

Build a Docker Jenkins Pipeline to Implement CI/CD Workflow

- Sai Ram Upparipally

Contents

- 1. *Introduction to the Project***
- 2. *Installation of pre-requisites/tools***
 - ***Installing Git3***
 - ***Creating GitHub Account***
 - ***Setting up Jenkins***
 - ***Docker Community Edition Installation***
- 3. *Execution of the Project***
- 4. *Project Results***
- 5. *Conclusion***

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

- o 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 Docker file 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

Installing Git

Step 1: Verifying the Git installation

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

```
sairamslcgmail@ip-172-31-29-202:~$ git --version
git version 2.39.0
```

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/multiverse 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)
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.
```

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

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:

```
sairamslc@gmail@ip-172-31-29-202:~$ java -version
openjdk version "11.0.13" 2021-10-19
OpenJDK Runtime Environment (build 11.0.13+8-Ubuntu-0ubuntu1.20.04)
OpenJDK 64-Bit Server VM (build 11.0.13+8-Ubuntu-0ubuntu1.20.04, mixed mode, sharing)
```

Step 2: Downloading and installing the Jenkins app

1.1 Open the terminal.

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

```
sairamslc@gmail@ip-172-31-21-159:~$ wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -
OK
```

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

```
sairamslc@gmail@ip-172-31-21-159:~$ 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.

```
sairamslc@gmail@ip-172-31-29-202:~$ sudo apt-get install jenkins
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  daemon
Use 'sudo apt autoremove' to remove it.
The following packages will be upgraded:
  jenkins
1 upgraded, 0 newly installed, 0 to remove and 407 not upgraded.
Need to get 92.9 MB of archives.
After this operation, 21.3 MB of additional disk space will be used.
Get:1 https://pkg.jenkins.io/debian-stable binary/ jenkins 2.375.1 [92.9 MB]
Fetched 92.9 MB in 8s (11.5 MB/s)
(Reading database ... 242343 files and directories currently installed.)
Preparing to unpack .../jenkins_2.375.1_all.deb ...
Unpacking jenkins (2.375.1) over (2.319.2) ...
Setting up jenkins (2.375.1) ...
Installing new version of config file /etc/default/jenkins ...
Installing new version of config file /etc/init.d/jenkins ...
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.service → /lib/systemd/system/jenkins.service.
Processing triggers for systemd (245.4-4ubuntu3.15) ...
```

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.

```
sairamslc@gmail@ip-172-31-29-202:~$ sudo service jenkins status
● jenkins.service - Jenkins Continuous Integration Server
  Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
  Active: active (running) since Fri 2022-12-23 15:33:14 UTC; 2min 14s ago
    Main PID: 13155 (java)
      Tasks: 45 (limit: 18834)
     Memory: 2.3G
       CGroup: /system.slice/jenkins.service
           └─13155 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war -->

Dec 23 15:33:12 ip-172-31-29-202 jenkins[13155]: WARNING: Use --illegal-access=warn to enable w>
Dec 23 15:33:12 ip-172-31-29-202 jenkins[13155]: WARNING: All illegal access operations will be>
Dec 23 15:33:13 ip-172-31-29-202 jenkins[13155]: 2022-12-23 15:33:13.829+0000 [id=31] IN>
Dec 23 15:33:13 ip-172-31-29-202 jenkins[13155]: 2022-12-23 15:33:13.829+0000 [id=31] IN>
Dec 23 15:33:13 ip-172-31-29-202 jenkins[13155]: 2022-12-23 15:33:13.841+0000 [id=31] IN>
Dec 23 15:33:13 ip-172-31-29-202 jenkins[13155]: 2022-12-23 15:33:13.846+0000 [id=35] IN>
Dec 23 15:33:13 ip-172-31-29-202 jenkins[13155]: 2022-12-23 15:33:13.986+0000 [id=36] IN>
Dec 23 15:33:13 ip-172-31-29-202 jenkins[13155]: 2022-12-23 15:33:13.995+0000 [id=37] IN>
Dec 23 15:33:14 ip-172-31-29-202 jenkins[13155]: 2022-12-23 15:33:14.029+0000 [id=23] IN>
Dec 23 15:33:14 ip-172-31-29-202 systemd[1]: Started Jenkins Continuous Integration Server.

[1]+  Stopped                  sudo service jenkins status
```

2.7 Run the following commands to start Jenkins.

sudo systemctl start jenkins
sudo systemctl status jenkins

```
sairamslcgma@ip-172-31-29-202:~$ sudo systemctl start jenkins
sairamslcgma@ip-172-31-29-202:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2022-12-23 15:33:14 UTC; 12min ago
     Main PID: 13155 (java)
       Tasks: 45 (limit: 18834)
      Memory: 2.3G
        CGroup: /system.slice/jenkins.service
                └─13155 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Dec 23 15:33:12 ip-172-31-29-202 jenkins[13155]: WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
Dec 23 15:33:12 ip-172-31-29-202 jenkins[13155]: WARNING: All illegal access operations will be denied in a future release
Dec 23 15:33:13 ip-172-31-29-202 jenkins[13155]: 2022-12-23 15:33:13.829+0000 [id=31]           INFO      jenkins.InitReactorRunner$1#onAttained: System config
Dec 23 15:33:13 ip-172-31-29-202 jenkins[13155]: 2022-12-23 15:33:13.829+0000 [id=31]           INFO      jenkins.InitReactorRunner$1#onAttained: System config
Dec 23 15:33:13 ip-172-31-29-202 jenkins[13155]: 2022-12-23 15:33:13.841+0000 [id=31]           INFO      jenkins.InitReactorRunner$1#onAttained: Loaded all
Dec 23 15:33:13 ip-172-31-29-202 jenkins[13155]: 2022-12-23 15:33:13.846+0000 [id=35]           INFO      jenkins.InitReactorRunner$1#onAttained: Configuration
Dec 23 15:33:13 ip-172-31-29-202 jenkins[13155]: 2022-12-23 15:33:13.986+0000 [id=36]           INFO      j.install.InstallState$Upgrade#applyForcedChanges: b
Dec 23 15:33:13 ip-172-31-29-202 jenkins[13155]: 2022-12-23 15:33:13.995+0000 [id=37]           INFO      jenkins.InitReactorRunner$1#onAttained: Completed i
Dec 23 15:33:14 ip-172-31-29-202 jenkins[13155]: 2022-12-23 15:33:14.029+0000 [id=23]           INFO      hudson.lifecycle.Lifecycle#onReady: Jenkins is full
Dec 23 15:33:14 ip-172-31-29-202 systemd[1]: Started Jenkins Continuous Integration Server.
[lines 1-19/19 (END)]
```

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

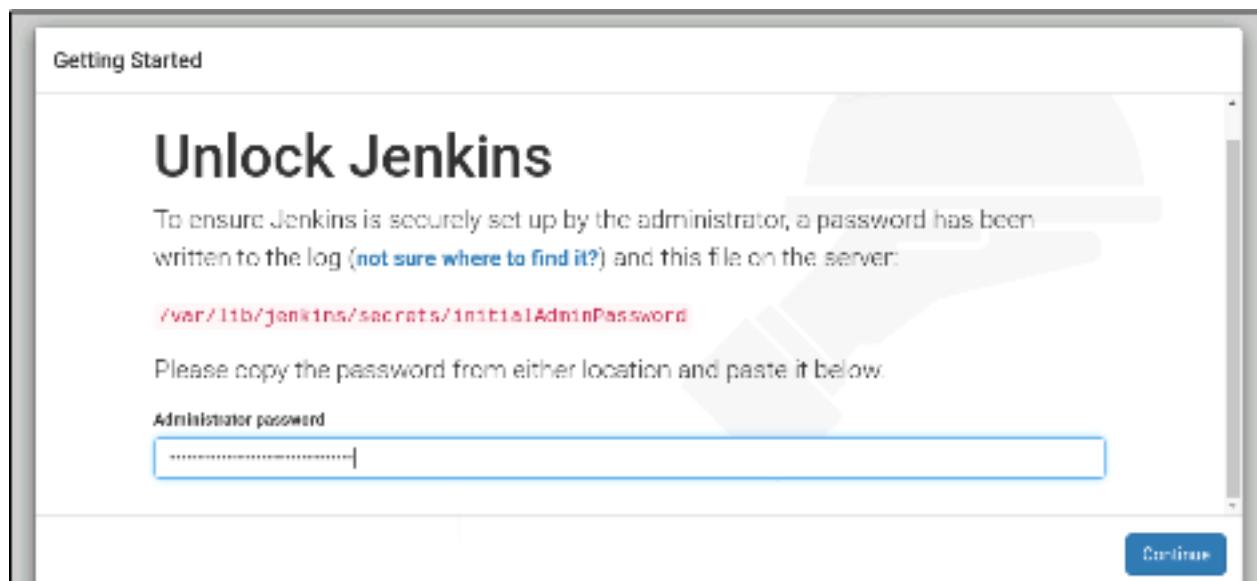


In your terminal run the following command:

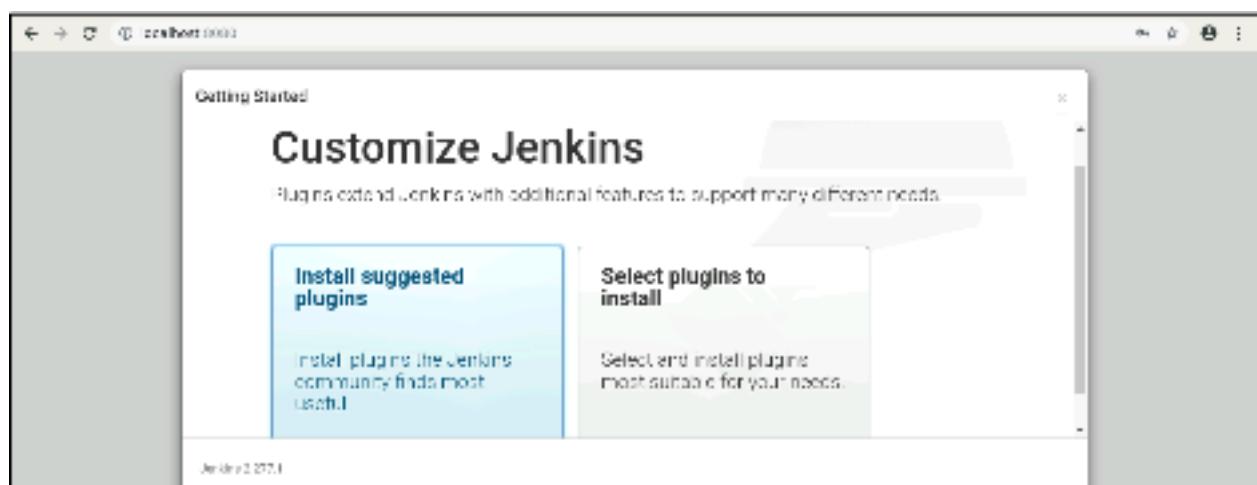
sudo cat /var/lib/jenkins/secrets/initialAdminPassword

```
sairamslcgma@ip-172-31-26-40:~$ sudo cat /var/lib/jenkins/secrets/initialAdmin
Password
Bc28f5b471124036943f58dcc27859b2
```

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



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



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

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](#)

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

Jenkins is ready!

You have skipped the setup of an admin user.

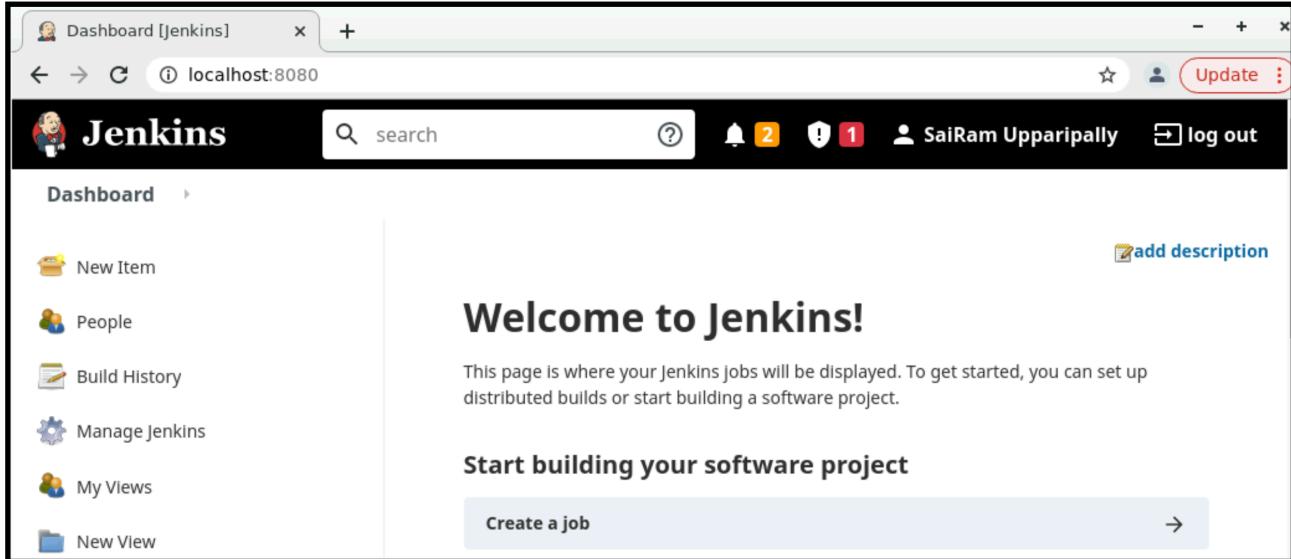
To log in, use the username: "admin" and the administrator password you used to access the setup wizard.

Your Jenkins setup is complete.

[Start using Jenkins](#)

Jenkins 2.277.1

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



Docker Community Edition Installation

Step 1: Install the Docker CE from Docker Repository

Use the following command to update the apt package:

sudo apt-get update

```
sairamslc@gmail@ip-172-31-26-40:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial-backports InRelease
Get:4 http://dl.google.com/linux/chrome/deb stable InRelease [1,811 B]
Hit:5 https://artifacts.elastic.co/packages/7.x/apt stable InRelease
Hit:6 https://deb.nodesource.com/node_14.x xenial InRelease
Hit:7 http://repo.zabbix.com/zabbix/3.0/ubuntu trusty InRelease
Hit:8 http://ppa.launchpad.net/ansible/ansible/ubuntu xenial InRelease
Hit:9 http://security.ubuntu.com/ubuntu xenial-security InRelease
Hit:10 http://ppa.launchpad.net/deadsnakes/ppa/ubuntu xenial InRelease
```

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
```

```
sairamslc@gmail@ip-172-31-26-40:~$ sudo apt-get install \
> apt-transport-https \
> 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 (1.2.35).
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 92 not upgraded.
sairamslc@gmail@ip-172-31-26-40:~$
```

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
```

```
sairamslc@gmail@ip-172-31-21-159:~$ curl -fsSL https://download.docker.com/linux/
ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
```

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
```

```
sairamslcgmail@ip-172-31-26-40:~$ echo \  
> "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]https://do  
wnload.docker.com/linux/ubuntu \  
> $(lsb_release -cs) stable" |sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

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

sudo apt-get install docker-ce

docker --version

```
sairamslcgmail@ip-172-31-26-40:~$ docker --version  
Docker version 19.03.14, build 5eb3275d40
```

Step 2: Verify the correctly installed Docker engine

Use the following command to verify the Docker engine:

sudo docker run hello-world

```
sairamslc@gmail@ip-172-31-26-40:~$ sudo docker run hello-world
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:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
sairamslc@gmail@ip-172-31-26-40:~$
```

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

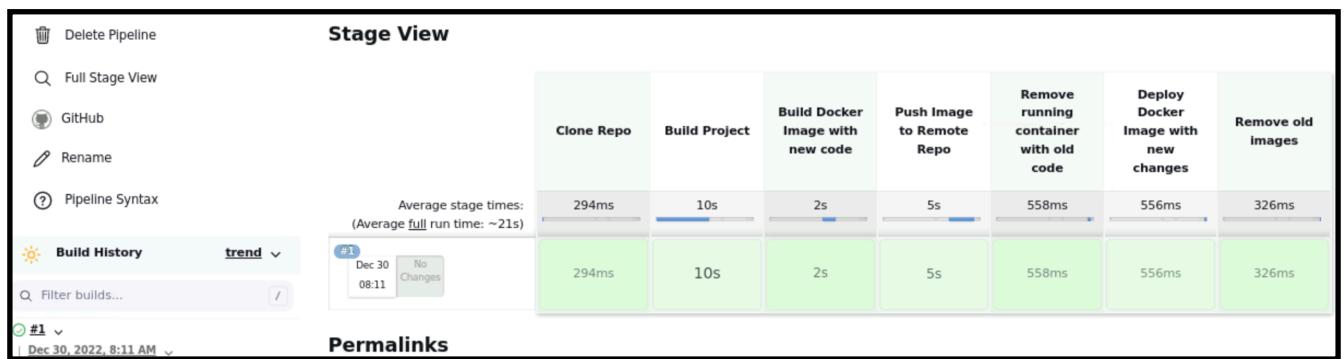
3. Execution of the Project

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:

Upload the source code to git by installing git to local machine .

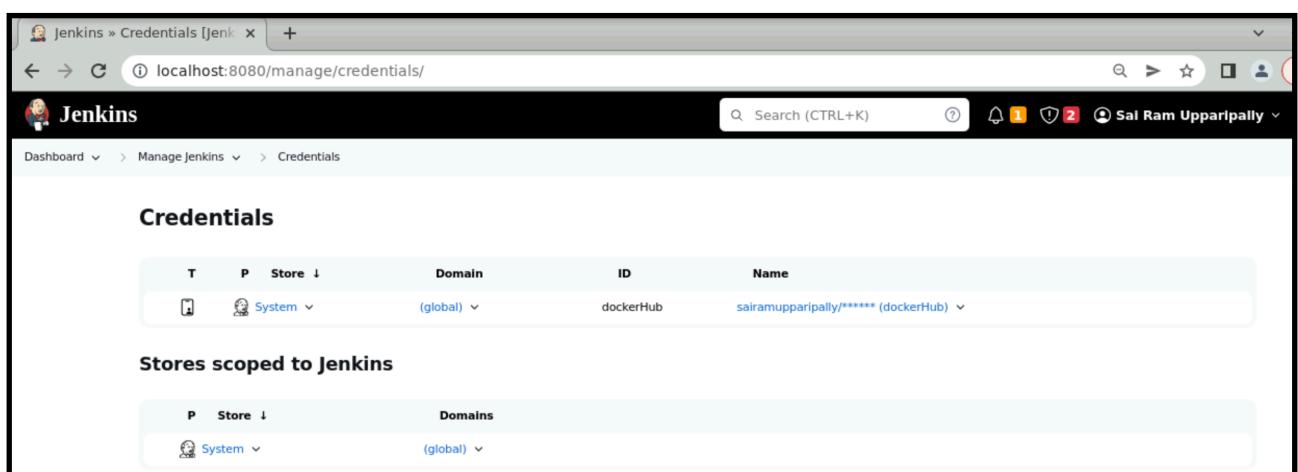
Step 2:

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

Jenkinsfile

```
//push image to remote repository , in your jenkins you na
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 Jenkins file of the repository



The screenshot shows the Jenkins management interface for credentials. The URL is `localhost:8080/manage/credentials/`. The page title is "Jenkins » Credentials [jenkins]". The main section is titled "Credentials" and lists one item:

T	P	Store	Domain	ID	Name
System			(global)	dockerHub	sairamupparipally***** (dockerHub)

Below this, there is a section titled "Stores scoped to Jenkins" which shows a single store entry:

P	Store	Domains
System		(global)

Step 3:

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

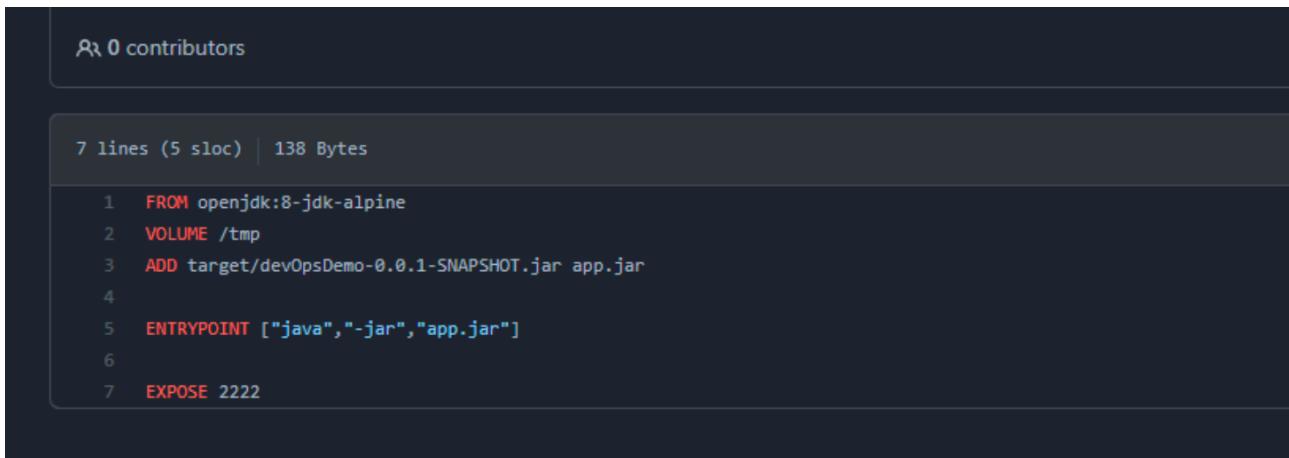
You can check if the above command was successful by doing grep docker /etc/group and you can see something like below:

docker:x:998:[user]

Step 4:

Review the Dockerfile available in the repo.

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



A screenshot of a GitHub repository page showing a Dockerfile. The Dockerfile contains the following code:

```
FROM openjdk:8-jdk-alpine
VOLUME /tmp
ADD target/devOpsDemo-0.0.1-SNAPSHOT.jar app.jar
ENTRYPOINT ["java","-jar","app.jar"]
EXPOSE 2222
```

Step 5:

Install required Jenkins plugins if not installed already Use manage plugins option to install the plugins

The screenshot shows the Jenkins Manage Plugins interface. A search bar at the top contains the text 'git'. Below it, a navigation bar has tabs for 'Updates', 'Available', 'Installed' (which is selected), and 'Advanced'. A table lists Jenkins plugins. The columns are 'Enabled', 'Name', 'Version', and 'Previously installed version'. The 'Name' column includes a dropdown arrow. The 'Version' column includes a link. A yellow box highlights the 'Git' plugin entry.

Enabled	Name	Version	Previously installed version
<input checked="" type="checkbox"/>	Apache HttpComponents Client 4.x API Plugin	4.5.13-1.0	
<input checked="" type="checkbox"/>	Caffeine API Plugin	2.9.1- 23.v51c4e2c879c8	
<input checked="" type="checkbox"/>	Credentials Plugin	2.5	
<input checked="" type="checkbox"/>	Display URL API	2.3.5	
<input checked="" type="checkbox"/>	Git	4.7.2	

The screenshot shows the Jenkins Manage Plugins interface. A search bar at the top contains the text 'maven'. Below it, a navigation bar has tabs for 'Updates', 'Available', 'Installed' (which is selected), and 'Advanced'. A table lists Jenkins plugins. The columns are 'Enabled', 'Name', 'Version', and 'Previously installed version'. The 'Name' column includes a dropdown arrow. The 'Version' column includes a link. A yellow box highlights the 'Maven Integration plugin' entry.

Enabled	Name	Version	Previously installed version
<input checked="" type="checkbox"/>	Apache HttpComponents Client 4.x API Plugin	4.5.13- 1.0	
<input checked="" type="checkbox"/>	Javadoc Plugin	1.6	
<input checked="" type="checkbox"/>	JSch dependency plugin	0.1.55.2	
<input checked="" type="checkbox"/>	JUnit	1.50	
<input checked="" type="checkbox"/>	Mailer Plugin	1.34	
<input checked="" type="checkbox"/>	Maven Integration plugin	3.12	

Updates	Available	Installed	Advanced
Enabled	Name	Version	Pre
<input checked="" type="checkbox"/>	Authentication Tokens API Plugin	1.4	
	This plugin provides an API for converting credentials into authentication tokens in Jenkins.		
<input checked="" type="checkbox"/>	Credentials Binding	1.25	
	Allows credentials to be bound to environment variables for use from miscellaneous build steps.		
<input checked="" type="checkbox"/>	Credentials Plugin	2.5	
	This plugin allows you to store credentials in Jenkins.		
<input checked="" type="checkbox"/>	Docker Commons Plugin	1.17	
	Provides the common shared functionality for various Docker-related plugins.		
<input checked="" type="checkbox"/>	Docker Pipeline	1.26	
	Build and use Docker containers from pipelines.		
<input checked="" type="checkbox"/>	Folders Plugin	6.15	
	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.		

Step 6:

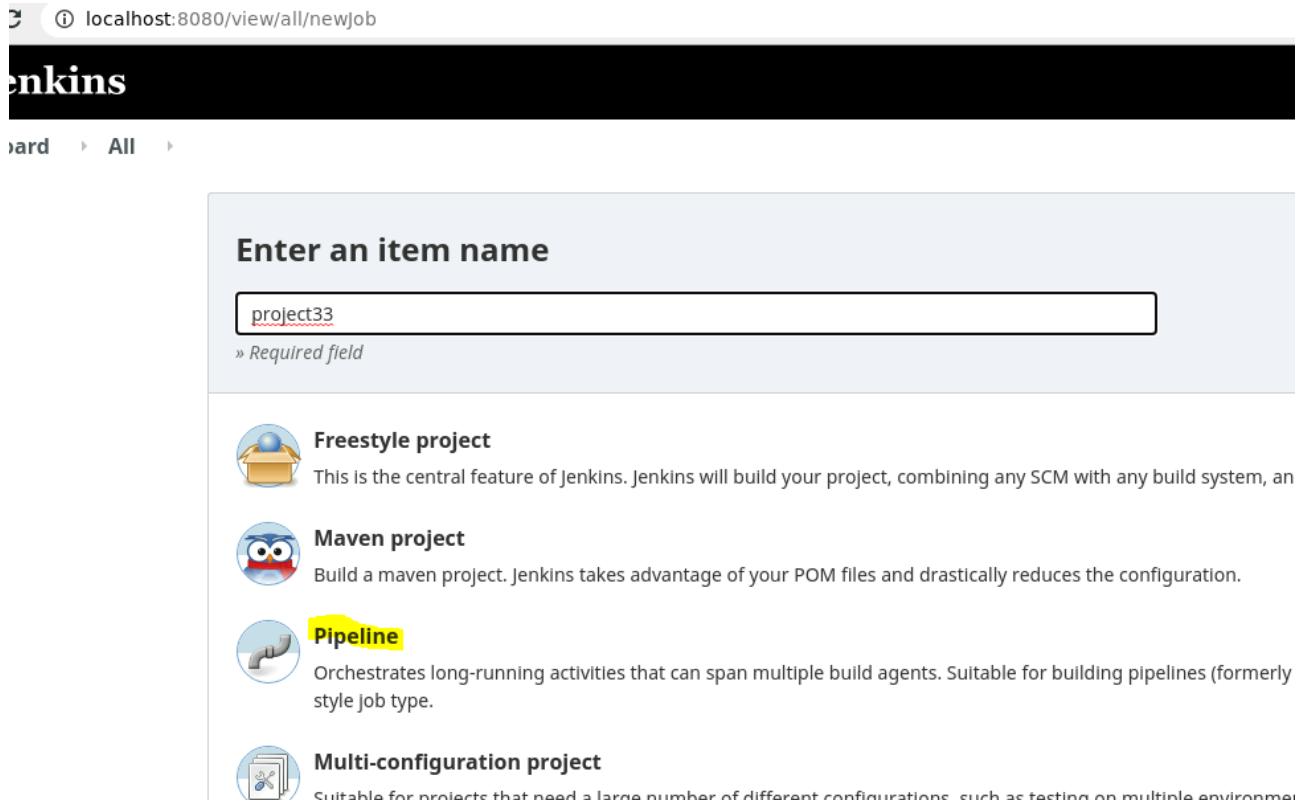
Global tools configuration in Jenkins is required as below:

The screenshot shows the 'Global Tool Configuration' page in Jenkins. The 'Maven Configuration' section is active, displaying options for setting providers. Under 'Default settings provider', there are two choices: 'Use default maven settings' (selected) and 'Use default maven global settings'. Below this, the 'JDK' section is shown, with its heading highlighted in yellow. It lists a single 'JDK' entry named 'jdk 1.8' with the path '/usr/lib/jvm/java-8-openjdk-amd64/'. There is also an unchecked checkbox for 'Install automatically'.

The screenshot shows the 'Global Tool Configuration' page in Jenkins, specifically the 'Maven' section. A checkbox labeled 'Install automatically' is checked. Underneath it, a sub-section titled 'Install from Apache' is expanded, showing a dropdown menu for 'Version' set to '3.5.2'. At the bottom of the page, there is a large 'Add Maven' button.

Step 07:

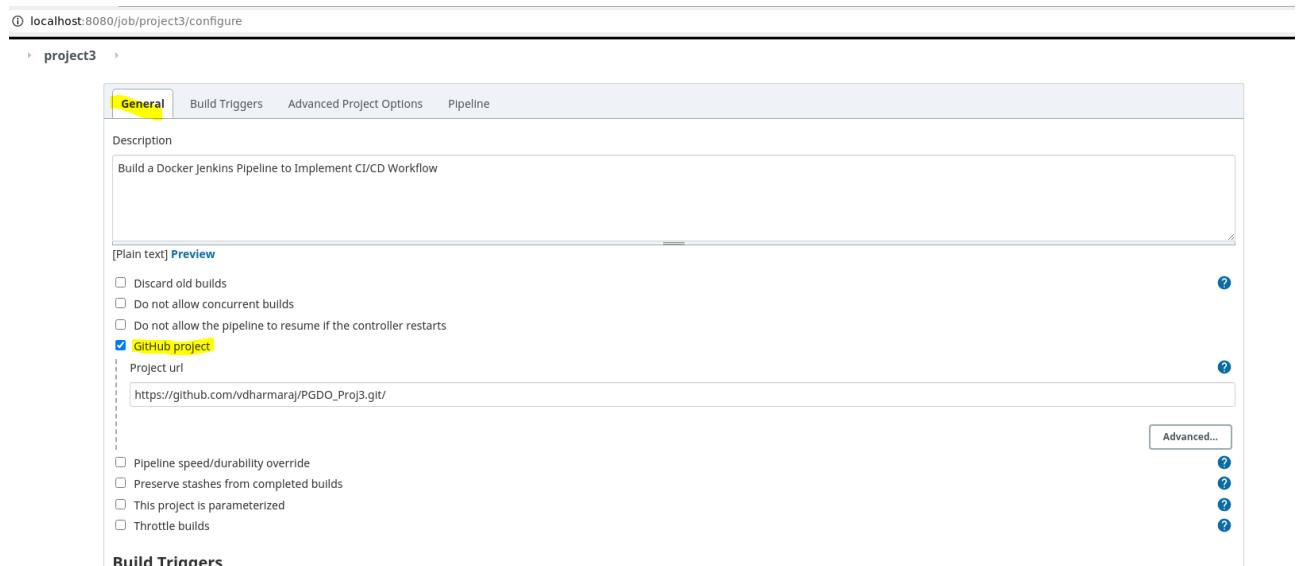
Create a Jenkins pipeline form the Git repository
New Pipeline Project has to be created in Jenkins server as below



The screenshot shows the Jenkins interface for creating a new pipeline project. The URL is localhost:8080/view/all/newJob. The title bar says "Jenkins". Below it, the breadcrumb navigation shows "Dashboard > All >". The main area has a heading "Enter an item name" with a text input field containing "project33". A note below the input says "» Required field". To the right of the input are four project types: "Freestyle project" (with a box icon), "Maven project" (with an owl icon), "Pipeline" (with a pipe icon), and "Multi-configuration project" (with a gear and wrench icon). Each type has a brief description.

- Freestyle project**
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, an
- 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



The screenshot shows the "General" configuration tab for the "project33" job. The URL is localhost:8080/job/project33/configure. The breadcrumb navigation shows "project33 >". The "General" tab is selected. The "Description" field contains "Build a Docker Jenkins Pipeline to Implement CI/CD Workflow". The "GitHub project" checkbox is checked. The "Project url" field contains "https://github.com/vdharma/PGDO_Proj3.git". The "Advanced..." button is visible at the bottom right. Other tabs include "Build Triggers", "Advanced Project Options", and "Pipeline".

In the build triggers section, Poll SCM to be configured

① 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

① 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

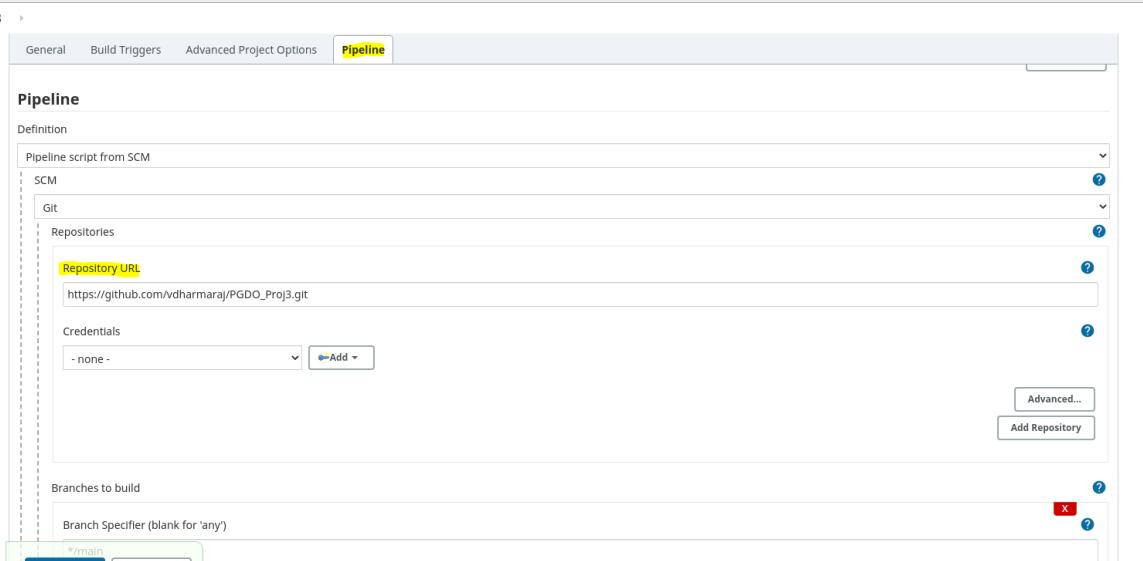
- none -

Advanced...
Add Repository

Branches to build

Branch Specifier (blank for 'any')

*main



Step 8:

In our Repository the index.html file, 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

```

<!DOCTYPE html>
<html>
<body>
<h1>Build a Docker Jenkins Pipeline to Implement CI/CD Workflow</h1>
<h2>PG DO - DevOps Certification Training</h2>
<h3>Project 3 - Submitted by - Sai Ram Upparipally (sairam.slc@gmail.com) </h3>
<p>Spring Boot DevOps Application</p>
</body>
</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

The screenshot shows the Jenkins Pipeline CICDWorkflow status page. The pipeline consists of several stages: 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, and Remove old images. The average stage times are listed as 294ms, 10s, 2s, 5s, 558ms, 556ms, and 326ms respectively. A build history table shows a single build (#1) from Dec 30 at 08:11 with 'No Changes'. The build status is green, indicating success.

Result 2:

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

The screenshot shows the Jenkins Console Output for build #1. The output log shows the Jenkinsfile being obtained from git, the pipeline starting, and the node running. It details the cloning of the repository, fetching upstream changes, and checking out the revision 48568ccb9e0f8cb2c0bf3b8a2095a60cbb8d07. The log ends with a warning about a second fetch and a sparse checkout configuration.

```
Started by user Sai Ram Upparipally
Obtained Jenkinsfile from git https://github.com/sairamupparipally/CICDWorkflow.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/CICDWorkflow
[Pipeline] {
[Pipeline] tool
[Pipeline] stage
[Pipeline] {
[Pipeline] git
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/sairamupparipally/CICDWorkflow.git
> git init /var/lib/jenkins/workspace/CICDWorkflow # timeout=10
Fetching upstream changes from https://github.com/sairamupparipally/CICDWorkflow.git
> git --version # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
> git fetch --tags --progress -- https://github.com/sairamupparipally/CICDWorkflow.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/sairamupparipally/CICDWorkflow.git # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 48568ccb9e0f8cb2c0bf3b8a2095a60cbb8d07 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
```

Result 3:

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

The screenshot shows the Docker Hub interface for the repository 'sairamupparipally/devopsexample'. The 'Tags' tab is selected. There are two tags listed: 'latest' and '1'. Both tags were pushed 4 minutes ago by the user 'sairamupparipally'. The 'latest' tag has a digest '583e1b37ea21' and is built for 'linux/amd64'. The '1' tag also has a digest '583e1b37ea21' and is built for 'linux/amd64'. Both tags have a compressed size of 90.46 MB. To the right of each tag, there is a 'docker pull' command and a download icon.

TAG	DIGEST	OS/ARCH	LAST PULL	COMPRESSED SIZE
latest	583e1b37ea21	linux/amd64	---	90.46 MB
1	583e1b37ea21	linux/amd64	---	90.46 MB

5. Conclusion

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