

# DevOps Jenkins Assignment

---

**Submitted to:**

Dr. Uma S

**Submitted by:**

Team 5

Akshaya Reddy 18BCS017

Bhavya Tripathi 18BCS019

Harshitha M 18BCS032

Meghana N 18BCS053

S Namratha 18BCS083

Vinita Yadav 18BCS109

Deepanshu Sachdeva 18BCS114



INDIAN INSTITUTE OF  
INFORMATION  
TECHNOLOGY

# TASK 1

## Setting up CI/CD Jenkins pipeline for kubernetes

### Tools and Technologies used :

- Github
- Docker and Docker hub
- Jenkins
- Kubernetes Cluster

### System Requirements :

3 AWS Ec2 Ubuntu instances of size t2.medium and 15 GB of volumes attached.

### Step - 1 : Setting Up kubernetes Cluster

We have used the kubeadm tool to set up the kubernetes cluster.

Setting up a kubernetes cluster containing 2 nodes master and worker.

#### 1.1 : Commands to run on both nodes

To update the system packages

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

##### 1.1.1 : Installing docker

```
$ sudo apt install apt-transport-https ca-certificates curl  
software-properties-common  
  
$ 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 focal stable"  
  
$ apt-cache policy docker-ce  
  
$ sudo apt-get install docker-ce docker-ce-cli containerd.io -y
```

To Confirm docker installation

```
$ sudo docker version
```

To add docker daemon



```
$ cat <<EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opt": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
EOF
```

Enabling and starting docker

```
$ sudo systemctl enable docker
$ sudo systemctl daemon-reload
$ sudo systemctl restart docker
```

### 1.1.2 : Installing kubernetes

Install Kubeadm,Kubelet and Kubectl

```
$ sudo apt-get update
$ sudo apt-get install -y apt-transport-https ca-certificates
curl
$ sudo curl -fsSLo
/usr/share/keyrings/kubernetes-archive-keyring.gpg
https://packages.cloud.google.com/apt/doc/apt-key.gpg
$ echo "deb
[signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg]
https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee
/etc/apt/sources.list.d/kubernetes.list
$ sudo apt-get update -y
$ sudo apt-get install -y kubelet kubeadm kubectl
```

To confirm kubectl installation

```
$ sudo kubectl version
```

To freeze versions of Kubeadm,Kubelet and Kubectl

```
$ sudo apt-mark hold kubelet kubeadm kubectl
```

## 1.2 : Commands to run only on master node

Initializing the master node using kubeadm

```
$ sudo kubeadm init --pod-network-cidr 10.0.0.0/16
```

Output of this command would be a key , through which worker nodes can join the kubernetes cluster.

Copy the token and save it somewhere.

```
[kubelet] Creating a ConfigMap "kubelet-config-1.22" in namespace kube-system with the configuration for the kubelets in the cluster
[upload-certs] Skipping phase. Please see --upload-certs
[mark-control-plane] Marking the node ip-172-31-27-210 as control-plane by adding the labels: [node-role.kubernetes.io/master(deprecated) node-role.kubernetes.io/control-plane node.kubernetes.io/exclude-from-external-load-balancers]
[mark-control-plane] Marking the node ip-172-31-27-210 as control-plane by adding the taints [node-role.kubernetes.io/master:NoSchedule]
[bootstrap-token] Using token: ws6v8y_fd9089anv7clc8nw
[bootstrap-token] configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles
[bootstrap-token] configured RBAC rules to allow Node Bootstrap tokens to get nodes
[bootstrap-token] configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate credential
[bootstrap-token] configured RBAC rules to allow the csraprover controller automatically approve CSRs from a Node Bootstrap Token
[bootstrap-token] configured RBAC rules to allow certificate rotation for all node client certificates in the cluster
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate and key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
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/
Then you can join any number of worker nodes by running the following on each as root:
Kubeadm join 172.31.27.210:6443 --token ws6v8y_fd9089anv7clc8nw \
--discovery-token-ca-cert-hash sha256:f6c8fdc0710b9296dd04ede995c7a13ea6a76c1bfcaac19ff5a10e6e625890fe
```

```
$ mkdir -p $HOME/.kube
$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
$ kubeadm version
$ kubectl apply -f
"https://cloud.weave.works/k8s/net?k8s-version=$ (kubectl version | base64 | tr -d '\n')"
$ kubectl get nodes
```

Gives the nodes and status of nodes present in the cluster

```
ip-172-31-27-210  NOTReady  control-plane,master  4m48s  v1.22.3
ubuntu@ip-172-31-27-210:~$ kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"
serviceaccount/weave-net created
clusterrole.rbac.authorization.k8s.io/weave-net created
clusterrolebinding.rbac.authorization.k8s.io/weave-net created
role.rbac.authorization.k8s.io/weave-net created
rolebinding.rbac.authorization.k8s.io/weave-net created
daemonset.apps/weave-net created
ubuntu@ip-172-31-27-210:~$ kubectl get nodes
NAME      STATUS   ROLES      AGE   VERSION
ip-172-31-27-210  Ready    control-plane,master  18m   v1.22.3
ubuntu@ip-172-31-27-210:~$
```

### 1.3 : Commands to run only on worker node

Using the token copied earlier

```
$ sudo kubeadm join 172.31.27.210:6443 --token
ws6v8y.fd9089anv7clc8nw --discovery-token-ca-cert-hash
sha256:f6c8fdc0710b9296dd04ede995c7a13ea6a76c1bfcaac19ff5a10e6e625890fe
```

Output of this command

```
The connection to the server localhost:8080 was refused - did you specify the right host or port?
ubuntu@ip-172-31-16-106:~$ sudo kubeadm join 172.31.27.210:6443 --token ws6v8y.fd9089anv7clc8nw --discovery-token-ca-cert-hash sha256:f6c8fdc0710b9296dd04ede995c7a13ea6a76c1bfcaac19ff5a10e6e625890fe
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -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-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

### 1.4 : Commands to run only on master node

```
$ kubectl get nodes
```

Now there would be 2 nodes one master and one newly joined worker node

Newly joined node's role name would be <none> to label it as worker ,

```
$ kubectl label node ip-172-31-16-106 node-role.kubernetes.io/worker=worker
```

```
The connection to the server localhost:8080 was refused - did you specify the right host or port?
ubuntu@ip-172-31-27-210:~$ kubectl get nodes
NAME      STATUS   ROLES      AGE   VERSION
ip-172-31-16-106  Ready    <none>     9m26s  v1.22.3
ip-172-31-27-210  Ready    control-plane,master  49m   v1.22.3
ubuntu@ip-172-31-27-210:~$ kubectl label node ip-172-31-16-106 node-role.kubernetes.io/worker=worker
node/ip-172-31-16-106 labeled
ubuntu@ip-172-31-27-210:~$ kubectl get nodes
NAME      STATUS   ROLES      AGE   VERSION
ip-172-31-16-106  Ready    worker     12m   v1.22.3
ip-172-31-27-210  Ready    control-plane,master  51m   v1.22.3
ubuntu@ip-172-31-27-210:~$
```

To check configurations of kubernetes cluster

```
$ cd .kube
```

```
$ cat config
```

Output :

```

Activities Terminal wed 16:13
ubuntu@ip-172-31-27-210:~/kube$ cat config
apiVersion: v1
kind: Config
clusters:
- cluster:
  certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCK1JSUMvakNDQWhZ0F3SUJB0lCQURBTkJna3FoazLHOxcwQkFRc0ZBREFHTVJNd0VRWURWUV
FERRxdwcmRXSwmKY19t1bcGRHvPnQJRYRFRfE1URxhNREf1TKrRM5h1hEVEf14TVRfd9FQfTfORGsw1TzvdZUfVRNQkVHQtfVPrQoBeE1LYTNWaVpYSpnVwf1sY30d0FtsxdEUV1LKS29a
Swh2y05BUUQCFBBDnRVRBRENDQFvQ2dnU2BTG9nCkErQkfFd1ErNTl1o2m3eVuUxdzeHgyN2JHJzK9BcKpwM1TdvTn1SQlpsQJMhCrnsU5tZvh1ue42aU3MMHF1LMEwQkVprSHASUV
RnvFo4Wkt3VJBjWkpraTVZFF02Ddqf1J0dZvTNH1vb1J1RVhMc0fUfMyccCttcGJTOHVLMT1Zzwp1WfRN3V2UmVHTVYz0udtUDhzTj30Nz2bc2hdWz50NDTmpubedvyUgzbfpTYRf
N3ZtQ1Qb1ubhURxMDz1LBGMktZ01AzL0VgM0RQZGDSk9wsHVkn21aeGRKU0k2dnFnMsFzTFHbnRjxk1Pbm3PMVdYSXN1RFZWMf02Zs8kbTfStmaiddJk1NrcCd0fTGRSuTdhuFfJ0eH
TzQTA4UhNejz0MnhZbw3p30GNVHZYTHJURFcwaE1K0sQyQdERwpktxyWJzjmx3QxUefYbfVNQf3RUFByUsaTUzjd0rnwURWU1BQoVF1l0JBURBz0t7rUe4R0ExxWrd0VCC193
UUZNUQ1CQWY4ddohRWRUWUjBpQkjZRUZNW9x9cEtaOnppdtkyc114d0JUT1lBU3NpeuVNQlVHQtFVZBzv5Ym1MMPyxDxEVULks29asWh2y05BUUVMQlfFBGdnRU
JBRG5aRxRKYs9Lbxc4Tkt3W8W0AppQ25EWhdiaE9mNxPzdoJGMnozykZkeGFOw4xZ25hNuKuzNDDM1KBLStMa2RvYkh0MsVb1fjd1RLteKzKKGwxCmlpdThqeFcyCvVZwtQc2YzTRK
Rk15uk4vnenxWu3d1bVpHmfJr1hla2QkZB1b0KeHtVUHRYVlyq2NPRGZJwd0JGeE1LNhvbXhwZDhqek1xa1pbGNNRnw2cHv15jJYL2t0NEuMTlGMV
JPRQpLk2VCTfdav0My3NncjQUT4bvTpaVu1CfH25uLWthHvkdyltwAEr6XZp1h035VnpdmxQnhd3CmStQmhRs1fFbeJETGdzctmU2hjMFQk2dMdl3z3BjQXNPUnVi
VXRNVLBNStW84VfH0L12aqLzUYXF0MXMK0F4PQotLs0tLUvORCBDRVjUSU2J0qFURS0tLs0tcgg=
server: https://172.31.27.210:6443
  name: kubernetes
contexts:
  contexts:
    - context:
      cluster: kubernetes
      user: kubernetes-admin
      name: kubernetes-admin@kubernetes
    current-context: kubernetes-admin@kubernetes
    kind: Config
    preferences: {}
  users:
    - name: kubernetes-admin
      user:
        client-certificate-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCK1JSURJVENDQwdtZ0F3SUJB0lJQ2dtUnAvdFNEUnN3RFFZSktvWklodmN0QVFFTEJRQxdGVEVUT
UJFR0ExVUUKQXHN52EZVmLaEp1WlsbGN6QVgdzB5TfRfEEtQXdpfVfE1TrkWYU3MhLnakV4TVRbd09UUTVORGhtURReApceKfWQmdovkJbb1REbk-41YzNsbgQjUCHRzWe4wWhkeki
5a39111EVfLFRREVQJnJ0pysY201bGRHvnpmV8zrCmJxbHvNULcsKpBtKjna3FoazLHOxcwQkFRRUZBQ9dQve4Q1u12d0J28tDQVFF0QXZdm8wzm12ed9L1zLcTQKb25TnFh1RIR
HtYCNWVfXWu3d1bVpHmfJr1hla2QkZB1b0KeHtVUHRYVlyq2NPRGZJwd0JGeE1LNhvbXhwZDhqek1xa1pbGNNRnw2cHv15jJYL2t0NEuMTlGMV
jytsd1z0RHZhck4LybmrxMDJtN0R3hD30URPRVNTWJGd0NPNxFwZf3yCTFPz1hTz34ewExaGpuGsrM2R0eksFU1VhTlFTXYKu2d1TzdKOGFKY1pmSFVndTJH1mRuSXNbfbHCL
OEyehhX0Hn2ZHJBQJd3ZfPUYXksLS14Qss3R0yBvR8cE10bpgp1QVlCR2V1Y1Vz1cVbmrppVhJYRnVHzmShL2f3TG5pc0pqeJvScvBuc1hVvSV51vvn915DRCtndFm0wRupChkJv2U
jdoleQVFbm8xWxJhRfEFPOmdrcwhraucSdzBcqvFzRkFbt0BuUvBvUhFzVTbUvKsxZm5jeVufA1UnvSVmwrN1zLkpsRERfVnZnCmVndVhaCtVHJQjUvkhkovkvvn
WM0cnZkckVnTUfVfemZRRX1Jcwp0REFQ0mdrcwhraucSdzBcqvFzRkFbt0BuUvBvUhFzVTbUvKsxZm5jeVufA1UnvSVmwrN1zLkpsRERfVnZnCmVndVhaCtVHJQjUvkhkovkvvn
lUi1g3V3Fp2ZRUcwl1wNz2Utd1UDYwM202bHBmZwIvas2xqld03cvgyvuVMKcktMVm4-Rj1wSUZH0VphbznbupcYmnlbjhMNWcrnJRLM005d1JNT1lUmz1awdbdQVYxdEthsfh1dzVFY3RN
Qp15GU0Y0NlNk14VkhmGbGlia1RoamxpeldFUTdxTVZlemluV3N53kxeWtFNDhta2xrUbDzq2xidplzaTuvoDUIrcjv1VVdadzhuQmRtLzd5edDV3u1VpMFBMeUx1b1ZPMUfMN21VZ2dnawD

```

## Step - 2 : Setting up jenkins On aws ec2 ubuntu

### 2.1 : Installing java and jenkins

```

$ sudo apt-get update -y
$ sudo apt install openjdk-8-jdk
$ java -version
$ wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -
$ sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ >
\ 
/etc/apt/sources.list.d/jenkins.list'
$ sudo apt-get update
$ sudo apt-get install jenkins

```

Open tcp custom port 8080 on ec2 machine , to verify jenkins installation open <http://public-ip-address-of-ec2-instance:8080> like <http://34.216.41.126:8080> on browser.

Output :

Getting Started

## Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

```
/var/lib/jenkins/secrets/initialAdminPassword
```

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

**Administrator password**

We need to provide Default jenkins(initialAdminPassword) administrator password, which is present at given path (in the page)

\$ cat given-path

For example

\$ cat /var/lib/jenkins/secrets/initialAdminPassword

In the next page select on Install suggested plugins

After you have installed all the suggested default plugins, it will prompt you for setting up admin username and password -

Activities Chromium Web Browser Sun 13:48

Setup Wizard [Jenkins] - Chromium

Getting Started

### Create First Admin User

Username:

Password:

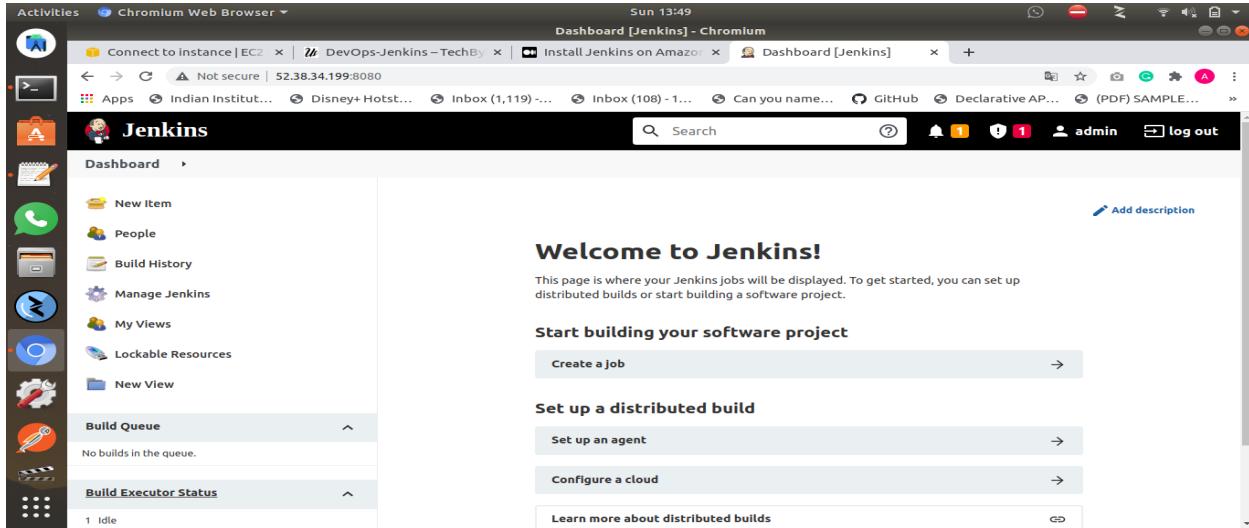
Confirm password:

Full name:

E-mail address:

Jenkins 2.319 Skip and continue as admin Save and Continue

After creating user credentials, jenkins is ready to use.



## 2.2 : Installing docker on jenkins server

Same steps mentioned in step 1.1.1 to install docker

To add current ubuntu usr and jenkins to docker group

```
$ sudo usermod -aG docker $USER
$ sudo usermod -aG docker jenkins
```

## Step - 3: Setting up code

We made a simple java application, code is on github.

Code link : [github url](#)

code also includes the Dockerfile for building the docker image

```
FROM openjdk:11
ARG JAR_FILE
COPY ${JAR_FILE}
ENTRYPOINT "java" "-jar" "/app.jar"
```

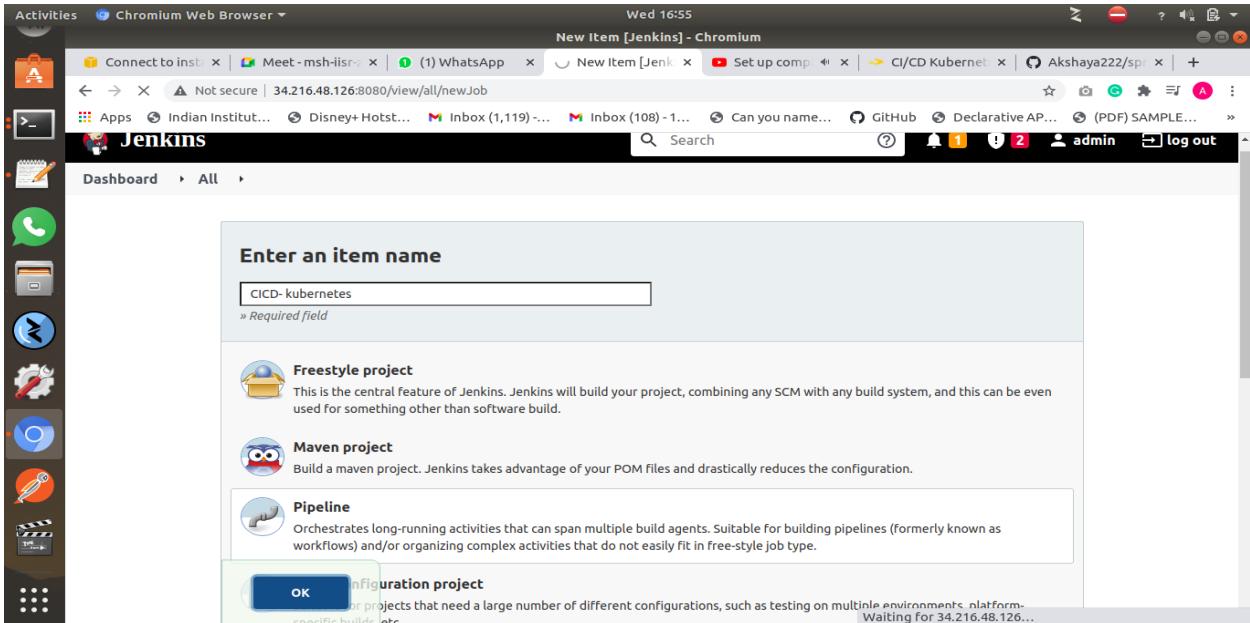
Deployment.yml file is present in the github repo.

## Step - 4: Building the pipeline

### 4.1 : Creating a pipeline

Go to Jenkins dashboard -> new items -> enter item name:CICD-pipeline -> select pipeline -> click ok

After creating a pipeline , select pipeline.



## 4.2 : Cloning the github repository

First stage in the pipeline is to clone the code from github repository , first we need to setup the github credentials

### 4.2.1 : setup github credentials

Go to jenkins dashboard -> manage jenkins -> manage credentials -> stores scope to jenkins -> global -> add credentials

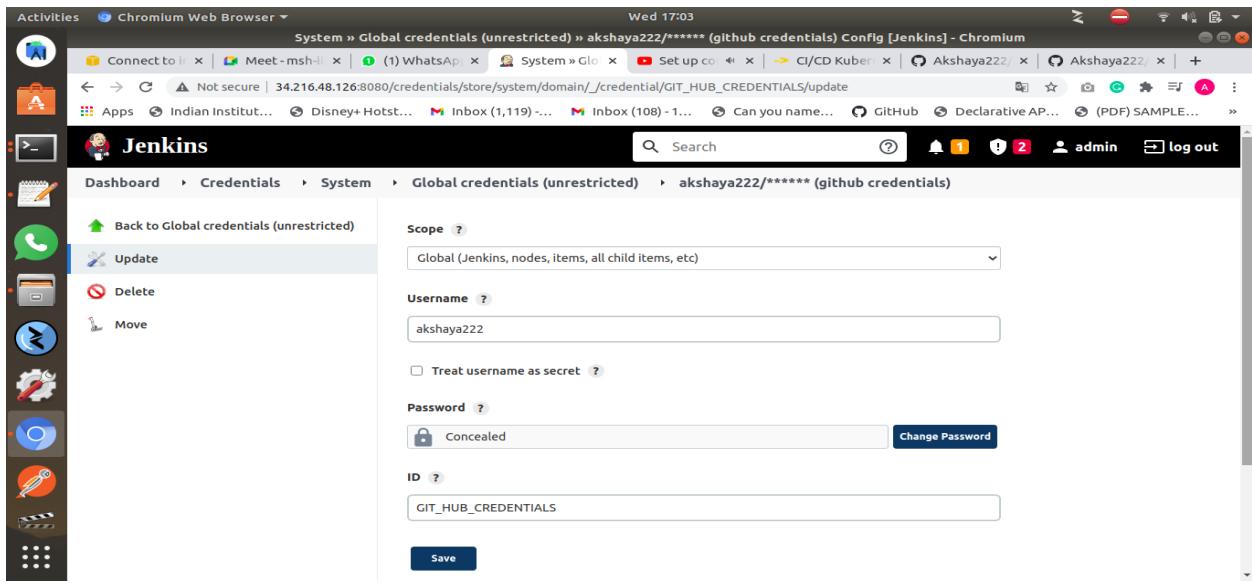
Select username and password from select menu

Scope : select global scope

For username : Github username

For password : Github Personal Access Token (PAT)

For ID : GIT\_HUB\_CREDENTIALS

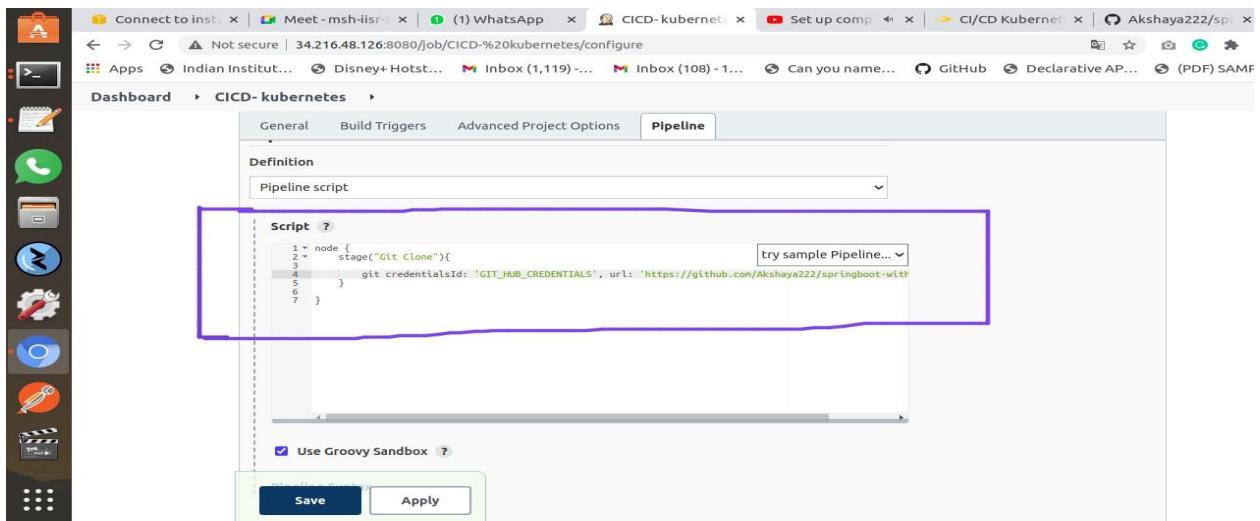


#### 4.2.2 : adding stage in pipeline

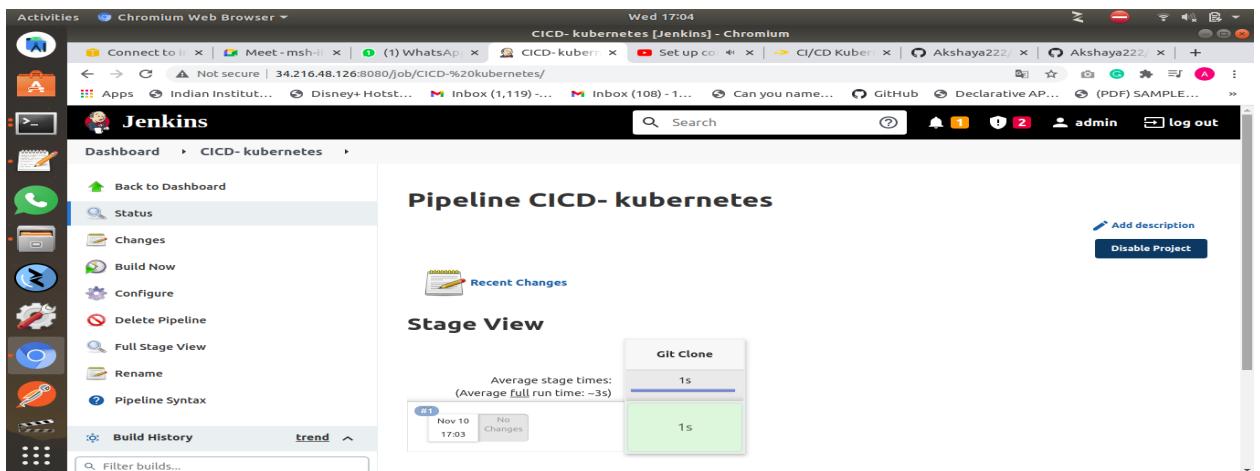
git credentialsId: Id of the github credentials- GIT\_HUB\_CREDENTIALS is the id of credentials created just before this

url : url of the github repository

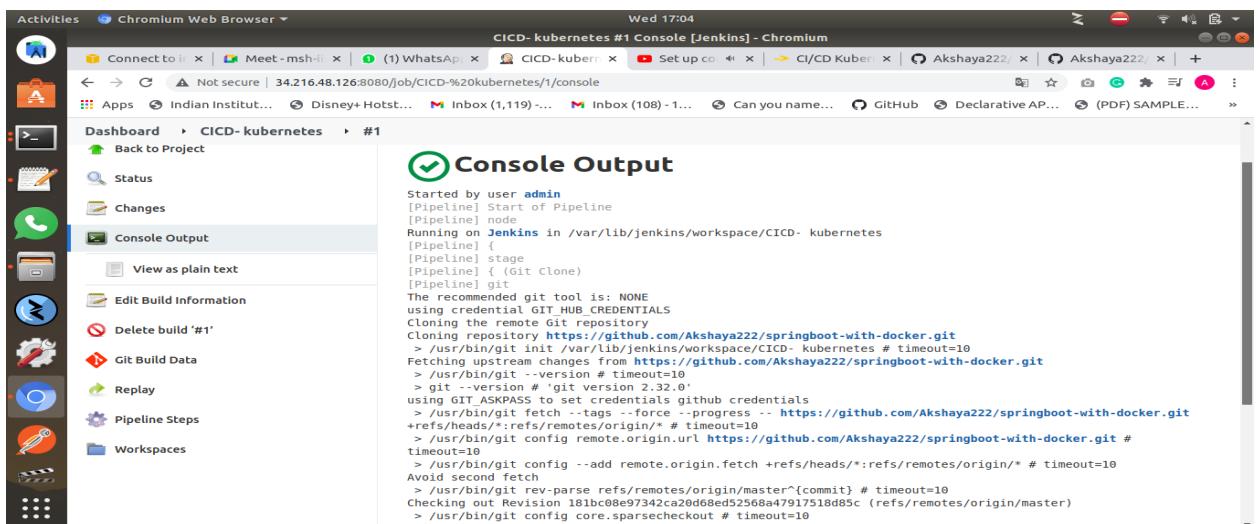
```
node {
  stage("Git Clone") {
    git credentialsId: 'GIT_HUB_CREDENTIALS',
    url:'https://github.com/Akshaya222/springboot-with-docker.git'
  }
}
```



## Build status of first stage



## Logs of the build:

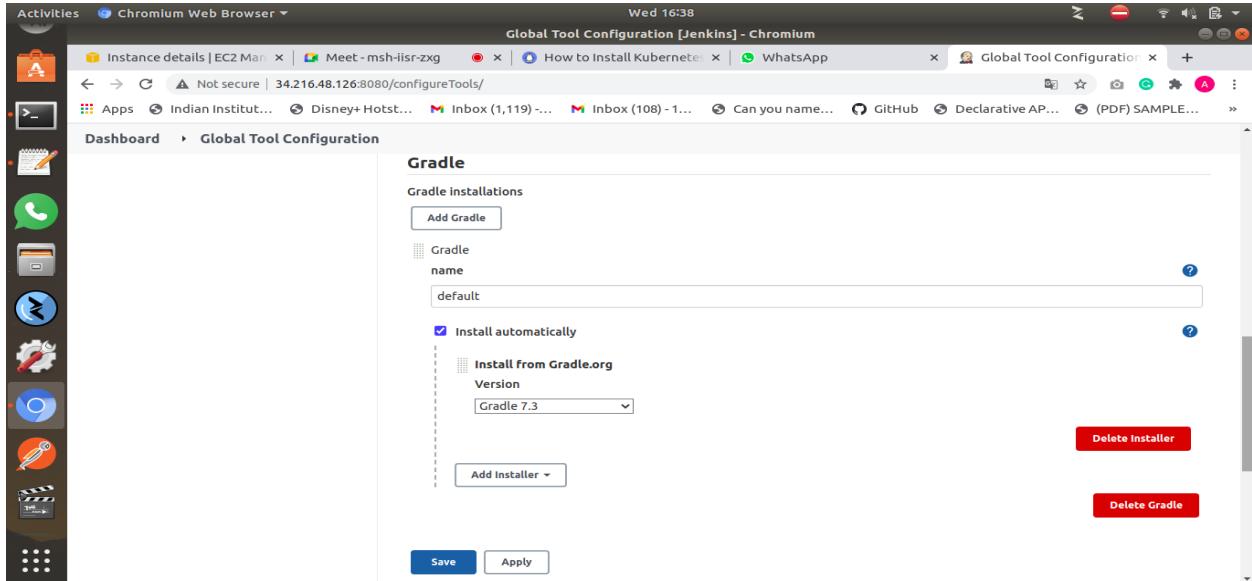


## 4.3 : Building the spring boot application using gradle

### 4.3.1 : setup gradle for building application

Go to jenkins dashboard -> manage jenkins -> global tool configuration -> gradle -> add gradle

Enter name as default ,Select latest version from select menu and enable install automatically.



### 4.3.2 : adding stage for building application using gradle

```
stage('Gradle Build') {
    sh './gradlew build'
}
```

## 4.4 : Build the docker image and tag it

### 4.4.1 : Create a repository in the docker hub

I have created a repository named akshaya-docker-hub-repos.

### 4.4.2 adding stage for building docker image and tagging it.

```
stage("Docker build") {
    sh 'docker version'
    sh 'docker build -t new-image-name.'
    sh 'docker image list'
```

```
sh 'docker tag new-image-name
docker_id/repository_name:new-image-name'
}
```

## 4.5 : Adding docker login stage

### 4.5.1 : create docker credentials in jenkins

Go to jenkins dashboard -> manage jenkins -> manage credentials -> stored scoped to jenkins -> global -> add credentials

Select secret text from the kind select menu

Select: scope as global scope

Secret :docker hub password

Id : DOCKER\_HUB\_PASSWORD

### 4.5.2 : adding stage for docker login for jenkins

credentialsId : ID of docker credentials

```
stage("Docker Login") {
    withCredentials([string(credentialsId: 'DOCKER_HUB_PASSWORD',
variable: 'PASSWORD')]) {
        sh 'docker login -u akshayalockerlid -p $PASSWORD'
    }
}
```

## 4.6 : Pushing the tagged docker image into docker hub

```
stage("Push Image to Docker Hub") {
    sh 'docker push
akshayalockerlid/akshaya-docker-hub-repos:akshaya-docker-image'
}
```

## 4.7 : add stage for login in to the kubernetes master node from jenkins server

We do this by through ssh

### 4.7.1 : add ssh pipeline plugin

Go to jenkins dashboard -> manage jenkins -> manage plugins -> available -> in the search box type ssh pipeline steps ->

select ssh pipeline plugin -> install without restart.

#### 4.7.2 : create password for kubernetes master server (ec2 ubuntu)

```
$ sudo vi /etc/ssh/sshd_config
```

Change the line passwordAuthentication from "no" to "yes"

Change the line permitRootLogin from "prohibit-password" to "yes"

```
$ sudo passwd ubuntu
```

It will prompt you to enter a password.

```
$ sudo service sshd restart
```

#### 4.7.3 : adding the stage for ssh-ing into k8s server

remote.name= any name

remote.host : public ip of kubernetes master

remote.user:ubuntu remote.password:password of ec2 instance

```
stage("SSH Into k8s Server") {
    def remote = [:]
    remote.name = 'kubernetes-master'
    remote.host = '35.86.87.241'
    remote.user = 'ubuntu'
    remote.password = 'ubuntu'
    remote.allowAnyHosts = true
}
```

### 4.8 : add stage for deploying the application on kubernetes

#### 4.8.1 : copy deployment.yml file to kubernetes master

Adding stage for copying deployment.yml to kubernetes master server

deployment.yml name : k8s-spring-boot-deployment.yml

This copies k8s-spring-boot-deployment.yml to the root directory of kubernetes master.

```
stage('Put deployment.yml onto kubernetes master') {
    sshPut remote: remote, from:
    'k8s-spring-boot-deployment.yml', into: '.'
}
```

Checking the k8s-spring-boot-deployment.yml on kubernetes server



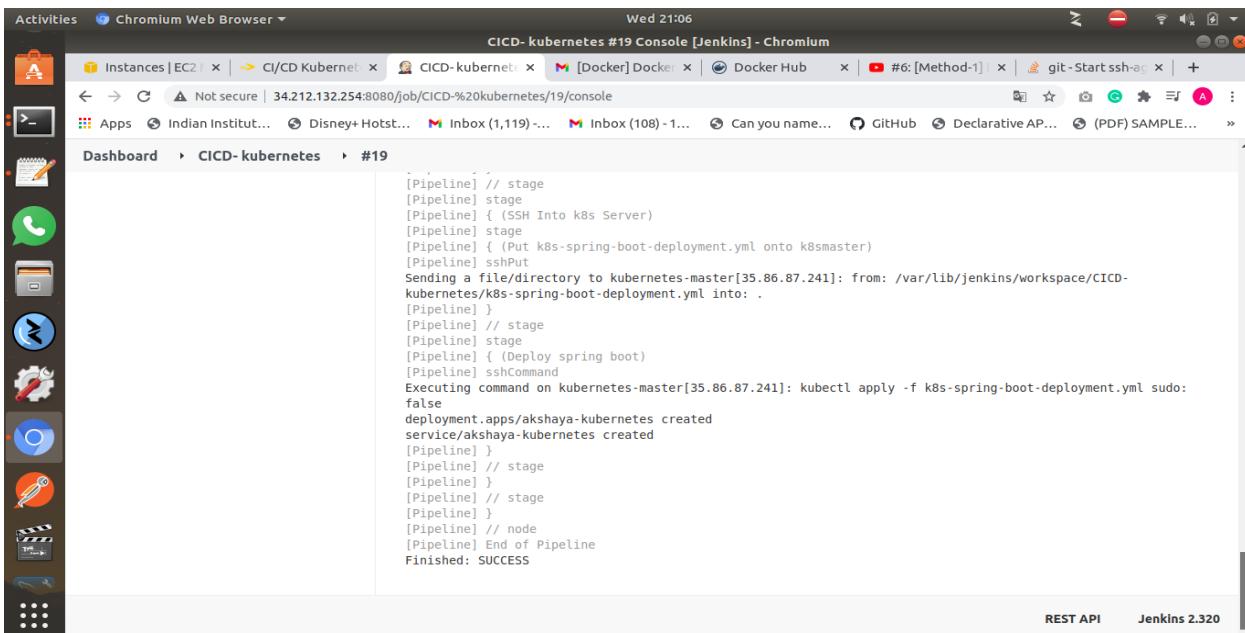
```
Last login: Wed Nov 10 15:00:46 2021 from 103.217.239.132
ubuntu@ip-172-31-27-210:~$ kubectl get nodes
NAME           STATUS    ROLES      AGE     VERSION
ip-172-31-16-106  NotReady  worker   5h6m    v1.22.3
ip-172-31-27-210  Ready     control-plane,master 5h45m   v1.22.3
ubuntu@ip-172-31-27-210:~$ ls
k8s-spring-boot-deployment.yml
ubuntu@ip-172-31-27-210:~$
```

#### 4.8.2 : Creating the deployment and service on kubernetes

This command creates deployment and exposes the service on nodeport

```
stage('Deploy spring boot') {
    sshCommand remote: remote, command: "kubectl apply -f k8s-spring-boot-deployment.yml"
}
```

Logs of deployment on jenkins :

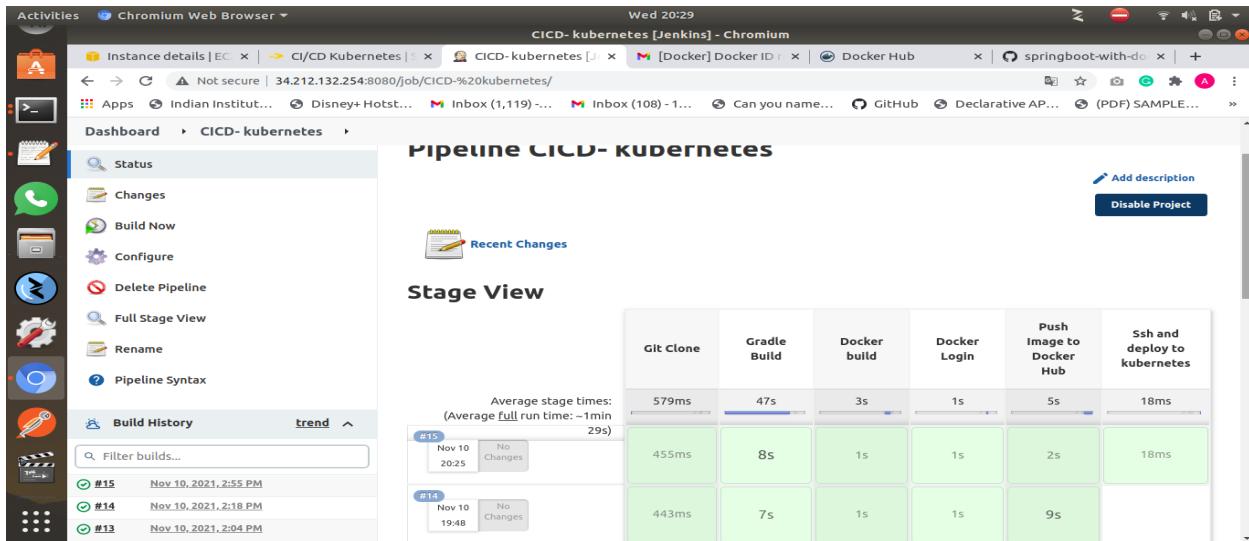


The Jenkins pipeline log shows the execution of a script to deploy a Spring Boot application to a Kubernetes cluster. The log output is as follows:

```
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (SSH Into k8s Server)
[Pipeline] stage
[Pipeline] { (Put k8s-spring-boot-deployment.yml onto k8smaster)
[Pipeline] sshPut
Sending a file/directory to kubernetes-master[35.86.87.241]: from: /var/lib/jenkins/workspace/CICD-kubernetes/k8s-spring-boot-deployment.yml into: .
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy spring boot)
[Pipeline] sshCommand
Executing command on kubernetes-master[35.86.87.241]: kubectl apply -f k8s-spring-boot-deployment.yml sudo: false
deployment.apps/akshaya-kubernetes created
service/akshaya-kubernetes created
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

At the bottom of the log, it indicates the Jenkins version is 2.320.

## Build view of all stages



## Checking the deployment on kubernetes server

To get list of deployments on kubernetes server

```
$ kubectl get deployments
```

To get list of services on kubernetes server

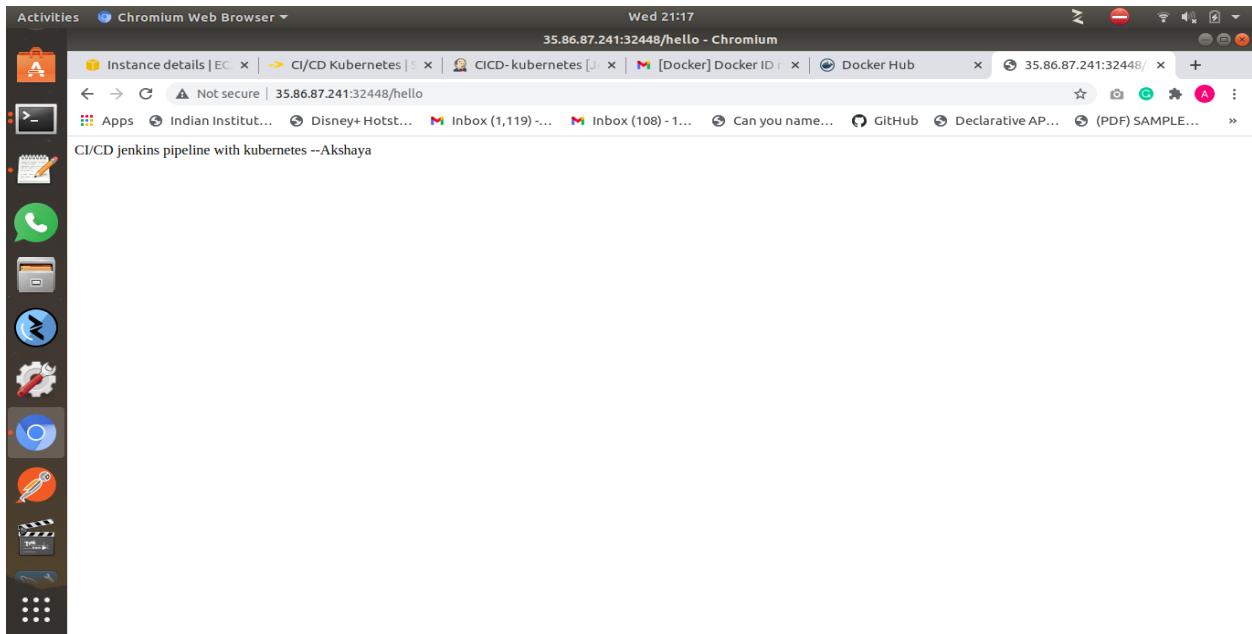
```
$ kubectl get service
```

```
ubuntu@ip-172-31-27-210:~$ ls
k8s-spring-boot-deployment.yml
ubuntu@ip-172-31-27-210:~$ kubectl get deployments
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
akshaya-kubernetes  3/3     3           3          4m14s
ubuntu@ip-172-31-27-210:~$ kubectl get service
NAME            TYPE      CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
akshaya-kubernetes  NodePort  10.100.183.205  <none>       80:32448/TCP  4m42s
kubernetes       ClusterIP  10.96.0.1    <none>       443/TCP   5h49m
ubuntu@ip-172-31-27-210:~$
```

Akshaya-kubernetes service is running on Nodeport, so Deployment on kubernetes is successful.

To verify the service running open [http://public-ip-of-kubernetes-server:port\\_of\\_service/hello](http://public-ip-of-kubernetes-server:port_of_service/hello)

From the terminal output, akshaya-kubernetes service is running on port number 37448, so open tcp custom port 37448 on kubernetes server instance.



Application is successfully deployed on to kubernetes through jenkins CICD pipeline.

# Task - 2 : Jenkins Master Slave pipeline

Tools and Technologies used :

- Github
- Docker
- Jenkins

System Requirements :

3 AWS Ec2 Linux instances of size t2.micro.

## Step - 1 : Setting up Jenkins Master

### 1.1 : Connect to ec2 linux instance through ssh

```
[SNamratha-18bc083-MacBook-Pro:~ harshitha$ cd Desktop
[SNamratha-18bc083-MacBook-Pro:Desktop harshitha$ chmod 400 master.pem
[SNamratha-18bc083-MacBook-Pro:Desktop harshitha$ ssh -i "master.pem" ec2-user@ec2-3-110-41-107.ap-south-1.compute.amazonaws.com
The authenticity of host 'ec2-3-110-41-107.ap-south-1.compute.amazonaws.com (3.110.41.107)' can't be established.
ECDSA key fingerprint is SHA256:Tma+Fj/AVJTYK+ZEYbSlTZHEkpmY0itXntToivaFvZk.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-110-41-107.ap-south-1.compute.amazonaws.com,3.110.41.107' (ECDSA) to the list of known hosts.

--| --|- )
--| (   /  Amazon Linux 2 AMI
--| \---|---|
https://aws.amazon.com/amazon-linux-2/
1 package(s) needed for security, out of 14 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-4-13 ~]$ sudo yum update -y
```

### 1.2 : Install Java and jenkins

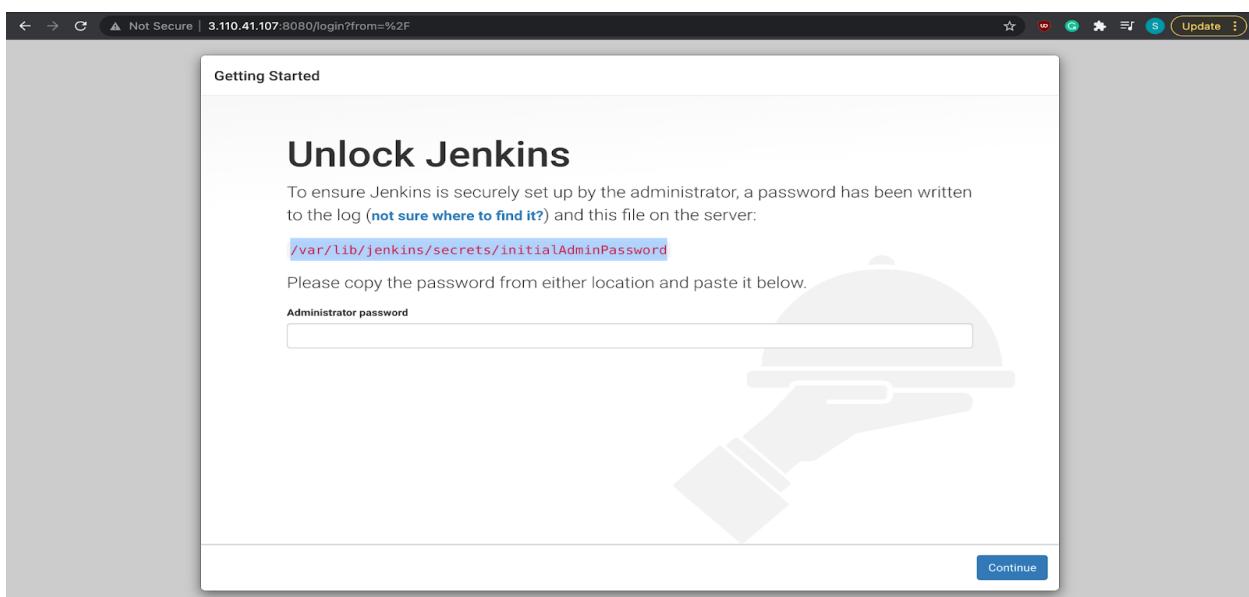
```
$ sudo yum update -y
$ sudo yum install java-1.8.0
$ sudo wget -O /etc/yum.repos.d/jenkins.repo
http://pkg.jenkins-ci.org/redhat/jenkins.repo
$ sudo rpm - import https://pkg.jenkins.io/redhat/jenkins.io.key
$ amazon-linux-extras install epel -y
$ sudo yum install jenkins -y
$ sudo service jenkins start
```

```
Desktop — ec2-user@ip-172-31-4-13:~ — ssh -i master.pem ec2-user@ec2-172-31-4-13
libXau.x86_64 0:1.0.8-2.1.amzn2.0.2
libXcomposite.x86_64 0:0.4.4-4.1.amzn2.0.2
libXcursor.x86_64 0:1.1.15-1.amzn2
libXdamage.x86_64 0:1.1.4-4.1.amzn2.0.2
libXext.x86_64 0:1.3.3-3.amzn2.0.2
libXfixes.x86_64 0:5.0.3-1.amzn2.0.2
libXft.x86_64 0:2.3.2-2.amzn2.0.2
libXi.x86_64 0:1.7.9-1.amzn2.0.2
libXinerama.x86_64 0:1.1.3-2.1.amzn2.0.2
libXrandr.x86_64 0:1.5.1-2.amzn2.0.3
libXrender.x86_64 0:0.9.10-1.amzn2.0.2
libXtst.x86_64 0:1.2.3-1.amzn2.0.2
libXxf86vm.x86_64 0:1.1.4-1.amzn2.0.2
libfontenc.x86_64 0:1.1.3-3.amzn2.0.2
libglvnd.x86_64 1:1.0.1-0.1.git5baa1e5.amzn2.0.1
libglvnd-egl.x86_64 1:1.0.1-0.1.git5baa1e5.amzn2.0.1
libglvnd-glx.x86_64 1:1.0.1-0.1.git5baa1e5.amzn2.0.1
libthai.x86_64 0:0.1.14-9.amzn2.0.2
libwayland-client.x86_64 0:1.17.0-1.amzn2
libwayland-server.x86_64 0:1.17.0-1.amzn2
libxcb.x86_64 0:1.12-1.amzn2.0.2
libxshmfence.x86_64 0:1.2-1.amzn2.0.2
libxslt.x86_64 0:1.1.28-6.amzn2
lksctp-tools.x86_64 0:1.0.17-2.amzn2.0.2
mesa-libEGL.x86_64 0:18.3.4-5.amzn2.0.1
mesa-libGL.x86_64 0:18.3.4-5.amzn2.0.1
mesa-libgbm.x86_64 0:18.3.4-5.amzn2.0.1
mesa-libglapi.x86_64 0:18.3.4-5.amzn2.0.1
pango.x86_64 0:1.42.4-4.amzn2
pcsc-lite-libs.x86_64 0:1.8.8-7.amzn2
pixman.x86_64 0:0.34.0-1.amzn2.0.2
python-javapackages.noarch 0:3.4.1-11.amzn2
python-lxml.x86_64 0:3.2.1-4.amzn2.0.3
ttmkfdi.x86_64 0:3.0.9-42.amzn2.0.2
tzdata-java.noarch 0:2021a-1.amzn2
xorg-x11-font-utils.x86_64 1:7.5-21.amzn2
xorg-x11-fonts-Type1.noarch 0:7.5-9.amzn2

Complete!
[ec2-user@ip-172-31-4-13 ~]$ sudo systemctl start jenkins
```

Open custom tcp port 8080 on this ec2 instance

Go to browser and hit <http://public-ip-of-instance:8080>



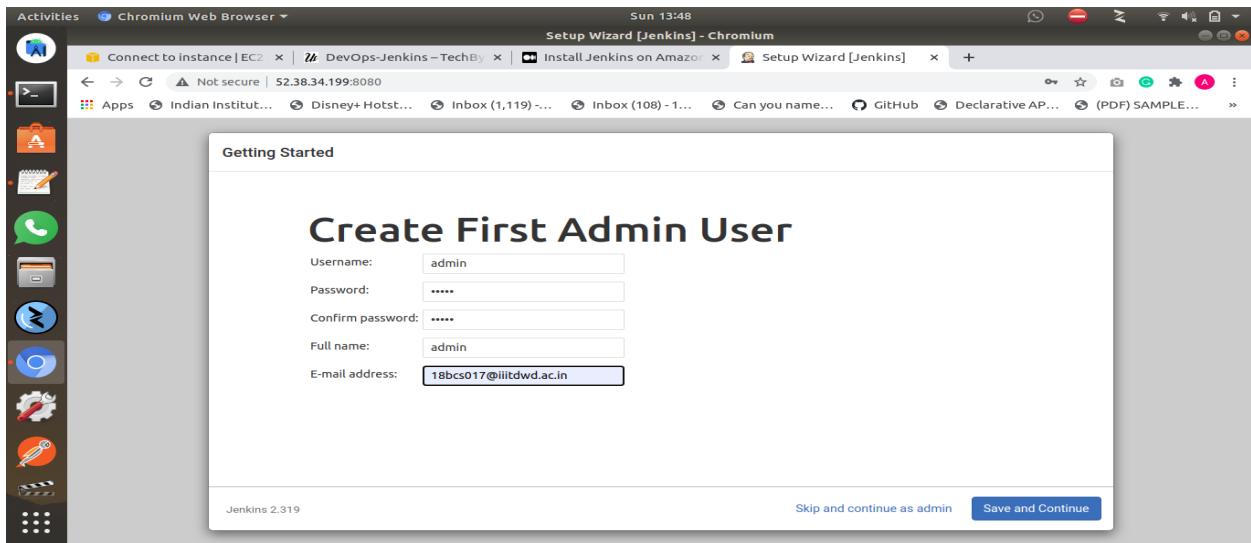
We need to provide Default jenkins(initialAdminPassword) administrator password, which is present at given path (in the page)

\$ cat given-path For example

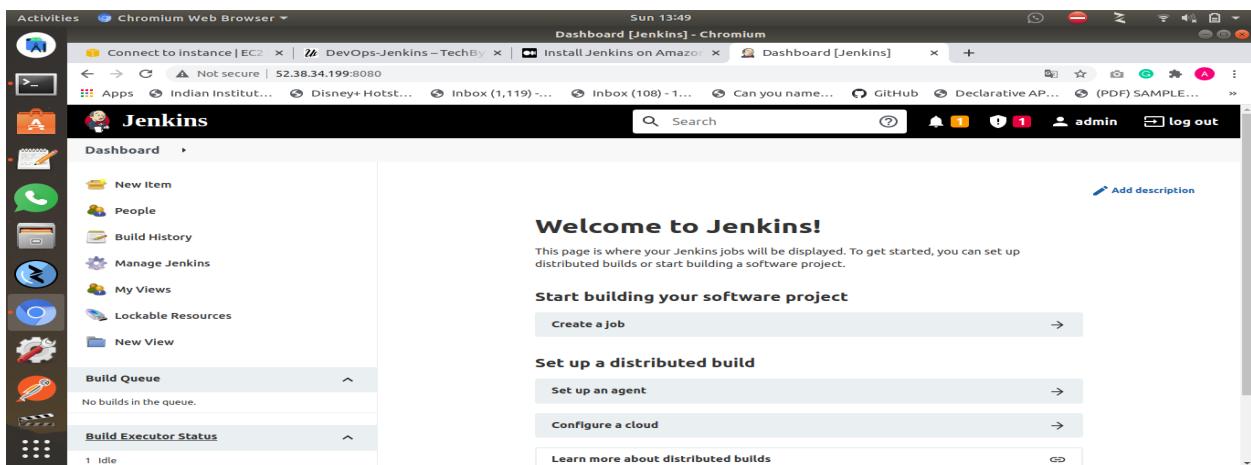
```
$ cat /var/lib/jenkins/secrets/initialAdminPassword
```

In the next page select on Install suggested plugins

After you have installed all the suggested default plugins, it will prompt you for setting up admin username and password -



After creating user credentials, jenkins is ready to use.



### 1.3 : Configuring this server as jenkins master

### 1.3.1 : Name other 2 instances as slave-1 and slave-2

1.3.2 : Go to jenkins -> Manage jenkins -> Configure global security. In agents, change TCP port for inbound agents to random and save.

The screenshot shows the Jenkins 'Configure Global Security' configuration page. The 'Agents' section is the primary focus, displaying the setting for 'TCP port for inbound agents' which is configured to 'Random'. Other sections visible include 'Authorization' (with 'Logged-in users can do anything' selected), 'Markup Formatter' (set to 'Plain text'), and 'CSRF Protection' (with 'Crumb Issuer' enabled). At the bottom are 'Save' and 'Apply' buttons.

### 1.3.3 connect slave-1 and slave-2 to this jenkins master (do same for both slaves)

Go to jenkins dashboard -> manage nodes -> add new node

Enter name as slave -1 for slave -1 and slave -2 for slave-2 and enable permanent agent and click ok.

The screenshot shows the Jenkins 'Nodes' configuration page with a modal dialog for adding a new node. The node name is set to 'slave-1'. The 'Permanent Agent' checkbox is checked, with a descriptive note explaining it adds a plain, permanent agent to Jenkins. Below the checkbox is an 'OK' button. The left sidebar shows navigation options like 'Dashboard', 'Manage Jenkins', and 'Build Queue'.

In the launch method select launch by connecting to master from select menu.

For root directory -> /home/ec2-user

Not Secure | 3.110.41.107:8080/computer/slave-1/configure

Dashboard > Nodes > slave-1

**Build Executor Status**

Idle
1

**Remote root directory** (Mandatory)

**Labels**

**Usage** (Use this node as much as possible)

**Launch method** (Launch agent by connecting it to the master)

- Disable WorkDir
- Custom WorkDir path /home/ec2-user
- Internal data directory remoting
- Fail if workspace is missing
- Use WebSocket

**Save**

Manage nodes page looks like :

Jenkins

Not Secure | 3.110.41.107:8080/computer/

Dashboard > Nodes >

**Build Queue**

No builds in the queue.

**Build Executor Status**

master
1 Idle
2 Idle
slave-1 (offline)
slave-2 (offline)

**Nodes**

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
master	Linux (amd64)	In sync	5.68 GB	0 B	5.68 GB	0ms	
slave-1		N/A	N/A	N/A	N/A	N/A	
slave-2		N/A	N/A	N/A	N/A	N/A	

Data obtained 1 min 49 sec 1 min 49 sec

**Refresh status**

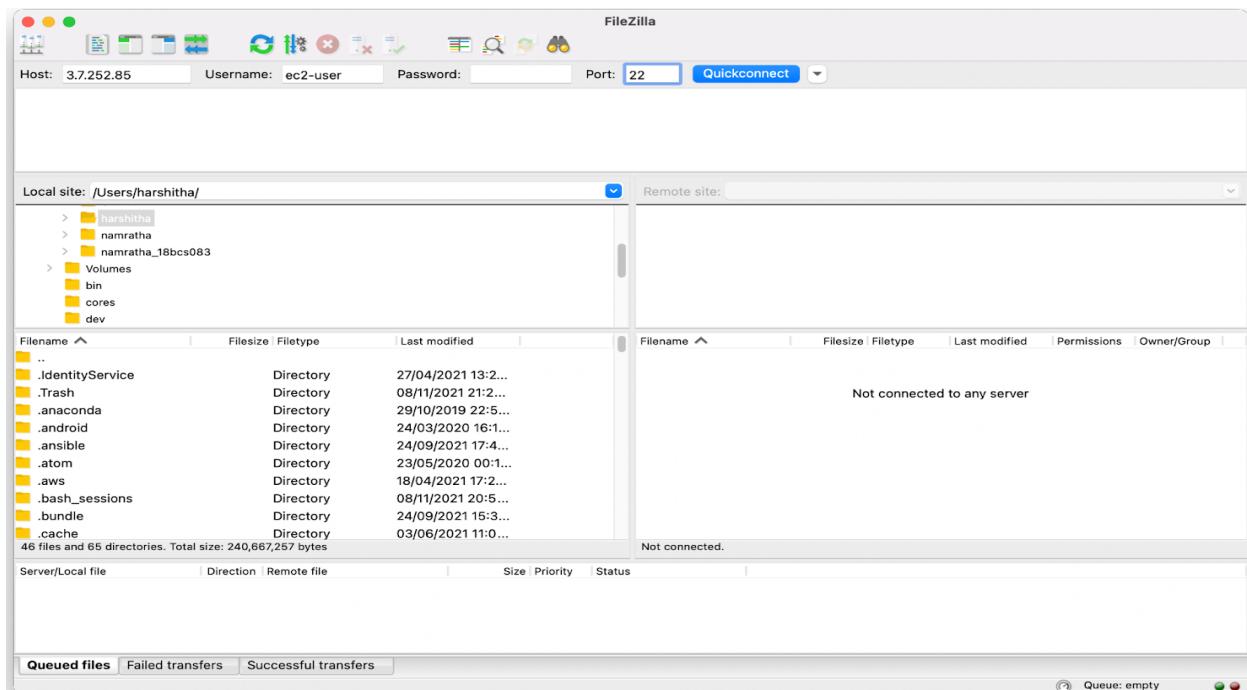
REST API Jenkins 2.303.3

Go to slave-1 and download the agent.jar file.

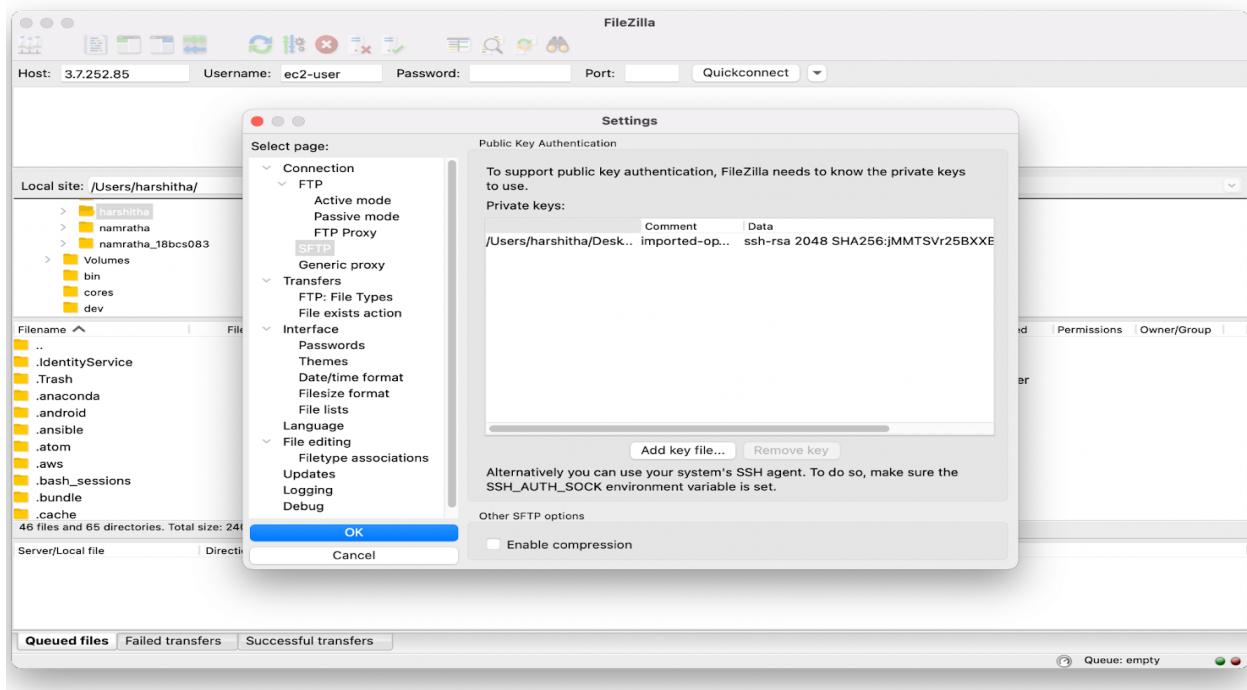
Go to slave1 EC2 instance and select public ip address.

Open filezilla.

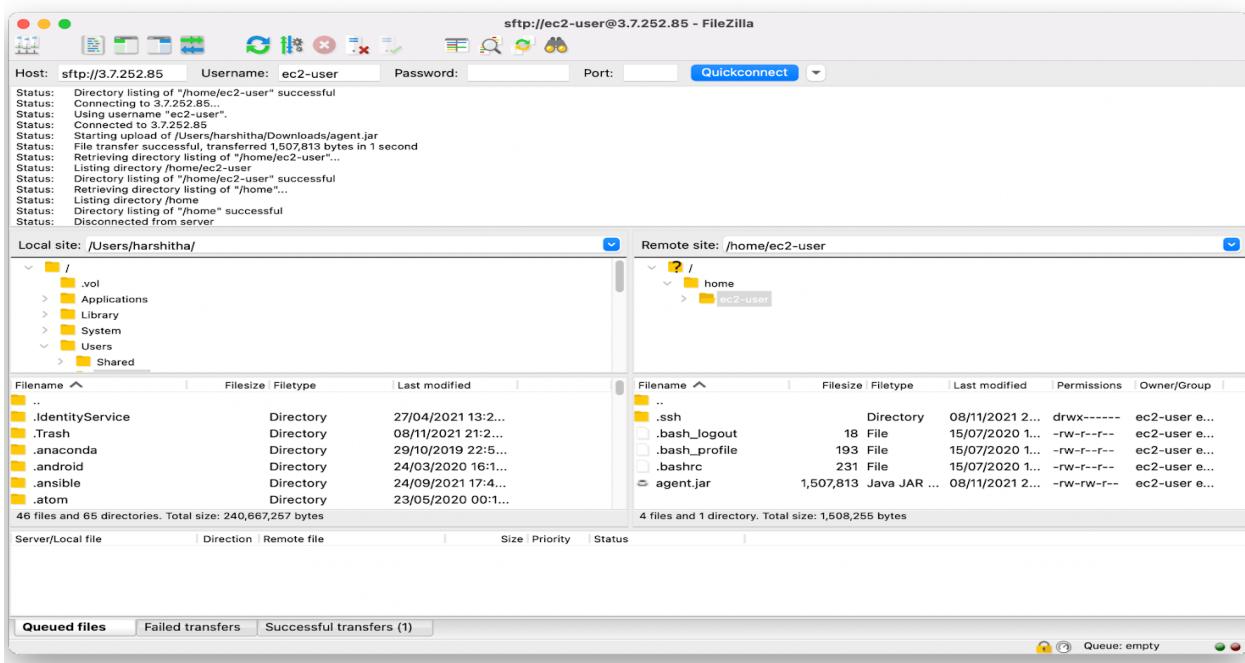
Paste the ip address, change username to ec2-user, and port number 22.



In setting -> SFTP, add the pem file.



After hosting, add the agent.jar file in ec2-user.



To verify, connect to slave-1 and type command - ls. Agent.jar is present.

```
Desktop — ec2-user@ip-172-31-13-3:~ — ssh -i master.pem ec2-user@ec2-3-7-252-85.ap-south-1.compute.amazonaws.com
Last login: Mon Nov  8 20:58:48 on ttys000
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
[SNamratha-18bc883-MacBook-Pro:~ harshitha$ cd Desktop
[SNamratha-18bc883-MacBook-Pro:Desktop harshitha$ chmod 400 master.pem
[SNamratha-18bc883-MacBook-Pro:Desktop harshitha$ ssh -i "master.pem" ec2-user@ec2-3-7-252-85.ap-south-1.compute.amazonaws.com
The authenticity of host 'ec2-3-7-252-85.ap-south-1.compute.amazonaws.com (3.7.252.85)' can't be established.
ECDSA key fingerprint is SHA256:ogI2LM78ludcmKbEJc5q7B2UFxxOjo7E9Y8XPJE+q0s.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-7-252-85.ap-south-1.compute.amazonaws.com,3.7.252.85' (ECDSA) to the list of known hosts.

--| --|_
_| (   /  Amazon Linux 2 AMI
---|---|---|
```

https://aws.amazon.com/amazon-linux-2/  
1 package(s) needed for security, out of 14 available  
Run "sudo yum update" to apply all updates.  
[ec2-user@ip-172-31-13-3 ~]\$ ls  
**agent.jar**  
[ec2-user@ip-172-31-13-3 ~]\$

Install java on slaves

```
$ sudo yum update -y
$ sudo yum install java-1.8.0
```

```
[ec2-user@ip-172-31-13-3 ~]$ java -jar agent.jar -jnlpUrl http://3.110.41.107:8080/computer/slave-1/jenkins-agent.jnlp -secret 176b64fb55878df5236c5b93c0f0fa0fd43e1fa579e1bdf53ccf81b35e13d4d -workDir "/home/ec2-user"
Nov 08, 2021 4:58:18 PM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /home/ec2-user/remoting as a remoting work directory
Nov 08, 2021 4:58:19 PM org.jenkinsci.remoting.engine.WorkDirManager setupLogging
INFO: Both error and output logs will be printed to /home/ec2-user/remoting
Nov 08, 2021 4:58:19 PM hudson.remoting.jnlp.Main createEngine
INFO: Setting up agent: slave-1
Nov 08, 2021 4:58:19 PM hudson.remoting.jnlp.Main$Cuilistener <init>
INFO: Jenkins agent is running in headless mode.
Nov 08, 2021 4:58:19 PM hudson.remoting.Engine startEngine
INFO: Using Remoting version: 4.10.1
Nov 08, 2021 4:58:19 PM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /home/ec2-user/remoting as a remoting work directory
Nov 08, 2021 4:58:19 PM hudson.remoting.jnlp.Main$Cuilistener status
INFO: Locating server among [http://3.110.41.107:8080/]
Nov 08, 2021 4:58:19 PM org.jenkinsci.remoting.engine.JnlpAgentEndpointResolver resolve
INFO: Remoting server accepts the following protocols: [JNLP4-connect, Ping]
Nov 08, 2021 4:58:19 PM hudson.remoting.jnlp.Main$Cuilistener status
INFO: Agent discovery successful
Agent address: 3.110.41.107
Agent port: 42959
Identity: 78:8d:f7:95:e6:e8:cd:22:8a:7c:69:e5:d5:f4:00:92
Nov 08, 2021 4:58:19 PM hudson.remoting.jnlp.Main$Cuilistener status
INFO: Handshaking
Nov 08, 2021 4:58:19 PM hudson.remoting.jnlp.Main$Cuilistener status
INFO: Connecting to 3.110.41.107:42959
Nov 08, 2021 4:58:19 PM hudson.remoting.jnlp.Main$Cuilistener status
INFO: Trying protocol: JNLP4-connect
Nov 08, 2021 4:58:19 PM org.jenkinsci.remoting.protocol.impl.BIONetworkLayer$Reader run
INFO: Waiting for ProtocolStack to start.
Nov 08, 2021 4:58:19 PM hudson.remoting.jnlp.Main$Cuilistener status
INFO: Remote identity confirmed: 78:8d:f7:95:e6:e8:cd:22:8a:7c:69:e5:d5:f4:00:92
Nov 08, 2021 4:58:21 PM hudson.remoting.jnlp.Main$Cuilistener status
INFO: Connected
[]
```

Slave -1 looks like this, Agent is connected.

The screenshot shows the Jenkins interface for managing a slave node. The top navigation bar includes links for 'Dashboard', 'Nodes', and 'slave-1'. The main content area displays the following information:

- Status:** Agent is connected.
- Projects tied to slave-1:** None
- Build Executor Status:** 1 idle
- File Manager:** Shows 'agent.jar' and 'FileZilla\_3.5....tar.bz2' files.

Slaves are in sync

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Respo
1	Built-In Node	Linux (amd64)	In sync	5.74 GB	0 B	5.74 GB	
2	Jenkins-slave-1	Linux (amd64)	In sync	6.12 GB	0 B	6.12 GB	
3	Jenkins-slave-2	Linux (amd64)	In sync	6.13 GB	0 B	6.13 GB	

## Step - 2 : Installing docker on slave-1 machine

```
$ sudo yum update -y
$ sudo amazon-linux-extras install docker
$ sudo yum install docker
$ sudo service docker start
$ sudo usermod -a -G docker ec2-user
```

```
Transaction test succeeded
Running transaction
  Installing : runc-1.0.0-2.amzn2.x86_64 1/5
  Installing : containerd-1.4.6-3.amzn2.x86_64 2/5
  Installing : libcgroup-0.41-21.amzn2.x86_64 3/5
  Installing : pigz-2.3.4-1.amzn2.0.1.x86_64 4/5
  Installing : docker-20.10.7-3.amzn2.x86_64 5/5
  Verifying : docker-20.10.7-3.amzn2.x86_64 1/5
  Verifying : containerd-1.4.6-3.amzn2.x86_64 2/5
  Verifying : pigz-2.3.4-1.amzn2.0.1.x86_64 3/5
  Verifying : runc-1.0.0-2.amzn2.x86_64 4/5
  Verifying : libcgroup-0.41-21.amzn2.x86_64 5/5

Installed:
  docker.x86_64 0:20.10.7-3.amzn2

Dependency Installed:
  containerd.x86_64 0:1.4.6-3.amzn2           libcgroup.x86_64 0:0.41-21.amzn2
  pigz.x86_64 0:2.3.4-1.amzn2.0.1             runc.x86_64 0:1.0.0-2.amzn2

Complete!
[ec2-user@ip-172-31-13-3 ~]$ docker --version
Docker version 20.10.7, build f0df350
[ec2-user@ip-172-31-13-3 ~]$ ]
```

```
$ sudo mkdir jenkins
$ sudo chmod 777 -R jenkins
```

## Step - 3 : CICD jenkins pipeline

Code is in the github repository : [github\\_link](#)

We are treating slave 1 as testing server and slave 2 as production server.

Configuring CICD pipeline in the way

- Configure web hook that triggers the slave-1's job.
- When the slave-1's job is successful, slave one's job is to dockerize the node js app and run that docker container on the slave-1 machine. If it is successful on post action build slave-2's job (production server) get triggered.
- Slave-2's job is to clone the repo, push the code to heroku and deploy the app on heroku, after completion of this job, it triggers mail notifications along with logs about status of the job.

### 3.1 Configuring the github web hook

Go to github repo -> settings -> select webhook -> add webhook ->

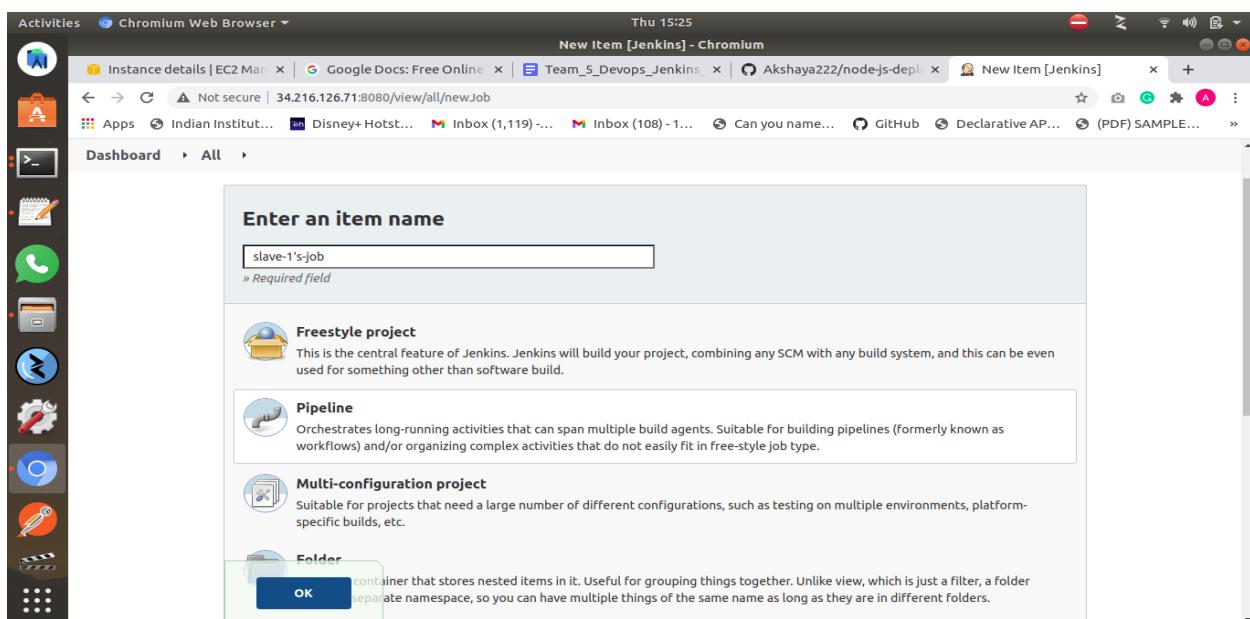
Under payload url <http://ip-address-of-jenkins-master:8080/github-webhook/>

Select push event

### 3.2 : Setting up job of slave-1

#### 3.2.1 : Creating new job

Go to jenkins dashboard -> create new job -> enter job name -> select pipeline project -> click ok



### 3.2.2 : Configuring pipeline

#### Adding gitscm polling for webhook triggers

Under build triggers -> select GitHub hook trigger for GITScm polling

So that this job will be triggered whenever we make changes in the github repo.

#### Restricting the job to be run only on slave-1

```
node(label: 'slave-1') {  
}
```

This job will now run only on the machine that has label as slave-1

#### Cloning the github repo

Creating credentials for github repo

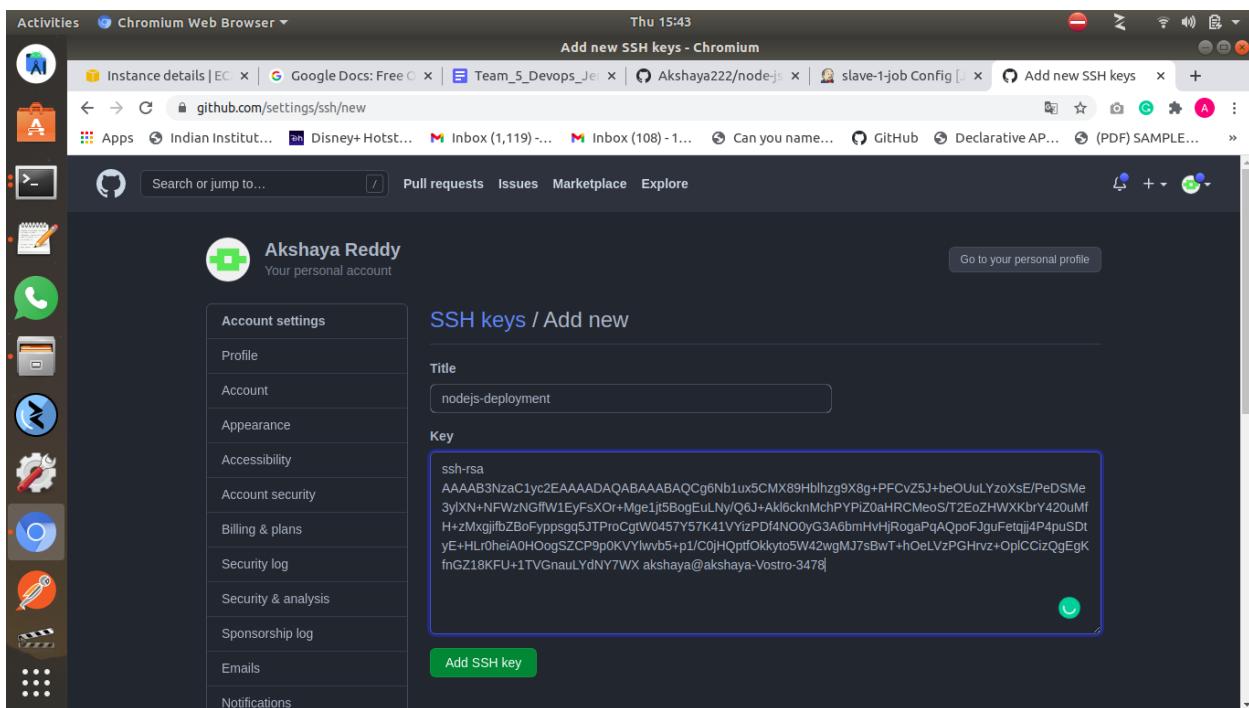
Setting up credentials in through ssh, github repo link looks like

<git@github.com:Akshaya222/node-js-deployment.git>.

Steps in creating credentials :

Create a ssh rsa key pair on local machine,

Go to github general settings -> ssh and gpg keys -> add new key -> paste the public key of the rsa key pair generated.



Key added

The screenshot shows the GitHub 'SSH keys' section for the user 'Akshaya Reddy'. On the left is a sidebar with links: Account settings, Profile, Account, Appearance, Accessibility, Account security, and Billing & plans. The main area is titled 'SSH keys' and contains a single key entry:

- nodejs-deployment**
- SHA256:4xTIMXg6NQIAA77cSC5MviUj7mudec5KpP1dmBh+ioQ
- Added on 16 Nov 2021
- Last used within the last week — Read/write
- [Delete](#)

At the bottom, there's a link to 'Check out our guide to generating SSH keys or troubleshoot common SSH problems.'

## Adding credentials on jenkins

Go to jenkins dashboard -> manage credentials -> system credentials -> add credentials ->

In kind choose ssh with private key ->

for private key give private key of ssh rsa key generated

for id ->GITHUB\_CREDENTIALS

## Adding stage for cloning the repository

```
stage("Git Clone") { 
    git credentialsId:'GITHUB_CREDENTIALS',
    url:'git@github.com:Akshaya222/node-js-deployment.git' ,branch:'main'
}
```

Run the job to test if it is working, to verify if the github repository is cloned.

When the job executes it creates a directory called workspace in root directory /home/ec2-user.

Jenkins clones the github repository into /home/ec2-user/workspace/job-name

```
[ec2-user@ip-172-31-3-68 workspace]$ cd "slave1-(testing server)"
[ec2-user@ip-172-31-3-68 slave1-(testing server)]$ ls
[ec2-user@ip-172-31-3-68 slave1-(testing server)]$ ls
[ec2-user@ip-172-31-3-68 slave1-(testing server)]$ ls
[ec2-user@ip-172-31-3-68 slave1-(testing server)]$ cd ..
[ec2-user@ip-172-31-3-68 workspace]$ ls
job-2 slave1-(testing server) slave1-test-server
[ec2-user@ip-172-31-3-68 workspace]$ cd slave1-test-server
[ec2-user@ip-172-31-3-68 slave1-test-server]$ ls
azure-pipelines.yml devopsIQ docker-compose Dockerfile test
[ec2-user@ip-172-31-3-68 slave1-test-server]$ cd devopsIQ
[ec2-user@ip-172-31-3-68 devopsIQ]$ ls
images index.html
[ec2-user@ip-172-31-3-68 devopsIQ]$ cd ..
[ec2-user@ip-172-31-3-68 slave1-test-server]$ ls
azure-pipelines.yml devopsIQ docker-compose Dockerfile test
[ec2-user@ip-172-31-3-68 slave1-test-server]$ pwd
/home/ec2-user/jenkins/workspace/slave1-test-server
[ec2-user@ip-172-31-3-68 slave1-test-server]$
```

## Adding stage for building docker image and run that in container

delete any existing docker containers , build docker container and open port 5000 for it.

```
stage("Dockerizing and deploying to test server"){
```

```
sh 'sudo docker rm -f $(sudo docker ps -a -q)'

sh 'sudo docker build /home/ec2-user/workspace/slave-1-job -t test'

sh 'sudo docker run -it -p 5000:5000 -d test'

}'
```

### Adding stage for post build action

To build slave-2's job after completing previous stages

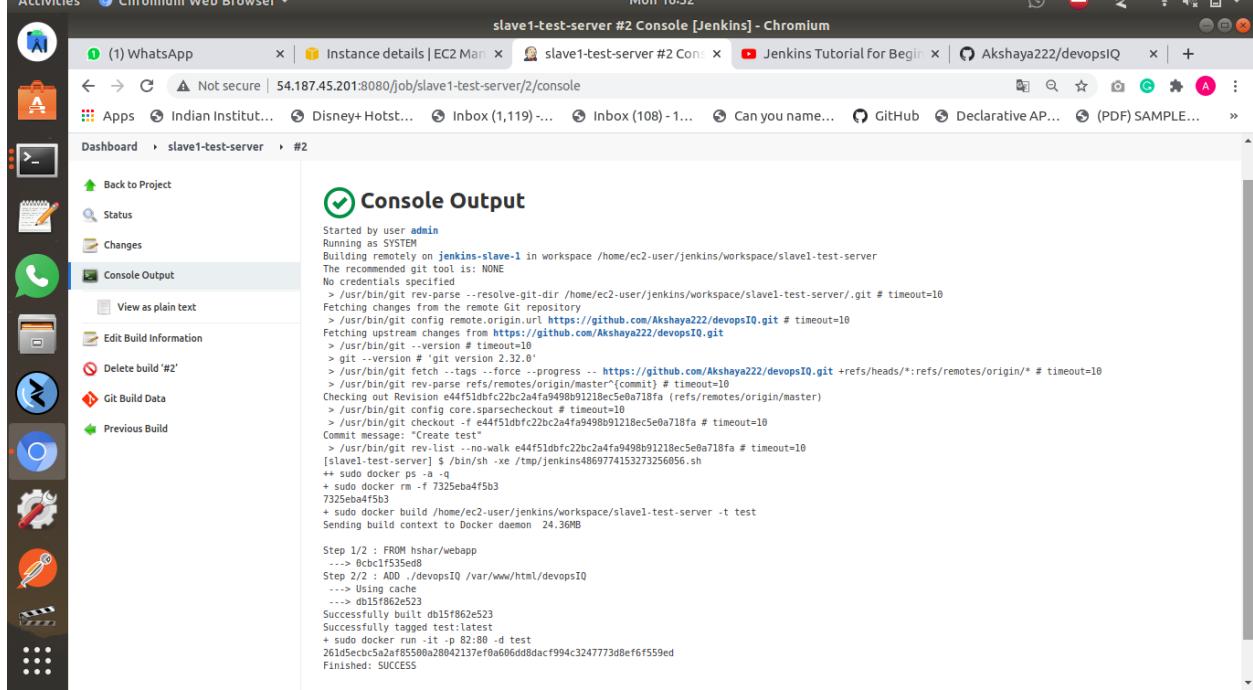
```
stage("post-build"){

    build job: 'nodejs-to-heroku'
```

```
}
```

Build the job.

To check if the build is successful



Activities Chromium Web Browser - slave-1-job [Jenkins] - Chromium

Thu 16:24

slave-1-job [Jenkins]

Instances | E... | Team\_5\_Dev | Akshaya222/ | slave-1-job | SSH and GPC | Photo Editor | (1) WhatsApp | E-commerce | +

Not secure | 34.216.126.71:8080/job/slave-1-job/

Apps Indian Institut... Disney+ Host... Inbox (1,119) ... Inbox (108)-1... Can you name... GitHub Declarative AP... (PDF) SAMPLE... »

Dashboard > slave-1-job >

Recent Changes

Stage View

	Git Clone	Dockerizing and deploying to test server	post-build
#29 Nov 18 16:14 No Changes	2s	16s	30s
#28 Nov 18 12:23 1 commit	1s	17s	26s
#27 Nov 18 11:54 1 commit	1s	16s	27s

Average stage times: (Average full run time: ~3min 2s)

Build History trend

Filter builds...

#29 Nov 18, 2021, 10:44 AM

#28 Nov 18, 2021, 6:53 AM

#27 Nov 18, 2021, 6:53 AM

To check if build is successfully deployed on testing server (on slave -1)

Open custom port 5000 on slave-1 instance

Go to browser and hit [http://ipaddress\\_of\\_slave-1:5000](http://ipaddress_of_slave-1:5000)

Activities Chromium Web Browser - E-commerce - Chromium

Thu 16:22

Instances | E... | Team\_5\_Dev | Akshaya222/ | slave-1-job | SSH and GPC | Photo Editor | WhatsApp | E-commerce | +

Not secure | 18.236.167.88:5000

Apps Indian Institut... Disney+ Host... Inbox (1,119) ... Inbox (108)-1... Can you name... GitHub Declarative AP... (PDF) SAMPLE... »

Home Products Testimonials Services Contact us

Waiting for 18.236.167.88...

### 3.3 : Setting up job of slave-2

Go to jenkins dashboard -> create new job -> enter job name -> select freestyle project -> click ok

Under project url give github repository url

<https://github.com/Akshaya222/node-js-deployment.git/>

Under Restrict where this project can be run select slave-2

#### Creating heroku credentials

For heroku repo create ssh rsa key pair on local system using the command

```
$ ssh-keygen rsa -t
```



```
akshaya@akshaya-Vostro-3478:~/.ssh$ ls
github_rsa      heroku_keys      id_rsa      known_hosts
github_rsa.pub   heroku_keys.pub  id_rsa.pub
akshaya@akshaya-Vostro-3478:~/.ssh$ cat github_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAQABAAQABQcgGNb1ux5CMX89hbhlgzg9X8g+PFCvZ5J+beOUuLYzoXsE/PeDSMe3ylXN+NFWzNGffW1EyFsX0r+Mge1jt5BogEuLNy/Q6J+Akl6cknM
chpYPlZoahRCMeoS/T2eoZXNXKbrY420uMfh+ZlXqjlfbzB0fyppsq5JTProCgtW457V57K41VV1zPf4N00yG3A6bmHvHjRogaPqA0poFJguFetqj4P4puS0tyE+Hlr0heiA0H0ogS
ZCP9p0KVYlwvb5+p1/C0jHQptfokkyto5W42wgM77sBwT+h0eLVzPGHrvz+OplCC1zQgEgKFngZ18KFU+1TVGnaULYdNY7WX akshaya@akshaya-Vostro-3478
akshaya@akshaya-Vostro-3478:~/.ssh$
```

```
$ heroku add keys
```

And select the key pair generated for heroku.

Go to jenkins dashboard -> manage credentials -> system credentials -> add credentials ->

In kind choose ssh with private key ->

for private key give private key of ssh rsa key generated for heroku

for id ->HEROKU\_CREDENTIALS

#### Job configuration

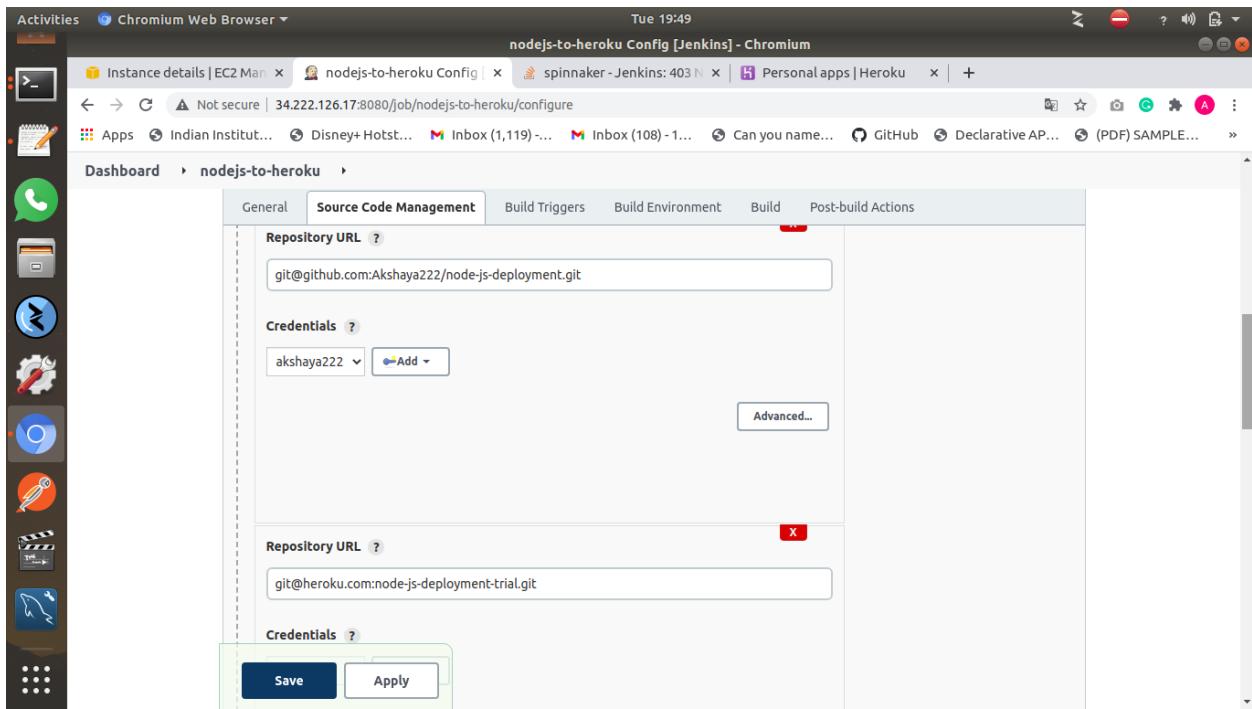
Under git -> click add

Add repository details of github and repository of heroku details

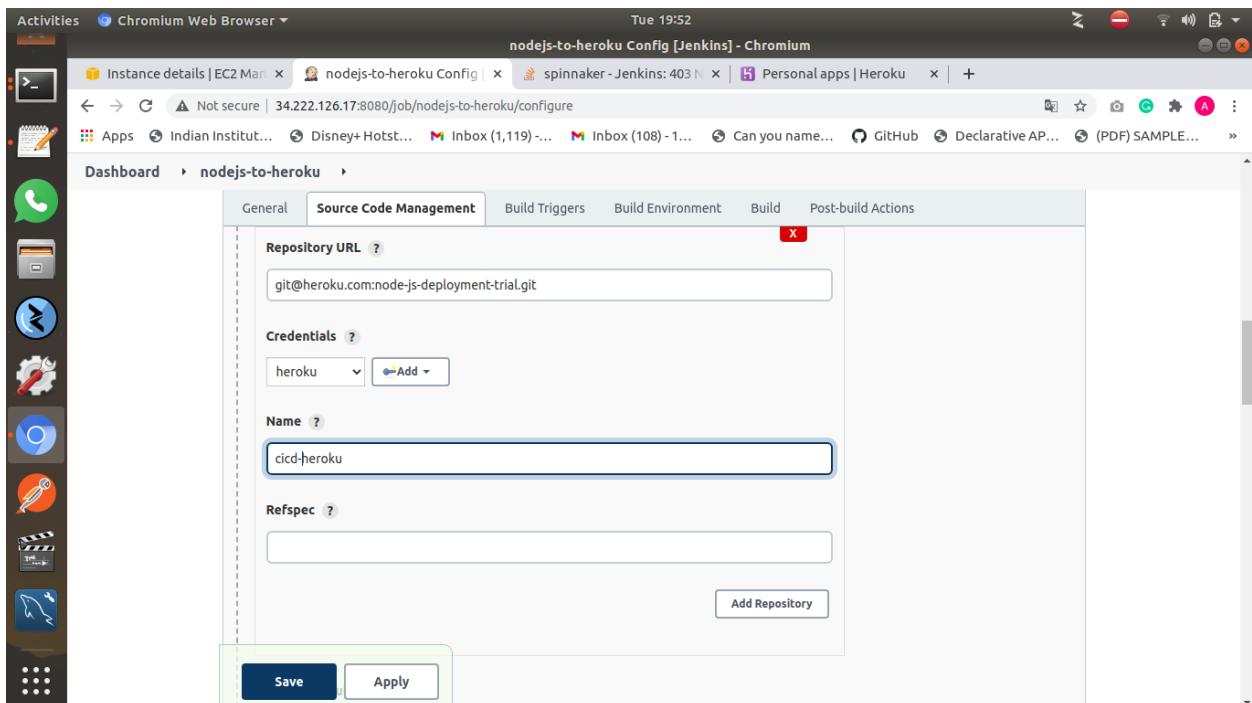
Add ssh urls for both,

For github repo credentials select the ones created during slave-1's job,

For heroku repo choose the credentials with id HEROKU\_CREDENTIALS



While adding heroku repo details under name add “cicd pipeline”,



To deploy the app cloned from first repo to heroku

Under post build actions -> git publisher -> push only if build succeeds

For Branch to push - main

For Target remote name - cicd-pipeline (name created while adding heroku repos)

Run the build once and check if the changes have been deployed to heroku

On heroku

The screenshot shows the 'Latest activity' section of a Heroku application's dashboard. It lists several events:

- 18bcs017@iitdwd.ac.in: Deployed 2d0c46ca Just now · v4
- 18bcs017@iitdwd.ac.in: Build succeeded Just now · [View build log](#)
- 18bcs017@iitdwd.ac.in: Deployed 4ce3abdc Today at 6:25 PM · v3
- 18bcs017@iitdwd.ac.in: Build succeeded Today at 6:25 PM · [View build log](#)
- 18bcs017@iitdwd.ac.in: Enable Logplex Today at 6:21 PM · v2

A blue oval highlights the first two items, which correspond to the most recent deployment and its successful build.

### 3.4 : Adding mail notifications

#### 3.4.1 : install the plugins

Go to jenkins dashboard -> manage jenkins -> manage plugins -> available ->

Mail extension -> install without restart.

#### 3.4.2 : Configuration

Go to jenkins dashboard -> system configuration

Under Extended E-mail Notification and E-mail Notification

Smtp server - smtp.gmail.com

Smtp port - 465

Enable use ssl

Smtp username and password - gmail username and password

Select enable debug mode

Enable less secure apps for the gmail account

#### 3.4.3: Using it with a job

Go to configuration of slave-2 job -> post build actions -> Editable email notification ->

Project Recipient List - mail id of people to whom the mail should go

Under Attach Build Log -> attach build log

The screenshot shows the Jenkins interface for the 'nodejs-to-heroku' job. In the 'Post-build Actions' tab, there is a configuration for an 'Editable Email Notification'. Under 'Project Recipient List', the value 'akshayareddy2211@gmail.com' is entered. A tooltip below the input field specifies that it is a 'Comma-separated list of email address that should receive notifications for this project.' At the bottom of the configuration panel are 'Save' and 'Apply' buttons.

The screenshot shows the Jenkins interface for the 'nodejs-to-heroku' job. In the 'Post-build Actions' tab, the 'Content Token Reference' dropdown is set to 'Attach Build Log'. Other options visible in the dropdown include 'Advanced Settings...', 'Save', and 'Apply' buttons.

### 3.4.1 : Setting up the pipeline view

Installing build pipeline for enable pipeline view

Go to jenkins dashboard -> manage jenkins -> manage plugins -> available -> search “build pipeline” ->

Select “install without restart”

The screenshot shows the Jenkins Update Center interface. On the left, there's a sidebar with links like 'Back to Dashboard', 'Manage Jenkins', and 'Manage Plugins'. The main content area is titled 'Installing Plugins/Upgrades'. It shows a list of available plugins under 'Available' and their current status. The 'Build Pipeline' plugin is specifically highlighted with a yellow circle around its status icon. Other plugins listed include Javadoc, Maven Integration, Conditional BuildStep, Parameterized Trigger, jQuery, Build Pipeline, and Loading plugin extensions, all marked as 'Success'.

After installation, go back to the dashboard and click on “+”.

The screenshot shows the Jenkins Dashboard. On the left, there's a sidebar with various management links. The main area is titled 'New View' and features a table for creating new views. The first row has columns for 'Name' (Prod), 'Last Success' (6 min 54 sec - #4), 'Last Failure' (9 min 6 sec - #2), and 'Last Duration' (2.2 sec). The second row has columns for 'Name' (Test), 'Last Success' (20 min - #9), 'Last Failure' (20 min - #8), and 'Last Duration' (25 sec). At the top of the 'New View' form, there's a large '+' button, which is circled in yellow. Below the table, there are icons for 'S' (Stable), 'M' (Medium), and 'L' (Long), and links for 'Atom feed for all', 'Atom feed for failures', and 'Atom feed for just latest builds'.

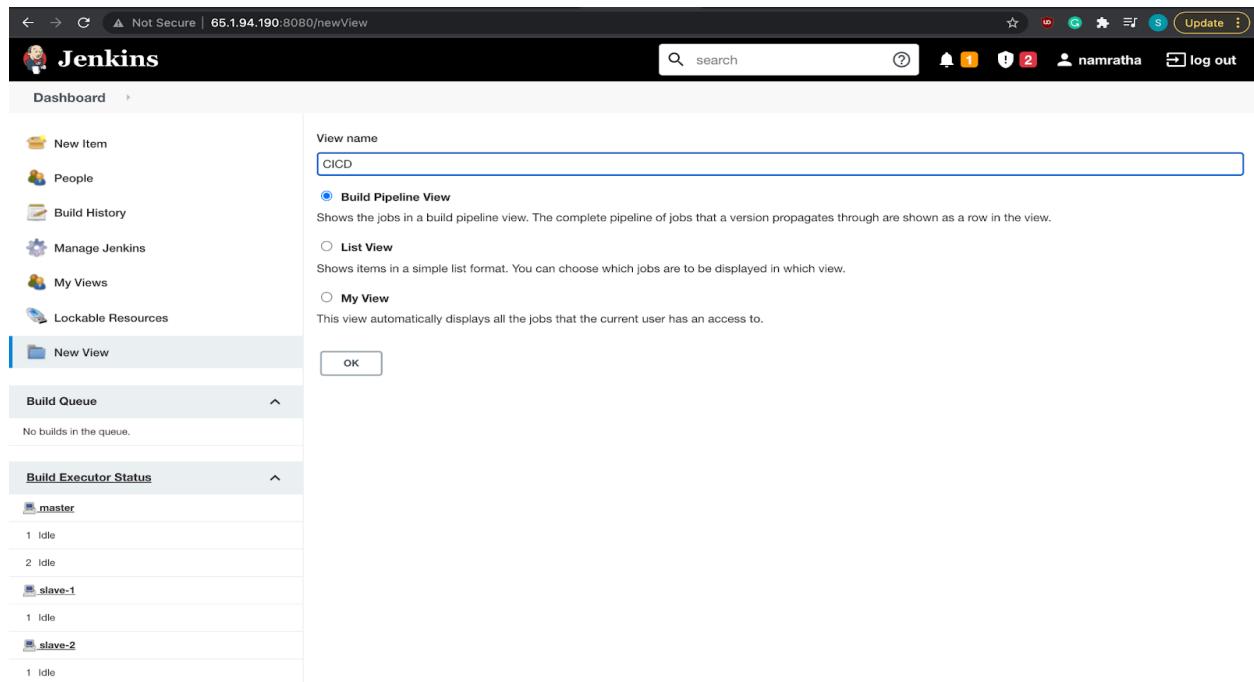
Select build pipeline view and name the view name as CICD.

Enter the name- CICD

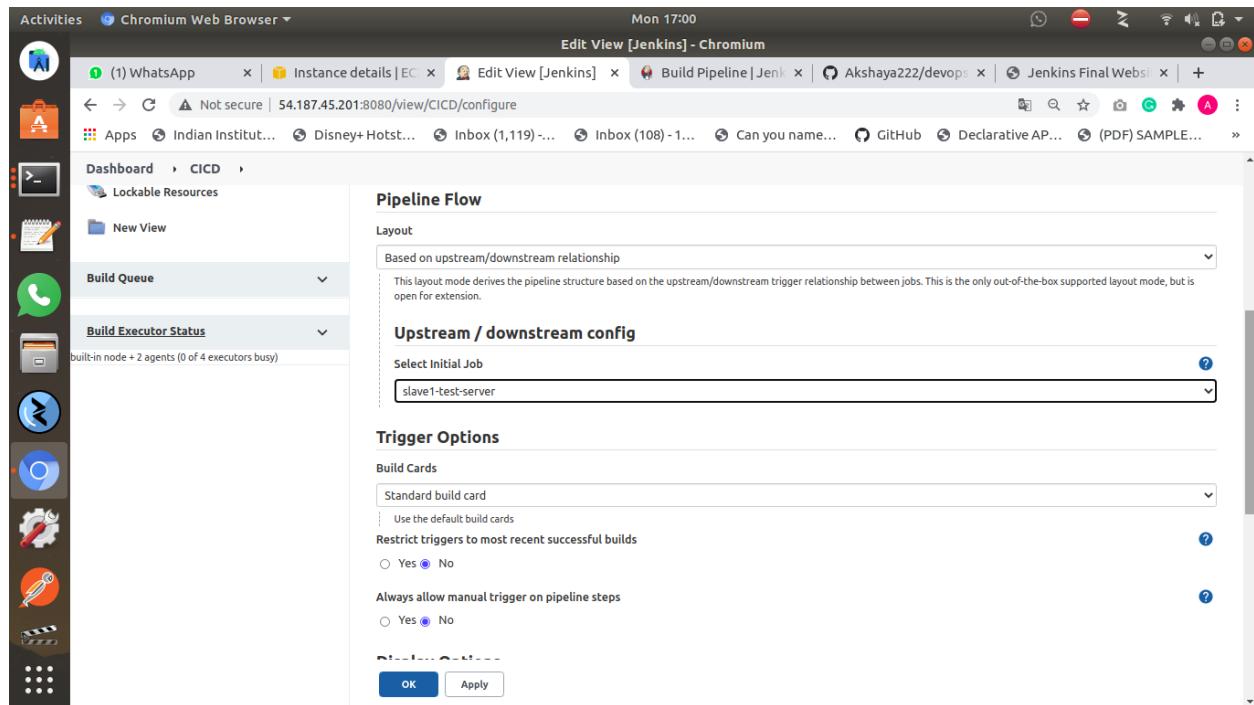
Build pipeline view title- CICD.

Select initial job - slave-1-job

Click Ok



Under pipeline options -> under select initial job -> select testing job -> click on apply -> click ok.





To trigger the pipeline we need to commit new changes and push the code to github.

When the new push event occurs on github, the slave-1's job automatically gets triggered with the help of webhook.

Go to pipeline view to see the jobs status

The screenshot shows a Chromium browser window with multiple tabs open. The active tab is 'CICD [Jenkins] - Chromium' displaying the Jenkins interface. The main content area is titled 'Build Pipeline: CICD'. It shows two pipeline items: '#3 slave1-test-server' and '#2 slave2-production-server'. Both items are shown in green boxes, indicating they have run successfully. The Jenkins sidebar on the left shows various management icons like Pipeline, Configuration, and Monitoring.

Both the jobs ran successfully.

**To verify if the changes have been deployed on slave-1's job( testing server )**

Go to browser and hit [http://ipaddress\\_of\\_slave-1:5000](http://ipaddress_of_slave-1:5000)

**To verify if the changes have been deployed on slave-2's job( on HEROKU SERVER)**

Go to heroku

The screenshot shows the Jenkins 'Latest activity' feed. A blue oval highlights the first two items:

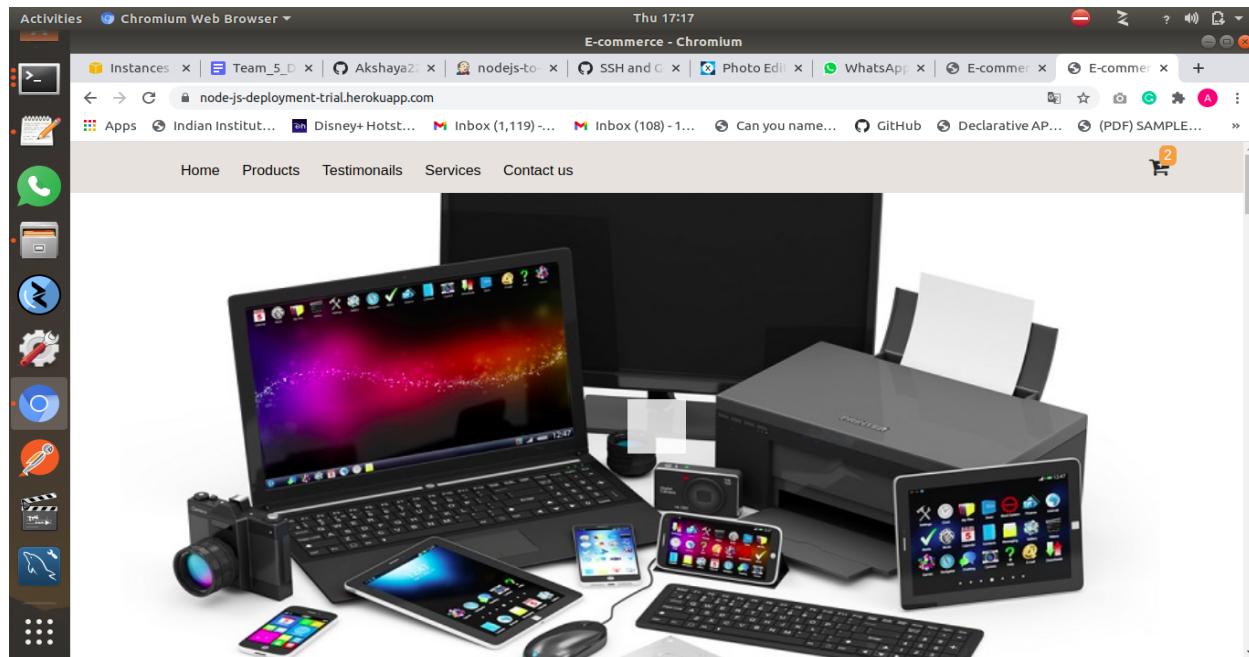
- 18bcs017@iitdwd.ac.in: Deployed [2d0c401a]** Just now · v4
- 18bcs017@iitdwd.ac.in: Build succeeded** Just now · [View build log](#)

Below these, other activity items are listed:

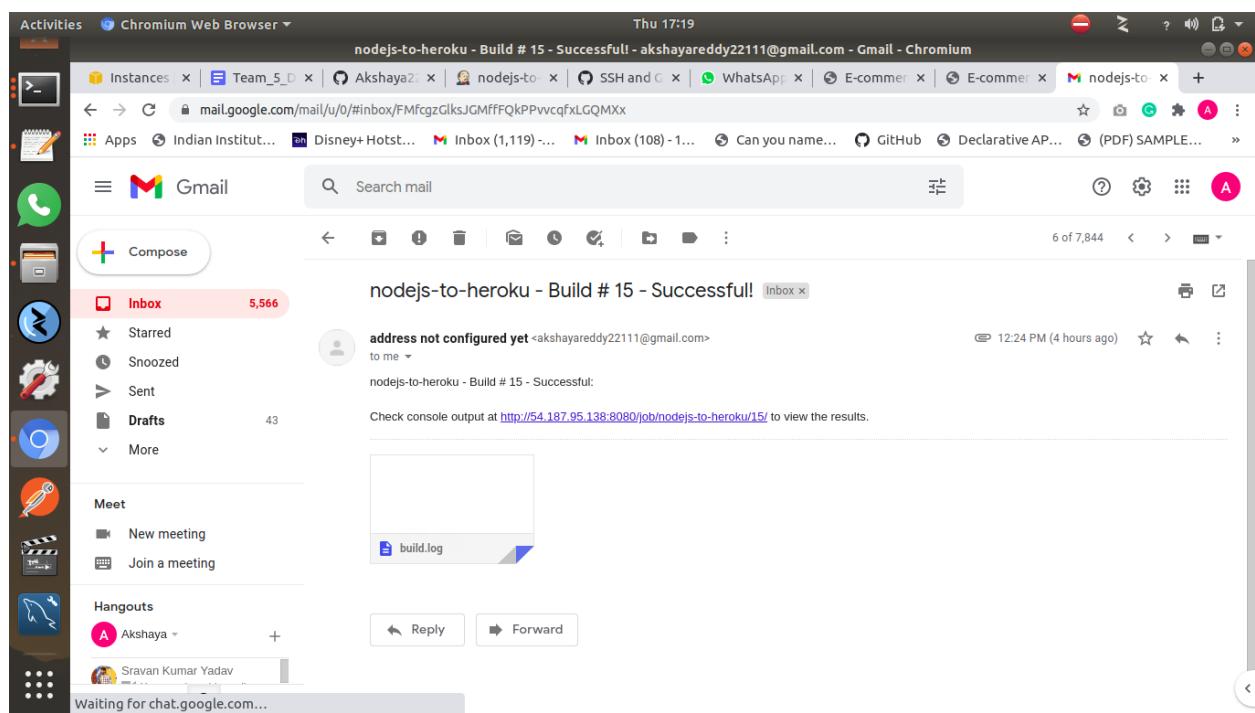
- 18bcs017@iitdwd.ac.in: Deployed [4ce3a8dc]** Today at 6:25 PM · v3
- 18bcs017@iitdwd.ac.in: Build succeeded** Today at 6:25 PM · [View build log](#)
- 18bcs017@iitdwd.ac.in: Enable Logplex** Today at 6:21 PM · v2

And

Go to browser and hit <https://node-js-deployment-trial.herokuapp.com/>



Production server job has been completed and jenkins will notify about the status of the job through email



Jenkins master slave architecture has been accomplished.