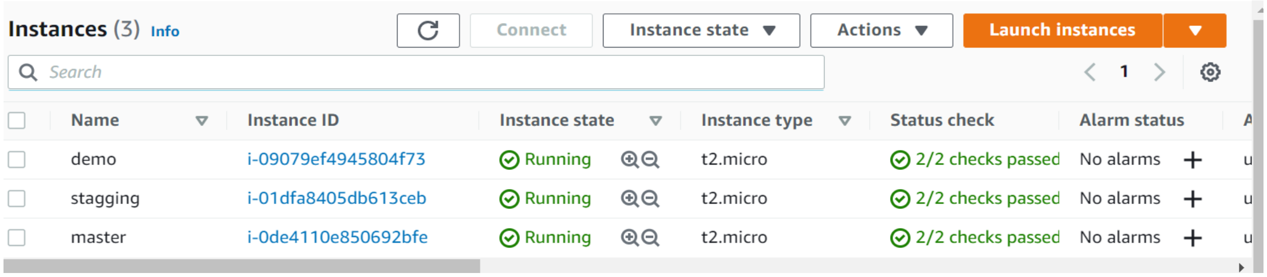
How to Automate the Deployment of a Web application

A Guide to build a CICD pipeline for automated deployment of web application using Git and Jenkins.

Tools/Technologies used –

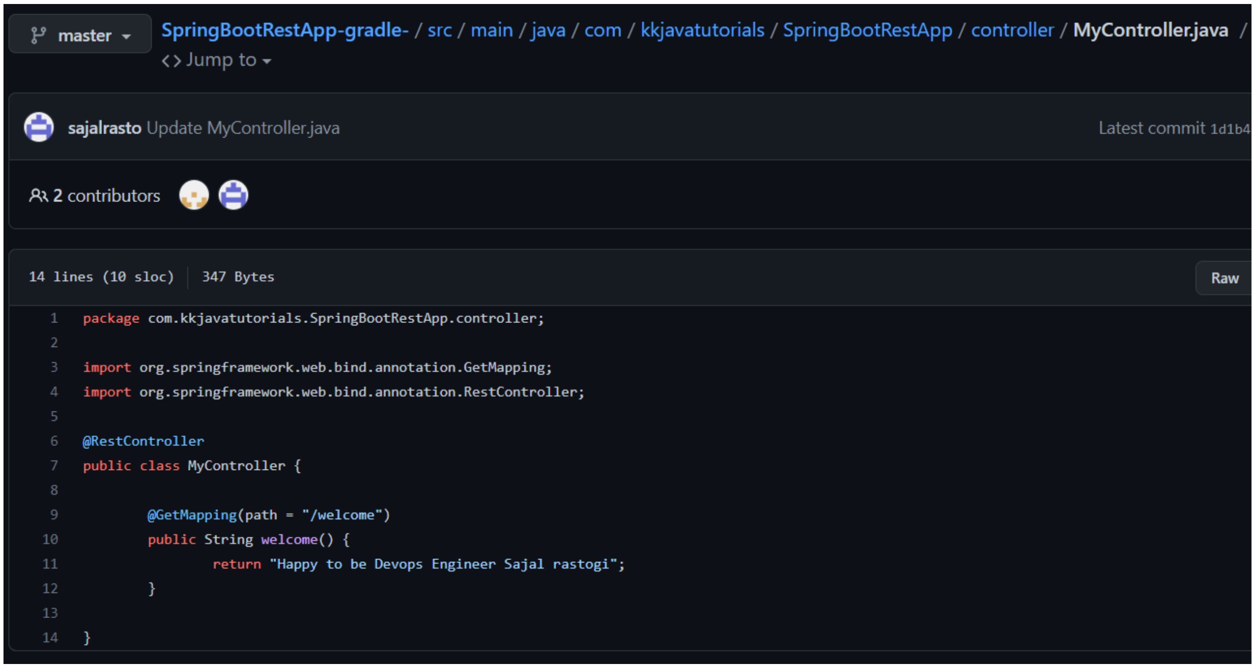
- Docker (Containerization platform)
- GitHub (for Source Code)
- Gradle (Build Tool)
- Jenkins (Automation CI/CD)
- AWS (Infrastructure/ Cloud platform)

Created two instances - one for master one for stagging(demo)



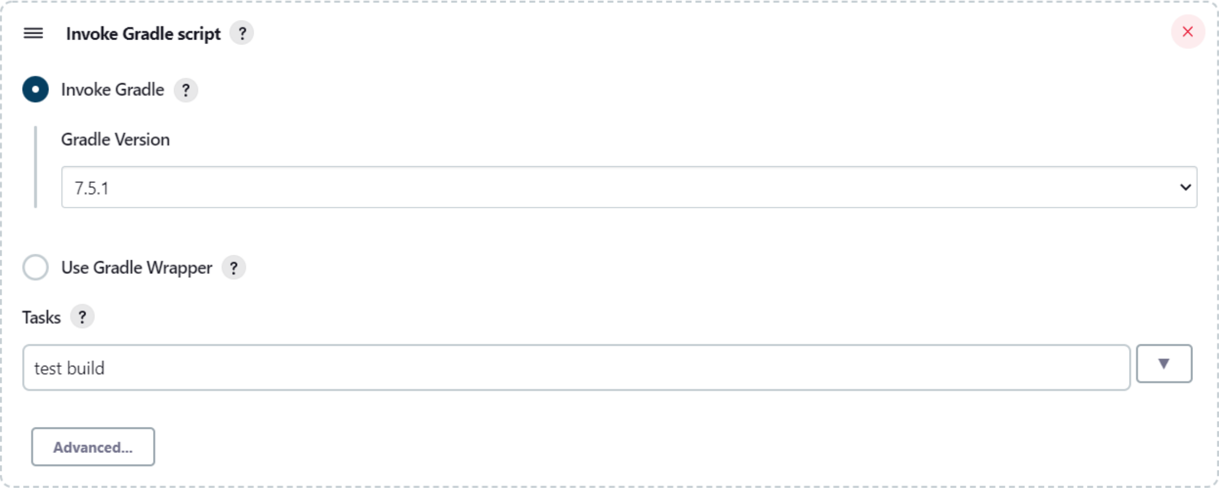
□	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	A
□	demo	i-09079ef4945804f73	✓ Running	t2.micro	✓ 2/2 checks passed	No alarms +	u
□	stagging	i-01dfa8405db613ceb	✓ Running	t2.micro	✓ 2/2 checks passed	No alarms +	u
□	master	i-0de4110e850692bfe	✓ Running	t2.micro	✓ 2/2 checks passed	No alarms +	u

Source code from GitHub repository


master
SpringBootRestApp-gradle- / src / main / java / com / kkjavatutorials / SpringBootRestApp / controller / MyController.java /
<> Jump to
sajalrasto Update MyController.java
Latest commit 1d1b4
2 contributors
14 lines (10 sloc) | 347 Bytes
Raw

```java
package com.kkjavatutorials.SpringBootRestApp.controller;

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

@RestController
public class MyController {

        @GetMapping(path = "/welcome")
        public String welcome() {
                return "Happy to be Devops Engineer Sajal rastogi";
        }

}
```

Test and Build using Gradle



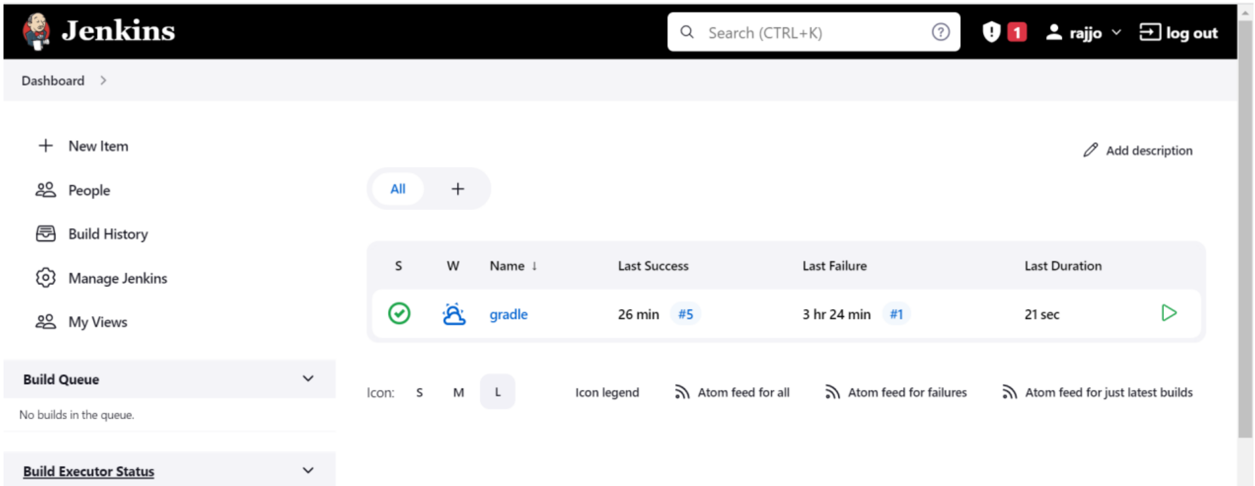
● Invoke Gradle ?

Gradle Version: 7.5.1

○ Use Gradle Wrapper ?

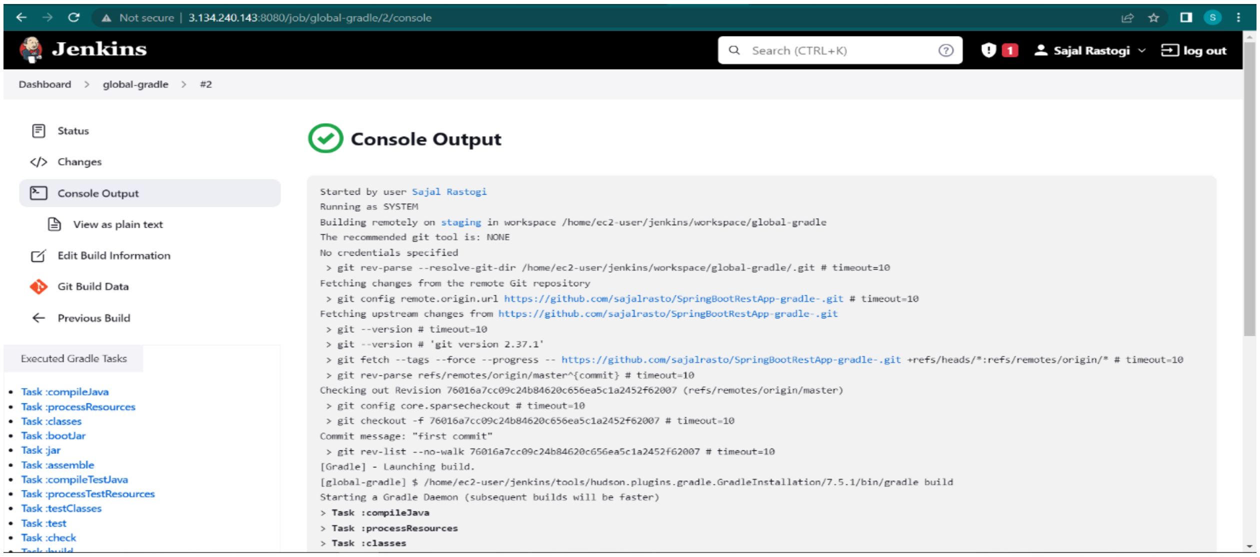
Tasks ?: test build

Created a Jenkins job



S	W	Name ↓	Last Success	Last Failure	Last Duration
✓		gradle	26 min #5	3 hr 24 min #1	21 sec

Console Output



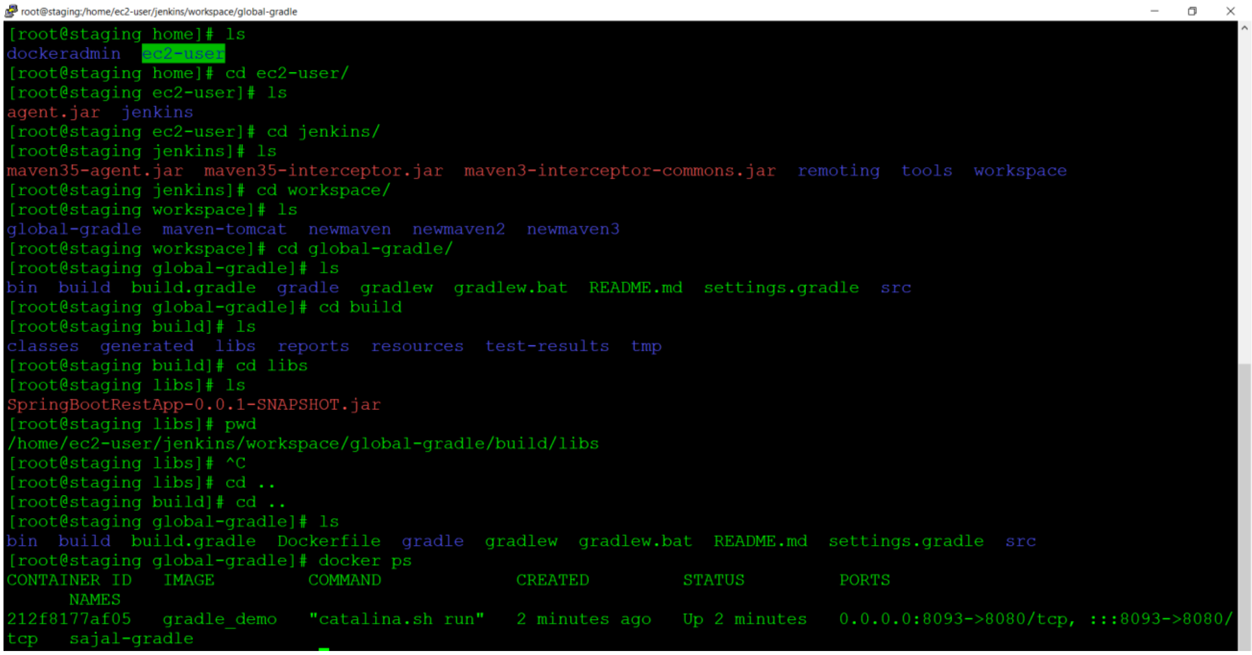
Executed Gradle Tasks

- Task :compileJava
- Task :processResources
- Task :classes
- Task :bootJar
- Task :jar
- Task :assemble
- Task :compileTestJava
- Task :processTestResources
- Task :testClasses
- Task :test
- Task :check
- Task :build

Console Output

```
Started by user Sajal Rastogi
Running as SYSTEM
Building remotely on staging in workspace /home/ec2-user/jenkins/workspace/global-gradle
The recommended git tool is: NONE
No credentials specified
 > git rev-parse --resolve-git-dir /home/ec2-user/jenkins/workspace/global-gradle/.git # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/sajalrasto/SpringBootRestApp-gradle-.git # timeout=10
Fetching upstream changes from https://github.com/sajalrasto/SpringBootRestApp-gradle-.git
 > git --version # timeout=10
 > git --version # 'git version 2.37.1'
 > git fetch --tags --force --progress -- https://github.com/sajalrasto/SpringBootRestApp-gradle-.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 76016a7cc09c24b84620c656ea5c1a2452f62007 (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 76016a7cc09c24b84620c656ea5c1a2452f62007 # timeout=10
Commit message: "first commit"
 > git rev-list --no-walk 76016a7cc09c24b84620c656ea5c1a2452f62007 # timeout=10
[Gradle] - Launching build.
[global-gradle] $ /home/ec2-user/jenkins/tools/hudson.plugins.gradle.GradleInstallation/7.5.1/bin/gradle build
Starting a Gradle Daemon (subsequent builds will be faster)
> Task :compileJava
> Task :processResources
> Task :classes
```

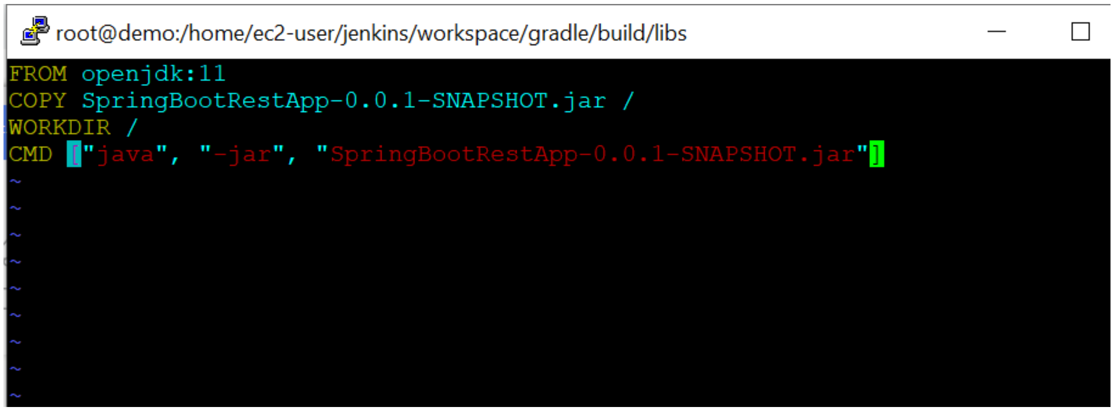
After building Gradle project, Container is created automatically using Dockerfile.



root@staging:/home/ec2-user/jenkins/workspace/global-gradle

```
[root@staging home]# ls
dockeradmin  ec2-user
[root@staging home]# cd ec2-user/
[root@staging ec2-user]# ls
agent.jar  jenkins
[root@staging ec2-user]# cd jenkins/
[root@staging jenkins]# ls
maven35-agent.jar  maven35-interceptor.jar  maven3-interceptor-commons.jar  remoting  tools  workspace
[root@staging jenkins]# cd workspace/
[root@staging workspace]# ls
global-gradle  maven-tomcat  newmaven  newmaven2  newmaven3
[root@staging workspace]# cd global-gradle/
[root@staging global-gradle]# ls
bin  build  build.gradle  gradle  gradlew  gradlew.bat  README.md  settings.gradle  src
[root@staging global-gradle]# cd build
[root@staging build]# ls
classes  generated  libs  reports  resources  test-results  tmp
[root@staging build]# cd libs
[root@staging libs]# ls
SpringBootRestApp-0.0.1-SNAPSHOT.jar
[root@staging libs]# pwd
/home/ec2-user/jenkins/workspace/global-gradle/build/libs
[root@staging libs]# ^C
[root@staging libs]# cd ..
[root@staging build]# cd ..
[root@staging global-gradle]# ls
bin  build  build.gradle  Dockerfile  gradle  gradlew  gradlew.bat  README.md  settings.gradle  src
[root@staging global-gradle]# docker ps
CONTAINER ID   IMAGE         COMMAND              CREATED         STATUS         PORTS
      NAMES
212f8177af05   gradle_demo   "catalina.sh run"    2 minutes ago   Up 2 minutes   0.0.0.0:8093->8080/tcp, :::8093->8080/
tcp   sajal-gradle
```

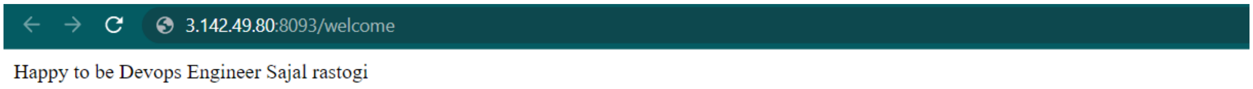
Content inside the Dockerfile



root@demo:/home/ec2-user/jenkins/workspace/gradle/build/libs

```
FROM openjdk:11
COPY SpringBootRestApp-0.0.1-SNAPSHOT.jar /
WORKDIR /
CMD ["java", "-jar", "SpringBootRestApp-0.0.1-SNAPSHOT.jar"]
~
~
~
~
~
~
~
~
~
```

Final output Displayed on 8093 port no. with get mapping endpoint : /welcome



Happy to be Devops Engineer Sajal rastogi