

- 1) create a user Jack with unilao password and find jack user from /etc/passwd file along with their UID and home directory. and assign sudo delegation for jack user. So, he can create a user with name Barbosa.

```
[root@ip-172-31-10-62 ~]# useradd Jack
[root@ip-172-31-10-62 ~]# passwd Jack
Changing password for user Jack.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: all authentication tokens updated successfully.
```

```
[root@ip-172-31-10-62 ~]# cat /etc/passwd | grep -i Jack
Jack:x:1001:1001:~/home/Jack:/bin/bash
```

```
[root@ip-172-31-10-62 ~]# whereis useradd
useradd: /usr/sbin/useradd /usr/share/man/man8/useradd.8.gz
[root@ip-172-31-10-62 ~]# whereis passwd
passwd: /usr/bin/passwd /etc/passwd /usr/share/man/man1/passwd.1.gz /usr/share/man/man5/passwd.5.gz
[root@ip-172-31-10-62 ~]# vim /etc/sudoers
[root@ip-172-31-10-62 ~]# su - Jack
[Jack@ip-172-31-10-62 ~]$ sudo useradd Barbosa

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

For security reasons, the password you type will not be visible.
[sudo] password for Jack:
```

```
[Jack@ip-172-31-10-62 ~]$ cat /etc/passwd | grep -i Barbosa
Barbosa:x:1002:1002:~/home/Barbosa:/bin/bash
[Jack@ip-172-31-10-62 ~]$ ~|
```

- 2) For an existing project on GitHub, create a Jenkins project. This project should be triggered automatically whenever there is any change in any of the file in GitHub repository. Also make sure that the artifacts should be built and saved in the pipeline.

I have set our hostname as jenkins-server.example.com

And this is the installation process of jenkins

```
[root@ip-172-31-10-62 ~]# hostnamectl set-hostname jenkins-server.example.com
[root@ip-172-31-10-62 ~]# bash
[root@jenkins-server ~]# hostname
jenkins-server.example.com
[root@jenkins-server ~]# yum update -y
Amazon Linux 2023 Kernel Livepatch repository                217 kB/s | 23 kB    00:00
Dependencies resolved.
Nothing to do.
Complete!
[root@jenkins-server ~]# yum install wget -y
Last metadata expiration check: 0:01:26 ago on Sat Sep 27 10:57:08 2025.
Package wget-1.21.3-1.amzn2023.0.4.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
```

```
[root@jenkins-server ~]# wget -O /etc/yum.repos.d/jenkins.repo \
https://pkg.jenkins.io/redhat-stable/jenkins.repo
--2025-09-27 10:58:43-- https://pkg.jenkins.io/redhat-stable/jenkins.repo
Resolving pkg.jenkins.io (pkg.jenkins.io)... 146.75.38.133, 2a04:4e42:78::645
Connecting to pkg.jenkins.io (pkg.jenkins.io)|146.75.38.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 85
Saving to: '/etc/yum.repos.d/jenkins.repo'

/etc/yum.repos.d/jenkins.repo  100%[=====] 85 --.-KB/s  in 0s

2025-09-27 10:58:43 (1.20 MB/s) - '/etc/yum.repos.d/jenkins.repo' saved [85/85]

[root@jenkins-server ~]# rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
[root@jenkins-server ~]# yum upgrade
Jenkins-stable
Dependencies resolved.
Nothing to do.
Complete!
```

```
Installed:
jenkins-2.516.3-1.1.noarch

Complete!
```

Finally Jenkins is installed.

And after setting the Jenkins its set to open

# Unlock Jenkins

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
```

Please copy the password from either location and paste it below.

Administrator password

.....

Continue

Webhook is created in github after generating token from jenkins

## Webhooks

Add webhook

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

✓ <http://34.235.150.123:8080/github-...> (push)

Last delivery was successful.

Edit

Delete

Copy the link of git repository

The screenshot shows a GitHub repository page for 'java-code-with-maven', which is a public repository forked from 'sanjayguruji/java-code-with-maven'. The repository is currently on the 'main' branch, which is up to date with the upstream. The file list on the left includes 'server', 'webapp', 'Dockerfile', 'README.md', 'pom.xml', and 'regapp-deploy.yml'. The 'Clone' dropdown menu is open, showing options for cloning the repository using HTTPS, SSH, or GitHub CLI. The HTTPS URL is 'https://github.com/Rahulpr89/java-code-with-maven'. Other options include 'Open with GitHub Desktop' and 'Download ZIP'.

java-code-with-maven Public

forked from sanjayguruji/java-code-with-maven

main 1 Branch 0 Tags

Go to file t Add file <> Code

This branch is up to date with sanjayguruji/java-code-with-maven:main .

sanjayguruji Update index.jsp

- server -a
- webapp Update index.jsp
- Dockerfile Update Dockerfile
- README.md -a
- pom.xml -a
- regapp-deploy.yml -a

Clone

HTTPS SSH GitHub CLI

https://github.com/Rahulpr89/java-code-with-maven

Clone using the web URL.

Open with GitHub Desktop

Download ZIP

Console output is success



## Console Output

[Download](#)[Copy](#)[View as plain text](#)

```
Started by user Rahul
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/java-project
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/Rahulpr89/java-code-with-maven.git
> git init /var/lib/jenkins/workspace/java-project # timeout=10
Fetching upstream changes from https://github.com/Rahulpr89/java-code-with-maven.git
> git --version # timeout=10
> git --version # 'git version 2.50.1'
> git fetch --tags --force --progress -- https://github.com/Rahulpr89/java-code-with-maven.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/Rahulpr89/java-code-with-maven.git # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
```

```
project/server/1.0-SNAPSHOT/server-1.0-SNAPSHOT.jar
[JENKINS] Archiving /var/lib/jenkins/workspace/java-project/
project/1.0-SNAPSHOT/maven-project-1.0-SNAPSHOT.pom
channel stopped
Finished: SUCCESS
```

This is the final output

```
[root@jenkins-server target]# cd /var/lib/jenkins/workspace/java-project/webapp/target
[root@jenkins-server target]# ll
total 4
drwxr-xr-x. 2 jenkins jenkins 28 Sep 27 11:45 maven-archiver
drwxr-xr-x. 2 jenkins jenkins 6 Sep 27 11:45 surefire
drwxr-xr-x. 4 jenkins jenkins 54 Sep 27 11:45 webapp
-rw-r--r--. 1 jenkins jenkins 2390 Sep 27 11:45 webapp.war
```

This ensures that build is automated

Triggers

Set up automated actions that start your build based on specific events, like code changes or scheduled times.

- ☐ Build whenever a SNAPSHOT dependency is built ?
- ☐ Trigger builds remotely (e.g., from scripts) ?
- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☐ GitHub Branches
- ☐ GitHub Pull Requests ?
- ☒ GitHub hook trigger for GITScm polling ?
- ☐ Poll SCM ?

Add description

All +

S	W	Name ↓	Last Success	Last Failure	Last Duration	
		java-project	8 min 43 sec <a href="#">#2</a>	10 min <a href="#">#1</a>	23 sec	



java-project

Add description



[Latest Test Result](#) (no failures)

Permalinks

- [Last build \(#2\), 9 min 3 sec ago](#)
- [Last stable build \(#2\), 9 min 3 sec ago](#)
- [Last successful build \(#2\), 9 min 3 sec ago](#)
- [Last failed build \(#1\), 11 min ago](#)
- [Last unsuccessful build \(#1\), 11 min ago](#)
- [Last completed build \(#2\), 9 min 3 sec ago](#)

