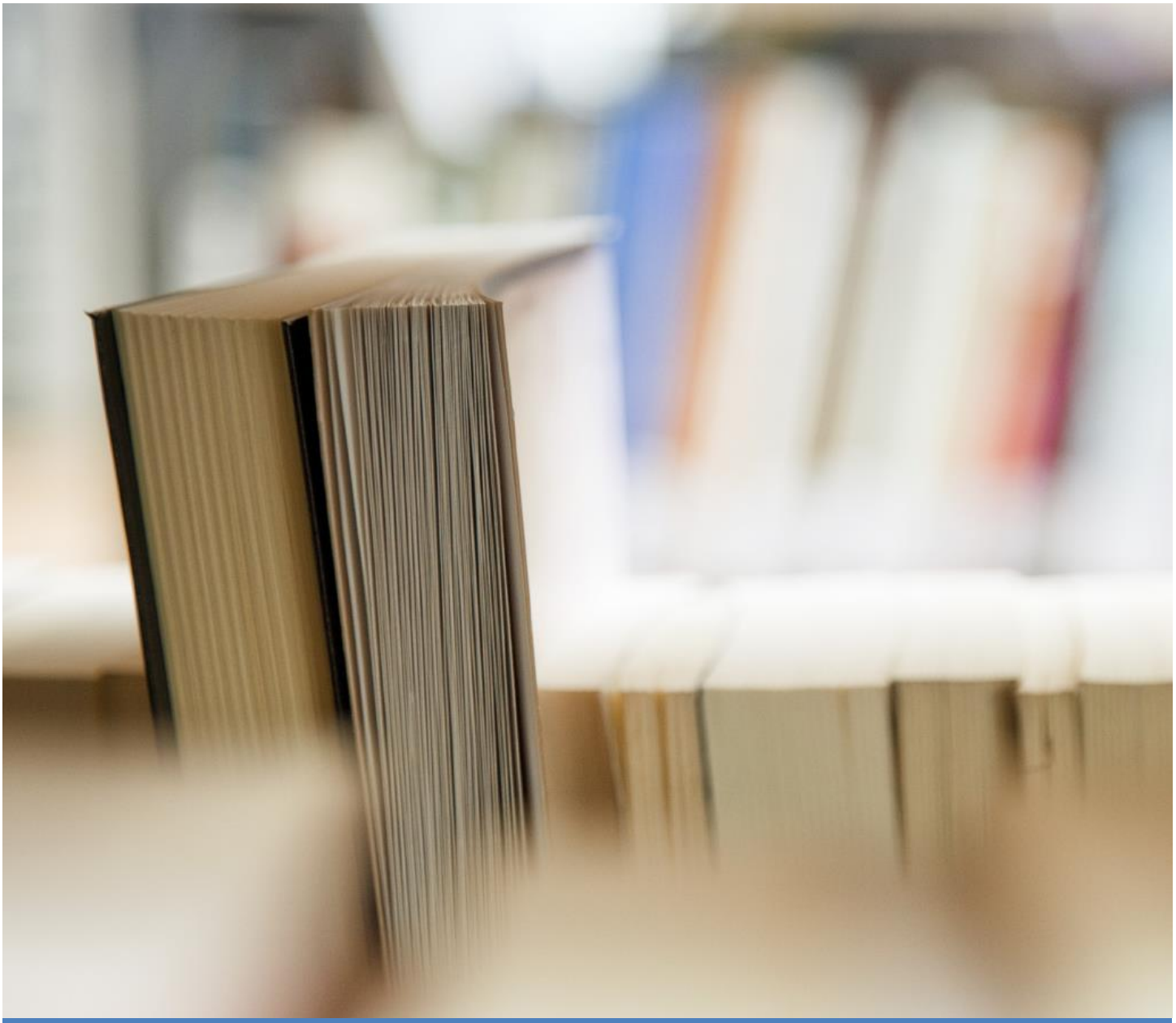


Build a Docker Jenkins Pipeline to Implement CI/CD Workflow

Project 3

Vignesh Dharmaraj

7/22/21

PG DO - DevOps Certification Training

Contents

1. Introduction to the Project

Objective: Demonstrate the continuous integration and delivery by building a Docker Jenkins Pipeline.

Solution build should demonstrate below capabilities:

- Availability of the application and its versions in the GitHub
    - Track their versions every time a code is committed to the repository
- Create a Docker Jenkins Pipeline that will create a Docker image from the Dockerfile and host it on Docker Hub
- It should also pull the Docker image and run it as a Docker container
- Build the Docker Jenkins Pipeline to demonstrate the continuous integration and continuous delivery workflow

Project goal is to deliver the software product frequently to the production with high-end quality.

Tools required: Docker, Docker Hub, GitHub, Git, Linux (Ubuntu), Jenkins

2. Installation of pre-requisites/tools

In this section we can see how the required tools are installed to execute the project

Note: This project is delivered in the "DevOps in AWS" Lab provided by SimpliLearn, so most of the tools may be already installed including the Linux (Ubuntu)

2.1 Installing Git

Step 1: Verifying the Git installation

- Use the following command to check the version of Git:

  git --version

```
vikidvggmail@ip-172-31-29-62:~$ git --version
git version 2.7.4
```

Note: Execute **Step 2** in case you don't get any results for **git --version** command.

Step 2: Installing the latest version of Git

- Execute the following commands on the terminal to install Git:

  sudo apt-get update

  sudo apt-get install git

```
etadata [130 kB]
Get:23 http://security.ubuntu.com/ubuntu xenial-security/multiverse amd64 DEP-11
 Metadata [2,468 B]
Get:25 http://ppa.launchpad.net/ansible/ansible/ubuntu xenial/main amd64 Package
s [696 B]
Fetched 1,206 kB in 1s (954 kB/s)
Reading package lists... Done
W: http://repo.zabbix.com/zabbix/3.0/ubuntu/dists/trusty/InRelease: Signature by
 key FBABD5FB20255ECAB22EE194D13D58E479EA5ED4 uses weak digest algorithm (SHA1)
vikidvggmail@ip-172-31-29-62:~$ sudo apt-get install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
git is already the newest version (1:2.7.4-0ubuntu1.10).
0 upgraded, 0 newly installed, 0 to remove and 85 not upgraded.
```

2.2 Creating GitHub Account

Make sure you have a Github Account available. If not, please create one using the given link.
https://github.com/join?ref_cta=Sign+up&ref_loc=header+logged+out&ref_page=%2F&source=header-home

2.3 Setting up Jenkins

Step 1: Downloading the Java Runtime Environment

1.1 Open the terminal.

1.2 Run ***sudo apt-get update*** to update the package lists.

1.3 Run ***sudo apt-get install openjdk-8-jdk*** to install the Java Runtime Environment.

1.4 Run ***java -version*** to verify the installation. It will print the JDK version as shown below:

```
vikidvggmail@ip-172-31-29-62:~$ java -version
openjdk version "1.8.0_282"
OpenJDK Runtime Environment (build 1.8.0_282-8u282-b08-0ubuntu1~16.04-b08)
OpenJDK 64-Bit Server VM (build 25.282-b08, mixed mode)
```

Step 2: Downloading and installing the Jenkins app

1.1 Open the terminal.

1.2 Run ***wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -*** to install Jenkins.

```
susmitaadhyapak@susmitaadhyapak:~$ wget -q -O - https://pkg.jenkins.io/debian/je
nkins.io.key | sudo apt-key add -
```

2.3 Run ***sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'*** command.

```
susmitaadhyapak@susmitaadhyapak:~$  sudo sh -c 'echo deb http://pkg.jenkins.io/d
ebian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
```

2.4 Run ***sudo apt-get update***

2.5 Run ***sudo apt-get install jenkins*** to install Jenkins.

```
susmitaadhyapak@susmitaadhyapak:~$ sudo apt-get install jenkins
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  fonts-lato javascript-common jruby libbytelist-java libhawtjni-runtime-java
  libheadius-options-java libinvokebinder-java libjansi-java
  libjansi-native-java libjcodings-java libjffi-java libjffi-jni
  libjnr-constants-java libjnr-enxio-java libjnr-ffi-java libjnr-netdb-java
  libjnr-posix-java libjnr-unixsocket-java libjnr-x86asm-java
  libjoda-time-java libjruby-joni-java libjs-jquery libjzlib-java liblua5.2-0
  libreadline7 libssl1.0.2 libtcl8.6 libunsafe-mock-java libyaml-snake-java
  libyecht-java nailgun rake vim-gui-common
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  daemon
The following NEW packages will be installed:
  daemon jenkins
0 upgraded, 2 newly installed, 0 to remove and 306 not upgraded.
Need to get 70.6 MB of archives.
After this operation, 71.2 MB of additional disk space will be used.
```

2.6 Run ***sudo service jenkins status*** to check the status of the installation. Once you verify the status as active, you can press ***Ctrl+z*** to exit from the process.

```
vikidvggmail@ip-172-31-29-62:~$ sudo service jenkins status
● jenkins.service - LSB: Start Jenkins at boot time
   Loaded: loaded (/etc/init.d/jenkins; bad; vendor preset: enabled)
   Active: active (exited) since Thu 2021-07-22 09:59:50 UTC; 30min ago
     Docs: man:systemd-sysv-generator(8)
  Process: 1563 ExecStart=/etc/init.d/jenkins start (code=exited, status=0/SUCCE
    Tasks: 0
   Memory: 0B
      CPU: 0

Jul 22 09:59:48 ip-172-31-29-62 systemd[1]: Starting LSB: Start Jenkins at boot
Jul 22 09:59:49 ip-172-31-29-62 jenkins[1563]: Correct java version found
Jul 22 09:59:49 ip-172-31-29-62 jenkins[1563]:  * Starting Jenkins Automation Se
Jul 22 09:59:49 ip-172-31-29-62 su[1745]: Successful su for jenkins by root
Jul 22 09:59:49 ip-172-31-29-62 su[1745]: + ??? root:jenkins
Jul 22 09:59:49 ip-172-31-29-62 su[1745]: pam_unix(su:session): session opened f
Jul 22 09:59:50 ip-172-31-29-62 jenkins[1563]:    ...done.
Jul 22 09:59:50 ip-172-31-29-62 systemd[1]: Started LSB: Start Jenkins at boot t
lines 1-17/17 (END)
```

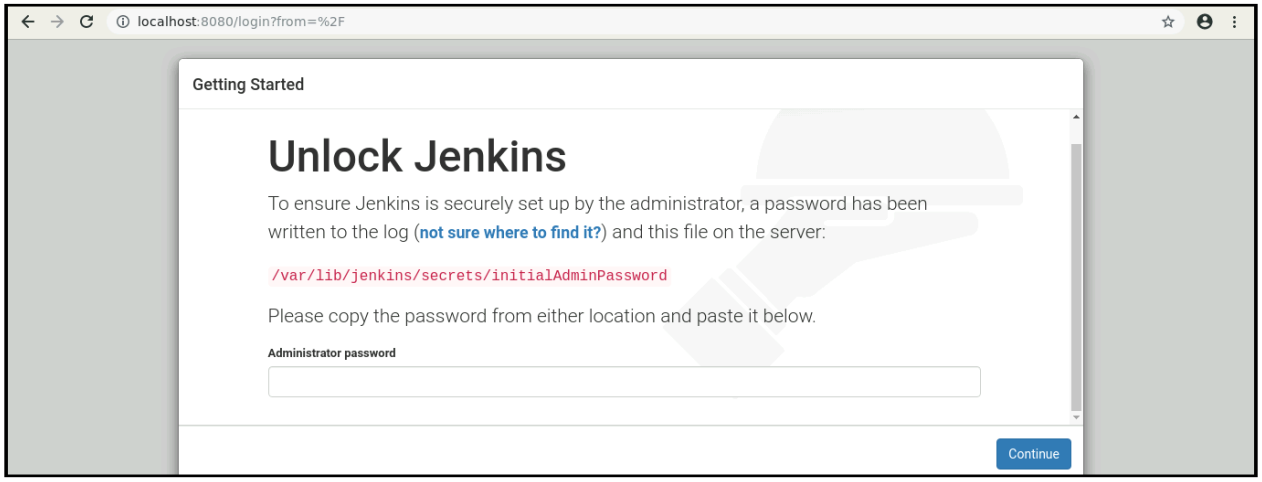
2.7 Run the following commands to start Jenkins.

```
sudo systemctl start jenkins
sudo systemctl status jenkins
```

```
susmitaadhyapak@susmitaadhyapak:~$ sudo systemctl start jenkins
susmitaadhyapak@susmitaadhyapak:~$ sudo systemctl status jenkins
● jenkins.service - LSB: Start Jenkins at boot time
   Loaded: loaded (/etc/init.d/jenkins; bad; vendor preset: enabled)
   Active: active (exited) since Tue 2021-03-23 08:02:03 UTC; 4min 2s ago
     Docs: man:systemd-sysv-generator(8)

Mar 23 08:02:01 susmitaadhyapak systemd[1]: Starting LSB: Start Jenkins at boot
Mar 23 08:02:01 susmitaadhyapak jenkins[5672]: Correct java version found
Mar 23 08:02:01 susmitaadhyapak jenkins[5672]:  * Starting Jenkins Automation Se
Mar 23 08:02:01 susmitaadhyapak su[5737]: Successful su for jenkins by root
Mar 23 08:02:01 susmitaadhyapak su[5737]: + ??? root:jenkins
Mar 23 08:02:01 susmitaadhyapak su[5737]: pam_unix(su:session): session opened f
Mar 23 08:02:03 susmitaadhyapak jenkins[5672]:    ...done.
Mar 23 08:02:03 susmitaadhyapak systemd[1]: Started LSB: Start Jenkins at boot t
Mar 23 08:05:53 susmitaadhyapak systemd[1]: Started LSB: Start Jenkins at boot t
lines 1-14/14 (END)
```

2.8 Open **localhost:8080** in the browser, and you will need to enter the initial password.



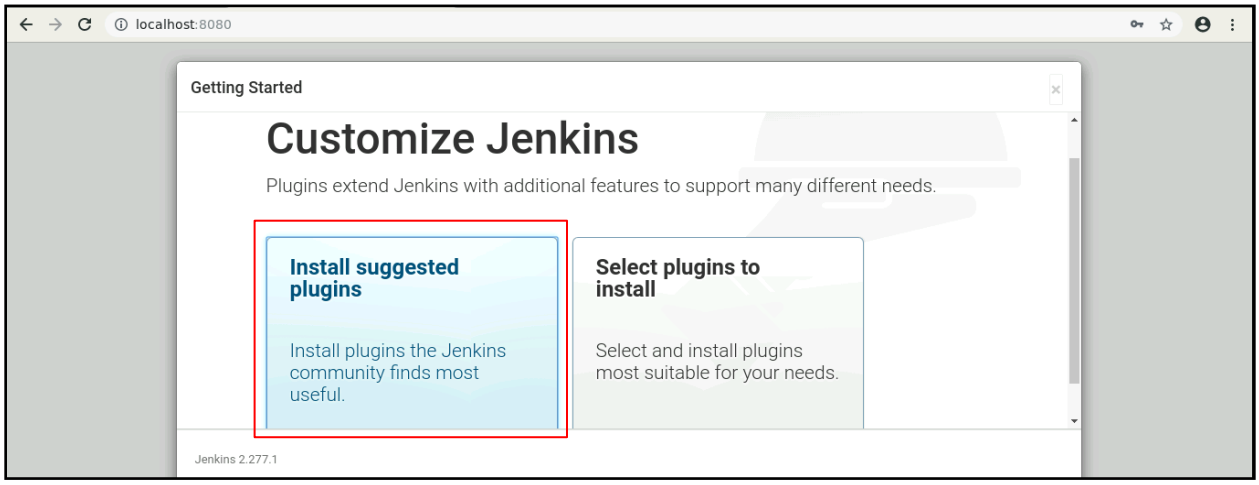
2.9 In your terminal run the following command:

sudo cat /var/lib/jenkins/secrets/initialAdminPassword

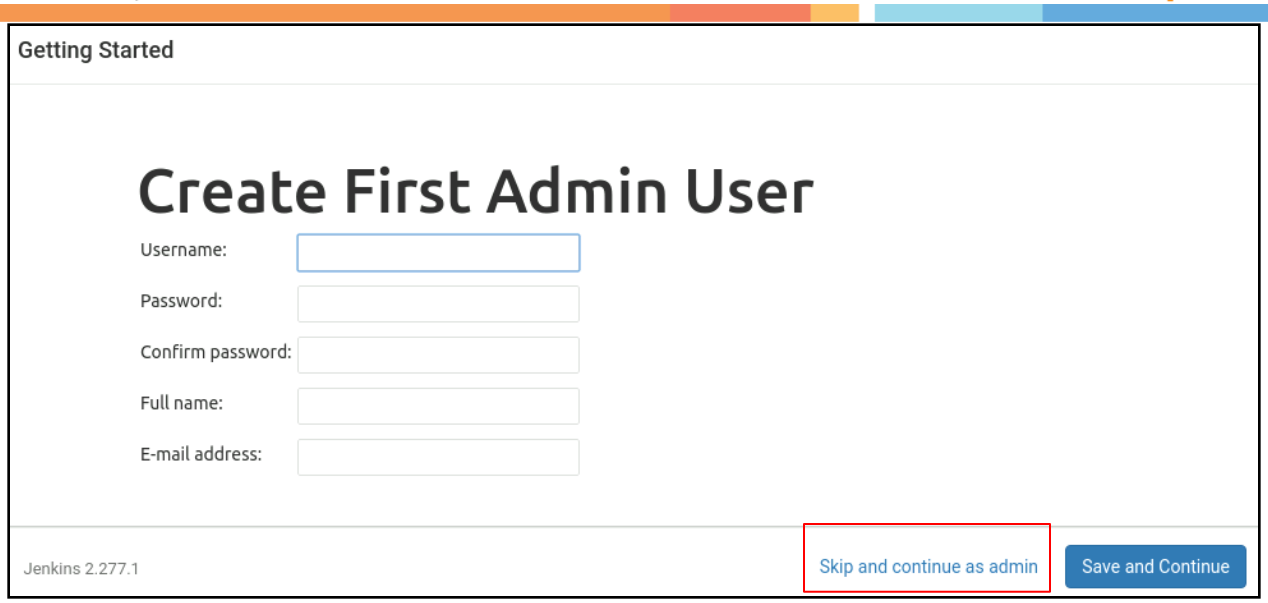
```
susmitaadhyapak@susmitaadhyapak:~$ sudo cat /var/lib/jenkins/secrets/initialAdmi
nPassword
876821d4689a453c87c48116a59a001b
susmitaadhyapak@susmitaadhyapak:~$
```

2.10 Copy this password and paste it on your Jenkins page in the browser.

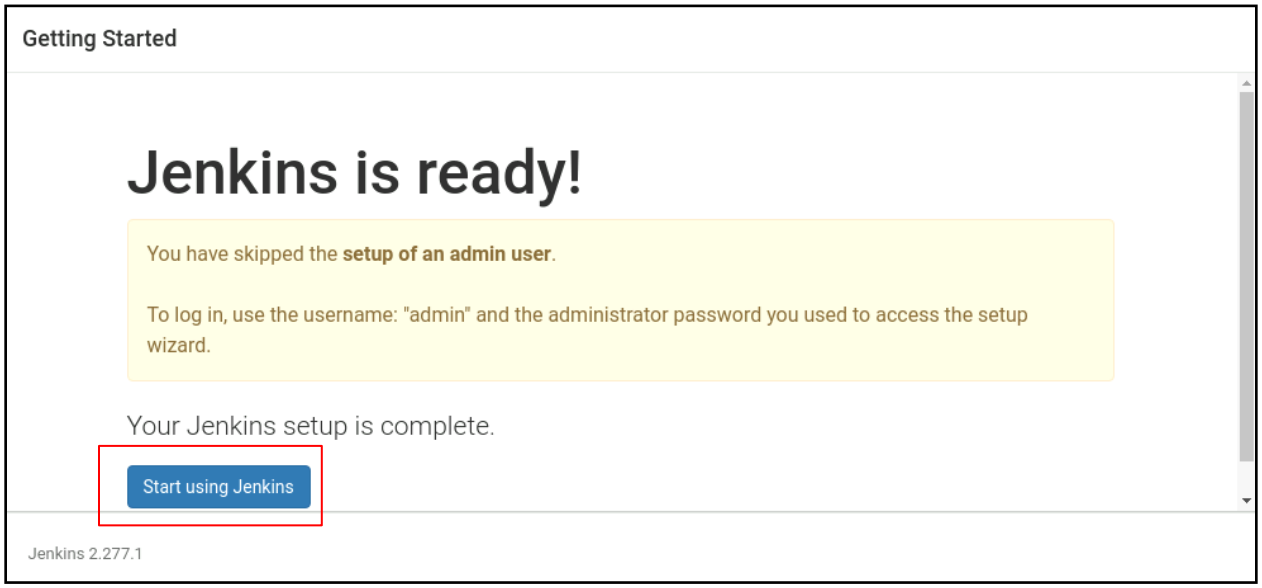
2.11 Now, click on **Install the suggested plugins**.



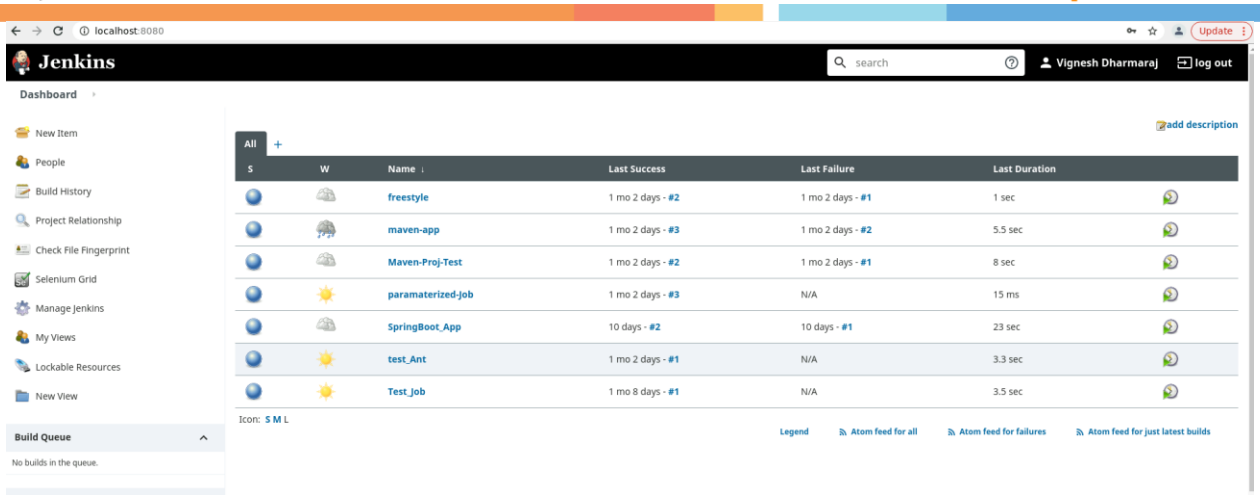
2.12 You can either create an admin user or skip and continue as admin. Select **Skip and continue as admin**.



2.13 In the Instance configuration page, click on the **Start using Jenkins** button.



2.14 Now, you can work with Jenkins as shown in the screenshot below.



2.4 Docker Community Edition Installation

Step 1: Install the Docker CE from Docker Repository

1.1 Use the following command to update the apt package:

sudo apt-get update

```
labsuser@ip-172-31-29-216:~$ sudo apt-get update
Hit:1 http://us-west-2.ec2.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://us-west-2.ec2.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://us-west-2.ec2.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Hit:4 https://download.docker.com/linux/ubuntu bionic InRelease

Hit:5 http://dl.google.com/linux/chrome/deb stable InRelease

Hit:6 http://packages.microsoft.com/repos/vscode stable InRelease
```

1.2 Use the following command to install packages to allow the apt to use a repository over HTTPS

***sudo apt-get install ***

***apt-transport-https ***

***ca-certificates ***

***curl ***

***gnupg ***

lsb-release

```
[redacted]~$ sudo apt-get install \
[redacted]
> ca-certificates \
> curl \
> gnupg \
> lsb-release
Reading package lists... Done
Building dependency tree
Reading state information... Done
apt-transport-https is already the newest version (1.2.32ubuntu0.2).
ca-certificates is already the newest version (20210119~16.04.1).
curl is already the newest version (7.47.0-1ubuntu2.19).
gnupg is already the newest version (1.4.20-1ubuntu3.3).
lsb-release is already the newest version (9.20160110ubuntu0.2).
0 upgraded, 0 newly installed, 0 to remove and 52 not upgraded.
[redacted]~$
```

1.3 Use the following curl command to add Docker's official GPG key:

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

```
rishabhpathaksi@ip-172-31-66-101:~$  curl -fsSL https://download.docker.com/linu
x/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.
gpg
```

1.4 Use the following command to set up a stable repository:

***echo ***

***"deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu ***

$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

```
echo \
>   "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] h
ttps://download.docker.com/linux/ubuntu \
>
> $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /d
ev/null
```

1.5 Use the following commands to install the latest version of Docker CE and check the version:

sudo apt-get install docker-ce

docker --version

```
vikidvggmail@ip-172-31-29-62:~$ docker --version
Docker version 19.03.14, build 5eb3275d40
```

Step 2: Verify the correctly installed Docker engine

2.1 Use the following command to verify the Docker engine:

sudo docker run hello-world

```
vikidvggmail@ip-172-31-29-62:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
b8dfde127a29: Pull complete
Digest: sha256:df5f5184104426b65967e016ff2ac0bfcd44ad7899ca3bbcf8e44e4461491a9e
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
```

Step 3: Create a Docker Hub account to store or get images from remote repository

Go to https://hub.docker.com/ and create your own account

3. Execution of the Project

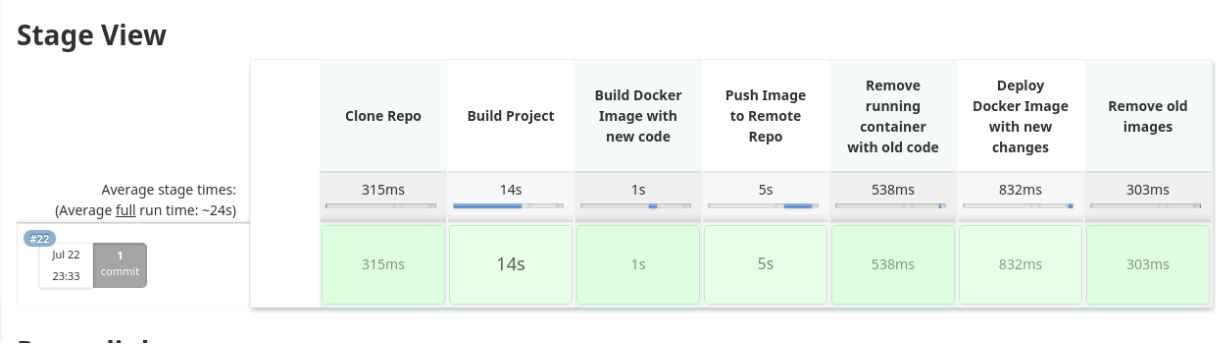
All the necessary code is created in the below GitHub repository, this repo can be cloned to execute the Jenkins pipeline
https://github.com/vdharmaraj/PGDO_Proj3

The Repo contains a sample Spring Boot Application and when the Jenkins pipeline runs, the spring boot application is built with maven tools and it is deployed in OpenJDK docker image.

Any commit made in the repo automatically triggers the Jenkins pipeline, which does all the actions of below:

1) Create build with Maven by using the new code committed
2) Create a new docker image
3) Push the new image to docker hub
4) Remove already running container in the server with old code
5) Deploy the new code with the updated version of image from Docker Hub
6) Remove the outdated images stored locally

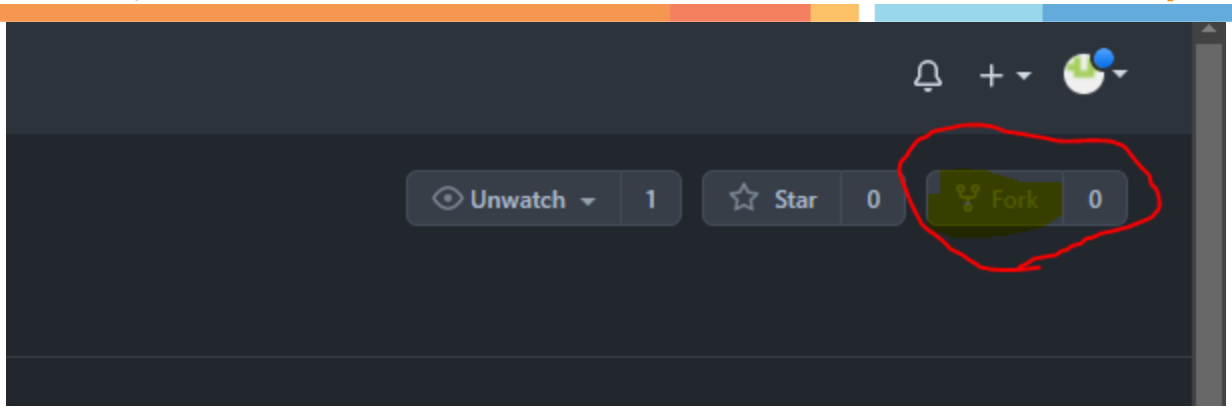
Below are the stages in the Jenkins pipeline to fulfil the objectives of this project:



Step 1:

Fork the Github project

By clicking the Fork button as shown below in github, you can fork the project into your github account



Step 2:

with $git clone https://github.com/vdharmaraj/PGDO_Proj3.git command you can clone the project into your machine and the changes to be made in Jenkinsfile available in the repository

Step 3:
Change the docker account ID in Jenkinsfile

Step 4:

Change the repo URL in the Jenkinsfile , this should be your forked Repo URL

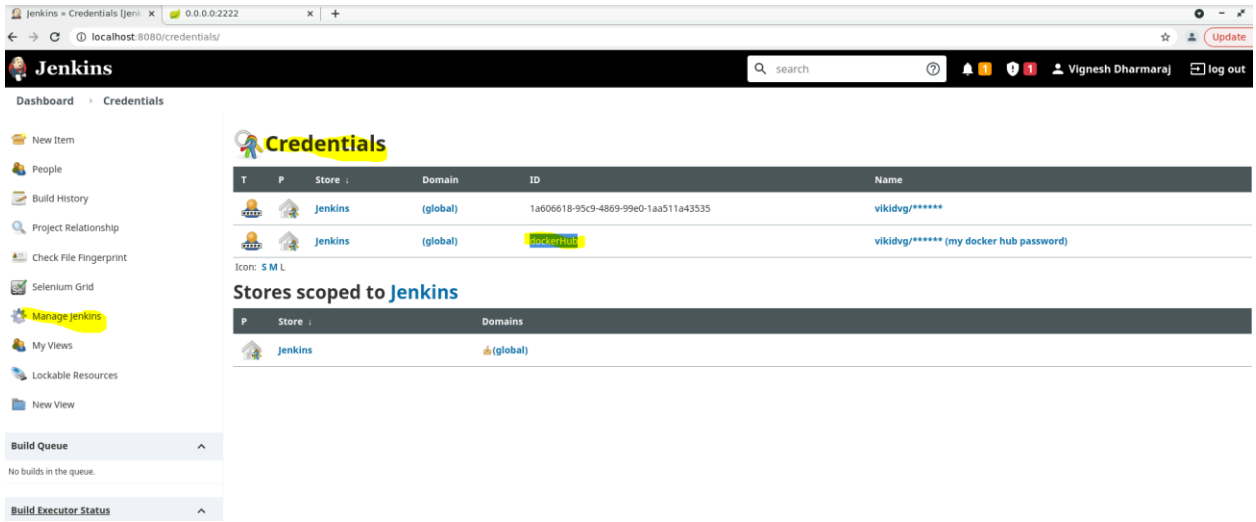
```
16
17      stage('Clone Repo') {
18        // Get some code from a GitHub repository
19        git url:'https://github.com/vdharmaraj/PGDO_Proj3.git',branch:'main' //update your forked repo
20        // Get the Maven tool.
21        // ** NOTE: This 'maven-3.5.2' Maven tool must be configured
22        // **       in the global configuration.
```

Step 5:

Docker Hub Credentials to be maintained in Jenkins server as mentioned in the Jenkinsfile

```
    //push image to remote repository , in your jenkins you ha
  stage('Push Image to Remote Repo'){
      echo "Docker Image Tag Name ---> ${dockerImageTag}"
          docker.withRegistry('', 'dockerHub') {
          dockerImage.push("${env.BUILD_NUMBER}")
          dockerImage.push("latest")
        }
    }
```

In the Jenkins server , go to Manage Jenkins -> Manage Credentials to maintain the Docker hub userID and Password with the same Credential ID ‘dockerHub’ as maintained in Jenkinsfile of the repository



Step 6:

The user **jenkins** needs to be added to the group docker with the below linux command:

sudo usermod -a -G docker Jenkins

Note: If this is not done, then the Jenkins server cannot create docker containers and we will get permission error

You can check if the above command was successful by doing

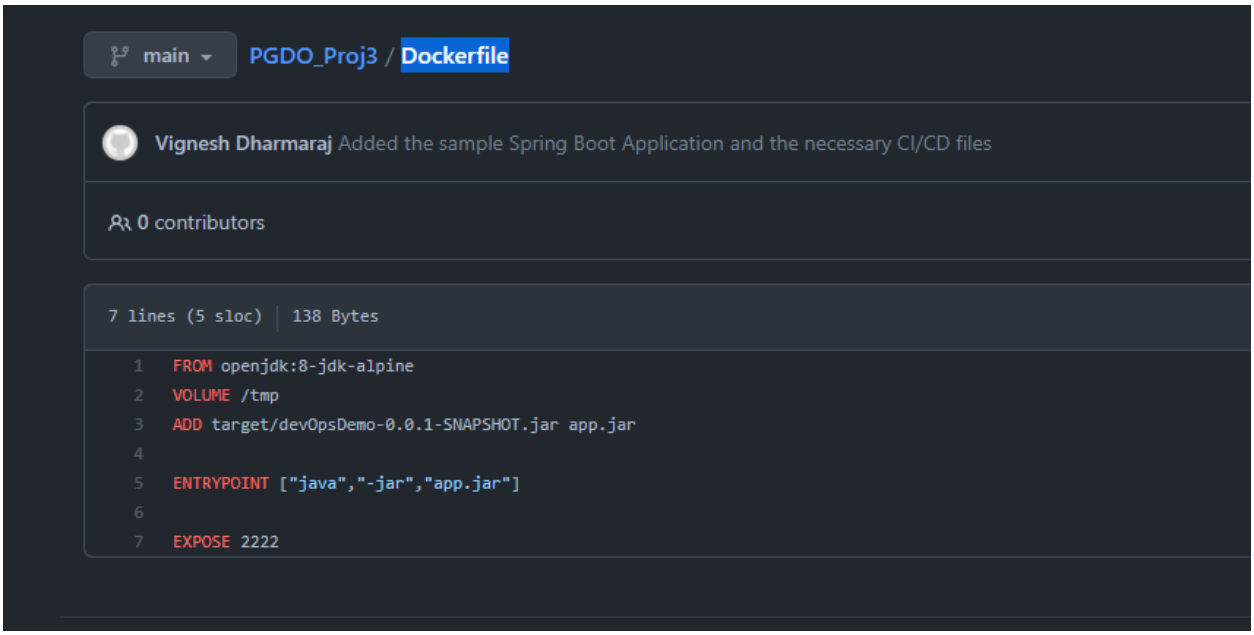
grep docker /etc/group

and you can see something like below:

docker:x:998:[user]

Step 7:
Review the Dockerfile available in the repo, which needs no change from your end.

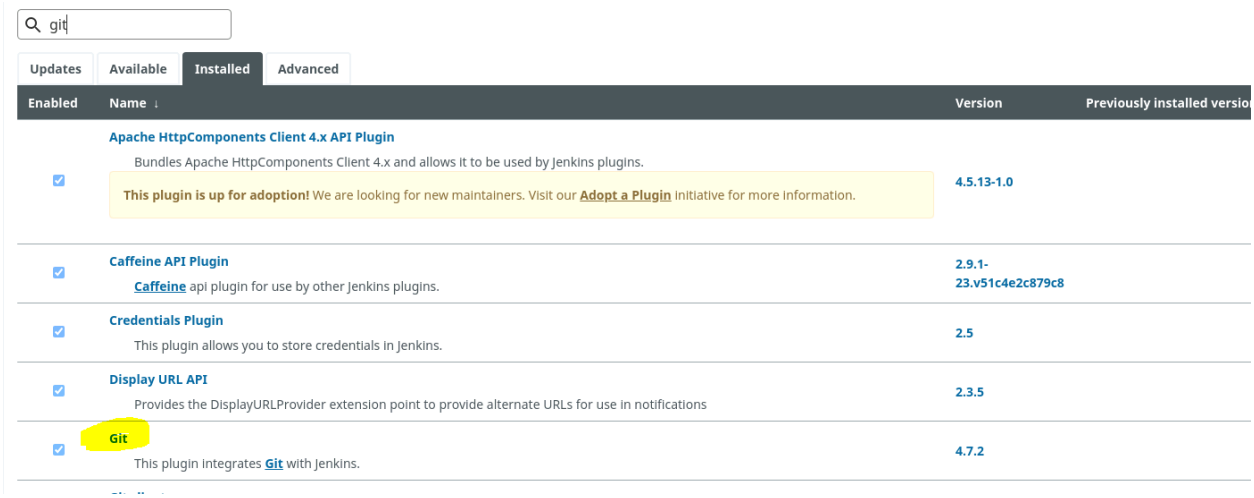
It uses the OpenJDK image where we place our executable JAR file in the required path and expose port 2222

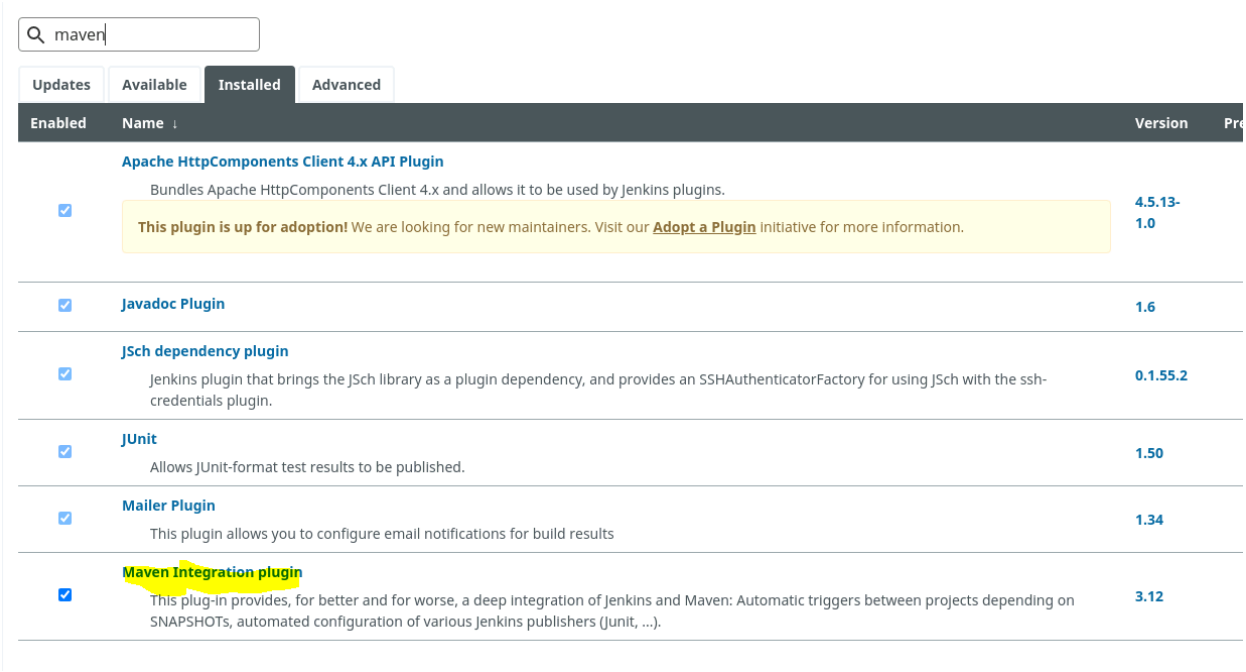


Step 8:

Install required Jenkins plugins if not installed already

Use manage plugins option to install the plugins





Updates | Available | Installed | Advanced

Enabled	Name ↓	Version	Pre
☑	**Authentication Tokens API Plugin** This plugin provides an API for converting credentials into authentication tokens in Jenkins.	1.4	
☑	**Credentials Binding** Allows credentials to be bound to environment variables for use from miscellaneous build steps.	1.25	
☑	**Credentials Plugin** This plugin allows you to store credentials in Jenkins.	2.5	
☑	**Docker Commons Plugin** Provides the common shared functionality for various Docker-related plugins.	1.17	
☑	**Docker Pipeline** Build and use Docker containers from pipelines.	1.26	
☑	**Folders Plugin** This plugin allows users to create "folders" to organize jobs. Users can define custom taxonomies (like by project type, organization type etc). Folders are nestable and you can define views within folders. Maintained by CloudBees, Inc.	6.15	

Step 9:

Global tools configuration in Jenkins is required as below:

Global Tool Configuration

Maven Configuration

Default settings provider

Use default maven settings

Default global settings provider

Use default maven global settings

JDK

JDK installations

Add JDK

JDK

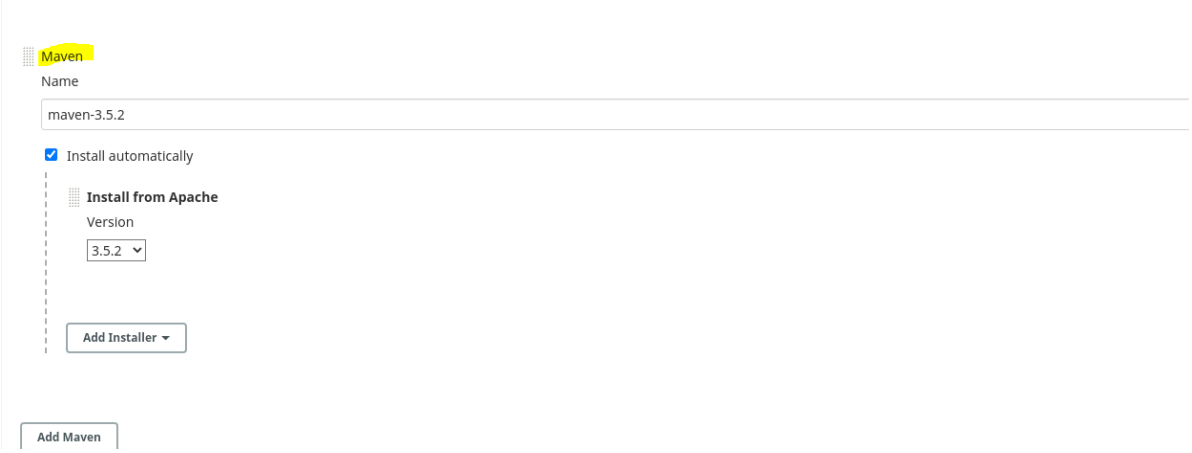
Name

jdk 1.8

JAVA_HOME

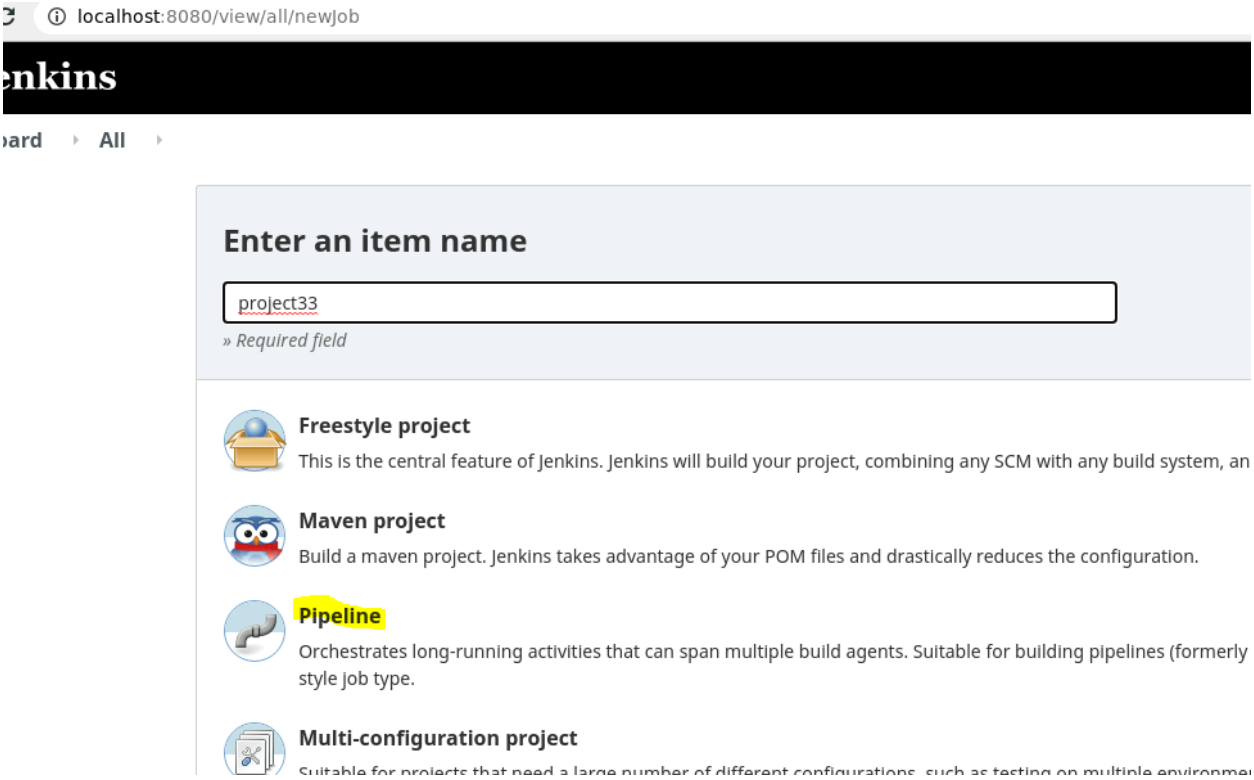
/usr/lib/jvm/java-8-openjdk-amd64/

☐ Install automatically

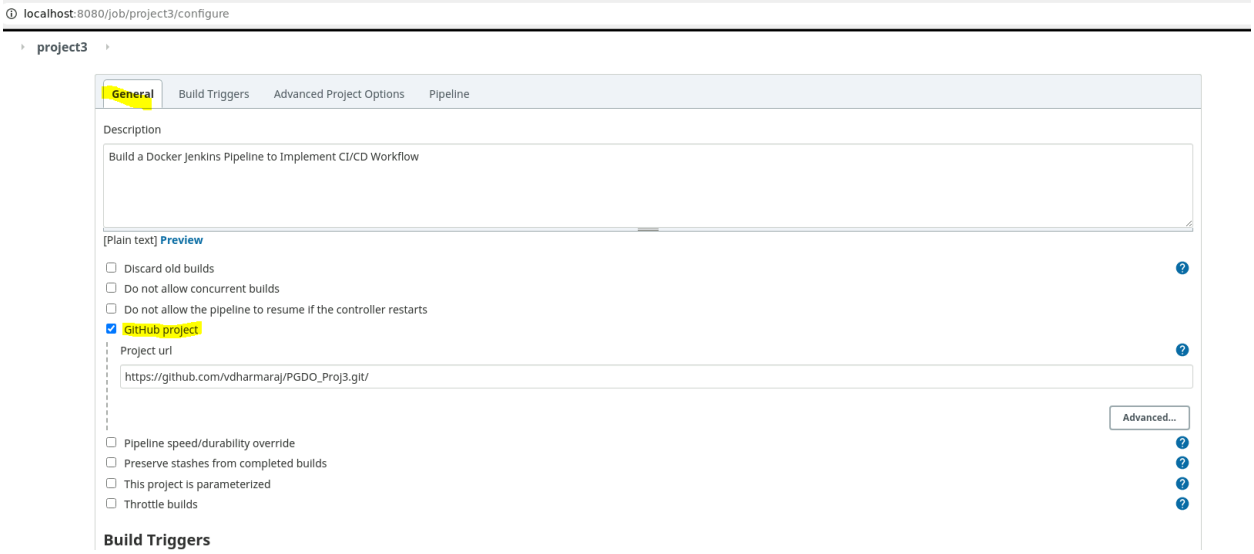


Step 10:
Create a Jenkins pipeline form the Git repository

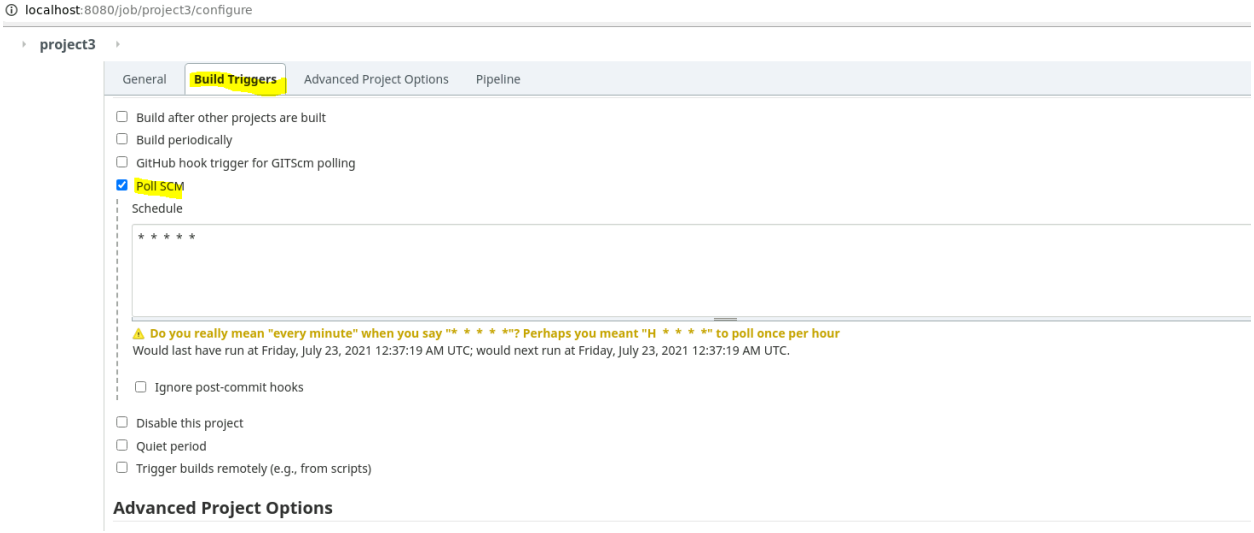
New Pipeline Project has to be created in Jenkins server as below



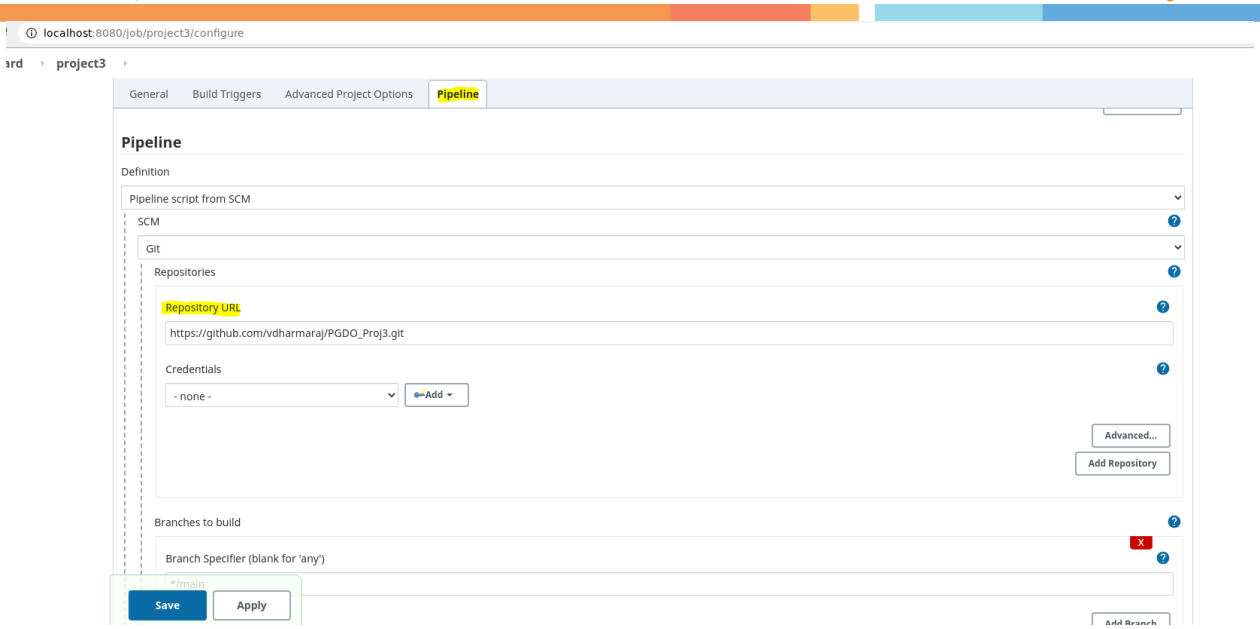
In General Tab, GitHub Project URL to be configured

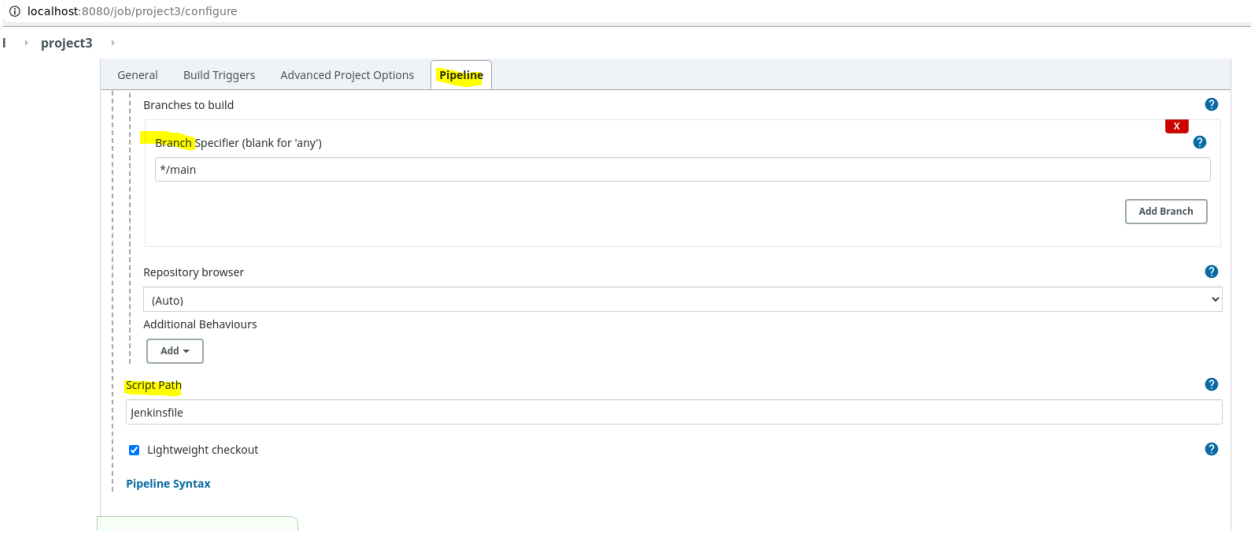


In the build triggers section, Poll SCM to be configured



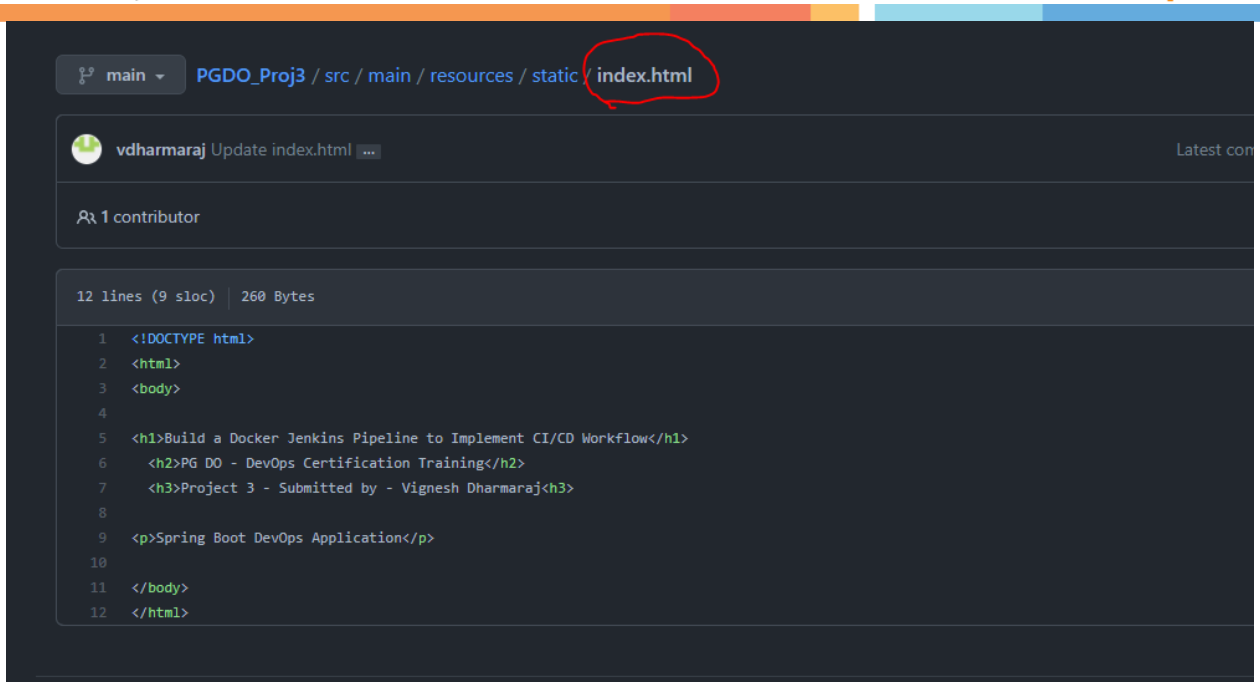
In the pipeline section, our repo needs to be configured





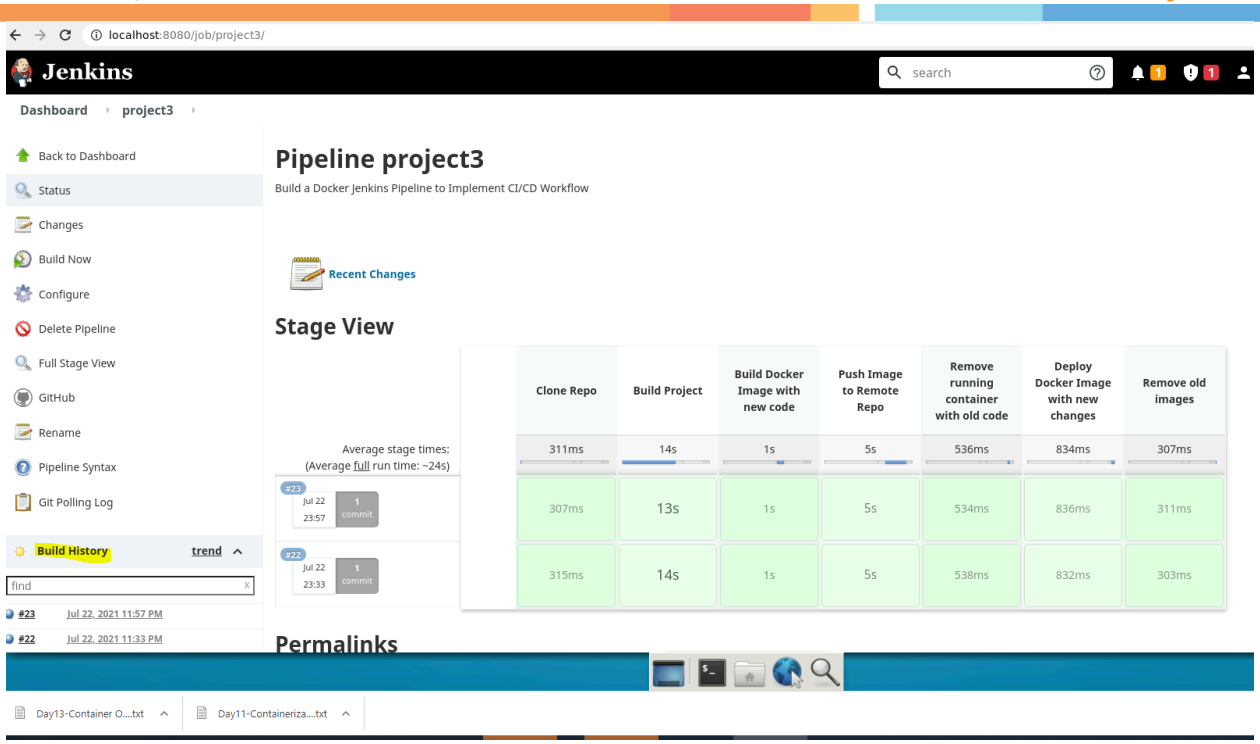
Step 11:

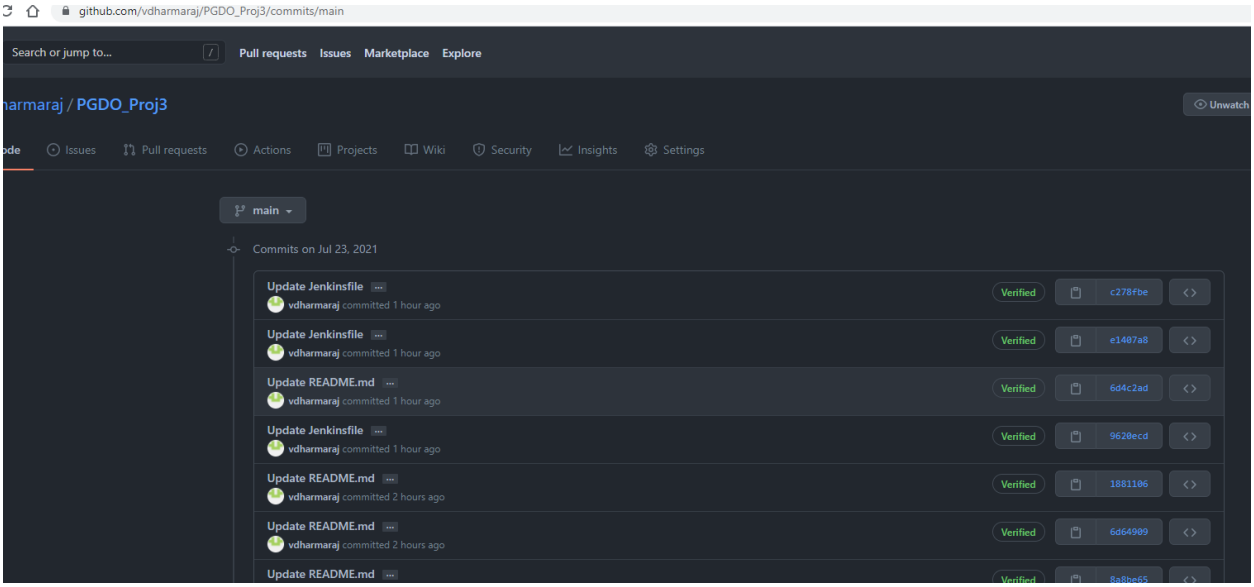
In out Repository the index.html file, which is the mail Spring Boot App source code, this HTML file can be modified and the changes can be committed to the repository which will trigger the Jenkins Poll SCM job and the CI/CD process will be triggered to deploy the new application changes



4. Project Results

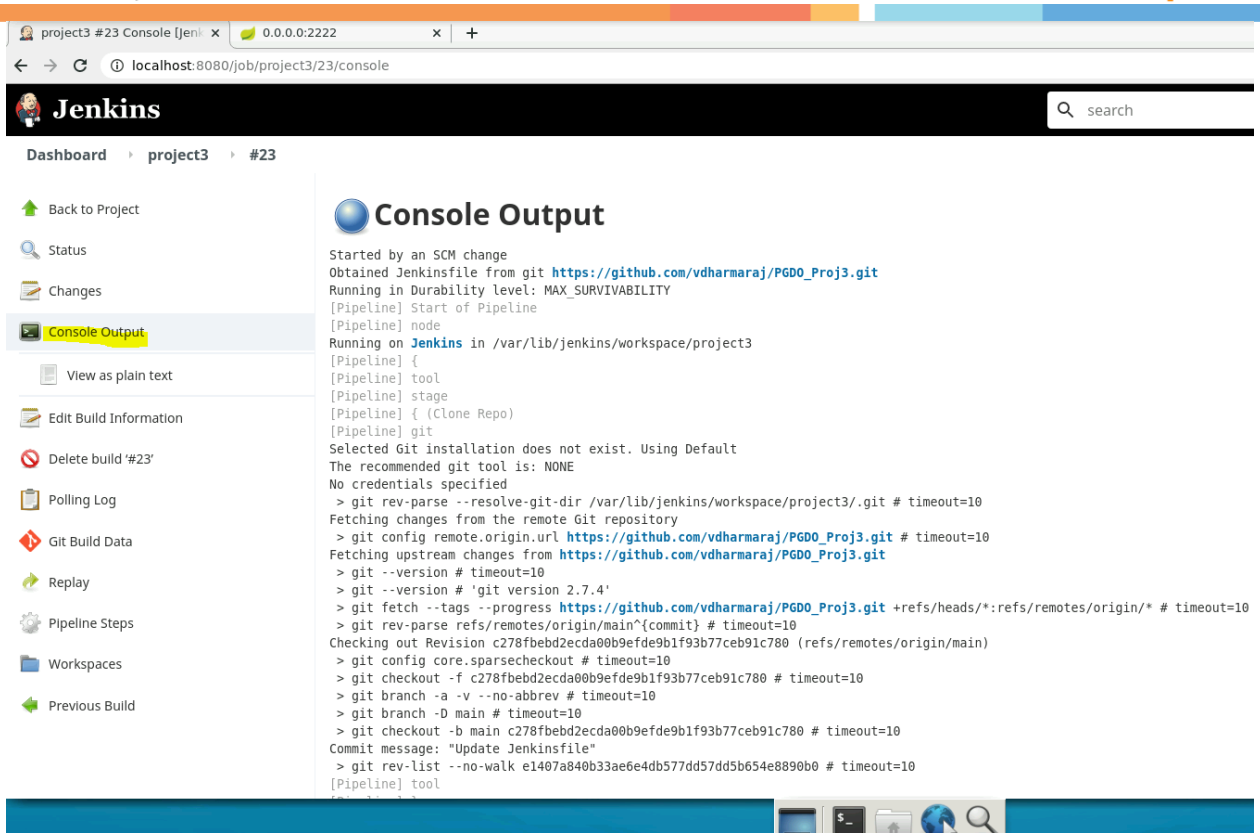
Result 1:
Jenkins job will be triggered automatically corresponding to the repository commit version, this can be seen in the Job build history





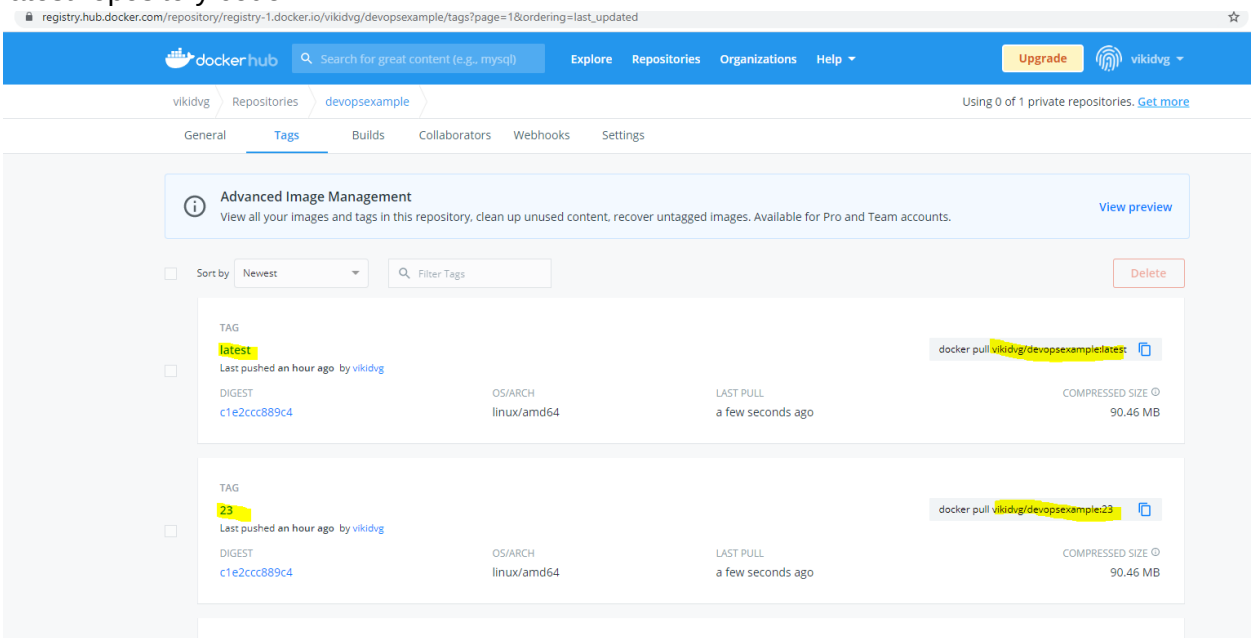
Result 2:

Every build and its console output can be seen to check the status of every Stage in the Jenkins pipeline



Result 3:

You can check in the Docker Hub for the latest image version published, which is having the latest repository code



Result 4:

You can see that the required application container is started and it is running with the port 2222 exposed, below command can be used to check this

$docker ps

```
vikidvggmail@ip-172-31-29-62:~$ docker ps
CONTAINER ID        IMAGE                     COMMAND                  CREATED             STATUS              PORTS                    NAMES
e94c4ebc19ba        vikidvg/devopsexample:23  "java -jar app.jar"      55 minutes ago      Up 55 minutes       0.0.0.0:2222->2222/tcp   devopsexample
5f53c4ece041        7f92d556d4ff              "/usr/bin/launch.sh"     4 hours ago         Up 4 hours                                   k8s_weave-npc_weave-net-8glwb
0d-4370-b411-c33b76ab3e44_0
e5b43c870a95        df29c0a4002c              "/home/weave/launch.…"   4 hours ago         Up 4 hours                                   k8s_weave_weave-net-8glwb_kub
370-b411-c33b76ab3e44_0
cc2e32efecf2        6be0dc1302e3              "kube-scheduler --au…"   4 hours ago         Up 4 hours                                   k8s_kube-scheduler_kube-sched
```

Above result also confirms that, always there is only one container with name 'devopsexample' is running at any given point in time.

New version of changes does not create a new container, old container is first deleted and then the container with latest code version is created.

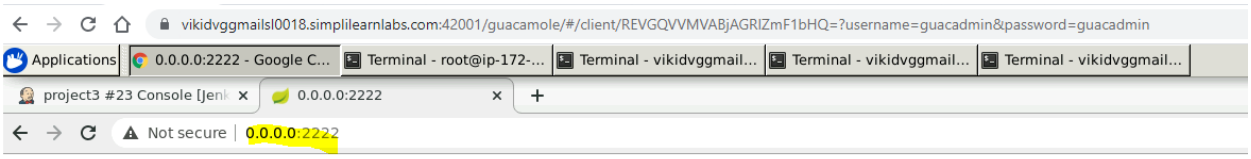
With below command, we can also see the related version of the image in local system

```
vikidvggmail@ip-172-31-29-62:~$ docker images
REPOSITORY                                  TAG                 IMAGE ID            CREATED             SIZE
vikidvg/devopsexample                       23                  520af0e38a80        57 minutes ago      129MB
```

Result 5:

We can browse our Spring Boot application running and exposed in the URL http://0.0.0.0:2222/

Any new changes will be deployed automatically with our CI/CD pipeline with jenkins



Build a Docker Jenkins Pipeline to Implement CI/CD Workflow

PG DO - DevOps Certification Training

Project 3 - Submitted by - Vignesh Dharmaraj

Spring Boot DevOps Application

5. Conclusion

Results shown is the previous section is the evidence that the Docker Jenkins Pipeline is implemented, enabling the CI/CD workflow.

Without any human intervention, incremental code can be delivered to production with Quality, Governance and Agility.