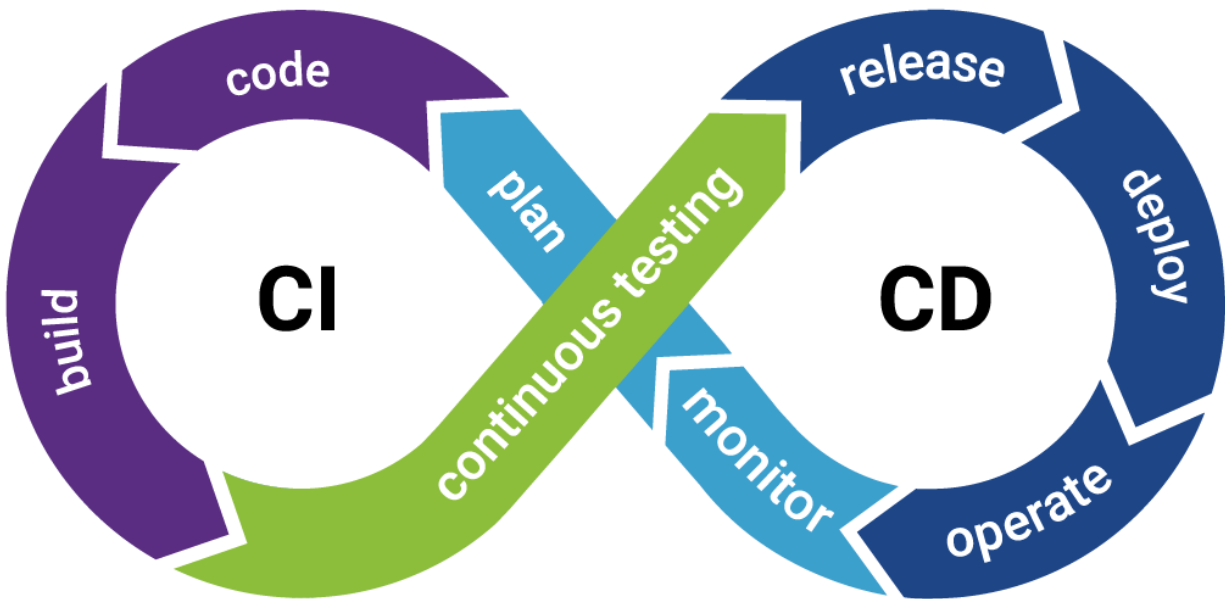


Implementing a Continuous Integration/Continuous Delivery (CI/CD) Pipeline



Requirements : CI/CD pipeline System

- Git - local version control system
- GitHub - As Distributed version control system.
- Jenkins - Continuous Integration tool.
- Maven - As a Build Tool.
- Docker -Containerization.
- Kubernetes - As a Container Management Tool.

The Project Covers

- Setup CI/CD with GitHub, Jenkins, Maven & Tomcat.
- Setup Jenkins
- Setup & Configure Maven , Git.
- Setup Tomcat Server.
- Integrating GitHub,Maven ,Tomcat Server with Jenkins
- Create a CI and CD Job.
- Test the Deployment.

Resources to Setup CI and CD pipeline.

- AWS account.
- GitHub account (for source code and documentation).
- Git – local version control system.

Let's Start

Setup Jenkins Server

1. Setup a Linux EC2 instance
2. Install Java
3. Install Jenkins
4. Start Jenkins
5. Access Web UI on port 8080

```
sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key
```

```
yum install epel-release //fails
sudo amazon-linux-extras install epel
sudo amazon-linux-extras install java-openjdk11
yum install jenkins
```

```
~\  #####      Amazon Linux 2
~~~\#####\
~~~\###|
~~~\#/
~~~V~'-'>
~~~~
~~~~-.-/
~~~~/_/
~~~~/_m/'

AL2 End of Life is 2025-06-30.

A newer version of Amazon Linux is available!

Amazon Linux 2023, GA and supported until 2028-03-15.
https://aws.amazon.com/linux/amazon-linux-2023/

25 package(s) needed for security, out of 47 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-4-218 ~]$ history
 1 sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
 2 sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
 3 sudo amazon-linux-extras install java-openjdk11
 4 sudo yum update
 5 sudo yum install jenkins
 6 sudo systemctl start jenkins
 7 sudo systemctl enable jenkins
 8 sudo systemctl status jenkins
 9 sudo jenkins version
10 sudo jenkins --version
11 sudo cd /var/lib/jenkins
```

i-07cf8e4909a3a13e0 (jenkins-master)

PublicIPs: 13.127.120.128 PrivateIPs: 172.31.4.218

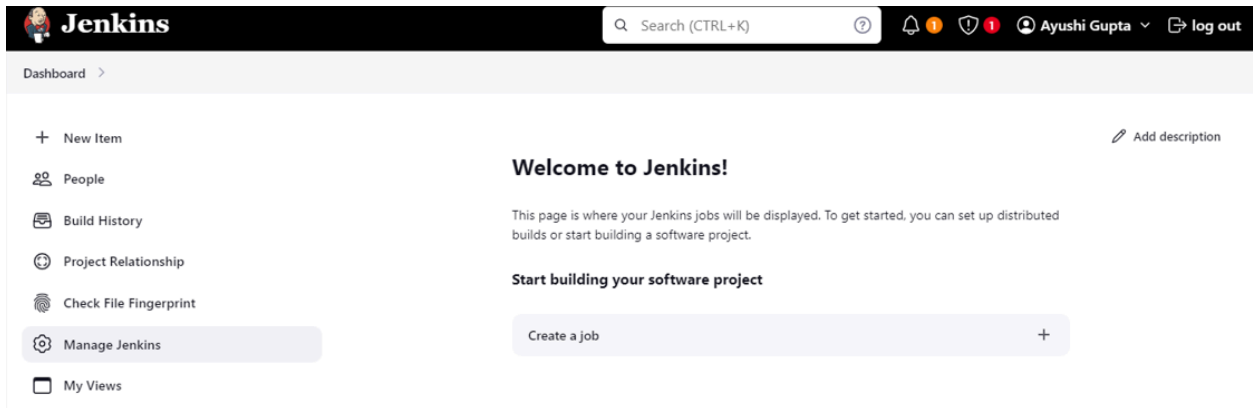


Fig. Access Web UI on port 8080

Integrate Git with Jenkins

- Install Git on Jenkins Instances
- Install GitHub plug in on Jenkins GUI
- Configure Git on Jenkins GUI Install Git on Jenkins Instances: `yum install git`

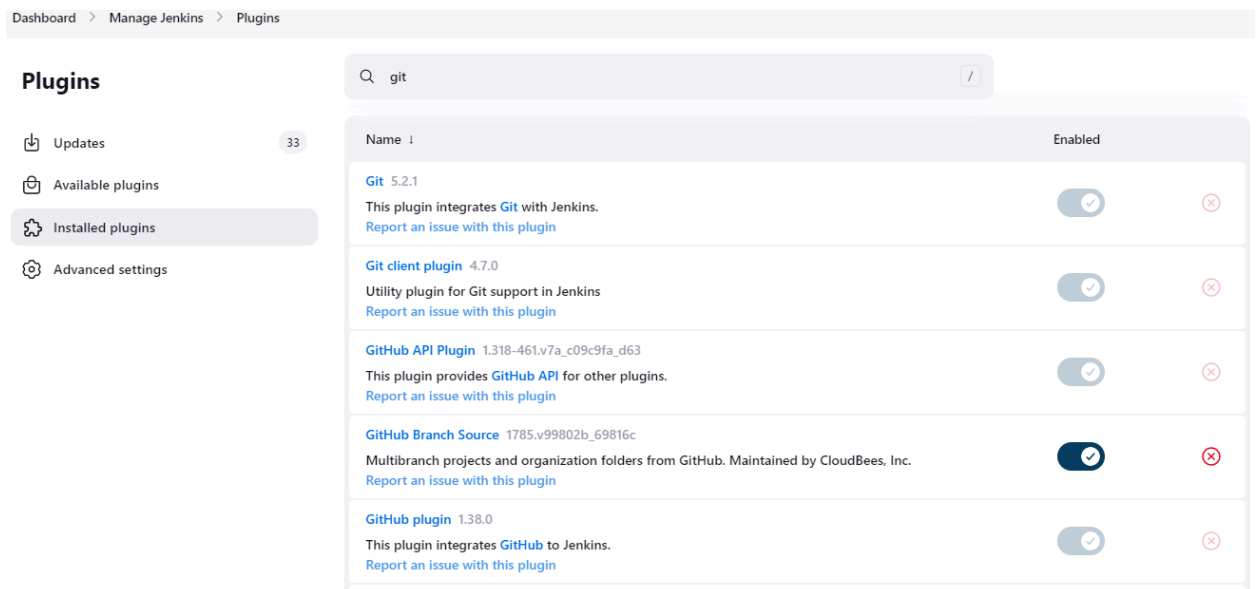


Fig: Integrate git with jenkins

Integrate Maven with Jenkins

- Setup Maven on Jenkins Server
- Install Maven Plugin
- Configure Maven and Java
- Launch an ec2 Instance and make it a jenkins slave
- Configure Maven on the ec2 slave

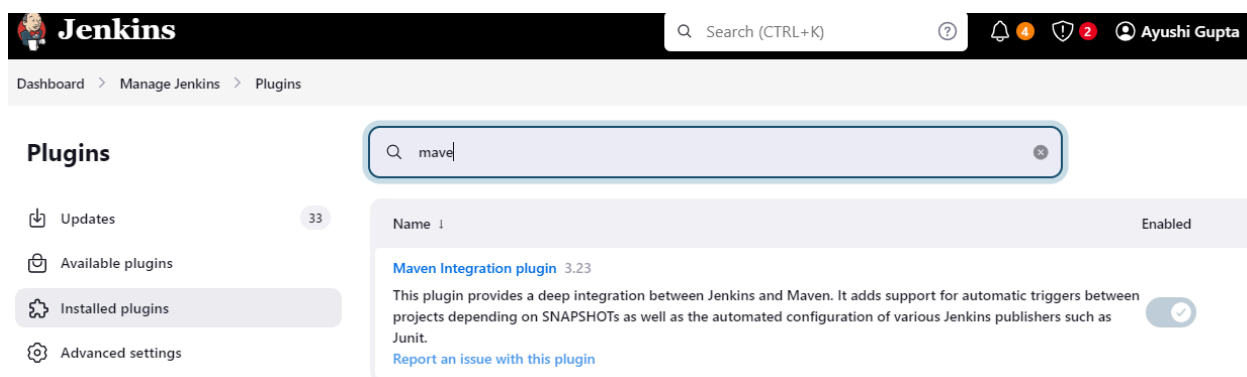
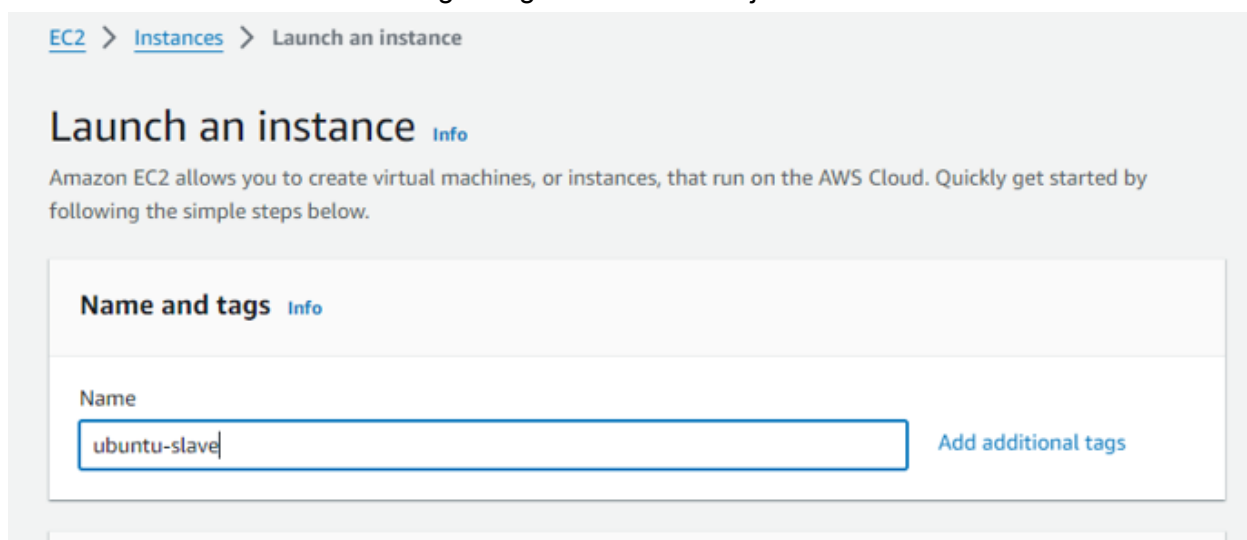
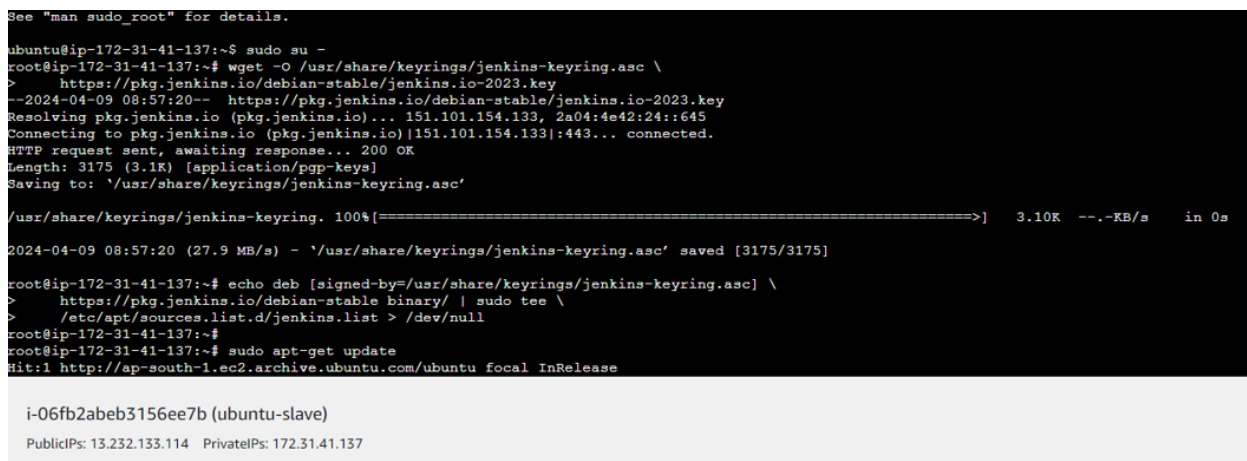


Fig: Integrate maven with jenkins



Launch a EC2 instance to build code with maven



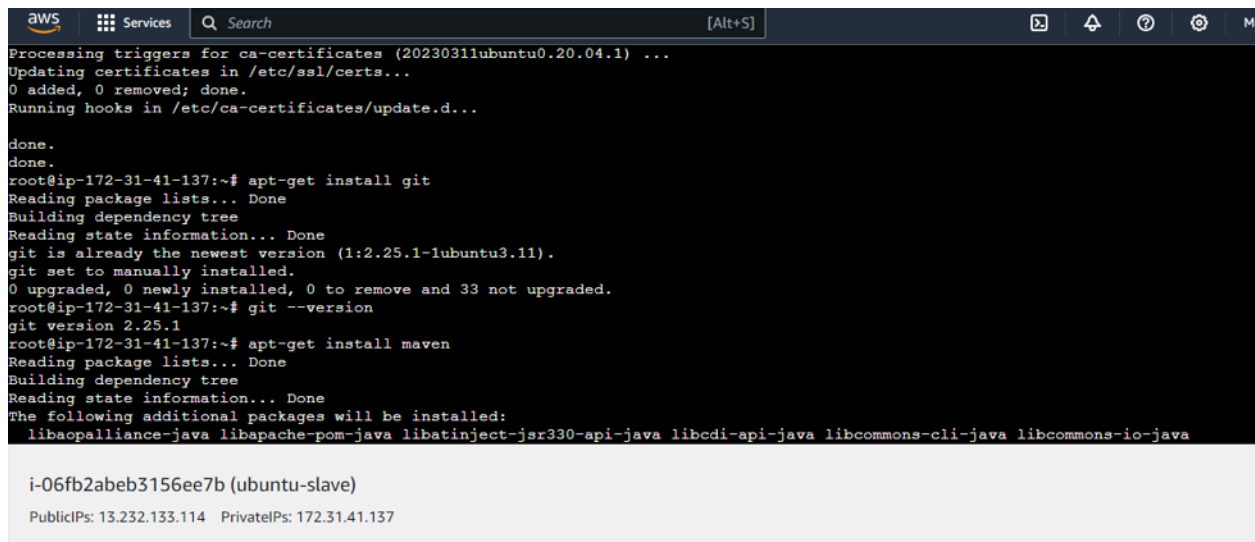
Making the Instance a Jenkins slave

```

sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" \
https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
/etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update

```

Commands to make Jenkins slave.

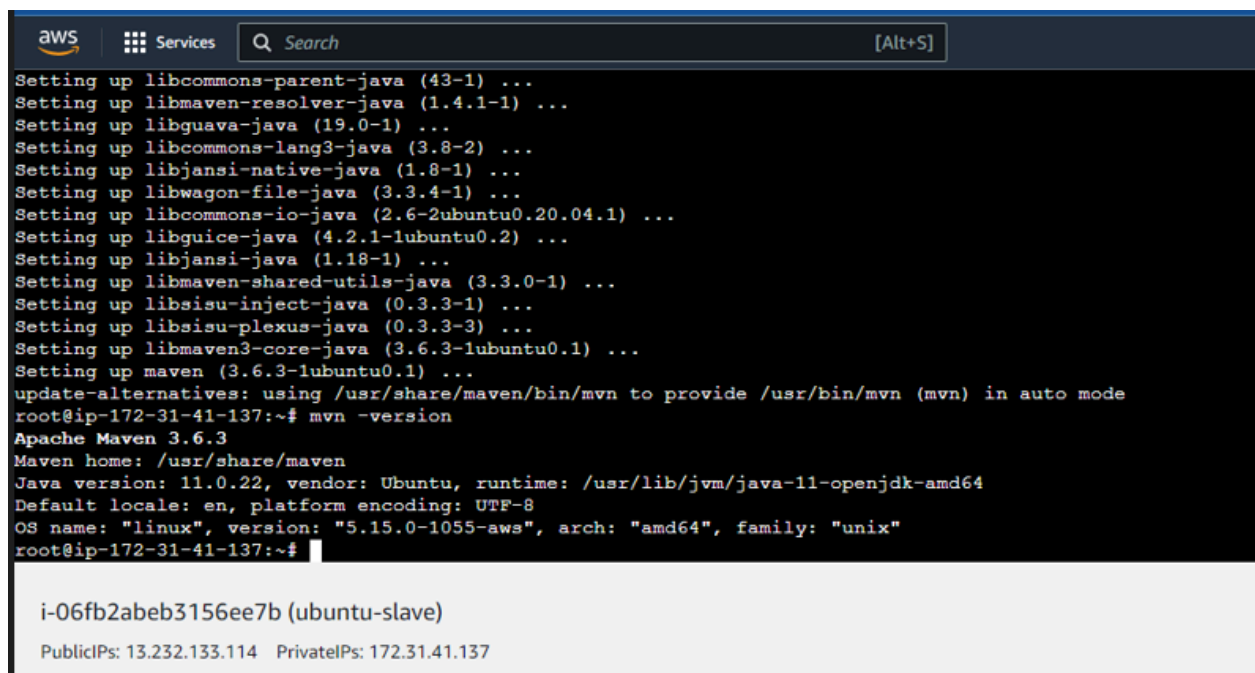


```

aws Services Search [Alt+S]
Processing triggers for ca-certificates (20230311ubuntu0.20.04.1) ...
Updating certificates in /etc/ssl/certs...
0 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
done.
root@ip-172-31-41-137:~# apt-get install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
git is already the newest version (1:2.25.1-1ubuntu3.11).
git set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 33 not upgraded.
root@ip-172-31-41-137:~# git --version
git version 2.25.1
root@ip-172-31-41-137:~# apt-get install maven
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libaopalliance-java libapache-pom-java libatinject-jsr330-api-java libcdi-api-java libcommons-cli-java libcommons-io-java
i-06fb2abeb3156ee7b (ubuntu-slave)
PublicIPs: 13.232.133.114 PrivateIPs: 172.31.41.137

```

Installing maven

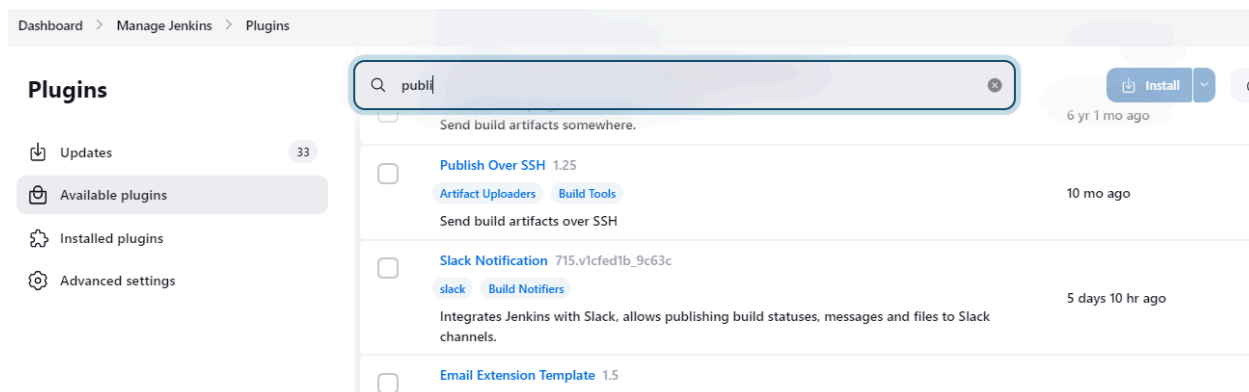


```

aws Services Search [Alt+S]
Setting up libcommons-parent-java (43-1) ...
Setting up libmaven-resolver-java (1.4.1-1) ...
Setting up libguava-java (19.0-1) ...
Setting up libcommons-lang3-java (3.8-2) ...
Setting up libjansi-native-java (1.8-1) ...
Setting up libwagon-file-java (3.3.4-1) ...
Setting up libcommons-io-java (2.6-2ubuntu0.20.04.1) ...
Setting up libguice-java (4.2.1-1ubuntu0.2) ...
Setting up libjansi-java (1.18-1) ...
Setting up libmaven-shared-utils-java (3.3.0-1) ...
Setting up libsisu-inject-java (0.3.3-1) ...
Setting up libsisu-plexus-java (0.3.3-3) ...
Setting up libmaven3-core-java (3.6.3-1ubuntu0.1) ...
Setting up maven (3.6.3-1ubuntu0.1) ...
update-alternatives: using /usr/share/maven/bin/mvn to provide /usr/bin/mvn (mvn) in auto mode
root@ip-172-31-41-137:~# mvn -version
Apache Maven 3.6.3
Maven home: /usr/share/maven
Java version: 11.0.22, vendor: Ubuntu, runtime: /usr/lib/jvm/java-11-openjdk-amd64
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "5.15.0-1055-aws", arch: "amd64", family: "unix"
root@ip-172-31-41-137:~#
i-06fb2abeb3156ee7b (ubuntu-slave)
PublicIPs: 13.232.133.114 PrivateIPs: 172.31.41.137

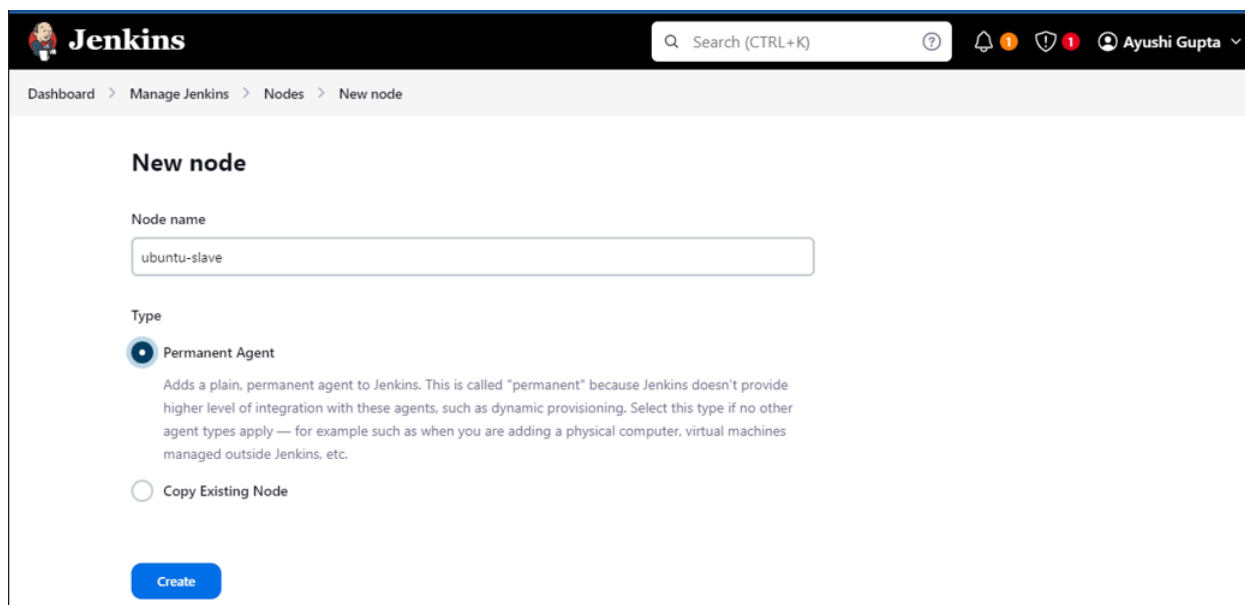
```

Maven Installed



The screenshot shows the Jenkins 'Plugins' page. The breadcrumb navigation is 'Dashboard > Manage Jenkins > Plugins'. On the left, there's a sidebar with 'Plugins' as the main heading and four sub-items: 'Updates' (33), 'Available plugins' (selected), 'Installed plugins', and 'Advanced settings'. A search bar at the top of the main content area contains the text 'publi'. Below the search bar, a list of plugins is displayed. The first plugin is 'Publish Over SSH' version 1.25, with tags 'Artifact Uploaders' and 'Build Tools', and a description 'Send build artifacts over SSH'. It was last updated '10 mo ago'. The second plugin is 'Slack Notification' version 715.v1cfed1b_9c63c, with tags 'slack' and 'Build Notifiers', and a description 'Integrates Jenkins with Slack, allows publishing build statuses, messages and files to Slack channels.' It was last updated '5 days 10 hr ago'. The third plugin is 'Email Extension Template' version 1.5. An 'Install' button is visible in the top right corner.

Install "Publish Over SSH" plugin



The screenshot shows the Jenkins 'New node' page. The breadcrumb navigation is 'Dashboard > Manage Jenkins > Nodes > New node'. The page has a dark header with the Jenkins logo, a search bar, and a user profile 'Ayushi Gupta'. The main content area is titled 'New node'. It has a 'Node name' field with the value 'ubuntu-slave'. Below this, there's a 'Type' section with two radio buttons: 'Permanent Agent' (selected) and 'Copy Existing Node'. The 'Permanent Agent' option has a description: 'Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.' At the bottom, there is a blue 'Create' button.

Make a ubuntu-slave a permanent node on jenkins web UI

Dashboard > Manage Jenkins > Nodes >

Remote root directory ?

Labels ?

Usage ?

Launch method ?

Host ?

[Save](#)

Configuration of a agent

Dashboard > Manage Jenkins > Nodes >

Usage ?

Launch method ?

Host ?

Credentials ?

[+ Add](#)

[Save](#)

Jenkins Credentials Provider: Jenkins

Add Credentials

Domain
Global credentials (unrestricted)

Kind
SSH Username with private key

Scope ?
Global (Jenkins, nodes, items, all child items, etc)

ID ?
ubuntu-slave-ID

Description ?

Give Credentials in the node to take SSH Session of ubuntu slave

new_key - Notepad

File Edit Format View Help

```
W6VQGm9AGJQhV6BihYI15a+ZUAz/xr14nkzE9YtQQ1Q40ctIdAkiWIf+tgq1kIhZ
tTxwBaN3+1r3wHo8QW7yOdR09Gnj/O1g3LdfNekUB12N4R6NB8T8fjQdLEPWrJ2bM
Xz6BnnYwTACu6v6Gcv1rYB1BfZ1Lx7c5jqEU5QIDAQABoIbAEAh1tcwhp+RwvMt
QtX76qKYMFBqrusICzY+WGx3uSLENE5dbrRgyNUPJQ1MP2Af80nJucrU441AKM/q
HA0HBeTw1FhxMCTRjvGHTNDb+++T2DsgXfCeeT0mwQQNAvSimeTehOcv195be+Cc
668Nw5mjY2ctG+aeMuQ8T2UYPOh8G4vpCiLqnnQRjrZkmiWzTHHBZ555HDdZVBFi
0f7TVEeKr7iu6KPPKa/sYEyKrbjJA4xgBQ1KKnbFuhFBQtm7paKr3MkFGVQvLMUv
HBB+pnFkDZ+Q4FK7Hxqte0m8ZcBzd2w7394okEVz7+V63D8eUj+jJBjMURAg5nU5
sYnTRzkCgYEA4YeTyO+RR0S2Z1aDdkoaIHH0r4rYITdiexQsHGEtIXjZyMpWPsi
CLjtt76ZFkVDJqLXdTeK0Jfr8Fch0Rsr+qUdMk6D647quqZVs1SdBGXgjIdXLaVa
AIIn4JTRW+9KDtTcYy6dL1Wp45hSczMfr7jffXk4FiRgTe6fLdw18hXsCgYEAyywK
TmzgdmMxmr7eDiuTiJfa4FIRp1GUTnaz6iY50zkXiHMxZq/aLm290t8pHbvt7nDC
V4k9MYIqm1uJ35B0ux0t9uYrja2IDA01PCs7Yro+NKjbA1QFEYJxyJ0h2+961HyE
0WdVhusNkA+LzzCK1aMznRgEwlsqBZu+a8ujUR8CgYBAHFBQgHm1Qfbv5Iv+Z1ZC
/uH1vNxpNTj8/LUIZD92EmqmF45AzaLi/N/2jiechYct6vT8HPkXps6YqJZH3ZD
DAjesh8iKUBkyr/1r8PrNkSpClnn6edJ2+6kdNp9P3ydzXnmFqqiAmBX12gzdmmF
EGmyU5+TNzrinfJ153yMXQKBgDJWJfOKPA56VhuDfGaMw1ATN3IuQxtuNc0+0ZJo
iBkR4we2an3jga3RGj+bbP+160wTWcLwSI30jvPQuRb/pjwXQxmRqya94KADwbW3
jE4IkJLhZu1rwZm19NTZ5tcjvQ3aD9oFvQXhRIn1oBDkGOG7+A0YyM0SYxXioUn5
u0vHAoGafPHm9TG3gHJGpay6JK2MFNy09h4nHzJvRoCM5b7AJ9AZ812G8o3kciOG
Vt8tLh7sApj1M9JIn68I+t/cuWrQnLSQWjPkkW1jQng+DBY6c1XfmuXGaH1py39N
NM9PTF1eyJj9hApPtJsV8Q5TmKo9U5h+3B+gxWeJhRjEu0W9sUU=
-----END RSA PRIVATE KEY-----
```


Jenkins Credentials Provider: Jenkins

Private Key

☒ Enter directly

Key

Enter New Secret Below

QOVNACQRTPhm9T05ghJ0pay05K2mFny09n4mH2JvKOCN507A39AZ8120805KC100
Vt8tLh7sApj1M9JIIn68I+t/cuWnQLSQWJpKkw1jQng+DBY6c1XfmuXGaH1py39N
NM9PTFleyJj9hApPtJsV8Q5TmKo9U5h+3B+gxWeJhRjEu0W9sUU=
-----END RSA PRIVATE KEY-----

Passphrase

Cancel

Add

SSH Private key based authentication

Dashboard > Manage Jenkins > Nodes >

Credentials ?

ubuntu (slave on ubuntu)

+ Add ▾

Host Key Verification Strategy ?

Non verifying Verification Strategy

Advanced ▾

Availability ?

Keep this agent online as much as possible

Node Properties

☐ Disable deferred wipeout on this node ?

☐ Disk Space Monitoring Thresholds

Save

By : Ayushi Gupta

Email : ayushigupta1496@gmail.com

Dashboard > Manage Jenkins > Nodes >

Nodes

[+ New Node](#) [Configure Monitors](#) [Refresh](#)

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	5.33 GiB	1 0 B	5.33 GiB	0ms Settings
	jenkins_agent1		N/A	N/A	N/A	N/A	N/A Settings
	ubuntu-slave	Linux (amd64)	In sync	5.14 GiB	1 0 B	5.14 GiB	53ms Settings
last checked		1 min 16 sec	1 min 16 sec	1 min 16 sec	1 min 16 sec	1 min 16 sec	1 min 16 sec

Icon: ☒ S ☐ M ☐ L

Build Queue

No builds in the queue.

Build Executor Status

Built-In Node

- 1 Idle

jenkins_agent1 (launching...)

- 2 Idle

ubuntu-slave

- 1 Idle

Now i created Git repo to push my code on github

ayushigupta1496 / new_java_app

new_java_app Public [Pin](#) [Unwatch](#) 1

main 1 Branch 0 Tags

Go to file Add file Code

ayushigupta1496 Initial commit c3387b6 · now 1 Commits

README.md Initial commit now

README

new_java_app

Now i created post commit hooks to push the code automatically when code committed

```
MINGW64/d/git_new/new_java_app/.git/hooks
Cloning into 'new_java_app'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.

DELL@DESKTOP-P6NHTN3 MINGW64 /d/git_new
$ ls
new_java_app/

DELL@DESKTOP-P6NHTN3 MINGW64 /d/git_new
$ ls
new_java_app/

DELL@DESKTOP-P6NHTN3 MINGW64 /d/git_new
$ cd new_java_app/

DELL@DESKTOP-P6NHTN3 MINGW64 /d/git_new/new_java_app (main)
$ ls
README.md

DELL@DESKTOP-P6NHTN3 MINGW64 /d/git_new/new_java_app (main)
$ ls
README.md

DELL@DESKTOP-P6NHTN3 MINGW64 /d/git_new/new_java_app (main)
$ ls -la
./  ../  .git/  README.md

DELL@DESKTOP-P6NHTN3 MINGW64 /d/git_new/new_java_app (main)
$ cd .git/

DELL@DESKTOP-P6NHTN3 MINGW64 /d/git_new/new_java_app/.git (GIT_DIR!)
$ ls
HEAD  config  description  hooks/  index  info/  logs/  objects/  packed-refs  refs/

DELL@DESKTOP-P6NHTN3 MINGW64 /d/git_new/new_java_app/.git (GIT_DIR!)
$ cd h
HEAD  hooks/

DELL@DESKTOP-P6NHTN3 MINGW64 /d/git_new/new_java_app/.git (GIT_DIR!)
$ cd hooks/

DELL@DESKTOP-P6NHTN3 MINGW64 /d/git_new/new_java_app/.git/hooks (GIT_DIR!)
$ ls
applypatch-msg.sample*  fsmonitor-watchman.sample*  pre-applypatch.sample*  pre-merge-commit.sample*  pre-rebase.sample*  prepare-commit-msg.sample*  sendemail-validate.sample*
commit-msg.sample*     post-update.sample*         pre-commit.sample*     pre-push.sample*         pre-receive.sample*  push-to-checkout.sample*  update.sample*

DELL@DESKTOP-P6NHTN3 MINGW64 /d/git_new/new_java_app/.git/hooks (GIT_DIR!)
$ vim
```

```
DELL@DESKTOP-P6NHTN3 MINGW64 /d/git_new/new_java_app/.git (GIT_DIR!)
$ cd h
HEAD  hooks/

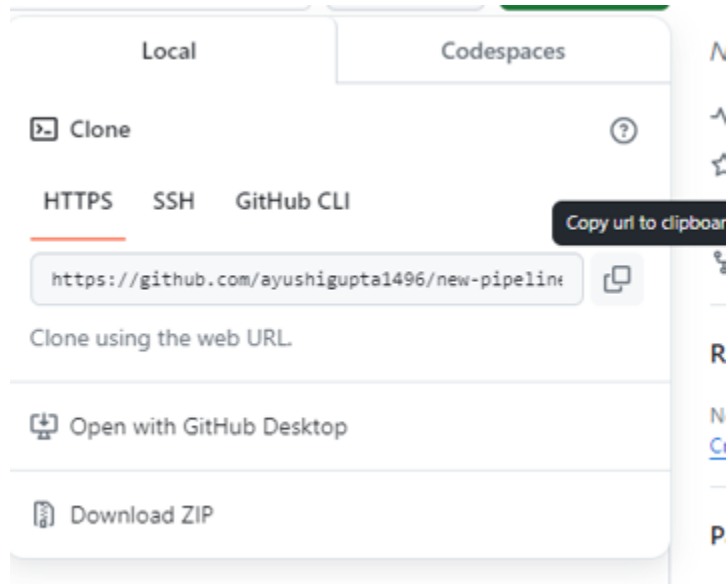
DELL@DESKTOP-P6NHTN3 MINGW64 /d/git_new/new_java_app/.git (GIT_DIR!)
$ cd hooks/

DELL@DESKTOP-P6NHTN3 MINGW64 /d/git_new/new_java_app/.git/hooks (GIT_DIR!)
$ ls
applypatch-msg.sample*  fsmonitor-watchman.sample*  pre-applypatch.sample*  pre-merge-commit.sample*
commit-msg.sample*     post-update.sample*         pre-commit.sample*     pre-push.sample*

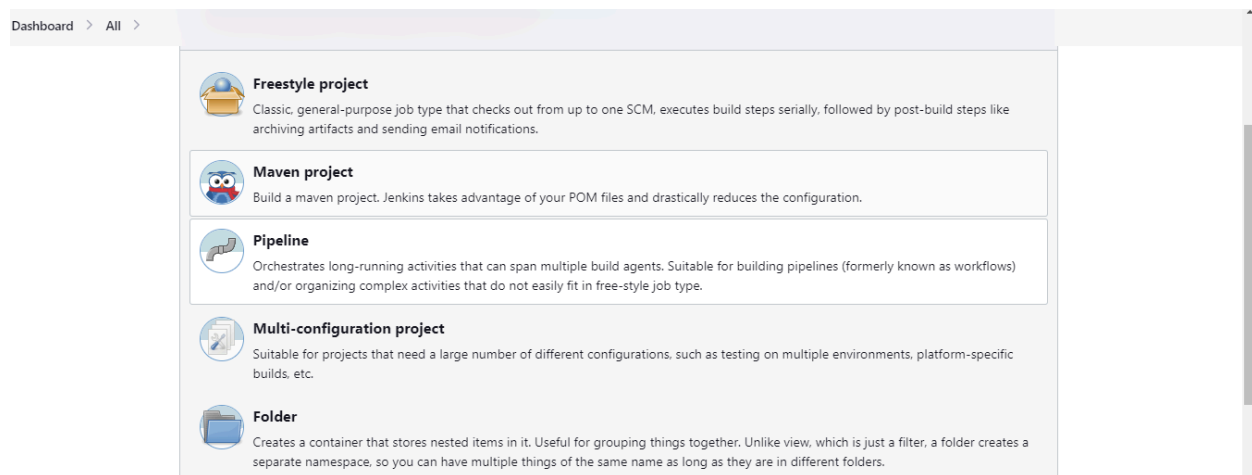
DELL@DESKTOP-P6NHTN3 MINGW64 /d/git_new/new_java_app/.git/hooks (GIT_DIR!)
$ vim post-commit
```

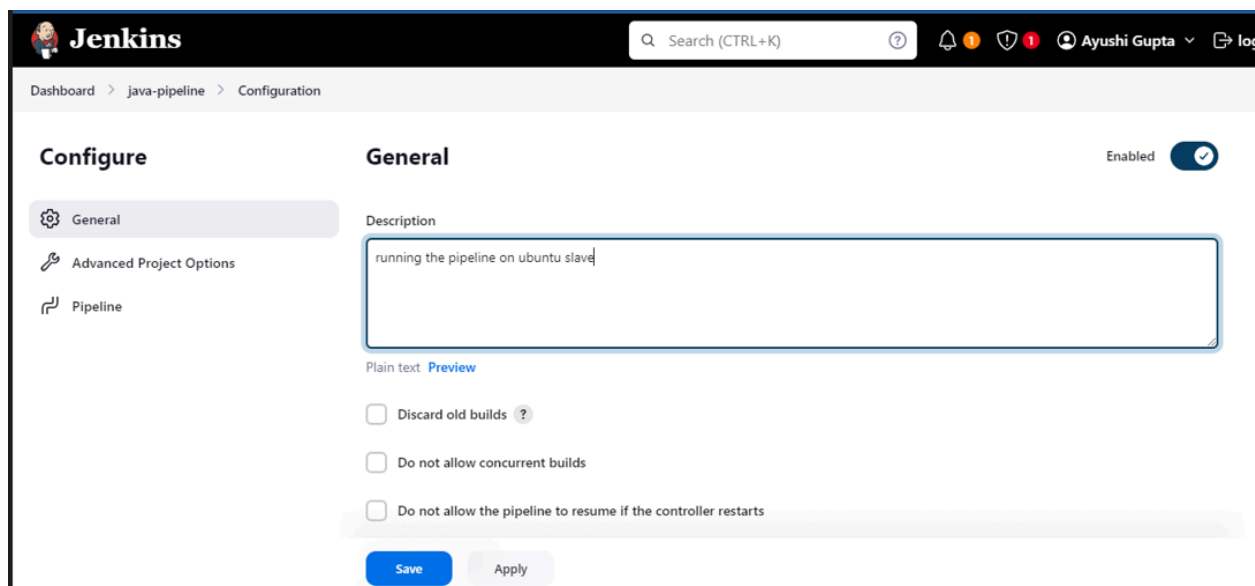
```
MINGW64:/d/git_new/new_java_app/.git/hooks
#!/bin/bash
git push -u origin main
```

Clone the repo on local system



Now create jenkins pipeline by choosing pipeline project on jenkins Web-UI





The image shows the Jenkins Configuration page for a pipeline named 'java-pipeline'. The 'General' tab is selected, and the 'Enabled' toggle is turned on. The 'Description' field contains the text 'running the pipeline on ubuntu slave'. Below the description, there are three unchecked checkboxes: 'Discard old builds', 'Do not allow concurrent builds', and 'Do not allow the pipeline to resume if the controller restarts'. At the bottom, there are 'Save' and 'Apply' buttons.

Jenkins Search (CTRL+K) Ayushi Gupta

Dashboard > java-pipeline > Configuration

Configure

- General
- Advanced Project Options
- Pipeline

General

Enabled

Description

running the pipeline on ubuntu slave

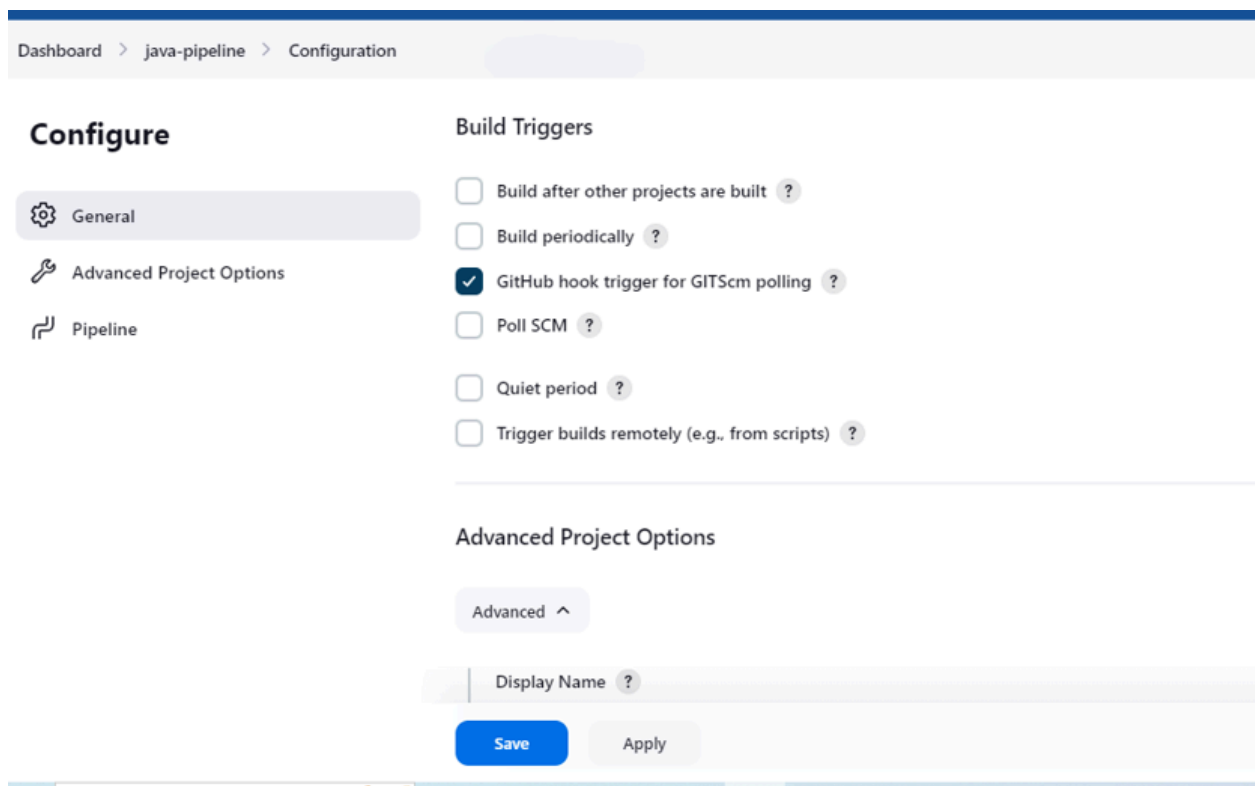
Plain text [Preview](#)

☐ Discard old builds

☐ Do not allow concurrent builds

☐ Do not allow the pipeline to resume if the controller restarts

[Save](#) [Apply](#)



The image shows the Jenkins Configuration page for a pipeline named 'java-pipeline'. The 'Build Triggers' section has the 'GitHub hook trigger for GITScm polling' option checked. The 'Advanced Project Options' section is expanded, showing the 'Display Name' field. At the bottom, there are 'Save' and 'Apply' buttons.

Dashboard > java-pipeline > Configuration

Configure

- General
- Advanced Project Options
- Pipeline

Build Triggers

☐ Build after other projects are built

☐ Build periodically

☒ GitHub hook trigger for GITScm polling

☐ Poll SCM

☐ Quiet period

☐ Trigger builds remotely (e.g., from scripts)

Advanced Project Options

Advanced

Display Name

[Save](#) [Apply](#)

Dashboard > java-pipeline > Configuration

Pipeline script from SCM

Configure

- General
- Advanced Project Options
- Pipeline

SCM ?

Git

Repositories ?

Repository URL ?

https://github.com/ayushigupta1496/new_java_app.git

Please enter Git repository.

Credentials ?

- none -

+ Add

Give the github repository URL

To Trigger the pipeline on Git push generate Webhooks in github

Instances | EC2 | ap-... | EC2 Instance Connec... | Add webhook | gouravaas/new-pipe... | java-pipeline [jenkin...]

github.com/ayushigupta1496/new_java_app/settings/hooks/new

What is Unmask and... | Job Search | Learnings | Freelancing

ayushigupta1496 / new_java_app

Type [] to search

<> Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

General

Access

Collaborators

Moderation options

Code and automation

Branches

Tags

Rules

Actions

Webhooks

Environments

Codespaces

Pages

Webhooks / Add webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

Payload URL *

https://example.com/postreceive

Content type

application/x-www-form-urlencoded

Secret

Which events would you like to trigger this webhook?

Just the push event.

Fig:Github Webhooks

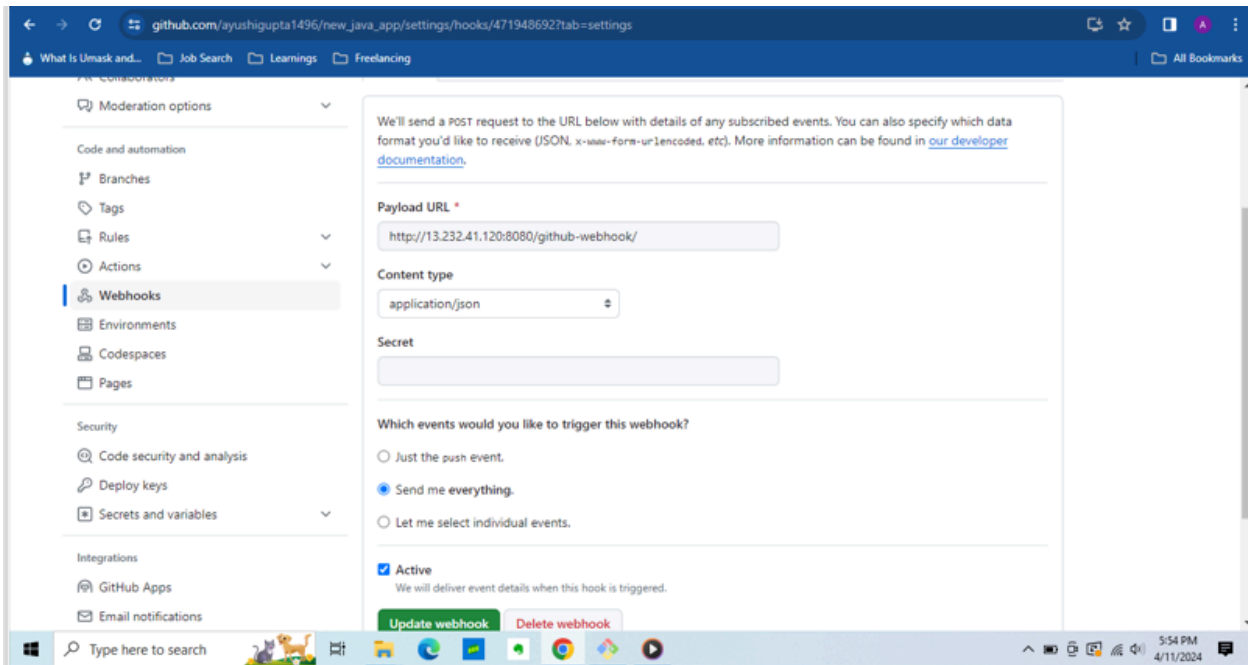
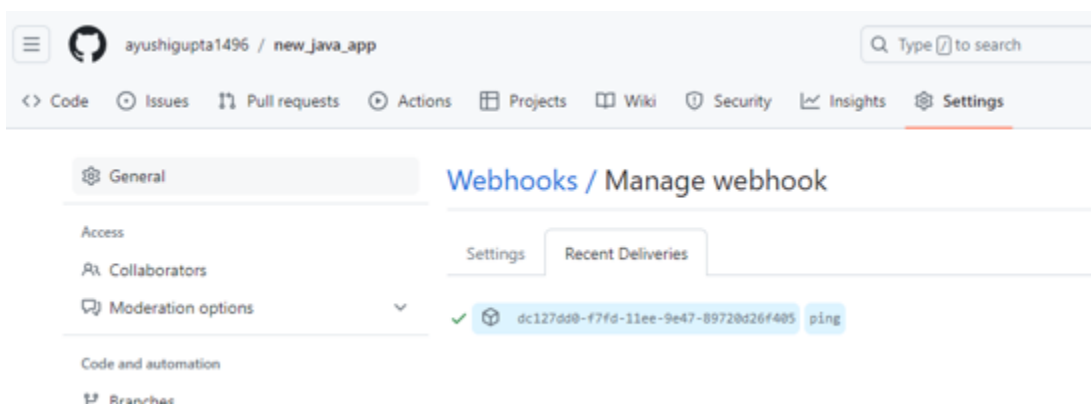


Fig: Adding Github Webhooks



Install Docker Plugin in Jenkins & Integrate Jenkins with Docker

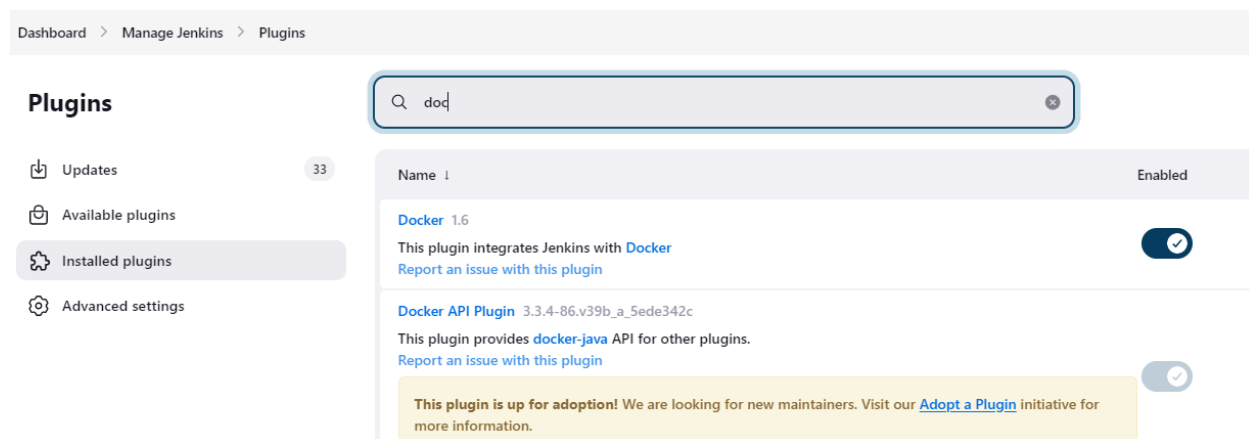
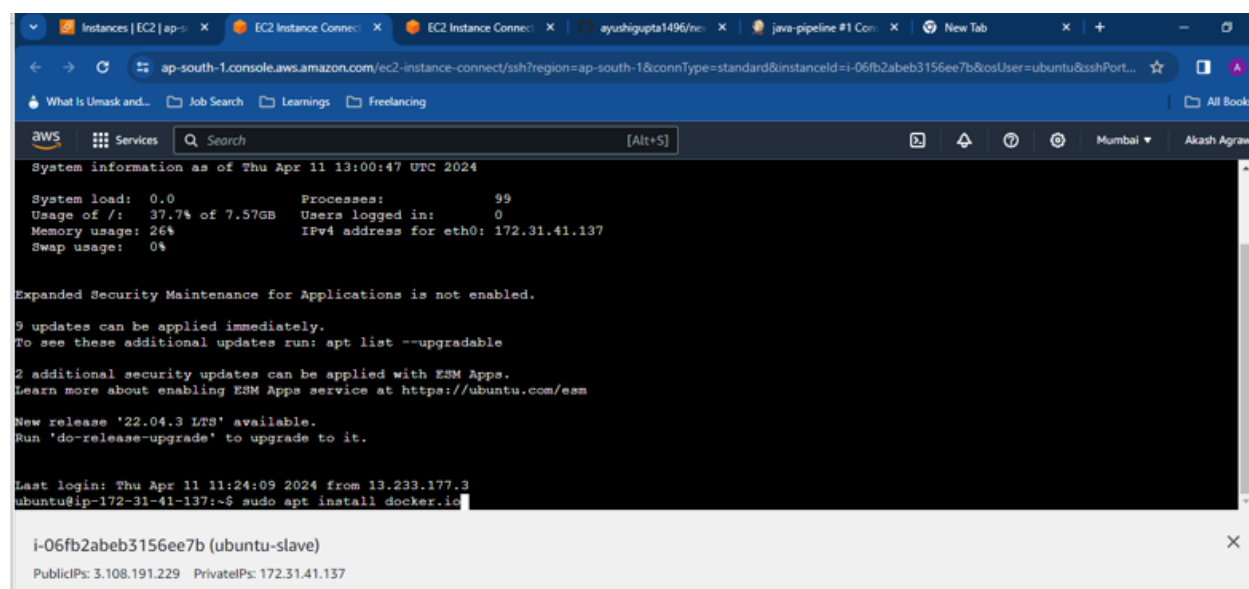
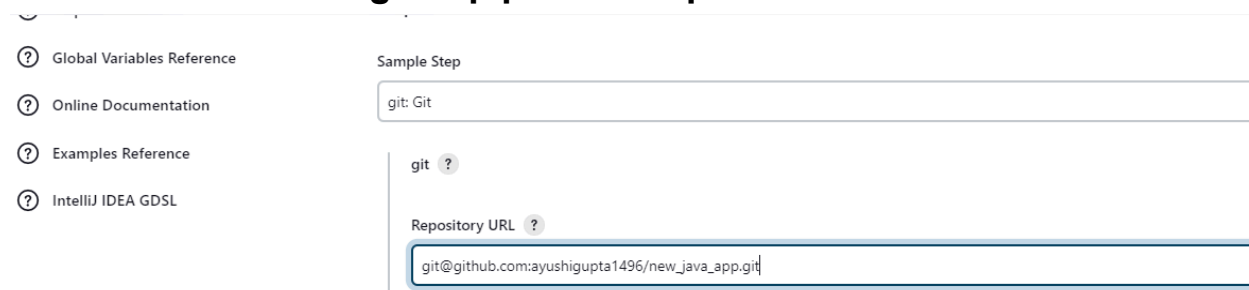


Fig: Integrate Docker with Jenkins



Now we start creating the pipeline script



Dashboard > java-pipeline > Pipeline Syntax

master

Credentials ?

- none -

+ Add ▾

☒ Include in polling? ?

☒ Include in changelog? ?

Generate Pipeline Script

git 'https://github.com/ayushigupta1496/new_java_app.git'

```
DELL@DESKTOP-P6NHTNJ MINGW64 /d/git_new
$ ls
new-pipelines/  new_java_app/

DELL@DESKTOP-P6NHTNJ MINGW64 /d/git_new
$ cd new_j
bash: cd: new_j: No such file or directory

DELL@DESKTOP-P6NHTNJ MINGW64 /d/git_new
$ cd new_java_app/

DELL@DESKTOP-P6NHTNJ MINGW64 /d/git_new/new_java_app (main)
$ ls
Dockerfile  Jenkinsfile  README.md  deploymentservice.yaml  pom.xml  src/

DELL@DESKTOP-P6NHTNJ MINGW64 /d/git_new/new_java_app (main)
$ vim Jenkinsfile |
```

Create Jenkinsfile to write pipeline script

Stage 1 : Pulling the code from SCM

```
pipeline {
    agent {
        label "ubuntu-slave"
    }
    stages {
        stage ("Pulling the code from SCM") {
            steps {
                git branch: 'main', url: 'https://github.com/ayushigupta1496/new_java_app.git'
            }
        }
    }
}
```

pull the code from git repository

```
pipeline {
  agent {
    label "ubuntu-slave"
  }
  stages {
    stage ("Pulling the code from SCM") {
      steps {
        git branch: 'main', url: 'https://github.com/ayushigupta1496/new_java_app.git'
      }
    }
    stage ("Build the code") {
      steps {
        sh 'sudo mvn dependency:purge-local-repository'
        sh 'sudo mvn clean package'
      }
    }
  }
}
```

Stage 2: Build the code from Maven

Update Tomcat Docker File to automate deployment process

**FROM tomcat:latest RUN cp -R /usr/local/tomcat webpp.dist/*
/usr/local/tomcatwebapps COPY ./*.war /usr/local/tomcatwebapps**

```
FROM tomcat:8.0.20-jre8
COPY target/java-web-app*.war /usr/local/tomcat/webapps/java-web-app.war
~
~
~
~
~
```

Dockerfile containing Docker Image

Stage 3: Create docker image

```
pipeline {
  agent {
    label "ubuntu-slave"
  }
  stages {
    stage ("Pulling the code from SCM") {
      steps {
        git branch: 'main', url: 'https://github.com/ayushigupta1496/new_java_app.git'
      }
    }
    stage ("Build the code") {
      steps {
        sh 'sudo mvn dependency:purge-local-repository'
        sh 'sudo mvn clean package'
      }
    }
    stage ("Create docker image"){
      steps {
        sh 'sudo docker build -t java-app:$BUILD_TAG .'
        sh 'sudo docker tag java-app:$BUILD_TAG ayushigupta1496/java-app:$BUILD_TAG'
      }
    }
  }
}
```

By : Ayushi Gupta

Email : ayushigupta1496@gmail.com

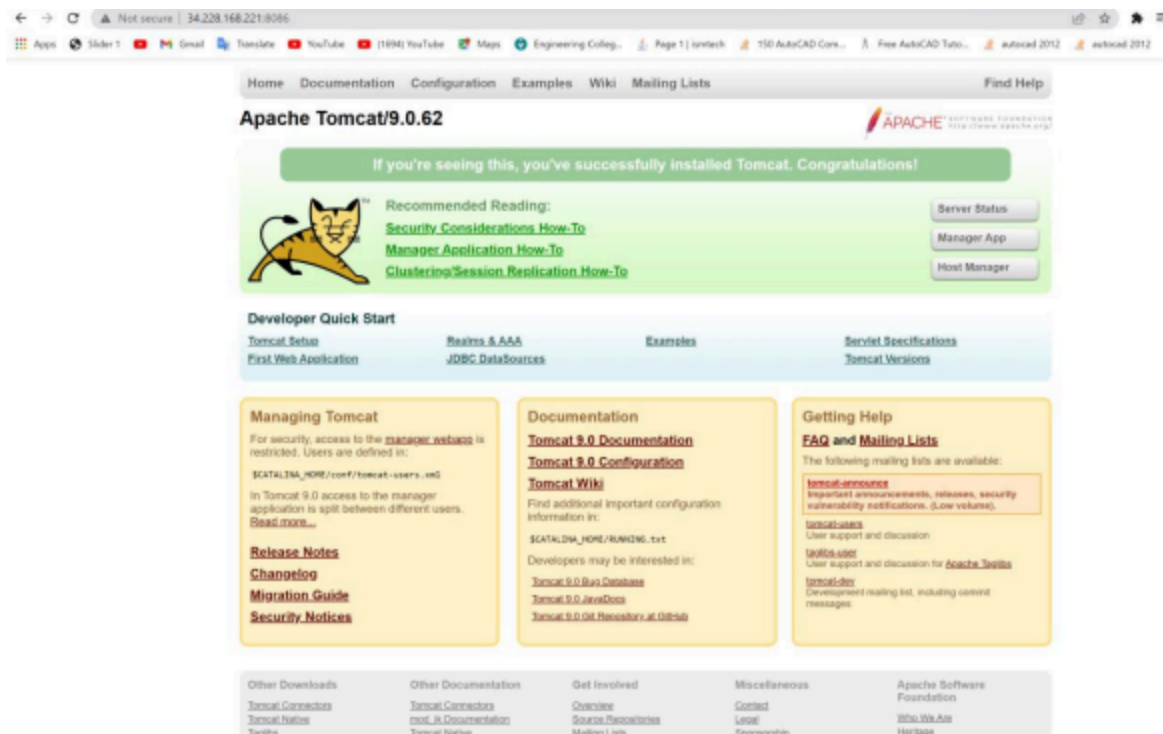


Fig. Tomcat Started

push the image on DockerHub for that we have to save Dockerhub Credentials to jenkins Global credentials

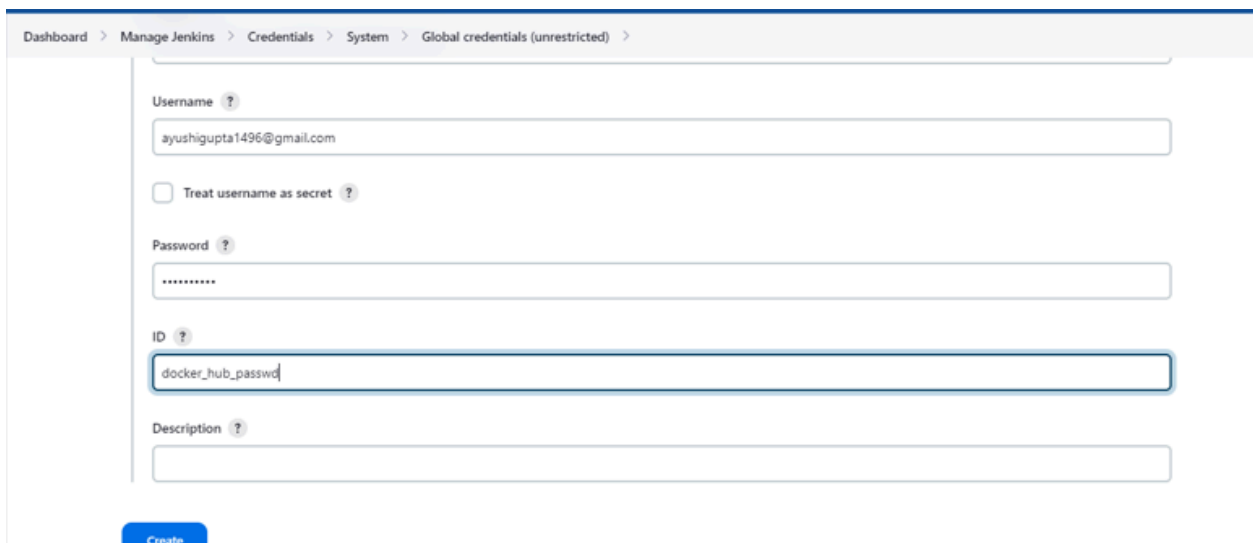


Fig:Dockerhub credentials

Dashboard > java-pipeline > Pipeline Syntax

Secret text ?

Variable ?

docker_hub_pass_var

Credentials ?

docker_passwd

+ Add ▾

Add ▾

Generate Pipeline Script

```
withCredentials([string(credentialsId: 'Docker_pass_ID', variable: 'docker_hub_pass_var')]) {  
    // some block  
}
```

Stage 4 : push the image on dockerhub

```
pipeline {  
    agent {  
        label "ubuntu-slave"  
    }  
    stages {  
        stage ("Pulling the code from SCM") {  
            steps {  
                git branch: 'main', url: 'https://github.com/ayushigupta1496/new_java_app.git'  
            }  
        }  
        stage ("Build the code") {  
            steps {  
                sh 'sudo mvn dependency:purge-local-repository'  
                sh 'sudo mvn clean package'  
            }  
        }  
        stage ("Create docker image"){  
            steps {  
                sh 'sudo docker build -t java-app:$BUILD_TAG .'  
                sh 'sudo docker tag java-app:$BUILD_TAG ayushigupta1496/java-app:$BUILD_TAG'  
            }  
        }  
        stage ("Push on Docker-Hub"){  
            steps {  
                withCredentials([string(credentialsId: 'Docker_pass_ID', variable: 'docker_hub_pass_var')]) {  
                    sh 'sudo docker login -u ayushigupta1496 -p ${docker_hub_pass_var}'  
                    sh 'sudo docker push ayushigupta1496/java-app:$BUILD_TAG'  
                }  
            }  
        }  
    }  
}
```

Stage 5 :Testing the code

```
pipeline {
  agent {
    label "ubuntu-slave"
  }
  stages {
    stage ("Pulling the code from SCM") {
      steps {
        git branch: 'main', url: 'https://github.com/ayushigupta1496/new_java_app.git'
      }
    }
    stage ("Build the code") {
      steps {
        sh 'sudo mvn dependency:purge-local-repository'
        sh 'sudo mvn clean package'
      }
    }
    stage ("Create docker image"){
      steps {
        sh 'sudo docker build -t java-app:$BUILD_TAG .'
        sh 'sudo docker tag java-app:$BUILD_TAG ayushigupta1496/java-app:$BUILD_TAG'
      }
    }
    stage ("Push on Docker-Hub"){
      steps {
        withCredentials([string(credentialsId: 'Docker_pass_ID', variable: 'docker_hub_pass_var')]) {
          sh 'sudo docker login -u ayushigupta1496 -p ${docker_hub_pass_var}'
          sh 'sudo docker push ayushigupta1496/java-app:$BUILD_TAG'
        }
      }
    }
    stage (" Testing the pipeline" ){
      steps {
        sh 'sudo docker run -dit --name java-test$BUILD_TAG -p 8083:8080 java-app:$BUILD_TAG'
      }
    }
  }
}
```

The screenshot displays the Jenkins web interface for a pipeline named 'java-pipeline'. The 'Stage View' is active, showing a table of stages and their execution times. The stages are: Declarative: Checkout SCM, Pulling the code from SCM, Build the code, Create docker image, Push on Docker-Hub, and Testing the pipeline. The table shows the average stage times and the actual times for the current build (#15).

Stage	Declarative: Checkout SCM	Pulling the code from SCM	Build the code	Create docker image	Push on Docker-Hub	Testing the pipeline
Average stage times: (Average full run time: ~36s)	957ms	615ms	14s	958ms	15s	1s
#15 (Apr 22 18:06)	957ms	615ms	14s	958ms	15s	1s

Below the table, there is a 'Permalinks' section with a list of links for the last build, last stable build, last successful build, and last failed build.

- Last build (#14), 7.7 sec ago
- Last stable build (#13), 5 days 19 hr ago
- Last successful build (#13), 5 days 19 hr ago
- Last failed build (#12), 5 days 19 hr ago

On the left side, there is a 'Build History' section showing a list of builds with their status and timestamps.

Stage 6 : QAT Testing



fig:QAT Testing completed

Now the next step is Deployment: For that we use kubernetes

Kubernetes is an open source container orchestration engine for automating deployment, scaling, and management of containerized applications.

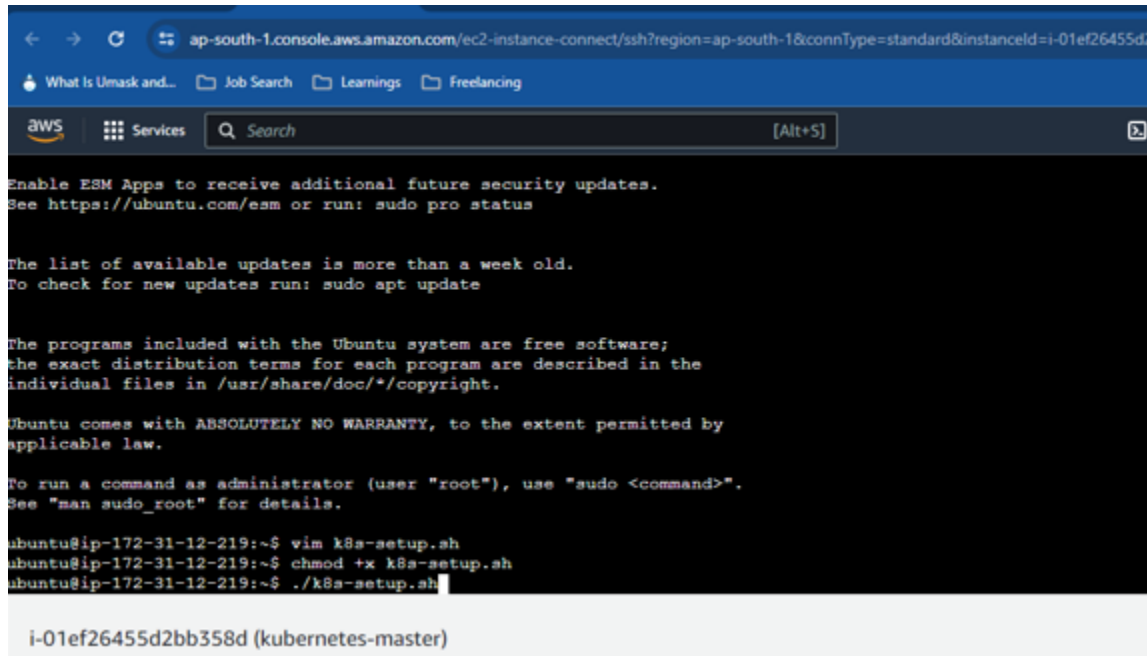
For deployment we launch two other server i.e kubernetes master & kubernetes worker

Setup of k8s-master node

- Run jenkins Installation Commands to make node jenkins slave
- Run kubernetes setup commands

Setup of k8s worker node

- Run kubernetes setup commands
- Run k8s join token command to make it a worker node



```

ap-south-1.console.aws.amazon.com/ec2-instance-connect/ssh?region=ap-south-1&connType=standard&instanceId=i-01ef26455d2
What Is Umask and... Job Search Learnings Freelancing
aws Services Search [Alt+S]
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-12-219:~$ vim k8s-setup.sh
ubuntu@ip-172-31-12-219:~$ chmod +x k8s-setup.sh
ubuntu@ip-172-31-12-219:~$ ./k8s-setup.sh

i-01ef26455d2bb358d (kubernetes-master)

```

Now we will run set-up command to install kubernetes on kubernetes-master server

SET-UP COMMANDS FOR K8S

kubernetes

Learning k8s configuration

```
sudo apt-get update -y
```

```
sudo apt-get install -y apt-transport-https ca-certificates curl
```

```
sudo curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --dearmor -o
/etc/apt/keyrings/kubernetes-apt-keyring.gpg
```

```
sudo mkdir -p -m 755 /etc/apt/keyrings
```

```
sudo curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --dearmor -o
/etc/apt/keyrings/kubernetes-apt-keyring.gpg
```

```
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
```

```
https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
```

```
sudo apt-get update
```

```
sudo apt-get install -y kubelet kubeadm kubectl
```

```
sudo apt-mark hold kubelet kubeadm kubectl
sudo sed -i 's/^(\s*)$/#\1/g' /etc/fstab
sudo swapoff -a
sudo modprobe overlay
sudo modprobe br_netfilter
```

```
sudo tee /etc/sysctl.d/kubernetes.conf<<EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
EOF
```

```
sudo sysctl --system
```

```
sudo tee /etc/modules-load.d/containerd.conf <<EOF
overlay
br_netfilter
EOF
```

```
sudo sysctl --system
```

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release
-cs) stable"
```

```
sudo apt update
```

```
sudo apt install -y containerd.io
```

```
mkdir -p /etc/containerd
```

```
containerd config default | sudo tee /etc/containerd/config.toml
```

```
sudo systemctl restart containerd
```

```
sudo systemctl enable containerd
```

```
sudo systemctl enable kubelet
```

```
kubectl version
```

```
sudo kubeadm config images pull --cri-socket /run/containerd/containerd.sock --kubernetes-version
v1.30.0
```



```
sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --upload-certs --kubernetes-version=v1.30.0
--control-plane-endpoint=ip --ignore-preflight-errors=all --cri-socket /run/containerd/containerd.sock
```

```
mkdir -p $HOME/.kube
```

```
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
```

```
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

```
kubectrl apply -f https://github.com/coreos/flannel/raw/master/Documentation/kube-flannel.yml
```



```
aws Services Search [Alt+S]
sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f (podnetwork).yaml" with one of the options listed at:
https://kubernetes.io/docs/concepts/cluster-administration/addons/

You can now join any number of the control-plane node running the following command on each as root:

kubeadm join 172.31.15.196:6443 --token o7nnjk.uuvunrzmz6r0421f \
--discovery-token-ca-cert-hash sha256:f4b47af46104f50c135587eb4c8bc0d2d60749c0ac00147a2d2d00f44477c131 \
--control-plane --certificate-key d2539856de9dfba6a1c524db2a5876d5e2e52b0148ff0b95ec9c5c2df1052176

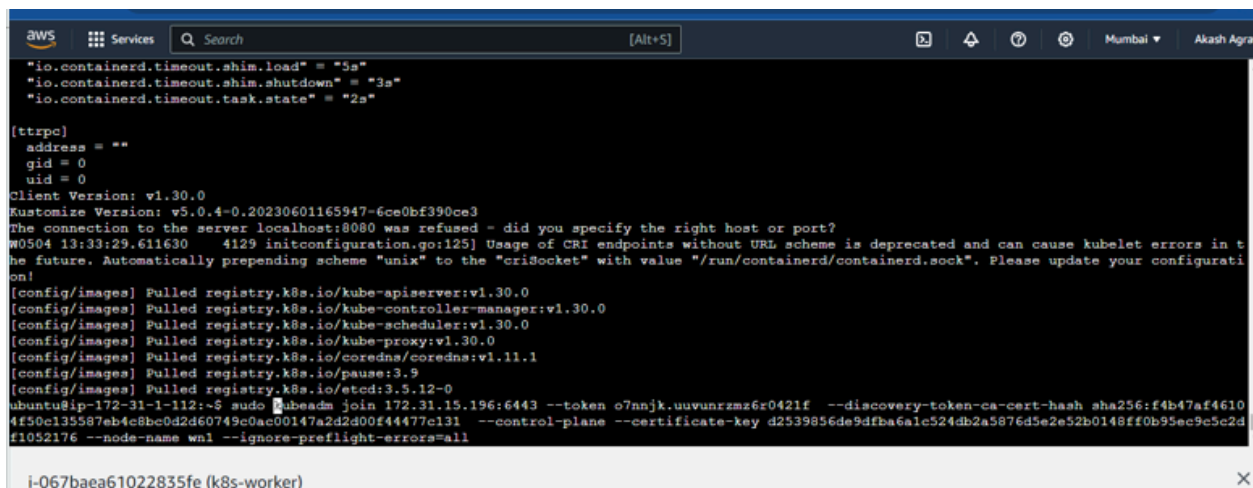
Please note that the certificate-key gives access to cluster sensitive data, keep it secret!
As a safeguard, uploaded-certs will be deleted in two hours; If necessary, you can use
"kubeadm init phase upload-certs --upload-certs" to reload certs afterward.

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 172.31.15.196:6443 --token o7nnjk.uuvunrzmz6r0421f \
--discovery-token-ca-cert-hash sha256:f4b47af46104f50c135587eb4c8bc0d2d60749c0ac00147a2d2d00f44477c131
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
```

Fig: With the help of kubeadm join token we can join worker nodes

Now we will launch an ec2 instance as kubernetes worker node



```
aws Services Search [Alt+S] Mumbai Akash Agra

"io.containerd.timeout.shim.load" = "5s"
"io.containerd.timeout.shim.shutdown" = "3s"
"io.containerd.timeout.task.state" = "2s"

[tttrpc]
address = ""
gid = 0
uid = 0

Client Version: v1.30.0
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
The connection to the server localhost:8080 was refused - did you specify the right host or port?
W0504 13:33:29.611630 4129 initconfiguration.go:125] Usage of CRI endpoints without URL scheme is deprecated and can cause kubelet errors in the future. Automatically prepending scheme "unix" to the "criSocket" with value "/run/containerd/containerd.sock". Please update your configuration!
[config/images] Pulled registry.k8s.io/kube-apiserver:v1.30.0
[config/images] Pulled registry.k8s.io/kube-controller-manager:v1.30.0
[config/images] Pulled registry.k8s.io/kube-scheduler:v1.30.0
[config/images] Pulled registry.k8s.io/kube-proxy:v1.30.0
[config/images] Pulled registry.k8s.io/coredns/coredns:v1.11.1
[config/images] Pulled registry.k8s.io/pause:3.9
[config/images] Pulled registry.k8s.io/etcd:3.5.12-0
ubuntu@ip-172-31-1-112:~$ sudo kubeadm join 172.31.15.196:6443 --token o7nnjk.uuvunrzmz6r0421f --discovery-token-ca-cert-hash sha256:f4b47af46104f50c135587eb4c8bc0d2d60749c0ac00147a2d2d00f44477c131 --control-plane --certificate-key d2539856de9dfba6a1c524db2a5876d5e2e52b0148ff0b95ec9c5c2df1052176 --node-name wn1 --ignore-preflight-errors=all

i-067baea61022835fe (k8s-worker)
```

To make the node kubernetes worker-node run the following commands:

```
sudo apt-get update -y
sudo apt-get install -y apt-transport-https ca-certificates curl

sudo curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --dearmor -o
/etc/apt/keyrings/kubernetes-apt-keyring.gpg

sudo mkdir -p -m 755 /etc/apt/keyrings

sudo curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --dearmor -o
/etc/apt/keyrings/kubernetes-apt-keyring.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

sudo apt-get update

sudo apt-get install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl
sudo sed -i ' / swap / s/^(.*)$/#\1/g' /etc/fstab
sudo swapoff -a
sudo modprobe overlay
sudo modprobe br_netfilter

sudo tee /etc/sysctl.d/kubernetes.conf<<EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
EOF

sudo sysctl --system

sudo tee /etc/modules-load.d/containerd.conf <<EOF
overlay
br_netfilter
EOF

sudo sysctl --system

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release
-cs) stable"

sudo apt update
```

```
sudo apt install -y containerd.io
```

```
mkdir -p /etc/containerd
```

```
containerd config default | sudo tee /etc/containerd/config.toml
```

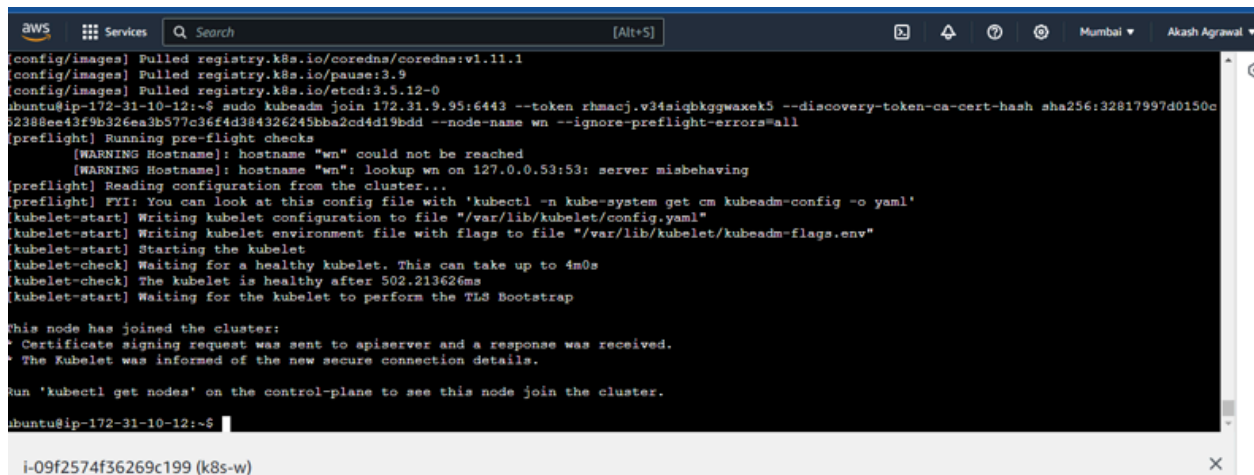
```
sudo systemctl restart containerd
```

```
sudo systemctl enable containerd
```

```
sudo systemctl enable kubelet
```

```
kubctl version
```

```
sudo kubeadm config images pull --cri-socket /run/containerd/containerd.sock --kubernetes-version v1.30.0
```

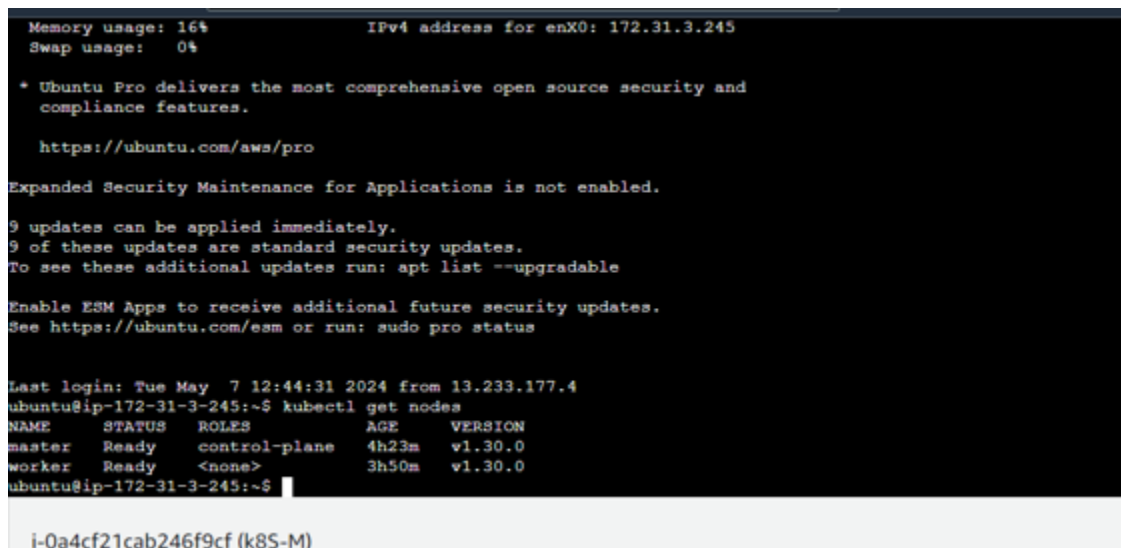


```

aws Services Search [Alt+S] Mumbai Akash Agrawal
[config/images] Pulled registry.k8s.io/coredns/coredns:v1.11.1
[config/images] Pulled registry.k8s.io/pause:3.9
[config/images] Pulled registry.k8s.io/etcd:3.5.12-0
ubuntu@ip-172-31-10-12:~$ sudo kubeadm join 172.31.9.95:6443 --token rhmacj.v34siqbkggwaxek5 --discovery-token-ca-cert-hash sha256:32817997d0150c
32388ee43f9b326ea3b577c36f4d384326245bba2cd4d19bdd --node-name wn --ignore-preflight-errors=all
[preflight] Running pre-flight checks
[WARNING Hostname]: hostname "wn" could not be reached
[WARNING Hostname]: hostname "wn": lookup wn on 127.0.0.53:53: server misbehaving
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectcl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 502.213626ms
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap
This node has joined the cluster:
* Certificate signing request was sent to apiserer and a response was received.
* The Kubelet was informed of the new secure connection details.
Run 'kubectcl get nodes' on the control-plane to see this node join the cluster.
ubuntu@ip-172-31-10-12:~$
i-09f2574f36269c199 (k8s-w)

```

Fig: join token run the node as worker node



```

Memory usage: 16% IPv4 address for enX0: 172.31.3.245
Swap usage: 0%
* Ubuntu Pro delivers the most comprehensive open source security and
compliance features.
https://ubuntu.com/aws/pro
Expanded Security Maintenance for Applications is not enabled.
9 updates can be applied immediately.
9 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
Last login: Tue May 7 12:44:31 2024 from 13.233.177.4
ubuntu@ip-172-31-3-245:~$ kubectcl get nodes
NAME      STATUS    ROLES    AGE      VERSION
master    Ready     control-plane  4h23m    v1.30.0
worker    Ready     <none>      3h50m    v1.30.0
ubuntu@ip-172-31-3-245:~$
i-0a4cf21cab246f9cf (k8S-M)

```

STAGE- 7 -Approval From Testing Team

Now before deployment we put a boolean parameter which will create a pop-up message to proceed or abort the process

```

sh 'sudo mvn dependency:purge-local-repository'
sh 'sudo mvn clean package'
}
}
stage ("Create docker image"){
  steps {
    sh 'sudo docker build -t java-app:$BUILD_TAG .'
    sh 'sudo docker tag java-app:$BUILD_TAG ayushigupta1496/java-app:$BUILD_TAG'
  }
}
stage ("Push on Docker-Hub"){
  steps {
    withCredentials([string(credentialsId: 'Docker_pass_ID', variable: 'docker_hub_pass_var')]) {
      sh 'sudo docker login -u ayushigupta1496 -p ${docker_hub_pass_var}'
      sh 'sudo docker push ayushigupta1496/java-app:$BUILD_TAG'
    }
  }
}
stage (" Testing the pipeline" ){
  steps {
    sh 'sudo docker run -dit --name java-test$BUILD_TAG -p 8083:8080 java-app:$BUILD_TAG'
  }
}
stage ("QAT Testing"){
  steps {
    retry(5) {
      script {
        sh 'sudo curl --silent http://13.233.224.99:8083/java-web-app/ | grep -i -E "(india|sr)"'
      }
    }
  }
}
stage ("Approval from QAT"){
  steps {
    script {
      Boolean userInput = input(id: 'Proceed1', message: 'Do you want to Promote this build?', parameters: [[class: 'BooleanParameterDefinition', defaultValu
e: true, description: '', name: 'Please confirm you agree with this']])
      echo 'userInput: ' + userInput
    }
  }
}
}
}

```

```

sh 'sudo mvn dependency:purge-local-repository'
sh 'sudo mvn clean package'
}
}
stage ("Create docker image"){
  steps {
    sh 'sudo docker build -t java-app:$BUILD_TAG .'
    sh 'sudo docker tag java-app:$BUILD_TAG ayushigupta1496/java-app:$BUILD_TAG'
  }
}
stage ("Push on Docker-Hub"){
  steps {
    withCredentials([string(credentialsId: 'Docker_pass_ID', variable: 'docker_hub_pass_var')]) {
      sh 'sudo docker login -u ayushigupta1496 -p ${docker_hub_pass_var}'
      sh 'sudo docker push ayushigupta1496/java-app:$BUILD_TAG'
    }
  }
}
stage (" Testing the pipeline" ){
  steps {
    sh 'sudo docker run -dit --name java-test$BUILD_TAG -p 8083:8080 java-app:$BUILD_TAG'
  }
}
stage ("QAT Testing"){
  steps {
    retry(5) {
      script {
        sh 'sudo curl --silent http://13.233.224.99:8083/java-web-app/ | grep -i -E "(india|sr)"'
      }
    }
  }
}
stage ("Approval from QAT"){
  steps {
    script {
      Boolean userInput = input(id: 'Proceed1', message: 'Do you want to Promote this build?', parameters: [[class: 'BooleanParameterDefinition', defaultValu
e: true, description: '', name: 'Please confirm you agree with this']])
      echo 'userInput: ' + userInput
    }
  }
}
}
}

```

```
script {
```

```
Boolean userInput = input(id: 'Proceed1', message: 'Do you want to Promote this build?',
parameters: [[class: 'BooleanParameterDefinition', defaultValue: true, description: "", name: 'Please confirm you
agree with this']])
```

```
echo 'userInput: ' + userInput
```

```
}
```

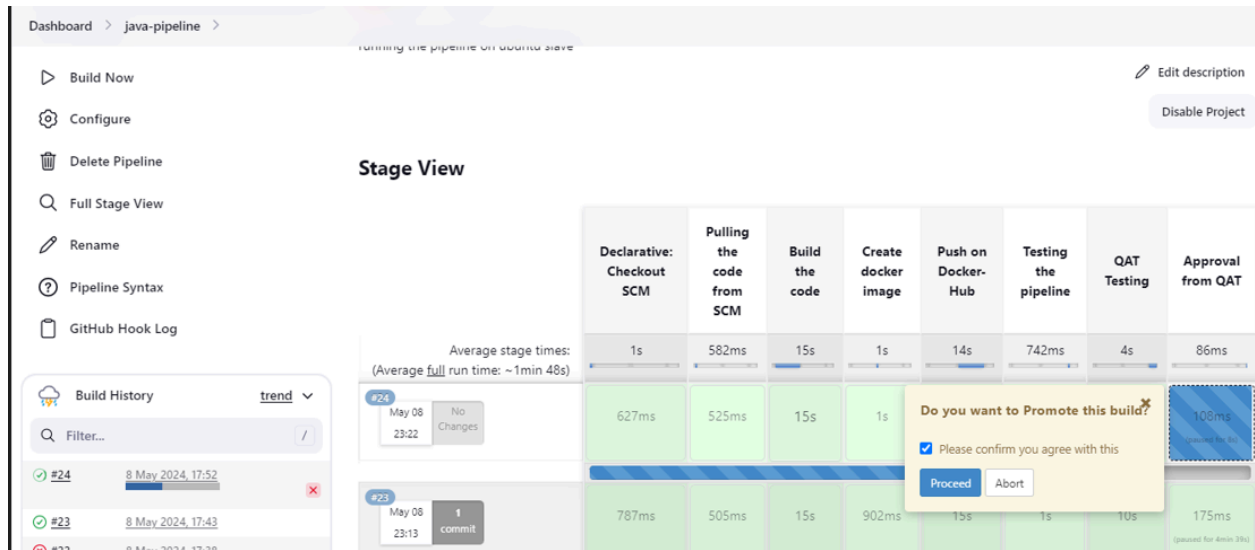
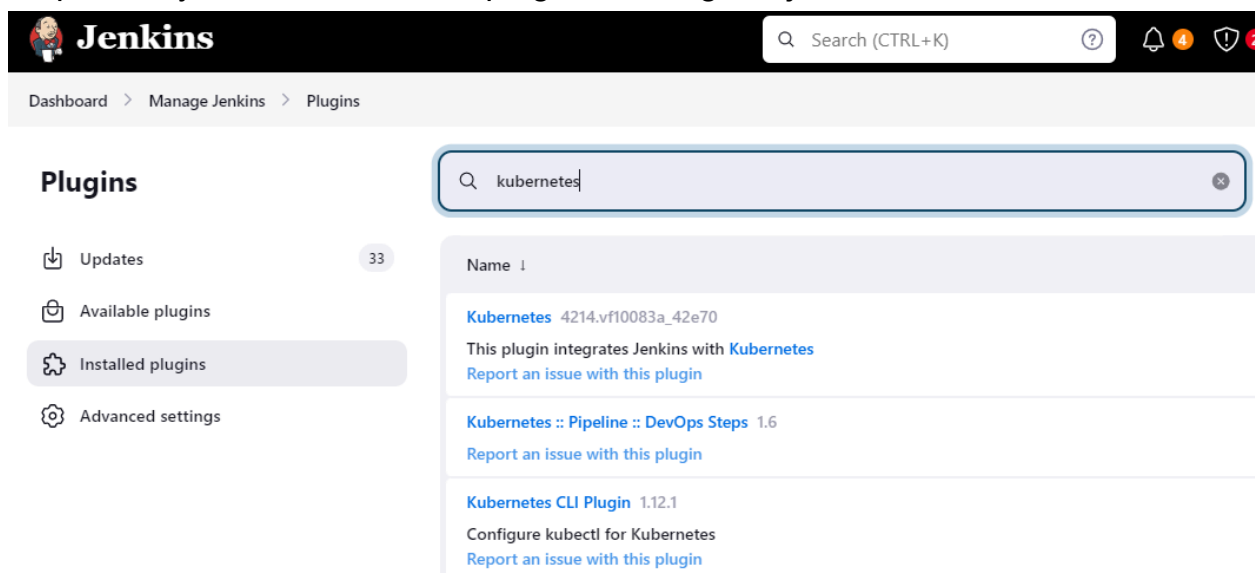


Fig: pipeline build successfully

STAGE-8 Deployment on kubernetes

Step1 firstly Install kubernetes plugins to integrate jenkins and kubernetes



Step-2 creating a deployment.yaml & service.

```
MINGW64/d/git_new/new_java_app
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tomcat-app-deployment
spec:
  selector:
    matchLabels:
      app: tomcat-app
  replicas: 2
  template:
    metadata:
      labels:
        app: tomcat-app
    spec:
      containers:
        - name: tomcat-app
          image: ayushigupta1496/java-app:{{BUILD_TAG}}
          imagePullPolicy: IfNotPresent
          ports:
            - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: tomcat-appsvc
spec:
  selector:
    app: tomcat-app
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 8080
  type: NodePort
```

Fig: Deploymentservice.yaml file

Step-3 Now to make jenkins master to take session of k8s-master where deployment will take place we have to Install SSHAgent-Plugin

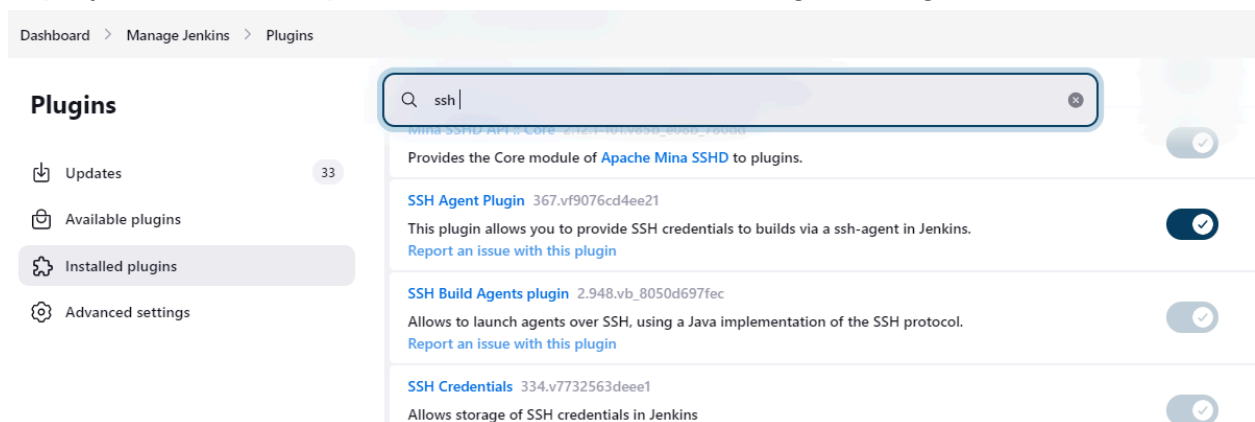
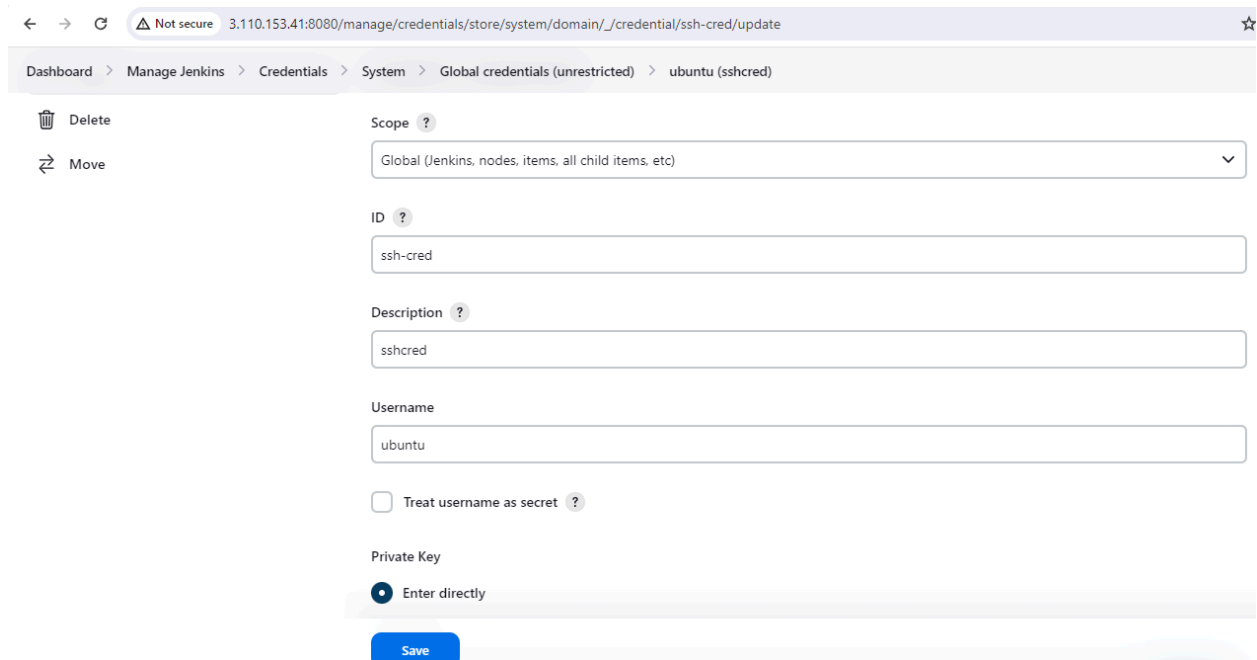


Fig: ssh- agent plugin

Step-4 Generate ssh credentials on jenkins web-UI



The screenshot shows the Jenkins web-UI interface for creating or updating SSH credentials. The breadcrumb navigation is: Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) > ubuntu (sshcred). On the left, there are 'Delete' and 'Move' actions. The main form includes: a 'Scope' dropdown menu set to 'Global (Jenkins, nodes, items, all child items, etc)'; an 'ID' text field containing 'ssh-cred'; a 'Description' text field containing 'sshcred'; a 'Username' text field containing 'ubuntu'; a checkbox for 'Treat username as secret' which is unchecked; and a 'Private Key' section with a radio button selected for 'Enter directly'. A blue 'Save' button is at the bottom.

Fig: ssh credentials

```
DELL@DESKTOP-P6NHTNJ MINGW64 /d/git_new
$ ls
new-pipelines/  new_java_app/

DELL@DESKTOP-P6NHTNJ MINGW64 /d/git_new
$ cd new_java_app/

DELL@DESKTOP-P6NHTNJ MINGW64 /d/git_new/new_java_app (main)
$ ls
Dockerfile Jenkinsfile README.md deploymentservice.yaml pom.xml src/

DELL@DESKTOP-P6NHTNJ MINGW64 /d/git_new/new_java_app (main)
$ |
```

Step-5 Now as our deploymentservice.yaml file is on github so we have to pull the code from github and copy it to k8s-master server and then run the deployment

```
stage ("QAT Testing"){
  steps {
    retry(5) {
      script {
        sh 'sudo curl --silent http://13.233.224.99:8083/java-web-app/ | grep -i -E "(india|sr)"'
      }
    }
  }
}
stage ("Approval from QAT"){
  steps {
    script {
      Boolean userInput = input(id: 'Proceed1', message: 'Do you want to Promote this build?', parameters: [[class: 'BooleanParameterDefinition', defaultValu
e: true, description: '', name: 'Please confirm you agree with this']])
      echo 'userInput: ' + userInput
    }
  }
}
stage('Deploy to K8s') {
  steps {
    sshagent(credentials: ['ssh-cred']) {
      git branch: 'main', url: 'https://github.com/ayushigupta1496/new_java_app.git'
      sh 'scp -r -o StrictHostKeyChecking=no deploymentservice.yaml ubuntu@3.108.215.162:/home/ubuntu/'
      script {
        sh 'ssh ubuntu@3.108.215.162 kubectl apply -f deploymentservice.yaml'
      }
    }
  }
}
```

Fig: pipeline script for deployment

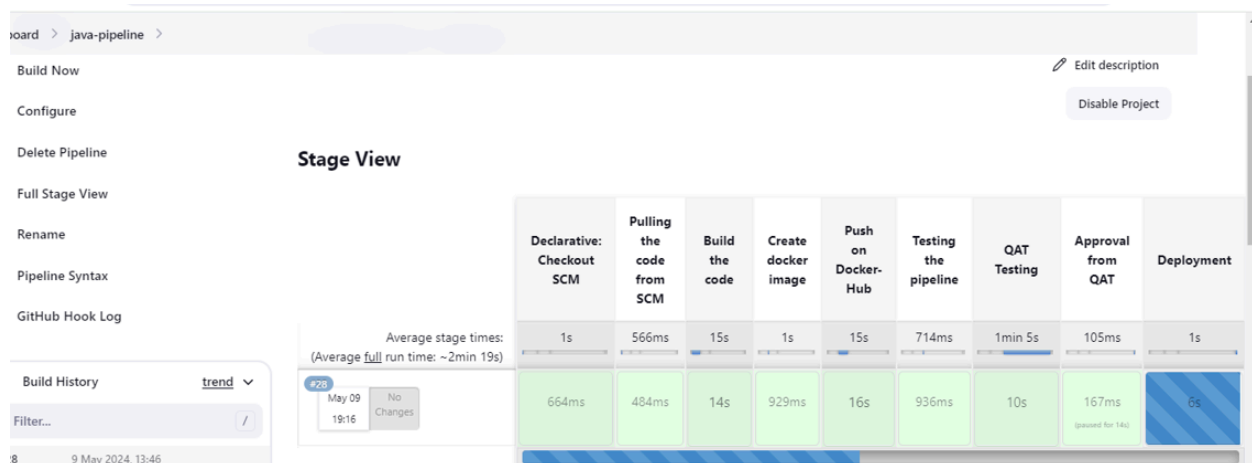


Fig: pipeline script is running

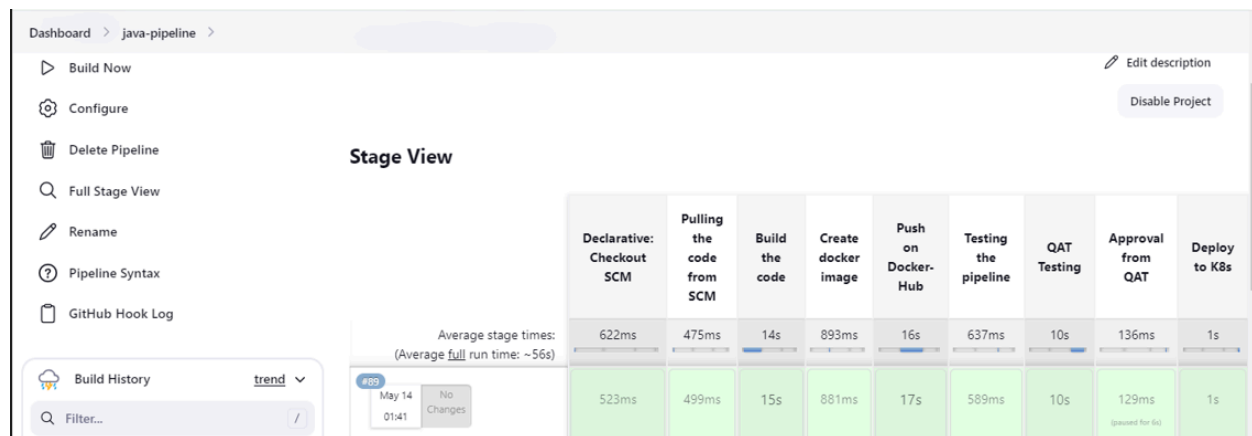


Fig: Deployment is completed Successfully