

## **DEVOPS PROJECT**

### **CI/CD PIPELINE**

#### **CONTINUOUS INTEGRATION AND CONTINUOUS DEPLOYMENT**

#### **TOOLS: GIT, DOCKER, JENKINS.**

**Continuous deployment and continuous integration is a process.**

**Automation is a major one to do the automated work.**

**Automation-continuous process.**

**Devops is a methodology.**

**Devops is a development and operation.**

**Based on the code we develop the application.**

#### **Continuous integration**

**Build the code, test the code if I face the mistake in a code, then test the code, then again test the code if we face any error in the code. (it includes the build and test stage) – continuous integration happens in build and test stage.**

#### **Continuous delivery/ deployment**

**It focuses on release and deploy. It involves the release of code.**

#### **Stages**

**1. development**

**2. testing**

**3. quality assurance staging (pre production)**

#### **4.production**

**If I done the deployment manually then it is a continuous delivery.**

**Build,test,staging,deploy to a production manually is continuous delivery.**

**Manual deployment that occurs in production is known as continuous delivery.(deploying upto staging is continous delivery).**

**Automatic deployment occurs in a production is known as Continuous deployment.**

**Jenkins is used for integrate with all other tools.**

**Jenkins is an open source tools.**

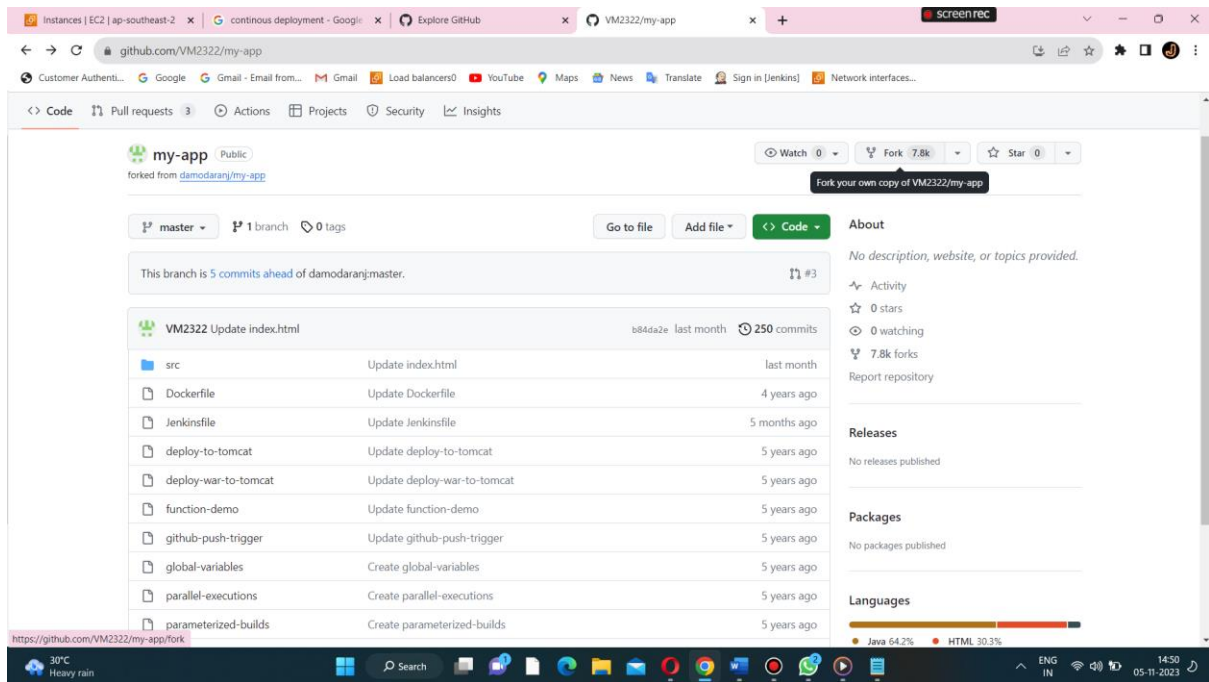
**Maven is used to build the application.**

**Sonarqube is used for code quality check.**

**Nexus is used for artifact.**

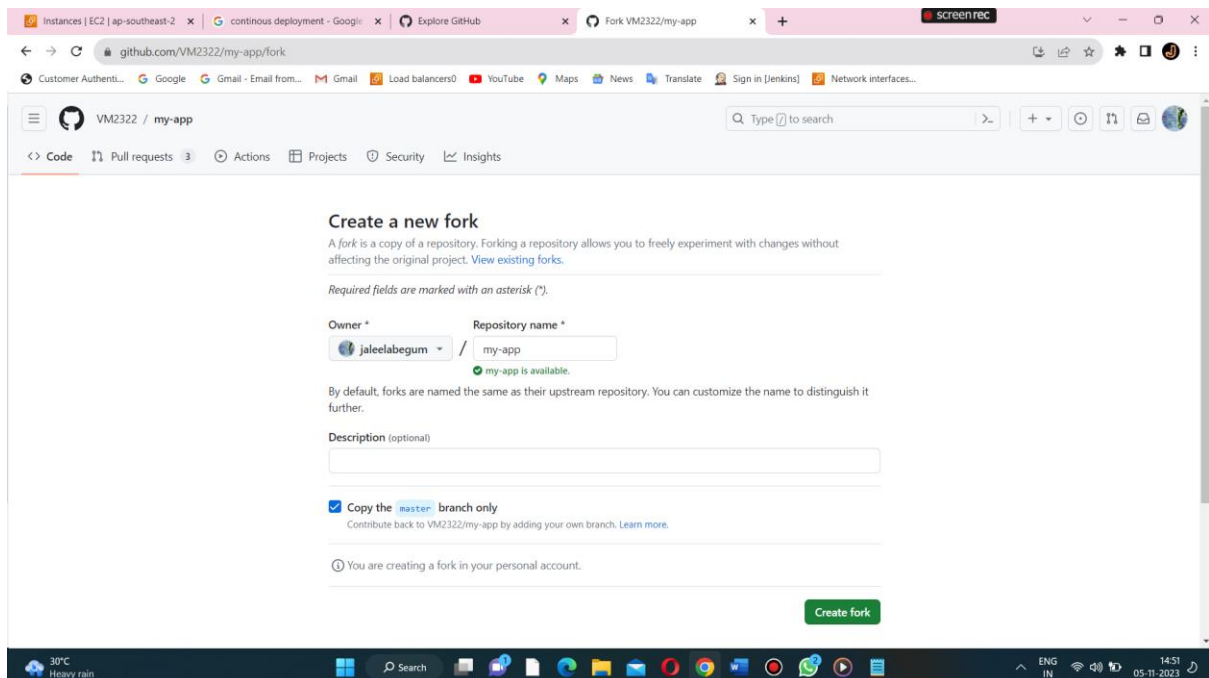
#### **STAGES IN CI/CD**

##### **1.source code management(SCM)**

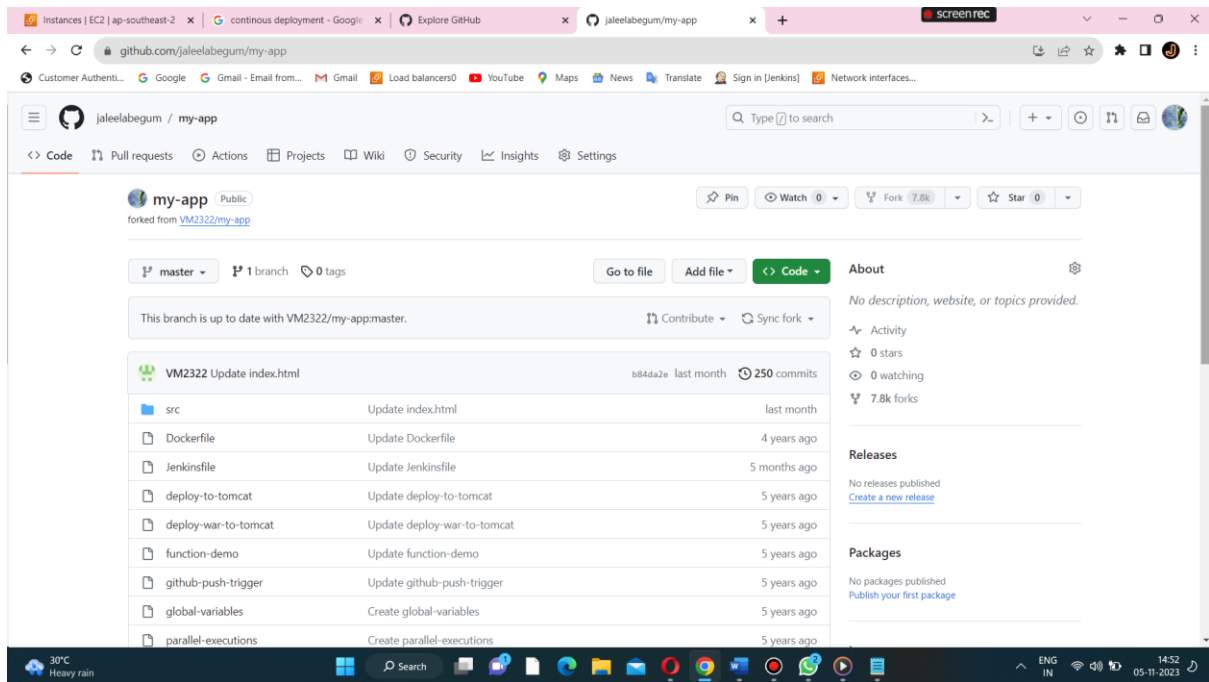


Git and github(fork from this website)

<https://github.com/VM2322/my-app>



Click on create fork.



**That particular repository is fork into our project(to copy the from one repository into our repository).**

**Its present inside the master branch.**

## **SCM CHECKOUT**

### **STEP2:**

**Jenkins installation(t2.medium)-4gb ram(for installing four tools).**

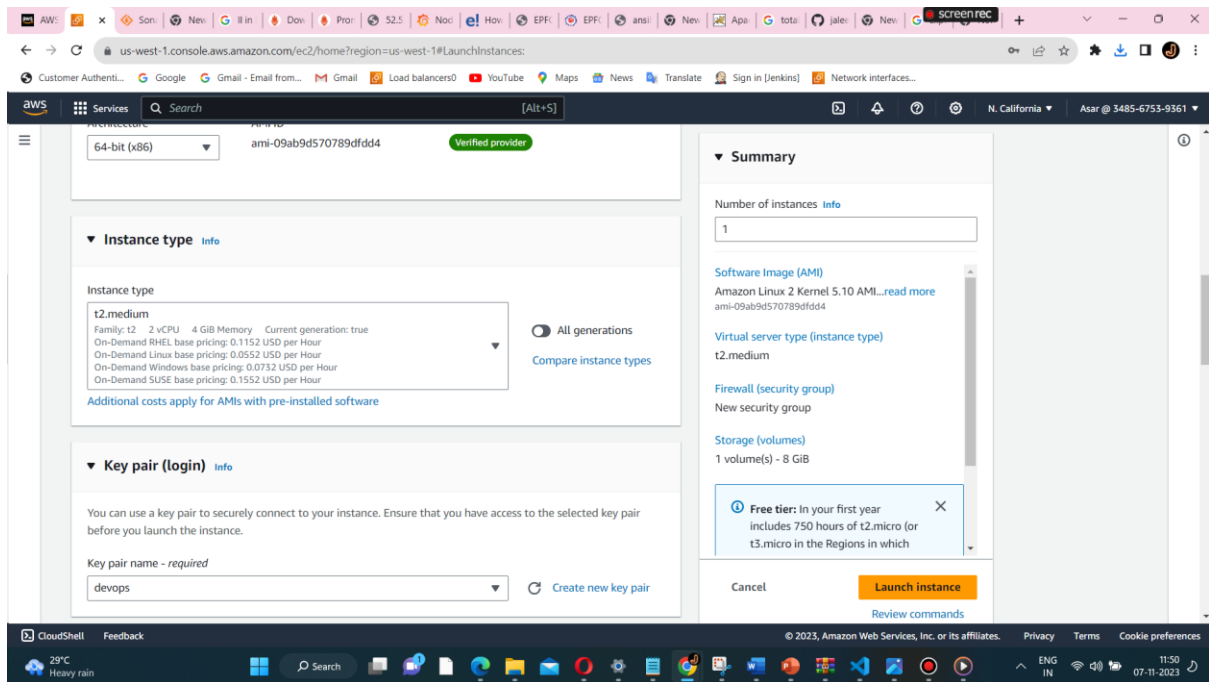
**Number of servers used for ci/cd process:**

**Have a three typical servers for ci/cd implementation**

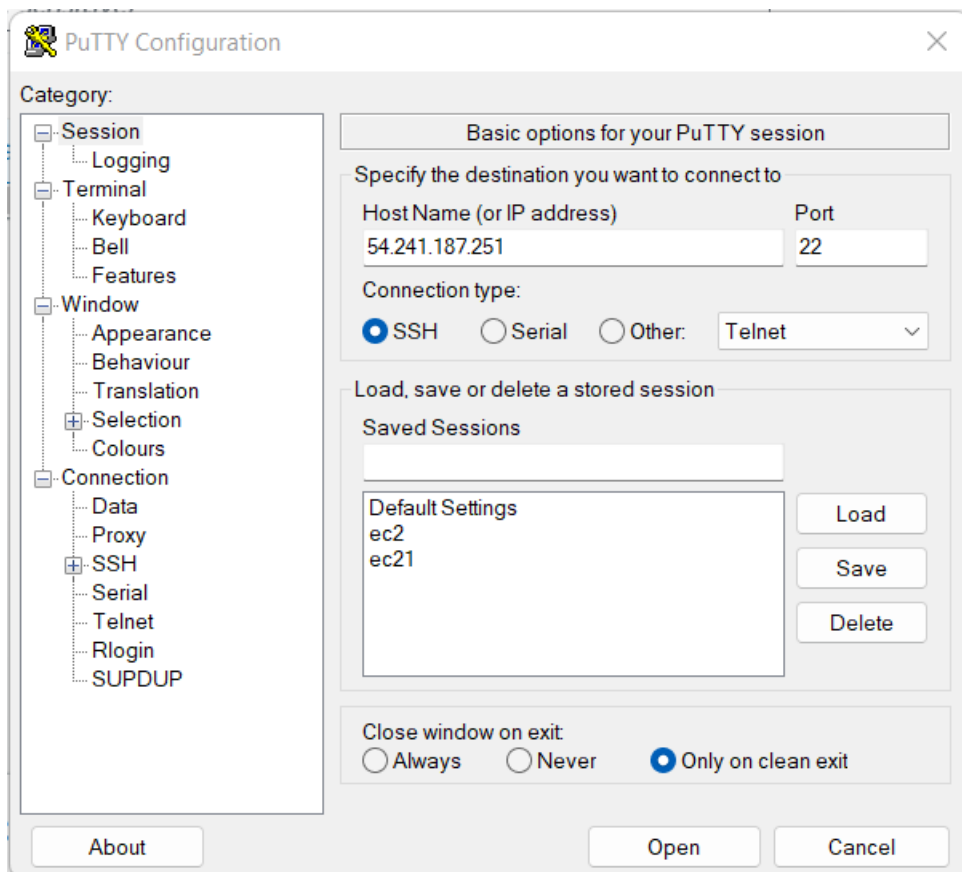
**Server1: Jenkins,git,maven,docker (4GB RAM)**

**Server2: sonaube**

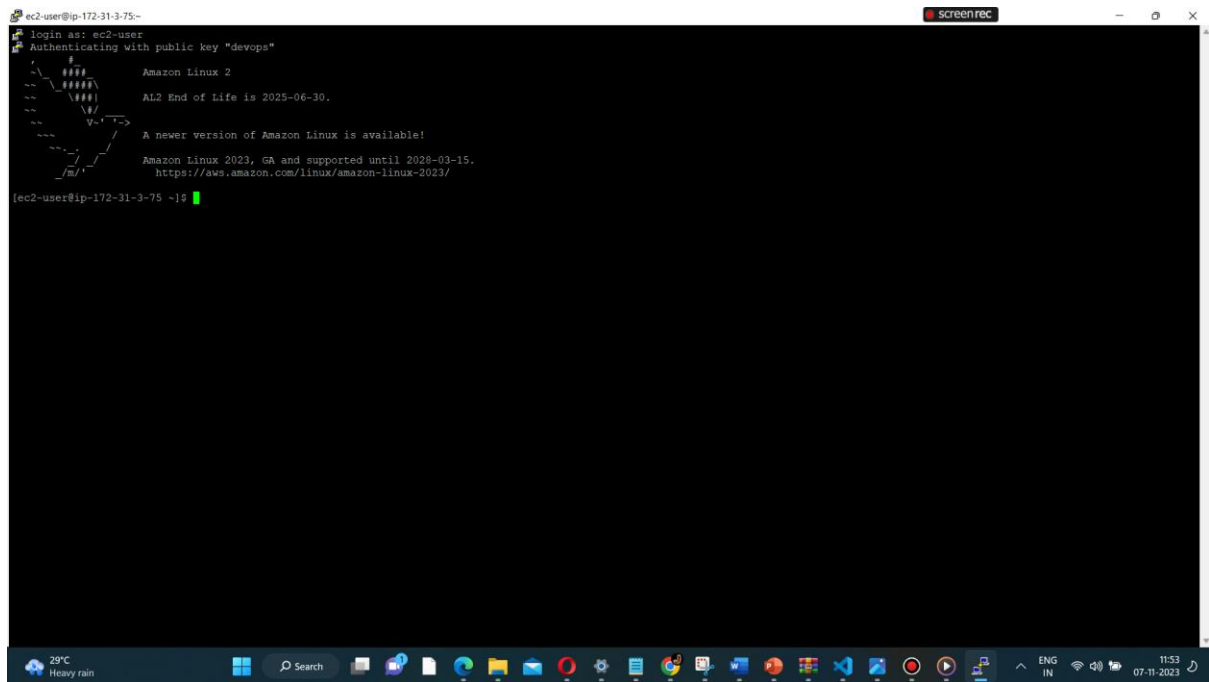
**Server3:nexus(artifact).**



**Launch the instances in the instance type t2.medium.**



**Connect the instance using key pair.**



**Now instance is launched successfully.**

## COMMANDS:

1. `amazon-linux-extras install epel` (for to install the new packages)
2. `yum update -y`
3. `wget -O /etc/yum.repos.d/jenkins.repo`  
<https://pkg.jenkins.io/redhat-stable/jenkins.repo>
4. `rpm --import` <https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key>
5. `amazon-linux-extras install java-openjdk11`
6. `yum install jenkins`

```
root@ip-172-31-3-75:/home/ec2-user
67 awscli1 available [ =stable ]
68 php8.2 available [ =stable ]
69 dnsmasq available [ =stable ]
70 unbound1:7 available [ =stable ]
72 collectd-python3 available [ =stable ]
+ Note on end-of-support. Use 'info' subcommand.
[root@ip-172-31-3-75 ec2-user]# java --version
openjdk 11.0.20 2023-07-18 LTS
OpenJDK Runtime Environment (Red_Hat-11.0.20.0.8-1.amzn2.0.1) (build 11.0.20+8-LTS)
OpenJDK 64-Bit Server VM (Red_Hat-11.0.20.0.8-1.amzn2.0.1) (build 11.0.20+8-LTS, mixed mode, sharing)
[root@ip-172-31-3-75 ec2-user]# yum install jenkins
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
227 packages excluded due to repository priority protections
Resolving Dependencies
--> Running transaction check
--> Package jenkins.noarch 0:2.414.3-1.1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

Package Arch Version Repository Size
-----
jenkins noarch 2.414.3-1.1 jenkins 85 M

Transaction Summary
Install 1 Package

Total download size: 85 M
Installed size: 85 M
Is this ok [y/d/N]: y
Downloading packages:
jenkins-2.414.3-1.1.noarch.rpm | 85 MB 00:00:03
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Installing : jenkins-2.414.3-1.1.noarch 1/1
Verifying : jenkins-2.414.3-1.1.noarch 1/1
Installed:
jenkins.noarch 0:2.414.3-1.1

Complete!
[root@ip-172-31-3-75 ec2-user]# jenkins --version
2.414.3
[root@ip-172-31-3-75 ec2-user]#
```

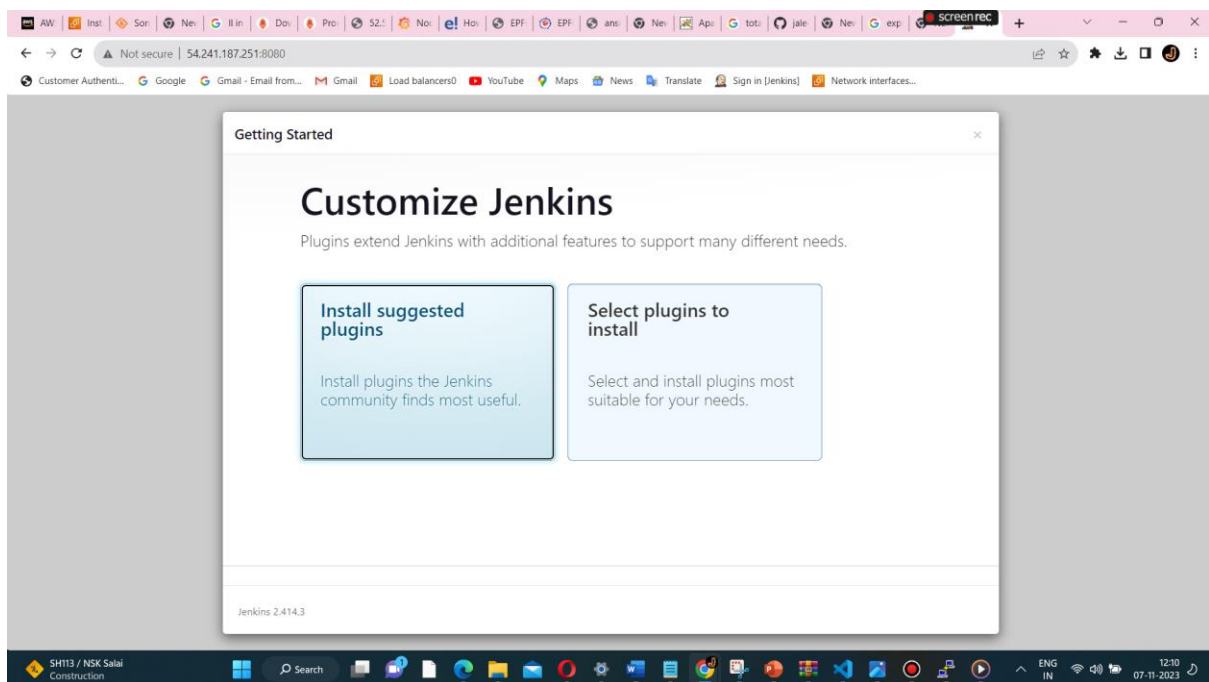
**Install the Jenkins inorder to integrate with git.**

**To start the Jenkins using this commands.**

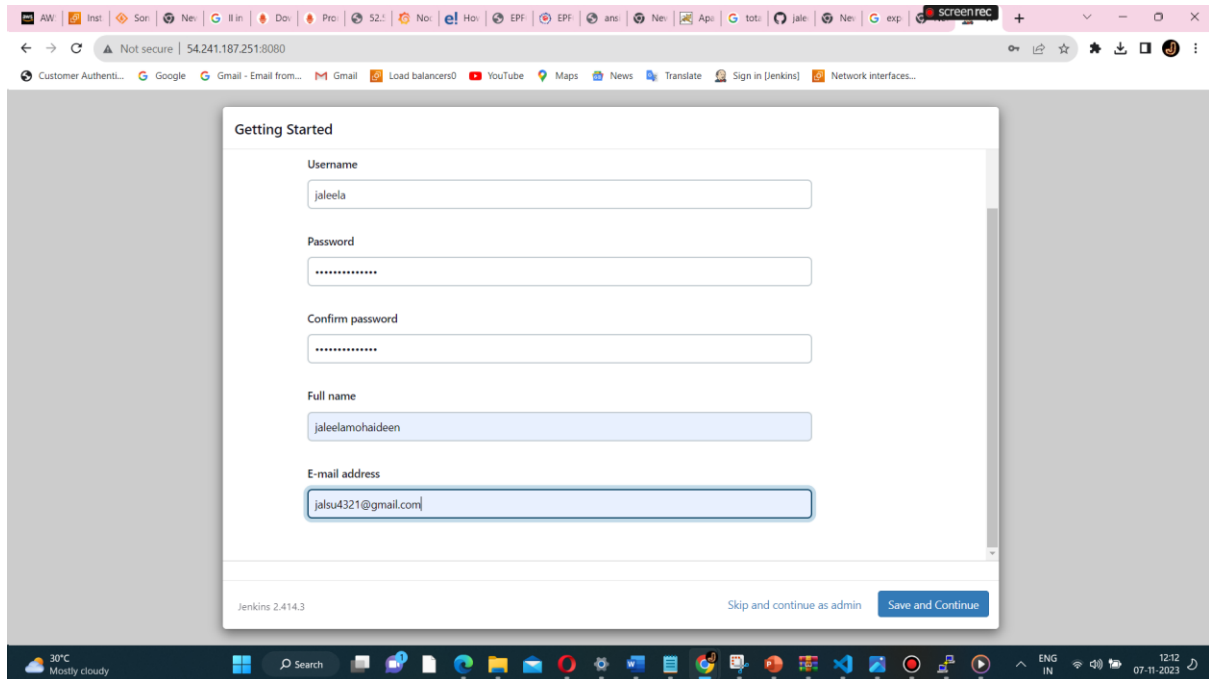
**1.sudo systemctl enable Jenkins**

**2.sudo systemctl start Jenkins**

**3.sudo systemctl status Jenkins**



## Jenkins installation to install the suggested plugins.



Getting Started

Username  
jaleela

Password  
\*\*\*\*\*

Confirm password  
\*\*\*\*\*

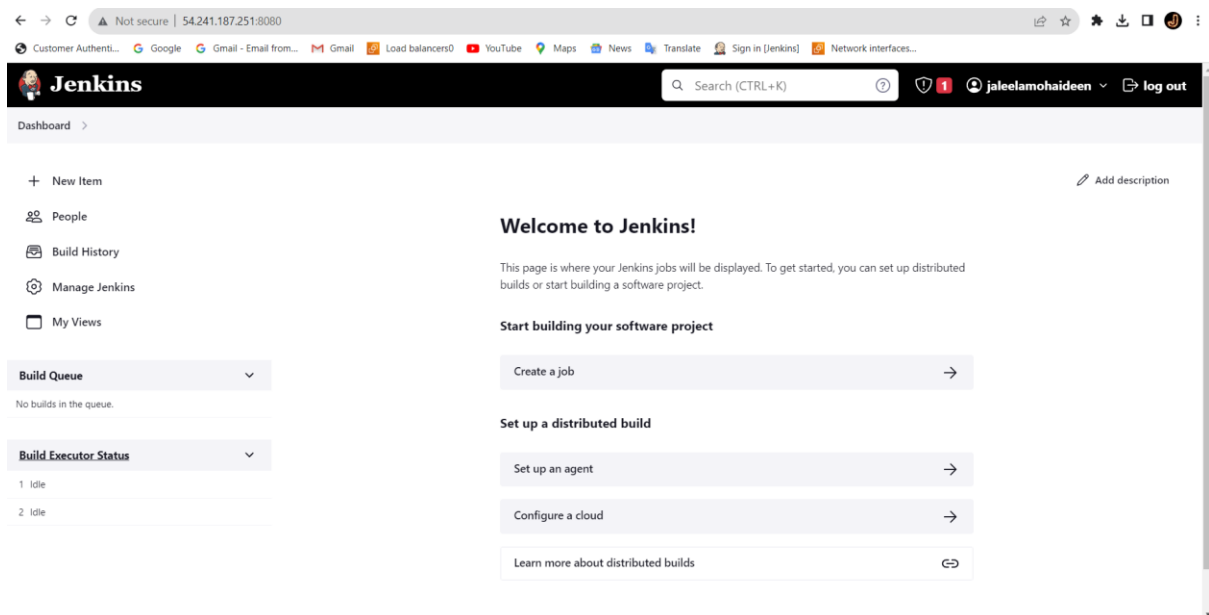
Full name  
jaleelamohaideen

E-mail address  
jalsu4321@gmail.com

Jenkins 2.414.3

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

Click on save and continue.



Dashboard

+ New Item

People

Build History

Manage Jenkins

My Views

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job →

Set up a distributed build

Set up an agent →

Configure a cloud →

Learn more about distributed builds ↗

Then the Jenkins dashboard got displayed.



## STEP3: STAGING

Once we need to integrate with Jenkins we need to install Jenkins and configure the home path.

← → ↻ Not secure | 54.241.187.251:8080/view/all/newJob

Customer Authent... Google Gmail - Email from... Gmail Load balancers0 YouTube Maps News Translate Sign in [Jenkins] Network interfaces...

Dashboard > All >

### Enter an item name

pipeline

» Required field

- Freestyle project**  
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
- Pipeline**  
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Multi-configuration project**  
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
- Folder**  
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
- Multibranch Pipeline**  
Creates a set of Pipeline projects according to detected branches in one SCM repository.

OK Cancel

Dashboard > pipeline > Configuration

### Configure

- General
- Advanced Project Options
- Pipeline

#### Pipeline

Definition

Pipeline script

Script ?

1

Try sample Pipeline...

☒ Use Groovy Sandbox ?

[Pipeline Syntax](#)

Save Apply

Need to write the script here.

Dashboard > pipeline > Configuration

Configure

General

Advanced Project Options

Pipeline

Definition

Pipeline script

Script ?

```

1- node{
2-   stage('SON Checkout'){
3-     git 'https://github.com/damodaranj/my-app.git'
4-   }
5- }

```

try sample Pipeline...

☒ Use Groovy Sandbox ?

[Pipeline Syntax](#)

Save

Apply

To add the staging file here

```

root@ip-172-31-3-75:/home/ec2-user
--> Running transaction check
--> Package perl-Error.noarch 1:0.17020-2.amzn2 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package                                Arch                                Version                                Repository                                Size
=====
Installing:
git                                     x86_64                                2.40.1-1.amzn2.0.1                    amzn2-core                                54 k
Installing for dependencies:
git-core                               x86_64                                2.40.1-1.amzn2.0.1                    amzn2-core                                10 M
git-core-doc                           noarch                                2.40.1-1.amzn2.0.1                    amzn2-core                                3.0 M
perl-Error                              noarch                                1:0.17020-2.amzn2                     amzn2-core                                32 k
perl-Git                                noarch                                2.40.1-1.amzn2.0.1                    amzn2-core                                41 k
perl-TermReadKey                        x86_64                                2.30-20.amzn2.0.2                     amzn2-core                                31 k
=====

Transaction Summary
Install 1 Package (+5 dependent packages)
Total download size: 13 M
Installed size: 44 M
Downloading packages:
(1/6): git-2.40.1-1.amzn2.0.1.x86_64.rpm                                | 54 kB 00:00:00
(2/6): git-core-doc-2.40.1-1.amzn2.0.1.noarch.rpm                      | 3.0 MB 00:00:00
(3/6): perl-Error-0.17020-2.amzn2.noarch.rpm                            | 32 kB 00:00:00
(4/6): perl-Git-2.40.1-1.amzn2.0.1.noarch.rpm                           | 41 kB 00:00:00
(5/6): git-core-2.40.1-1.amzn2.0.1.x86_64.rpm                          | 10 MB 00:00:00
(6/6): perl-TermReadKey-2.30-20.amzn2.0.2.x86_64.rpm                   | 31 kB 00:00:00
Total                                                                    58 MB/s | 13 MB 00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : git-core-2.40.1-1.amzn2.0.1.x86_64                      1/6
  Installing : git-core-doc-2.40.1-1.amzn2.0.1.noarch                  2/6
  Installing : 1:perl-Error-0.17020-2.amzn2.noarch                     3/6
  Installing : perl-TermReadKey-2.30-20.amzn2.0.2.x86_64               4/6
  Installing : perl-Git-2.40.1-1.amzn2.0.1.noarch                     5/6
  Installing : git-2.40.1-1.amzn2.0.1.x86_64                           6/6
  Verifying  : perl-TermReadKey-2.30-20.amzn2.0.2.x86_64              1/6
  Verifying  : git-core-2.40.1-1.amzn2.0.1.x86_64                    2/6
  Verifying  : git-core-doc-2.40.1-1.amzn2.0.1.noarch                 3/6
  Verifying  : perl-Git-2.40.1-1.amzn2.0.1.noarch                     4/6
  Verifying  : 1:perl-Error-0.17020-2.amzn2.noarch                    5/6
  Verifying  : git-2.40.1-1.amzn2.0.1.x86_64                          6/6

```

Need to install the git in a server.

## Console Output

Output

plain text

Information

Id '#2'

Data

Steps

Steps

Build

```
Started by user jaleelamohaideen
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/pipeline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (SCM Checkout)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/damodaranj/my-app.git
> git init /var/lib/jenkins/workspace/pipeline # timeout=10
Fetching upstream changes from https://github.com/damodaranj/my-app.git
> git --version # timeout=10
> git --version # 'git version 2.40.1'
> git fetch --tags --force --progress -- https://github.com/damodaranj/my-app.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/damodaranj/my-app.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/master^(commit) # timeout=10
Checking out Revision 59bdda88fa3c53787e0ef508f7f2fc77cf02aa41 (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f 59bdda88fa3c53787e0ef508f7f2fc77cf02aa41 # timeout=10
> git branch -a -v --no-abbrev # timeout=10
```

Now I completed the git checkout by adding the scripted pipeline.

To view the pipelines in a Jenkins using  
`cd /var/lib/Jenkins/workspace`

```
root@ip-172-31-3-75 ec2-user]# cd /var/lib/jenkins/workspace
root@ip-172-31-3-75 workspace]# ls
pipeline
root@ip-172-31-3-75 workspace]# cd pipeline
bash: cd: pipeline: No such file or directory
root@ip-172-31-3-75 workspace]# cd pipeline
root@ip-172-31-3-75 pipeline]# ls
deploy-to-tomcat  deploy-war-to-tomcat  Dockerfile  function-demo  github-push-trigger  global-variables  Jenkinsfile  parallel-executions  parameterized-builds  pom.xml  src
```

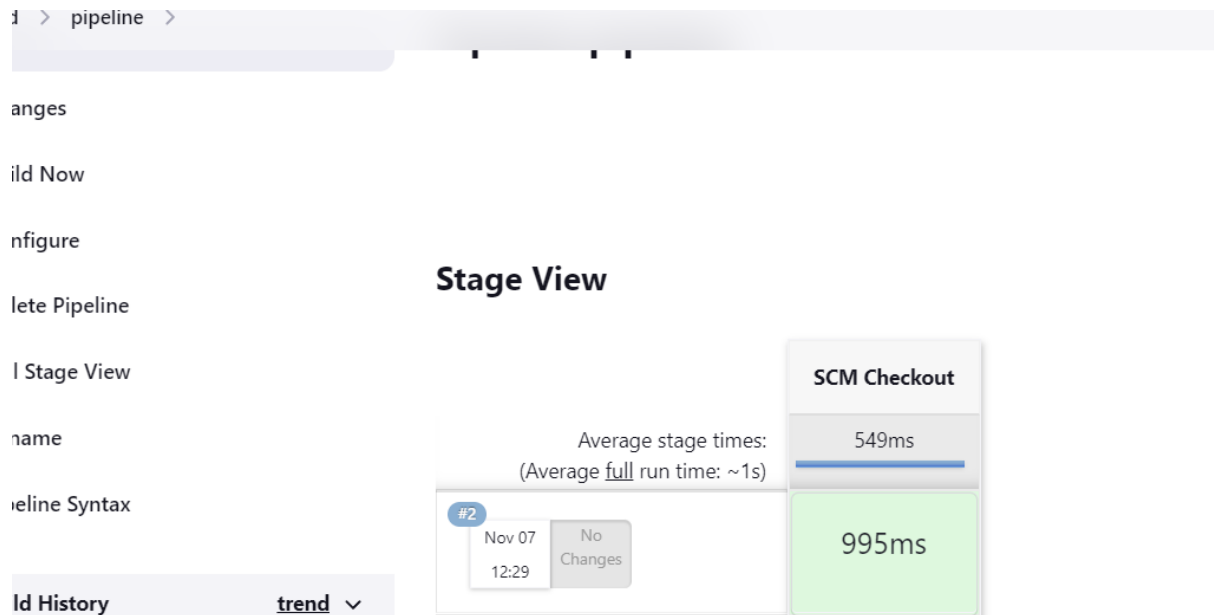
```
[root@ip-172-31-3-75 pipeline]# ll
total 40
-rw-r--r-- 1 jenkins jenkins 824 Nov 7 06:59 deploy-to-tomcat
-rw-r--r-- 1 jenkins jenkins 938 Nov 7 06:59 deploy-war-to-tomcat
-rw-r--r-- 1 jenkins jenkins 109 Nov 7 06:59 Dockerfile
-rw-r--r-- 1 jenkins jenkins 1108 Nov 7 06:59 function-demo
-rw-r--r-- 1 jenkins jenkins 234 Nov 7 06:59 github-push-trigger
-rw-r--r-- 1 jenkins jenkins 339 Nov 7 06:59 global-variables
-rw-r--r-- 1 jenkins jenkins 1245 Nov 7 06:59 Jenkinsfile
-rw-r--r-- 1 jenkins jenkins 311 Nov 7 06:59 parallel-executions
-rw-r--r-- 1 jenkins jenkins 328 Nov 7 06:59 parameterized-builds
-rwxr-xr-x 1 jenkins jenkins 1822 Nov 7 06:59 pom.xml
drwxr-xr-x 4 jenkins jenkins 47 Nov 7 06:59 src
```

2.ls

3.cd pipeline

## 4. Is

**All the repository files are present in the Jenkins now.**



### **STEP2:**

**Build the file(here java packages is there so we can build using maven).**

### **MAVEN**

**If we integrate with Jenkins we need to install plugins and to set the path.**

**Build the application using maven.**

**To convert a code into a package.**

**(executable format is .war or .jar file)**

**Other than maven-ant or gradle is used for building the application.**

**Maven is used to build the application only for java based application.**

**Maven file is a pom.xml file**

---

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>in.javahome</groupId>
  <artifactId>myweb</artifactId>
  <packaging>war</packaging>
  <version>0.0.5</version>
  <name>my-app</name>
```

---

**For a my-app application I deploy the myweb.war file**

**Install the maven using**

**wget https://mirrors.estointernet.in/apache/maven/maven-3/3.8.5/binaries/apache-maven-3.8.5-bin.tar.gz**

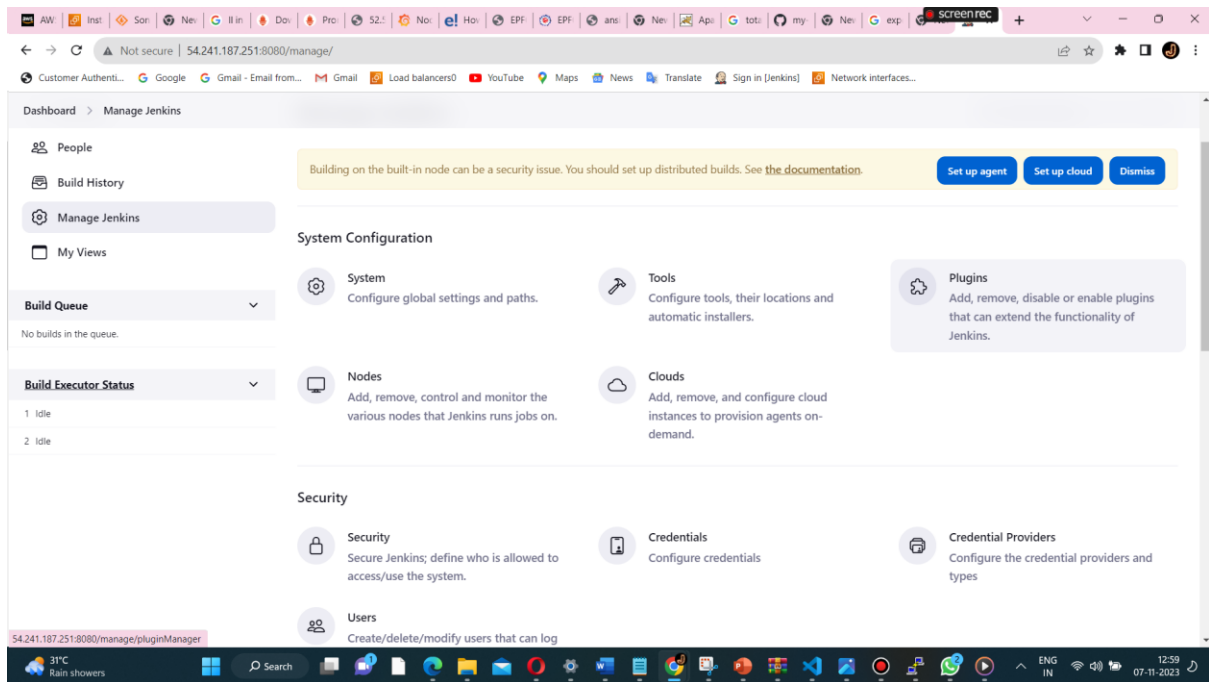
**using wget command I can install the maven.**

**Unzip the tar file**

**ls**

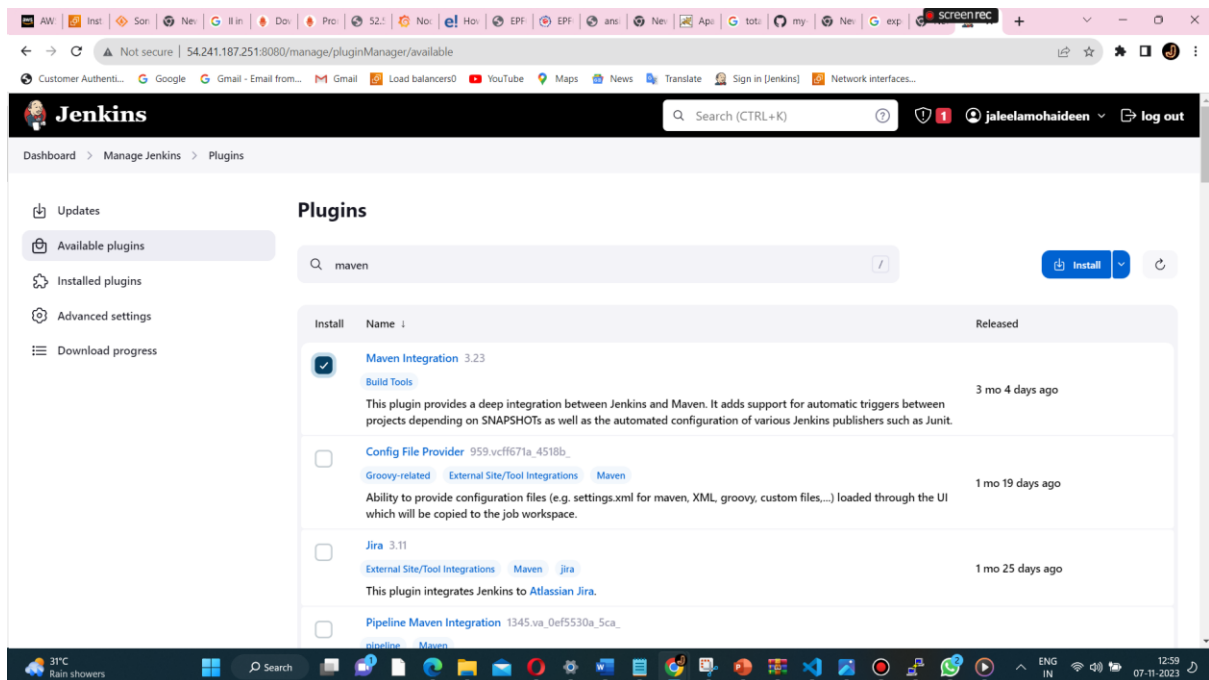
**tar -xvzf apache-maven-3.8.5-bin.tar.gz (to extract the file).**

**Then install the plugins in jenkins.**



Click on manage plugins choose plugins

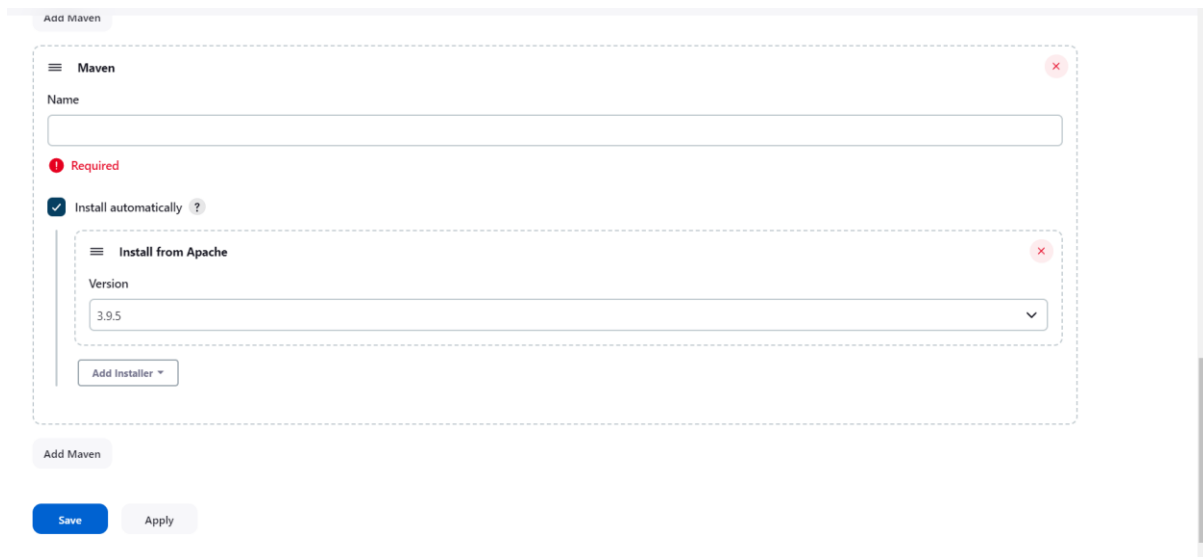
Click on available plugins



Select maven integration for installing.

Click on install then maven plugins is installed successfully.

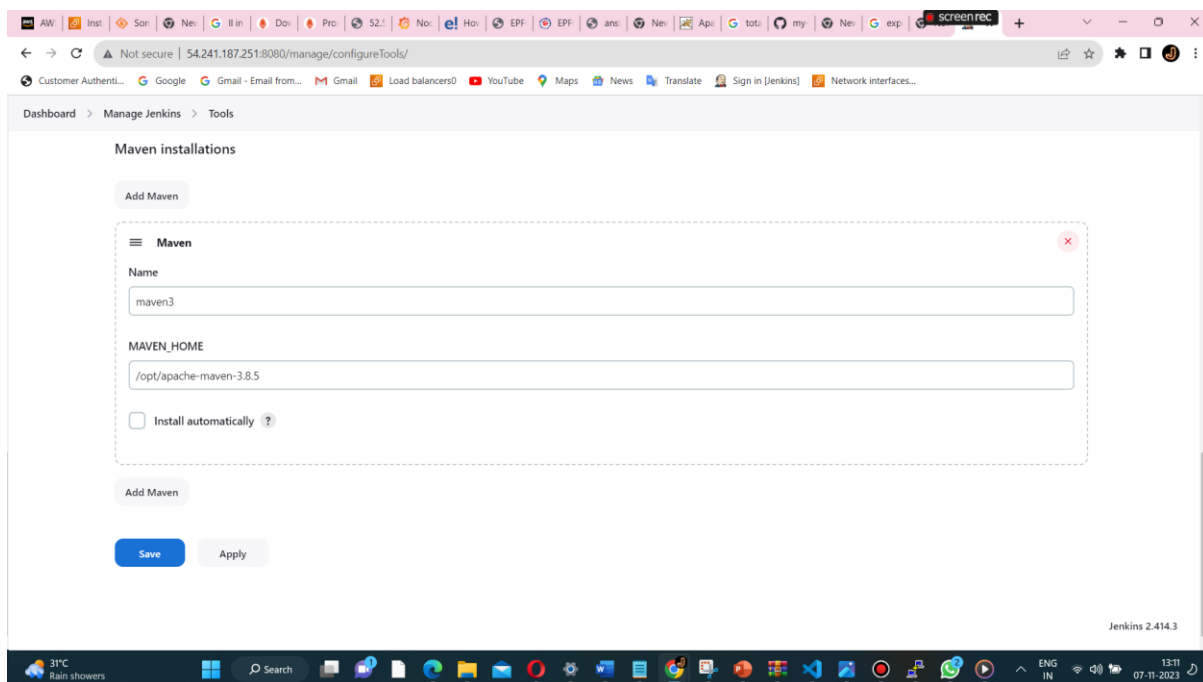
## Add the maven integration to a Jenkins.



The screenshot shows the 'Add Maven' configuration form in Jenkins. The form is titled 'Maven' and has a 'Name' field. Below the 'Name' field, there is a red 'Required' label. The 'Install automatically' checkbox is checked. Below this, there is a section titled 'Install from Apache' with a 'Version' dropdown menu set to '3.9.5'. At the bottom of the form, there is an 'Add Installer' button. The form is enclosed in a dashed border with a close button in the top right corner. Below the form, there are 'Save' and 'Apply' buttons.

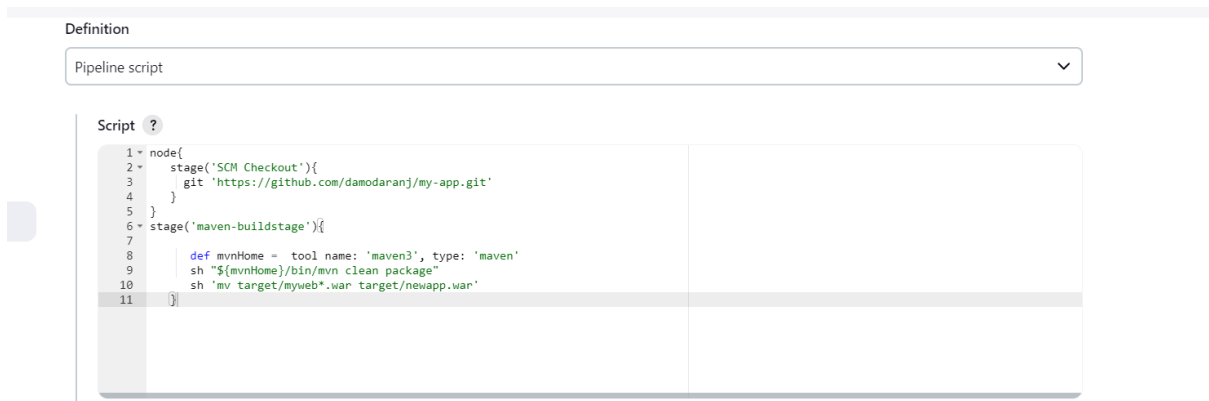
Click on manage Jenkins, choose tools for configure to a Jenkins.

Here we choose maven.



The screenshot shows the 'Maven installations' configuration page in Jenkins. The page is titled 'Maven installations' and has an 'Add Maven' button. Below the button, there is a form for adding a new Maven installation. The form has a 'Name' field with the value 'maven3', a 'MAVEN\_HOME' field with the value '/opt/apache-maven-3.8.5', and an 'Install automatically' checkbox which is unchecked. The form is enclosed in a dashed border with a close button in the top right corner. Below the form, there are 'Save' and 'Apply' buttons. The page also shows the breadcrumb 'Dashboard > Manage Jenkins > Tools' and the Jenkins version 'Jenkins 2.414.3'.

In tools we are adding completed the maven.



To add the maven staging in the pipeline.

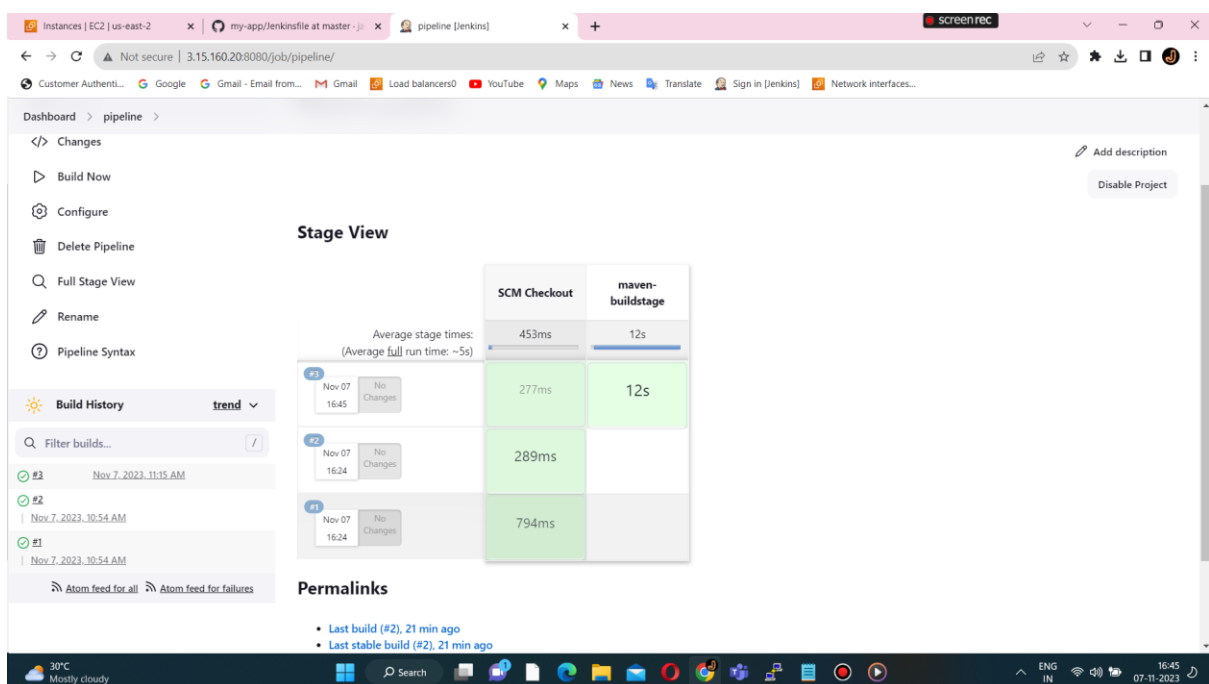
This script indicates- need to build the code using maven.

Tool name: maven3

Using sh mvnhome indicates /opt/apache-maven-3.8.5 /bin/mvn clean package

Next line is about create a target folder then create a myweb.war inside the folder.

newapp.war - to provide the unique name to deploy the war file.





Definition

Pipeline script

Script ?

```

1 node{
2   stage('SCM Checkout'){
3     git 'https://github.com/jaleelabegum/my-app'
4   }
5   stage('maven-buildstage'){
6
7     def mvnHome = tool name: 'maven3', type: 'maven'
8     sh "${mvnHome}/bin/mvn clean package"
9     sh 'mv target/myweb*.war target/newapp.war'
10  }
11 }

```

☒ Use Groovy Sandbox ?

[Pipeline Syntax](#)

**Save** **Apply**

To add the maven staging to build the files.

```

apache-maven-3.8.5-bin.tar.gz  deploy-war-to-tomcat  function-demo  global-variables  parallel-executions  pom.xml  target
deploy-to-tomcat             Dockerfile        github-push-trigger  Jenkinsfile      parameterized-builds  src
[root@ip-172-31-28-90 pipeline]# ll
total 8512
-rw-r--r-- 1 root root 8673123 Mar  5 2022 apache-maven-3.8.5-bin.tar.gz
-rw-r--r-- 1 jenkins jenkins 824 Nov  7 10:54 deploy-to-tomcat
-rw-r--r-- 1 jenkins jenkins 938 Nov  7 10:54 deploy-war-to-tomcat
-rw-r--r-- 1 jenkins jenkins 109 Nov  7 10:54 Dockerfile
-rw-r--r-- 1 jenkins jenkins 1109 Nov  7 10:54 function-demo
-rw-r--r-- 1 jenkins jenkins 234 Nov  7 10:54 github-push-trigger
-rw-r--r-- 1 jenkins jenkins 339 Nov  7 10:54 global-variables
-rw-r--r-- 1 jenkins jenkins 1243 Nov  7 10:54 Jenkinsfile
-rw-r--r-- 1 jenkins jenkins 311 Nov  7 10:54 parallel-executions
-rw-r--r-- 1 jenkins jenkins 328 Nov  7 10:54 parameterized-builds
-rwxr-xr-x 1 jenkins jenkins 1822 Nov  7 10:54 pom.xml
drwxr-xr-x 4 jenkins jenkins 47 Nov  7 10:54 src
drwxr-xr-x 10 jenkins jenkins 199 Nov  7 11:15 target

```

After inside the pipeline path target folder is created

Inside the folder.

```

-rw-r--r-- 1 jenkins jenkins 1109 Nov  7 10:54 function-demo
-rw-r--r-- 1 jenkins jenkins 234 Nov  7 10:54 github-push-trigger
-rw-r--r-- 1 jenkins jenkins 339 Nov  7 10:54 global-variables
-rw-r--r-- 1 jenkins jenkins 1243 Nov  7 10:54 Jenkinsfile
-rw-r--r-- 1 jenkins jenkins 311 Nov  7 10:54 parallel-executions
-rw-r--r-- 1 jenkins jenkins 328 Nov  7 10:54 parameterized-builds
-rwxr-xr-x 1 jenkins jenkins 1822 Nov  7 10:54 pom.xml
drwxr-xr-x 4 jenkins jenkins 47 Nov  7 10:54 src
drwxr-xr-x 10 jenkins jenkins 199 Nov  7 11:15 target
[root@ip-172-31-28-90 pipeline]# cd target
[root@ip-172-31-28-90 target]# ls
classes generated-sources generated-test-sources maven-archiver maven-status myweb-0.0.5 newapp.war surefire-reports test-classes

```

newapp.war file is created. Now the code is converted into a war package. Unique name to a file.(newapp.war)

			SCM Checkout	maven-buildstage
Average stage times: (Average <u>full</u> run time: ~5s)			453ms	12s
#3	Nov 07 16:45	No Changes	277ms	12s
#2	Nov 07 16:24	No Changes	289ms	
#1	Nov 07 16:24	No Changes	794ms	

## Permalinks

# TO CONVERT A WAR FILE TO A DOCKER IMAGE AND PUSH INTO A DOCKER HUB

## COMMANDS

### Install the docker on same server

**yum install docker -y**

Pipeline script

Script ?

```

1 node{
2   stage('SCM Checkout'){
3     git 'https://github.com/jaleelabegum/my-app'
4   }
5   stage('maven-buildstage'){
6     def mvnHome = tool name: 'maven3', type: 'maven'
7     sh "${mvnHome}/bin/mvn clean package"
8     sh mv target/myweb*.war target/newapp.war
9   }
10  }
11  stage('Build Docker Image'){
12    sh 'docker build -t jaleelal310/myweb:0.0.2 .'
13  }
14 }

```

**After installing the docker, we need to build the docker file to add the stage here.**

**Already Dockerfile is present in the pipeline directory so we just build the image in a script.**

The screenshot shows the Jenkins 'Stage Logs (Build Docker Image)' window. The log content is as follows:

```
Shell Script -- docker build -t jaleela1310/myweb:0.0.2 . (self time 275ms)

+ docker build -t jaleela1310/myweb:0.0.2 .
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/build?buildargs=%7B%7D&cachefrom=%5B%5D&groupparent=&cpuperiod=0&cpuquota=0&cpusetcpus=&cpusetmems=&pushshares=0&dockerfile=Dockerfile&labels=%7B%7D&memory=0&memswap=0&networkmode=default&rm=1&shmsize=0&t=jaleela1310%2Fmyweb%3A0.0.2&target=&ulimits=null&version=1": dial unix /var/run/docker.sock: connect: permission denied
```

The 'Stage View' table below the logs shows the following data:

Build	SCM Checkout	maven-buildstage	Shell Script
#4 Nov 07 17:14	275ms	5s	305ms failed
#3 Nov 07 16:45	277ms	12s	
#2 Nov 07 16:24	289ms		

An error message box is displayed over the failed build row: 'Failed with the following error(s): Shell Script script returned exit code 1. See stage logs for more detail.' A 'Logs' button is also present.

**to provide the permission to a Jenkins to build a docker file otherwise it throws an error.**

**To run the docker file provide the permission to a Jenkins.**

**chmod 0777 /var/run/docker.sock**  
**command provide in a server.**

The screenshot shows the Jenkins 'Stage View' after the permission change. The 'Average full run time' is now ~8s. The 'buildstage' and 'Image' columns are now green, indicating success.

Build	buildstage	Image
#5 Nov 07 17:21	330ms	4s
#4 Nov 07 17:14	275ms	5s

**Now it image is built successfully.**

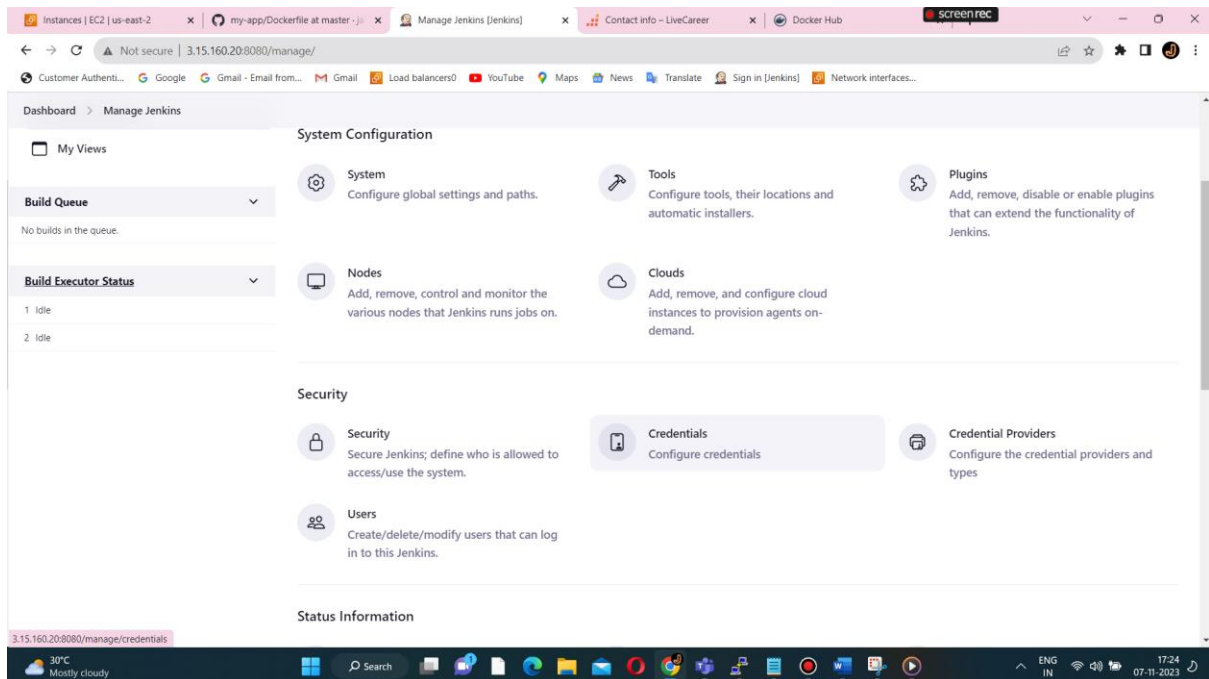
**To push the image to a docker.**

```
[root@ip-172-31-28-90 target]# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
jaleela1310/myweb	0.0.2	b7a2382ed514	About a minute ago	462MB
tomcat	8	3ae2bbab2686	7 days ago	461MB

```
[root@ip-172-31-28-90 target]#
```

**To pass the sensitive information to a secrets.**



**Click on manage Jenkins->credentials->system->globalcredentials**

Instances | EC2 | us-east-2 x my-app/Dockerfile at master · jaleelamohaideen/my-app · GitHub Jenkins > Credentials [Jenkins] x Contact info - LiveCareer x Docker Hub screenrec

Not secure | 3.15.160.20:8080/manage/credentials/

Customer Authent... Google Gmail - Email from... Gmail Load balancers0 YouTube Maps News Translate Sign in [Jenkins] Network interfaces...

**Jenkins** Search (CTRL+K) jaleelamohaideen log out

Dashboard > Manage Jenkins > Credentials

## Credentials

T	P	Store	Domain	ID	Name
---	---	-------	--------	----	------

### Stores scoped to Jenkins

P	Store	Domains
	System	(global)

Icon: S M L

3.15.160.20:8080/manage/credentials/store/system REST API Jenkins 2.414.3

30°C Mostly cloudy Search 17:27 07-11-2023

Instances | EC2 | us-east-2 x my-app/Dockerfile at master · jaleelamohaideen/my-app · GitHub System [Jenkins] x Contact info - LiveCareer x Docker Hub screenrec

Not secure | 3.15.160.20:8080/manage/credentials/store/system/

Customer Authent... Google Gmail - Email from... Gmail Load balancers0 YouTube Maps News Translate Sign in [Jenkins] Network interfaces...

**Jenkins** Search (CTRL+K) jaleelamohaideen log out

Dashboard > Manage Jenkins > Credentials > System

## System

+ Add domain

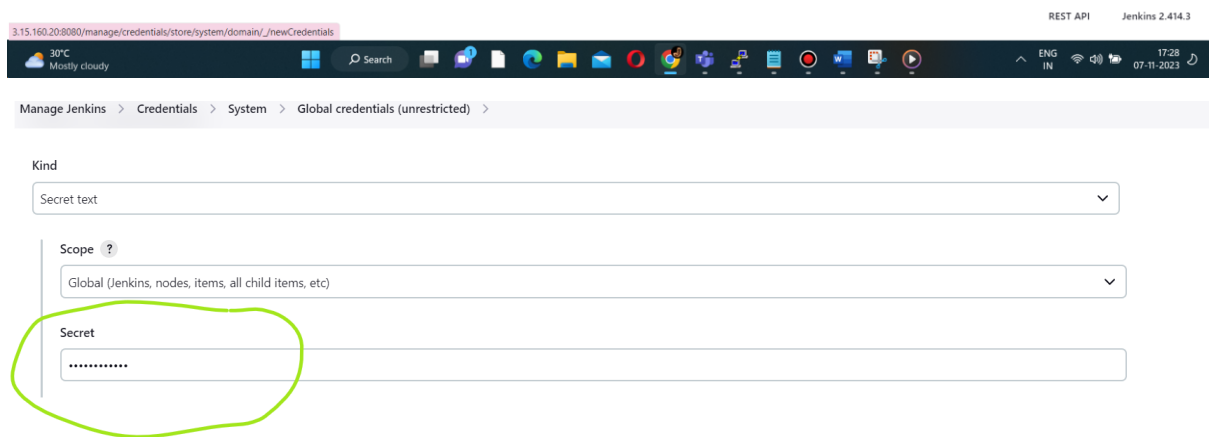
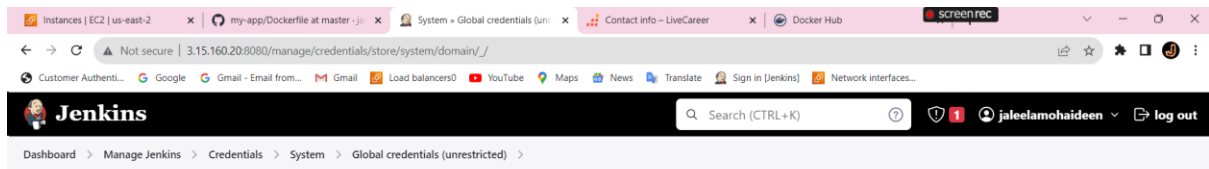
Domain	Description
Global credentials (unrestricted)	Credentials that should be available irrespective of domain specification to requirements matching.

Icon: S M L

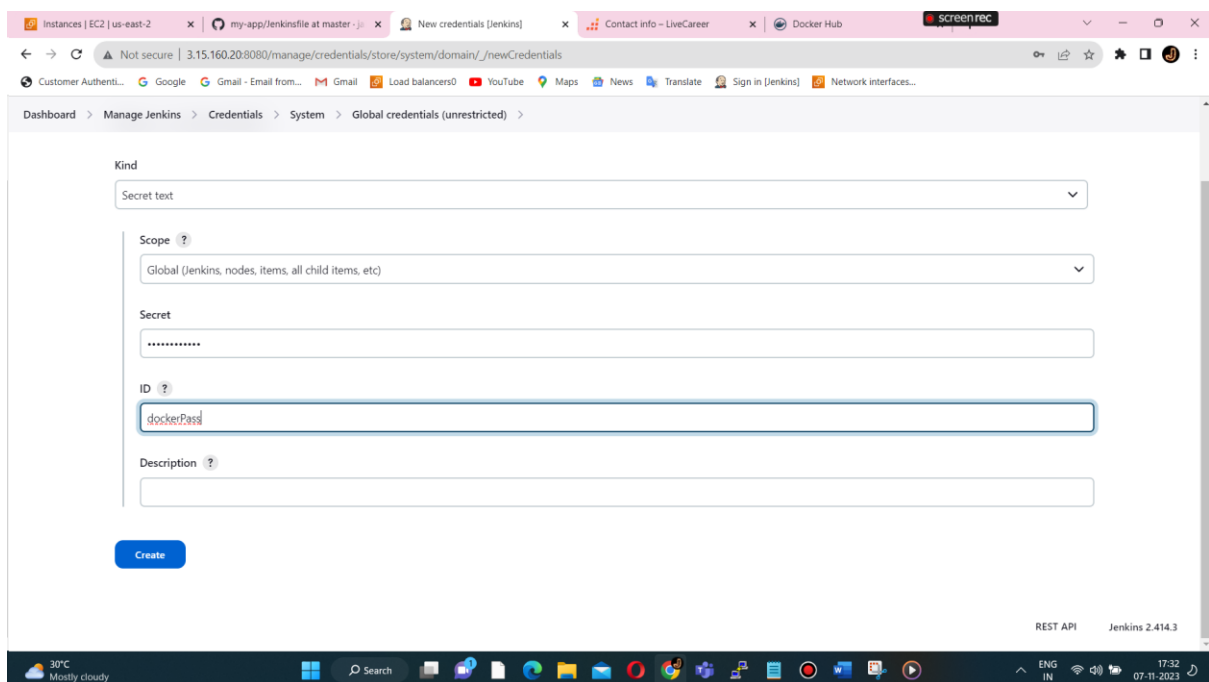
3.15.160.20:8080/manage/credentials/store/system/domain/ REST API Jenkins 2.414.3

30°C Mostly cloudy Search 17:27 07-11-2023

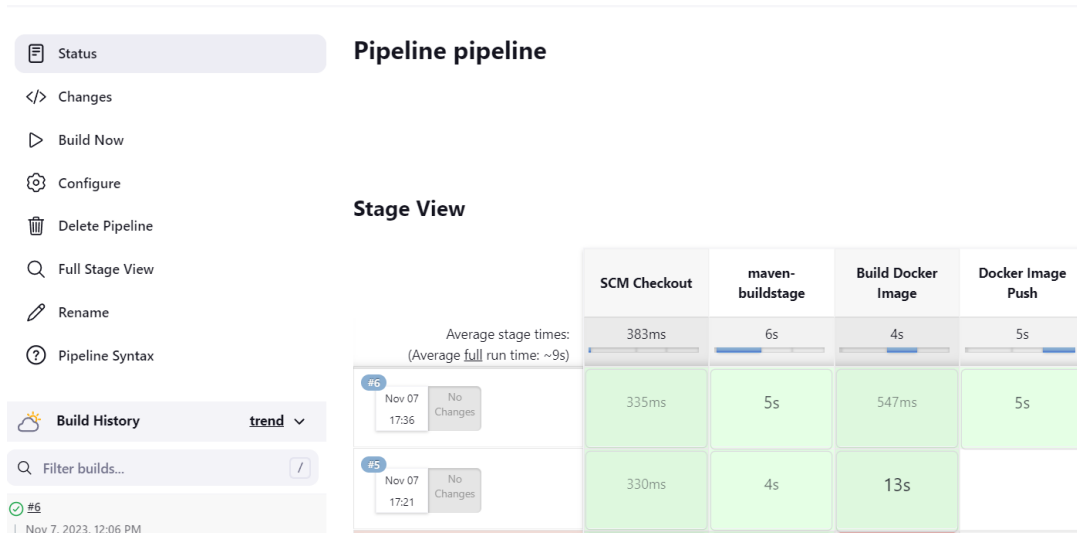
Then add credentials.



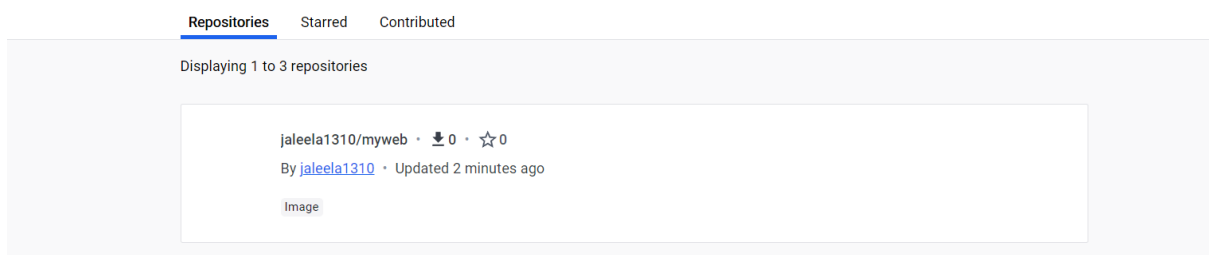
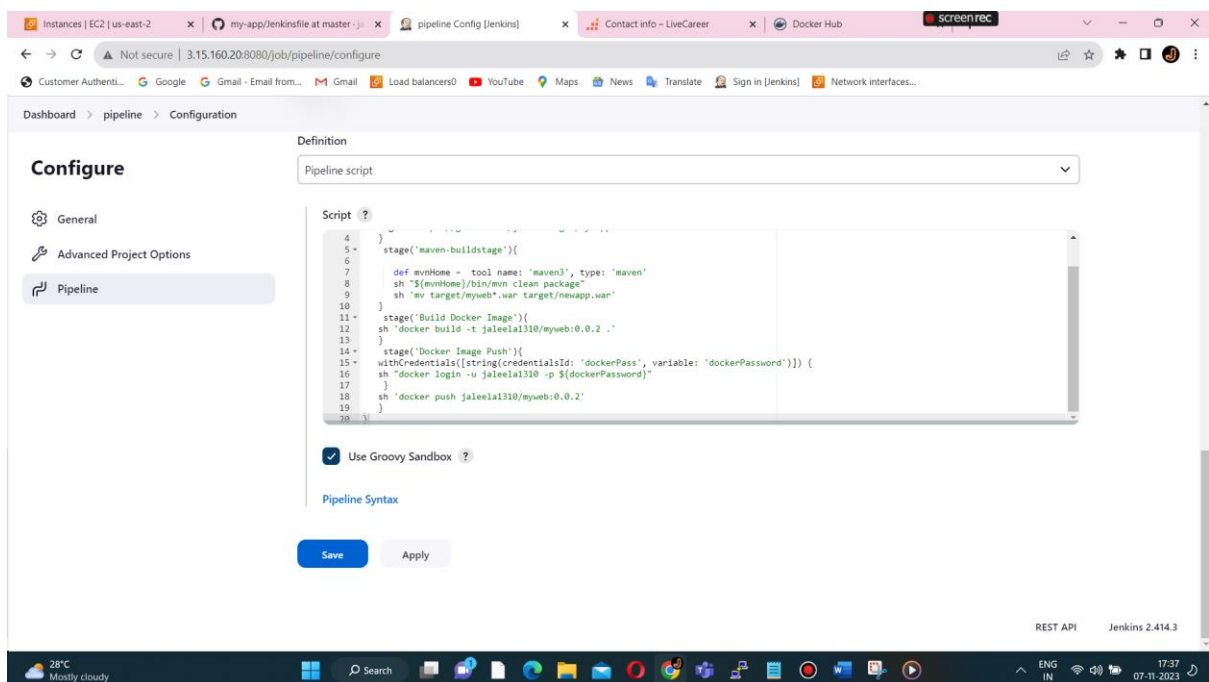
**In the secret give the dockerhub password.**



Id is dockerPass that should be mentioned in a script file.



Now the docker image is pushed into our docker hub.



Myweb got pushed into a docker hub.