => resolve docker.io/library/eclipse-temurin:17@sha256:54a16ef91e00c11ac9b05029faa5efd02c37296844204a6561d244 0.1s
=> => sha256:006f3cd456642ee316bd7db920367b38f89371ld1b0b36b723dd5578de8524f 160B / 160B 0.1s
=> => sha256:9a480c17294ce36bce38ec711e88d04df2c95895b9404dd66cc4a789564ed3aa 2.28kB / 2.28kB 0.1s
=> => sha256:aeb110f3ed52df42805ce6b699e7a00c2f9b36f9418035551c7c2661f7e90e82 144.71MB / 144.71MB 11.7s
=> => sha256:6c1fc2dd06a475df08db25803458076f8149c63f5ac52acf5c3c00b0fc857c6 22.96MB / 22.96MB 4.5s
=> => sha256:4b3ffd8ccb5201a0fc03585952effb4ed2d1ea5e704d2e7330212fb8b16c86a3 29.72MB / 29.72MB 3.6s
=> => extracting sha256:4b3ffd8ccb5201a0fc03585952effb4ed2d1ea5e704d2e7330212fb8b16c86a3 1.3s
=> => extracting sha256:6c1fc2d06a475df08db25803458076f8149c63f5ac52acf5c3c00b0fc857c6 0.8s
=> => extracting sha256:aeb110f3ed52df42805ce6b699e7a00c2f9b36f9418035551c7c2661f7e90e82 2.0s
=> => extracting sha256:006f3cd456642ee316bd7db920367b38f89371ld1b0b36b723dd5578de8524f 0.0s
=> => extracting sha256:9a480c17294ce36bce38ec711e88d04df2c95895b9404dd66cc4a789564ed3aa 0.0s
=> [internal] load build context 1.9s
=> => transferring context: 21.00MB 1.9s
=> [2/3] WORKDIR /app 0.5s
=> [3/3] COPY /target/springboot-docker-demo-0.0.1-SNAPSHOT.jar /app/springboot-docker-demo.jar 0.1s
=> exporting to image 2.1s
=> => exporting layers 1.6s
=> => exporting manifest sha256:1935eb942ecaa2e49f84bb1f0a1cf13f46a9a4ae3c457220b7a68c7851fb1f6a 0.0s
=> => exporting config sha256:825655099518ba1f08ef6a6d61a8561acl7235104adcedf2c4db0b64ba79e4ada 0.0s
=> => exporting attestation manifest sha256:b6b2ed6150a89f7ab06134c2fa05015de5f3030b3861d8c39b04b59a3606bafe 0.1s
=> => exporting manifest list sha256:179031f3e0calc1331b1ad8db93c2b96a2731762752fd2fec7c3598cba9f021 0.0s
=> => naming to docker.io/library/springboot-docker-demo:latest 0.0s
=> => unpacking to docker.io/library/springboot-docker-demo:latest 0.3s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/yak92vgqjmlrr4k98gm4o7yl
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo |
```

Head back to the Docker desktop



You can see the image has been built. Let us verify using the command:

docker images

```

Windows PowerShell
+ - x
=> [1/3] FROM docker.io/library/eclipse-temurin:17@sha256:54a16ef91e00c11ac9b05029faa5efd02c37296844204a6561d24 14.0s
=> => resolve docker.io/library/eclipse-temurin:17@sha256:54a16ef91e00c11ac9b05029faa5efd02c37296844204a6561d244 0.1s
=> => sha256:006f3cd456642ee316bd7db920367b38f8b93371d1b0b36b723dd5578de8524f 160B / 160B 0.1s
=> => sha256:9a480c17294ce36bce38ec711e88d04df2c95895b9404dd66cc4a789564ed3aa 2.28kB / 2.28kB 0.1s
=> => sha256:aeb110f3ed52df42805ce6b699e7a00c2f9b36f9418035551c7c2661f7e90e82 144.71MB / 144.71MB 11.7s
=> => sha256:6c1fc2dd06a475df08db25803458076f8149c63f5ac5c2acf5c3c00b0fc857c6 22.96MB / 22.96MB 4.5s
=> => sha256:4b3ffd8ccb5201a0fc03585952effb4ed2d1ea5e704d2e7330212fb8b16c86a3 29.72MB / 29.72MB 3.6s
=> => extracting sha256:4b3ffd8ccb5201a0fc03585952effb4ed2d1ea5e704d2e7330212fb8b16c86a3 1.3s
=> => extracting sha256:6c1fc2dd06a475df08db25803458076f8149c63f5ac5c2acf5c3c00b0fc857c6 0.8s
=> => extracting sha256:aeb110f3ed52df42805ce6b699e7a00c2f9b36f9418035551c7c2661f7e90e82 2.0s
=> => extracting sha256:006f3cd456642ee316bd7db920367b38f8b93371d1b0b36b723dd5578de8524f 0.0s
=> => extracting sha256:9a480c17294ce36bce38ec711e88d04df2c95895b9404dd66cc4a789564ed3aa 0.0s
=> [internal] load build context 1.9s
=> => transferring context: 21.00MB 1.9s
=> [2/3] WORKDIR /app 0.5s
=> [3/3] COPY /target/springboot-docker-demo-0.0.1-SNAPSHOT.jar /app/springboot-docker-demo.jar 0.1s
=> exporting to image 2.1s
=> => exporting layers 1.6s
=> => exporting manifest sha256:1935eb942ecaa2e49f84bb1f0a1cf13f46a9a4ae3c457220b7a68c7851fb1f6a 0.0s
=> => exporting config sha256:82565509518ba1fc08ef6a6d61a8561ac7235104adcedf2c4db0b64ba79e4ada 0.0s
=> => exporting attestation manifest sha256:b6b2ed6150a89f7ab06134c2fa05015de5f3030b3861d8c39b04b59a3606baf 0.1s
=> => exporting manifest list sha256:179031f3e0ca1c331b1ad8db93c2b96a2731762752fd2fec7c3598cba93af021 0.0s
=> => naming to docker.io/library/springboot-docker-demo:latest 0.0s
=> => unpacking to docker.io/library/springboot-docker-demo:latest 0.0s
=> [internal] load build context 0.3s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/yak92vgqjmlrr4k98gm4o7vl
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
springboot-docker-demo latest 179031f3e0ca About a minute ago 673MB
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo>

```

You can see our image. We can also add a tag to the image. We can add a tag name “0.1.RELEASE” by using the command:

```
docker build -t springboot-docker-demo:0.1.RELEASE .
```

```

Windows PowerShell
+ - x
=> => unpacking to docker.io/library/springboot-docker-demo:latest 0.3s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/yak92vgqjmlrr4k98gm4o7vl
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
springboot-docker-demo latest 179031f3e0ca About a minute ago 673MB
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> docker build -t springboot-docker-demo:0.1.RELEASE .
[+] Building 0.7s (8/8) FINISHED docker:desktop-linux
=> [internal] load build definition from Dockerfile 0.1s
=> => transferring dockerfile: 565B 0.0s
=> [internal] load metadata for docker.io/library/eclipse-temurin:17 0.4s
=> [internal] load .dockerrcignore 0.0s
=> => transferring context: 2B 0.0s
=> [1/3] FROM docker.io/library/eclipse-temurin:17@sha256:54a16ef91e00c11ac9b05029faa5efd02c37296844204a6561d244 0.0s
=> => resolve docker.io/library/eclipse-temurin:17@sha256:54a16ef91e00c11ac9b05029faa5efd02c37296844204a6561d244 0.0s
=> [internal] load build context 0.0s
=> => transferring context: 98B 0.0s
=> CACHED [2/3] WORKDIR /app 0.0s
=> CACHED [3/3] COPY /target/springboot-docker-demo-0.0.1-SNAPSHOT.jar /app/springboot-docker-demo.jar 0.0s
=> exporting to image 0.1s
=> => exporting layers 0.0s
=> => exporting manifest sha256:1935eb942ecaa2e49f84bb1f0a1cf13f46a9a4ae3c457220b7a68c7851fb1f6a 0.0s
=> => exporting config sha256:82565509518ba1fc08ef6a6d61a8561ac7235104adcedf2c4db0b64ba79e4ada 0.0s
=> => exporting attestation manifest sha256:695d47b84bd784cea67048bc72b5da98fa5dedb03119d2803d745fdb315a81d1 0.0s
=> => exporting manifest list sha256:c8f68c908b49d55d8091a81178bc1605690aec03f1b9488453a8fcc6756fe9c4 0.0s
=> => naming to docker.io/library/springboot-docker-demo:0.1.RELEASE 0.0s
=> => unpacking to docker.io/library/springboot-docker-demo:0.1.RELEASE 0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/w6zmtwjxykymd875e5pt318nz
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo>

```

Let us check the list of docker images again using the command:

```
docker images
```

```
Windows PowerShell x + - x
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
springboot-docker-demo  latest   179031f3e0ca  About a minute ago  673MB
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> docker build -t springboot-docker-demo:0.1.RELEASE .
[+] Building 0.7s (8/8) FINISHED
          docker:desktop-linux
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 565B
=> [internal] load metadata for docker.io/library/eclipse-temurin:17
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/3] FROM docker.io/library/eclipse-temurin:17@sha256:54a16ef91e00c11ac9b05029faa5efd02c37296844204a6561d244
=> => resolve docker.io/library/eclipse-temurin:17@sha256:54a16ef91e00c11ac9b05029faa5efd02c37296844204a6561d244
=> [internal] load build context
=> => transferring context: 98B
=> CACHED [2/3] WORKDIR /app
=> CACHED [3/3] COPY /target/springboot-docker-demo-0.0.1-SNAPSHOT.jar /app/springboot-docker-demo.jar
=> exporting to image
=> => exporting layers
=> => exporting manifest sha256:1935eb942ecaa2e49f84bb1f0a1cf13f46a9a4ae3c457220b7a68c7851fb1f6a
=> => exporting config sha256:82565509518ba1c08ef6a6d61a8561ac7235104adcedf2c4db0b64ba79e4ada
=> => exporting attestation manifest sha256:695d47b84bd784cea67048bc72b5da98fa5dedb03119d2803d745fdb315a81d1
=> => exporting manifest list sha256:c8f68c908b49d55d8091a81178bc1605690aec03f1b9488453a8fcc6756fe9c4
=> => naming to docker.io/library/springboot-docker-demo:0.1.RELEASE
=> => unpacking to docker.io/library/springboot-docker-demo:0.1.RELEASE
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/w6zmtwjxykymd875e5pt318nz
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
springboot-docker-demo  0.1.RELEASE  c8f68c908b49  3 minutes ago  673MB
springboot-docker-demo  latest   179031f3e0ca  3 minutes ago  673MB
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> |
```

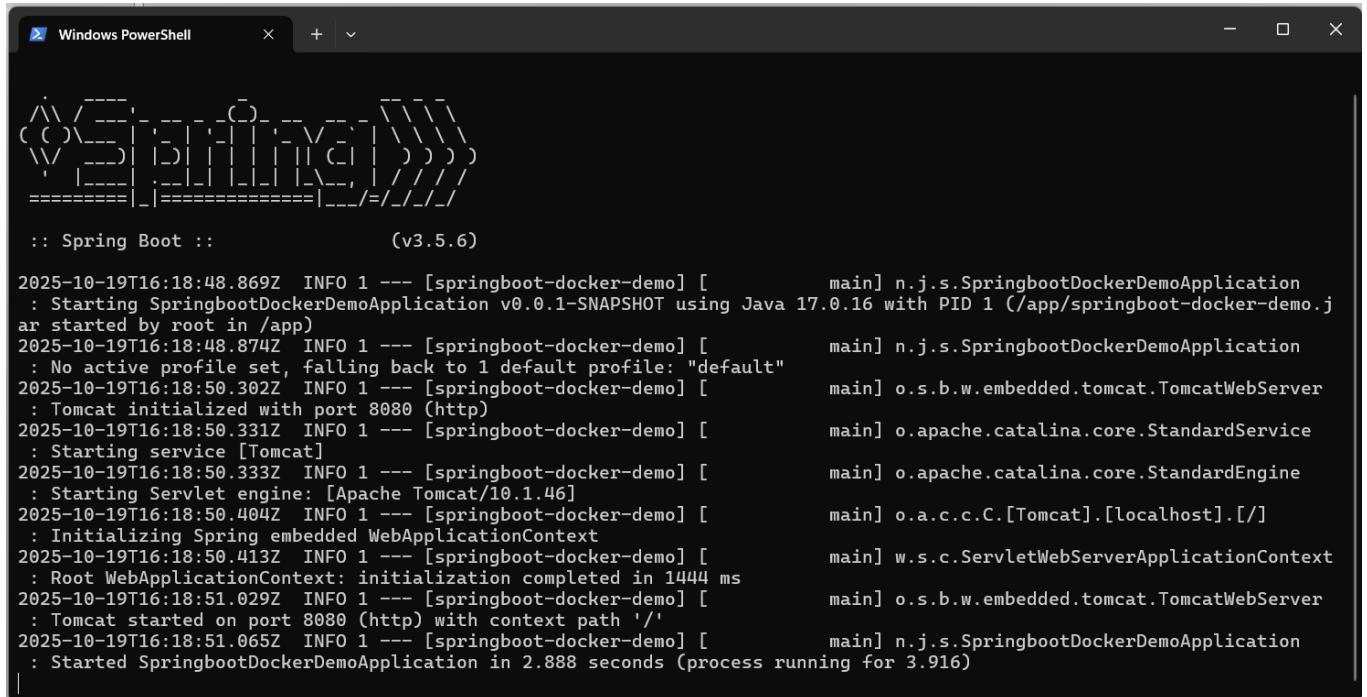
You can see the tag.

STEP 6: Run Docker Image in a Docker Container

We will run the Docker Image in a Docker container. This will be done using the command:

```
docker run -p 8080:8080 springboot-docker-demo
```

The **-p** is used to map the operating system port 8080 to the docker container port 8080. The image name is “**springboot-docker-demo**”

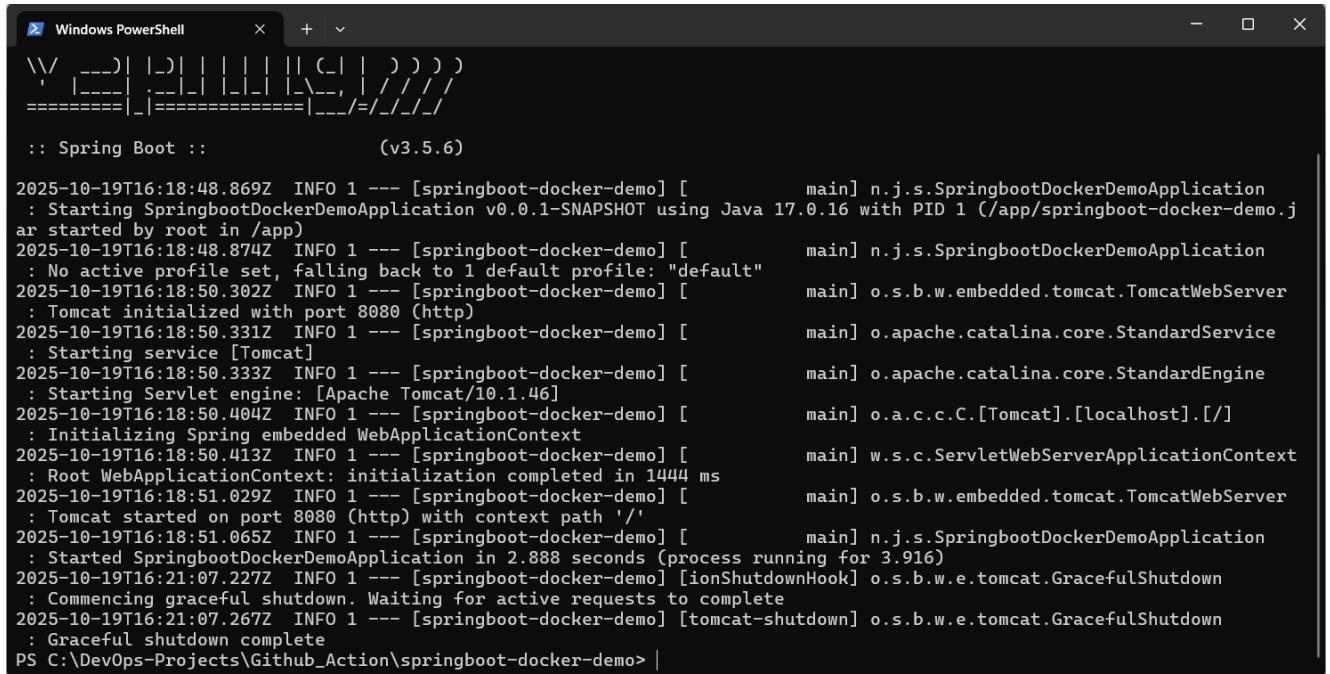


```
Windows PowerShell

:: Spring Boot :: (v3.5.6)

2025-10-19T16:18:48.869Z INFO 1 --- [springboot-docker-demo] [main] n.j.s.SpringbootDockerDemoApplication : Starting SpringbootDockerDemoApplication v0.0.1-SNAPSHOT using Java 17.0.16 with PID 1 (/app/springboot-docker-demo.jar started by root in /app)
2025-10-19T16:18:48.874Z INFO 1 --- [springboot-docker-demo] [main] n.j.s.SpringbootDockerDemoApplication : No active profile set, falling back to 1 default profile: "default"
2025-10-19T16:18:50.302Z INFO 1 --- [springboot-docker-demo] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8080 (http)
2025-10-19T16:18:50.331Z INFO 1 --- [springboot-docker-demo] [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2025-10-19T16:18:50.333Z INFO 1 --- [springboot-docker-demo] [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.46]
2025-10-19T16:18:50.404Z INFO 1 --- [springboot-docker-demo] [main] o.a.c.c.C.[Tomcat].[localhost].[] : Initializing Spring embedded WebApplicationContext
2025-10-19T16:18:50.413Z INFO 1 --- [springboot-docker-demo] [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1444 ms
2025-10-19T16:18:51.029Z INFO 1 --- [springboot-docker-demo] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path '/'
2025-10-19T16:18:51.065Z INFO 1 --- [springboot-docker-demo] [main] n.j.s.SpringbootDockerDemoApplication : Started SpringbootDockerDemoApplication in 2.888 seconds (process running for 3.916)
```

Switch to the command mode using “CTRL + c”



```
Windows PowerShell

:: Spring Boot :: (v3.5.6)

2025-10-19T16:18:48.869Z INFO 1 --- [springboot-docker-demo] [main] n.j.s.SpringbootDockerDemoApplication : Starting SpringbootDockerDemoApplication v0.0.1-SNAPSHOT using Java 17.0.16 with PID 1 (/app/springboot-docker-demo.jar started by root in /app)
2025-10-19T16:18:48.874Z INFO 1 --- [springboot-docker-demo] [main] n.j.s.SpringbootDockerDemoApplication : No active profile set, falling back to 1 default profile: "default"
2025-10-19T16:18:50.302Z INFO 1 --- [springboot-docker-demo] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8080 (http)
2025-10-19T16:18:50.331Z INFO 1 --- [springboot-docker-demo] [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2025-10-19T16:18:50.333Z INFO 1 --- [springboot-docker-demo] [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.46]
2025-10-19T16:18:50.404Z INFO 1 --- [springboot-docker-demo] [main] o.a.c.c.C.[Tomcat].[localhost].[] : Initializing Spring embedded WebApplicationContext
2025-10-19T16:18:50.413Z INFO 1 --- [springboot-docker-demo] [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1444 ms
2025-10-19T16:18:51.029Z INFO 1 --- [springboot-docker-demo] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path '/'
2025-10-19T16:18:51.065Z INFO 1 --- [springboot-docker-demo] [main] n.j.s.SpringbootDockerDemoApplication : Started SpringbootDockerDemoApplication in 2.888 seconds (process running for 3.916)
2025-10-19T16:21:07.227Z INFO 1 --- [springboot-docker-demo] [ionShutdownHook] o.s.b.w.e.tomcatGracefulShutdown : Commencing graceful shutdown. Waiting for active requests to complete
2025-10-19T16:21:07.267Z INFO 1 --- [springboot-docker-demo] [tomcatShutdown] o.s.b.w.e.tomcatGracefulShutdown : Graceful shutdown complete
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo>
```

STEP 7: Run the Docker container in Detached mode

Running the docker container in detached mode means running it in the background. To do this we run the command:

```
docker run -p 8081:8080 -d springboot-docker-demo
```

```
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> docker run -p 8081:8080 -d springboot-docker-demo  
a7aa@0c07d2b6dbd8c2cd8e33196cc8e9f219711285fb89d44fb801140f5a244c  
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> |
```

You can see the container's ID

We can now check the logs of the docker container using the command:

```
docker logs -f a7aa
```

a7aa is the first four digits of the docker container's ID

```
Windows PowerShell x + - < > x

:: Spring Boot ::          (v3.5.6)

2025-10-19T16:22:04.653Z INFO 1 --- [springboot-docker-demo] [main] n.j.s.SpringbootDockerDemoApplication
: Starting SpringbootDockerDemoApplication v0.0.1-SNAPSHOT using Java 17.0.16 with PID 1 (/app/springboot-docker-demo.jar started by root in /app)
2025-10-19T16:22:04.659Z INFO 0 1 --- [springboot-docker-demo] [main] n.j.s.SpringbootDockerDemoApplication
: No active profile set, falling back to 1 default profile: "default"
2025-10-19T16:22:05.870Z INFO 0 1 --- [springboot-docker-demo] [main] o.s.b.w.embedded.tomcat.TomcatWebServer
: Tomcat initialized with port 8080 (http)
2025-10-19T16:22:05.898Z INFO 0 1 --- [springboot-docker-demo] [main] o.apache.catalina.core.StandardService
: Starting service [Tomcat]
2025-10-19T16:22:05.899Z INFO 0 1 --- [springboot-docker-demo] [main] o.apache.catalina.core.StandardEngine
: Starting Servlet engine: [Apache Tomcat/10.1.46]
2025-10-19T16:22:05.998Z INFO 0 1 --- [springboot-docker-demo] [main] o.a.c.c.C.[Tomcat].[localhost].[/]
: Initializing Spring embedded WebApplicationContext
2025-10-19T16:22:06.002Z INFO 0 1 --- [springboot-docker-demo] [main] w.s.c.ServletWebServerApplicationContext
: Root WebApplicationContext: initialization completed in 1268 ms
2025-10-19T16:22:06.646Z INFO 0 1 --- [springboot-docker-demo] [main] o.s.b.w.embedded.tomcat.TomcatWebServer
: Tomcat started on port 8080 (http) with context path '/'
2025-10-19T16:22:06.662Z INFO 0 1 --- [springboot-docker-demo] [main] n.j.s.SpringbootDockerDemoApplication
: Started SpringbootDockerDemoApplication in 2.603 seconds (process running for 3.608)
```

This is the log of the container. If you use “**ctrl + c**” now, you will exit to command mode but the container will still be running in the background

```
Windows PowerShell x + - □ ×

:: Spring Boot ::          (v3.5.6)

2025-10-19T16:22:04.653Z INFO 1 --- [springboot-docker-demo] [main] n.j.s.SpringbootDockerDemoApplication
: Starting SpringbootDockerDemoApplication v0.0.1-SNAPSHOT using Java 17.0.16 with PID 1 (/app/springboot-docker-demo.j
ar started by root in /app)
2025-10-19T16:22:04.659Z INFO 1 --- [springboot-docker-demo] [main] n.j.s.SpringbootDockerDemoApplication
: No active profile set, falling back to 1 default profile: "default"
2025-10-19T16:22:05.870Z INFO 1 --- [springboot-docker-demo] [main] o.s.b.w.embedded.tomcat.TomcatWebServer
: Tomcat initialized with port 8080 (http)
2025-10-19T16:22:05.898Z INFO 1 --- [springboot-docker-demo] [main] o.apache.catalina.core.StandardService
: Starting service [Tomcat]
2025-10-19T16:22:05.899Z INFO 1 --- [springboot-docker-demo] [main] o.apache.catalina.core.StandardEngine
: Starting Servlet engine: [Apache Tomcat/10.1.46]
2025-10-19T16:22:05.998Z INFO 1 --- [springboot-docker-demo] [main] o.a.c.c.C.[Tomcat].[localhost].[/]
: Initializing Spring embedded WebApplicationContext
2025-10-19T16:22:06.002Z INFO 1 --- [springboot-docker-demo] [main] w.s.c.ServletWebServerApplicationContext
: Root WebApplicationContext: initialization completed in 1268 ms
2025-10-19T16:22:06.646Z INFO 1 --- [springboot-docker-demo] [main] o.s.b.w.embedded.tomcat.TomcatWebServer
: Tomcat started on port 8080 (http) with context path '/'
2025-10-19T16:22:06.662Z INFO 1 --- [springboot-docker-demo] [main] n.j.s.SpringbootDockerDemoApplication
: Started SpringbootDockerDemoApplication in 2.603 seconds (process running for 3.608)
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo>
```

Verify using the command:

```
docker ps
```

```
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
NAMES
a7aa0c07d2b6        springboot-docker-demo   "java -jar /app/springboot-docker-demo.jar"   20 minutes ago    Up 20 minutes   0.0.0.0:8081->8080/tcp
, [::]:8081->8080/tcp   sad_ramanujan
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> |
```

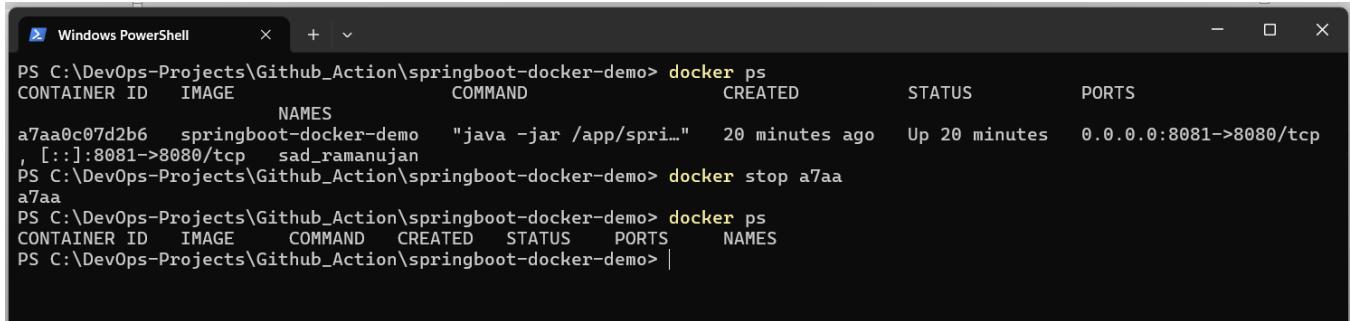
You can see that the container is still running in the background. To stop the container from running completely, you use the command:

```
docker stop a7aa
```

```
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
NAMES
a7aa0c07d2b6        springboot-docker-demo   "java -jar /app/spri..."   20 minutes ago    Up 20 minutes    0.0.0.0:8081->8080/tcp
, [::]:8081->8080/tcp      sad_ramanujan
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> docker stop a7aa
a7aa
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> |
```

If you run the command to check the containers now, that is

```
docker ps
```



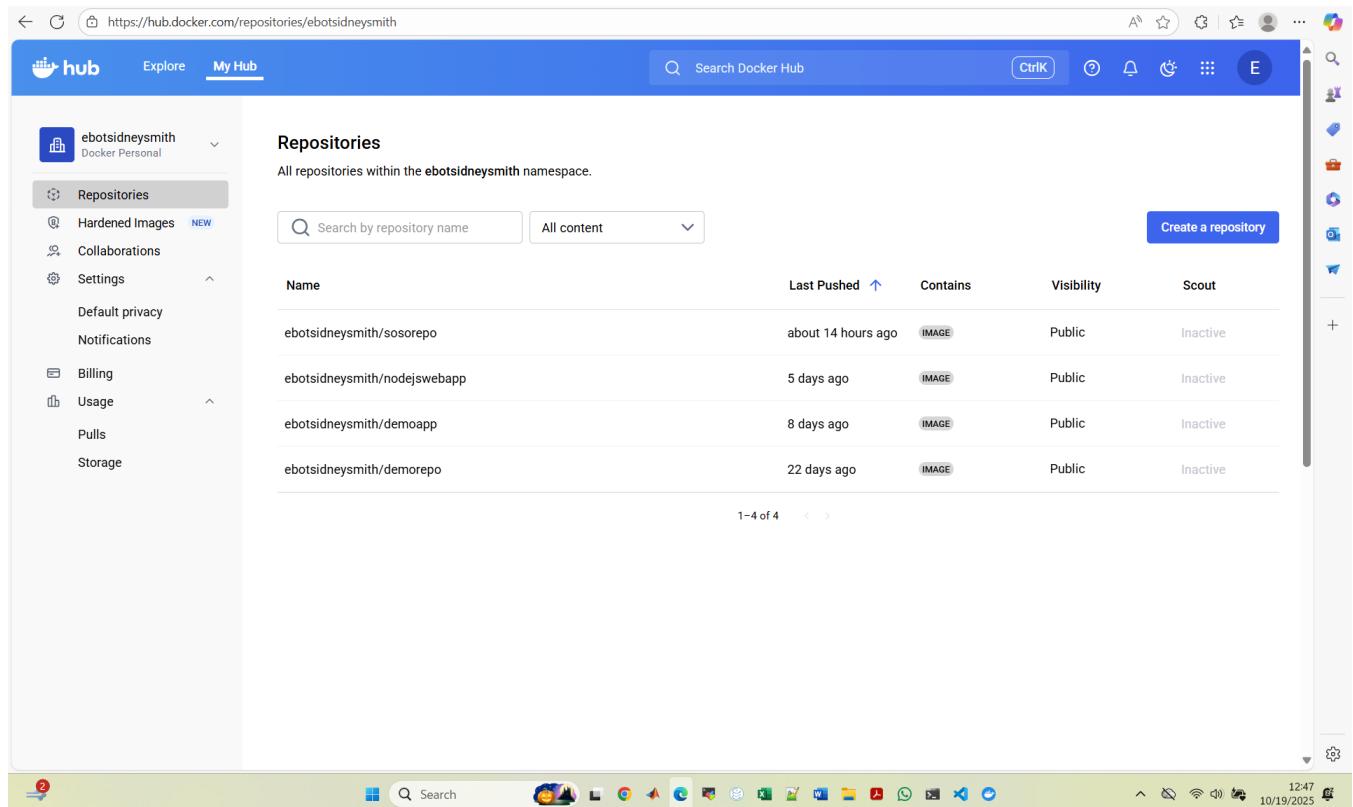
A screenshot of a Windows PowerShell window titled "Windows PowerShell". The command "docker ps" is run, showing one container named "sad_ramanujan" with ID "a7aa0c07d2b6". The container runs a Java application from the "springboot-docker-demo" image, mapping port 8080 to 8080. The container was created 20 minutes ago and is currently up. The command "docker stop a7aa" is then run to stop the container. Finally, "docker ps" is run again, showing no containers are running.

```
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> docker ps
CONTAINER ID   IMAGE       COMMAND                  CREATED        STATUS        PORTS
a7aa0c07d2b6   springboot-docker-demo   "java -jar /app/spri..."   20 minutes ago   Up 20 minutes   0.0.0.0:8081->8080/tcp
, [::]:8081->8080/tcp   sad_ramanujan
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> docker stop a7aa
a7aa
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> docker ps
CONTAINER ID   IMAGE       CREATED      STATUS    PORTS      NAMES
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> |
```

You can see no container is running.

STEP 8: Push Docker Image from local Machine to Docker Hub

Finally, we will push the Docker image to the Docker hub. To do this you have to first create an account in Docker hub. I already have an account in Docker hub. So, I will skip that step.



The screenshot shows the Docker Hub 'My Hub' interface. On the left, there's a sidebar with options like 'ebotsidneysmith Docker Personal', 'Repositories' (selected), 'Hardened Images', 'Collaborations', 'Settings', 'Default privacy', 'Notifications', 'Billing', 'Usage', 'Pulls', and 'Storage'. The main area is titled 'Repositories' with the sub-instruction 'All repositories within the ebotsidneysmith namespace.' Below this is a search bar and a dropdown menu. A large blue button on the right says 'Create a repository'. The main table lists four repositories:

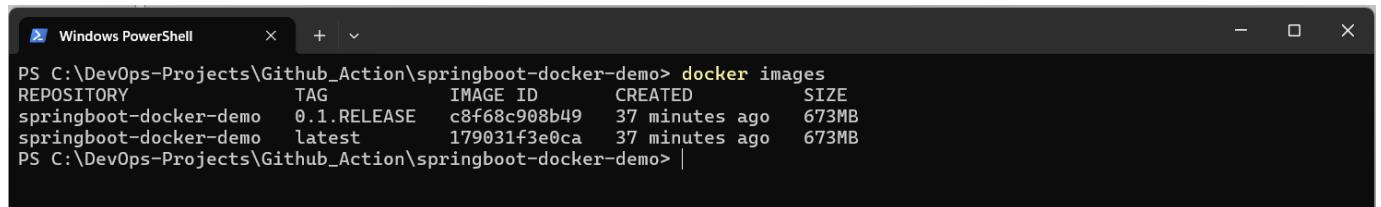
Name	Last Pushed	Contains	Visibility	Scout
ebotsidneysmith/sosorepo	about 14 hours ago	IMAGE	Public	Inactive
ebotsidneysmith/nodejswebapp	5 days ago	IMAGE	Public	Inactive
ebotsidneysmith/demoapp	8 days ago	IMAGE	Public	Inactive
ebotsidneysmith/demorepo	22 days ago	IMAGE	Public	Inactive

At the bottom, it says '1-4 of 4'.

Whenever you push a docker image to a docker hub, a new repository is created in your Docker hub to hold the image.

Let us check the docker images using the command:

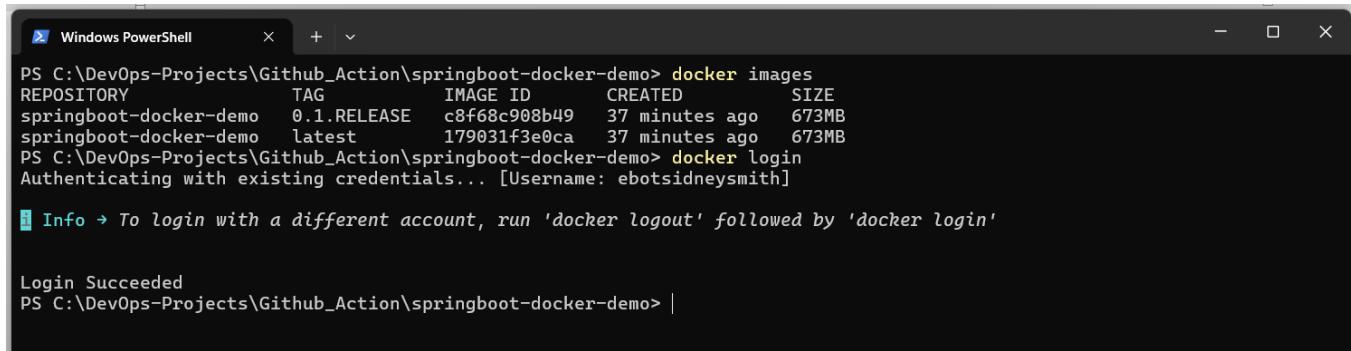
```
docker images
```



```
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> docker images
REPOSITORY          TAG      IMAGE ID   CREATED        SIZE
springboot-docker-demo  0.1.RELEASE  c8f68c908b49  37 minutes ago  673MB
springboot-docker-demo  latest    179031f3e0ca  37 minutes ago  673MB
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo>
```

We have one docker image. Now, let us push this image to the docker hub. To do this, we have to login to docker in our terminal. We can do this using the command:

```
docker login
```



```
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> docker images
REPOSITORY          TAG      IMAGE ID   CREATED    SIZE
springboot-docker-demo  0.1.RELEASE  c8f68c908b49  37 minutes ago  673MB
springboot-docker-demo  latest     179031f3e0ca  37 minutes ago  673MB
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> docker login
Authenticating with existing credentials... [Username: ebotsidneysmith]

Info → To login with a different account, run 'docker logout' followed by 'docker login'

Login Succeeded
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> |
```

I have logged into docker. Now, let us first associate our docker image with the Docker hub using the command:

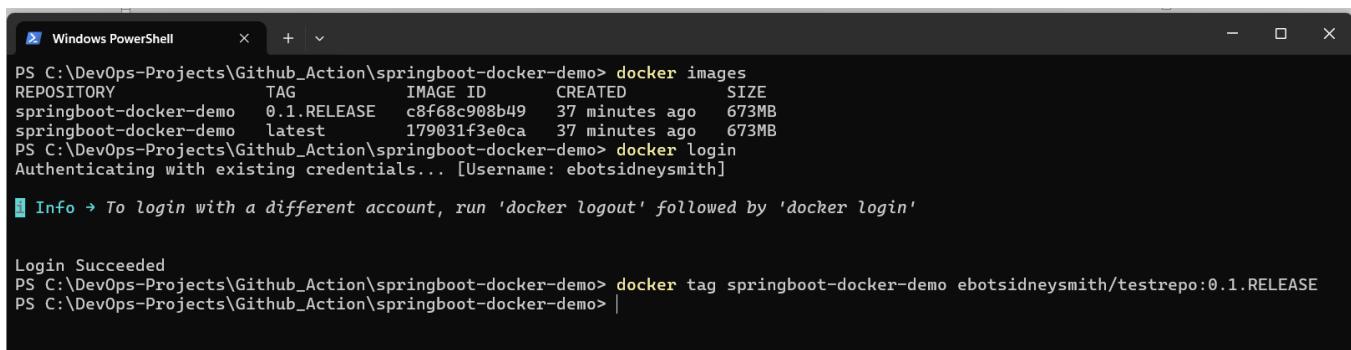
docker tag springboot-docker-demo ebotsidneysmith/testrepo:0.1.RELEASE

springboot-docker-demo is the image name

ebotsidneysmith is my docker account ID

testrepo is the repository to be created in Docker hub to hold the image

0.1.RELEASE is our image tag



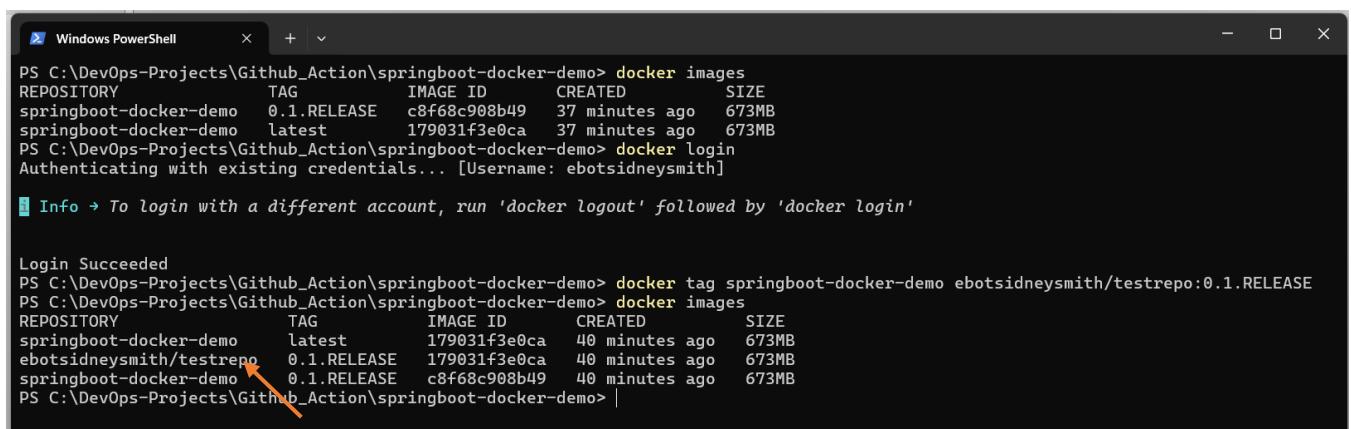
```
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> docker images
REPOSITORY          TAG      IMAGE ID   CREATED    SIZE
springboot-docker-demo  0.1.RELEASE  c8f68c908b49  37 minutes ago  673MB
springboot-docker-demo  latest     179031f3e0ca  37 minutes ago  673MB
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> docker login
Authenticating with existing credentials... [Username: ebotsidneysmith]

Info → To login with a different account, run 'docker logout' followed by 'docker login'

Login Succeeded
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> docker tag springboot-docker-demo ebotsidneysmith/testrepo:0.1.RELEASE
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> |
```

Run the command:

docker images



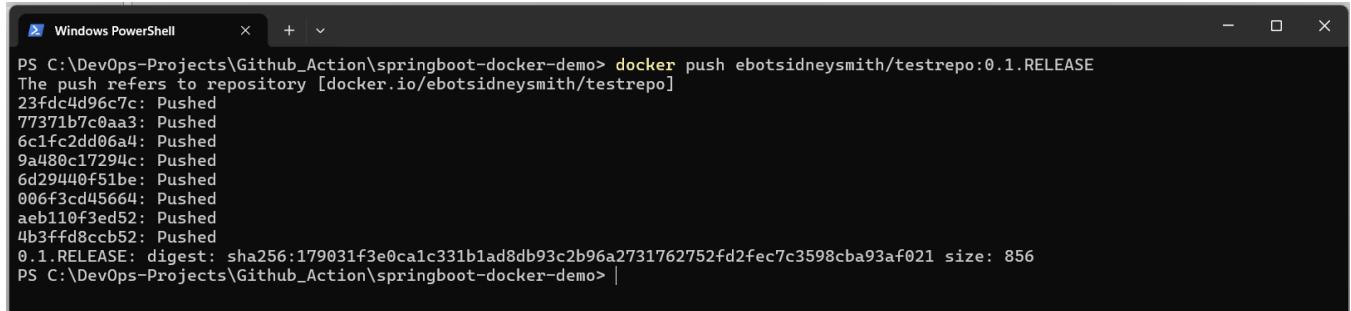
```
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> docker images
REPOSITORY          TAG      IMAGE ID   CREATED    SIZE
springboot-docker-demo  0.1.RELEASE  c8f68c908b49  37 minutes ago  673MB
springboot-docker-demo  latest     179031f3e0ca  37 minutes ago  673MB
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> docker login
Authenticating with existing credentials... [Username: ebotsidneysmith]

Info → To login with a different account, run 'docker logout' followed by 'docker login'

Login Succeeded
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> docker tag springboot-docker-demo ebotsidneysmith/testrepo:0.1.RELEASE
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> docker images
REPOSITORY          TAG      IMAGE ID   CREATED    SIZE
springboot-docker-demo  latest     179031f3e0ca  40 minutes ago  673MB
ebotsidneysmith/testrepo  0.1.RELEASE  179031f3e0ca  40 minutes ago  673MB
springboot-docker-demo  0.1.RELEASE  c8f68c908b49  40 minutes ago  673MB
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> |
```

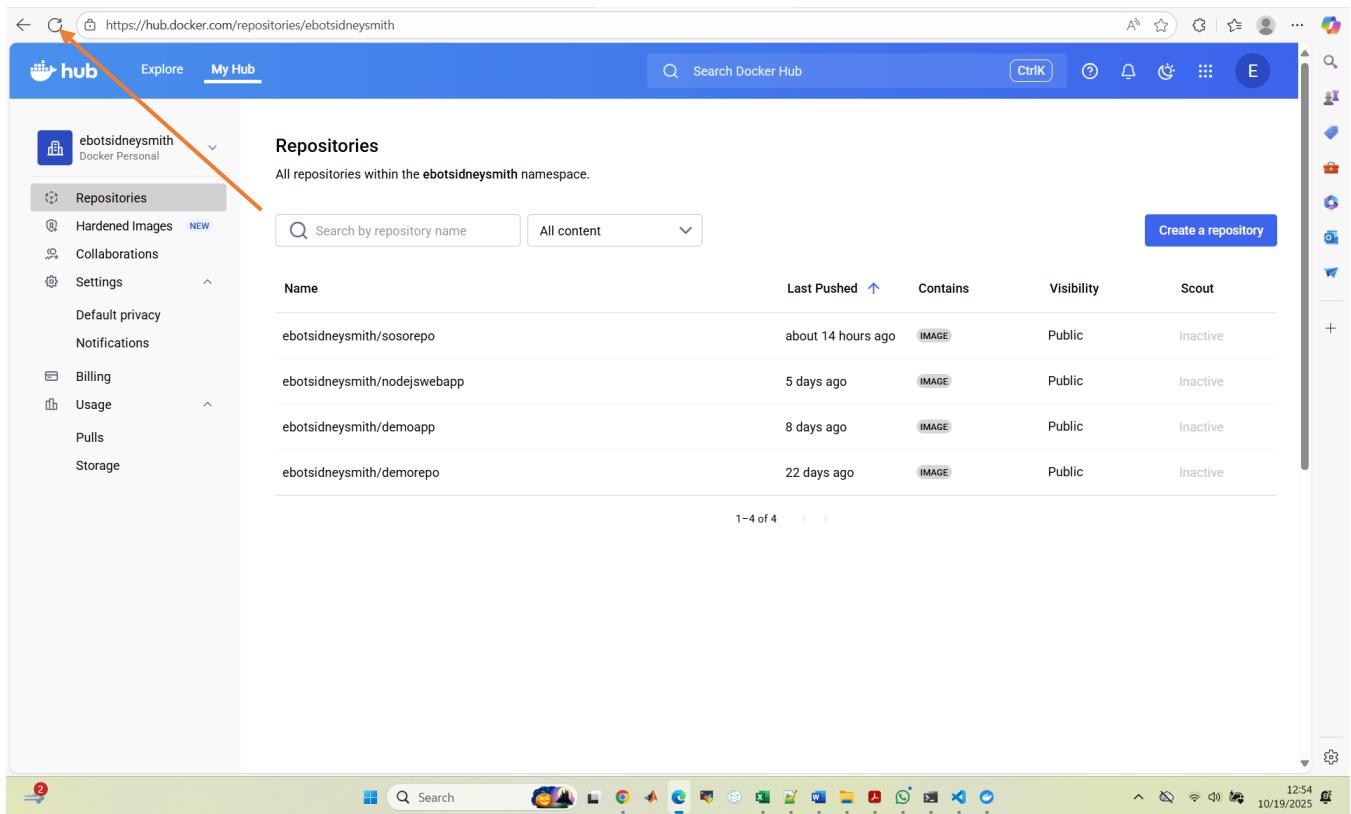
You can see that the repository ebotsidneysmith/testrepo. Then push the image to the docker hub using the command:

```
docker push ebotsidneysmith/testrepo:0.1.RELEASE
```



```
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> docker push ebotsidneysmith/testrepo:0.1.RELEASE
The push refers to repository [docker.io/ebotsidneysmith/testrepo]
23fdc4d96c7c: Pushed
77371b7c0aa3: Pushed
6c1fc2dd06a4: Pushed
9a480c17294c: Pushed
6d29440f51be: Pushed
006f3cd45664: Pushed
aeb110f3ed52: Pushed
4b3ffd8ccb52: Pushed
0.1.RELEASE: digest: sha256:179031f3e0calc331b1ad8db93c2b96a2731762752fd2fec7c3598cba93af021 size: 856
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> |
```

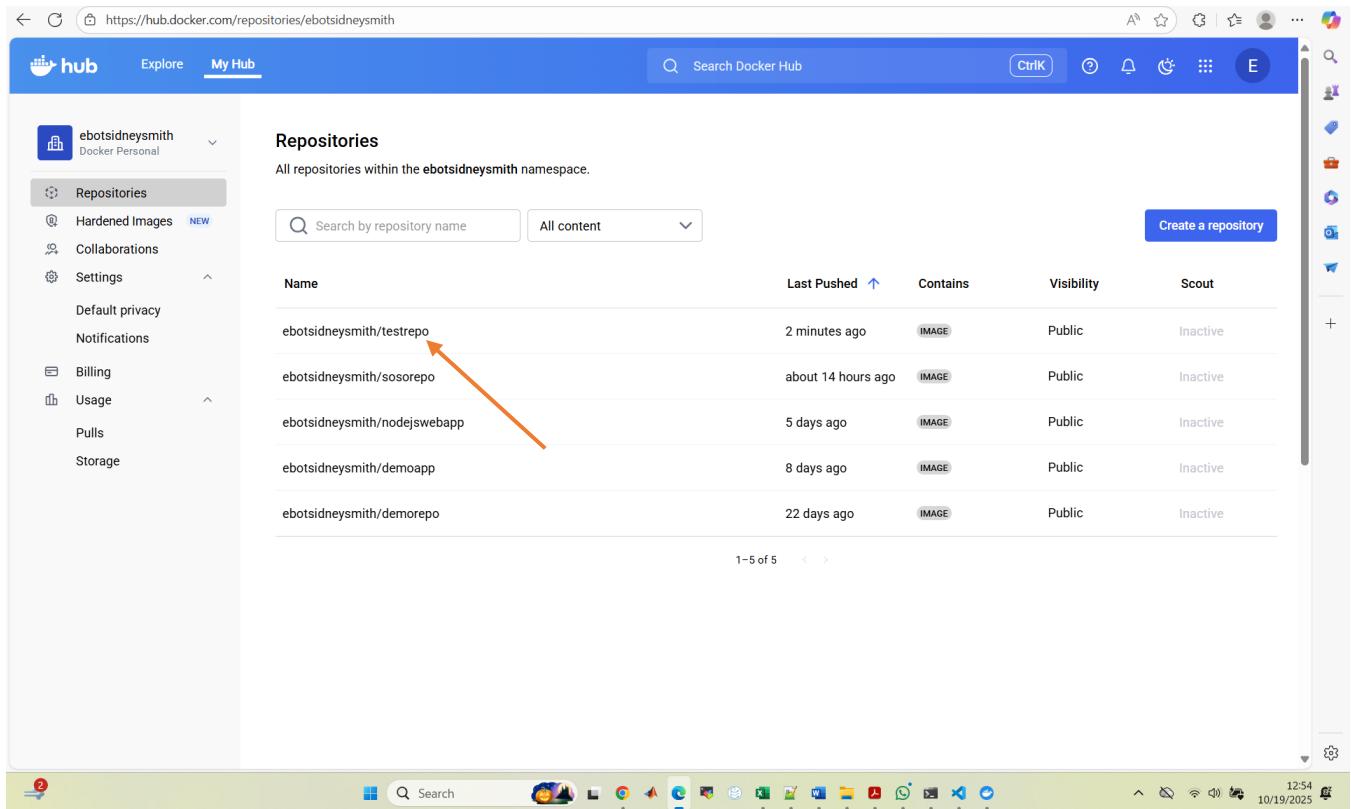
Let us head to our Docker hub now



The screenshot shows the Docker Hub interface for the user 'ebotsidneysmith'. The left sidebar includes sections for 'Repositories', 'Hardened Images', 'Collaborations', 'Settings', 'Billing', 'Usage', and 'Storage'. The main content area is titled 'Repositories' and lists four repositories under the heading 'All repositories within the ebotsidneysmith namespace.' The repositories are:

Name	Last Pushed	Contains	Visibility	Scout
ebotsidneysmith/sosorepo	about 14 hours ago	IMAGE	Public	Inactive
ebotsidneysmith/nodejswebapp	5 days ago	IMAGE	Public	Inactive
ebotsidneysmith/demoapp	8 days ago	IMAGE	Public	Inactive
ebotsidneysmith/demorepo	22 days ago	IMAGE	Public	Inactive

Refresh the page



The screenshot shows a web browser displaying the Docker Hub interface. The URL in the address bar is <https://hub.docker.com/repositories/ebotsidneysmith>. The page title is "My Hub". On the left, there's a sidebar with options like "Repositories", "Hardened Images", "Collaborations", "Settings", "Default privacy", "Notifications", "Billing", "Usage", "Pulls", and "Storage". The "Repositories" tab is selected. The main content area is titled "Repositories" and shows a list of repositories under the namespace "ebotsidneysmith". The list includes:

Name	Last Pushed	Contains	Visibility	Scout
ebotsidneysmith/testrepo	2 minutes ago	IMAGE	Public	Inactive
ebotsidneysmith/sosorepo	about 14 hours ago	IMAGE	Public	Inactive
ebotsidneysmith/nodejswebapp	5 days ago	IMAGE	Public	Inactive
ebotsidneysmith/demoapp	8 days ago	IMAGE	Public	Inactive
ebotsidneysmith/demorepo	22 days ago	IMAGE	Public	Inactive

An orange arrow points from the text "You can see our repository now." below to the "testrepo" entry in the list.

You can see our repository now.

STEP 9: Pull an Image from a Docker Hub

We can also pull an image from a docker hub and deploy it on different environments. This will be demonstrated here.

The screenshot shows the Docker Hub interface with the URL <https://hub.docker.com/repositories/ebotsidneysmith>. On the left, there's a sidebar with options like Repositories, Hardened Images, Collaborations, Settings, Billing, Usage, Pulls, and Storage. The main area is titled 'Repositories' and shows a list of repositories within the 'ebotsidneysmith' namespace. The list includes:

Name	Last Pushed	Contains	Visibility	Scout
ebotsidneysmith/testrepo	2 minutes ago	IMAGE	Public	Inactive
ebotsidneysmith/sosorepo	about 14 hours ago	IMAGE	Public	Inactive
ebotsidneysmith/nodejswebapp	5 days ago	IMAGE	Public	Inactive
ebotsidneysmith/demoapp	8 days ago	IMAGE	Public	Inactive
ebotsidneysmith/demorepo	22 days ago	IMAGE	Public	Inactive

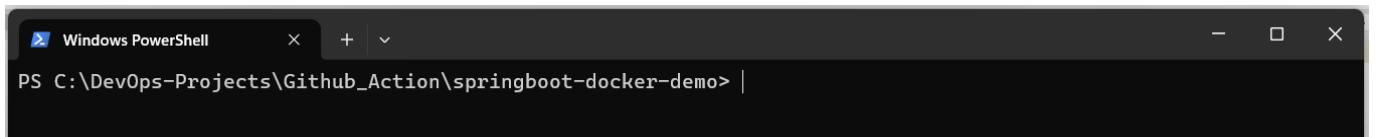
An orange arrow points to the 'testrepo' entry in the list.

Click on the docker repository

The screenshot shows the Docker Hub repository details page for 'ebotsidneysmith/testrepo'. The URL is <https://hub.docker.com/repository/docker/ebotsidneysmith/testrepo/general>. The page has a breadcrumb navigation bar: 'Repositories / testrepo / General'. An orange arrow points to this breadcrumb bar. The main content area shows the repository information: 'Last pushed 3 minutes ago • Repository size: 206.3 MB'. There are sections for 'Add a description' and 'Add a category'. Below these are tabs for 'General', 'Tags', 'Image Management (BETA)', 'Collaborators', 'Webhooks', and 'Settings'. The 'General' tab is selected. It shows a 'Tags' section with one tag: '0.1.RELEASE'. To the right, there's a 'Docker commands' section with the command 'docker push ebotsidneysmith/testrepo:tagname' and a 'Public view' button. A 'buildcloud' advertisement is visible on the right side.

Copy the docker ID and the repository: **ebotsidneysmith/testrepo**

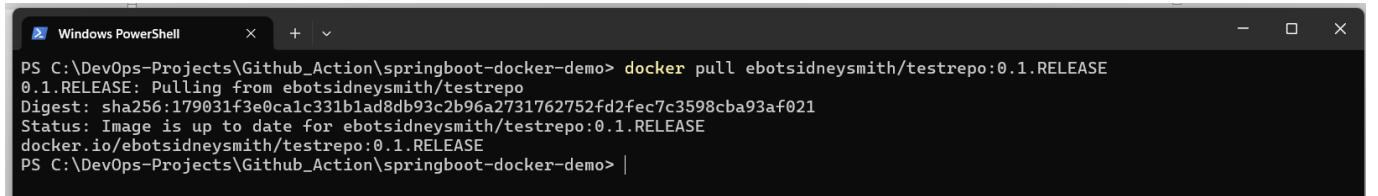
Head back to our terminal



```
Windows PowerShell
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> |
```

Then run the command:

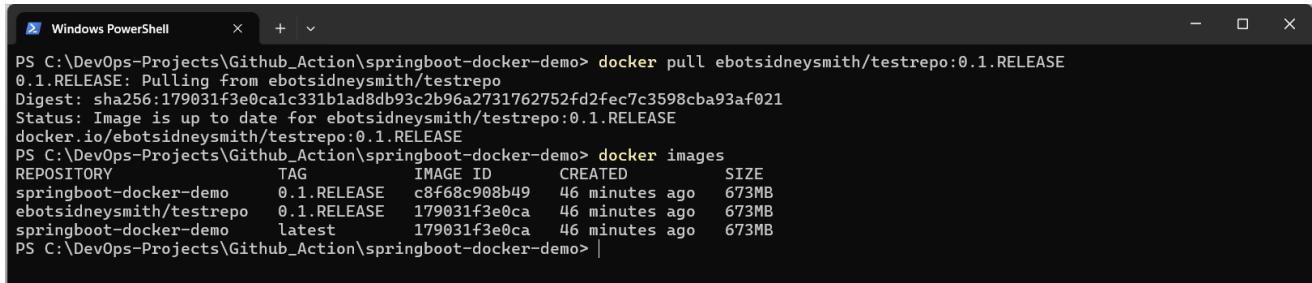
```
docker pull ebotsidneymsmith/testrepo:0.1.RELEASE
```



```
Windows PowerShell
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> docker pull ebotsidneymsmith/testrepo:0.1.RELEASE
0.1.RELEASE: Pulling from ebotsidneymsmith/testrepo
Digest: sha256:179031f3e0ca1c331b1ad8db93c2b96a2731762752fd2fec7c3598cba93af021
Status: Image is up to date for ebotsidneymsmith/testrepo:0.1.RELEASE
docker.io/ebotsidneymsmith/testrepo:0.1.RELEASE
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> |
```

We have successfully pulled the docker image from the docker hub. Let us check using the command:

```
docker images
```



```
Windows PowerShell
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> docker pull ebotsidneymsmith/testrepo:0.1.RELEASE
0.1.RELEASE: Pulling from ebotsidneymsmith/testrepo
Digest: sha256:179031f3e0ca1c331b1ad8db93c2b96a2731762752fd2fec7c3598cba93af021
Status: Image is up to date for ebotsidneymsmith/testrepo:0.1.RELEASE
docker.io/ebotsidneymsmith/testrepo:0.1.RELEASE
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> docker images
REPOSITORY          TAG           IMAGE ID    CREATED       SIZE
springboot-docker-demo  0.1.RELEASE   c8f68c908b49  46 minutes ago  673MB
ebotsidneymsmith/testrepo  0.1.RELEASE   179031f3e0ca  46 minutes ago  673MB
springboot-docker-demo  latest        179031f3e0ca  46 minutes ago  673MB
PS C:\DevOps-Projects\Github_Action\springboot-docker-demo> |
```