



Build a Docker Jenkins Pipeline to Implement CI/CD Workflow

Project 3

Contents

1. Introduction to the Project	2
2. Installation of pre-requisites/tools	2
2.1 Installing Git	3
2.2 Creating GitHub Account	3
2.3 Setting up Jenkins	4
2.4 Docker Community Edition Installation	9
3. Execution of the Project	13
4. Project Results	22
5. Conclusion	26

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`

```
vikidvgg@mail@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`

```
Get:22 http://security.ubuntu.com/ubuntu xenial-security/universe amd64 DEP-11 Metadata [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 Packages [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)
vikidvgg@mail@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/jenkins.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/debian-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
 libyeicht-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/SUCCESS)
    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 for
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)
```

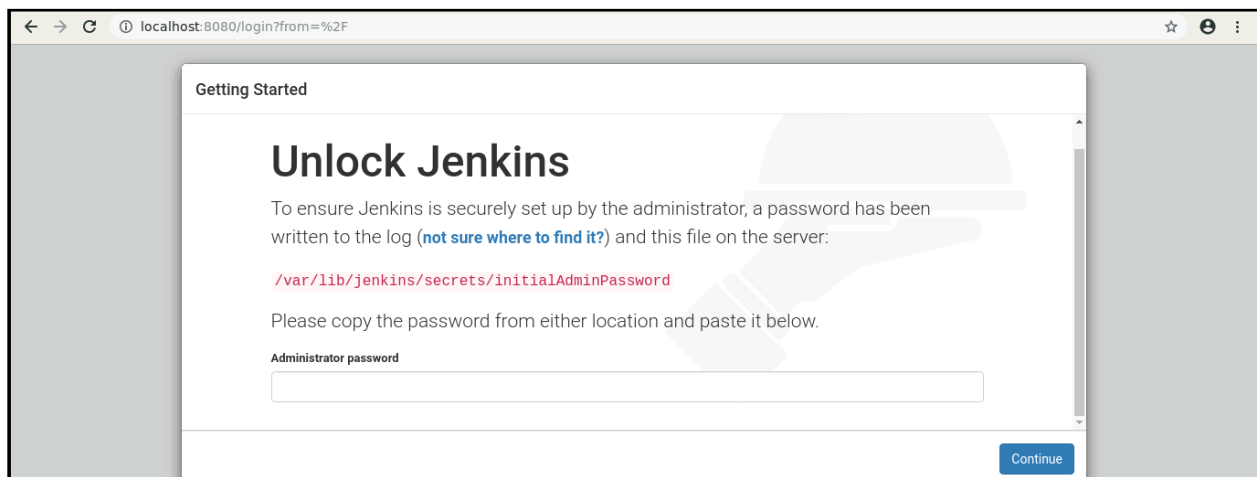
2.7 Run the following commands to start Jenkins.

```
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

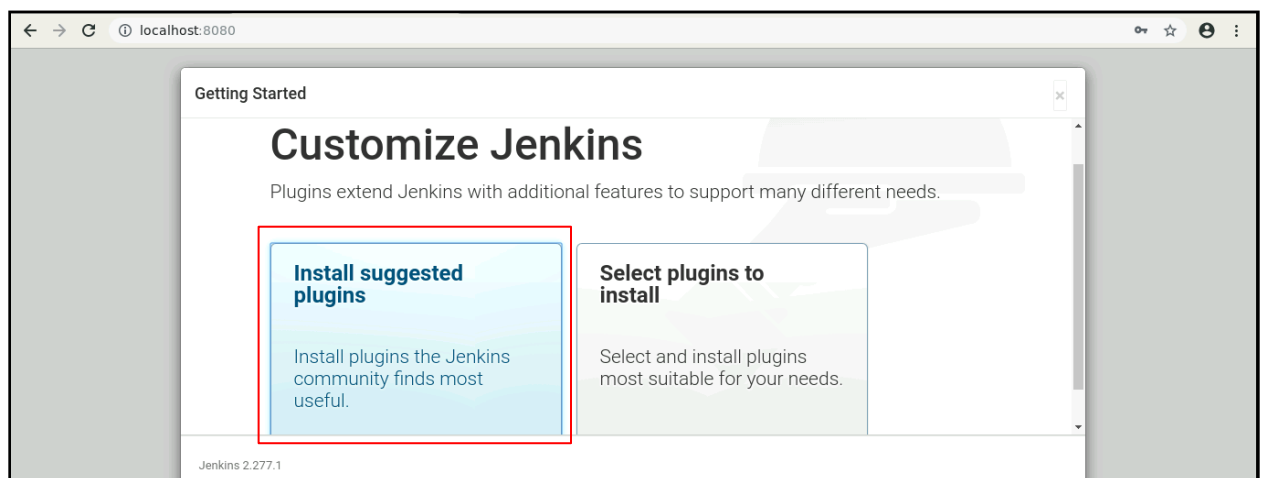
```
susmitaadhyapak@susmitaadhyapak:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
876821d4689a453c87c48116a59a001b
susmitaadhyapak@susmitaadhyapak:~$
```


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



The screenshot shows the 'Getting Started' section of the Jenkins installation page. The main heading is 'Unlock Jenkins'. Below it, a message states: 'To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server: /var/lib/jenkins/secrets/initialAdminPassword'. It then asks the user to 'Please copy the password from either location and paste it below.' There is a text input field labeled 'Administrator password' with a red border. A 'Continue' button is located at the bottom right.

2.11 Now, click on Install the suggested plugins.



The screenshot shows the 'Customize Jenkins' screen in a browser window. The browser address bar shows 'localhost:8080'. The page title is 'Customize Jenkins'. Below the title, it says 'Plugins extend Jenkins with additional features to support many different needs.' There are two main options: 'Install suggested plugins' (highlighted with a red border) and 'Select plugins to install'. The 'Install suggested plugins' option includes the text 'Install plugins the Jenkins community finds most useful.' The 'Select plugins to install' option includes the text 'Select and install plugins most suitable for your needs.' The Jenkins version 'Jenkins 2.277.1' is visible at the bottom left.

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

Getting Started

Create First Admin User

Username:

Password:

Confirm password:

Full name:

E-mail address:

Jenkins 2.277.1

[Skip and continue as admin](#) [Save and Continue](#)

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

Getting Started

Jenkins is ready!

You have skipped the **setup of an admin user**.

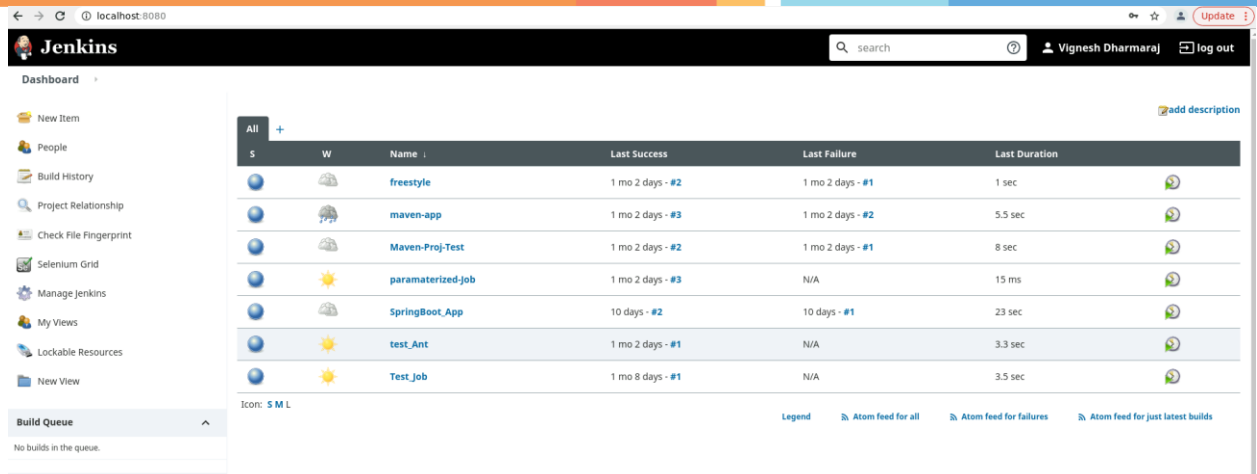
To log in, use the username: "admin" and the administrator password you used to access the setup wizard.

Your Jenkins setup is complete.

[Start using Jenkins](#)

Jenkins 2.277.1

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



The screenshot shows the Jenkins Dashboard interface. On the left is a sidebar with navigation links like 'New Item', 'People', 'Build History', etc. The main area displays a table of builds. The table has columns for status (S), icon (W), name, last success, last failure, and last duration. Below the table is a 'Build Queue' section showing 'No builds in the queue.'.

S	W	Name	Last Success	Last Failure	Last Duration
		freestyle	1 mo 2 days - #2	1 mo 2 days - #1	1 sec
		maven-app	1 mo 2 days - #2	1 mo 2 days - #2	5.5 sec
		Maven-Proj-Test	1 mo 2 days - #2	1 mo 2 days - #1	8 sec
		paramaterized-job	1 mo 2 days - #3	N/A	15 ms
		SpringBoot_App	10 days - #2	10 days - #1	23 sec
		test_Ant	1 mo 2 days - #1	N/A	3.3 sec
		Test_Job	1 mo 8 days - #1	N/A	3.5 sec

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

```
~$ sudo apt-get install \
> 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.
~$
```

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/linux/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

```
vikidvgg@gmail@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

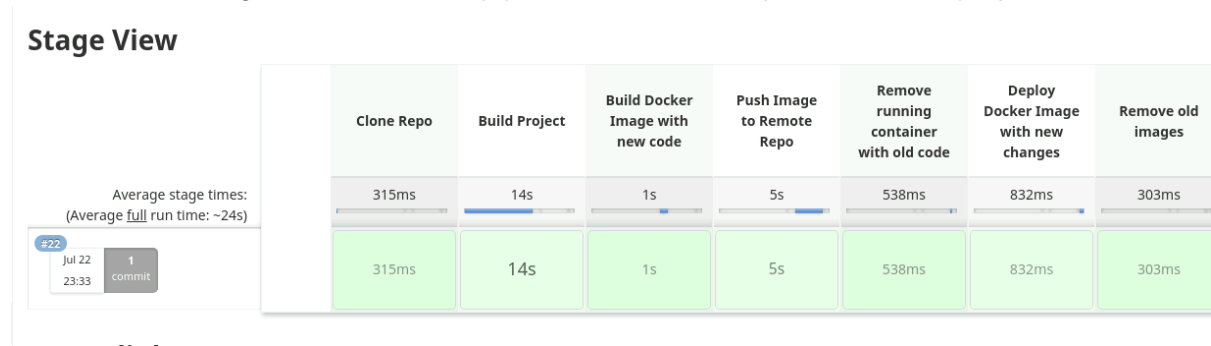
https://github.com/vdharmaraj/PGDO_Proj3

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

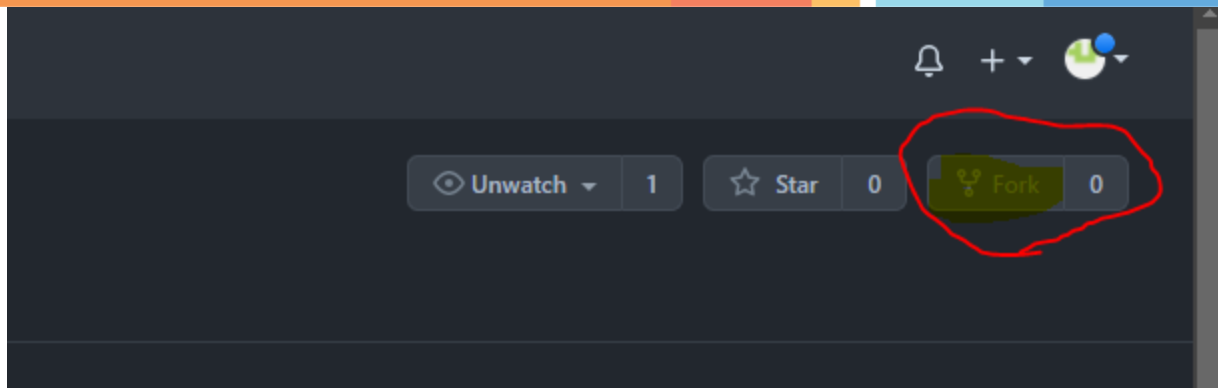
Below are the stages in the Jenkins pipeline to fulfil the objectives of this project:



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

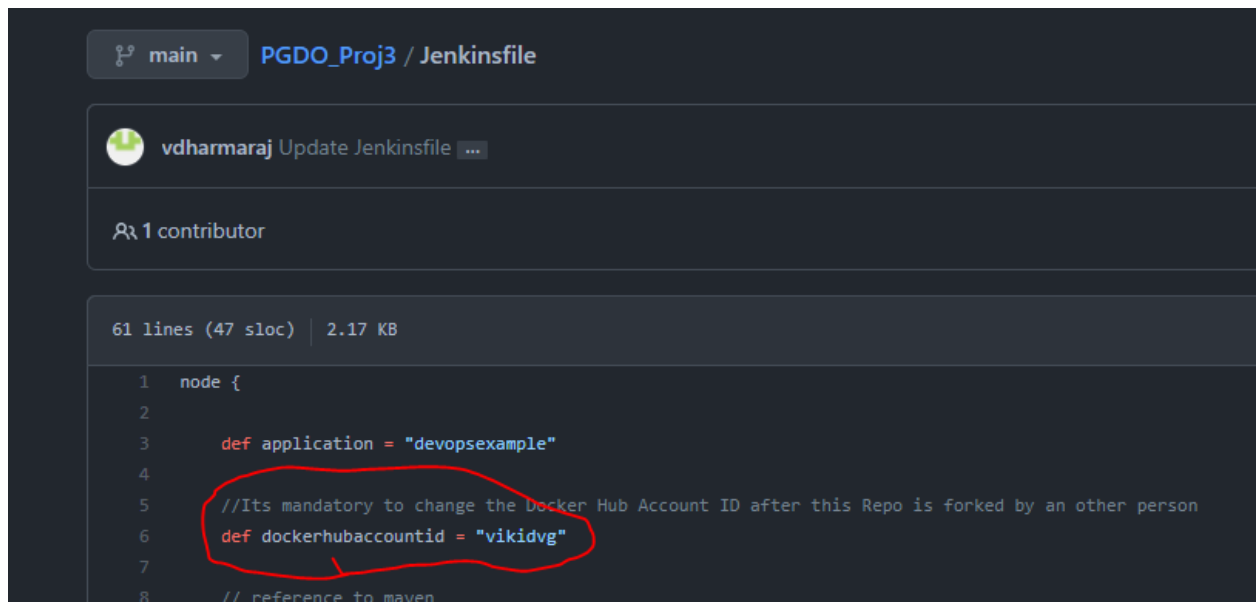


Step 2:

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


```

6
7 stage('Clone Repo') {
8     // Get some code from a GitHub repository
9     git url:'https://github.com/vdharmaraj/PGDO_Proj3.git',branch:'main' //update your forked repo
10    // Get the Maven tool.
11    // ** NOTE: This 'maven-3.5.2' Maven tool must be configured
12    // **      in the global configuration.

```

Step 5:

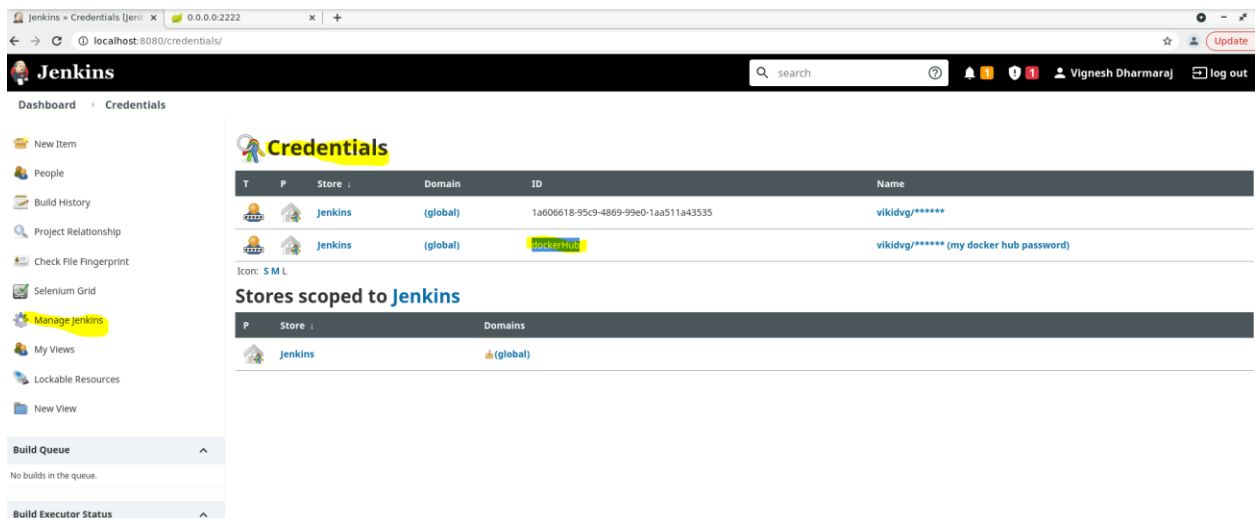
Docker Hub Credentials to be maintained in Jenkins server as mentioned in the Jenkinsfile

```

//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



The screenshot shows the Jenkins web interface at localhost:8080/credentials/. The 'Manage Jenkins' sidebar is visible on the left. The main content area is titled 'Credentials' and displays a table of credentials:

T	P	Store	Domain	ID	Name
		Jenkins	(global)	1a606618-95c9-4869-99e0-1aa511a43535	vikidvg/*****
		Jenkins	(global)	dockerHub	vikidvg/***** (my docker hub password)

Below the table, there is a section 'Stores scoped to Jenkins' with a table showing the 'Jenkins' store for the '(global)' domain.

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
```

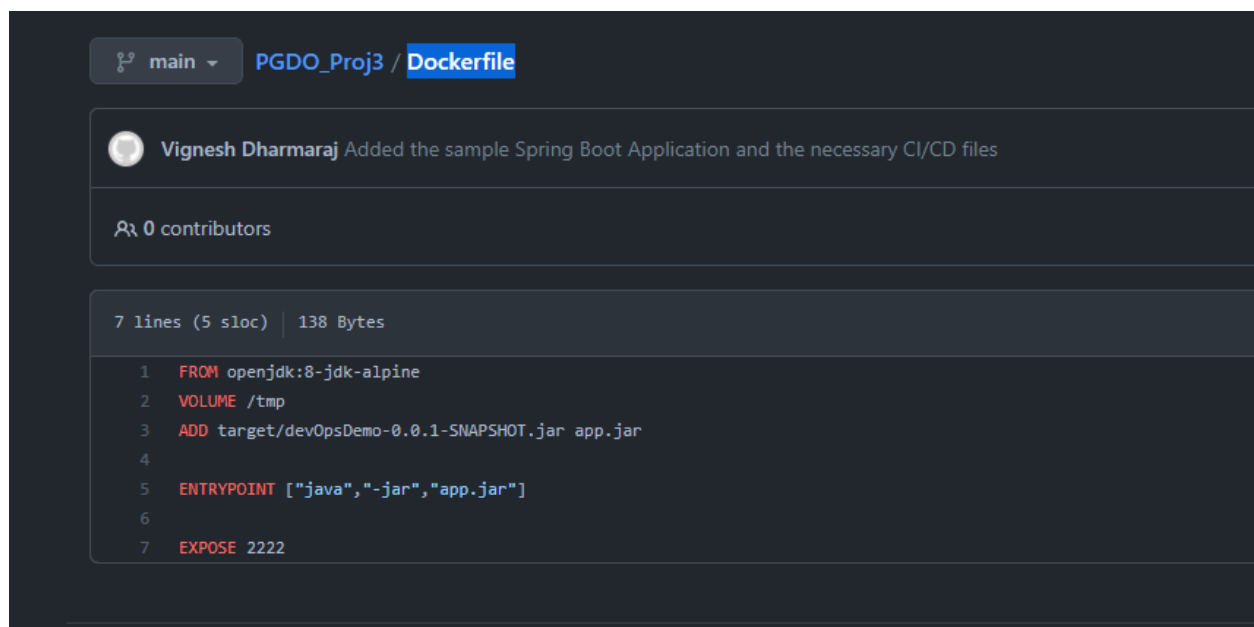
and you can see something like below:

```
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



The screenshot shows a GitHub repository interface. At the top, there's a navigation bar with 'main' selected and 'PGDO_Proj3 / Dockerfile' highlighted. Below this, a commit message by 'Vignesh Dharmaraj' is visible: 'Added the sample Spring Boot Application and the necessary CI/CD files'. Underneath the commit, it says '0 contributors'. The main part of the screenshot shows the content of the 'Dockerfile' file, which is 7 lines long (5 sloc) and 138 Bytes. The code is as follows:

```
1 FROM openjdk:8-jdk-alpine
2 VOLUME /tmp
3 ADD target/devOpsDemo-0.0.1-SNAPSHOT.jar app.jar
4
5 ENTRYPOINT ["java", "-jar", "app.jar"]
6
7 EXPOSE 2222
```

Step 8:

Install required Jenkins plugins if not installed already

Use manage plugins option to install the plugins

Q git

Updates Available **Installed** Advanced

Enabled	Name	Version	Previously installed version
<input checked="" type="checkbox"/>	Apache HttpComponents Client 4.x API Plugin Bundles Apache HttpComponents Client 4.x and allows it to be used by Jenkins plugins. This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.	4.5.13-1.0	
<input checked="" type="checkbox"/>	Caffeine API Plugin Caffeine api plugin for use by other Jenkins plugins.	2.9.1-23.v51c4e2c879c8	
<input checked="" type="checkbox"/>	Credentials Plugin This plugin allows you to store credentials in Jenkins.	2.5	
<input checked="" type="checkbox"/>	Display URL API Provides the DisplayURLProvider extension point to provide alternate URLs for use in notifications	2.3.5	
<input checked="" type="checkbox"/>	Git This plugin integrates Git with Jenkins.	4.7.2	

Q maven

Updates Available **Installed** Advanced

Enabled	Name	Version	Previously installed version
<input checked="" type="checkbox"/>	Apache HttpComponents Client 4.x API Plugin Bundles Apache HttpComponents Client 4.x and allows it to be used by Jenkins plugins. This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.	4.5.13-1.0	
<input checked="" type="checkbox"/>	Javadoc Plugin	1.6	
<input checked="" type="checkbox"/>	JSch dependency plugin Jenkins plugin that brings the JSch library as a plugin dependency, and provides an SSHAuthenticatorFactory for using JSch with the ssh-credentials plugin.	0.1.55.2	
<input checked="" type="checkbox"/>	JUnit Allows JUnit-format test results to be published.	1.50	
<input checked="" type="checkbox"/>	Mailer Plugin This plugin allows you to configure email notifications for build results	1.34	
<input checked="" type="checkbox"/>	Maven Integration plugin This plug-in provides, for better and for worse, a deep integration of Jenkins and Maven: Automatic triggers between projects depending on SNAPSHOTS, automated configuration of various Jenkins publishers (JUnit, ...).	3.12	

Updates Available Installed Advanced			
Enabled	Name	Version	Pre
<input checked="" type="checkbox"/>	Authentication Tokens API Plugin This plugin provides an API for converting credentials into authentication tokens in Jenkins.	1.4	
<input checked="" type="checkbox"/>	Credentials Binding Allows credentials to be bound to environment variables for use from miscellaneous build steps.	1.25	
<input checked="" type="checkbox"/>	Credentials Plugin This plugin allows you to store credentials in Jenkins.	2.5	
<input checked="" type="checkbox"/>	Docker Commons Plugin Provides the common shared functionality for various Docker-related plugins.	1.17	
<input checked="" type="checkbox"/>	Docker Pipeline Build and use Docker containers from pipelines.	1.26	
<input checked="" type="checkbox"/>	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

Maven

Name
maven-3.5.2

☒ Install automatically

Install from Apache

Version
3.5.2

Add Installer

Add Maven

Step 10:

Create a Jenkins pipeline form the Git repository

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

localhost:8080/view/all/newjob

enkins

ard > All >

Enter an item name

project33

» Required field



Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and



Maven project

Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.



Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly style job type).



Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments.

In General Tab, GitHub Project URL to be configured

localhost:8080/job/project3/configure

project3

General
Build Triggers
Advanced Project Options
Pipeline

Description

Build a Docker Jenkins Pipeline to Implement CI/CD Workflow

[Plain text] [Preview](#)

- ☐ Discard old builds
- ☐ Do not allow concurrent builds
- ☐ Do not allow the pipeline to resume if the controller restarts
- ☒ **GitHub project**

Project url

https://github.com/vdharmaaraj/PGDO_Proj3.git/

[Advanced...](#)

- ☐ Pipeline speed/durability override
- ☐ Preserve stashes from completed builds
- ☐ This project is parameterized
- ☐ Throttle builds

Build Triggers

In the build triggers section, Poll SCM to be configured

localhost:8080/job/project3/configure

project3

General
Build Triggers
Advanced Project Options
Pipeline

- ☐ Build after other projects are built
- ☐ Build periodically
- ☐ GitHub hook trigger for GITScm polling
- ☒ **Poll SCM**

Schedule

* * * * *

⚠ Do you really mean "every minute" when you say "* * * * *"? Perhaps you meant "H * * * *" to poll once per hour
Would last have run at Friday, July 23, 2021 12:37:19 AM UTC; would next run at Friday, July 23, 2021 12:37:19 AM UTC.

- ☐ Ignore post-commit hooks
- ☐ Disable this project
- ☐ Quiet period
- ☐ Trigger builds remotely (e.g., from scripts)

Advanced Project Options

In the pipeline section, our repo needs to be configured

localhost:8080/job/project3/configure

rd > project3 >

General Build Triggers Advanced Project Options **Pipeline**

Pipeline

Definition

Pipeline script from SCM

SCM

Git

Repositories

Repository URL

https://github.com/vdharmaraj/PGDO_Proj3.git

Credentials

- none - Add

Advanced...

Add Repository

Branches to build

Branch Specifier (blank for 'any')

*/main

Save Apply

Add Branch

localhost:8080/job/project3/configure

I > project3 >

General Build Triggers Advanced Project Options **Pipeline**

Branches to build

Branch Specifier (blank for 'any')

*/main

Add Branch

Repository browser

(Auto)

Additional Behaviours

Add

Script Path

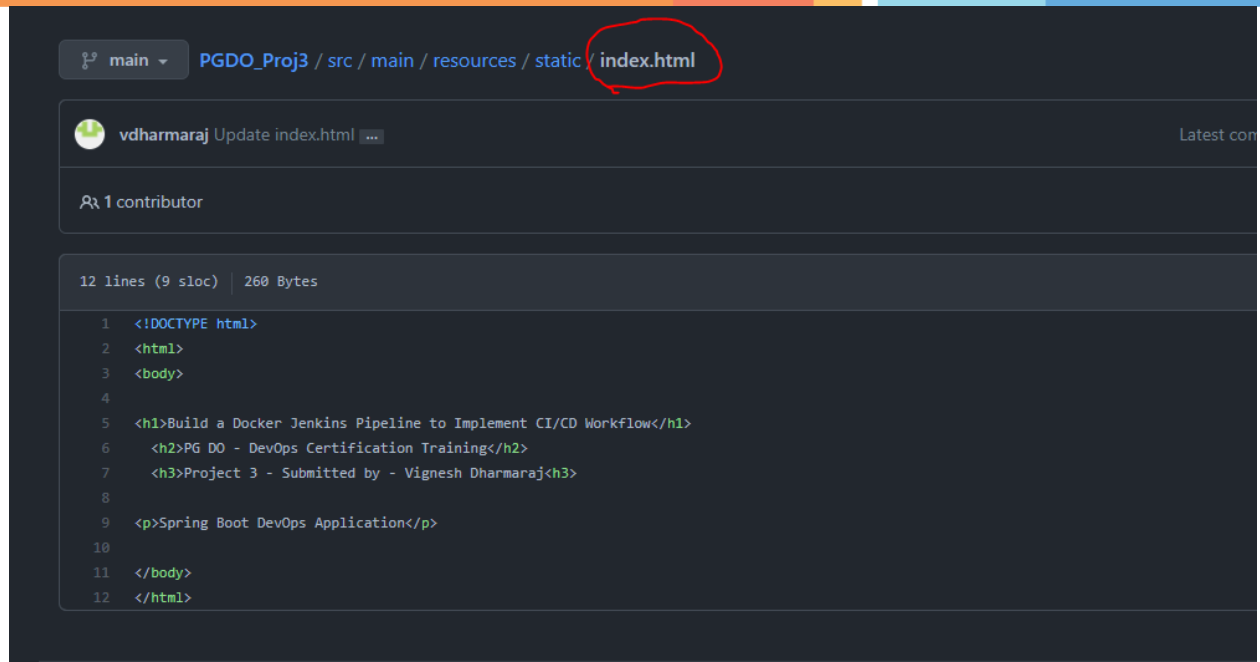
jenkinsfile

☒ Lightweight checkout

Pipeline Syntax

Step 11:

In our Repository the index.html file, which is the main 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



The screenshot shows a GitHub web interface for a repository named PGDO_Proj3. The file path is /src/main/resources/static/index.html, with 'index.html' circled in red. The file was updated by user vdhammaraj. It shows 1 contributor and 12 lines of code (9 sloc) totaling 260 bytes. The code is an HTML document with the following content:

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 <h1>Build a Docker Jenkins Pipeline to Implement CI/CD Workflow</h1>
6 <h2>PG DO - DevOps Certification Training</h2>
7 <h3>Project 3 - Submitted by - Vignesh Dharmaraj</h3>
8
9 <p>Spring Boot DevOps Application</p>
10
11 </body>
12 </html>
```

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

localhost:8080/job/project3/

Jenkins

Dashboard > project3

Back to Dashboard

Status

Changes

Build Now

Configure

Delete Pipeline

Full Stage View

GitHub

Rename

Pipeline Syntax

Git Polling Log

Build History trend

find

#23 Jul 22 23:57 1 commit

#22 Jul 22 23:33 1 commit

Pipeline project3

Build a Docker Jenkins Pipeline to Implement CI/CD Workflow

Recent Changes

Stage View

Average stage times:
(Average full run time: ~24s)

Clone Repo	Build Project	Build Docker Image with new code	Push Image to Remote Repo	Remove running container with old code	Deploy Docker Image with new changes	Remove old images
311ms	14s	1s	5s	536ms	834ms	307ms
307ms	13s	1s	5s	534ms	836ms	311ms
315ms	14s	1s	5s	538ms	832ms	303ms

Permalinks

Day13-Container O...txt

Day11-Containeriza...txt

github.com/vdhamaraj/PGDO_Proj3/commits/main

Search or jump to...

Pull requests Issues Marketplace Explore

vdhamaraj / PGDO_Proj3

code Issues Pull requests Actions Projects Wiki Security Insights Settings

main

Commits on Jul 23, 2021

- Update Jenkinsfile [Verified](#) [c278fbc](#) [vdhamaraj](#) committed 1 hour ago
- Update Jenkinsfile [Verified](#) [e1487a8](#) [vdhamaraj](#) committed 1 hour ago
- Update README.md [Verified](#) [6d4c2ad](#) [vdhamaraj](#) committed 1 hour ago
- Update Jenkinsfile [Verified](#) [9628ecd](#) [vdhamaraj](#) committed 1 hour ago
- Update README.md [Verified](#) [1881106](#) [vdhamaraj](#) committed 2 hours ago
- Update README.md [Verified](#) [6d64999](#) [vdhamaraj](#) committed 2 hours ago
- Update README.md [Verified](#) [8a80e05](#) [vdhamaraj](#) committed 2 hours ago

Result 2:

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

Console Output

```

Started by an SCM change
Obtained Jenkinsfile from git https://github.com/vdharmaraj/PGD0_Proj3.git
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/project3
[Pipeline] {
[Pipeline] tool
[Pipeline] stage
[Pipeline] { (Clone Repo)
[Pipeline] git
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/project3/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/vdharmaraj/PGD0_Proj3.git # timeout=10
Fetching upstream changes from https://github.com/vdharmaraj/PGD0_Proj3.git
> git --version # timeout=10
> git --version # 'git version 2.7.4'
> git fetch --tags --progress https://github.com/vdharmaraj/PGD0_Proj3.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision c278fbed2ecda00b9efde9b1f93b77ceb91c780 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f c278fbed2ecda00b9efde9b1f93b77ceb91c780 # timeout=10
> git branch -a -v --no-abbrev # timeout=10
> git branch -D main # timeout=10
> git checkout -b main c278fbed2ecda00b9efde9b1f93b77ceb91c780 # timeout=10
Commit message: "Update Jenkinsfile"
> git rev-list --no-walk e1407a840b33ae6e4db577dd57dd5b654e8890b0 # timeout=10
[Pipeline] tool
  
```

Result 3:

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

registry.hub.docker.com/repository/registry-1.docker.io/vikidvg/devopsexample/tags?page=1&ordering=last_updated

Advanced Image Management
View all your images and tags in this repository, clean up unused content, recover untagged images. Available for Pro and Team accounts. [View preview](#)

Sort by: Newest [Delete](#)

TAG	DIGEST	OS/ARCH	LAST PULL	COMPRESSED SIZE
latest Last pushed an hour ago by vikidvg	c1e2ccc889c4	linux/amd64	a few seconds ago	90.46 MB
23 Last pushed an hour ago by vikidvg	c1e2ccc889c4	linux/amd64	a few seconds ago	90.46 MB

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

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	"/home/weave/launch..."	4 hours ago	Up 4 hours		k8s_weave_weave-net-8glwb_kub
e5b43c870a95	df29c0a4002c		4 hours ago	Up 4 hours		k8s_kube-scheduler_kube-sched
370-b411-c33b76ab3e44	0		4 hours ago	Up 4 hours		
cc2e32efecf2	6be0dc1302e3	"kube-scheduler --au..."	4 hours ago	Up 4 hours		

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.

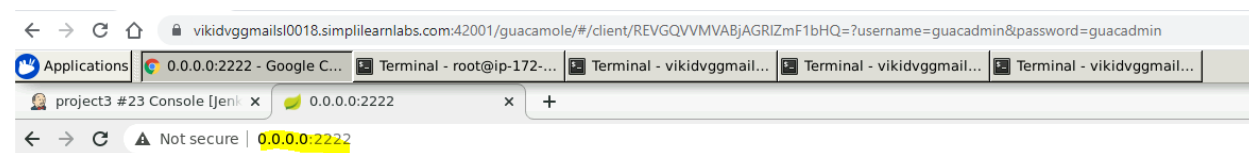
With below command, we can also see the related version of the image in local system

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
vikidvg/devopsexample	23	520af0e38a80	57 minutes ago	129MB
vikidvg/devopsexample	22	ad027415572b	About an hour 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 in 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.