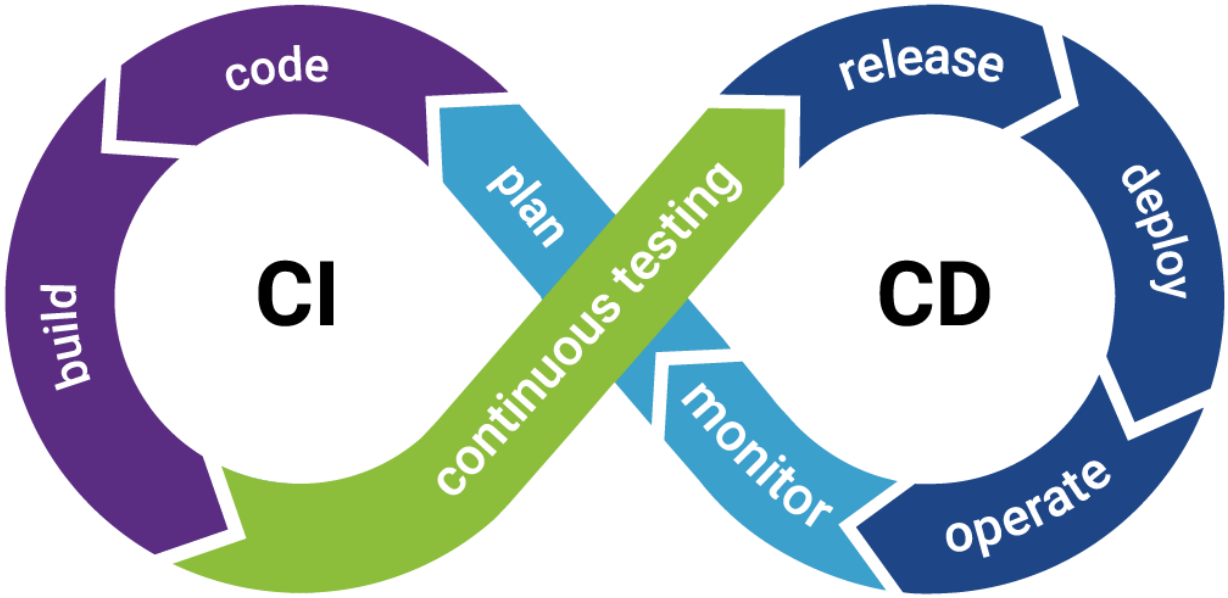
By : Ayushi Gupta Email : ayushigupta1496@gmail.com

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
 ~~  \_#####\
 ~~     \###|       AL2 End of Life is 2025-06-30.
 ~~       \#/ ___
  ~~       V~' '->
   ~~~         /    A newer version of Amazon Linux is available!
     ~~._.   _/
        _/ _/       Amazon Linux 2023, GA and supported until 2028-03-15.
      _/m/'           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

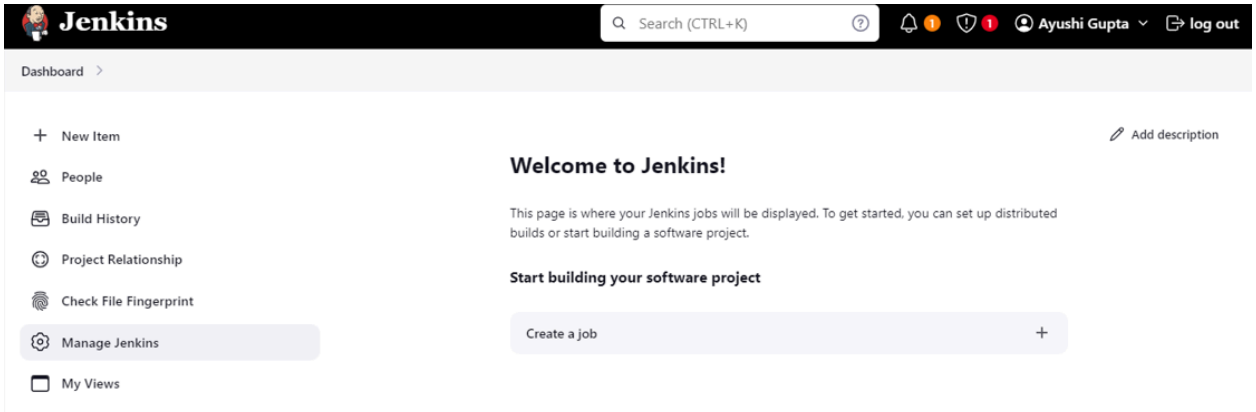


Fig. Access Web UI on port 8080

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

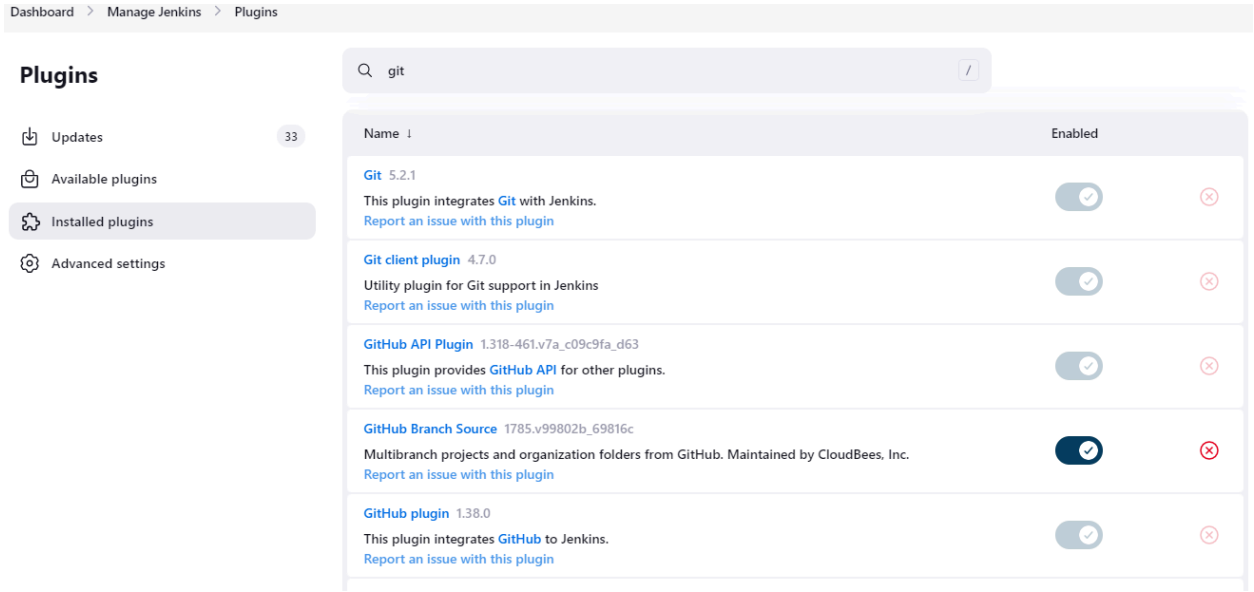


Fig: Integrate git with jenkins

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

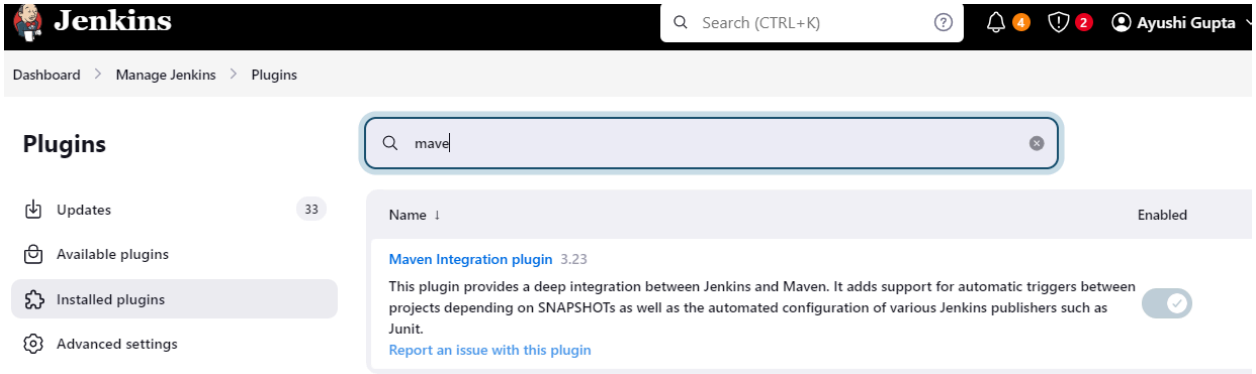
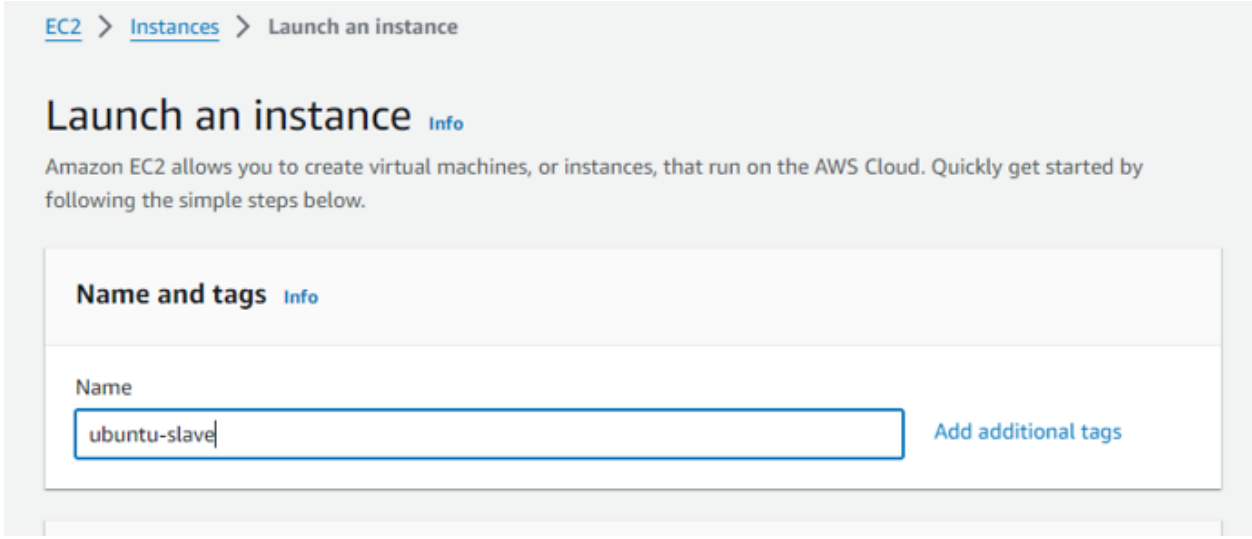
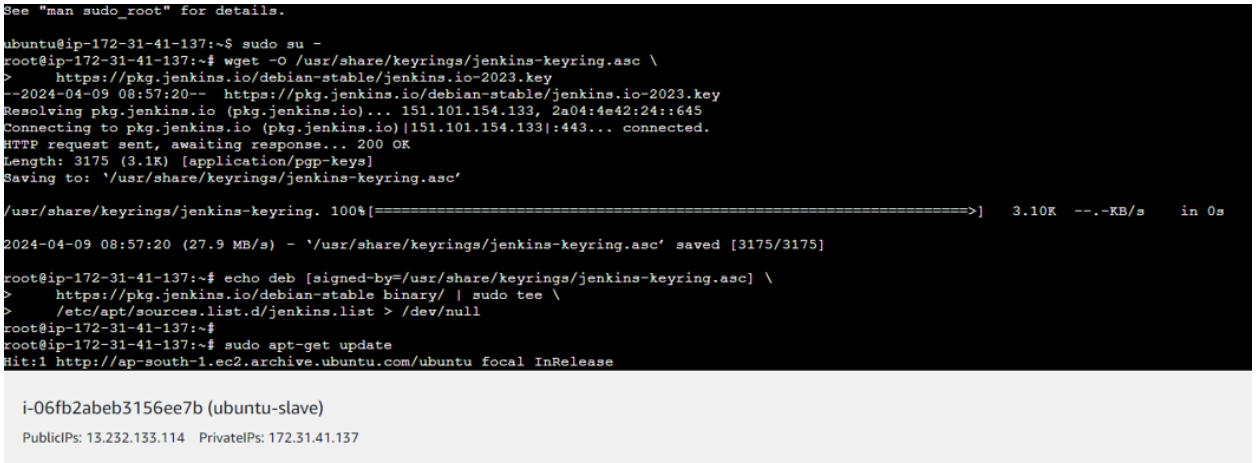


Fig: Integrate maven with jenkins



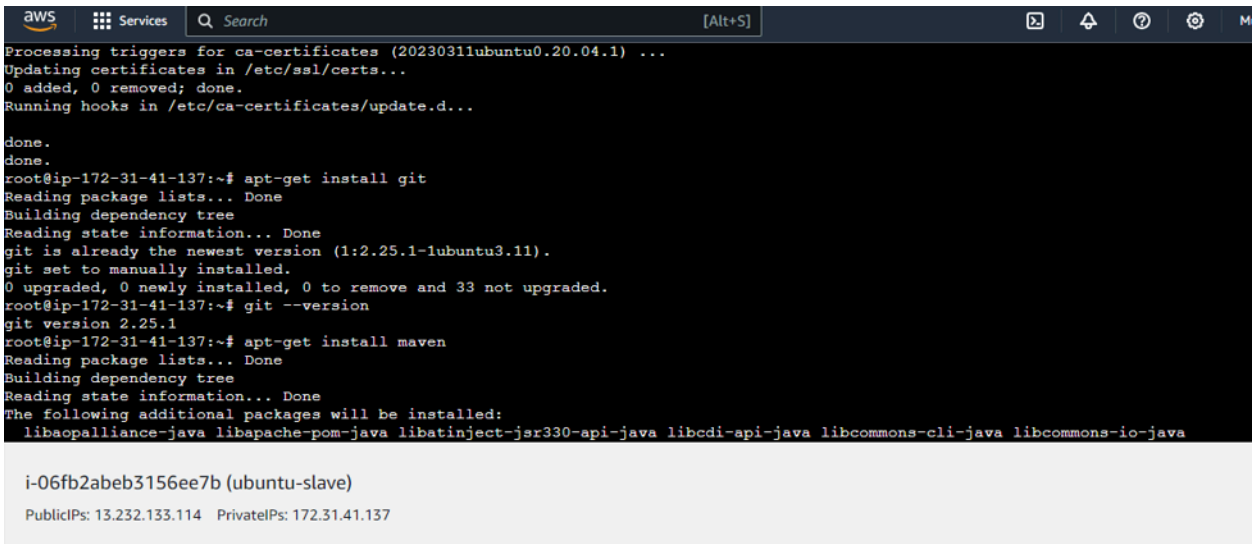
Launch a EC2 instance to build code with maven



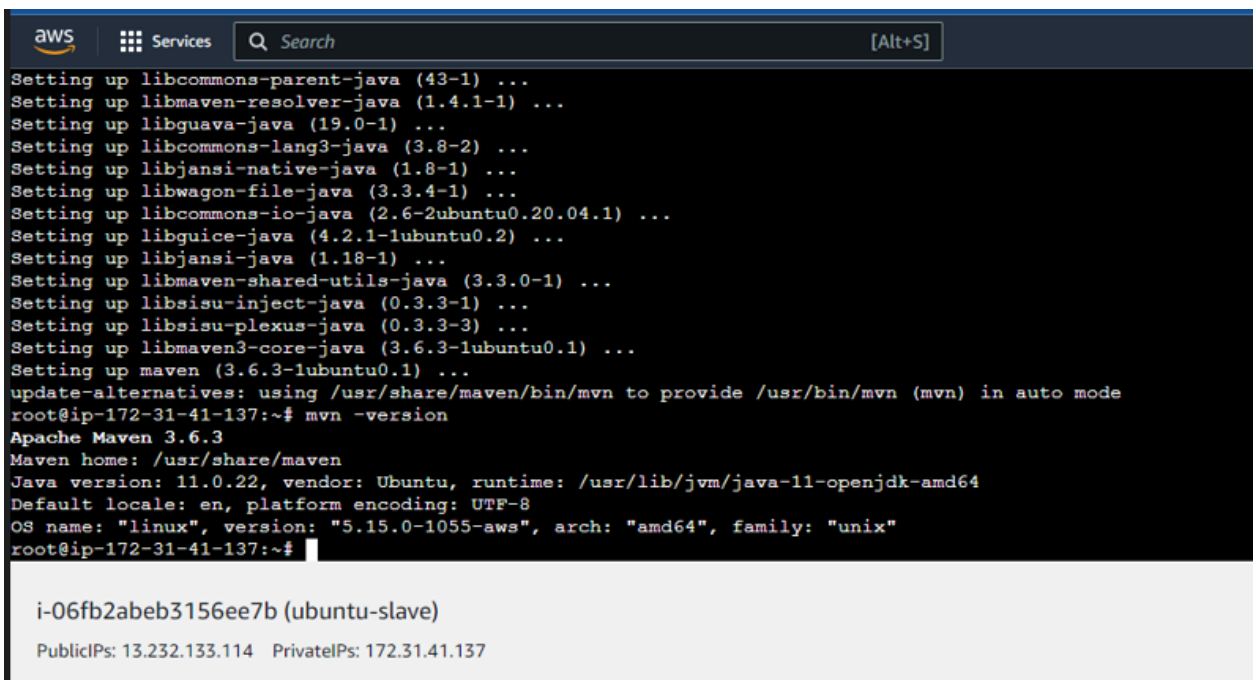
Making the Instance a Jenkins slave

```
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
```

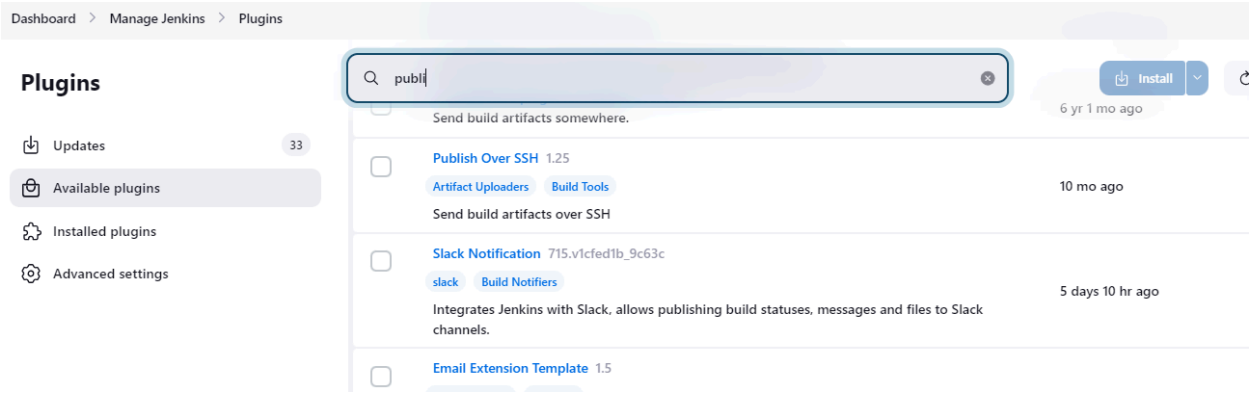
Commands to make Jenkins slave.



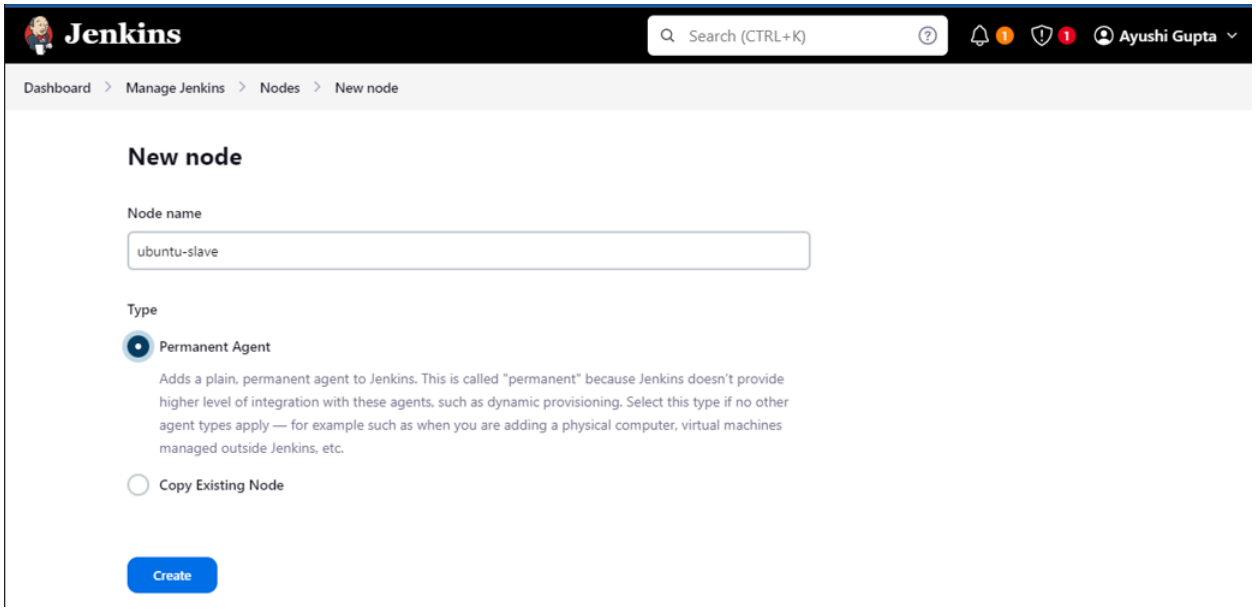
Installing maven



Mavan Installed

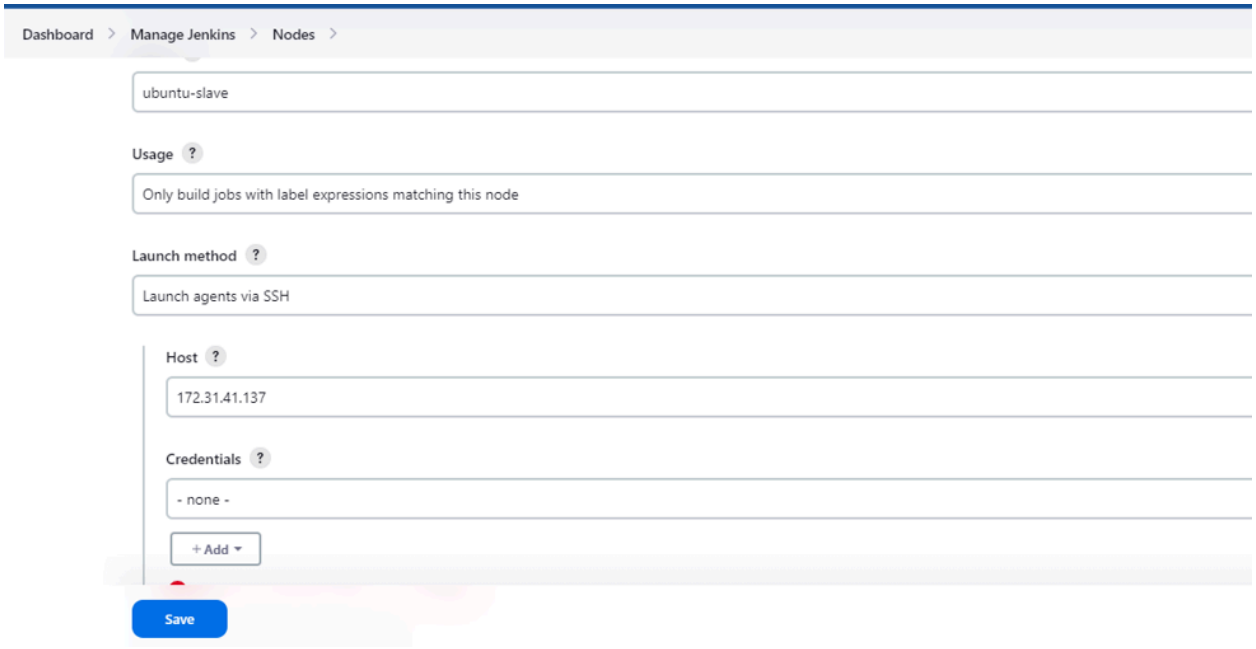


Install “Publish Over SSH “plugin

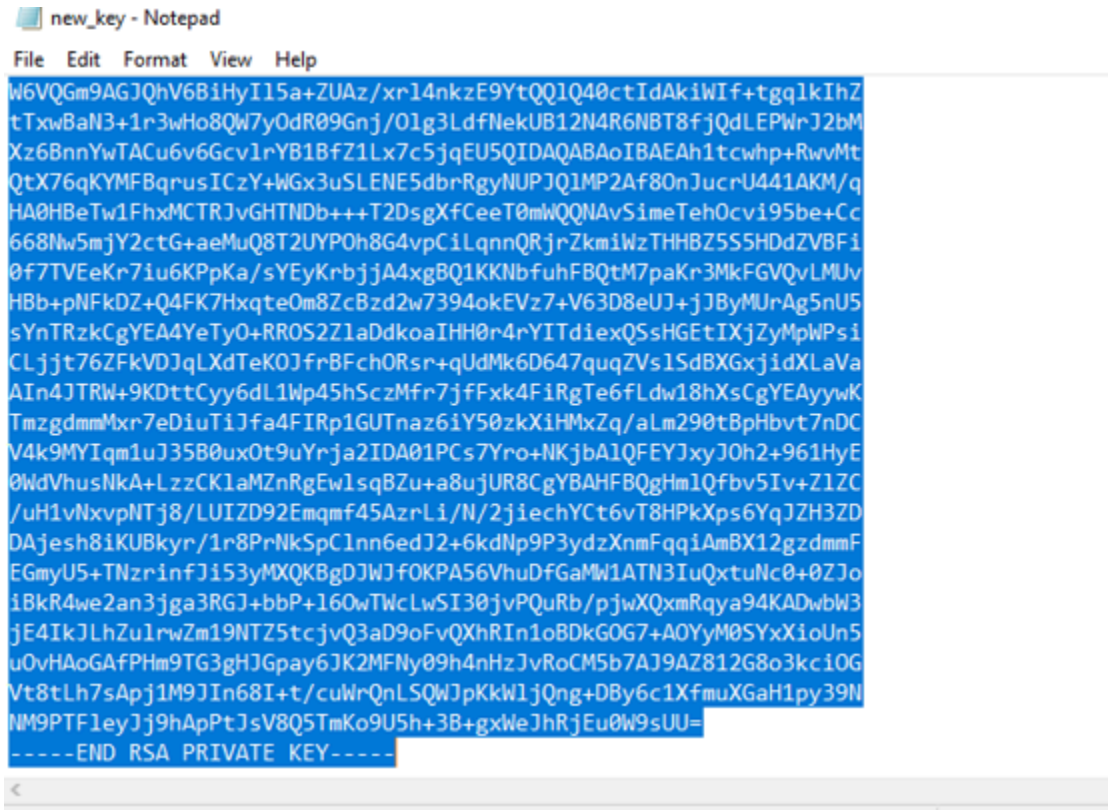


Make a ubuntu-slave a permanent node on jenkins web UI

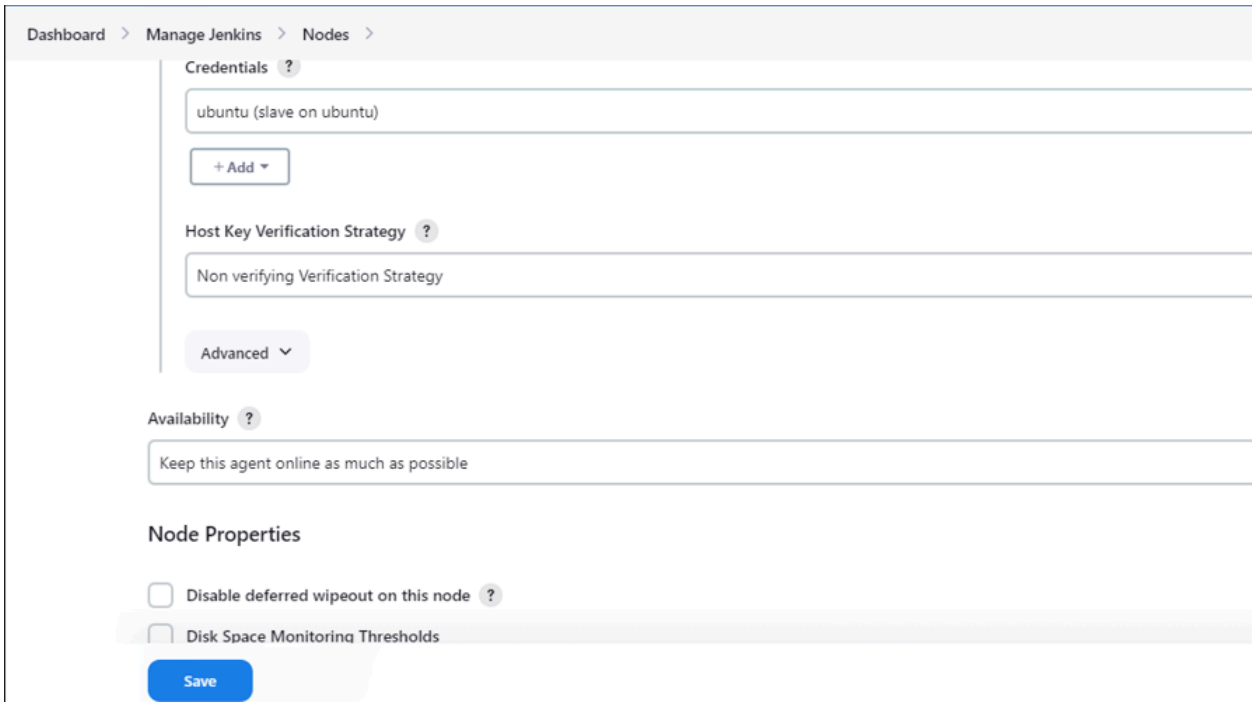
Configuration of a agent

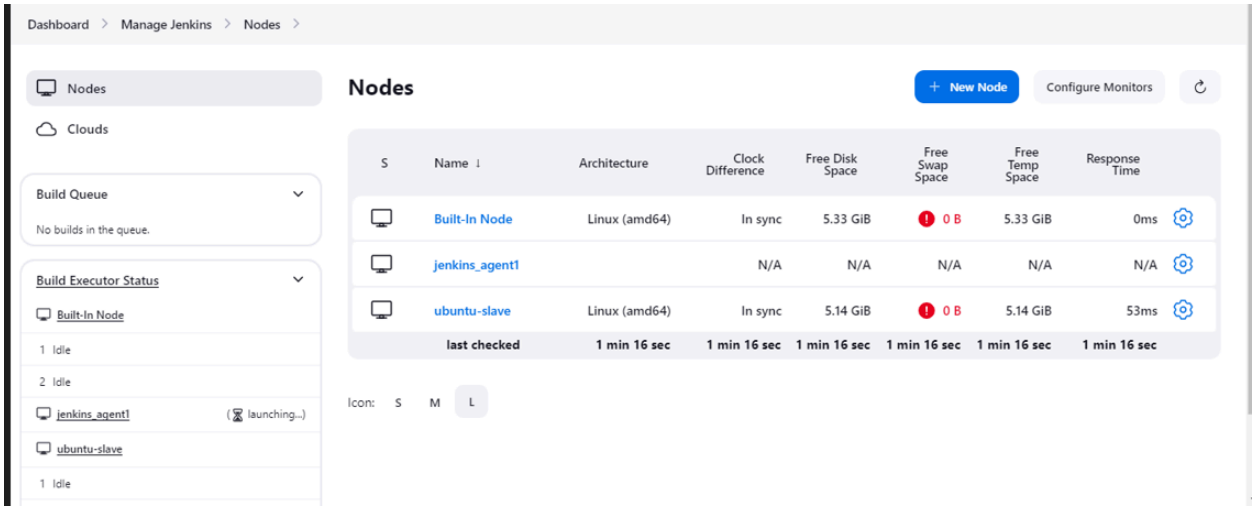


Give Credentials in the node to take SSH Session of ubuntu slave

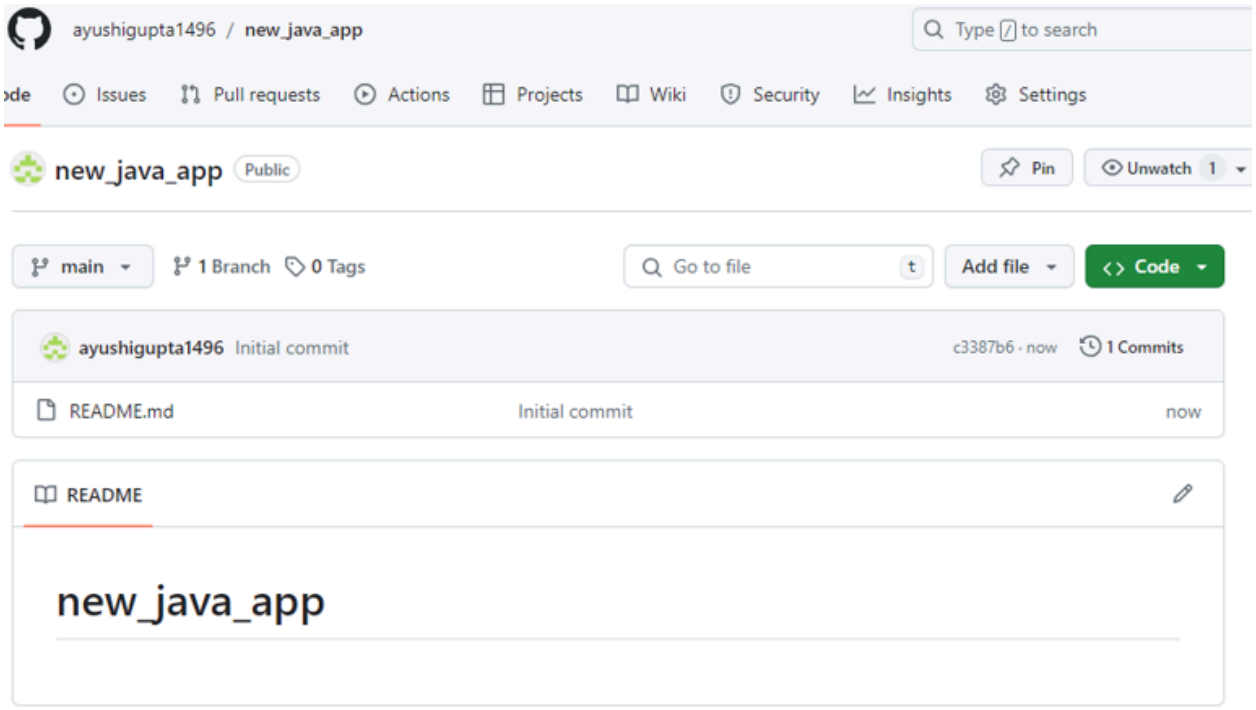


SSH Private key based authentication





Now i created Git repo to push my code on github



By : Ayushi Gupta Email : ayushigupta1496@gmail.com

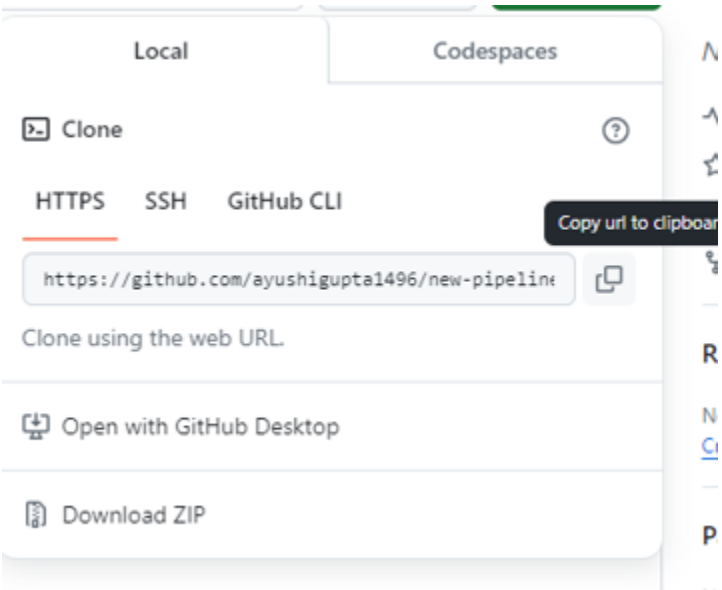
Now i created post commit hooks to push the code automatically when code committed

```
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-P6NHTNJ MINGW64 /d/git_new
$ ls
new_java_app/

DELL@DESKTOP-P6NHTNJ MINGW64 /d/git_new
$ ls
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
README.md

DELL@DESKTOP-P6NHTNJ MINGW64 /d/git_new/new_java_app (main)
$ ls
README.md

DELL@DESKTOP-P6NHTNJ MINGW64 /d/git_new/new_java_app (main)
$ ls -a
./  ../  .git/  README.md

DELL@DESKTOP-P6NHTNJ MINGW64 /d/git_new/new_java_app (main)
$ cd .git/

DELL@DESKTOP-P6NHTNJ MINGW64 /d/git_new/new_java_app/.git (GIT_DIR!)
$ ls
HEAD  config  description  hooks/  index  info/  logs/  objects/  packed-refs  refs/

DELL@DESKTOP-P6NHTNJ MINGW64 /d/git_new/new_java_app/.git (GIT_DIR!)
$ cd h
HEAD   hooks/

DELL@DESKTOP-P6NHTNJ MINGW64 /d/git_new/new_java_app/.git (GIT_DIR!)
$ cd hooks/

DELL@DESKTOP-P6NHTNJ 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-P6NHTNJ MINGW64 /d/git_new/new_java_app/.git/hooks (GIT_DIR!)
$ vim
```

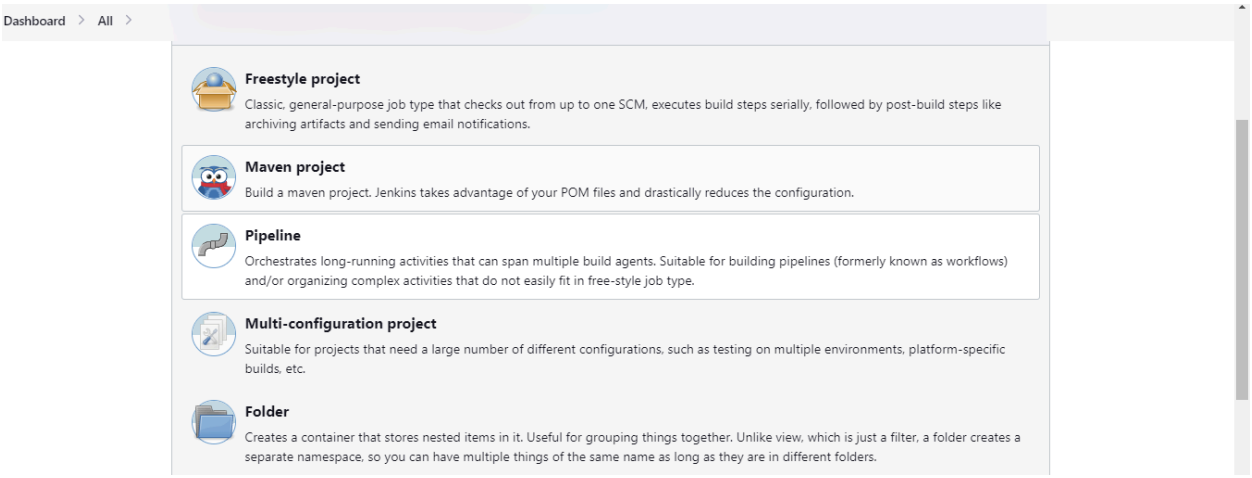
```
DELL@DESKTOP-P6NHTNJ MINGW64 /d/git_new/new_java_app/.git (GIT_DIR!)
$ cd h
HEAD   hooks/

DELL@DESKTOP-P6NHTNJ MINGW64 /d/git_new/new_java_app/.git (GIT_DIR!)
$ cd hooks/

DELL@DESKTOP-P6NHTNJ MINGW64 /d/git_new/new_java_app/.git/hooks (GIT_DIR!)
$ ls
applypatch-msg.sample*  fsmonitor-watchman.sample*  pre-applypatch.sample*  pre-merge-co
commit-msg.sample*      post-update.sample*         pre-commit.sample*      pre-push.sam

DELL@DESKTOP-P6NHTNJ MINGW64 /d/git_new/new_java_app/.git/hooks (GIT_DIR!)
$ vim post-commit
```

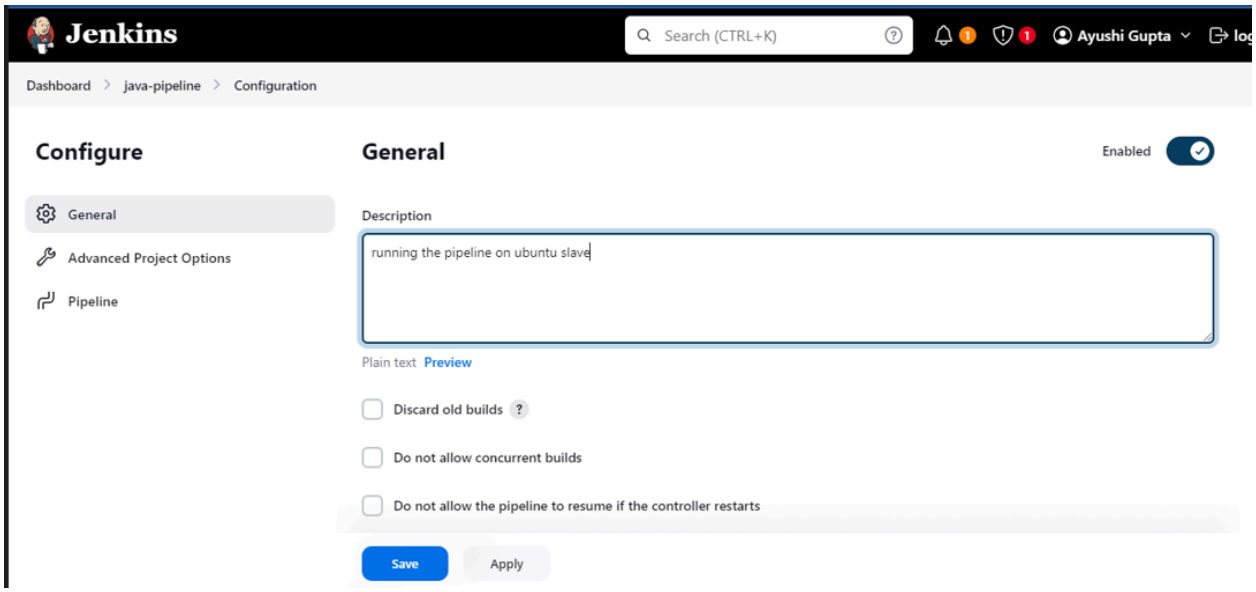
```
MINGW64:/d/git_new/new_java_app/.git/hooks
#!/bin/bash
git push -u origin main
~
```

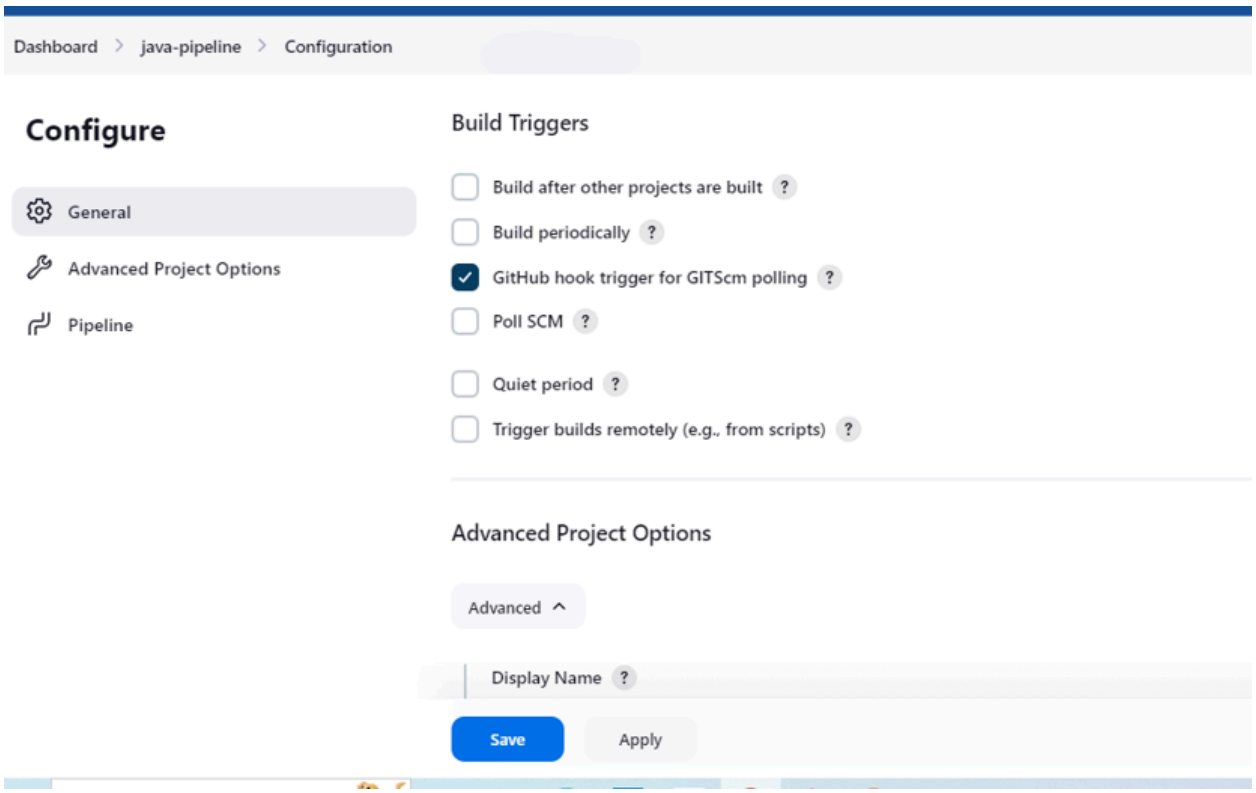
Clone the repo on local system

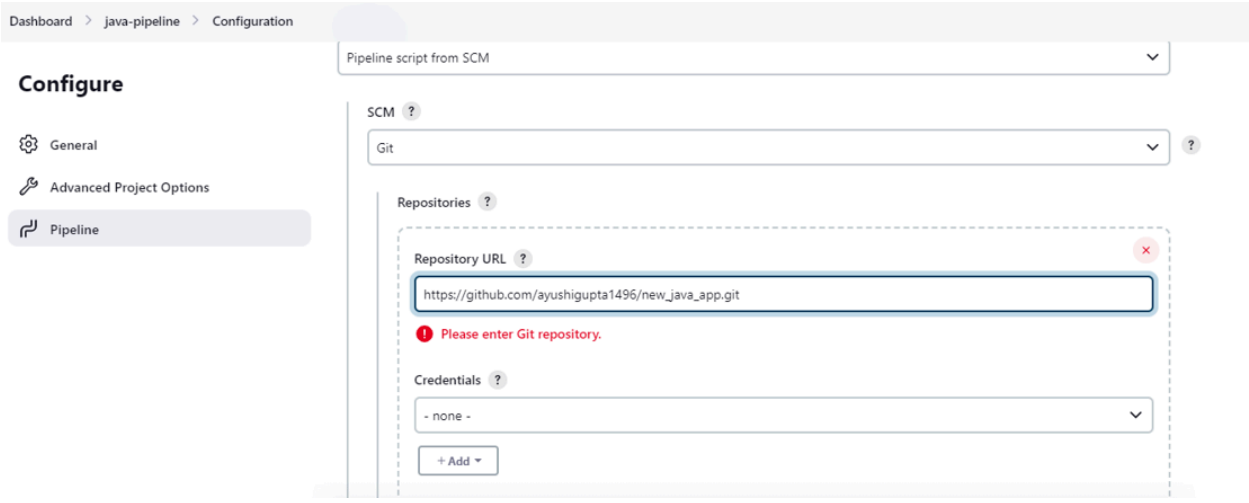


Now create jenkins pipeline by choosing pipeline project on jenkins Web-UI




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


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



Give the github repository URL

To Trigger the pipeline on Git push generate Webhooks in github

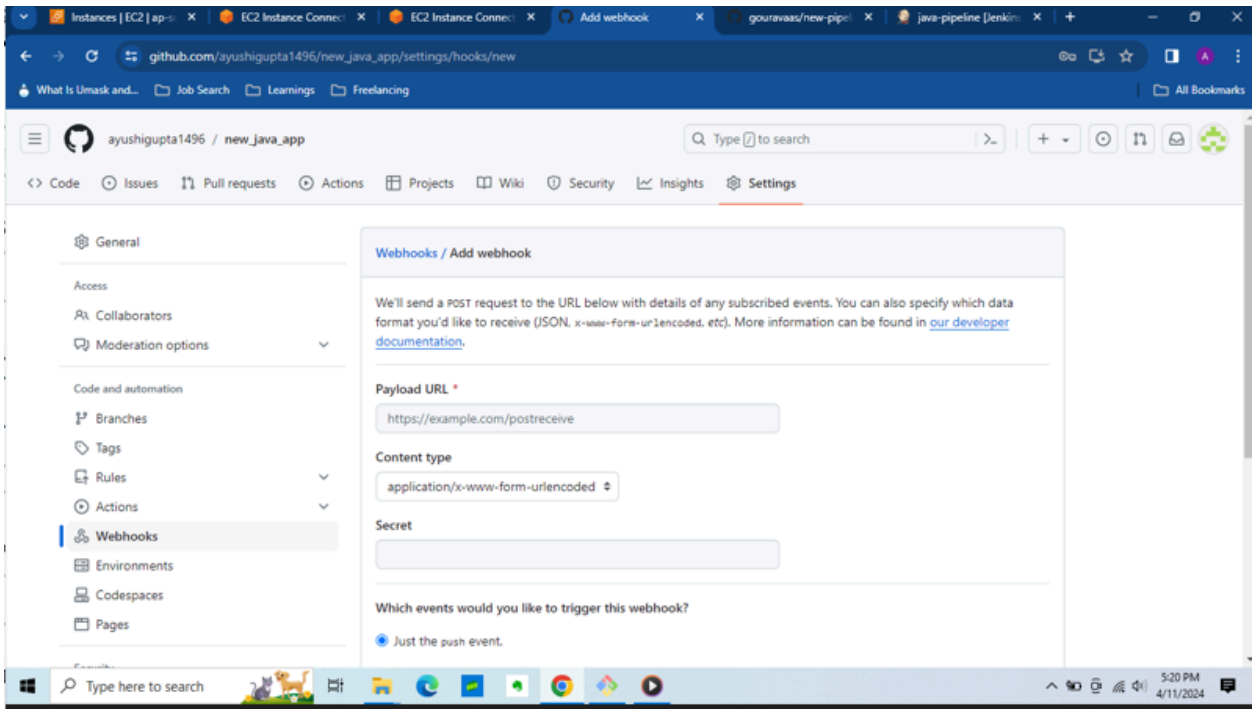


Fig:Github Webhooks

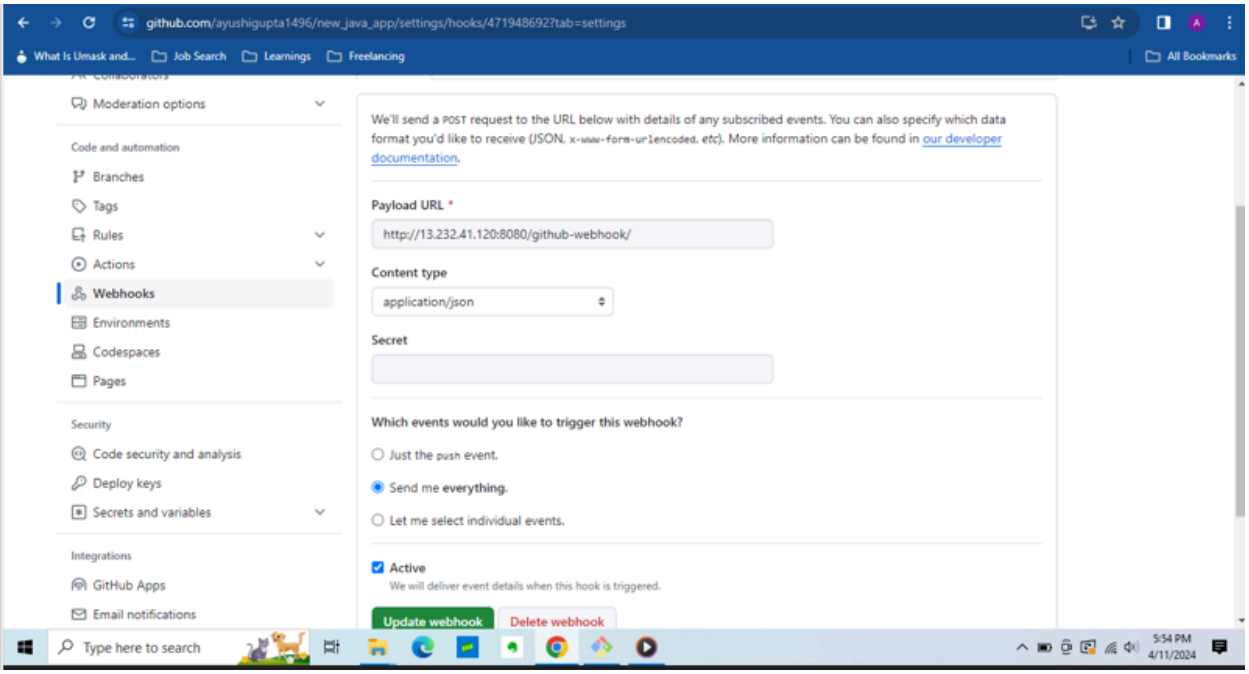
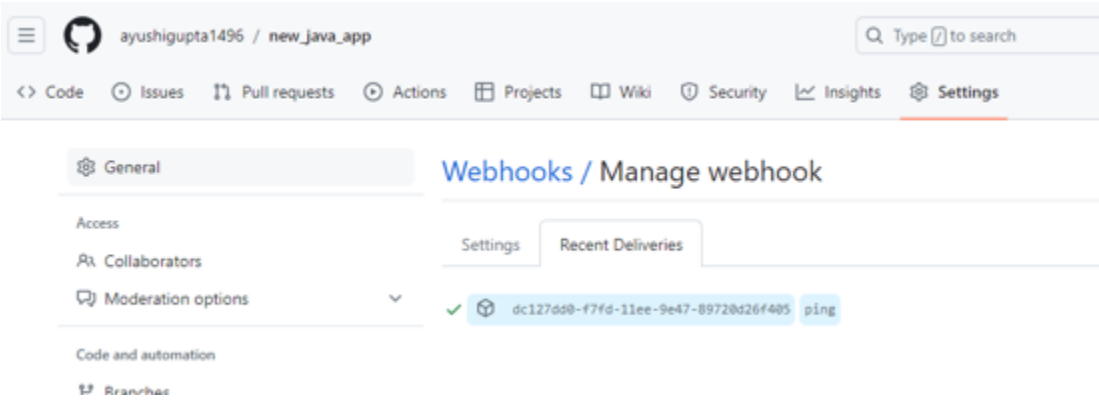


Fig: Adding Github Webhooks



Install Docker Plugin in Jenkins & Integrate Jenkins with Docker

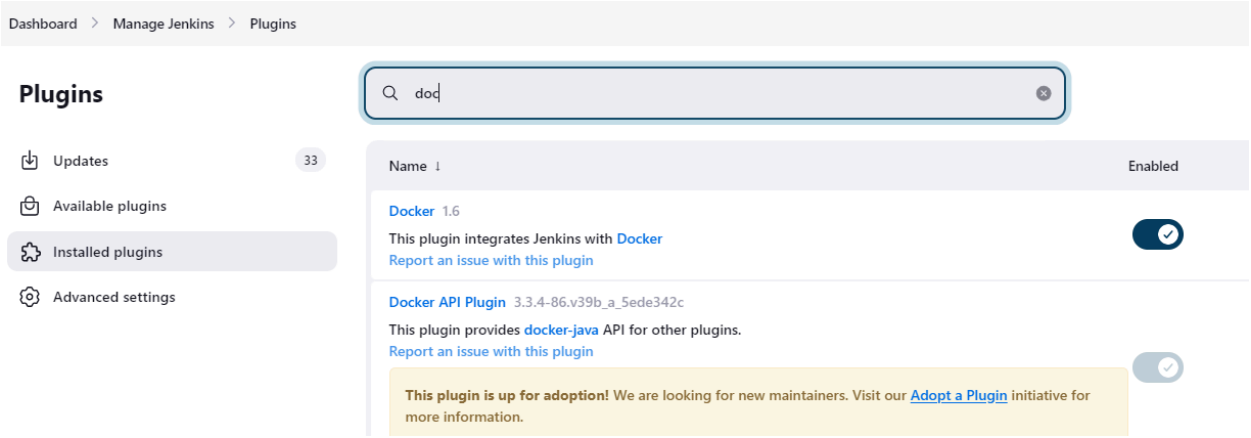
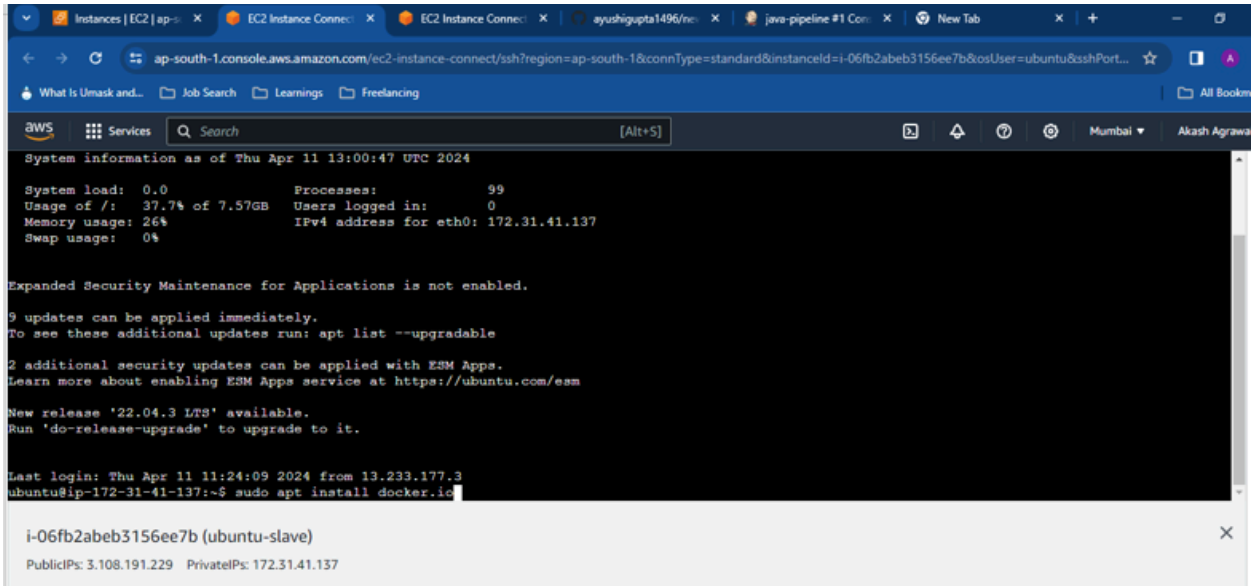
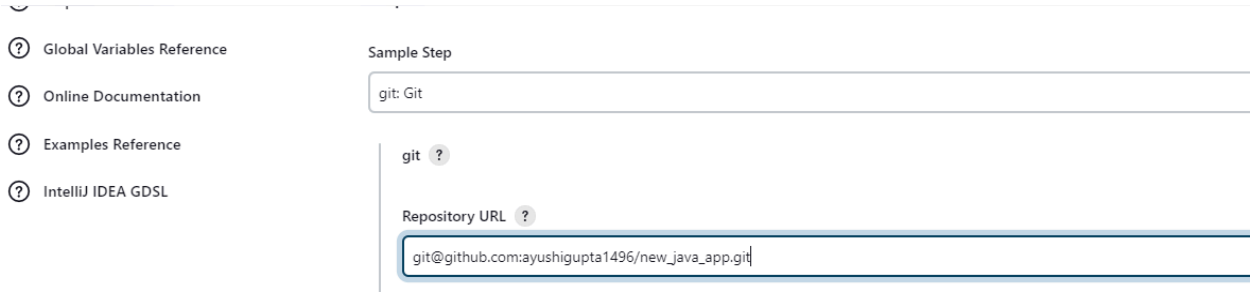
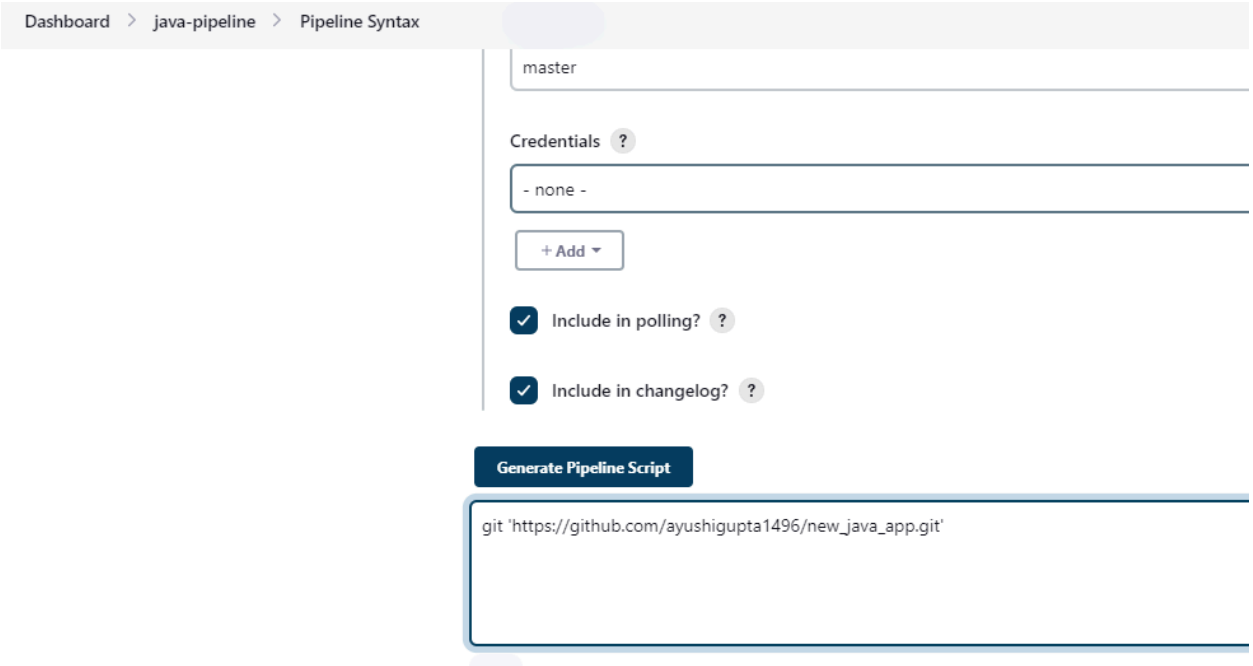


Fig:Integrate Docker with jenkins



Now we start creating the pipeline script






```
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'
                        }
                }
```

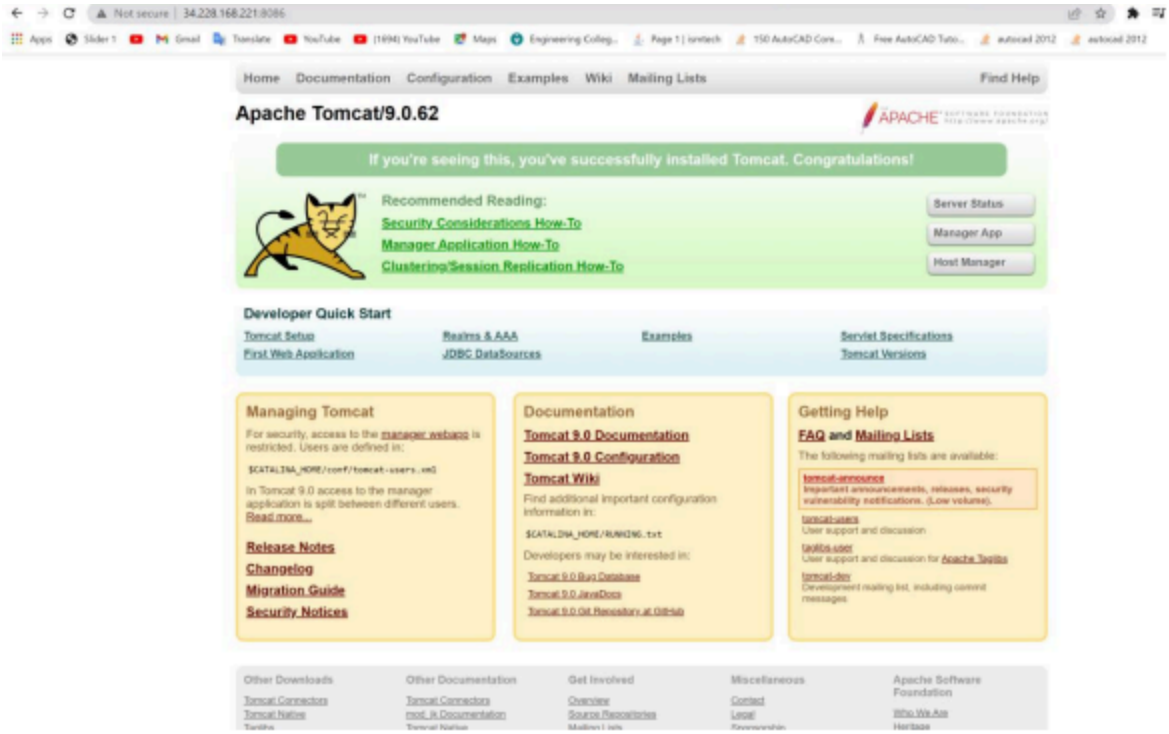


Fig. Tomcat Started

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

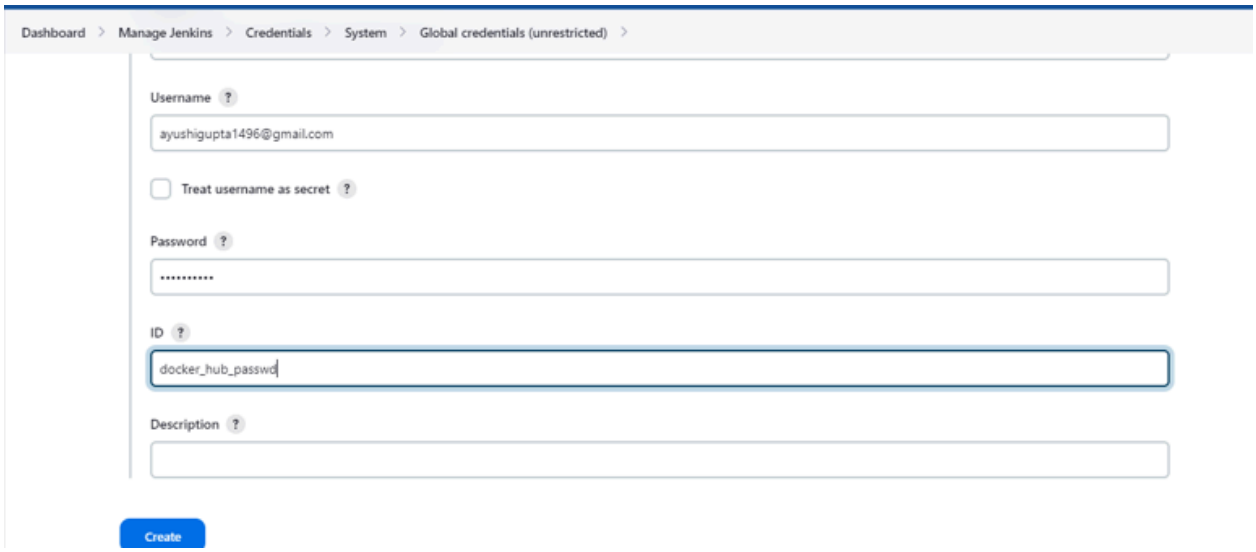
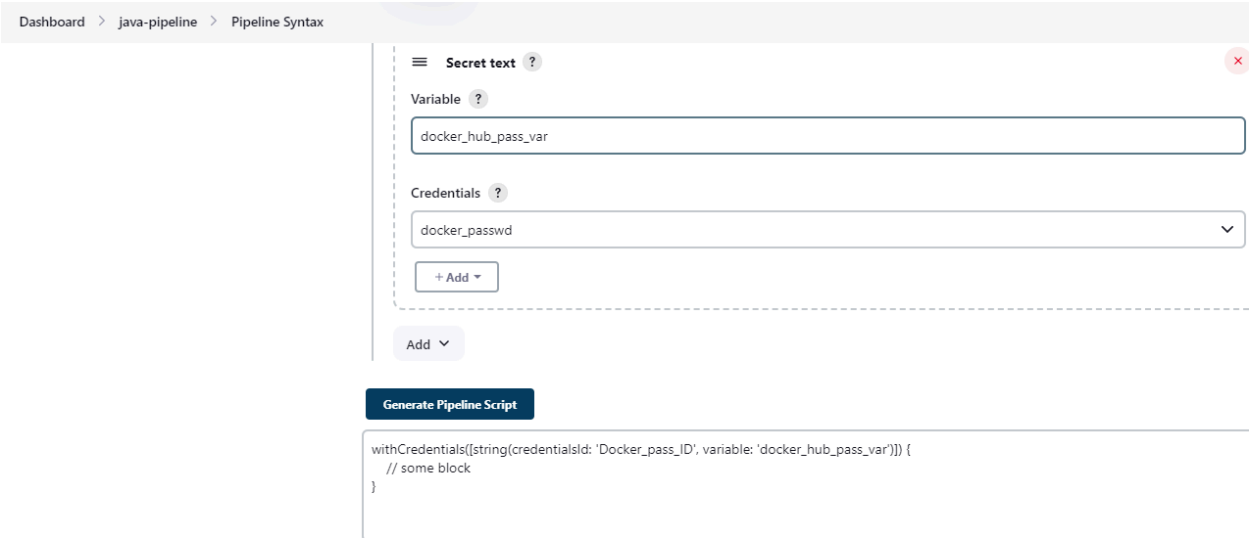


Fig:Dockerhub credentials

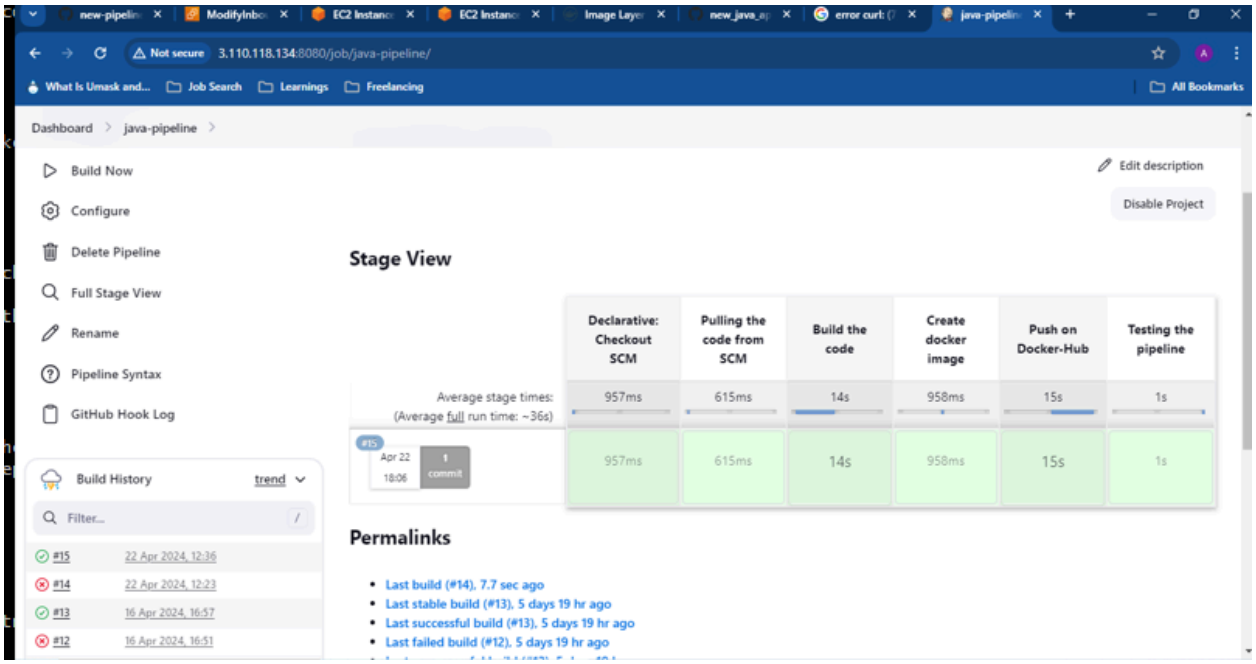


Stage 4 : push the image on dockerhub

```
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'
                                }
                        }
```



Stage 6 : QAT Testing

```
                }
                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)"'
                                }
                        }
                }
        }
```

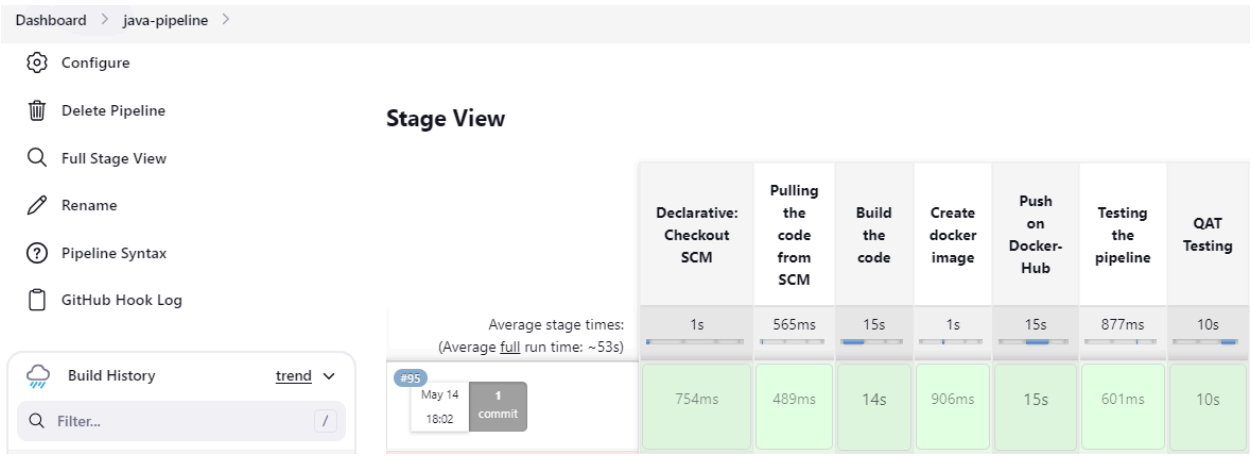


fig:OAT Testing completed

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.

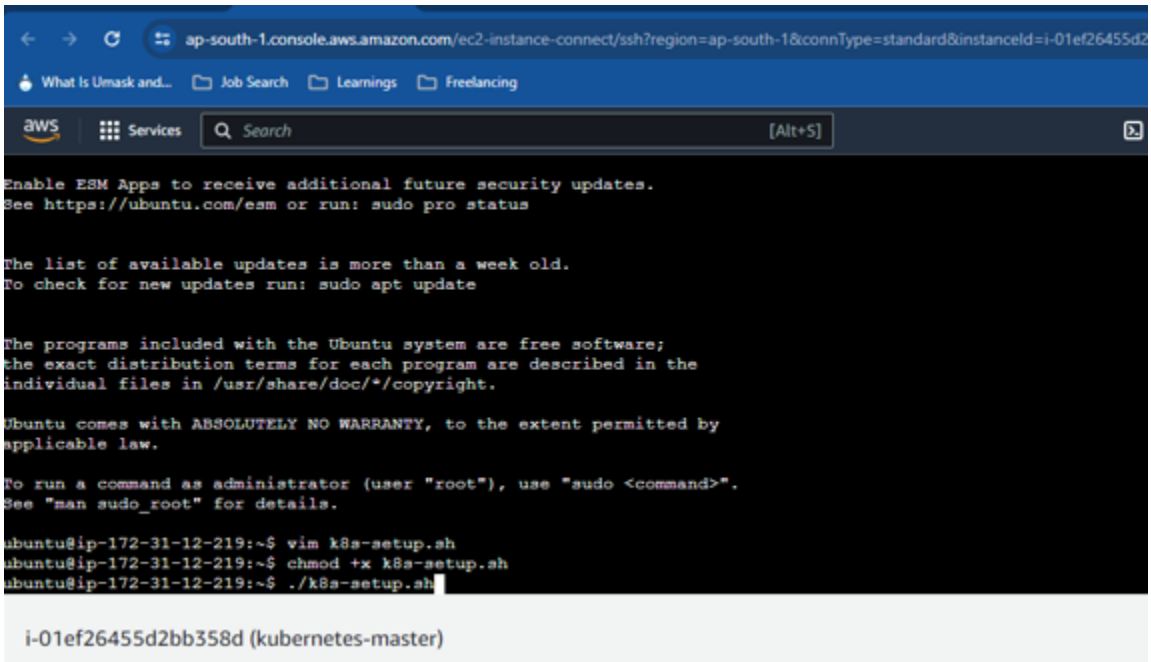
For deployment we launch two other server i.e kubernetes master & kubernetes worker

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**



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

SET-UP COMMANDS FOR K8S

kubernetes
Learning k8s confriguration
---
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

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

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

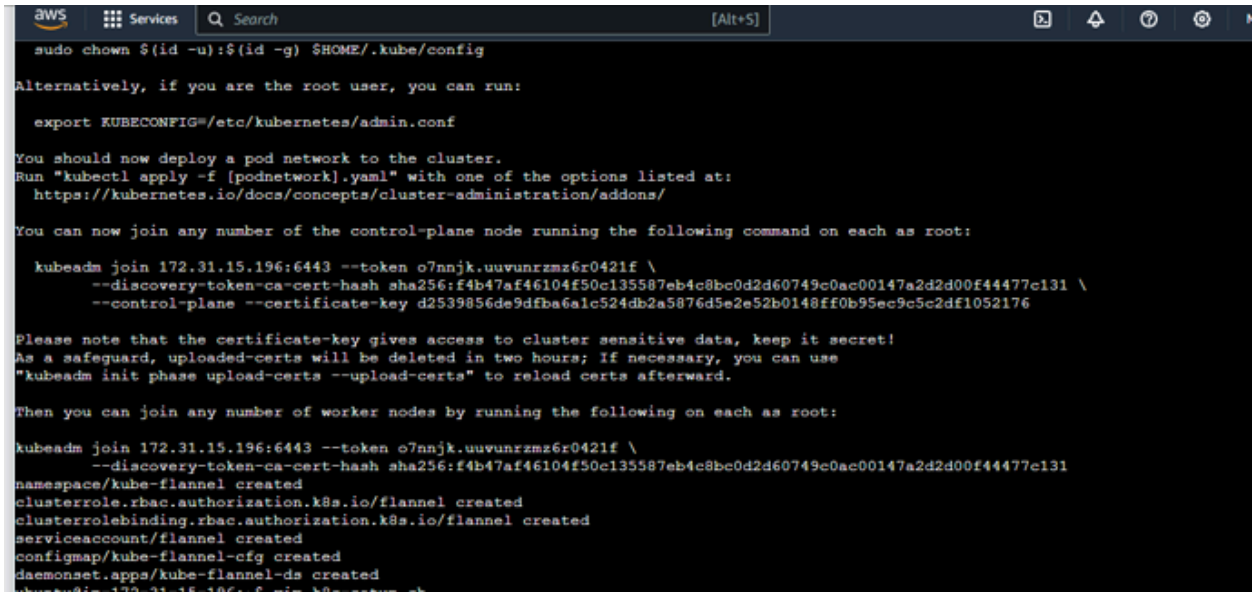
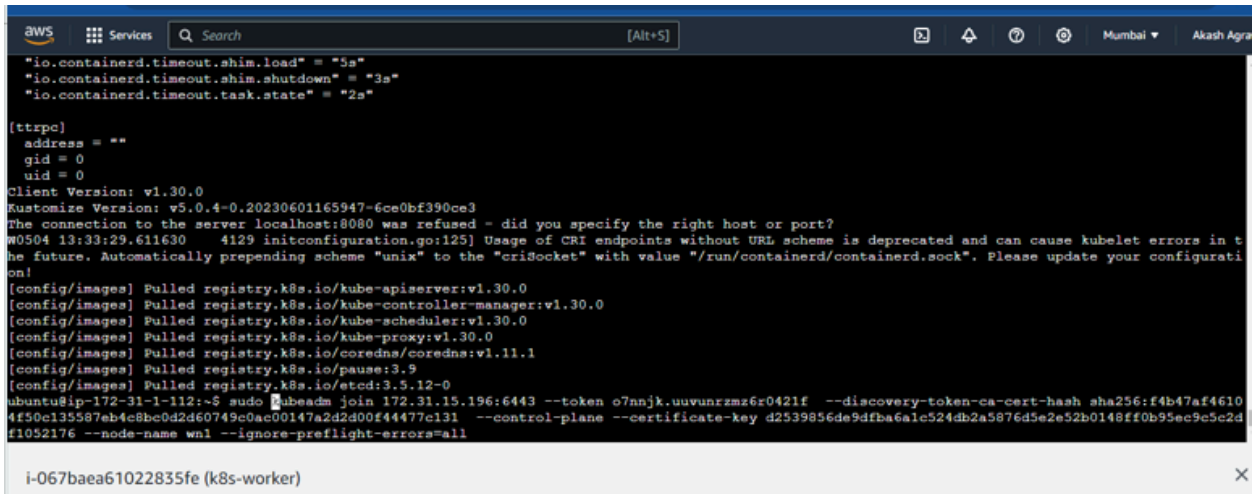


Fig: With the help of kubeadm join token we can join worker nodes

Now we will launch an ec2 instance as kubernetes worker node



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

kubectl version

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

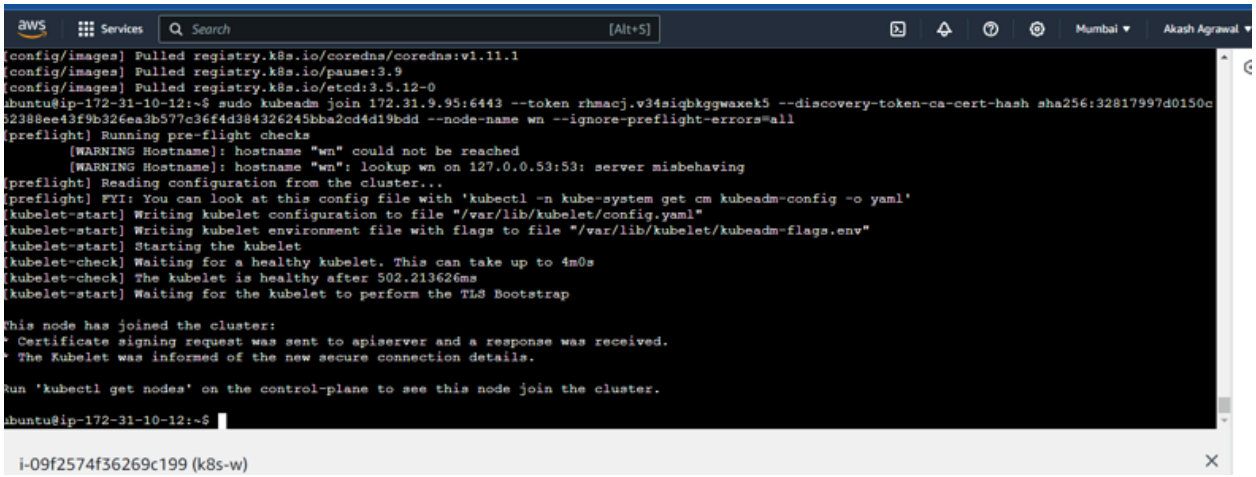
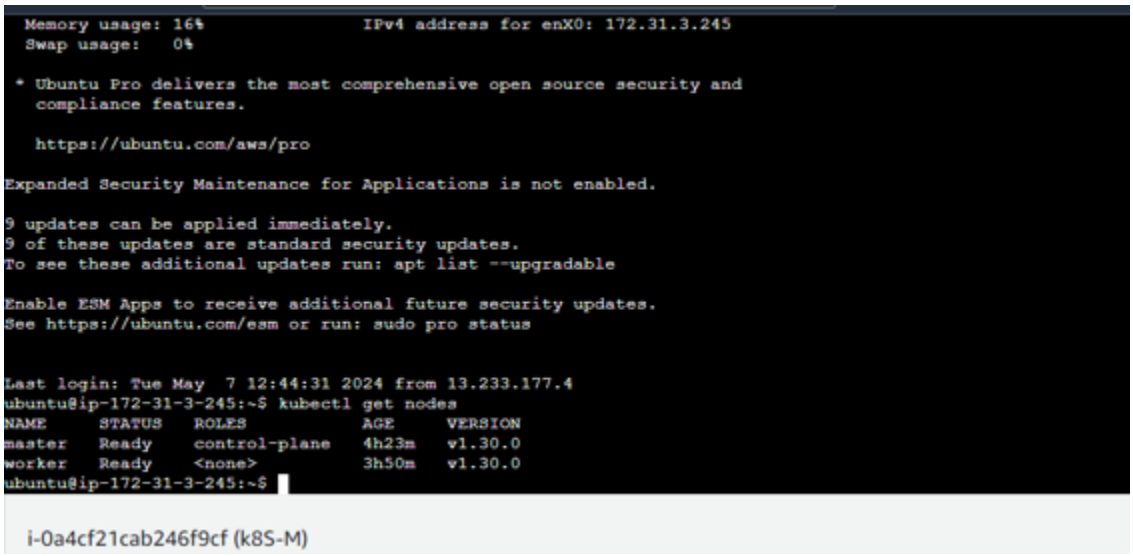


Fig: join token run the node as worker node



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
                        }
```

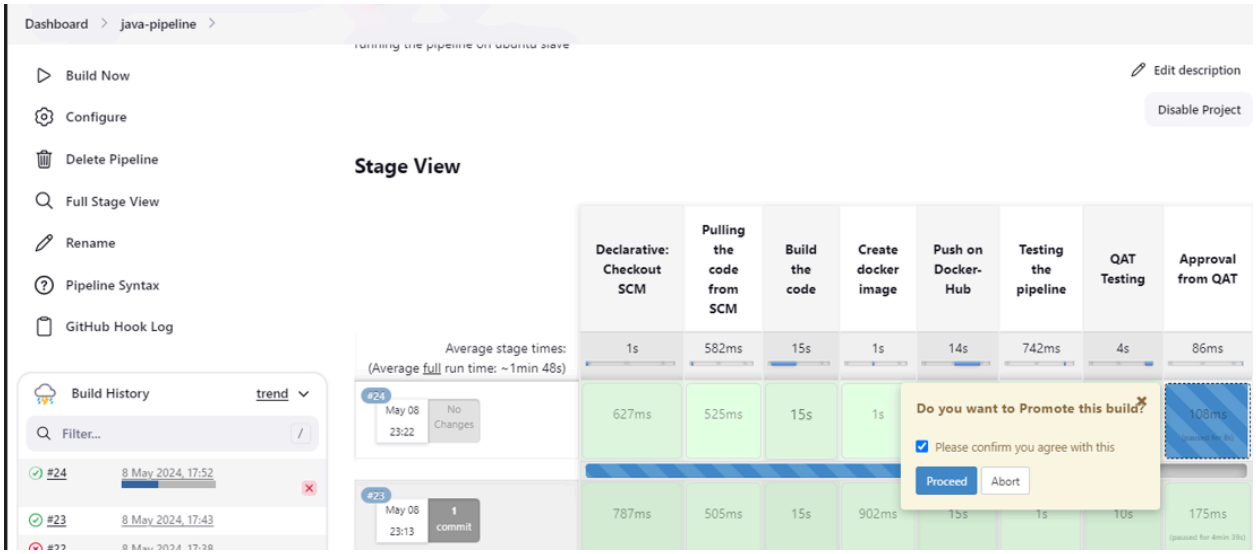
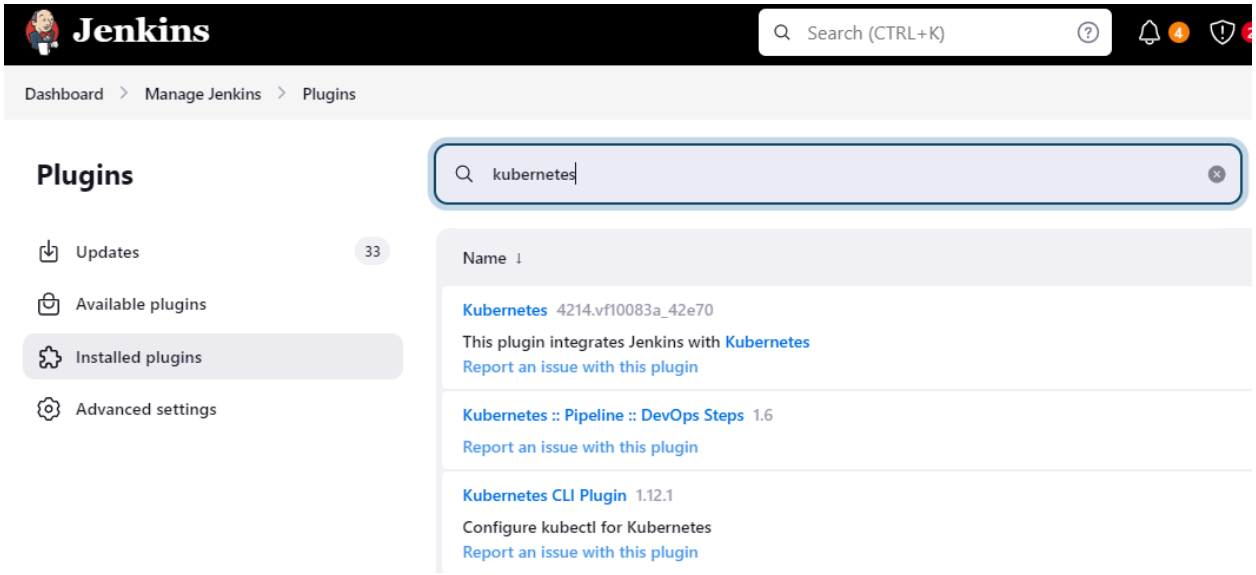


Fig: pipeline build successfully

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

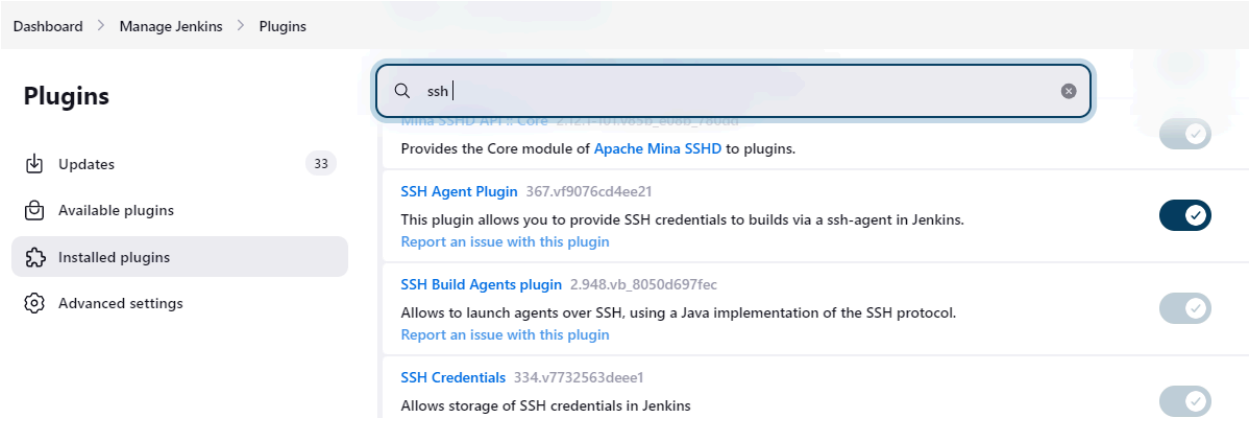


Fig: ssh- agent plugin

Step-4 Generate ssh credentials on jenkins web-UI

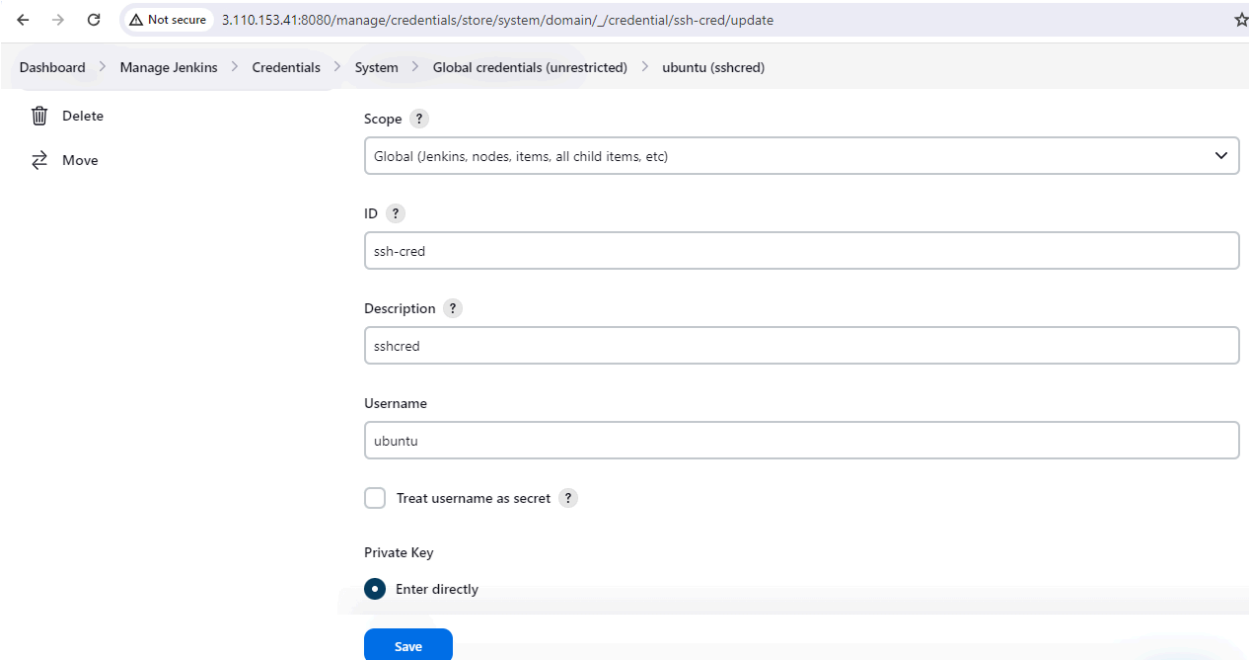


Fig: ssh credentials

```
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

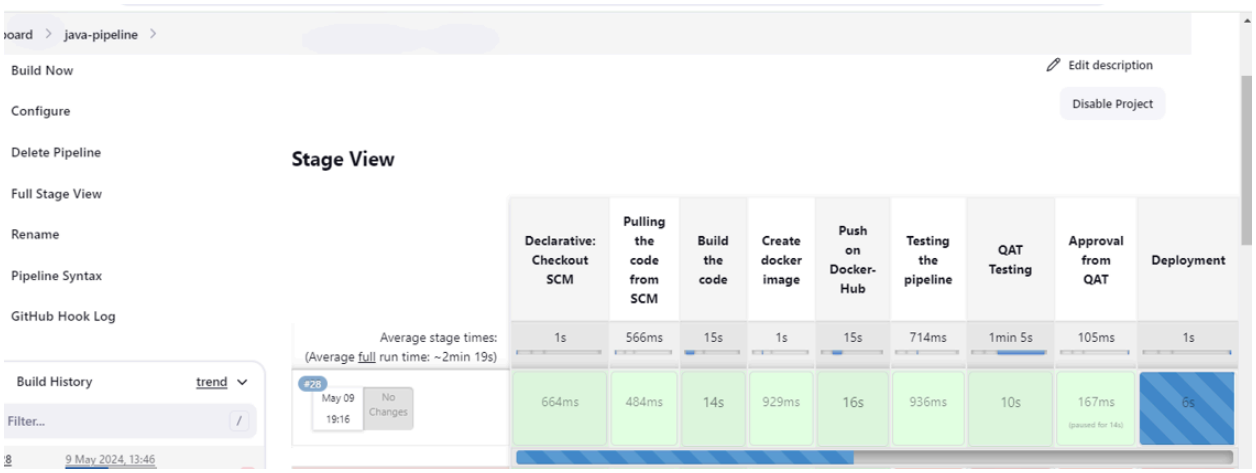


Fig: pipeline script is running

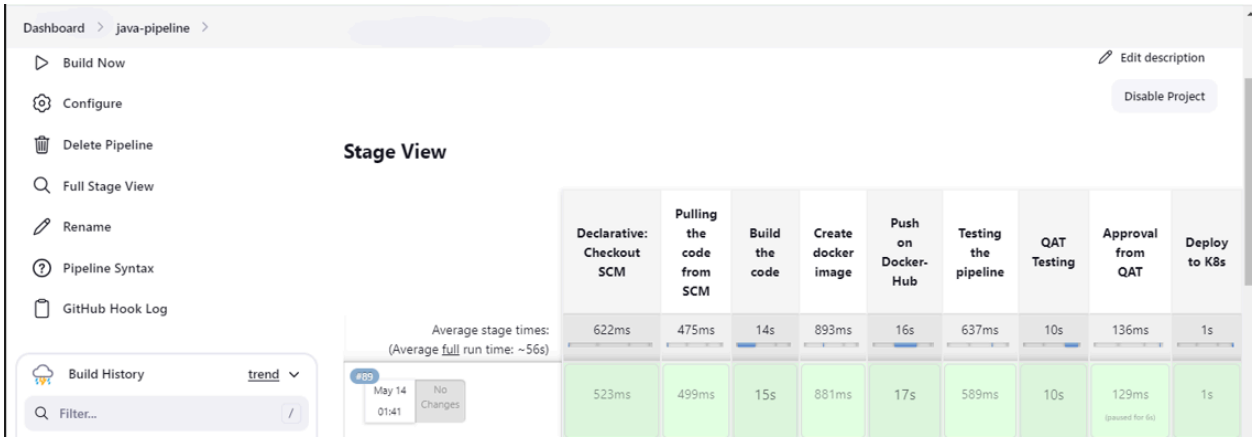


Fig: Deployment is completed Successfully