

Build a Docker Jenkins Pipeline to Implement CI/CD Workflow

This document has been written to explain how to create “the continuous integration and delivery by building a Docker image and pushed it in docker hub using Jenkins Pipeline”. A CI CD Pipeline serves as a way of automating software application build, tests, and deployments, which is backbone of any organization with a DevOps culture. CI CD Pipelines with Docker are best for your organization to improve code quality and deliver software releases quickly without any human errors and boosts your business greatly.

❖ Objective:

Explain step-by-step walkthrough on how CI/CD pipeline can be build using Jenkins pipeline.

❖ Solution build will provide below capabilities:

- Availability of the application and its versions in the GitHub.
 - Track their version 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 will be frequently to the production with high-end quality.
- Have track on the build status of Jenkins for every commit of the project

❖ Tools which have been used:

- Docker: To build the application from a Dockerfile and push it to Docker Hub.
- Docker Hub: To store the Docker image.
- Git: To connect and push files from the local system to GitHub
- Docker, Docker Hub, GitHub, Git, Linux (Ubuntu), Jenkins the GitHub.
- Linux (Ubuntu): As a base operating system to start and execute the project.
- Jenkins: To automate the deployment process during continuous integration.
- Eclipse or any other IDE for write and debug app.

❖ Project Expected Result:

- Jenkins pipeline with CICD pipeline as shown below, demonstrating the java simple application build and deployment process automated with git poll scm, Docker and Jenkins with 'Pipeline as a Code' approach.

❖ Project Documentation:

In this document, we are going to see the following topics in detail:

1. Prepare Your System (Installation of pre-requisites/tools).
2. Application and their docker build creation and testing on local system
3. Configure Continuous Integration using git as Version Control System or Source Code Management System
4. Configure Continuous Deployment and execute a Jenkins pipeline job and execute dockerfile respectively through a shell script
5. Use a Jenkins plugin "CloudBees Docker Build and Publish" for Continuous Deployment in docker hub.
6. Automate the process using Jenkins Build Trigger

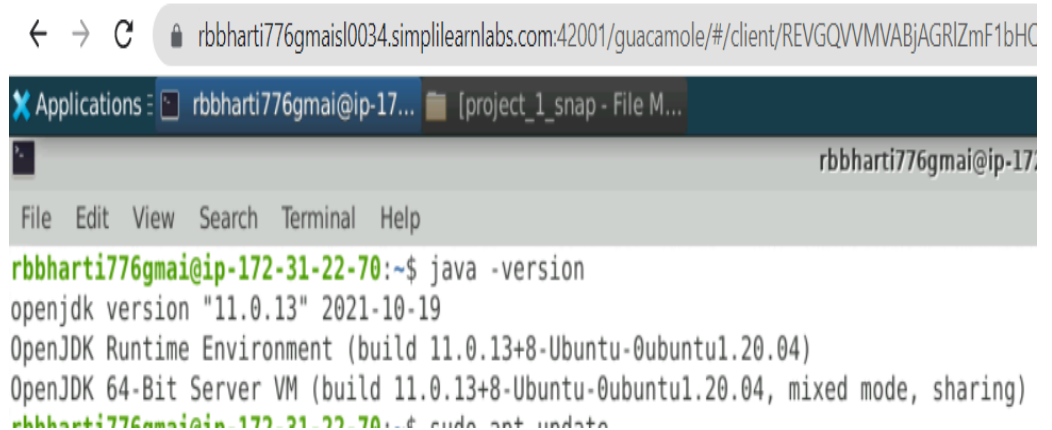
➤ Prepare Your System (Installation of pre-requisites/tools)

- Create your Virtual Machine with any cloud services or pick the operating system that suits you best. Generally, Ubuntu is preferred more, and we will be using the same. Operating system must be updated and have installed with below pre-requisites/tools and check they have installed successfully or not.

- Install & check java (1.8)

- ◆ For create and run your java application on machine, you must have Java i.e JDK installed on your machine. For check java is installed or not run below command

- java - version



```
← → ↻ rbbharti776gmail@0034.simplilearnlabs.com:42001/guacamole/#/client/REVGQVVMVABjAGRIZmF1bH...  
Applications: rbbharti776gmail@ip-17... [project_1_snap - File M...]  
rbbharti776gmail@ip-17...  
File Edit View Search Terminal Help  
rbbharti776gmail@ip-172-31-22-70:~$ 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)  
rbbharti776gmail@ip-172-31-22-70:~$ sudo apt update
```

Figure 1.check_java_installed.

- ◆ For install java in you can run below command



```
$ sudo apt update  
$ sudo apt install -y openjdk-8-jdk  
$ java -version [Verify java]
```

```

rbbharti776gmai@ip-172-31-22-70:~$ sudo apt install -y openjdk-8-jdk
Reading package lists... Done
Building dependency tree
Reading state information... Done
openjdk-8-jdk is already the newest version (8u362-ga-0ubuntu1~20.04.1).
0 upgraded, 0 newly installed, 0 to remove and 445 not upgraded.

```

Figure 2. java_8_installations

```

C:\Users\robharti>java -version
java version "1.8.0_311"
Java(TM) SE Runtime Environment (build 1.8.0_311-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.311-b11, mixed mode)

```

Figure 3.java_8_installed.

- Install & check Jenkins (whether Jenkins running on port 8080)

- ◆ Jenkins is important tool which is required to create pipeline. First, you can check Jenkins is running or not on your machine, if not use below command.



Add the Jenkins key:

```
$ wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -
```

Add the jenkins pkg repo to package manager:

Edit the file:

```
$ sudo vi /etc/apt/sources.list
```

[OR]

```
$ sudo nano /etc/apt/sources.list
```

Add this line at the end of the file and save it:

```
deb https://pkg.jenkins.io/debian-stable binary/
```

Execute:

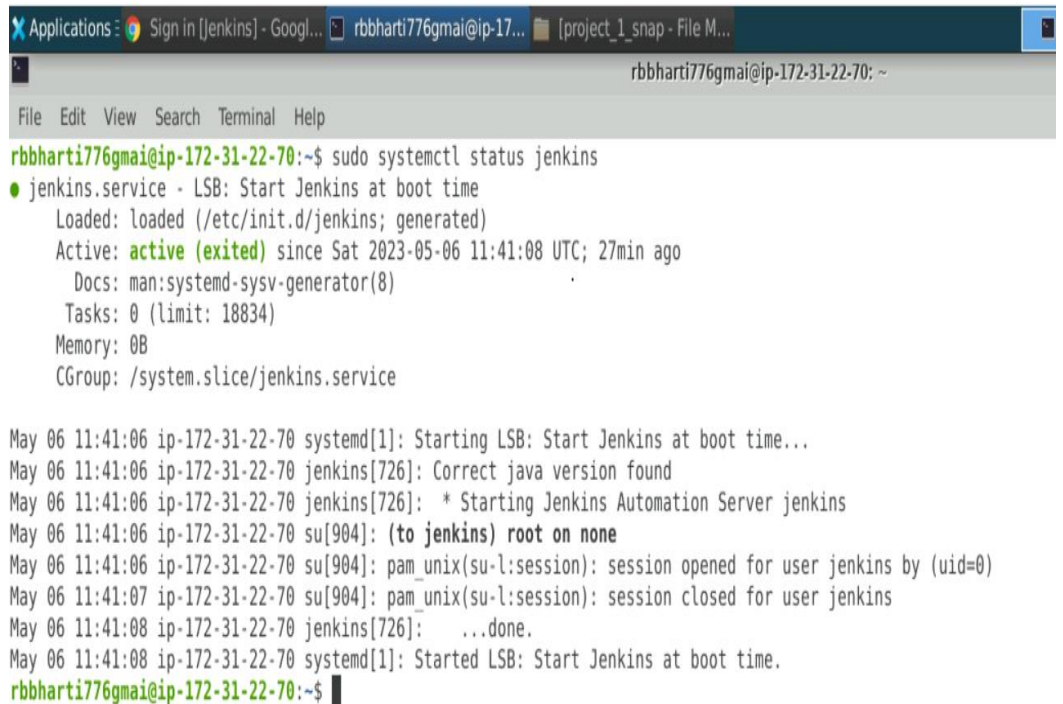
```
$ sudo apt-get update
```

Install Jenkins:

```
$ sudo apt-get install -y jenkins
```

check Jenkins installed on machine through below command.

```
sudo systemctl restart Jenkins
```

A terminal window with a dark blue title bar showing 'Applications: Sign in [Jenkins] - Googl...', 'rbbharti776gmai@ip-17...', and '[project_1_snap - File M...'. The terminal text shows the command 'sudo systemctl status jenkins' and its output, including service details and a log of Jenkins starting. The prompt is 'rbbharti776gmai@ip-172-31-22-70: ~'.

```
rbbharti776gmai@ip-172-31-22-70:~$ sudo systemctl status jenkins
● jenkins.service - LSB: Start Jenkins at boot time
   Loaded: loaded (/etc/init.d/jenkins; generated)
   Active: active (exited) since Sat 2023-05-06 11:41:08 UTC; 27min ago
     Docs: man:systemd-sysv-generator(8)
    Tasks: 0 (limit: 18834)
   Memory: 0B
    CGroup: /system.slice/jenkins.service

May 06 11:41:06 ip-172-31-22-70 systemd[1]: Starting LSB: Start Jenkins at boot time...
May 06 11:41:06 ip-172-31-22-70 jenkins[726]: Correct java version found
May 06 11:41:06 ip-172-31-22-70 jenkins[726]: * Starting Jenkins Automation Server jenkins
May 06 11:41:06 ip-172-31-22-70 su[904]: (to jenkins) root on none
May 06 11:41:06 ip-172-31-22-70 su[904]: pam_unix(su-l:session): session opened for user jenkins by (uid=0)
May 06 11:41:07 ip-172-31-22-70 su[904]: pam_unix(su-l:session): session closed for user jenkins
May 06 11:41:08 ip-172-31-22-70 jenkins[726]: ...done.
May 06 11:41:08 ip-172-31-22-70 systemd[1]: Started LSB: Start Jenkins at boot time.
rbbharti776gmai@ip-172-31-22-70:~$
```

Figure 4.check_jenkins_installed_or_not

- To get the initial admin password to unlock Jenkins:

To get the initial admin password to unlock Jenkins:

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

Install suggested plugins.

Create your first admin user.

This credential you need to login your Jenkins console.

```
May 06 11:41:08 ip-172-31-22-70 systemd[1]: Started LSB: Start Jenkins at boot time.
rbbharti776gmai@ip-172-31-22-70:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
cdba5b98f86a4ccc8291d598c6e6c468
rbbharti776gmai@ip-172-31-22-70:~$ ^C
```

Figure 5.Gennerate_jenkins_unlock_code

- On go to browser and crawl to localhost:8080, Jenkins will ask you to unlock himself. You have paste generated code in given filed.

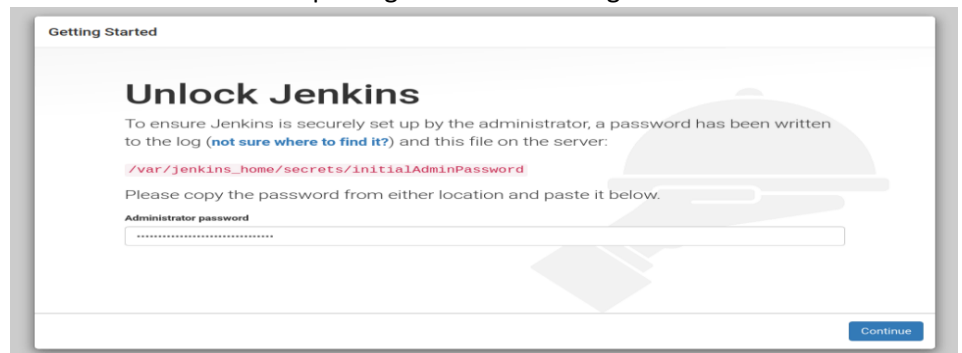
The image shows the 'Getting Started' page of Jenkins with the title 'Unlock Jenkins'. It explains that a password has been written to the log and a file on the server. The file path is shown as `/var/jenkins_home/secrets/initialAdminPassword`. It asks the user to copy the password from either location and paste it into a text field labeled 'Administrator password'. A 'Continue' button is at the bottom right.

Figure 6. Unlock_jenkins_console

- Now, on click on submit button, it was to create new user and for active this user you need to take restart of jenkins setup.

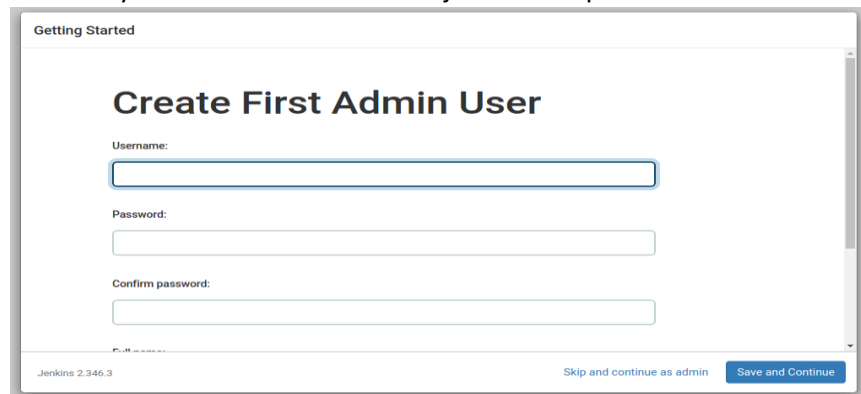
The image shows the 'Getting Started' page of Jenkins with the title 'Create First Admin User'. It has three input fields: 'Username:', 'Password:', and 'Confirm password:'. At the bottom, there are two buttons: 'Skip and continue as admin' and 'Save and Continue'. The Jenkins version 'Jenkins 2.346.3' is noted at the bottom left.

Figure 7. Create_new_jenkins_userforLogin

- For restart Jenkins, follow below command.

Manually restart the jenkins server:

```
$ sudo systemctl restart jenkins
```

[OR]

```
$ sudo service jenkins restart
```

Now go and run below url in browser

Browse: <http://localhost:8080/restart>

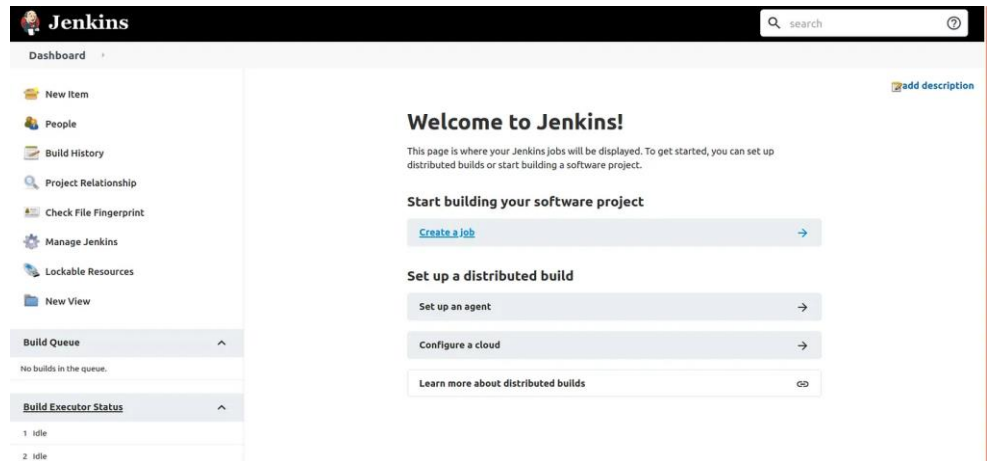


Figure 8.Jenkins_landing_page

- Git installation and configuration

- ◆ **Git installation**



```
$ sudo apt-get update
$ sudo apt-get install -y git
```

```
[Verify git installation]
git -- version
```

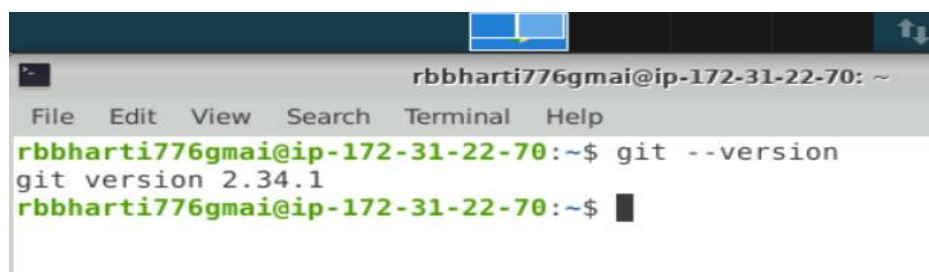


Figure 9.check_git_version

- ◆ **Git configuration**



```
$ git config --global user.email "YOUR_EMAIL_ID"
$ git config --global user.name "YOUR_USER_NAME"
$ git config --list
```

- ◆ **Git repo creation**



```
Create repo from git console by login it with your credentials
https://github.com/

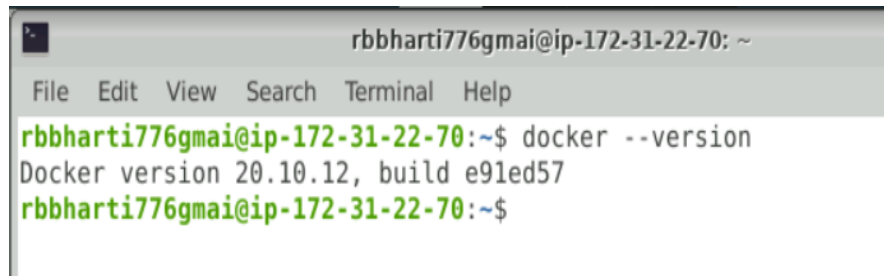
Now, Clone your repository in your Linux machine
$ git clone
https://github.com/robert776/Simplilearn_robharti_A1Project1.git
$ cd Simplilearn_robharti_A1Project1
```

- Docker installation and configuration

- ◆ Docker installation



```
$ sudo apt-get update  
  
$ sudo apt-get install -y docker.io  
  
$ docker --version
```

A terminal window with a title bar showing the user 'rbbharti776gmai' and IP 'ip-172-31-22-70'. The terminal has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The command 'docker --version' has been executed, resulting in the output 'Docker version 20.10.12, build e91ed57'.

```
rbbharti776gmai@ip-172-31-22-70: ~  
File Edit View Search Terminal Help  
rbbharti776gmai@ip-172-31-22-70:~$ docker --version  
Docker version 20.10.12, build e91ed57  
rbbharti776gmai@ip-172-31-22-70:~$
```

Figure 10.check_docker_installed_or_not

- ◆ Docker permission required for perform.



```
ONLY If you get Permission Denied error:  
  
$ sudo docker --version  
  
$ sudo usermod -aG docker <your_linux_user_name>  
  
[Relogin]  
  
$ docker --version  
  
And relog in
```

- Application and his docker build creation and testing on local System.

- Sample Application code in java

- For write and execute java application you have java installed in your machine which you can check from command “java-version”
 - Now, you can create the app for your requirement inside this local system and check output by execute your code on local system.

- ```
$ nano hello_lmsapp.java

class hello_lmsapp {

 public static void main(String[] args) {

 System.out.println("Hi Simplilearn lms - PG devops program");

 System.out.println("This is a java app created using Dockerfile");

 System.out.println("Assessment project ");

 System.out.println("Author :- Robert
Bharti,rbbharti776@gmail.com");

 }
}
```
- For execute, your java app i.e hello\_lmsapp.java locally, you have to run below command

```
javac hello_lmsapp.java → generate complied class file "hello_lmsapp.class"
java hello_lmsapp → for run class file
```

```
rbbharti776gmail@ip-172-31-22-70:~$ pwd
/home/rbbharti776gmail
rbbharti776gmail@ip-172-31-22-70:~$ mkdir rbbharti_lmsproject1
rbbharti776gmail@ip-172-31-22-70:~$ cd rbbharti_lmsproject1/
rbbharti776gmail@ip-172-31-22-70:~/rbbharti_lmsproject1$ pwd
/home/rbbharti776gmail/rbbharti_lmsproject1
rbbharti776gmail@ip-172-31-22-70:~/rbbharti_lmsproject1$ ls -la
total 8
drwxrwxr-x 2 rbbharti776gmail rbbharti776gmail 4096 May 6 13:15 .
drwxr-xr-x 18 rbbharti776gmail rbbharti776gmail 4096 May 6 13:15 ..
rbbharti776gmail@ip-172-31-22-70:~/rbbharti_lmsproject1$ nano hello_lmsapp.java
```

Figure 11. create\_folder\_for\_project.



```

File Edit View Search Terminal Help
rbbharti776gmai@ip-172-31-22-70:~/rbbharti_lmsproject1$ ls -la
total 12
drwxrwxr-x 2 rbbharti776gmai rbbharti776gmai 4096 May 6 13:08 .
drwxr-xr-x 18 rbbharti776gmai rbbharti776gmai 4096 May 6 13:05 ..
-rw-rw-r-- 1 rbbharti776gmai rbbharti776gmai 290 May 6 13:05 hello_lmsapp.java
rbbharti776gmai@ip-172-31-22-70:~/rbbharti_lmsproject1$ javac hello_lmsapp.java
rbbharti776gmai@ip-172-31-22-70:~/rbbharti_lmsproject1$ ls -la
total 16
drwxrwxr-x 2 rbbharti776gmai rbbharti776gmai 4096 May 6 13:09 .
drwxr-xr-x 18 rbbharti776gmai rbbharti776gmai 4096 May 6 13:05 ..
-rw-rw-r-- 1 rbbharti776gmai rbbharti776gmai 580 May 6 13:09 hello_lmsapp.class
-rw-rw-r-- 1 rbbharti776gmai rbbharti776gmai 290 May 6 13:05 hello_lmsapp.java
rbbharti776gmai@ip-172-31-22-70:~/rbbharti_lmsproject1$ java hello_lmsapp
Hi Simplilearn lms - PG devops program
This is a java app created using Dockerfile
Author :- Robert Bharti,rbbharti776@gmail.com
rbbharti776gmai@ip-172-31-22-70:~/rbbharti_lmsproject1$ █

```

Figure 12. run\_javaCode\_onlocal\_system

## - Dockerfile for sample application

- Dockerfile is used to create image for your application. A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image. Create a file called Docker File and edit it using vim or nano. Please note that the name of the file has to be "Dockerfile" with "D" as capital.
- Dockerfile:

```

use base image to run java app

FROM openjdk:8

define workdir

WORKDIR /var/www/java

copy all files & folders

COPY . /var/www/java

Run for generate java compile class

RUN javac hello_lmsapp.java

Excute java Class

CMD ["java", "hello_lmsapp"]

```

- For run Dockerfile and create image of your app on local system, you have check first docker is installed or not.

`docker -- version` → check docker version

then, execute below command for generate docker image

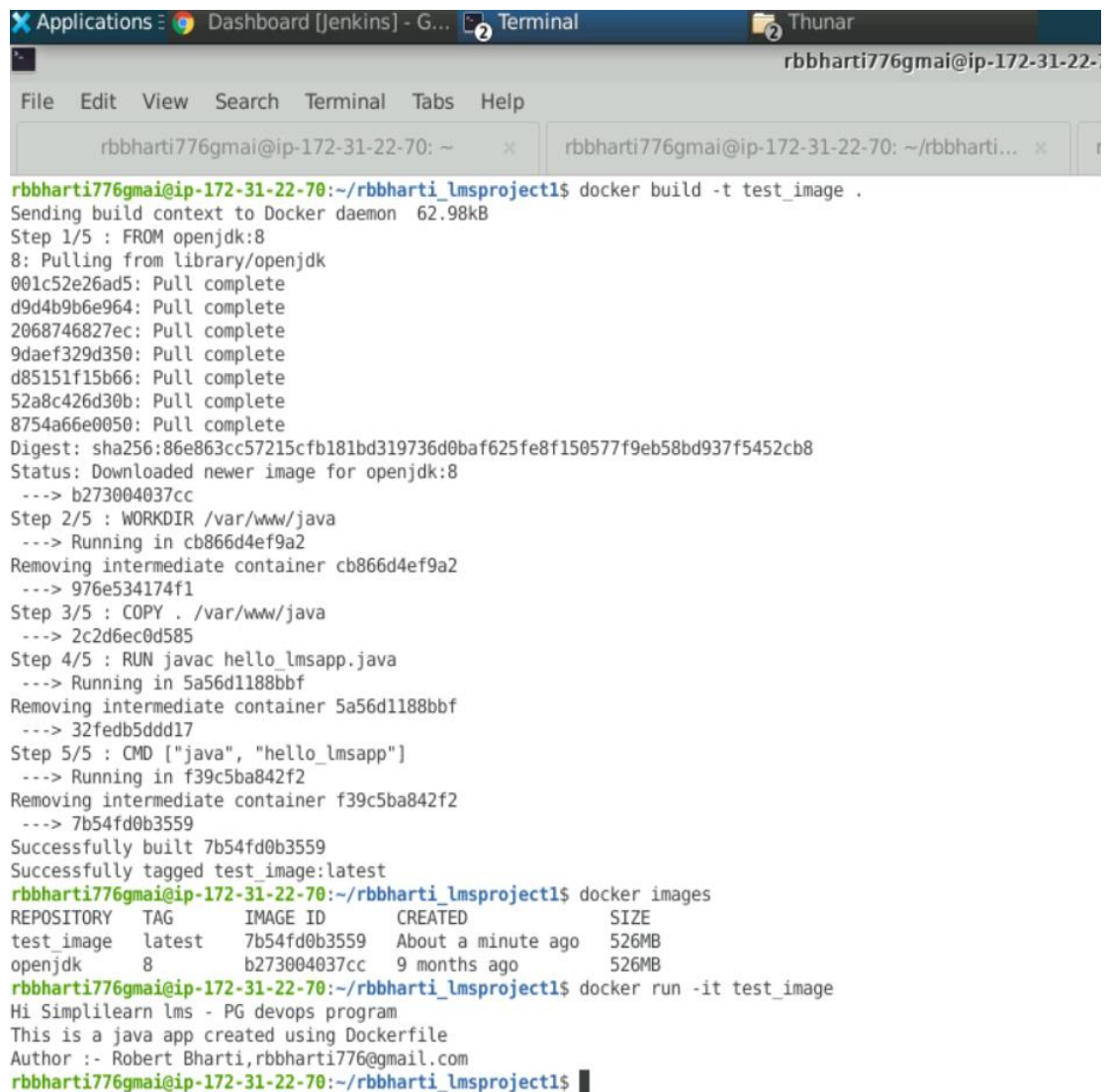
`docker build -t test_image .`

For check your image create or not run below command

`docker images`

For run your app docker image on system

`docker run -it test_images`



```
rbbharti776gmai@ip-172-31-22-70: ~/rbbharti_lmsproject1$ docker build -t test_image .
Sending build context to Docker daemon 62.98kB
Step 1/5 : FROM openjdk:8
8: Pulling from library/openjdk
001c52e26ad5: Pull complete
d9d4b9b6e964: Pull complete
2068746827ec: Pull complete
9daef329d350: Pull complete
d85151f15b66: Pull complete
52a8c426d30b: Pull complete
8754a66e0050: Pull complete
Digest: sha256:86e863cc57215cfb181bd319736d0baf625fe8f150577f9eb58bd937f5452cb8
Status: Downloaded newer image for openjdk:8
--> b273004037cc
Step 2/5 : WORKDIR /var/www/java
--> Running in cb866d4ef9a2
Removing intermediate container cb866d4ef9a2
--> 976e534174f1
Step 3/5 : COPY . /var/www/java
--> 2c2d6ec0d585
Step 4/5 : RUN javac hello_lmsapp.java
--> Running in 5a56d1188bbf
Removing intermediate container 5a56d1188bbf
--> 32fedb5ddd17
Step 5/5 : CMD ["java", "hello_lmsapp"]
--> Running in f39c5ba842f2
Removing intermediate container f39c5ba842f2
--> 7b54fd0b3559
Successfully built 7b54fd0b3559
Successfully tagged test_image:latest
rbbharti776gmai@ip-172-31-22-70:~/rbbharti_lmsproject1$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
test_image latest 7b54fd0b3559 About a minute ago 526MB
openjdk 8 b273004037cc 9 months ago 526MB
rbbharti776gmai@ip-172-31-22-70:~/rbbharti_lmsproject1$ docker run -it test_image
Hi Simplilearn lms - PG devops program
This is a java app created using Dockerfile
Author :- Robert Bharti,rbbharti776@gmail.com
rbbharti776gmai@ip-172-31-22-70:~/rbbharti_lmsproject1$
```

Figure 13. run\_docker\_file\_locally

## ➤ Configure Continuous Integration using git as Source Code Management System

- Once, java application and docker build run successfully on local system, commit these files in your git repo, so that can use for create CI/CD pipeline.

[https://github.com/robert776/Simplilearn\\_robharti\\_AIProject1.git](https://github.com/robert776/Simplilearn_robharti_AIProject1.git)

- You will get error, if your machine ssh key is not added in your repo

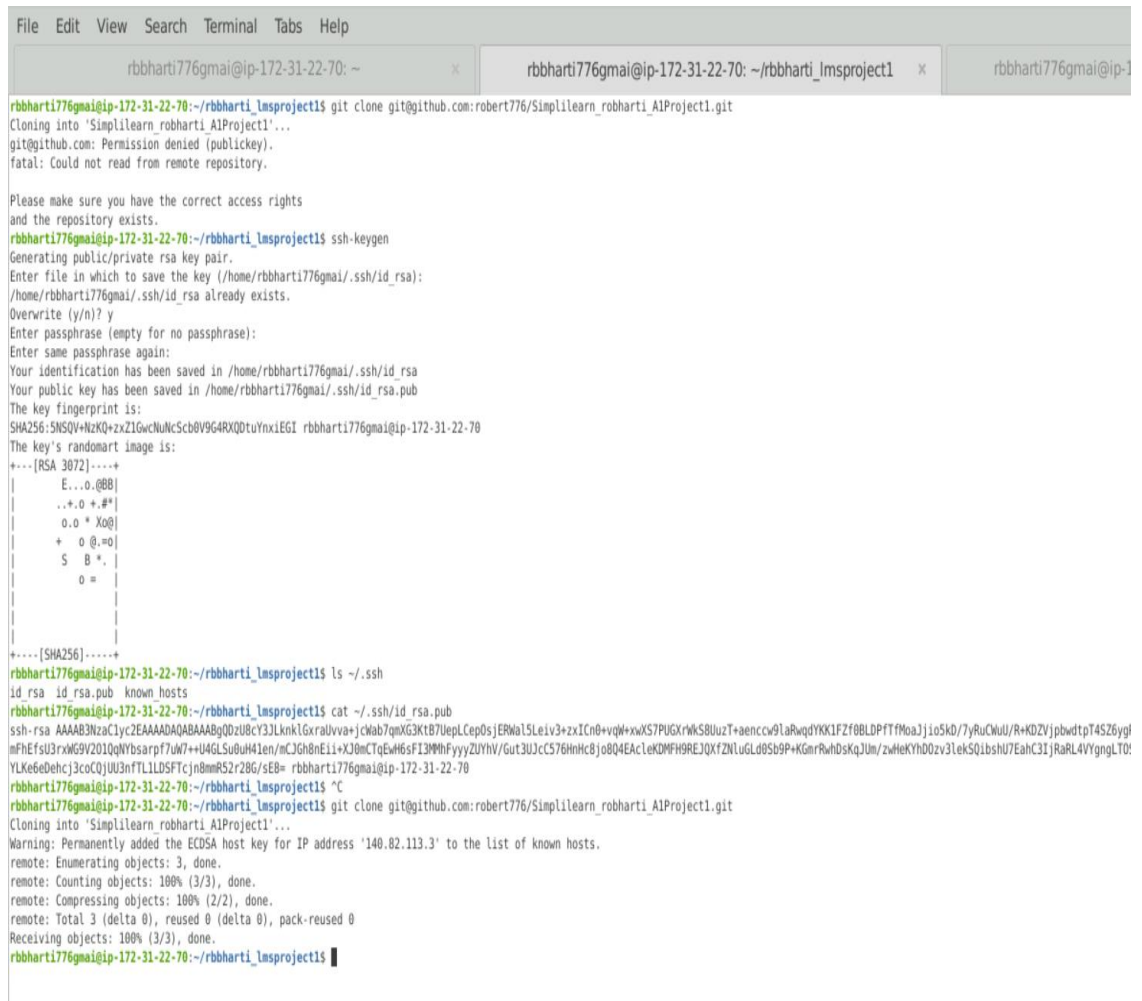


```
rbbharti776gmai@ip-172-31-22-70: ~/rbbharti_1msproject1
rbbharti776gmai@ip-172-31-22-70: ~
rbbharti776gmai@ip-172-31-22-70: ~$ git clone git@github.com:robert776/Simplilearn_robharti_AIProject1.git
Cloning into 'Simplilearn_robharti_AIProject1'...
Warning: Permanently added the ECDSA host key for IP address '140.82.114.3' to the list of known hosts.
git@github.com: Permission denied (publickey).
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
rbbharti776gmai@ip-172-31-22-70: ~/rbbharti_1msproject1$
```

Figure 14. ssh\_key\_missing\_ingit\_repo

- For fix below error, you need to generate ssh key, copy it and add to git repo.  
Below command and steps can help you.
  - Follow below step in your local machine



```
File Edit View Search Terminal Tabs Help
rbbharti776gmai@ip-172-31-22-70: ~
rbbharti776gmai@ip-172-31-22-70: ~/rbbharti_1msproject1
rbbharti776gmai@ip-172-31-22-70: ~/rbbharti_1msproject1$ git clone git@github.com:robert776/Simplilearn_robharti_AIProject1.git
Cloning into 'Simplilearn_robharti_AIProject1'...
git@github.com: Permission denied (publickey).
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
rbbharti776gmai@ip-172-31-22-70: ~/rbbharti_1msproject1$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/rbbharti776gmai/.ssh/id_rsa):
/home/rbbharti776gmai/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/rbbharti776gmai/.ssh/id_rsa
Your public key has been saved in /home/rbbharti776gmai/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:SN5QVH4zKQ+zxZ1GwCNUKScB8V9G4RXQ0tuYnxiEGi rbbharti776gmai@ip-172-31-22-70
The key's random image is:
+---[RSA 3072]----+
| E...o.@@B|
| ..+.o +.#*|
| o.o * Xo@|
| + o @.=o|
| S B *.|
| o =|
|
+---[SHA256]-----+
rbbharti776gmai@ip-172-31-22-70: ~/rbbharti_1msproject1$ ls ~/.ssh
id_rsa id_rsa.pub known_hosts
rbbharti776gmai@ip-172-31-22-70: ~/rbbharti_1msproject1$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGDzU8cY3JLknkL6xralVva+jcWab7qmXG3KtB7UepLCep0sJERMaLSLeiv3+zxICn0+vdq+xxS7PUGXrWkS8UuzT+aeccw9LaRwdYKK1FZF0BLDPFTfMoaJjio5Kd/7YRuCWuU/R+KDZVjpbwdtpT4S26ygl
mPhEfsU3rxM69V201QqNybSarpf7dW7++U4GLSu0uH41en/mCjGh8nE1i+XJ0mCTQ6wH6sF13MMhFyyZUYHv/Gut3UJCCS76Hhnc8joBQ4EAclEkDMFH9REJQXfZNLuGLd0Sb9P+KGrRwHdSkqJUm/zWekYhD0zv3lekSQibshU7EahC3IjRaRL4VYgngLT0i
YLKefedehcj3coC0jUu3nFTL1LD5FTcjn8mmR52r28G/sEB= rbbharti776gmai@ip-172-31-22-70
rbbharti776gmai@ip-172-31-22-70: ~/rbbharti_1msproject1$ ^C
rbbharti776gmai@ip-172-31-22-70: ~/rbbharti_1msproject1$ git clone git@github.com:robert776/Simplilearn_robharti_AIProject1.git
Cloning into 'Simplilearn_robharti_AIProject1'...
Warning: Permanently added the ECDSA host key for IP address '140.82.113.3' to the list of known hosts.
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
rbbharti776gmai@ip-172-31-22-70: ~/rbbharti_1msproject1$
```

Figure 15.clone\_repo\_inlocalsystem

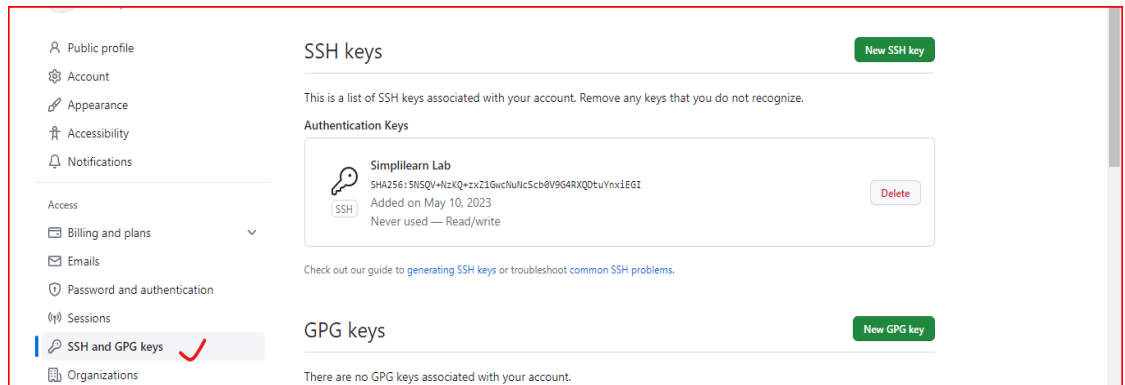


Figure 16.Add\_ssh\_key\_inGitRepo.

- Post after clone repo in local system, add your application and Dockerfile inside project folder and push it repo

```
Omit --global to set the identity only in this repository.
fatal: empty ident name (for <rbbharti776gmai@ip-172-31-22-70.ec2.internal>) not allowed
rbbharti776gmai@ip-172-31-22-70:~/rbbharti_lmsproject1/Simplilearn_robharti_A1Project1$ git config --global user.email "rbbharti776@gmail.com"
rbbharti776gmai@ip-172-31-22-70:~/rbbharti_lmsproject1/Simplilearn_robharti_A1Project1$ git config --global user.name "Robert Bharti"
rbbharti776gmai@ip-172-31-22-70:~/rbbharti_lmsproject1/Simplilearn_robharti_A1Project1$ git commit -m "java image & dockerfile has been pushed for create docker container"
[main ef573cd] java image & dockerfile has been pushed for create docker container
2 files changed, 21 insertions(+)
create mode 100644 Dockerfile
create mode 100644 hello_lmsapp.java
rbbharti776gmai@ip-172-31-22-70:~/rbbharti_lmsproject1/Simplilearn_robharti_A1Project1$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

nothing to commit, working tree clean
rbbharti776gmai@ip-172-31-22-70:~/rbbharti_lmsproject1/Simplilearn_robharti_A1Project1$ git push origin master
error: src refspec master does not match any
error: failed to push some refs to 'github.com:robert776/Simplilearn_robharti_A1Project1.git'
rbbharti776gmai@ip-172-31-22-70:~/rbbharti_lmsproject1/Simplilearn_robharti_A1Project1$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 742 bytes | 742.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:robert776/Simplilearn_robharti_A1Project1.git
7d5094a..ef573cd main -> main
```

Figure 17.commit\_code\_repo

- After successful commit change in repo, we need to configure our Jenkins pipeline for start CICD.

➤ **Configure Continuous Deployment and execute a Jenkins pipeline job and execute dockerfile respectively through a shall script.**

- For create Jenkins pipeline, Open Jenkins, create a freestyle project. Use the GitHub url as the SCM and In the build section execute docker build command to test whether Jenkins is able to execute it

- Jenkins > New Item > Freestyle project (BuildJob) - SCM > Git > Add your repo url

- Build > Add build step > Execute Shell

Command: **docker build -t my\_test\_image .**

- Save

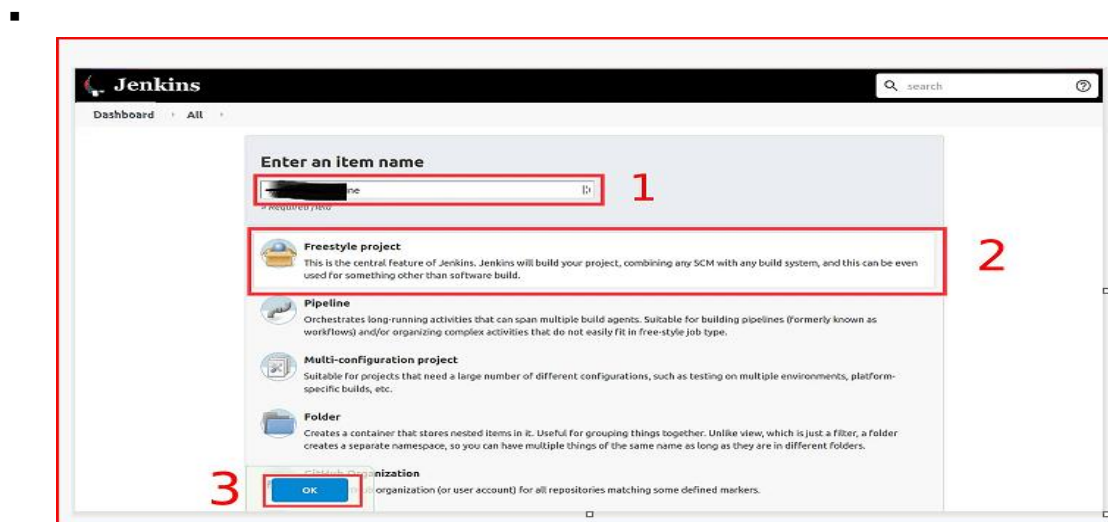


Figure 18.Create\_freestyle\_pipeline

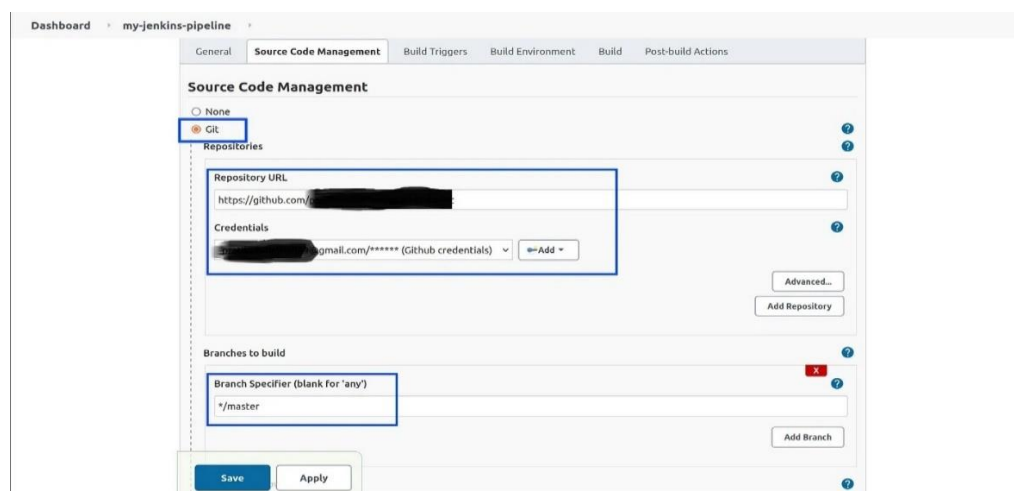


Figure 19.inserting\_git\_repo\_details\_inPipeline

- Once freestyle pipeline creation completed, click on Build bottom on left side panel and run the Job and Check the build console. If you get permission error, execute the below steps.

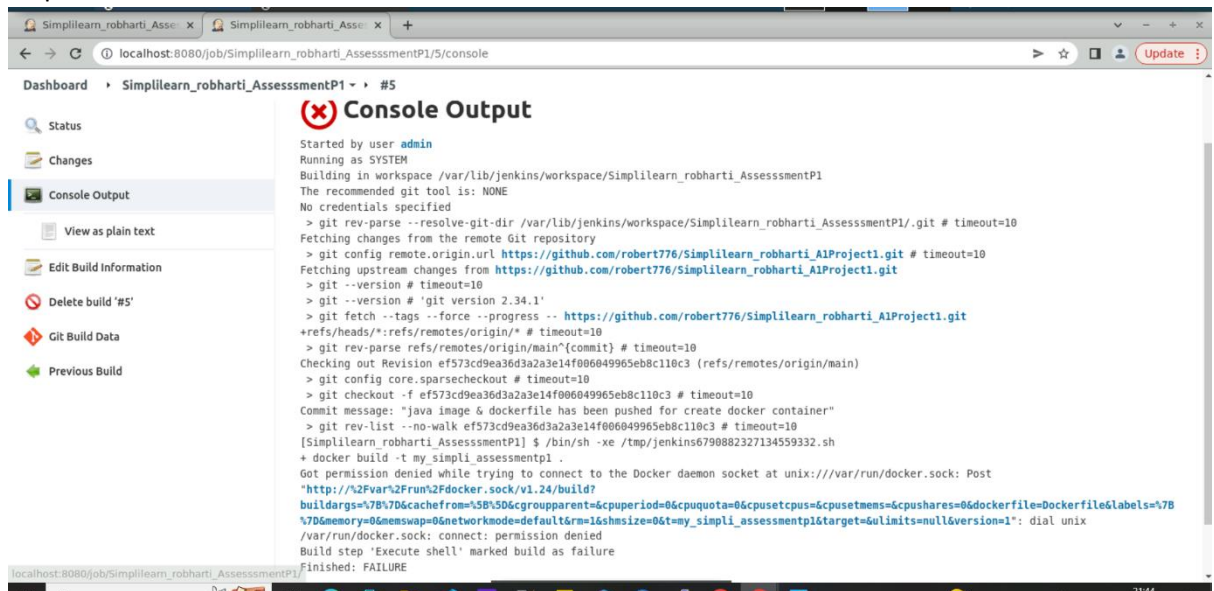


Figure 20. build\_failed\_dueToDockerpermission.

- For fix above error, run below command and restart Jenkins again from terminal.

```
sudo usermod -aG docker jenkins
sudo service jenkins restart
sudo chmod 777 /var/run/docker.sock
```

- Now, Re-login & build the job once again and see jenkins build logs.

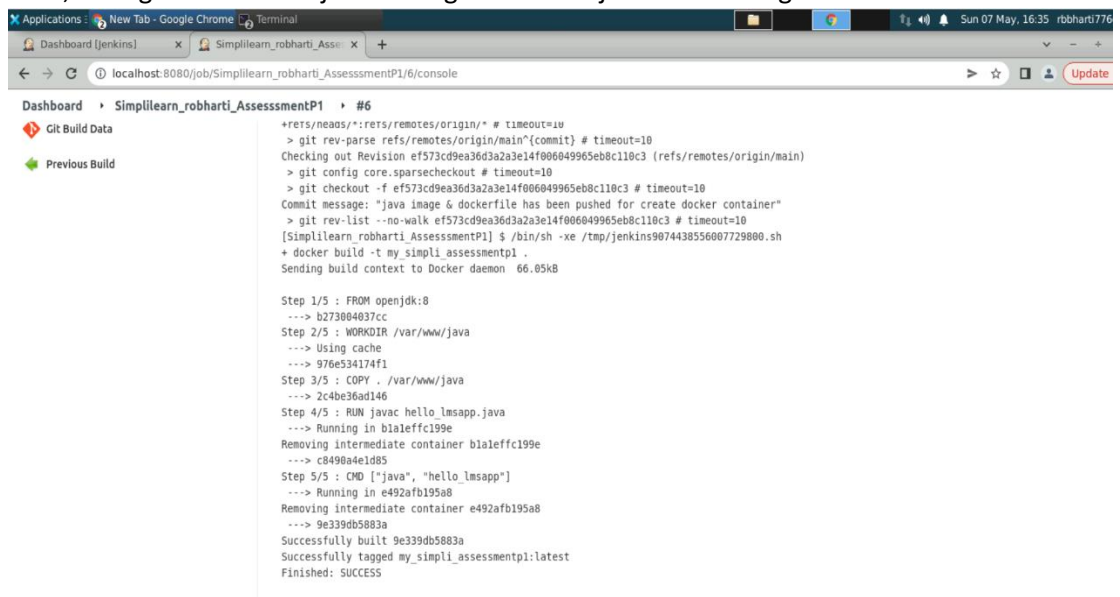


Figure 21 Build\_get\_success\_after\_permission.



## ➤ Use a Jenkins plugin "CloudBees Docker Build and Publish" for Continuous Deployment in docker hub.

- Above steps can we also complete by plugins "CloudBees Docker Build and Publish" provide by jenkins. For that we have to add plugins in jenkins

- > Jenkins > Manage Jenkins > Manage Plugins > Available(tab) > search for "CloudBees"

- > Select CloudBees Docker Build and Publish

- > Download now and Install after restart

- Restart Jenkins

- Relogin

- Follow below steps to add plugins

Let's install all the plugins that we will need. Click on "Manage Jenkins" in the left panel.

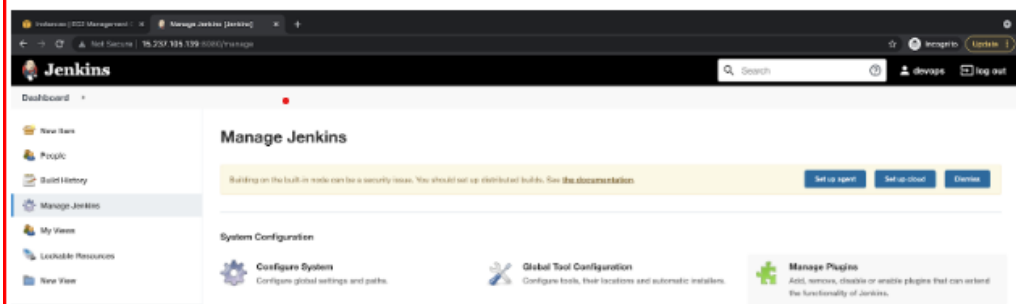


Figure 22.Add\_Plugin\_in\_jenkins

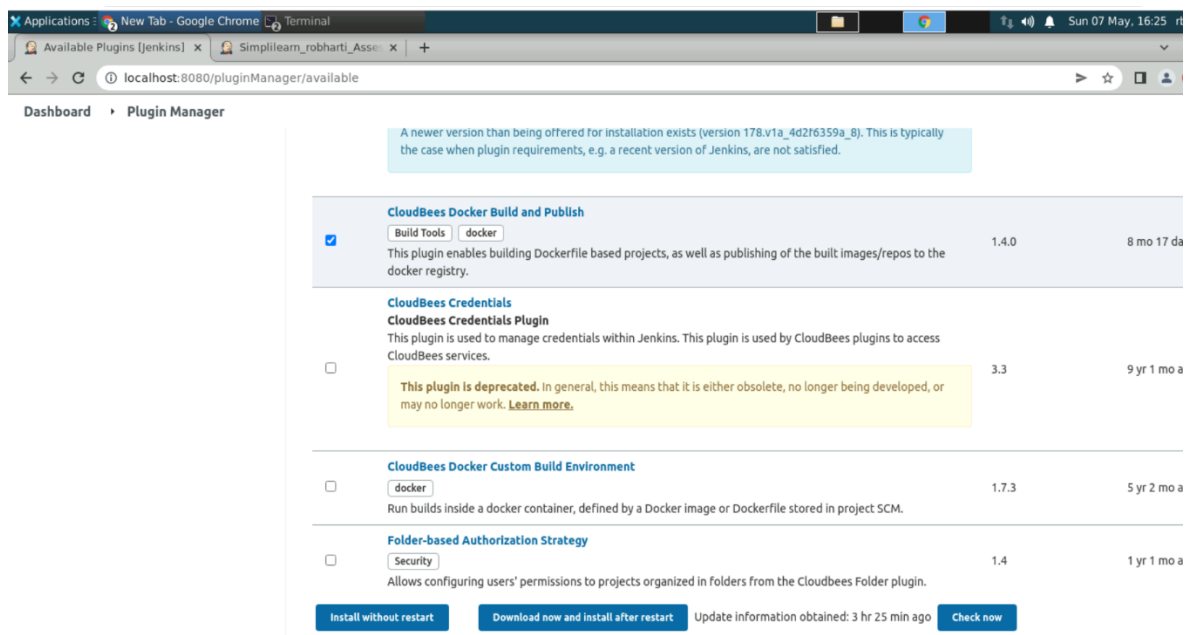


Figure 23.Add\_CloudBees\_plugin\_jenkins





- Add your docker credentials by follow below steps

click on “Manage Credentials”.

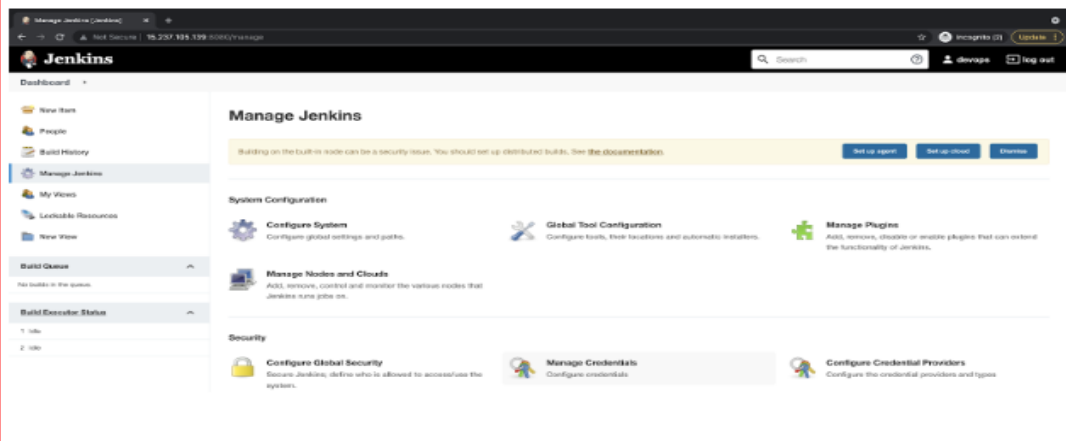


Figure 26.Add\_credential\_in\_jenkins\_1

- 

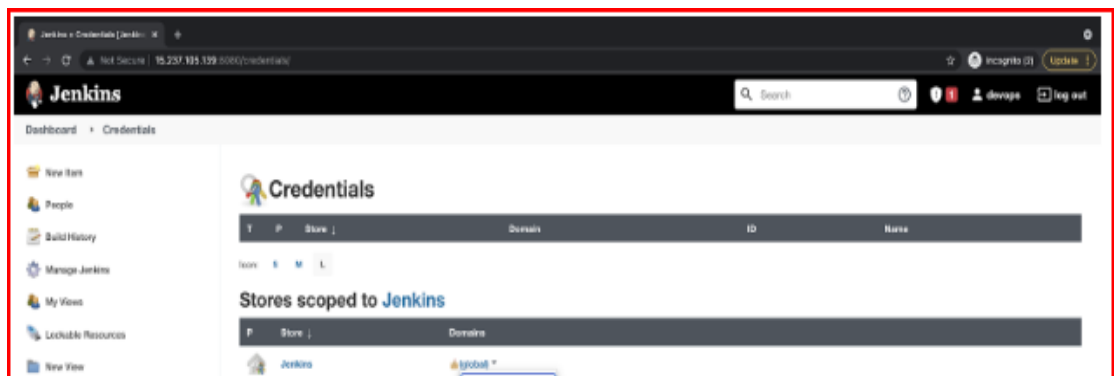


Figure 27..Add\_credential\_in\_jenkins\_2

- Post after configuration, click on save button. Now run build manually by click on build button on left side panel and see output in console.

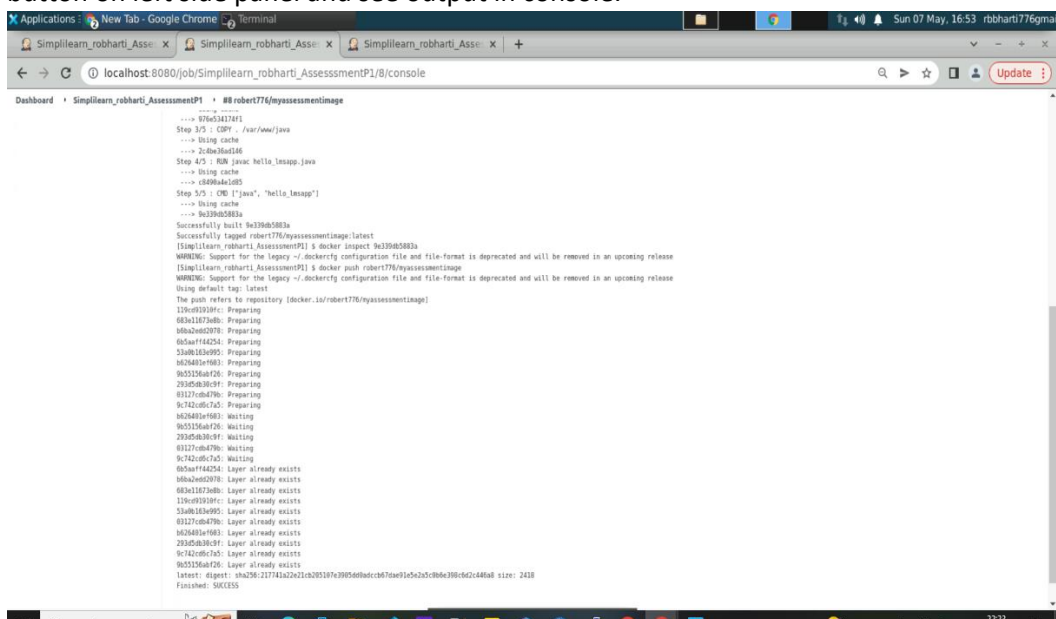


Figure 28.Build\_logs\_trigger\_with\_plugin

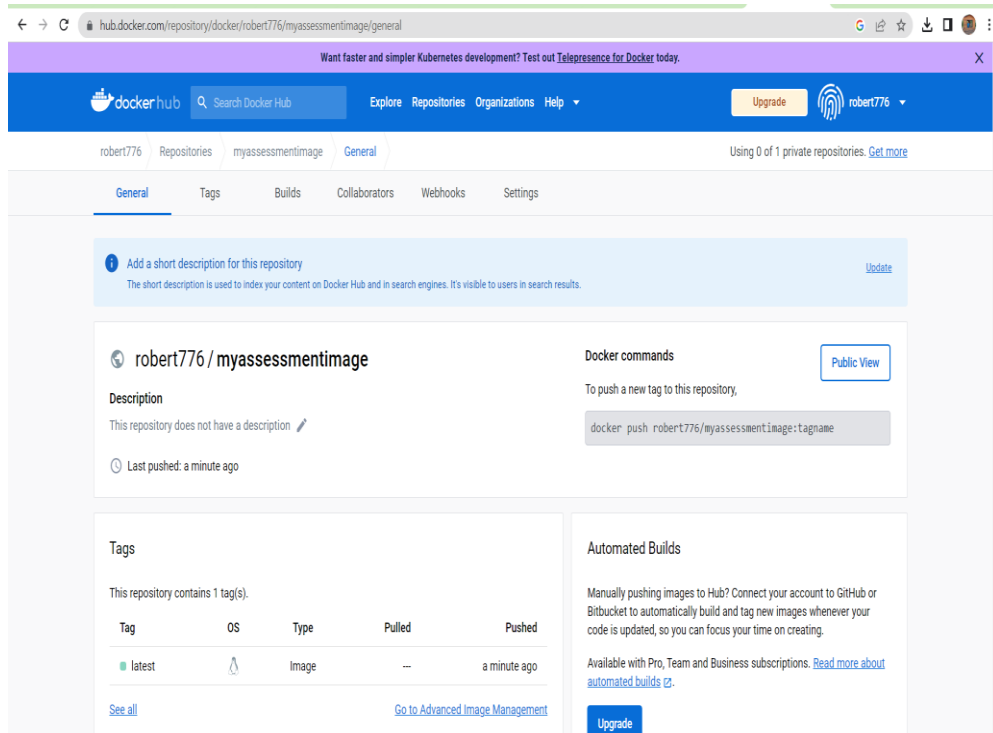


Figure 29.dockerHub\_details

- Check image tags in docker hub repo and match it in jenkins build logs.

## ➤ Automate the process using Jenkins Build Trigger

- Here for automate our build trigger, we use Poll SCM option, what this option does is that it continuously queries the VCS, based on a predefined schedule, for new changes.
- If new changes are encountered, then Jenkins will start the project build steps.
- The predefined schedule used here is \* \* \* \* \* which is a time schedule for every minute, Jenkins uses a syntax similar to Crontab, explore the format on [crontab.guru](https://crontab.guru)

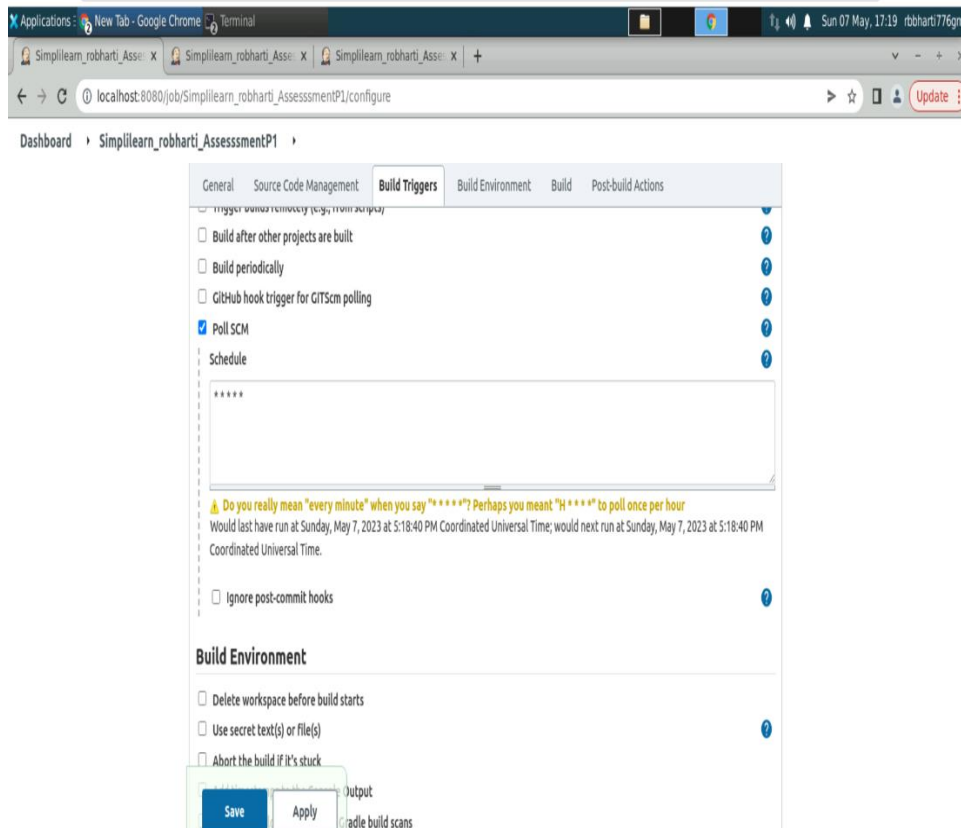


Figure 30.Poll\_SCM\_configuration\_for\_trigger

- After change and save, We are done setting up our project, the last step is to click Save at the bottom of screen and start the first build by clicking on the Build Now button.

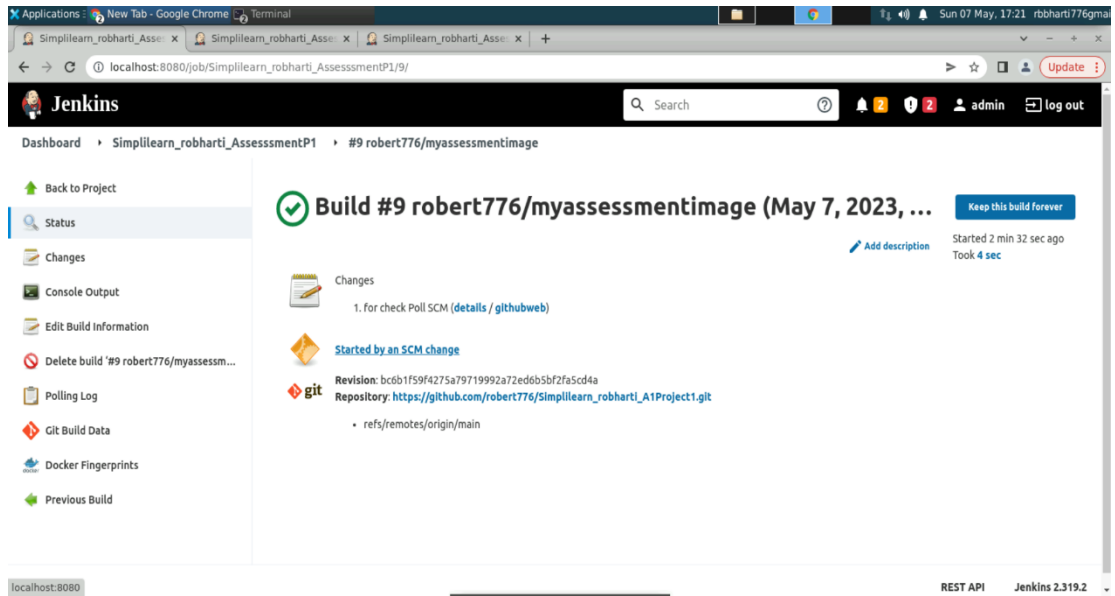


Figure 31. Build\_trigger\_pollSCM

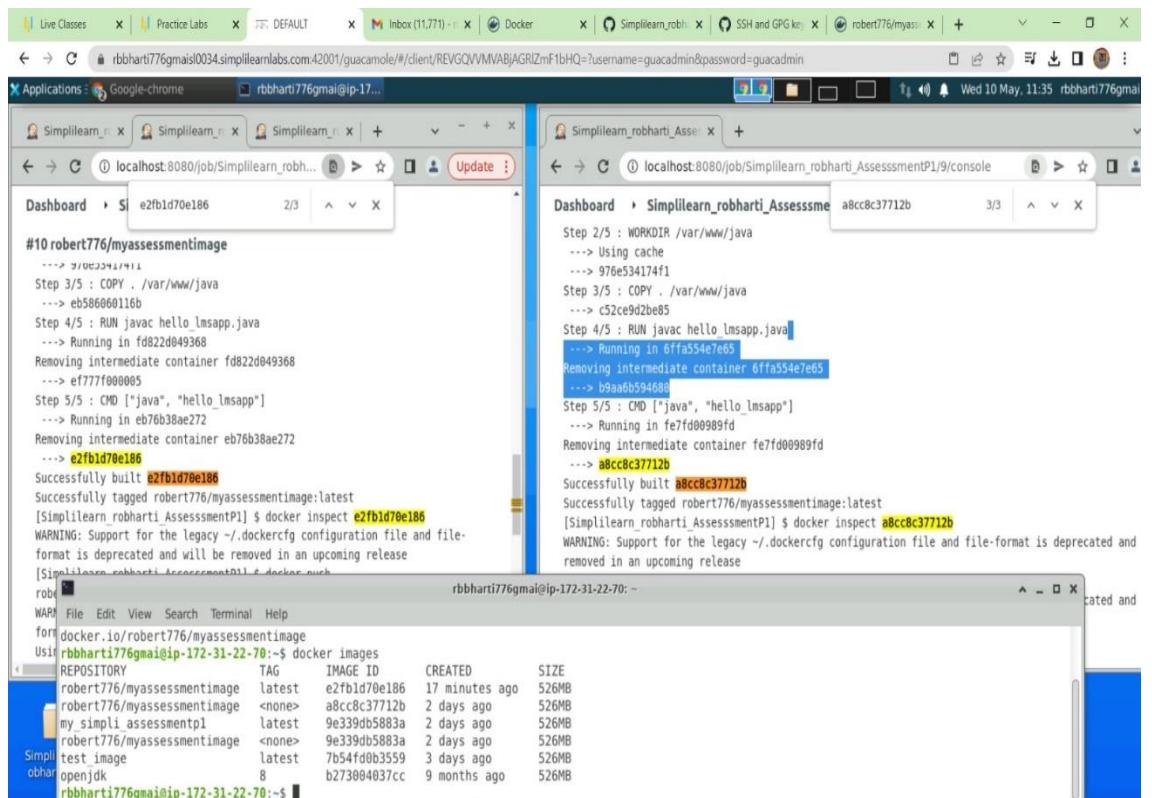


Figure 32. Final\_build\_details

## ❖ Solution overview:

- In this demonstrating, we have seen the continuous integration and delivery by building a Docker image and pushed it in docker hub using Jenkins Pipeline.
- On Linux (ubuntu) machine, we are installing and configuring Java & JDK, git, Jenkins server and docker so that app can be tested and build locally and further can be deployed.
- Excepted errors have been highlighted while configuration of CICD pipeline and provide solution how to fix it.
- Git poll scm play important role in identify any change or commit in repo periodically and trigger Jenkins build.
- Jenkins is a better choice to automate the Docker image build and pushing image to cloud repository provider.
- Docker follows the Domain registry concept to push and the images and have all images tags history also with it.