

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:

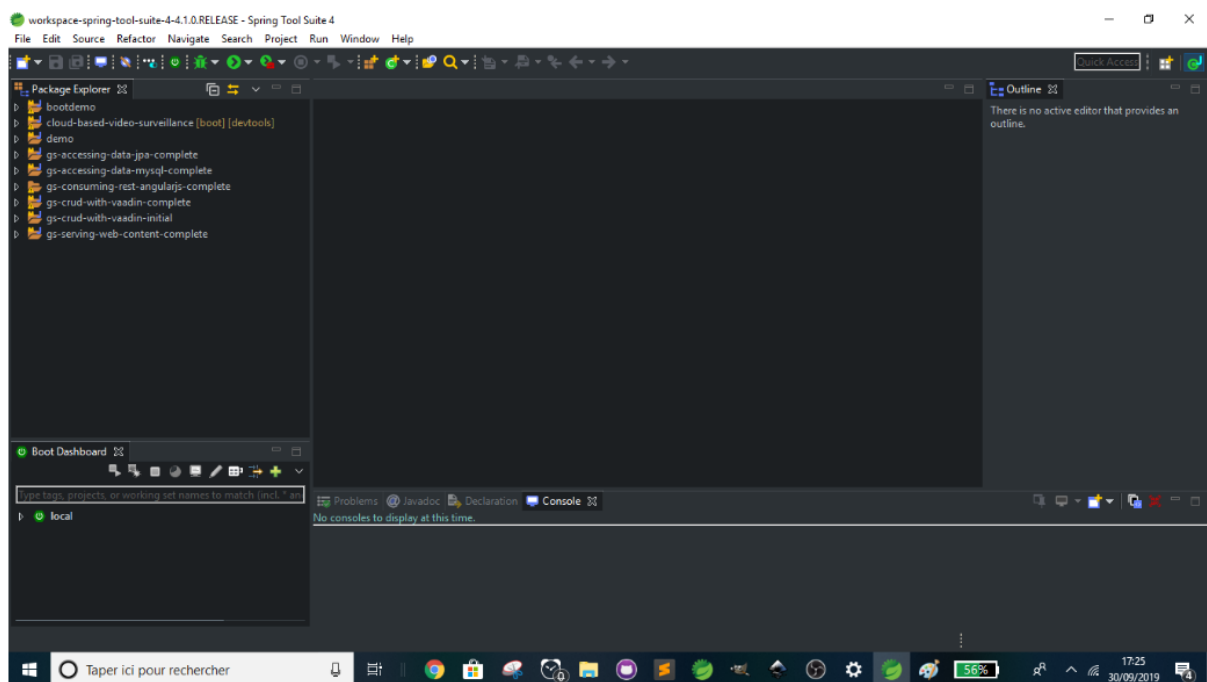
- Eclipse
- GitHub
- Jenkins
- AWS EC2/ Virtual machine

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**

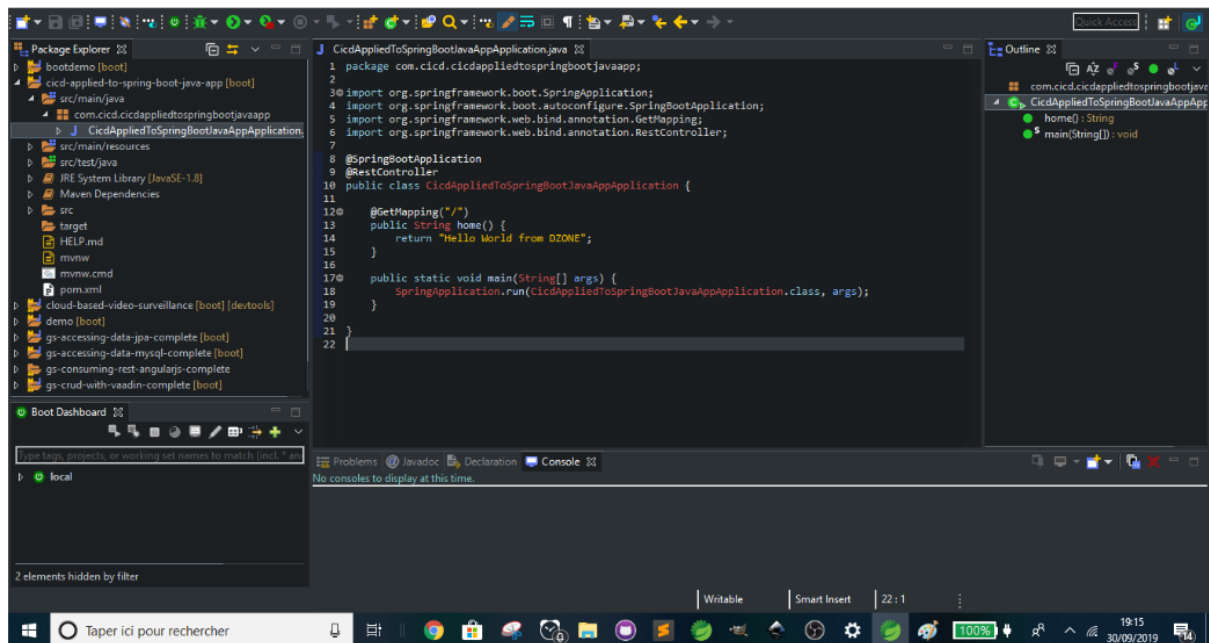
Packaging: **jar**

Java Version: **8**

Language: **Java**

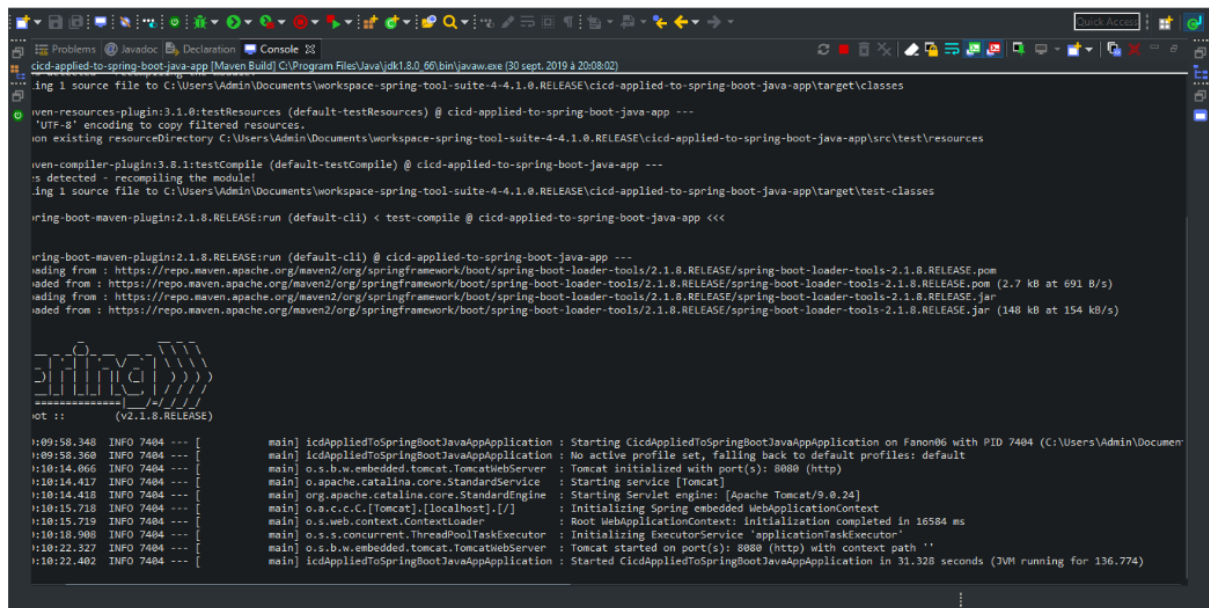
Open the **CicdAppliedToSpringBootJavaAppApplication.java** file.

We can then add a basic endpoint:



2. Create a GitHub Repository

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



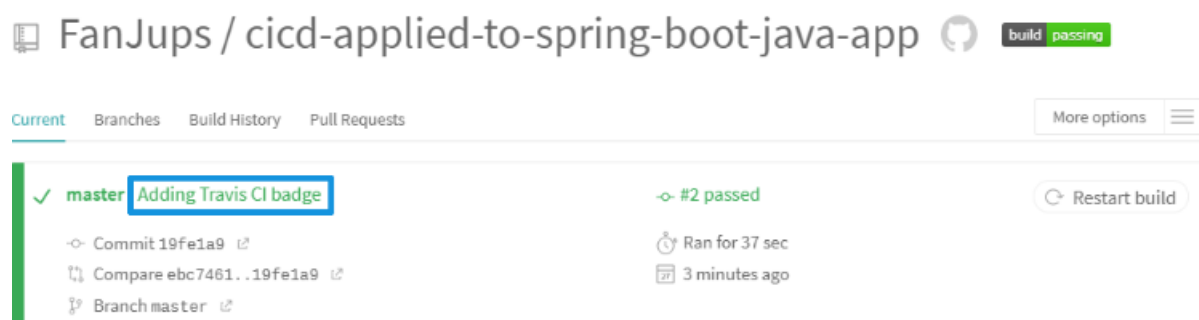
```
cd\applied-to-spring-boot-java-app [Maven Build] C:\Program Files\Java\jdk1.8.0_66\bin\java.exe (30 sept. 2019 à 20:08:02)
ing 1 source file to C:\Users\Admin\Documents\workspace-spring-tool-suite-4-4.1.0.RELEASE\cd\applied-to-spring-boot-java-app\target\classes
ven-resources-plugin:3.1.0:testResources (default-testResources) @ cd\applied-to-spring-boot-java-app ---
'UTF-8' encoding to copy filtered resources.
on existing resourceDirectory C:\Users\Admin\Documents\workspace-spring-tool-suite-4-4.1.0.RELEASE\cd\applied-to-spring-boot-java-app\src\test\resources
ven-compiler-plugin:3.8.1:testCompile (default-testCompile) @ cd\applied-to-spring-boot-java-app ---
is detected - recompiling the module!
ing 1 source file to C:\Users\Admin\Documents\workspace-spring-tool-suite-4-4.1.0.RELEASE\cd\applied-to-spring-boot-java-app\target\test-classes
ring-boot-maven-plugin:2.1.8.RELEASE:run (default-cli) < test-compile @ cd\applied-to-spring-boot-java-app <<<

ring-boot-maven-plugin:2.1.8.RELEASE:run (default-cli) @ cd\applied-to-spring-boot-java-app ---
ading from : https://repo.maven.apache.org/maven2/org/springframework/boot/spring-boot-loader-tools/2.1.8.RELEASE/spring-boot-loader-tools-2.1.8.RELEASE.pom
aded from : https://repo.maven.apache.org/maven2/org/springframework/boot/spring-boot-loader-tools/2.1.8.RELEASE/spring-boot-loader-tools-2.1.8.RELEASE.pom (2.7 kB at 691 B/s)
ading from : https://repo.maven.apache.org/maven2/org/springframework/boot/spring-boot-loader-tools/2.1.8.RELEASE/spring-boot-loader-tools-2.1.8.RELEASE.jar
aded from : https://repo.maven.apache.org/maven2/org/springframework/boot/spring-boot-loader-tools/2.1.8.RELEASE/spring-boot-loader-tools-2.1.8.RELEASE.jar (140 kB at 154 kB/s)

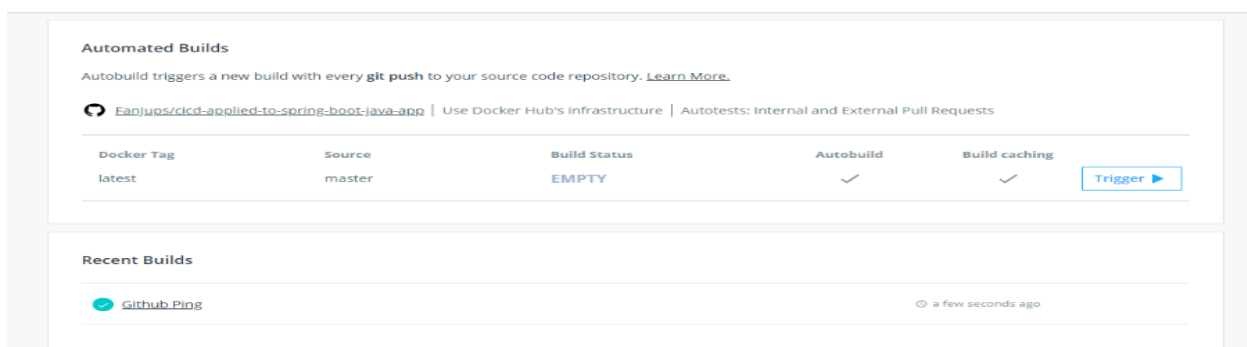
ot ::
(v2.1.8.RELEASE)

:09:50.348 INFO 7404 --- [main] cd\AppliedToSpringBootApplication : Starting CidAppliedToSpringBootApplication on Fanon86 with PID 7404 (C:\Users\Admin\Documents\workspace-spring-tool-suite-4-4.1.0.RELEASE\cd\applied-to-spring-boot-java-app\target\test-classes)
:09:50.360 INFO 7404 --- [main] cd\AppliedToSpringBootApplication : No active profile set, falling back to default profiles: default
:10:14.066 INFO 7404 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
:10:14.417 INFO 7404 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
:10:14.418 INFO 7404 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.24]
:10:15.718 INFO 7404 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
:10:15.719 INFO 7404 --- [main] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 16584 ms
:10:16.980 INFO 7404 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
:10:22.327 INFO 7404 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
:10:22.402 INFO 7404 --- [main] cd\AppliedToSpringBootApplication : Started CidAppliedToSpringBootApplication in 31.328 seconds (JVM running for 136.774)
```

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.



```
Administrateur: C:\Windows\System32\cmd.exe
Microsoft Windows [version 10.0.17134.1000]
(c) 2018 Microsoft Corporation. Tous droits réservés.

C:\WINDOWS\system32>docker

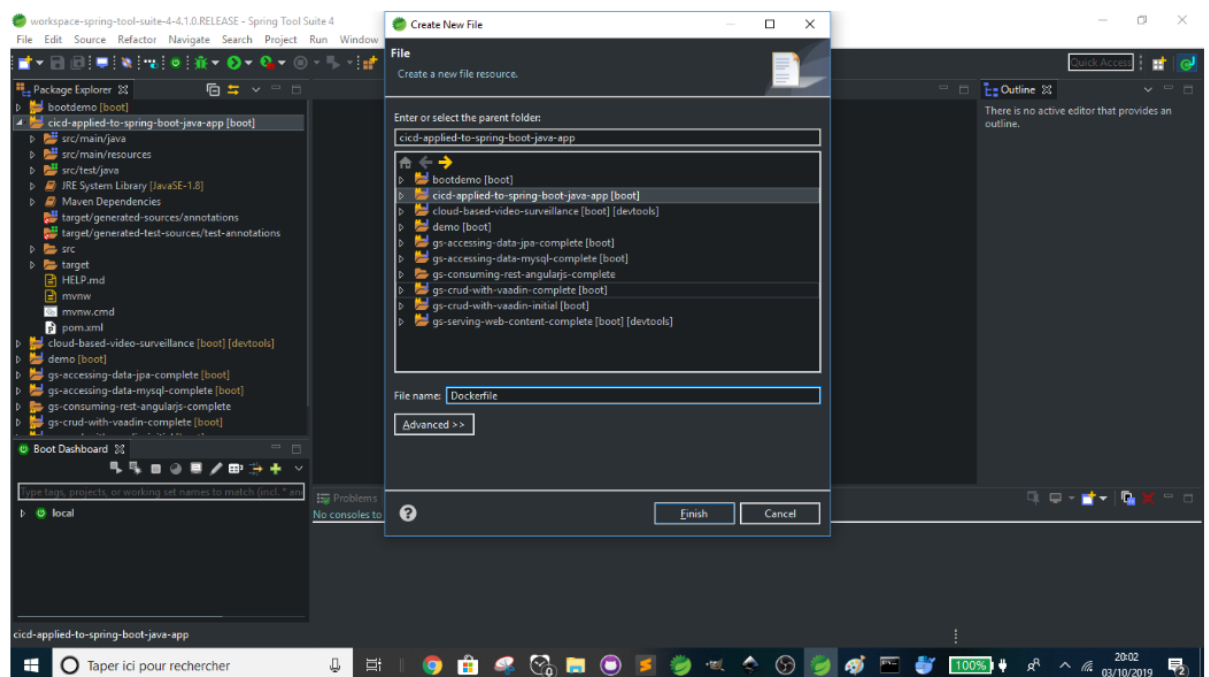
Usage: docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
  --config string      Location of client config files (default
                        "C:\Users\Admin\docker")
  -D, --debug           Enable debug mode
  --host list           Daemon socket(s) to connect to
  --log-level string    Set the logging level
                        ("debug"|"info"|"warn"|"error"|"fatal")
                        (default "info")
  --tls                Use TLS; implied by --tlsverify
  --tlscacert string    Trust certs signed only by this CA (default
                        "C:\Users\Admin\docker\ca.pem")
  --tlscert string      Path to TLS certificate file (default
                        "C:\Users\Admin\docker\cert.pem")
  --tlskey string       Path to TLS key file (default
                        "C:\Users\Admin\docker\key.pem")
  --tlsverify           Use TLS and verify the remote
  --version             Print version information and quit

Management Commands:
  builder              Manage builds
  config               Manage Docker configs
  container            Manage containers
  image                Manage images
  network              Manage networks
  node                 Manage Swarm nodes
  plugin               Manage plugins
  secret               Manage Docker secrets
  service              Manage services
  stack                Manage Docker stacks
  swarm                Manage Swarm
  system               Manage Docker
  trust                Manage trust on Docker images
  volume               Manage volumes

Commands:
  attach               Attach local standard input, output, and error streams to a running container
  build                Build an image from a Dockerfile
  commit               Create a new image from a container's changes
  cp                  Copy files/folders between a container and the local filesystem
  create               Create a new container
  deploy               Deploy a new stack or update an existing stack
  diff                 Inspect changes to files or directories on a container's filesystem
  events               Get real time events from the server
  exec                Run a command in a running container
  export               Export a container's filesystem as a tar archive
  history              Show the history of an image
  images               List images
  import               Import the contents from a tarball to create a filesystem image
  info                 Display system-wide information
```



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:

← → ↻ travis-ci.com/FanJups/cicd-applied-to-spring-boot-java-app/settings

Applications

- ✓ FanJups/arch-unit-maven-plugin # 54
Duration: 1 min 50 sec
Finished: 2 months ago
- ✓ FanJups/free-programming-book # 2
Duration: 51 sec
Finished: 3 months ago
- ✓ FanJups/travis-ci-tutorial-java # 4
Duration: 38 sec
Finished: 4 months ago

○ FanJups/learningSpringBoot
Duration: -

○ FanJups/learningApacheKafka
Duration: -

○ FanJups/quantumMechanics
Duration: -

Environment Variables

Customize your build using environment variables. For secure tips on generating private keys read our documentation

DOCKER_PASSWORD	*****	Available to all branches	🗑
DOCKER_USERNAME	*****	Available to all branches	🗑

🔑 If your secret variable has special characters like &, escape them by adding \ in front of each special character. For example, `ma&w!doc` would be entered as `ma&w\!doc`.

NAME	VALUE	BRANCH		DISPLAY VALUE IN BUILD LOG	Add
Name	Value	All branches	🔑	<input type="checkbox"/>	

Cron Jobs

BRANCH	INTERVAL	OPTIONS	
Select branch	Monthly	Always run	Add

Code Issues 0 Pull requests 0 Actions Projects 0 Wiki Security Insights Settings

cicd-applied-to-spring-boot-java-app / .travis.yml Cancel

Edit file Preview changes Spaces 2 No wrap

```
1 #we use java
2 language: java
3
4 #we add the java development kit (jdk)
5 jdk:
6 - openjdk8
7 services:
8 #Linking Travis CI and Docker
9 - docker
10 before_install:
11 - echo "$DOCKER_PASSWORD" | docker login -u "$DOCKER_USERNAME" --password-stdin
12 - docker pull openjdk:8-jdk-alpine
13
14 script:
15 #We avoid this error : " the job exceeded the maximum log length and has been terminated "
16 - echo -e "<?xml version='1.0' encoding='UTF-8'><n<settings xsi:schemaLocation='http://maven.apache.org/SETTINGS/1.1.0 http://maven.apache.org/xsd/maven-1.1.0'></n></xml>"
17 - cat $HOME/.m2/settings.xml
18 #deploying the app on Docker
19 - mvn deploy
```

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**

```
Administrateur : Invite de commandes - docker run -p 8080:8080 -t fanjups/cicd-applied-to-spring-boot-java-app
C:\WINDOWS\system32\docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
98ed420f9ccb       fanjups/cicd-appl... "java -Djava.secur... 4 minutes ago       Up 3 minutes       8080/tcp           sharp_golick

C:\WINDOWS\system32\docker stop 98ed420f9ccb
98ed420f9ccb

C:\WINDOWS\system32\docker rm 98ed420f9ccb
98ed420f9ccb

C:\WINDOWS\system32\docker run -p 8080:8080 -t fanjups/cicd-applied-to-spring-boot-java-app

Starting
=====
[+] Spring Boot [+] (v2.1.8.RELEASE)

2019-10-10 19:03:56.028 INFO 1 --- [main] iCdAppliedToSpringBootApplication : Starting iCdAppliedToSpringBootApplication v0.0.1-SNAPSHOT with PID 1 (/app.jar started by root in /)
2019-10-10 19:03:56.049 INFO 1 --- [main] iCdAppliedToSpringBootApplication : No active profile set, falling back to default profiles: default
2019-10-10 19:04:05.342 INFO 1 --- [main] o.s.s.w.embedded.tomcat.TomcatStarter : Tomcat initialized with port(s): 8080 (http)
2019-10-10 19:04:05.920 INFO 1 --- [main] o.apache.catalina.core.StandardEngine : Starting service [Tomcat]
2019-10-10 19:04:05.923 INFO 1 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.24]
2019-10-10 19:04:06.861 INFO 1 --- [main] o.s.s.w.embedded.tomcat.TomcatStarter : Initializing Spring embedded WebApplicationContext
2019-10-10 19:04:06.862 INFO 1 --- [main] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 10233 ms
2019-10-10 19:04:08.196 INFO 1 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2019-10-10 19:04:09.150 INFO 1 --- [main] o.s.s.w.embedded.tomcat.TomcatStarter : Tomcat started on port(s): 8080 (http) with context path ..
2019-10-10 19:04:09.205 INFO 1 --- [main] iCdAppliedToSpringBootApplication : Started iCdAppliedToSpringBootApplication in 17.231 seconds (JVM running for 21.021)
2019-10-10 19:04:18.484 INFO 1 --- [nio-8080-exec-1] o.s.s.c.c.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2019-10-10 19:04:18.492 INFO 1 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2019-10-10 19:04:18.863 INFO 1 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 170 ms
```

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

OverviewResourcesDeployMetricsActivityAccessSettings

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 GitHub apps, and create apps for [Learn more.](#)

Choose a pipeline

Fanon JUPKWO

jupstan@gmail.com

Account settings

Notifications

Sign out

Deployment method



App connected to GitHub