**Mayur Jagtap**

**Senior Software engineer**

**Jenkins**

**Topic covered**: -

1. Jenkins Installation
2. Create Job and run in Jenkins

* **Jenkins Installation**

1. First create Elastic Cloud Machine – Ubuntu 16.04 – 8 GB RAM
2. Go to Directory **cd /opt/**
3. RUN below commands to install Jenkins on Could machine
4. **sudo apt update -y 🡪 To update OS**
5. **sudo apt-get update -y**
6. **sudo apt-get install openjdk-8-jdk 🡪 To install Java on machine**
7. **wget -q -O - http://pkg.jenkins-ci.org/debian/jenkins-ci.org.key | sudo apt-key add – 🡪 to get Jenkins binary file, packages & its key**
8. **echo "deb http://pkg.jenkins-ci.org/debian binary/" | sudo tee -a /etc/apt/sources.list.d/jenkins.list**
9. **sudo apt-get update 🡪 Update OS again**
10. **sudo apt-get install Jenkins 🡪 Now run install Jenkins comma**
11. **service jenkins status 🡪 to check service status of Jenkins after installation**

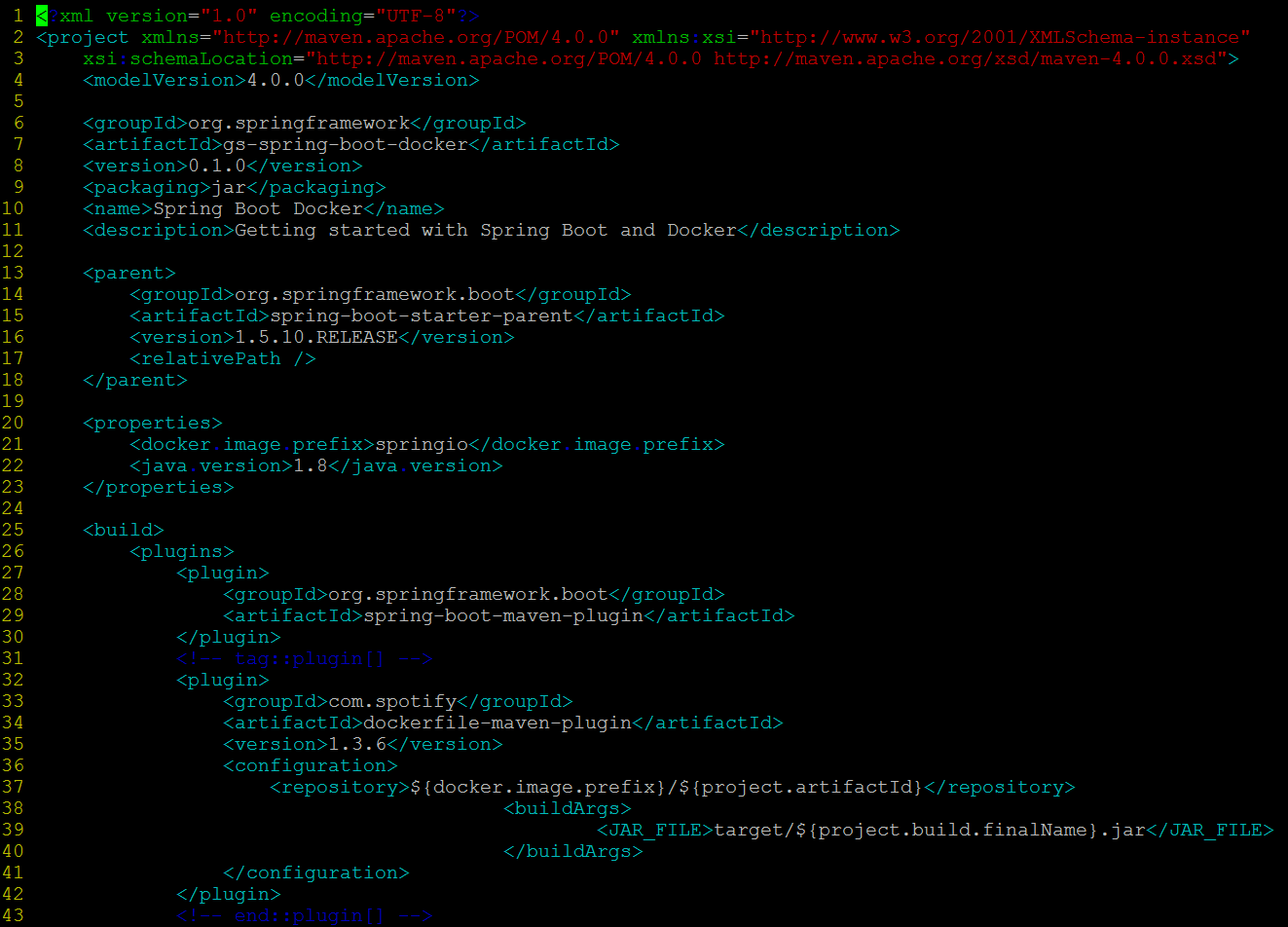
* **Maven Installation**

1. RUN below command on Cloud machine to install Maven
2. **apt install maven**
3. **sudo apt-get remove maven3**
4. **sudo apt-get remove maven**
5. **sudo add-apt-repository "deb http://ppa.launchpad.net/natecarlson/maven3/ubuntu precise main"**
6. **sudo apt-get update**
7. **sudo apt-get install maven3**
8. **sudo ln -s /usr/share/maven3/bin/mvn /usr/bin/mvn**
9. **echo $MAVEN\_HOME**
10. **echo $JAVA\_HOME**
11. **export MAVEN\_HOME=/usr/share/maven3**
12. **export JAVA\_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre**

**Directory Path**: **/opt/devops/docker-java-maven/gs-spring-boot-docker-master/complet**

* Now check the POM.xml file 🡪

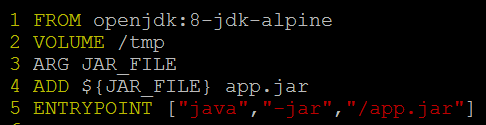
1. To check JAR file nomenclature. (Line 39)
2. Version/Release of Source Code for reference. (Line16/22)
3. Plugin detail to build JAR file. (Line 34)



* Now check Docker File 🡪

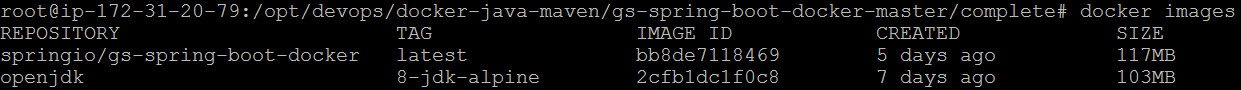
1. How JAVA image is getting pull from registry.
2. Directory is created **/tmp** to add JAR file in it.
3. Argument is pass as “JAR\_FILE” which is taken from POM.XML
4. Rename that JAR file to “app.jar”
5. Execute the JAR with command 🡪

**java -jar /app.jar**



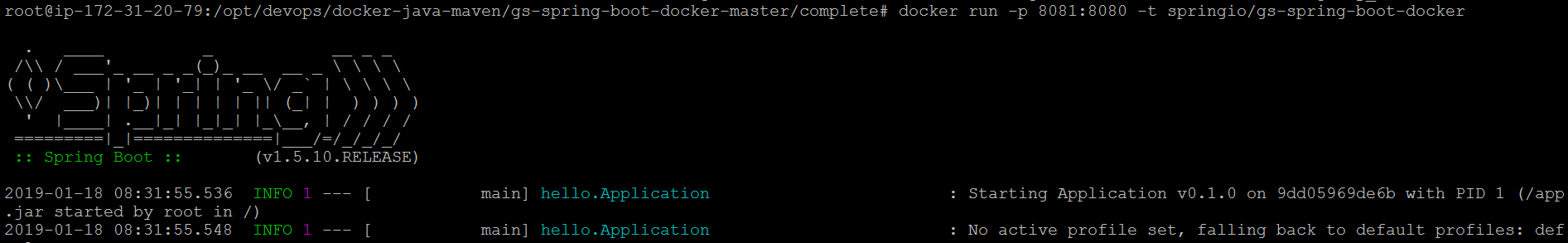
* Now to build the image using Docker first , execute below command 🡪

**mvn install dockerfile:build**



* Now to run image, we should expose port to run the application. Spring boot application by default hit the tomcat apache port 8080.

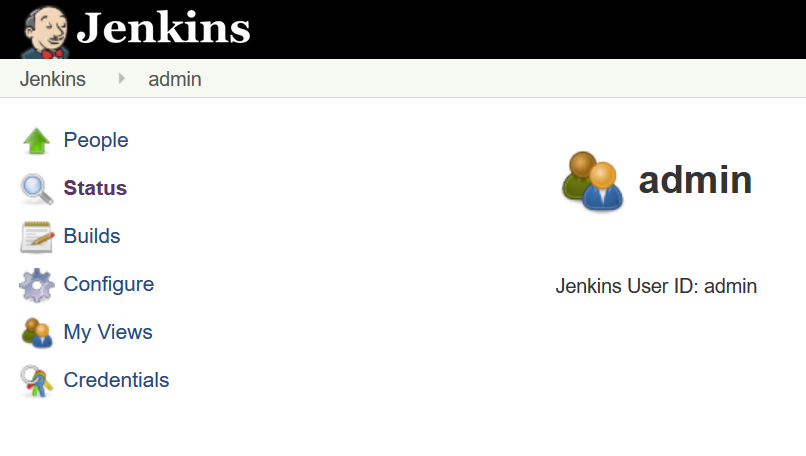
**docker run -p 8081:8080 -t springio/gs-spring-boot-docker**



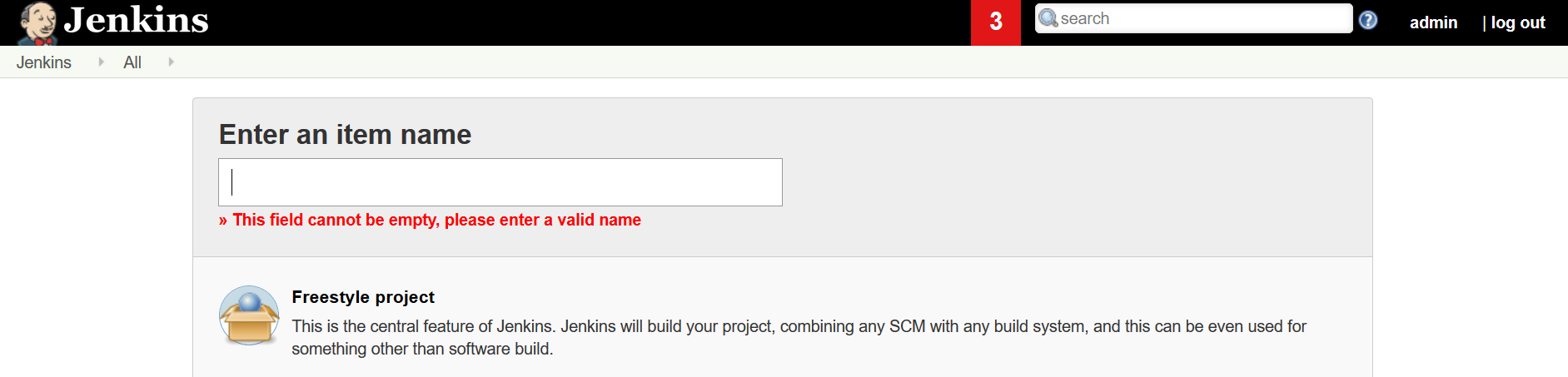
* Now build the image using Jenkins **🡪**

1. Hit the IP in explorer with port 8080 to open Jenkins Home Screen.

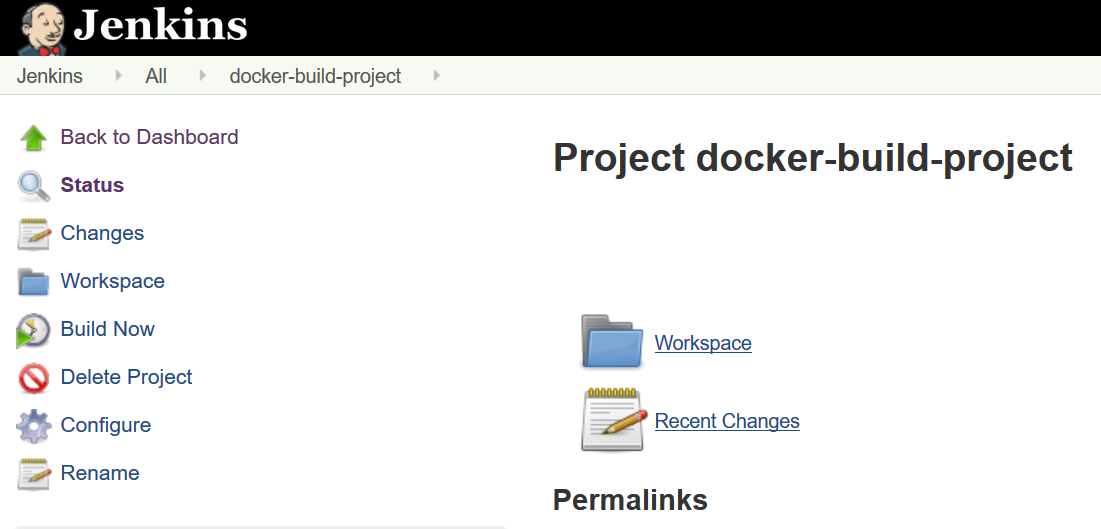
<http://3.16.36.152:8080/> ( Public IP :8080)



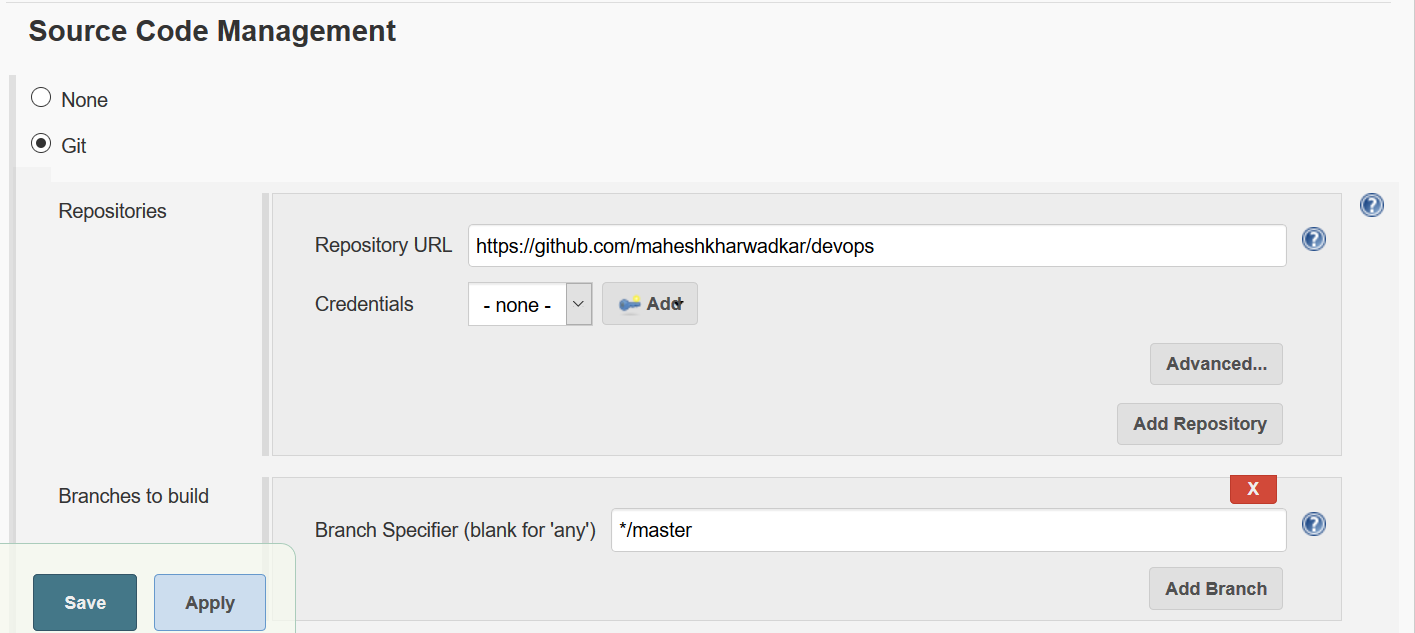
* **Create Project 🡪**



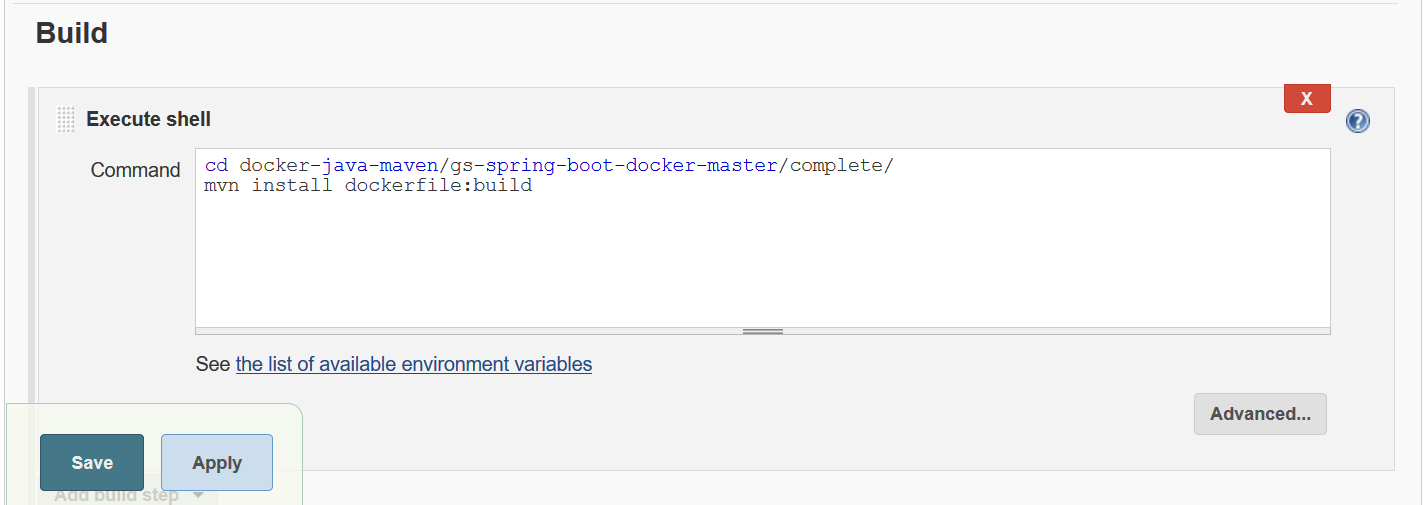
* Once project has been created , go to Configure



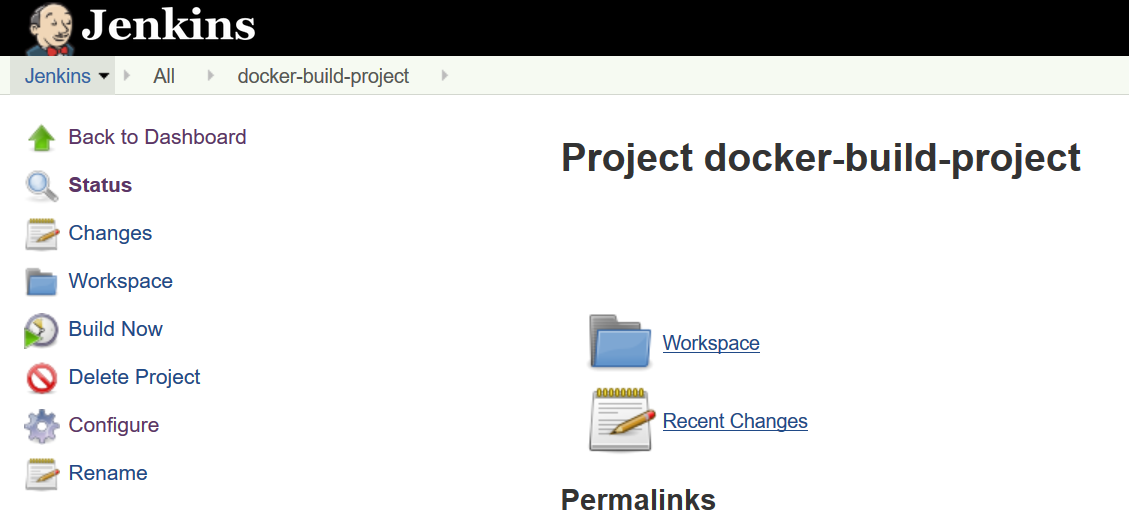
* Go to **Source Code Management**, select Git and paste repository URL to export the source code from Git. And save the changes.



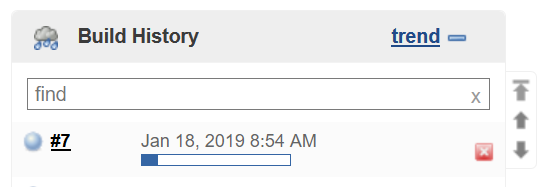
* Go to **BUILD**, and Shell Script to build the image **🡪**



* **cd docker-java-maven/gs-spring-boot-docker-master/complete/** 🡪It will pull the source code from Git Repository path
* **mvn install dockerfile:build** 🡪 It will build image and Jar file.
* Once it has been completed, go to project home screen.



* Click on Build Now, it will start building image 🡪



* Jenkins will do all tests like checking source code and all, building JAR file and image , and run the container out of it to open application.

**PFB Console Output from Jenkins**: -



