

DevOps Jenkins Assignment

Submitted to:

Dr. Uma S

Submitted by:

Team 8

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
clusters:
- cluster:
  certificate-authority-data: LS0tLS1CRDgtTlBDRVJUSUZJQ0FURS0tLS0tck1JSUVMskNDQWVnZ0f35UBz0lCQBRTkJna3f0a21H0XcwK0FRC0zBREFTNTVJnd0vRwURNUW
FERXDRmRSwXyK201bgHRVnpVbJYRFReje1URXHNRRE1TkrM5ewB1hEVE14TrVRfd9e0qEVOrGsw1TzYd0zURvRNQvKHFTRVQrpBt1LYTNwAvpYsvNaFj3y3pQ00FTSx
SwHz2Q0CQV1LFBgRnbVRBRENDQvP2qdnRuTBGncQkrEJkfQvd1ErNtQz2MBeUxDzHeGyNz7K9zBkCnRsrOsU5TzhJuE42a0JUHMhLMewK0PrvSHASU5
RnFv044ktJvBkgpVraTzFzQd0JtlDzvTNTbVh1J19hMcU0VylMyccTcGJ0HMLTz2wp1W1RQN3v2UWvHTVvzuodtuhDzTj19NzBc2hdH0D50NDTpudb0vUgbyfpTwYR
N3zQ1QzUbo2hXb0MzD1lbgMtzQ1z0LQmrGRQzJG5k9sHVNhK2ia1mDfNfMsZTfHbnv1WxPbmPMVdysSxLRFzWQfZ2BkTfStsTmaHdK1NrCrdG5r5UtdHfJ0eH
ZTQTAuHnJmoHmzHzb2p30GNvUHYZHJUFRcw1E5k0sydErUpwkytxYwZJxM3x0zQxyUFeVFnQ03RUFvByUsTzJd0rnuWmBzQ0QfJpL0JBU8zB2r0tTUE4R0exwRfFdCv9c3
UuZQnH0J0QyDw1mHmzHzb2p30GNvUHYZHJUFRcw1E5k0sydErUpwkytxYwZJxM3x0zQxyUFeVFnQ03RUFvByUsTzJd0rnuWmBzQ0QfJpL0JBU8zB2r0tTUE4R0exwRfFdCv9c3
JBGRs5aXQRX9y5L8Xk4T3vWBApp025EMhdauE9MnPxd0JGMnozykZKeGPyw4xZ25hNkzNzDgdk1BLStLhMa2rQvRykhMssvbtFjd1RtLeK2ZMgxwCmPdThqecyvCv
RKJ5s5aXQRX9y5L8Xk4T3vWBApp025EMhdauE9MnPxd0JGMnozykZKeGPyw4xZ25hNkzNzDgdk1BLStLhMa2rQvRykhMssvbtFjd1RtLeK2ZMgxwCmPdThqecyvCv
Q3P0plKzVCTfdav6y8v4Z3nxQjUpx4vbpvaNh1cf2sUVLwHvhkdkYktlaerWx4zpxJhvL63v5PvnDpx20nNwak03c51Qmhrsl5fbEjtGdj2ctMu2hJmfDQk2dWmD13L3BjQxNpnuV
VxVRMwLStsy8v4Thul1zQzQ1zUYxF0XMKH0F4Pq0tLS0tluUvORcBDRVJUSUZJQ0FURS0tLS0tck1
server: https://172.31.27.210:6443
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: LS0tLS1CRDgtTlBDRVJUSUZJQ0FURS0tLS0tck1JSUVMskNDQWVnZ0f35UBz0lCQBRTkJna3f0a21H0XcwK0FRC0zBREFTNTVJnd0vRwURNUW
U3TfPpRwUvXQ0XN9zE2vNv1ahEp1WhsbGn6WQdZB5TvrFe1U0QxdPvFe1TkrHwYzU3MhtNaV4TrVrbd9e0u1tVOrCxhTURReapGeKfW0md0kv3b1Reb41yN5bcJUchRzWe4wh1hEk1R
Sa3Gd11EvLFRER4vE0Jk1vKpsP201bgHRVnpVb0JCrnKjhNSnSULCSpBtkJna3f0a21H0XcwK0FRC0zBREFTNTVJnd0vRwURNUW
Hyy3NxkaYq3y0lpu0qjNfhsJNzDchksaLzK12wv5a2VtN9yxiS9y3zK3jVtlyenhtVxLcOpwybzDhvnQwklPvgZws2o3djh
xbtyts010zRhzKJtrN0tr030UPRVRNtWjd0GdNPNXfxwZpFvTg1z2t3z4ExEmxaGpuGmsr0ReeksFsuUvhTm1TxyXKU2d1TzdKogFkYlpmFvHd2vWruXbFhCzL
0EyedhX0Hn2ZHzBQ0u33ZfUpuyXk5N1RL4Qs3R0NybvRCE1obgplQVlCr2v1Y1VuZicvbmppvHJYhrnVhZmShL2FstG5cp0pej1VscBu
jD0fTpxyXwvXwxdw0pVkrh0BzJhBfJhPfnQ0j9hQxfD1leV1wB3d3d21J53d2jQRVUgQkOxdjdrBwMURwJuBqvF1L0JB5xdReEfNm
WM0cnzCzkVmTuFvemZRxxJcwp0ReF0QndrCwhbAc5sdzbCpVzRkT0BNUvBUUHvTvtvTbUvSXznsjEvFAu1uNvSwrn1zL2WkpsRefZvMzn
U1i93vFp2zR2UcW11WzZ2Utd1uywM2bHmBwSzAs2qgD3cvgUVMKccMvMr4JrjwUs0vH0nzbnpUvCn1bjhMmNcrWlJmL0q5dJnt1t1um2la
Qp15G0u0N1nk14Xhnbl1ra0amxp0FvFtDxTzLe1uN3NkxeWtFndhazxrDzQ0x2d1zatuWouir1Vj1VvdzduhQmrLtzdsDzv3u1vPmb
File Edit View Search Terminal Help
ubuntu@ip-172-31-27-210: ~/.kube$
```

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:

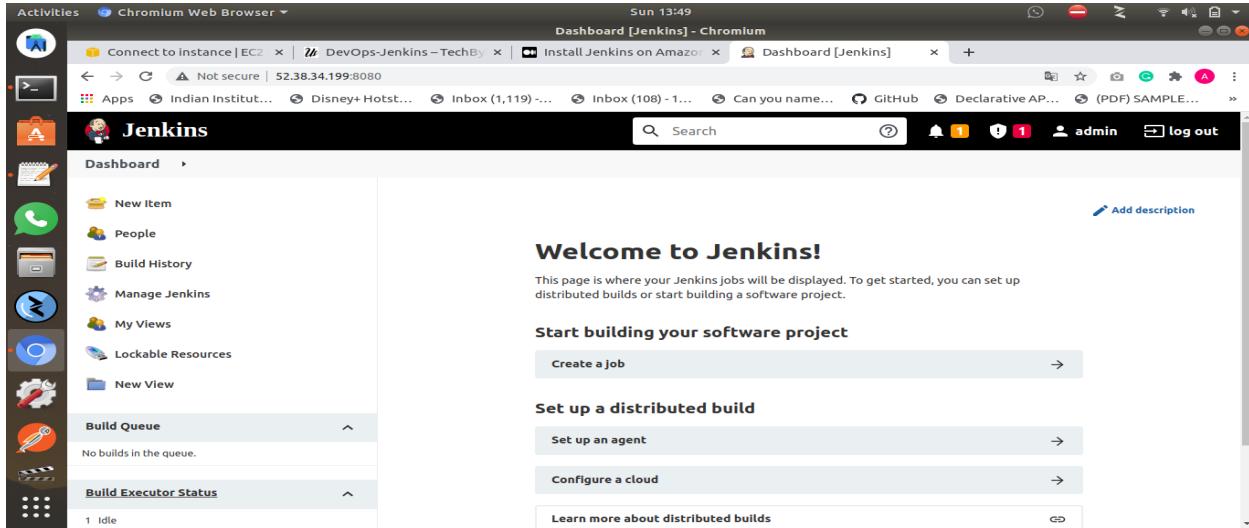
Confirm password:

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"
```

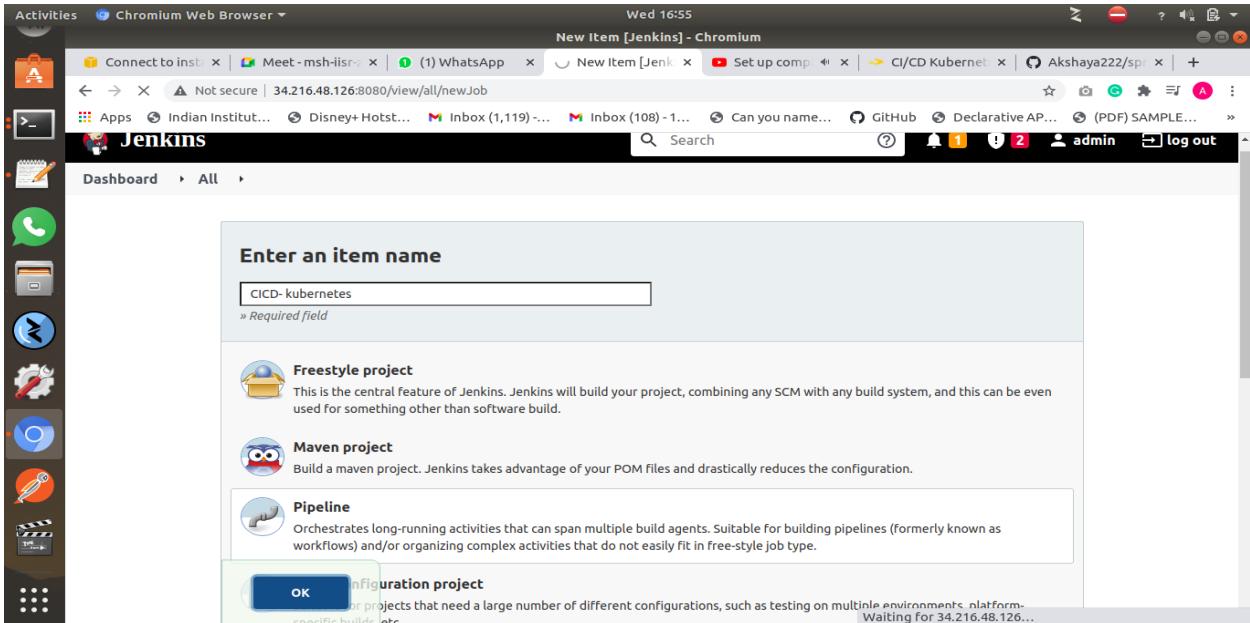
Deployment.yml file is present in the github repo.

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

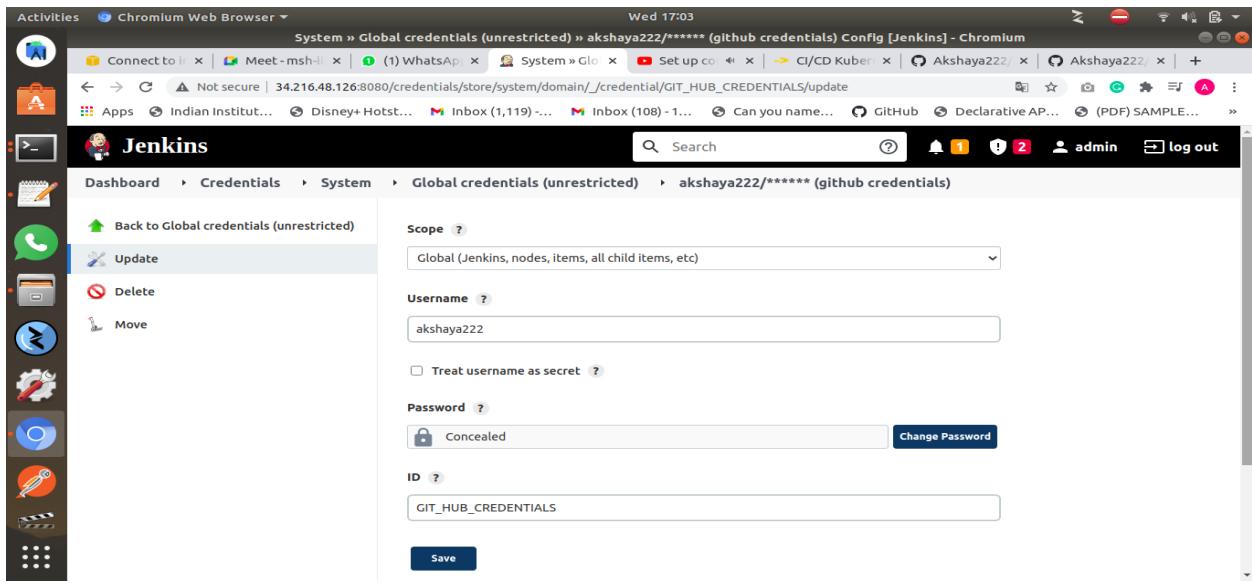
Select username and password from select menu

Scope : select global scope

For username : Github username

For password : Github Personal Access Token (PAT)

For ID : GIT_HUB_CREDENTIALS

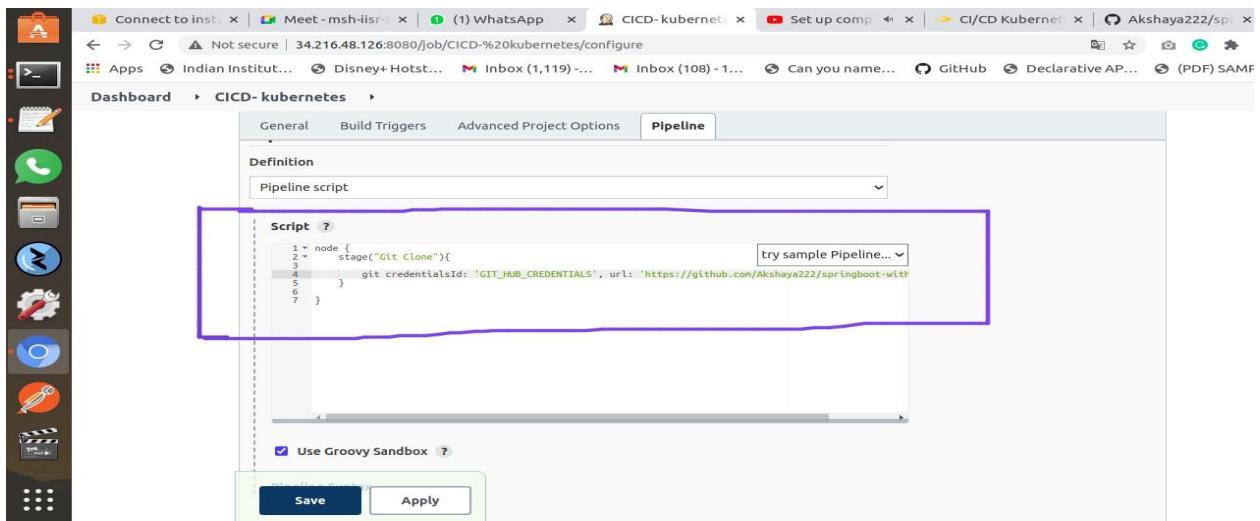


4.2.2 : adding stage in pipeline

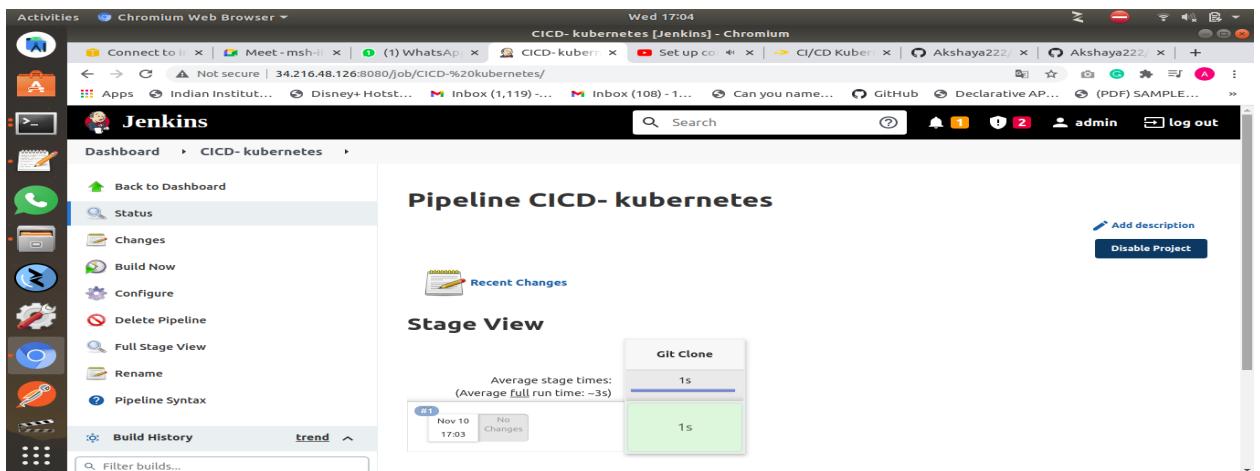
git credentialsId: Id of the github credentials- GIT_HUB_CREDENTIALS is the id of credentials created just before this

url : url of the github repository

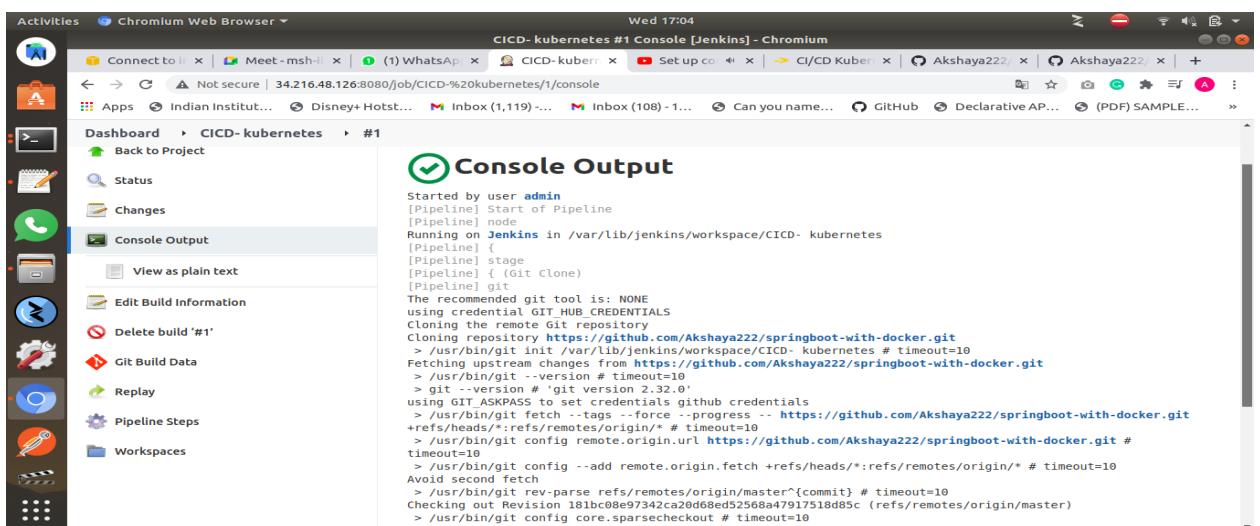
```
node {
  stage("Git Clone") {
    git credentialsId: 'GIT_HUB_CREDENTIALS',
    url:'https://github.com/Akshaya222/springboot-with-docker.git'
  }
}
```



Build status of first stage



Logs of the build:

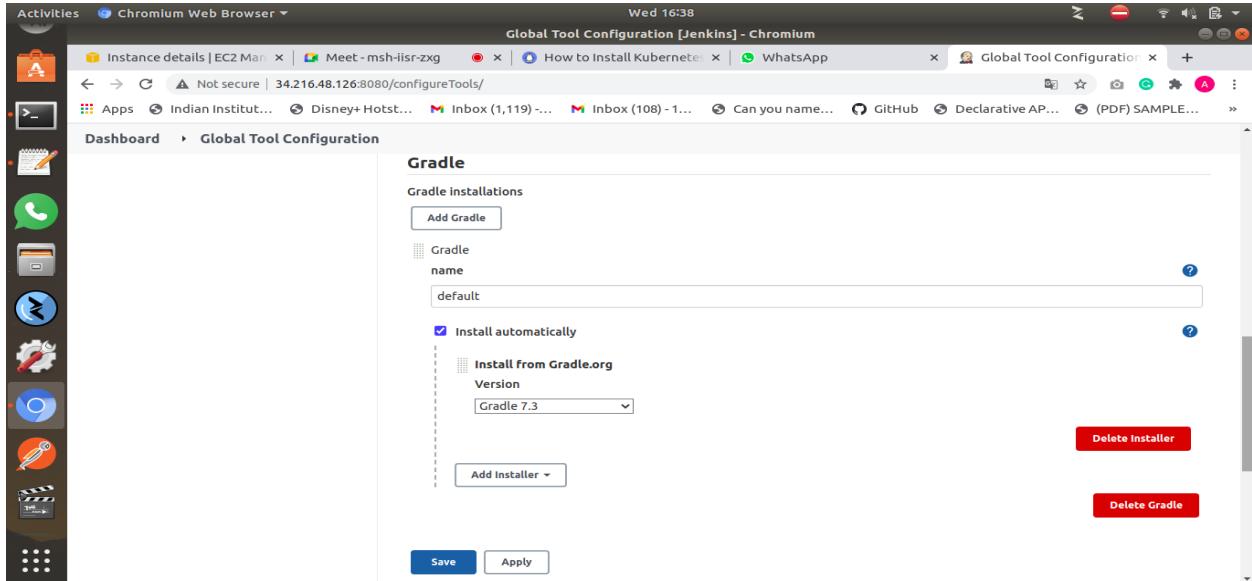


4.3 : Building the spring boot application using gradle

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



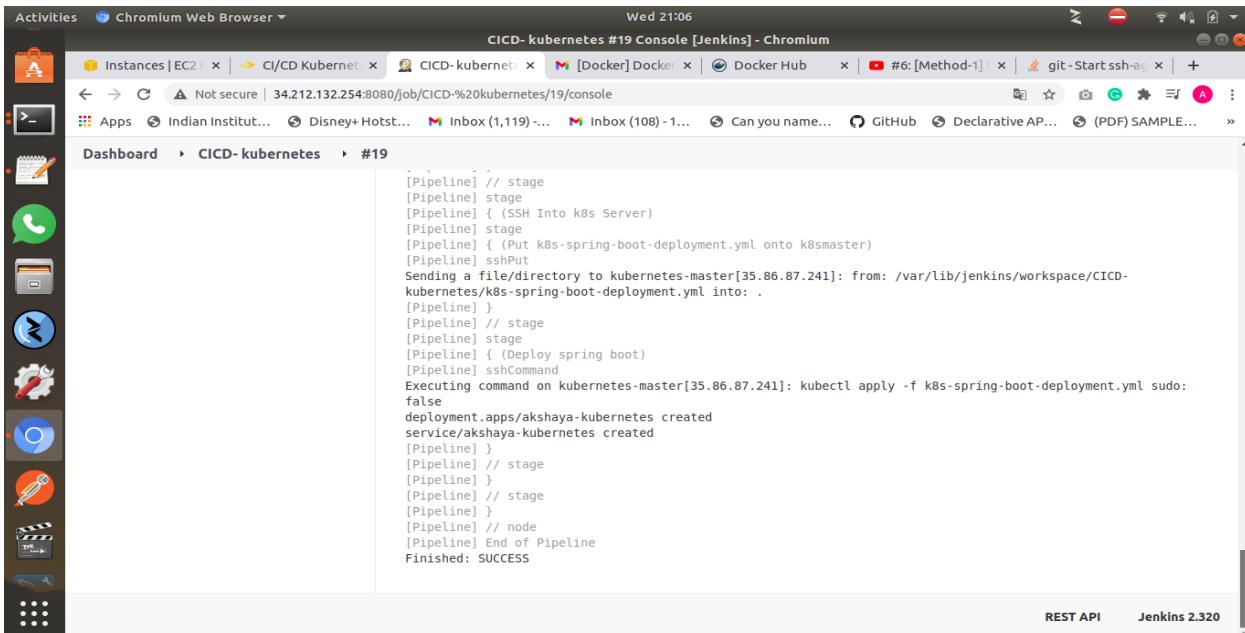
```
Last login: Wed Nov 10 15:00:46 2021 from 103.217.239.132
ubuntu@ip-172-31-27-210:~$ kubectl get nodes
NAME           STATUS    ROLES      AGE     VERSION
ip-172-31-16-106  NotReady  worker   5h6m    v1.22.3
ip-172-31-27-210  Ready     control-plane,master 5h45m   v1.22.3
ubuntu@ip-172-31-27-210:~$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

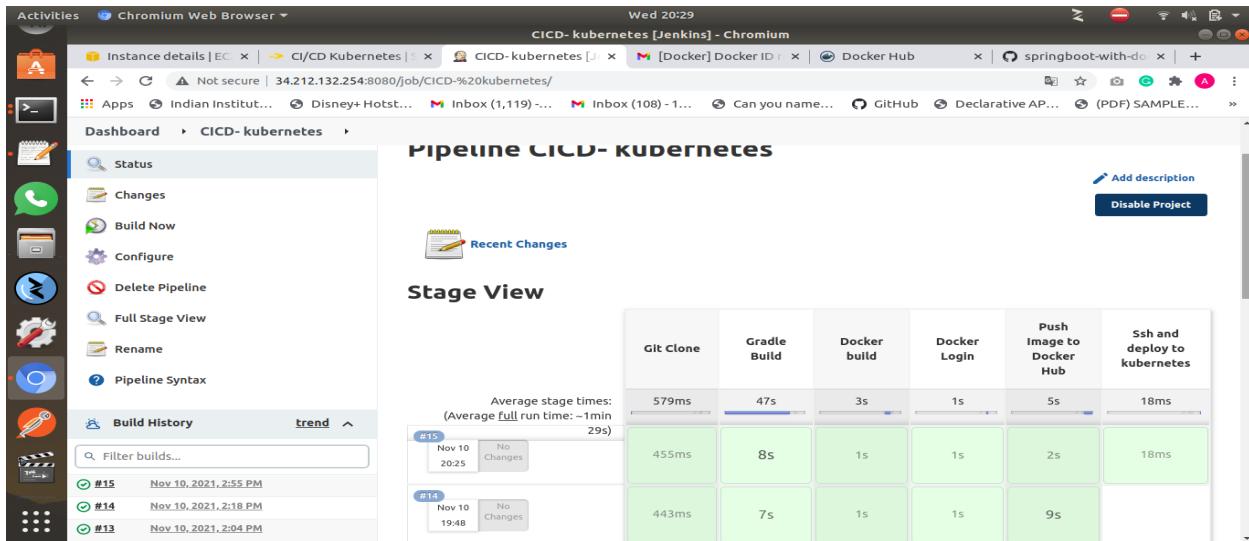


The Jenkins pipeline log shows the execution of a script to deploy a Spring Boot application to a Kubernetes cluster. The log output is as follows:

```
[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

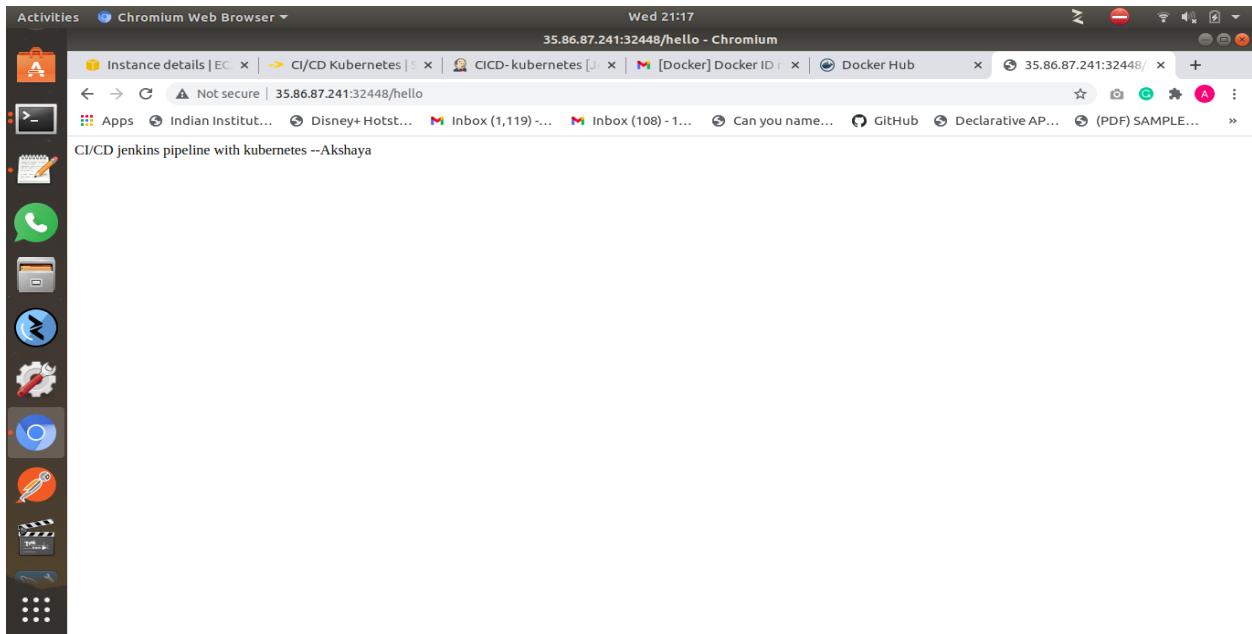
```
$ kubectl get service
```

```
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

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

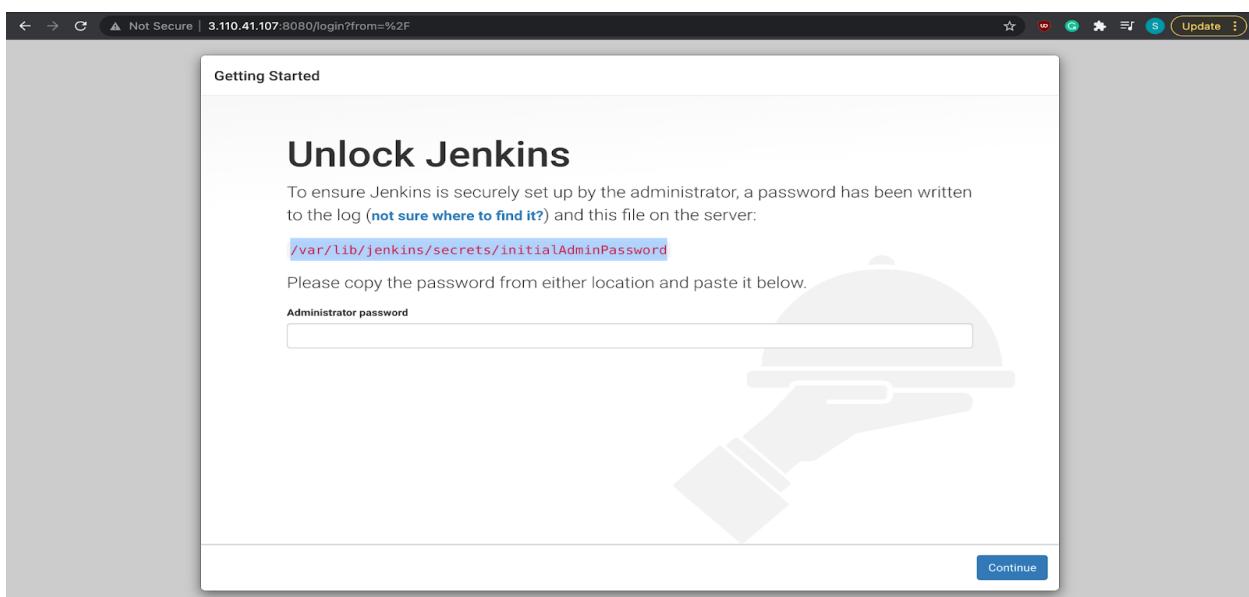
```
$ sudo yum update -y
$ sudo yum install java-1.8.0
$ sudo wget -O /etc/yum.repos.d/jenkins.repo
http://pkg.jenkins-ci.org/redhat/jenkins.repo
$ sudo rpm - import https://pkg.jenkins.io/redhat/jenkins.io.key
$ amazon-linux-extras install epel -y
$ sudo yum install jenkins -y
$ sudo service jenkins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



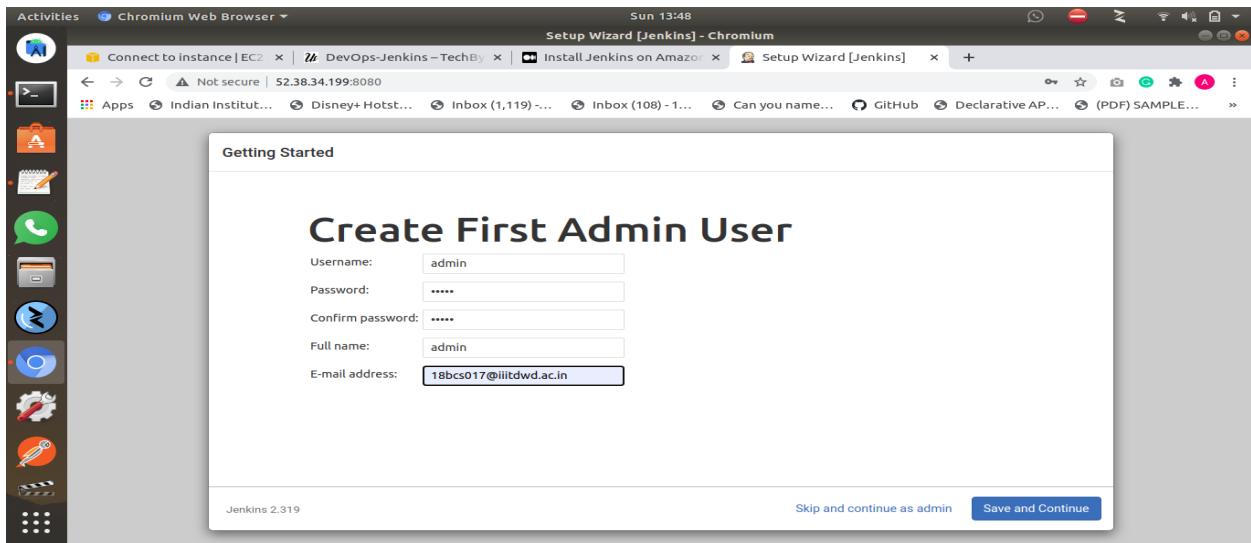
We need to provide Default jenkins(initialAdminPassword) administrator password, which is present at given path (in the page)

\$ cat given-path For example

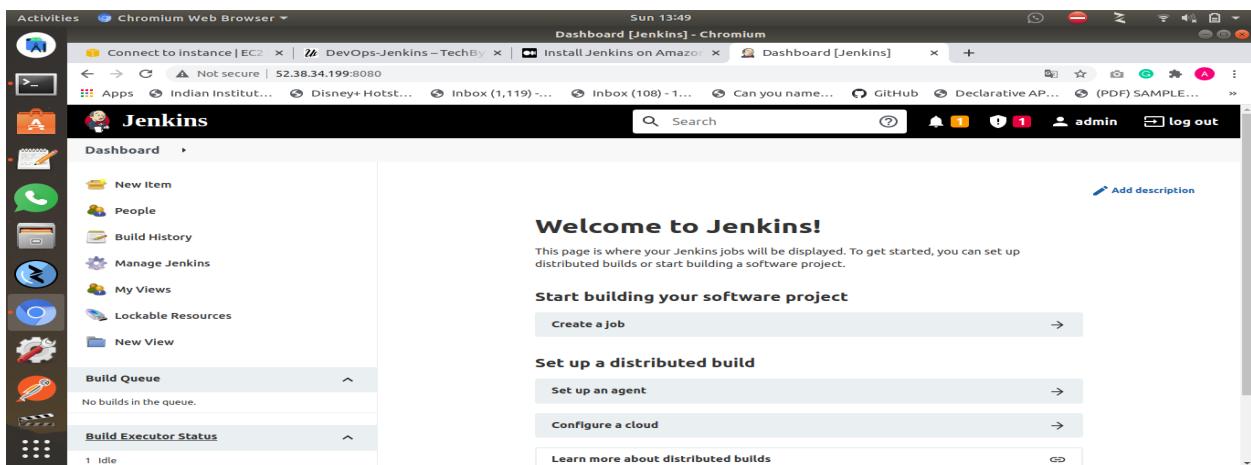
```
$ cat /var/lib/jenkins/secrets/initialAdminPassword
```

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 set 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 'slave-1' and the 'Permanent Agent' checkbox is selected. A descriptive note about permanent agents is visible. At the bottom of the dialog is an 'OK' button.

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

1 idle

Remote root directory (Required)

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

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

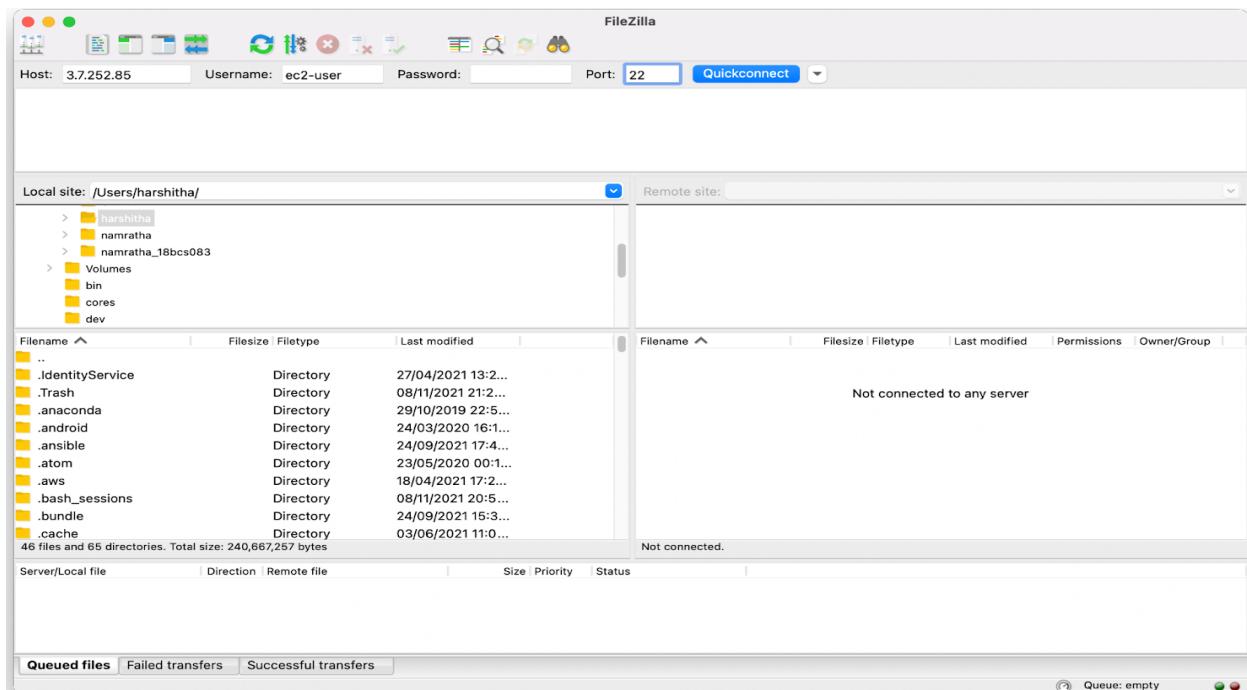
REST API Jenkins 2.303.3

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

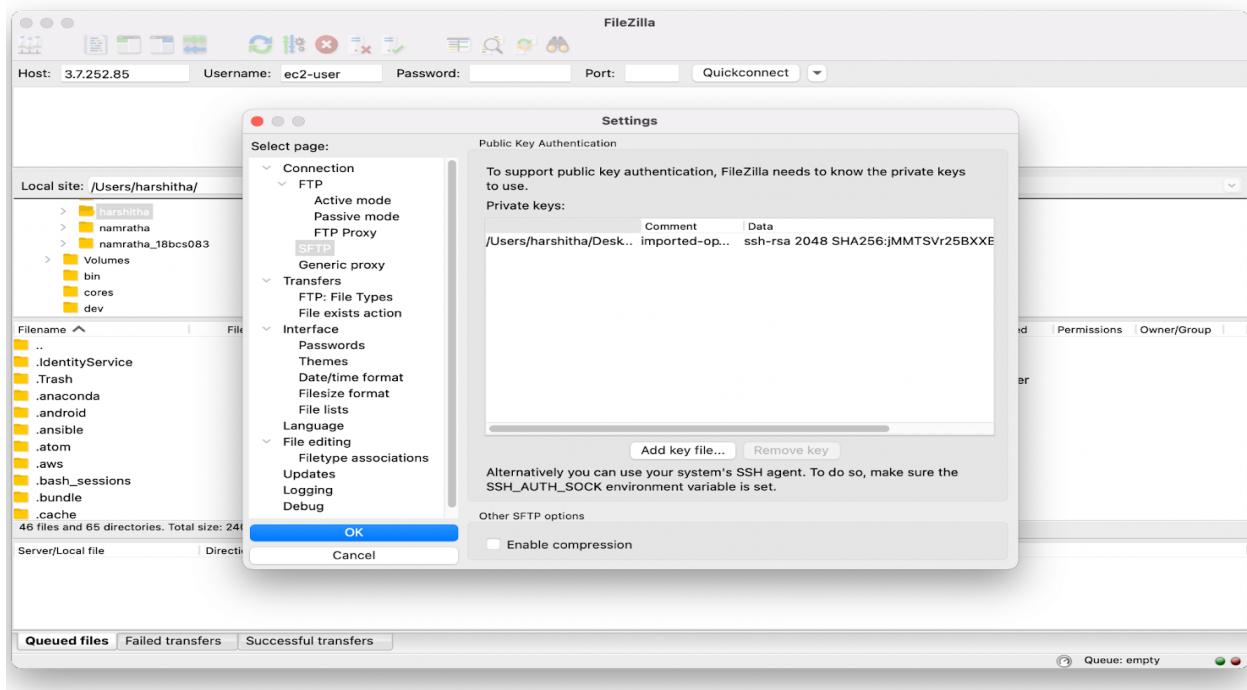
Go to slave1 EC2 instance and select public ip address.

Open filezilla.

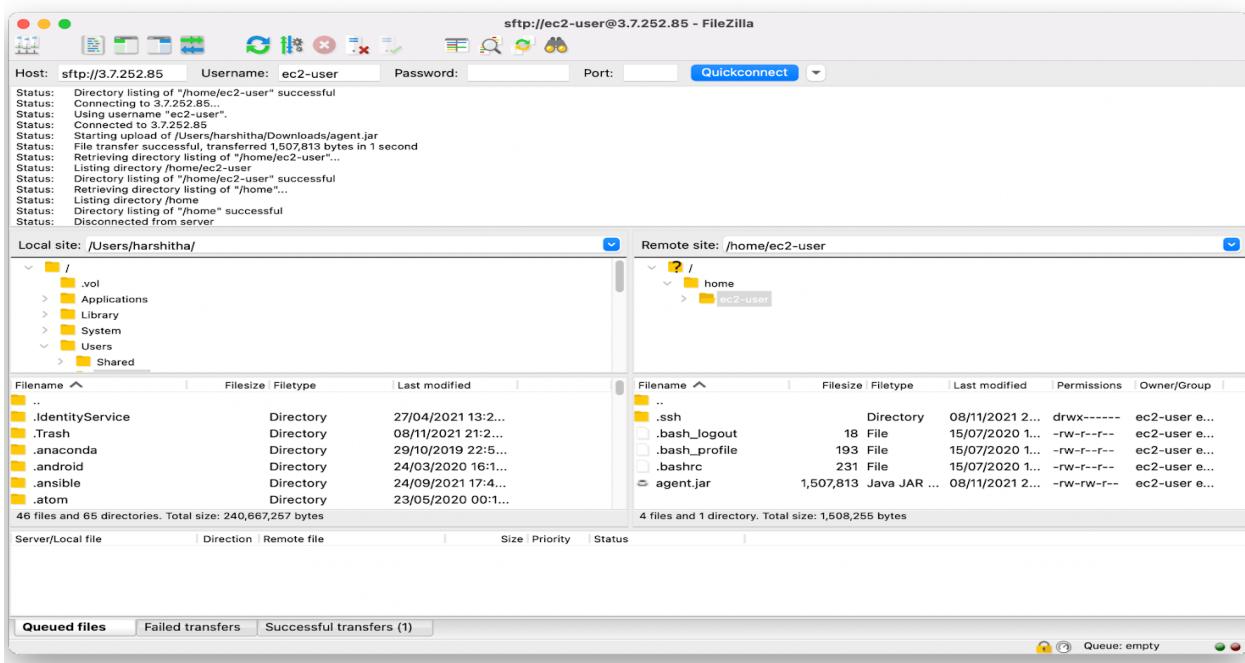
Paste the ip address, change username to ec2-user, and port number 22.



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 Node configuration page for '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

On the left sidebar, the following options are listed:

- Back to List
- Status (selected)
- Delete Agent
- Configure
- Build History
- Load Statistics
- Script Console
- Log
- System Information
- Disconnect

At the bottom of the page, there's a file manager interface showing two files:

- agent.jar
- FileZilla_3.5....tar.bz2

Page footer: REST API | Jenkins 2.303.3

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 machines

```
$ 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, only then slave-2's job is called.

3.1 Configuring the github web hook

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

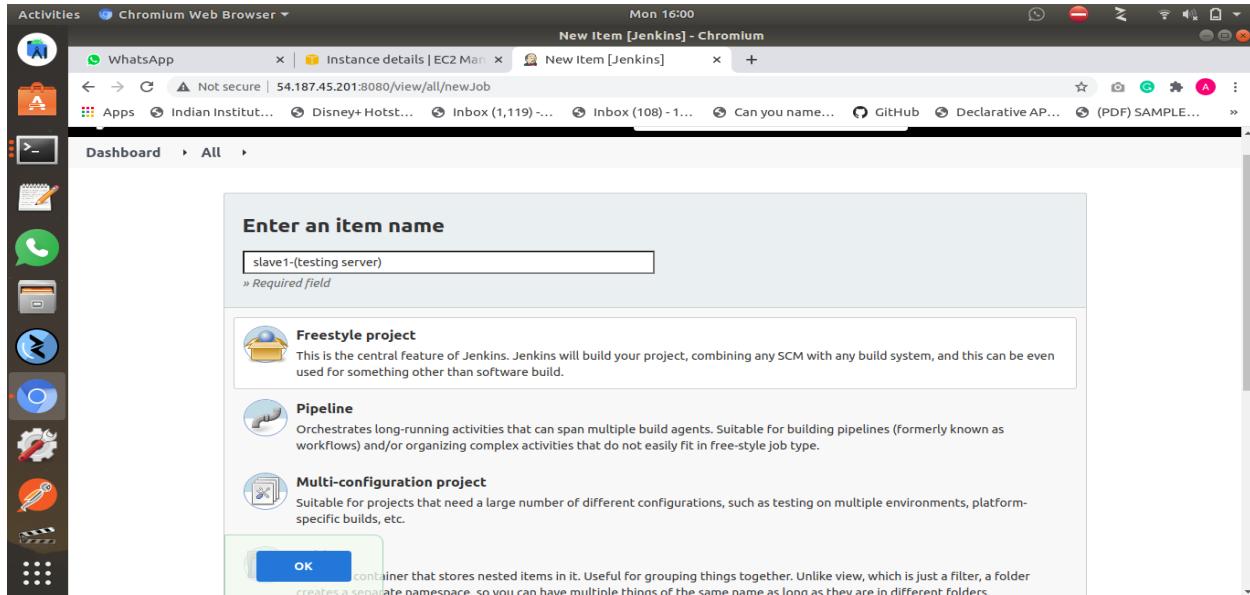
Under payload url <http://ip-address-of-jenkins-master:8080/github-webhook/>

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 freestyle project -> click ok



3.1.2 : Configuring job

1.Enable under github project

For github url add <https://github.com/Akshaya222/devops10.git>

2.Enable Restrict where the project can run

Under label expressions add slave-1

3.under source code management select git

The screenshot shows the Jenkins job configuration page for 'Test'. Under the 'General' tab, the 'Source Code Management' section is selected. It shows the 'GitHub project' option checked, and the 'Project url' field contains 'https://github.com/Namratha-shivaraju/task2.git'. There are several other build configurations like 'Discard old builds', 'Label Expression', and 'Restrict where this project can be run' (with 'slave-1' selected). The 'Save' and 'Apply' buttons are visible at the bottom.

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, to verify it

```
[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]$ 
```

Select execute shell put the commands for

delete any existing docker containers , build docker container and open port 82 for it.

Put these commands under execute shell

```
$ sudo docker rm -f $(sudo docker ps -a -q)
$ sudo docker build path_of_slave-1_job_under-workspace -t test
```

```
$ sudo docker run -it -p 82:80 -d test
```

Build the job.

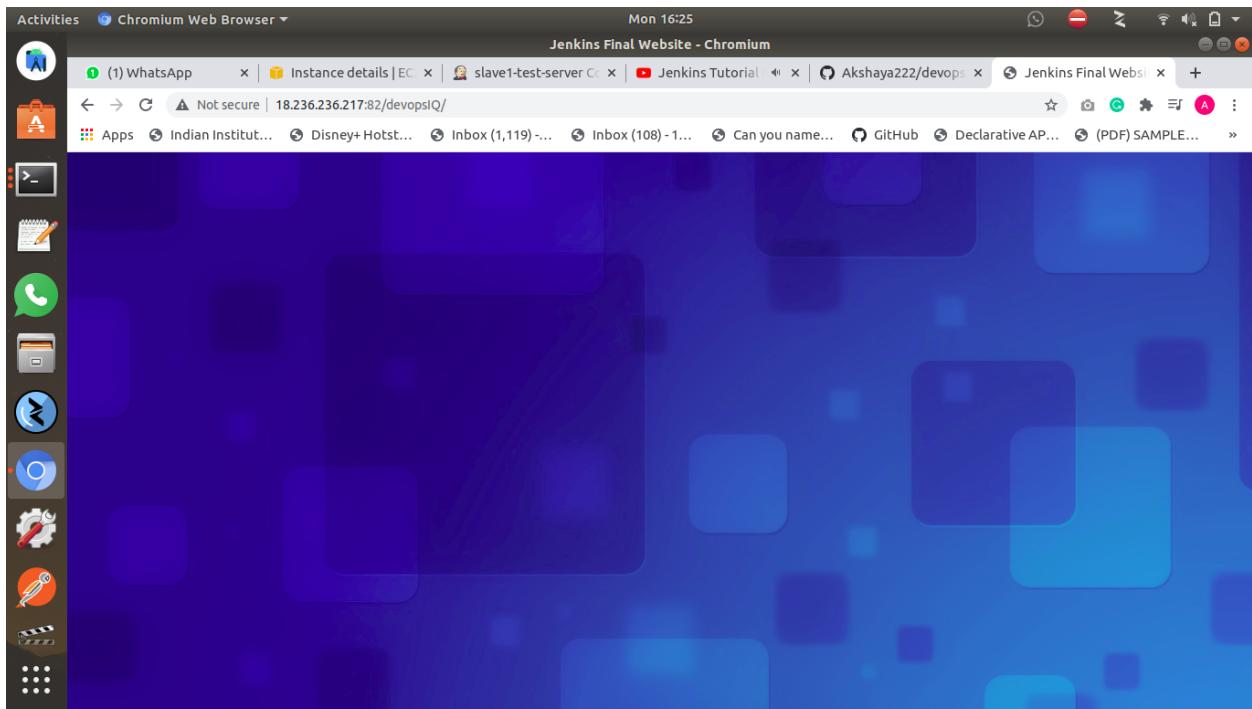
To check if the build is successful

```
Console Output
Started by user admin
Running as SYSTEM
Building remotely on jenkins-slave-1 in workspace /home/ec2-user/jenkins/workspace/slave1-test-server
The recommended git tool is: NONE
No credentials specified
> /usr/bin/git rev-parse --resolve-git-dir /home/ec2-user/jenkins/workspace/slave1-test-server/.git # timeout=10
Fetching changes from the remote Git repository
> /usr/bin/git config remote.origin.url https://github.com/Akshaya222/devopsIQ.git # timeout=10
Fetching upstream changes from https://github.com/Akshaya222/devopsIQ.git
> /usr/bin/git --version # timeout=10
> git --version # 'git' version 2.32.0
> /usr/bin/git fetch --tags --force --progress .. https://github.com/Akshaya222/devopsIQ.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> /usr/bin/git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision e44f51dfc22b2a4fa9498b91218ec5e0a718fa (refs/remotes/origin/master)
> /usr/bin/git config core.sparsecheckout # timeout=10
> /usr/bin/git checkout -f e44f51dfc22b2a4fa9498b91218ec5e0a718fa # timeout=10
Commit message: "Create test"
> /usr/bin/git rev-list --no-walk e44f51dfc22b2a4fa9498b91218ec5e0a718fa # timeout=10
[slave1-test-server] $ /bin/sh -xe /tmp/jenkins4869774153273256056.sh
++ sudo docker ps -a -n
+ sudo docker rm -f 7325eba4f5b3
7325eba4f5b3
+ sudo docker build /home/ec2-user/jenkins/workspace/slave1-test-server -t test
Sending build context to Docker daemon 24.36MB
Step 1/2 : FROM hshar/webapp
--> 0cbc1f535ed8
Step 2/2 : ADD devopsIQ /var/www/html/devopsIQ
--> db15f062e523
Successfully built db15f062e523
Successfully tagged test:latest
+ sudo docker run -it -p 82:80 -d test
2615eacb52af85590a28042137ef0a606dd8dacf994c3247773d8ef6f559ed
Finished: SUCCESS
```

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

Open custom port 82 on slave-1 instance

Go to browser and hit http://ipaddress_of_slave-1:82/devopsIQ



3.3 : Setting up job of slave-2

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

Configure this job same as job of slave-1 and under execute shell,open port 80 for docker

Put these commands under execute shell

```
$ sudo docker rm -f $(sudo docker ps -a -q)  
$ sudo docker build path_of_slave-2_job_under-workspace -t production  
$ sudo docker run -it -p 80:80 -d production
```

To check if the build is successful

```

Activities Chromium Web Browser Mon 16:49
slave2-production-server #1 Console [Jenkins] - Chromium
(1) WhatsApp Instance details | EC2 Man slave2-production-server Jenkins Tutorial for Beginner Akshaya222/devopsIQ + 
Not secure | 54.187.45.201:8080/job/slave2-production-server/1/console
Apps Indian Institut... Disney+ Hotst... Inbox (1,119) ... Inbox (108) - 1... Can you name... GitHub Declarative AP... (PDF) SAMPLE...
Dashboard > slave2-production-server > #1
0b4aba487617: Download complete
a48c509ed24e: Verifying Checksum
a48c509ed24e: Download complete
a18d6ba75273: Verifying Checksum
a18d6ba75273: Download complete
4c2cc0ff3ce8: Verifying Checksum
4c2cc0ff3ce8: Download complete
471db38bcfbf: Verifying Checksum
471db38bcfbf: Download complete
c2e32ec79cfcd: Verifying Checksum
c2e32ec79cfcd: Download complete
a48c509ed24e: Pull complete
1e1de0ff7e1: Pull complete
0330ca45a200: Pull complete
471db38bcfbf: Pull complete
0b4aba487617: Pull complete
c2e32ec79cfcd: Pull complete
a18d6ba75273: Pull complete
4c2cc0ff3ce8: Pull complete
Digest: sha256:3c7cbcab1a26c01410dcc9cbc57252b50d9ed2f31a2dc24e3f066c61b88e839b
Status: Downloaded newer image for hspark/webapp:latest
--- 0cb1f535ed8
Step 2/2 : ADD ./devopsIQ /var/www/html/devopsIQ
--> 2cba4113900f
Successfully built 2cba4113900f
Successfully tagged production:latest
+ sudo docker run -it -p 80:80 -d production
70d51b7b012982cf95b87c6f5a2d4210ae7f543303b5f3ad66a8ac08dd9208db
Finished: SUCCESS

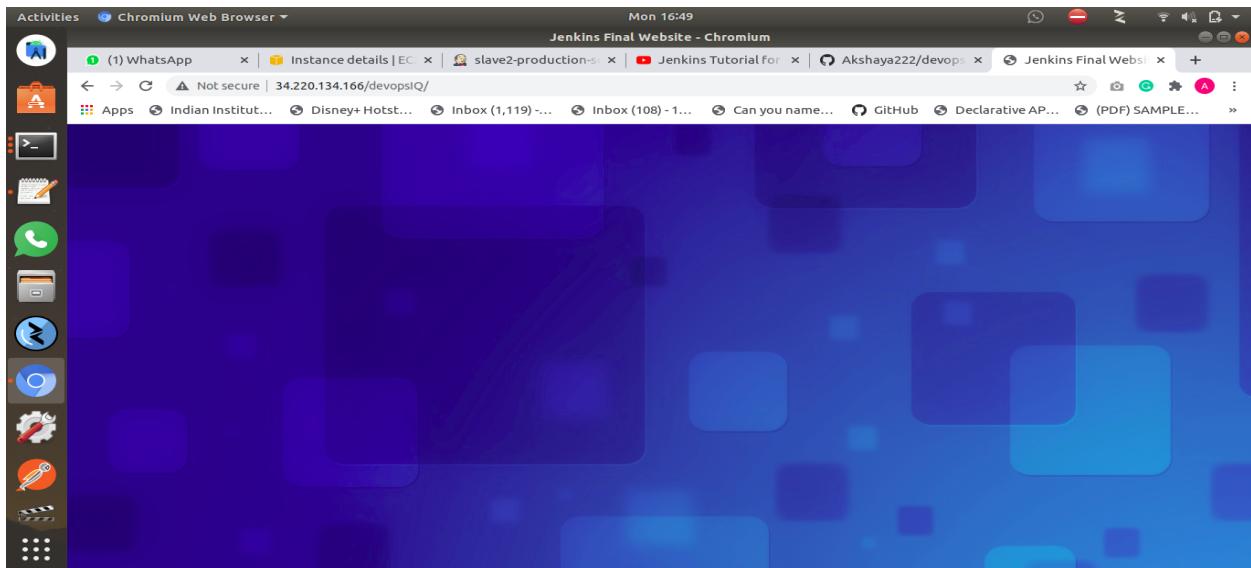
```

REST API Jenkins 2.319

To check if build is successful on production server (on slave -2)

Open custom port 80 on slave-2 instance

Go to browser and hit http://ipaddress_of_slave-2:80/devopsIQ



3.4 : Configuring pipeline to trigger production job (job of slave-2) only if testing job(job of slave-1) is successful.

Go to jenkins dashboard -> select testing job(job of slave 1) -> configure

Under post build actions -> **select build other projects** -> enter production job name (name of slave-2's job) ->

select trigger only if build is stable. -> click on save.

We can run the job of testing (job of slave 1) using pipeline view.

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. The main title is "Installing Plugins/Upgrades". On the left, there's a sidebar with links: "Back to Dashboard", "Manage Jenkins", and "Manage Plugins". The main content area has a section titled "Preparation" with three items: "Checking internet connectivity" (Success), "Checking update center connectivity" (Success), and "Success". Below this is a table-like structure showing the status of various plugin categories:

Category	Status
Run Condition	Success
Javadoc	Success
Maven Integration	Installing
Conditional BuildStep	Pending
Parameterized Trigger	Pending
jQuery	Pending
Build Pipeline	Pending
Loading plugin extensions	Pending

At the bottom of the page, there are two notes: "Go back to the top page" and "Restart Jenkins when installation is complete and no jobs are running".

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

The screenshot shows the Jenkins dashboard at 65.1.94.190:8080. On the left sidebar, under 'New View', there is a 'New View' button. The main area displays a list of jobs: 'Prod' and 'Test'. The '+' button in the top left corner of the job list is circled in yellow.

All	S	W	Name	Last Success	Last Failure	Last Duration
			Prod	6 min 54 sec - #4	9 min 6 sec - #2	2.2 sec
			Test	20 min - #9	20 min - #8	25 sec

Legend: S M L
[Atom feed for all](#) [Atom feed for failures](#) [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 - Test

Click Ok

The screenshot shows the 'New View' configuration page at 65.1.94.190:8080/newView. The 'View name' field contains 'CICD'. The 'Build Pipeline View' radio button is selected, with a note below it stating: 'Shows the jobs in a build pipeline view. The complete pipeline of jobs that a version propagates through are shown as a row in the view.' There are also 'List View' and 'My View' options.

View name
CICD

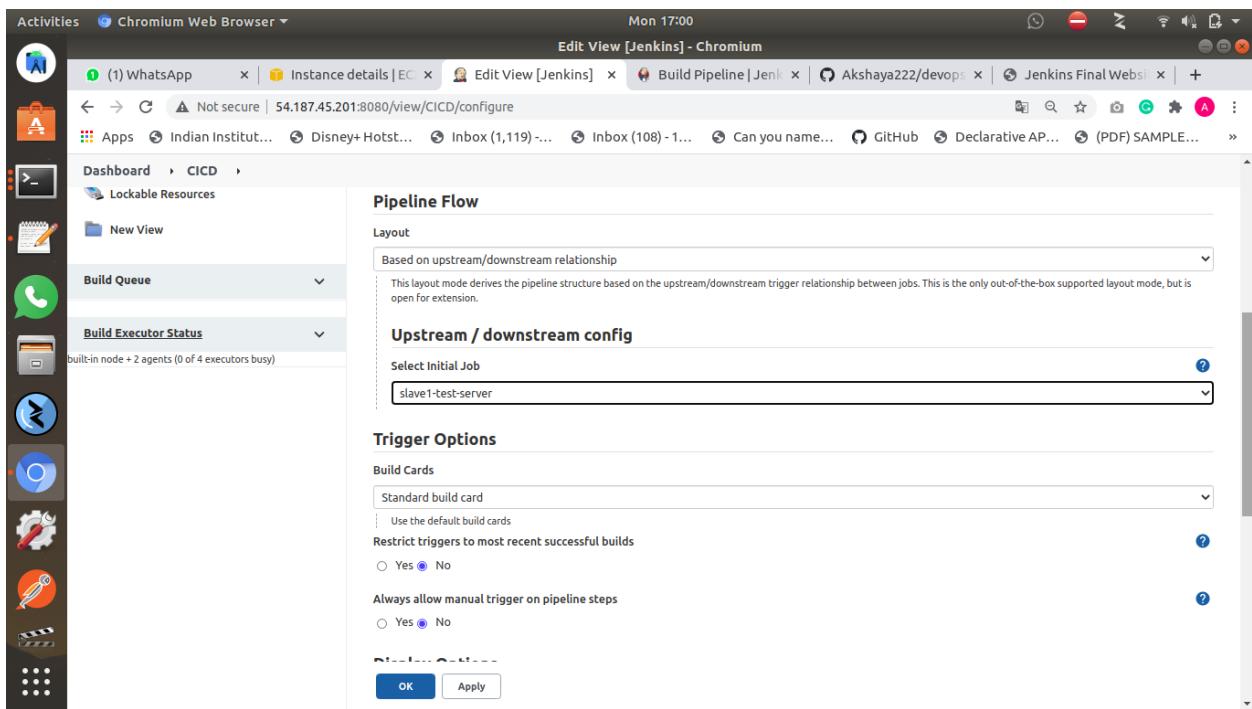
Build Pipeline View
Shows the jobs in a build pipeline view. The complete pipeline of jobs that a version propagates through are shown as a row in the view.

List View
Shows items in a simple list format. You can choose which jobs are to be displayed in which view.

My View
This view automatically displays all the jobs that the current user has an access to.

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.

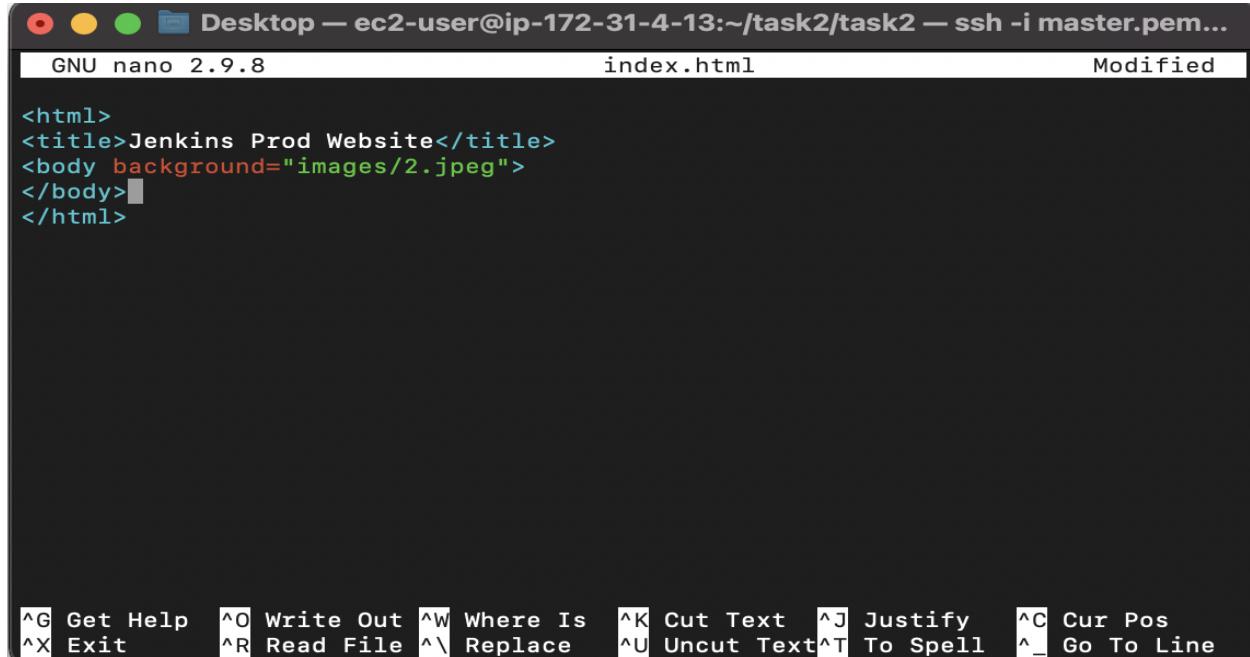
When the new push event occurs on github, the testing job automatically gets triggered with the help of webhook.

Go to jenkins master server to clone and make changes in the github.

Cloning the repository

```
[ec2-user@ip-172-31-4-13 ~]$ git clone https://github.com/Namratha-shivaraju/tas]
k2.git
Cloning into 'task2'...
remote: Enumerating objects: 28, done.
remote: Counting objects: 100% (28/28), done.
remote: Compressing objects: 100% (26/26), done.
remote: Total 28 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (28/28), 11.45 MiB | 7.56 MiB/s, done.
Resolving deltas: 100% (1/1), done.
[ec2-user@ip-172-31-4-13 ~]$
```

Editing index.html



```

Desktop — ec2-user@ip-172-31-4-13:~/task2/task2 — ssh -i master.pem...
GNU nano 2.9.8                               index.html                         Modified

<html>
<title>Jenkins Prod Website</title>
<body background="images/2.jpeg">
</body>
</html>

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line

```

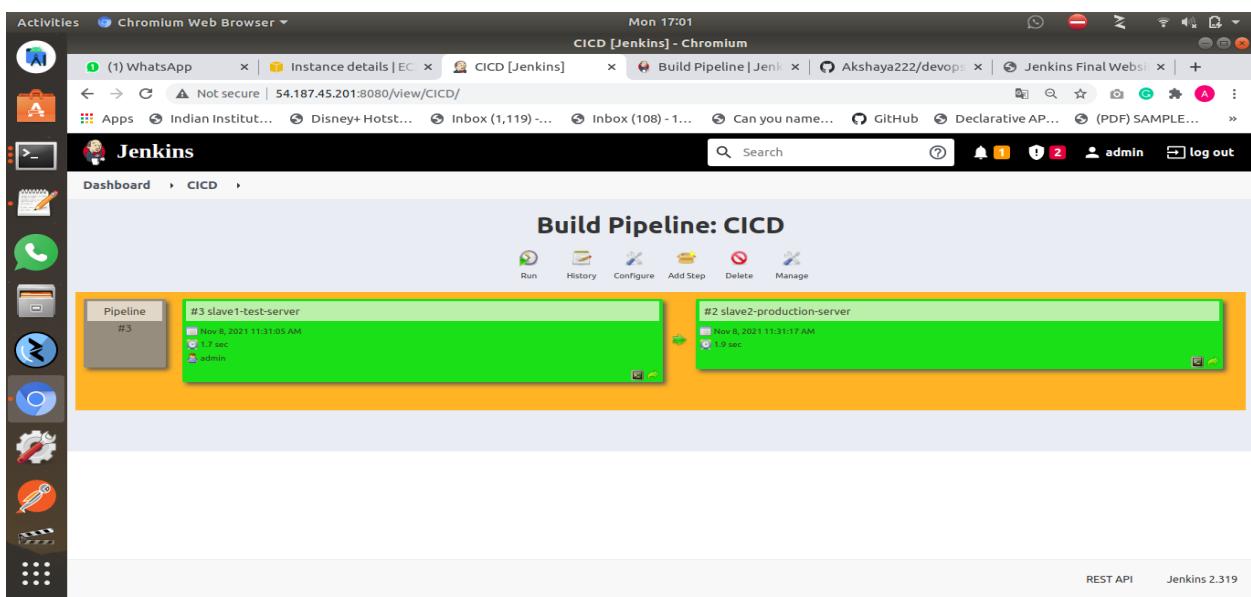
Committing and pushing the changes to github repository

```

akshaya@akshaya-Vostro-3478:~/Desktop/devopsIQ/devopsIQ$ gedit index.html
akshaya@akshaya-Vostro-3478:~/Desktop/devopsIQ/devopsIQ$ git add .
akshaya@akshaya-Vostro-3478:~/Desktop/devopsIQ/devopsIQ$ git commit -m "changing
image"
[master 17746ae] changing image
 1 file changed, 2 insertions(+), 2 deletions(-)
akshaya@akshaya-Vostro-3478:~/Desktop/devopsIQ/devopsIQ$ git push origin master
Username for 'https://github.com': akshaya222
Password for 'https://akshaya222@github.com':
Counting objects: 4, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 414 bytes | 414.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Akshaya222/devopsIQ.git
  e44f51d..17746ae  master -> master
akshaya@akshaya-Vostro-3478:~/Desktop/devopsIQ/devopsIQ$ 

```

Go to pipeline view to see the jobs status



Both the jobs ran successfully.

To verify if the changes have been deployed on testing server

Go to browser and hit http://ipaddress_of_slave-1:82/devopsIQ

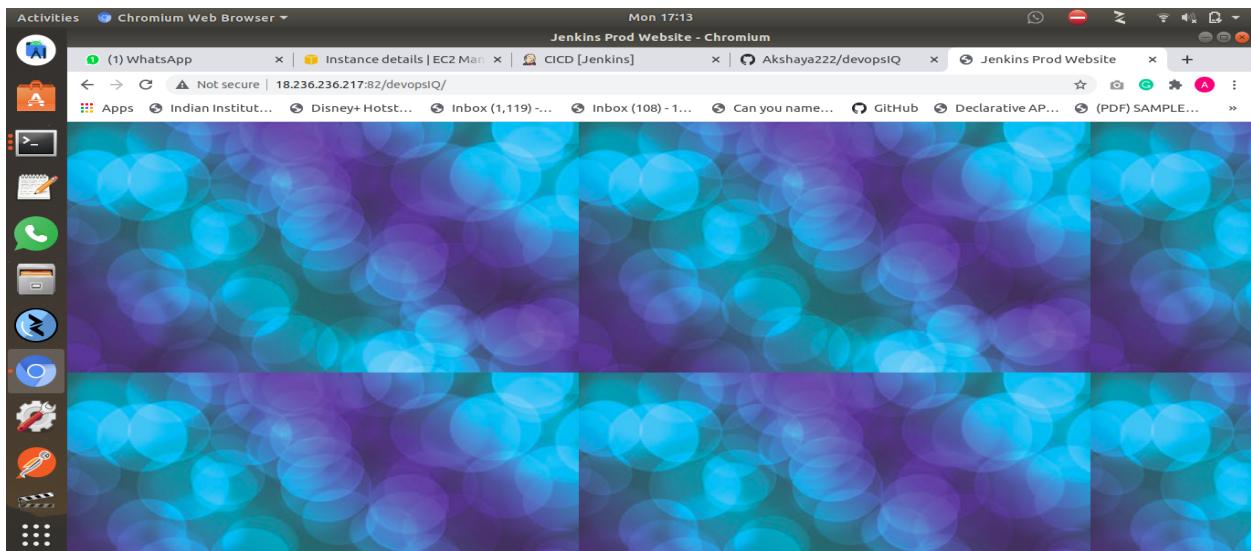
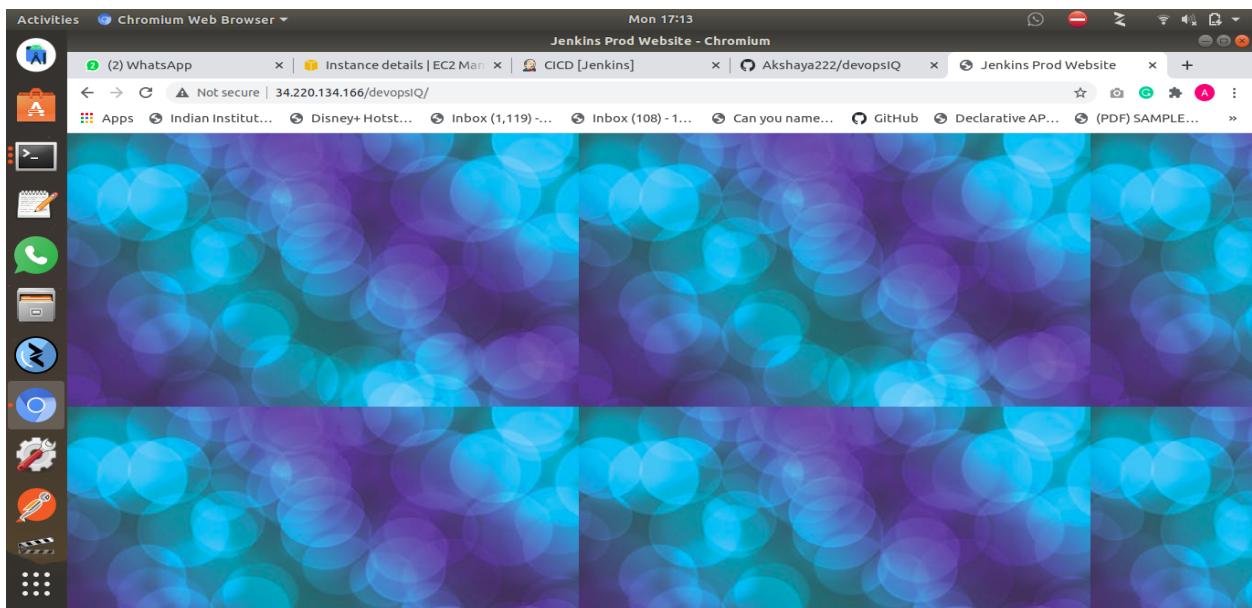


Image has been changed that means changes deployed successfully. So the testing job build was successful.

So the production job must have been triggered as the testing job was successful.

To verify if the changes have been deployed on production server

Go to browser and hit http://ipaddress_of_slave-2:80/devopsIQ



Changes have been successfully deployed to the production server.
Jenkins master slave pipeline has been successful.