Phase-5

CI/CD Deployment for Springboot Application

DESCRIPTION

Project Objective: As a Full Stack Developer, you have to build a CI/CD pipeline to demonstrate continuous deployment and host the application on AWS EC2 instance.

Background of the problem statement:

As the project is in the final stage, management has asked you to automate the integration and deployment of the web application. You are required to set up an environment where the application will be hosted and accessed by users. The source code is supposed to be fetched from a GitHub repository.

You must use the following:

- Eclipse
- GitHub
- Jenkins
- AWS EC2/ Virtual machine

Following requirements should be met:

- A part of the source code should be tracked on the GitHub repository. You need to document the tracked files that are ignored during the final push to the GitHub repository.
- The submission of your GitHub repository link is mandatory. In order to track your task, you need to share the link of the repository in the document.
- The step-by-step process involved in completing this task should be documented.

Step 1) Create a Spring Boot Java App using Spring Initializr

Step 2) Create a GitHub repository

Step 3) Use Travis CI and Docker to implement CI/CD

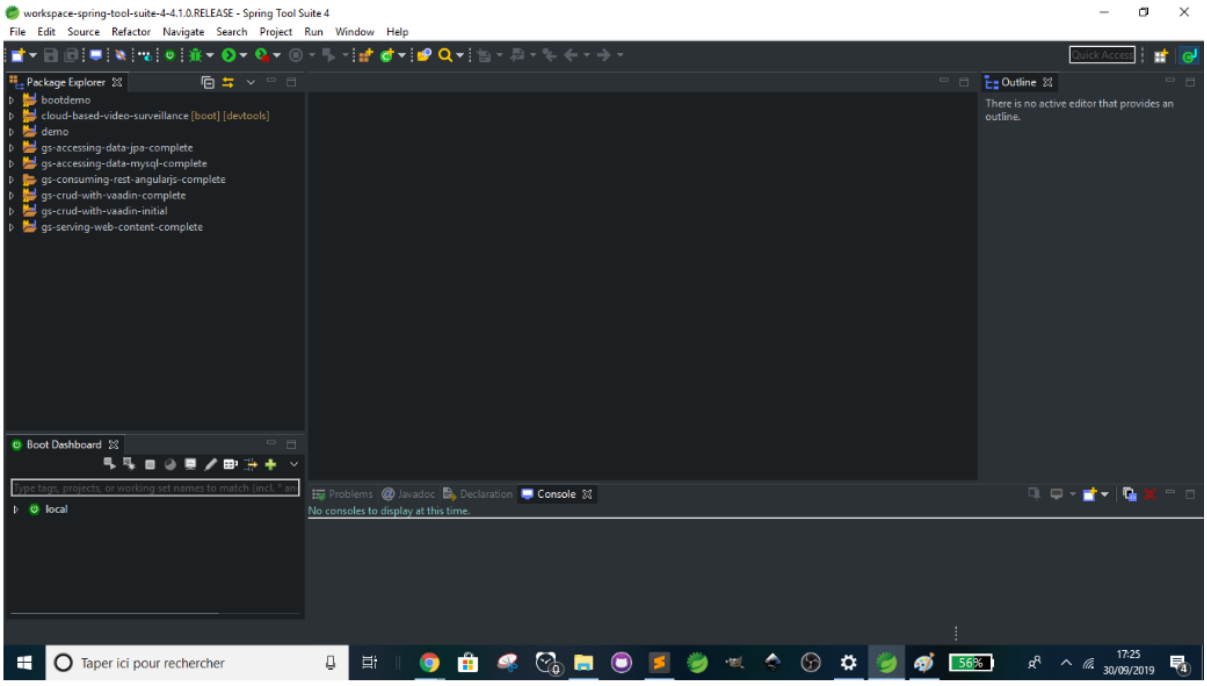
Step 4) Add Codecov to provide code coverage

Step 5) Use SonarCloud to write stellar code

Step 6) Build a project site using GitHub site-maven-plugin

Step 7) Deploy the app on Heroku using heroku-maven-plugin

1. Spring Tool Suite 4 (STS 4) IDE; you are free to use whatever tool you find suitable for this project.



Name: **cicd-applied-to-spring-boot-java-app**

Group: **com.cicd**

Artifact: **cicd-applied-to-spring-boot-java-app**

Description: **Implementing CI/CD on Spring Boot Java App**

Package: **com.cicd.cicd-applied-to-spring-boot-java-app**

By default:

Type: **Maven**

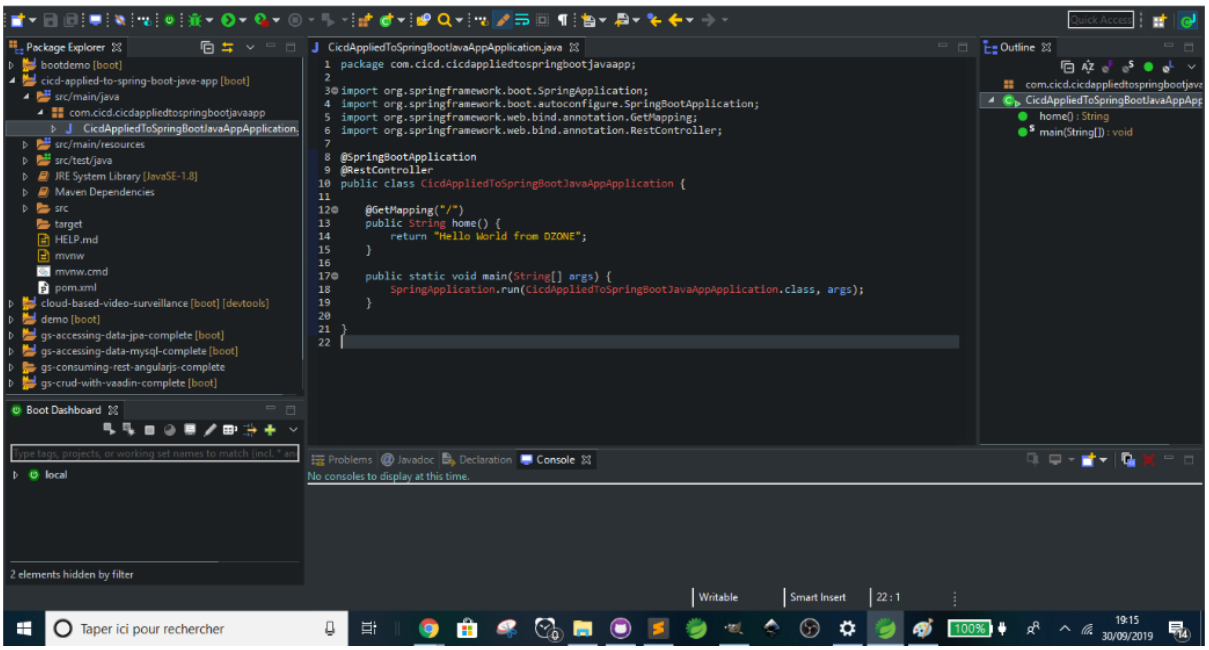
Packaging: **jar**

Java Version: **8**

Language: **Java**

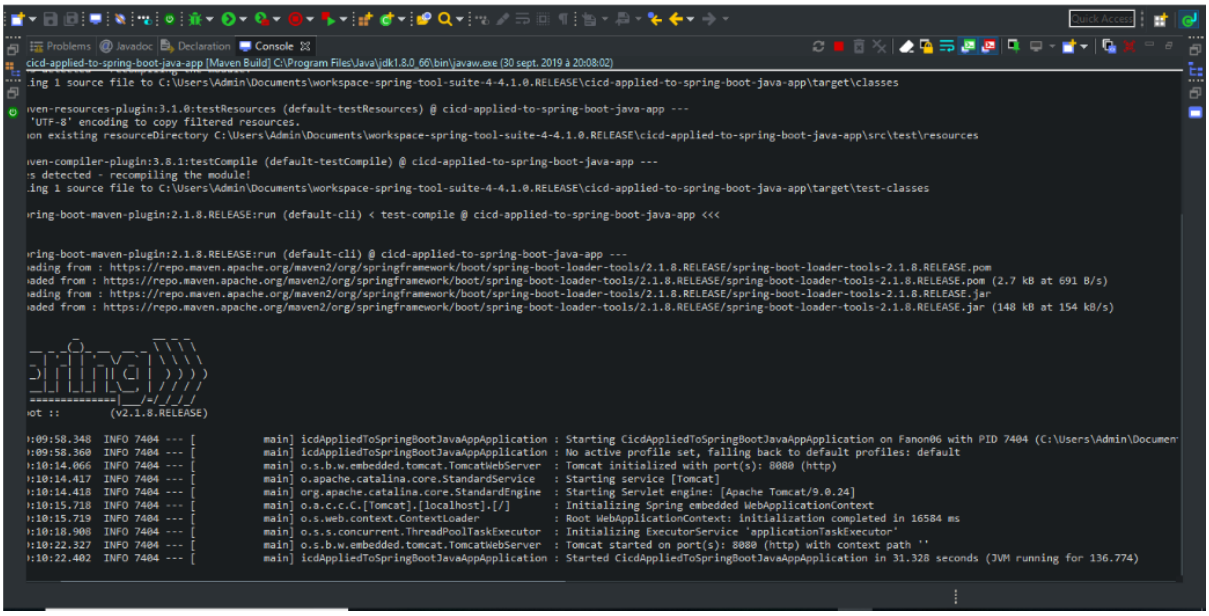
Open the **CicdAppliedToSpringBootJavaAppApplication.java file.**

We can then add a basic endpoint:

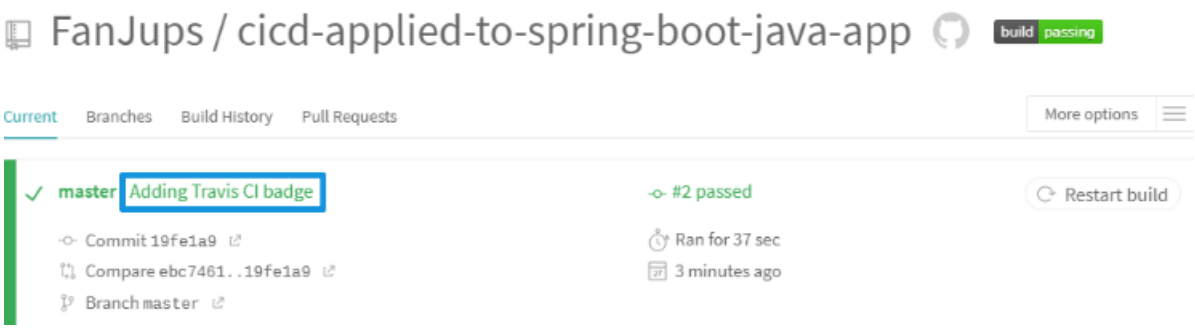


2. Create a GitHub Repository

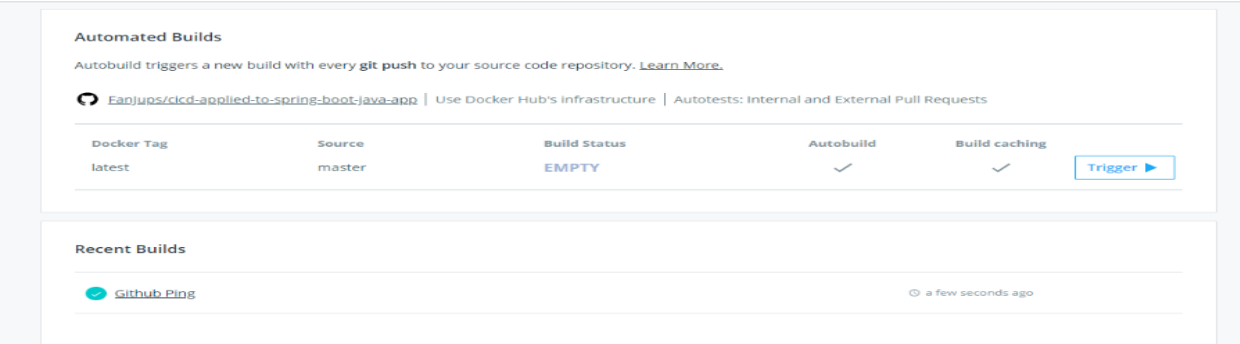
3. Use Travis CI and Docker to Implement CI/CD

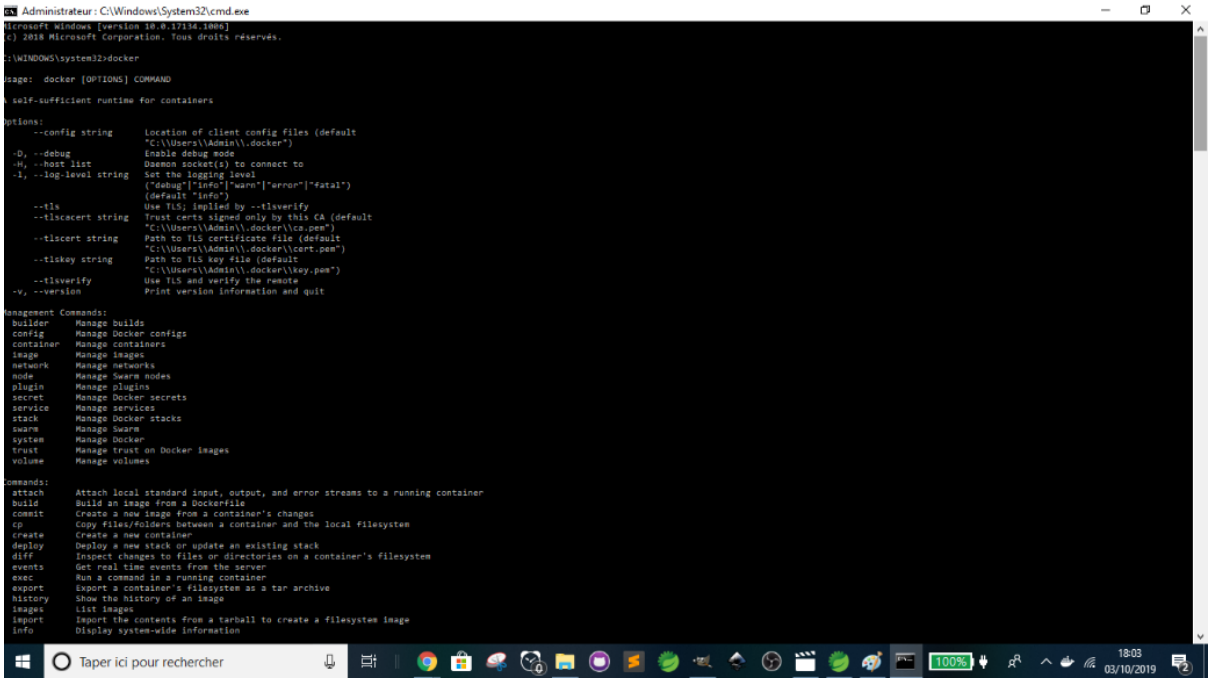


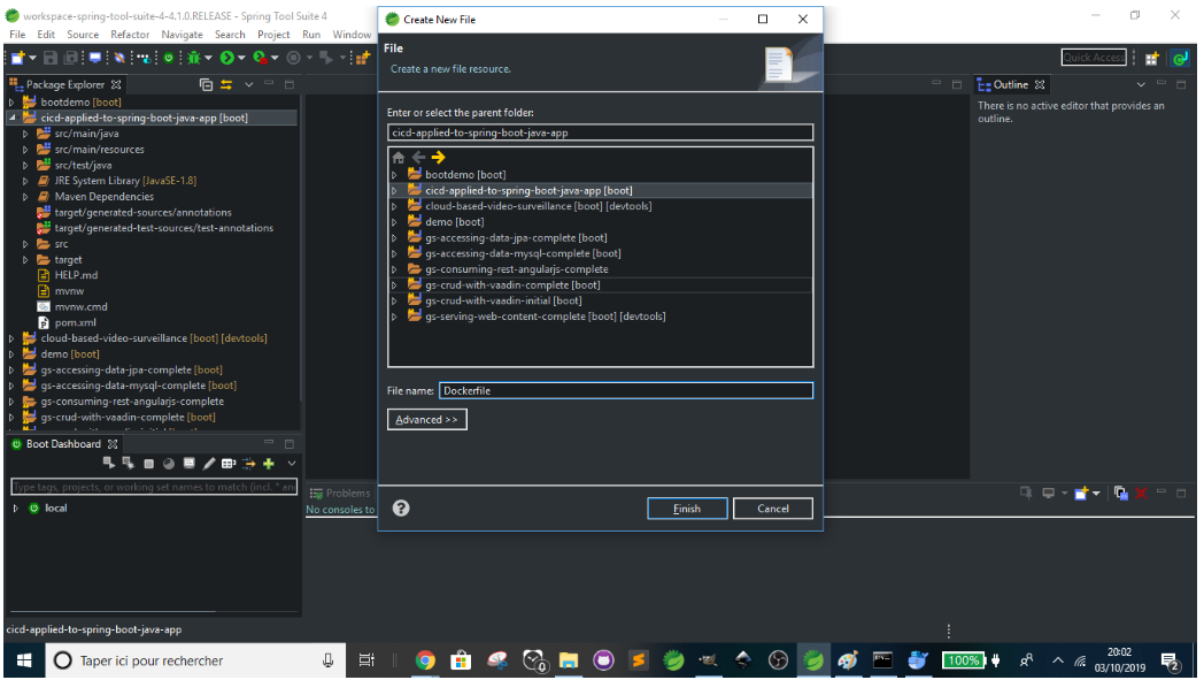
Sign up or sign in with GitHub and make sure Travis CI has access to your repository. Then, create a file named **.travis.yml**, which contains instructions that Travis CI will follow



The repository on Travis CI. We successfully added Travis CI and its badge. Next, we'll focus on Docker.

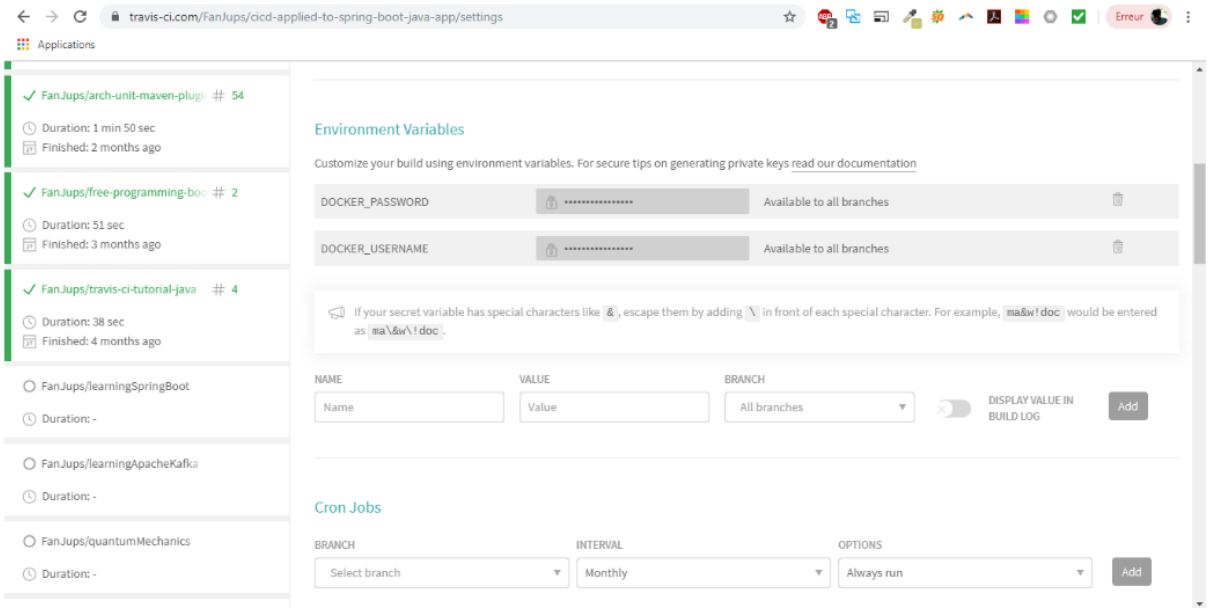


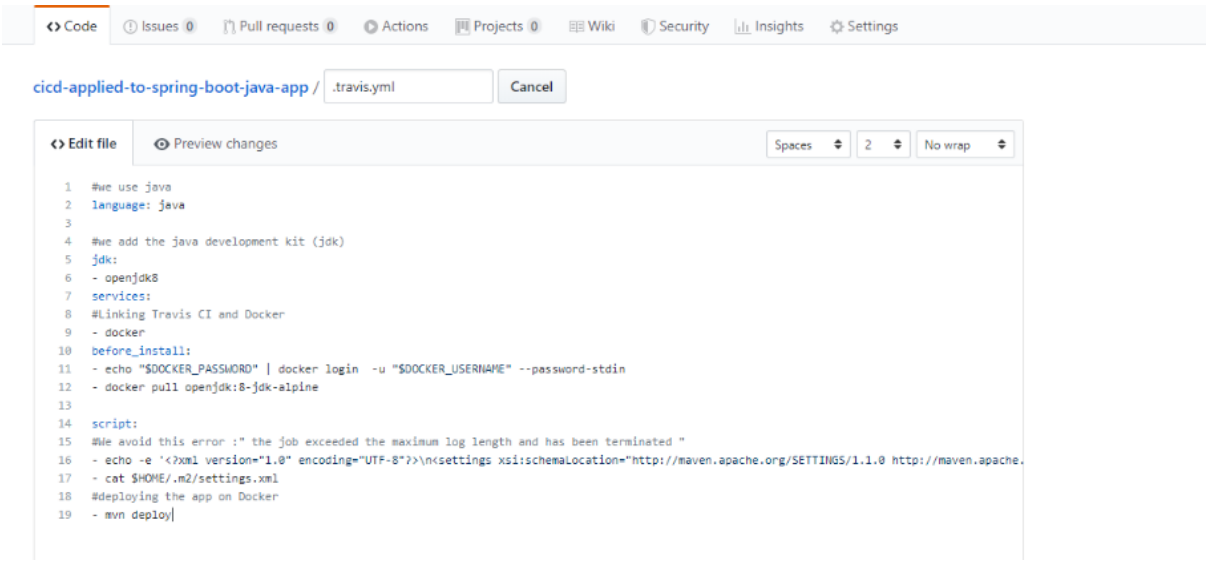




We will create two environment variables in Travis CI.

To get there, just copy and paste this (https://travis-ci.com/GITHUBUSERNAME/cicd-applied-to-spring-boot-java-app) in your browser. But replace **GITHUBUSERNAME** with your correct username or click on your Travis CI badge present in README.md:





Now that we're back to SonarCloud, choose a Key. I suggest using **"cicd-applied-to-spring-boot-java-app"** as the Key.

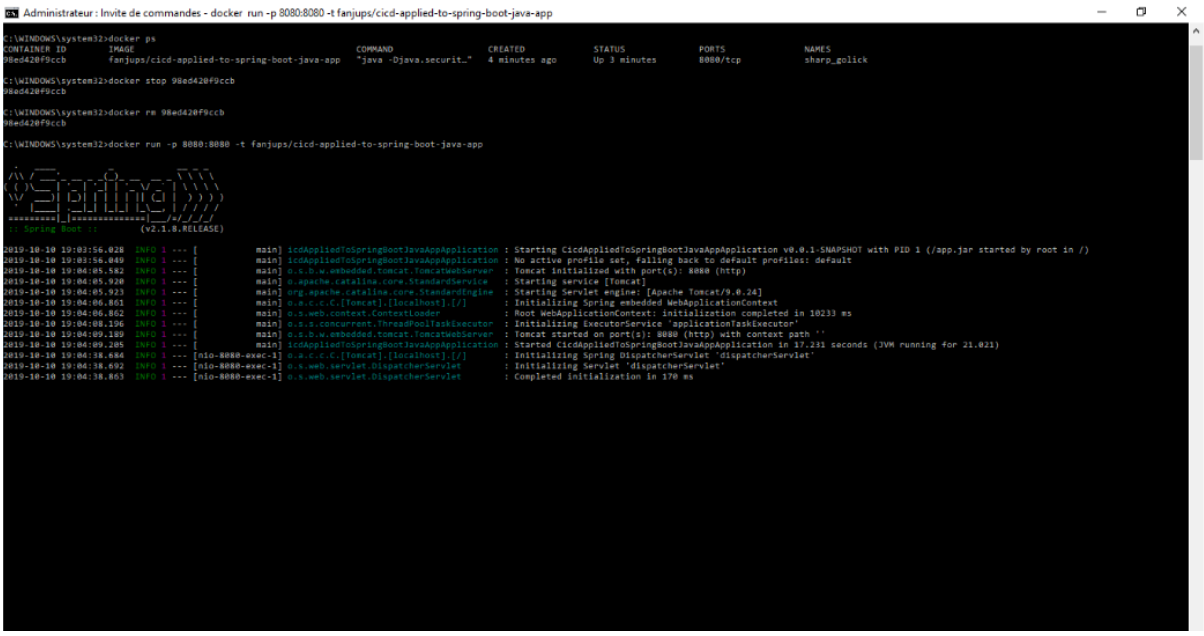
Then, click on **Continue -> Choose Free plan -> Create Organization -> Analyze new project -> Select your GitHub repository -> Set Up -> With Travis CI -> Provide and encrypt your token -> Copy**

Go back to Travis CI and create a SonarCloud environment variable named **SONAR_TOKEN.** As a value, paste the token you've just copied.

Now, back to SonarCloud and click on **Continue -> Edit your .travis.yml file -> Choose Maven as build technology -> Configure your platform -> Configure the scanner -> Copy.**

I chose to write SonarCloud script under **after_success** instead of **script** because I focus on deployment here. You are free to place it where you want.

Also, create a file named **sonar-project.properties** and edit as follows: **sonar.projectKey=GITHUBUSERNAME_cicd-applied-to-spring-boot-java-app**



Deploy the App on Heroku Using heroku-maven-plugin

Click on **New -> Create new app.** To continue, enter an app name (**cicd-spring-boot-java-app**). **cicd-applied-to-spring-boot-java-app** is too long as an app name. Choose a region and click **Create app**.

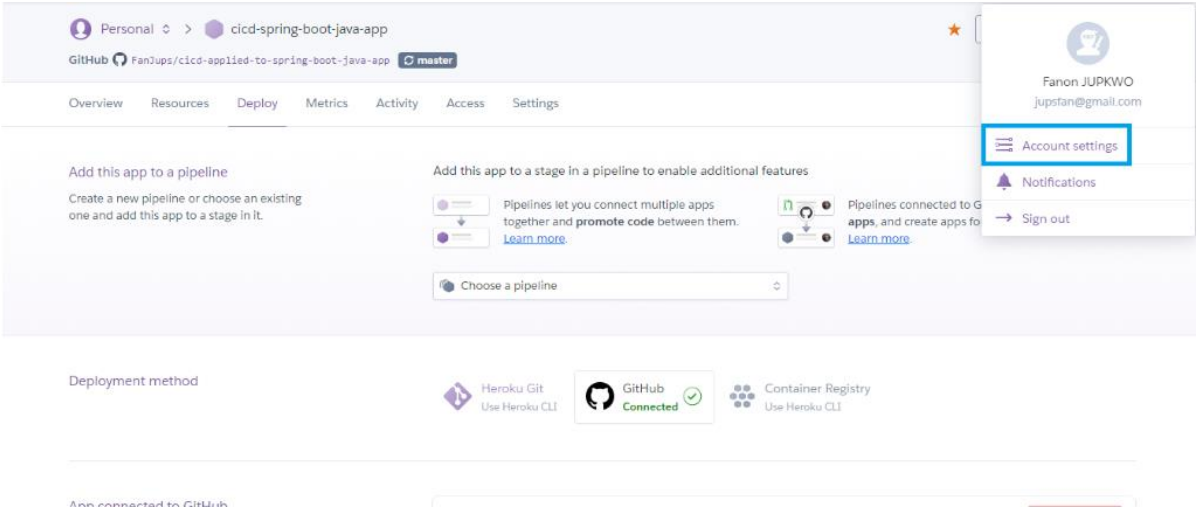
Next, click **Connect to GitHub.**

Search the GitHub repository. Once you find it, click **Connect.**

Check **Wait for CI to pass before deploy.**

Click **Enable Automatic Deploys.**

Go to **Account settings.**


Personal
cicd-spring-boot-java-app
GitHub FanJups/cicd-applied-to-spring-boot-java-app
master
Overview
Resources
Deploy
Metrics
Activity
Access
Settings
Fanon JUPKWO
jupsfan@gmail.com
Account settings
Notifications
Sign out
Add this app to a pipeline
Create a new pipeline or choose an existing one and add this app to a stage in it.
Add this app to a stage in a pipeline to enable additional features
Pipelines let you connect multiple apps together and promote code between them.
Learn more.
Pipelines connected to G
apps, and create apps fo
Learn more.
Choose a pipeline
Deployment method
Heroku Git
Use Heroku CLI
GitHub
Connected
Container Registry
Use Heroku CLI