



INDIAN INSTITUTE OF
INFORMATION
TECHNOLOGY

CS457

Devops Final Assignment-1

Submitted to:
Dr.Uma S

Submitted by:
Team 6
Pokala Dattatreya (18BCS067)
Rama Dundi Saketh(18BCS076)
S Sampath(18BCS087)

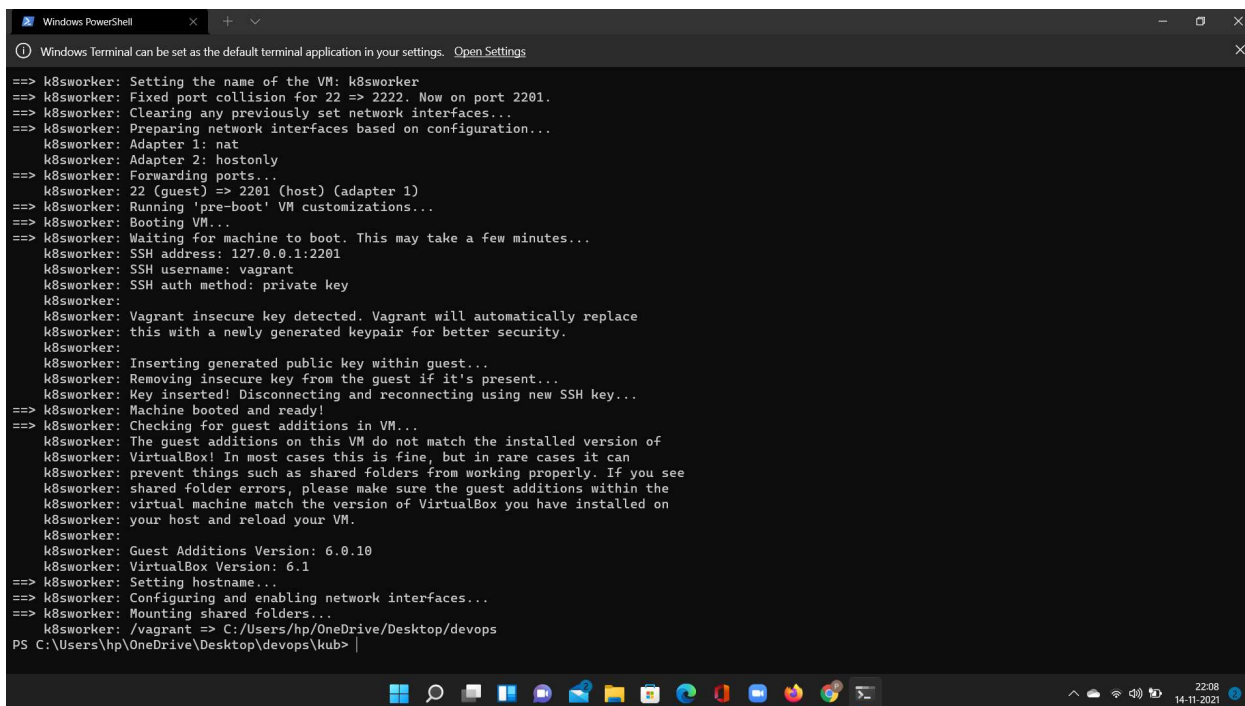
Set up complete CI/CD Jenkins pipeline for kubernetes

Tools and Technologies used:

- >Virtual Box
- >Docker
- >Jenkins
- >Kubernetes

Step1:

- >Setting up 3 VM's using vagrant



```
Windows PowerShell
Windows Terminal can be set as the default terminal application in your settings. Open Settings

==> k8sworker: Setting the name of the VM: k8sworker
==> k8sworker: Fixed port collision for 22 => 2222. Now on port 2201.
==> k8sworker: Clearing any previously set network interfaces...
==> k8sworker: Preparing network interfaces based on configuration...
k8sworker: Adapter 1: nat
k8sworker: Adapter 2: hostonly
==> k8sworker: Forwarding ports...
k8sworker: 22 (guest) => 2201 (host) (adapter 1)
==> k8sworker: Running 'pre-boot' VM customizations...
==> k8sworker: Booting VM...
==> k8sworker: Waiting for machine to boot. This may take a few minutes...
k8sworker: SSH address: 127.0.0.1:2201
k8sworker: SSH username: vagrant
k8sworker: SSH auth method: private key
k8sworker:
k8sworker: Vagrant insecure key detected. Vagrant will automatically replace
k8sworker: this with a newly generated keypair for better security.
k8sworker:
k8sworker: Inserting generated public key within guest...
k8sworker: Removing insecure key from the guest if it's present...
k8sworker: Key inserted! Disconnecting and reconnecting using new SSH key...
==> k8sworker: Machine booted and ready!
==> k8sworker: Checking for guest additions in VM...
k8sworker: The guest additions on this VM do not match the installed version of
k8sworker: VirtualBox! In most cases this is fine, but in rare cases it can
k8sworker: prevent things such as shared folders from working properly. If you see
k8sworker: shared folder errors, please make sure the guest additions within the
k8sworker: virtual machine match the version of VirtualBox you have installed on
k8sworker: your host and reload your VM.
k8sworker:
k8sworker: Guest Additions Version: 6.0.10
k8sworker: VirtualBox Version: 6.1
==> k8sworker: Setting hostname...
==> k8sworker: Configuring and enabling network interfaces...
==> k8sworker: Mounting shared folders...
k8sworker: /vagrant => C:/Users/hp/OneDrive/Desktop/devops
PS C:\Users\hp\OneDrive\Desktop\devops\kub>
```

Step2:

-> Clone the kubespray git repo

link: <https://github.com/kubernetes-sigs/kubespray.git>

-> Install the Kubespray Packages

-> Copy inventory file to current users

-> Prepare host.yml for kubespray

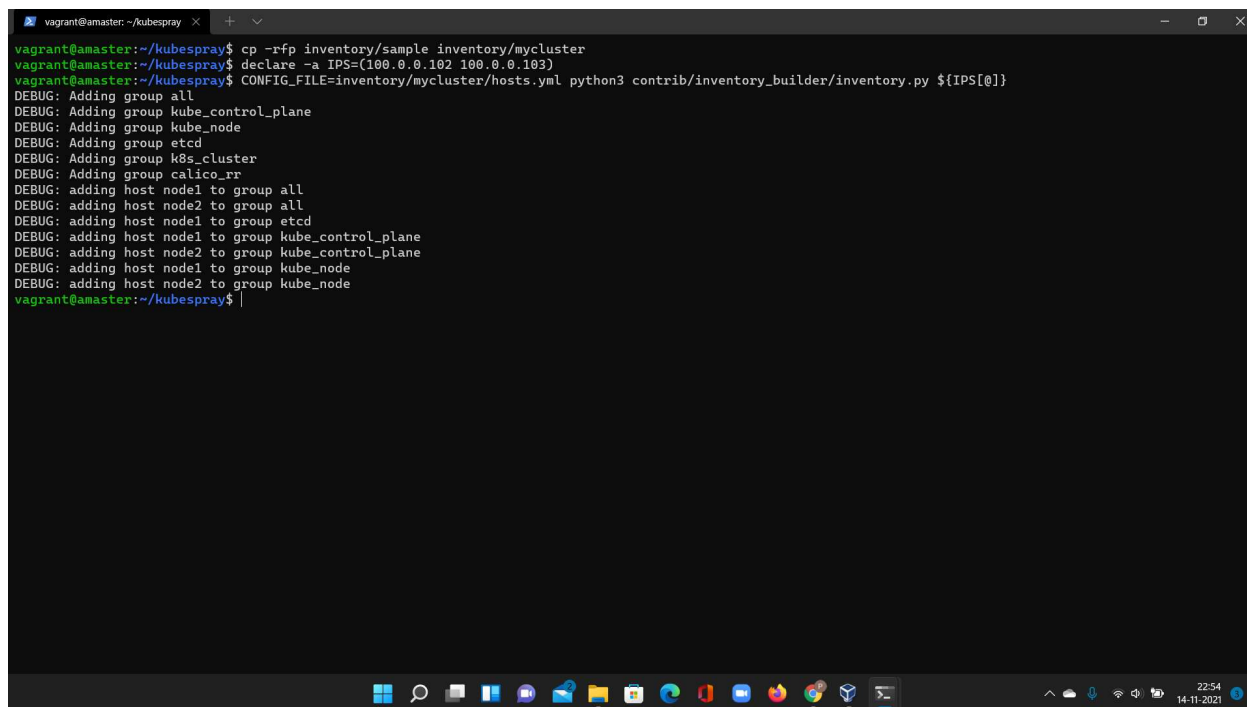
Commands to run:

-> sudo pip3 install -r requirements.txt

-> cp -rfp inventory/sample inventory/mycluster

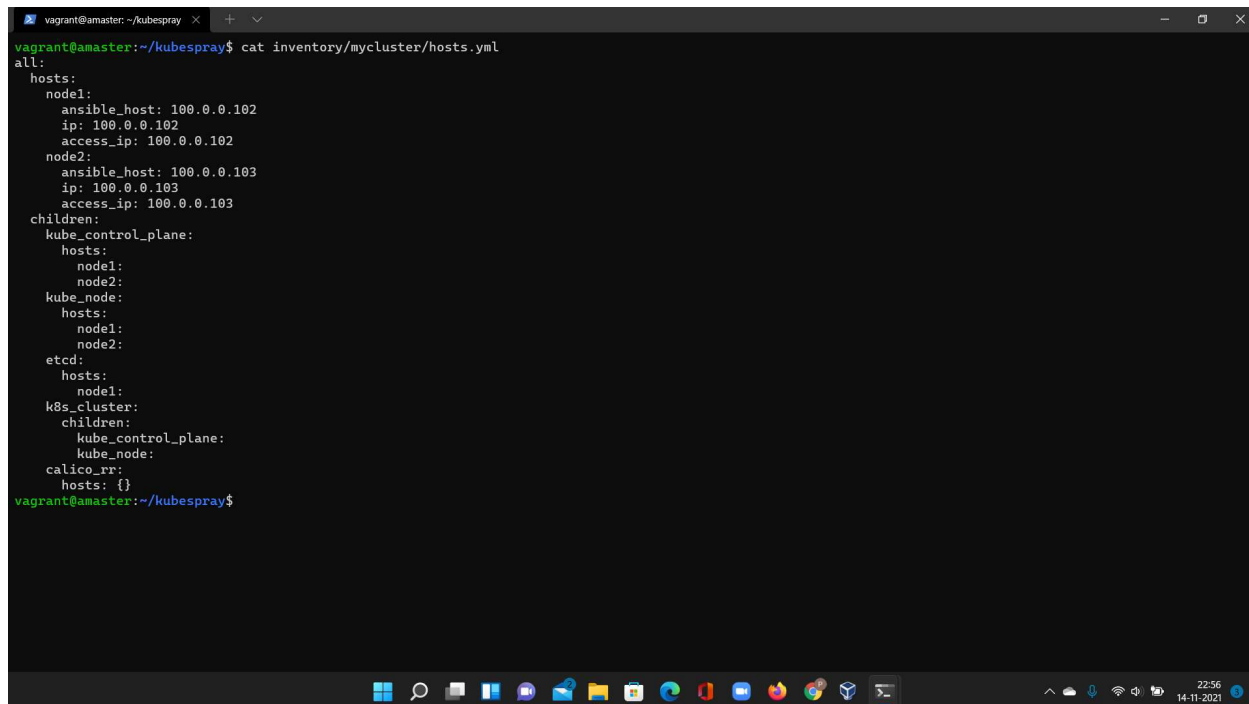
-> declare -a IPS=(100.0.0.102 100.0.0.103)

-> CONFIG_FILE=inventory/mycluster/hosts.yml python3 contrib/inventory_builder/inventory.py \${IPS[@]}



```
vagrant@amaster: ~/kubespray
vagrant@amaster:~/kubespray$ cp -rfp inventory/sample inventory/mycluster
vagrant@amaster:~/kubespray$ declare -a IPS=(100.0.0.102 100.0.0.103)
vagrant@amaster:~/kubespray$ CONFIG_FILE=inventory/mycluster/hosts.yml python3 contrib/inventory_builder/inventory.py ${IPS[@]}
DEBUG: Adding group all
DEBUG: Adding group kube_control_plane
DEBUG: Adding group kube_node
DEBUG: Adding group etcd
DEBUG: Adding group k8s_cluster
DEBUG: Adding group calico_rr
DEBUG: adding host node1 to group all
DEBUG: adding host node2 to group all
DEBUG: adding host node1 to group etcd
DEBUG: adding host node1 to group kube_control_plane
DEBUG: adding host node2 to group kube_control_plane
DEBUG: adding host node1 to group kube_node
DEBUG: adding host node2 to group kube_node
vagrant@amaster:~/kubespray$
```

->After running the above commands do verify the hosts.yml and it should be like below



```
vagrant@amaster: ~/kubespray
vagrant@amaster:~/kubespray$ cat inventory/mycluster/hosts.yml
all:
  hosts:
    node1:
      ansible_host: 100.0.0.102
      ip: 100.0.0.102
      access_ip: 100.0.0.102
    node2:
      ansible_host: 100.0.0.103
      ip: 100.0.0.103
      access_ip: 100.0.0.103
  children:
    kube_control_plane:
      hosts:
        node1:
        node2:
    kube_node:
      hosts:
        node1:
        node2:
    etcd:
      hosts:
        node1:
    k8s_cluster:
      children:
        kube_control_plane:
        kube_node:
    calico_rr:
      hosts: {}
vagrant@amaster:~/kubespray$
```

->Run the ansible-playbook on ansible node

Commands to run:

-> `ansible-playbook -i inventory/mycluster/hosts.yml --become --become-user=root cluster.yml`

Step3:

->Install Kubectl on kubernetes master

Commands to run:

1) curl -LO https://storage.googleapis.com/kubernetes-release/release/`curl -s <https://storage.googleapis.com/kubernetes-release/release/stable.txt>`/bin/linux/amd64/kubectl

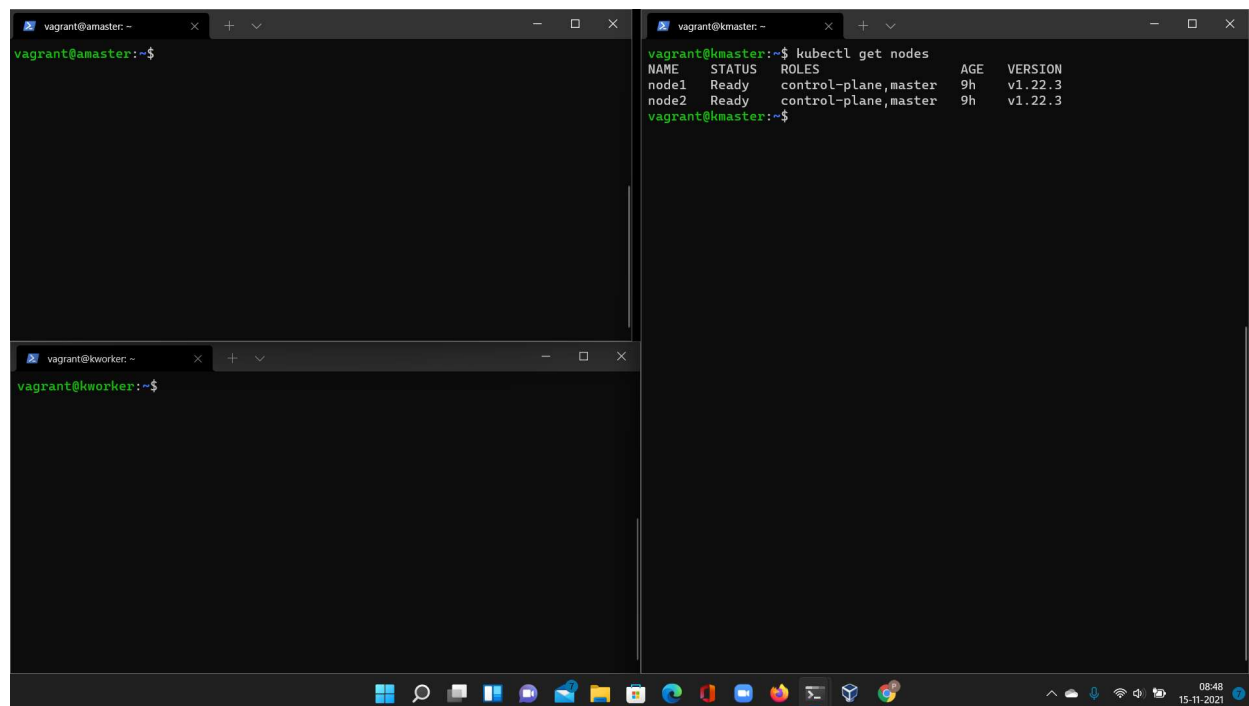
2)sudo cp /etc/kubernetes/admin.conf /home/vagrant/config

3)mv config .kube/

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

Now verify the kubernetes nodes:

Command:kubectl get nodes



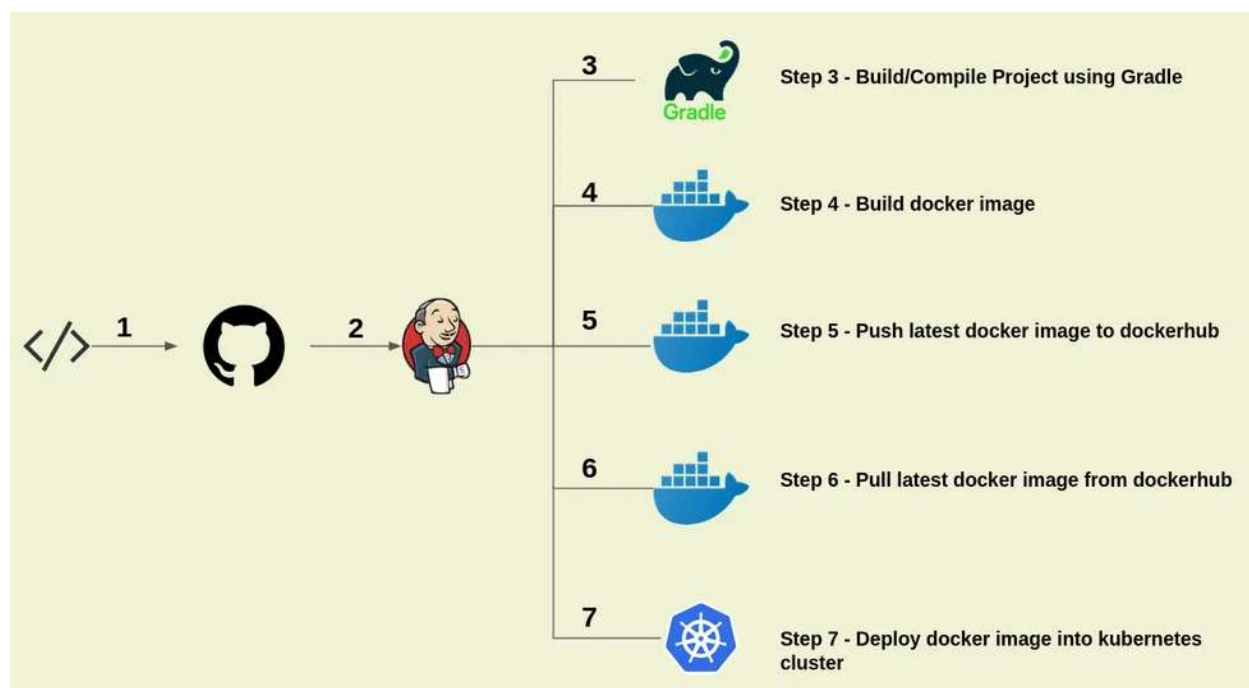
The screenshot shows three terminal windows. The top-left window is titled 'vagrant@amaster: ~' and shows a prompt 'vagrant@amaster:~\$'. The top-right window is titled 'vagrant@kmaster: ~' and shows the command 'vagrant@kmaster:~\$ kubectl get nodes' and its output:

NAME	STATUS	ROLES	AGE	VERSION
node1	Ready	control-plane,master	9h	v1.22.3
node2	Ready	control-plane,master	9h	v1.22.3

The bottom window is titled 'vagrant@kworker: ~' and shows a prompt 'vagrant@kworker:~\$'. The Windows taskbar is visible at the bottom with the time 08:48 and date 15-11-2021.

Step4:

overview of our GitHub and DockerHub flow :



4.1)Push code to github

Github link: <https://github.com/dattu24/devopsIQ>

4.2)Install Java and Jenkins

Commands to execute:

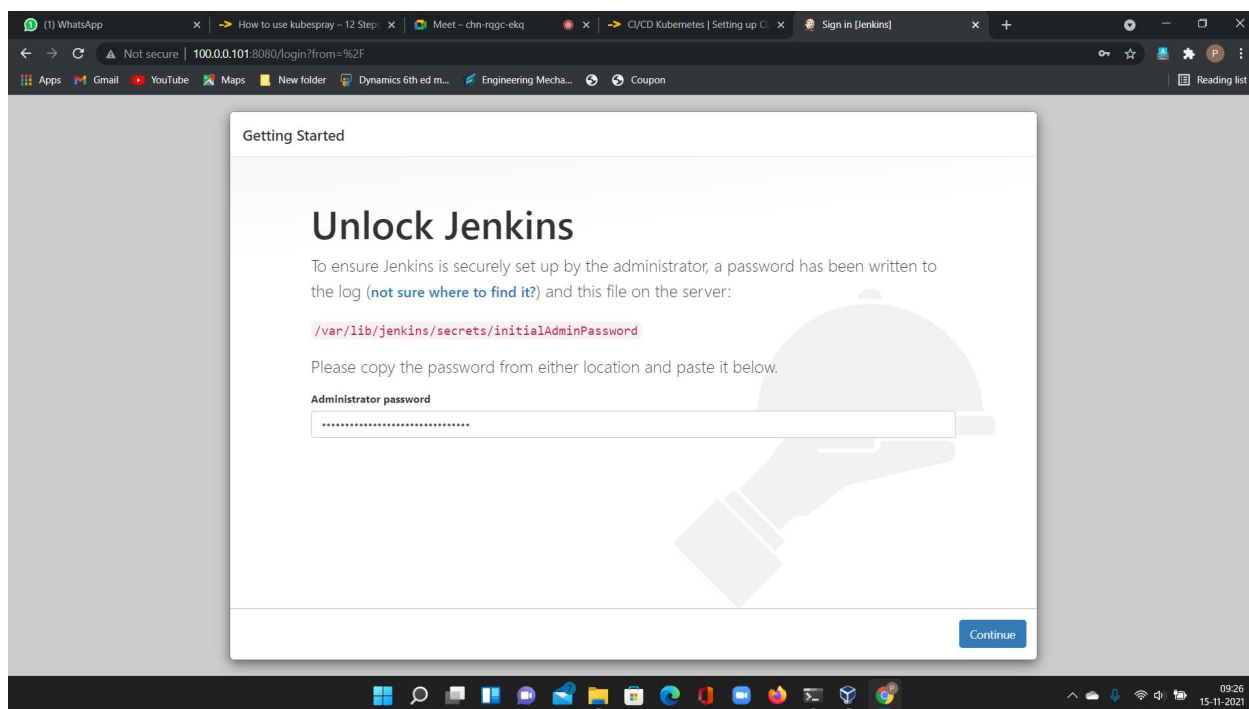
1)sudo apt install openjdk-11-jdk

2)wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -

3)sudo apt-get install jenkins

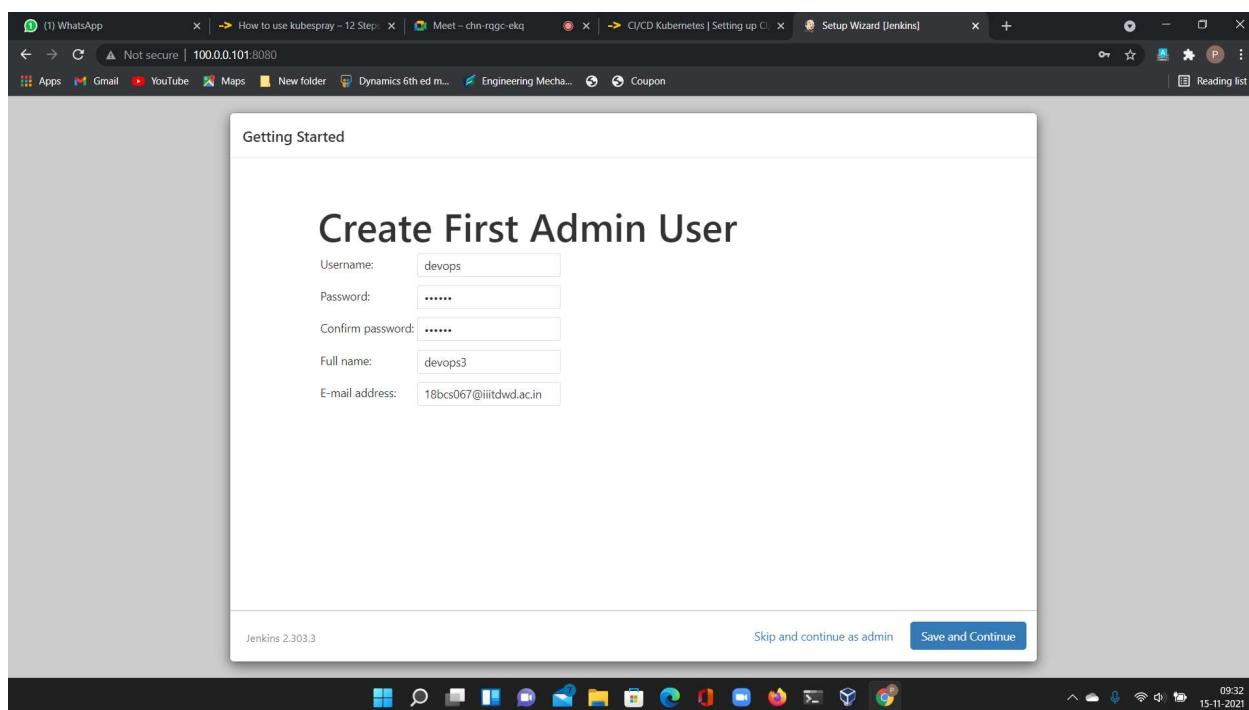
4.3)Now Verify Jenkins installation

Go to this link to verify <http://100.0.0.101:8080/>



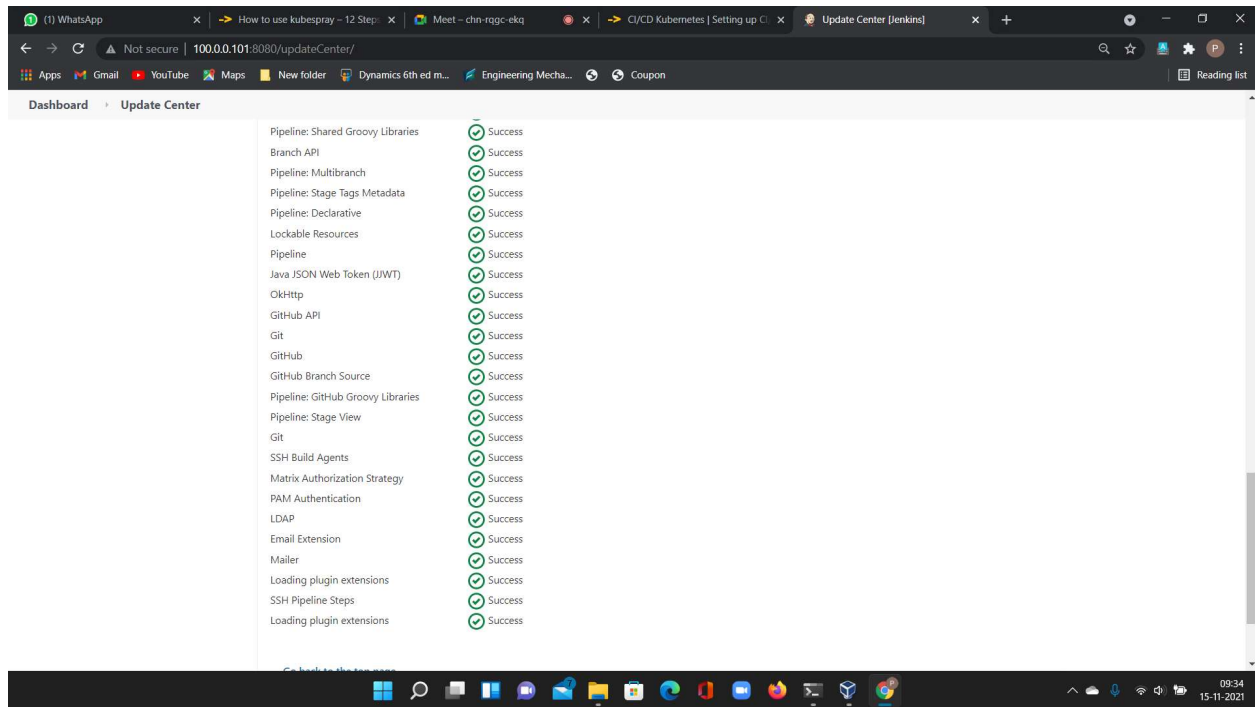
->Customize Jenkins and install suggested plugin

->Then setup username and password



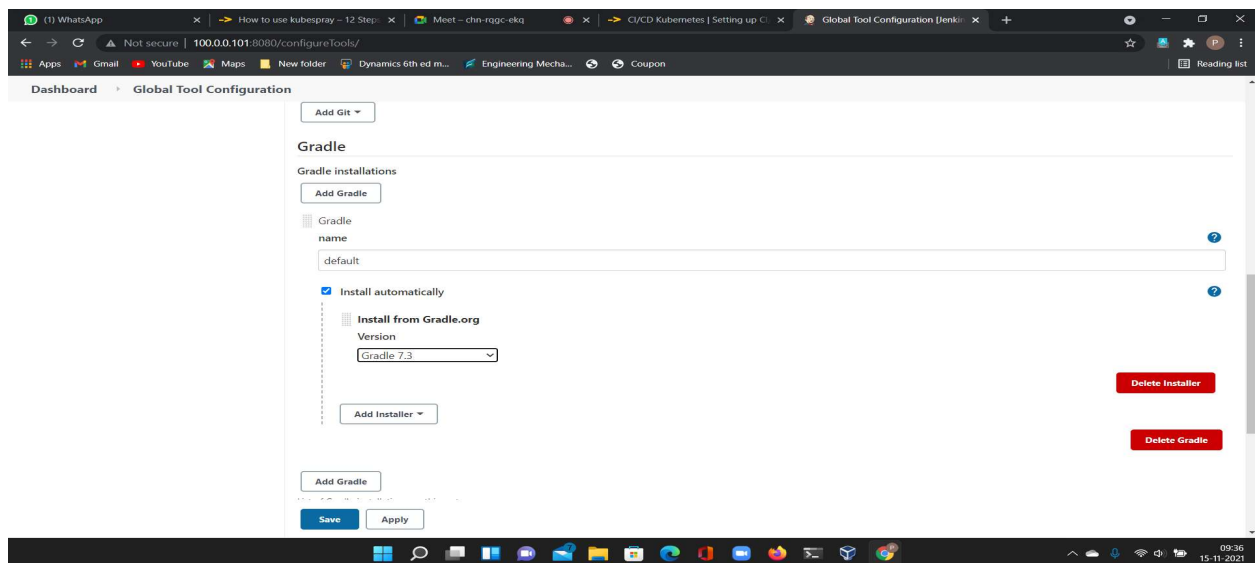
4.4) Install one more plugin SSH Pipeline steps:

For installing plugin please goto - Manage Jenkins -> Manage Plugin -> Available then in the search box type SSH Pipeline Steps.



4.5) Setup gradle

To setup Gradle Goto - Manage Jenkins -> Global Tool Configuration -> Gradle



4.6) Install Jenkins on Docker on Jenkins Server

Commands to execute

- 1) `vagrant ssh jenkinsserver`
- 2) `sudo apt install docker.io`
Adding current user to docker
- 3) `sudo usermod -aG docker $USER`
- 4) `sudo usermod -aG docker jenkins`

The screenshot shows three terminal windows in a Windows environment. The top-left window is titled 'vagrant@k8smaster: ~' and shows the prompt 'vagrant@k8smaster:~\$'. The bottom-left window is titled 'vagrant@k8sworker: ~' and shows the prompt 'vagrant@k8sworker:~\$'. The right window is titled 'vagrant@jenkinsserver: ~' and shows the output of the command 'docker version'.

```
vagrant@jenkinsserver:~$ docker version
Client:
 Version:           20.10.7
 API version:       1.41
 Go version:        go1.13.8
 Git commit:        20.10.7-0ubuntu5-18.04.3
 Built:             Mon Nov  1 01:04:14 2021
 OS/Arch:           linux/amd64
 Context:           default
 Experimental:      true

Server:
 Engine:
  Version:          20.10.7
  API version:      1.41 (minimum version 1.12)
  Go version:       go1.13.8
  Git commit:       20.10.7-0ubuntu5-18.04.3
  Built:            Fri Oct 22 00:57:37 2021
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.5.5-0ubuntu3-18.04.1
  GitCommit:        6f74d86fd1c06f2dc066a2981e4016e95d36c1e5
 runc:
  Version:          1.0.1-0ubuntu2-18.04.1
  GitCommit:        dc92c8f7dd29a2bfbaf4a70b607bd5e564236bf9
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0b244c85181225b24c2f1e2ac24f28329e
vagrant@jenkinsserver:~$
```

Step5:

Here we are using spring boot applications.

Now, we should write Pipeline Script:

5.1)

- 1) Goto : Jenkins -> New Items
- 2) Enter an item name : item_name
- 3) Select Pipeline

4)Click Ok

5.2)

Clone the git repo

Code:

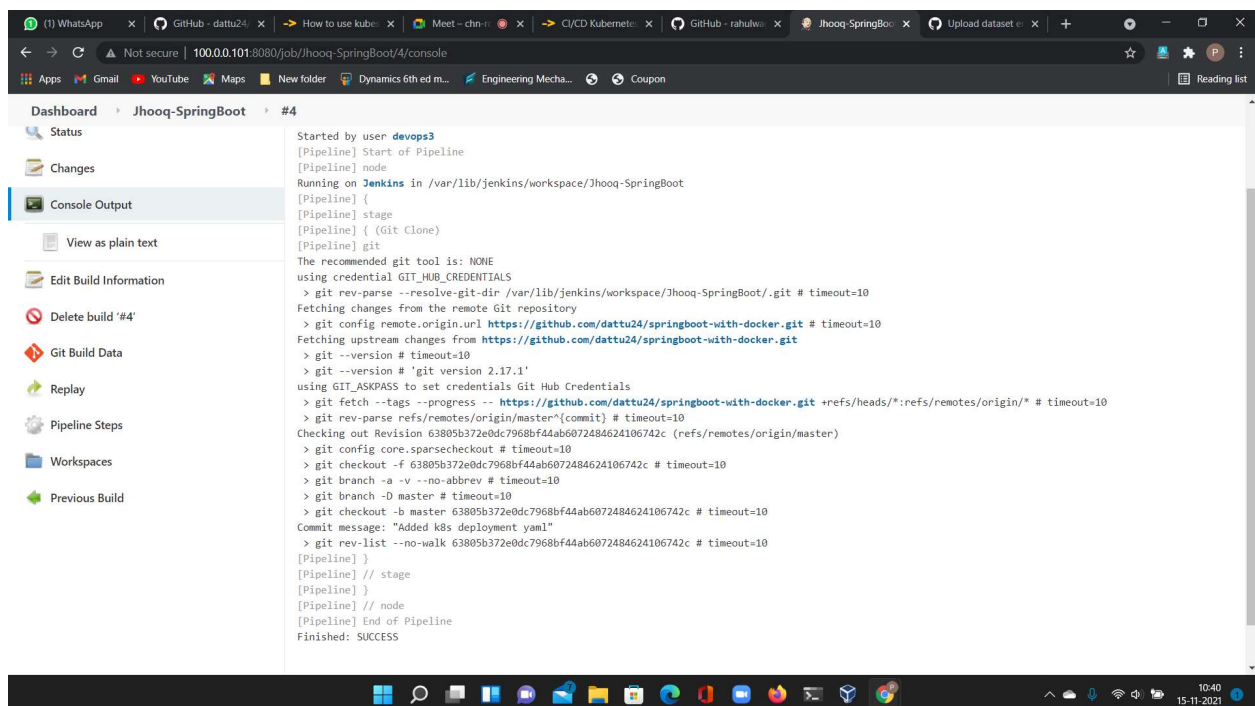
Adding a stage in pipeline(stage1):

```
stage("Git Clone"){
    git credentialsId: 'GIT_HUB_CREDENTIALS', url:
    'https://github.com/rahulwagh/spring-boot-docker.git'
}
```

Jenkins store git Credentials:

Goto : Jenkins -> Manage Jenkins -> Manage Credentials

Logs of Build



The screenshot shows the Jenkins console output for a pipeline build. The left sidebar contains navigation links: Dashboard, Jhooq-SpringBoot, #4, Status, Changes, Console Output (selected), View as plain text, Edit Build Information, Delete build '#4', Git Build Data, Replay, Pipeline Steps, Workspaces, and Previous Build. The main console output area displays the following text:

```
Started by user devops3
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/Jhooq-SpringBoot
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Git Clone)
[Pipeline] git
The recommended git tool is: NONE
using credential GIT_HUB_CREDENTIALS
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/Jhooq-SpringBoot/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/dattu24/springboot-with-docker.git # timeout=10
Fetching upstream changes from https://github.com/dattu24/springboot-with-docker.git
> git --version # timeout=10
> git --version # 'git version 2.17.1'
using GIT_ASKPASS to set credentials Git Hub Credentials
> git fetch --tags --progress -- https://github.com/dattu24/springboot-with-docker.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 63805b372e0dc7968bf44ab6072484624106742c (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f 63805b372e0dc7968bf44ab6072484624106742c # timeout=10
> git branch -a -v --no-abbrev # timeout=10
> git branch -D master # timeout=10
> git checkout -b master 63805b372e0dc7968bf44ab6072484624106742c # timeout=10
Commit message: "Added k8s deployment yaml"
> git rev-list --no-walk 63805b372e0dc7968bf44ab6072484624106742c # timeout=10
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

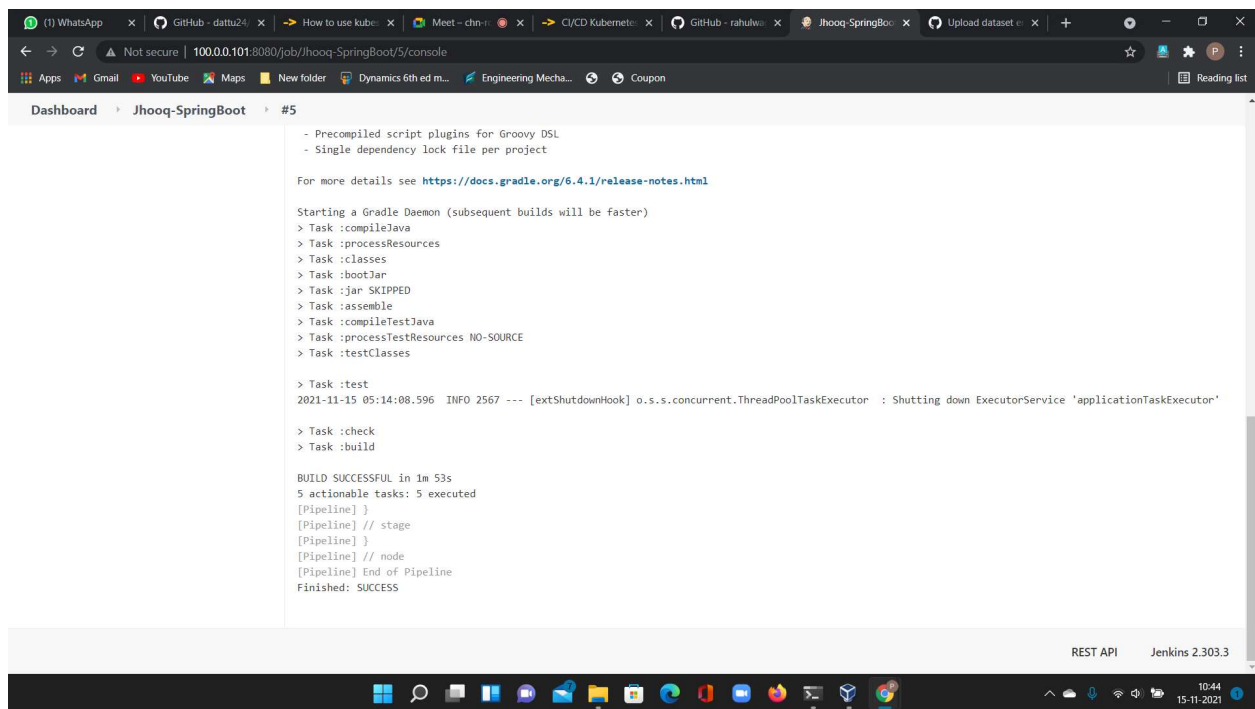
5.3) Build the spring boot application

Code:

Pipeline(stage2)

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

Build Log



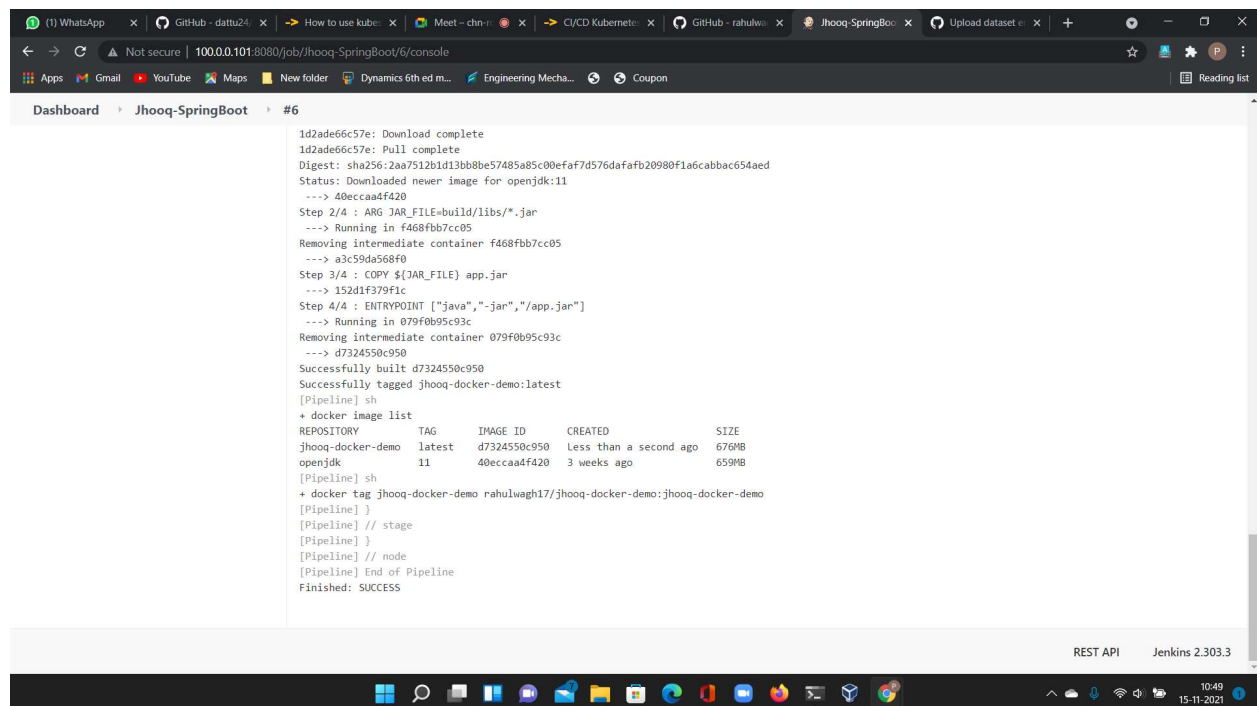
5.4) Build the docker image and tag it

Code:

Pipeline(stage3)

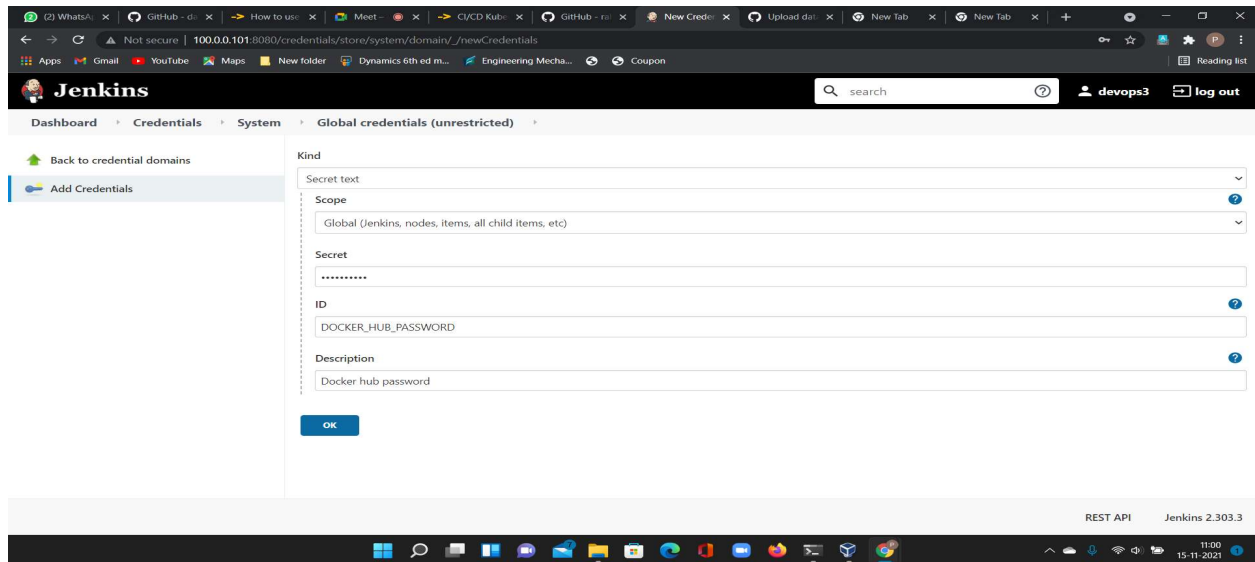
```
stage("Docker build"){
    sh 'docker version' sh 'docker build -t jhooq-docker-demo .' sh 'docker
    image list' sh 'docker tag jhooq-docker-demo
    dattu24/jhooq-docker-demo:jhooq-docker-demo'
}
```

Build log



5.5) Jenkins store DockerHub credentials:

For storing DockerHub Credentials you need to GOTO: Jenkins -> Manage Jenkins -> Manage Credentials -> Stored scoped to jenkins -> global -> Add Credentials



5.6) Docker login via CLI:

Code:

pipeline(stage4):

```
stage("Docker Login"){
    withCredentials([string(credentialsId:
    'DOCKER_HUB_PASSWORD', variable: 'PASSWORD')])
    { sh 'docker login -u rahulwagh17 -p $PASSWORD' }
}
```

Build log:



5.7)Push Docker hub in to Docker Image:

Code:

Pipeline(stage5):

```
stage("Push Image to Docker Hub"){
    sh 'docker push dattu24/jhooq-docker-demo:jhooq-docker-demo'
}
```

5.8)SSH Into k8smaster server:

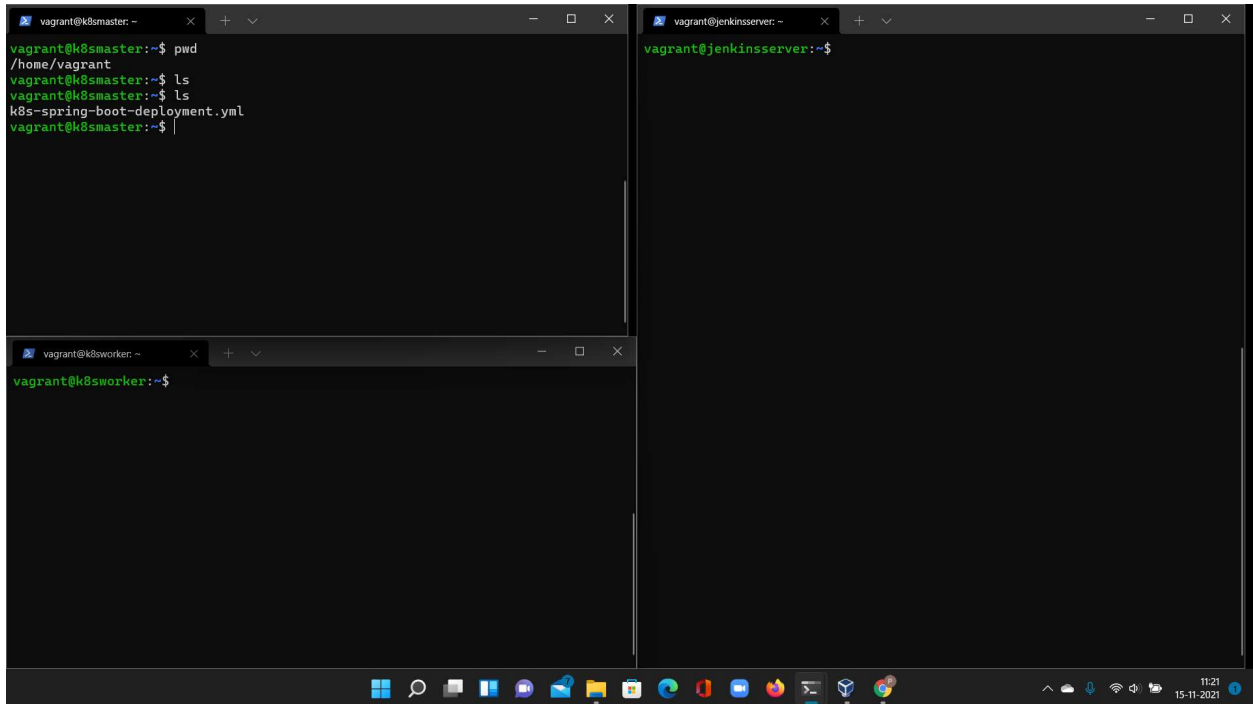
5.9)Copy k8s-spring-boot-deployment.yml to k8smaster server

5.10)Create kubernetes deployment and service

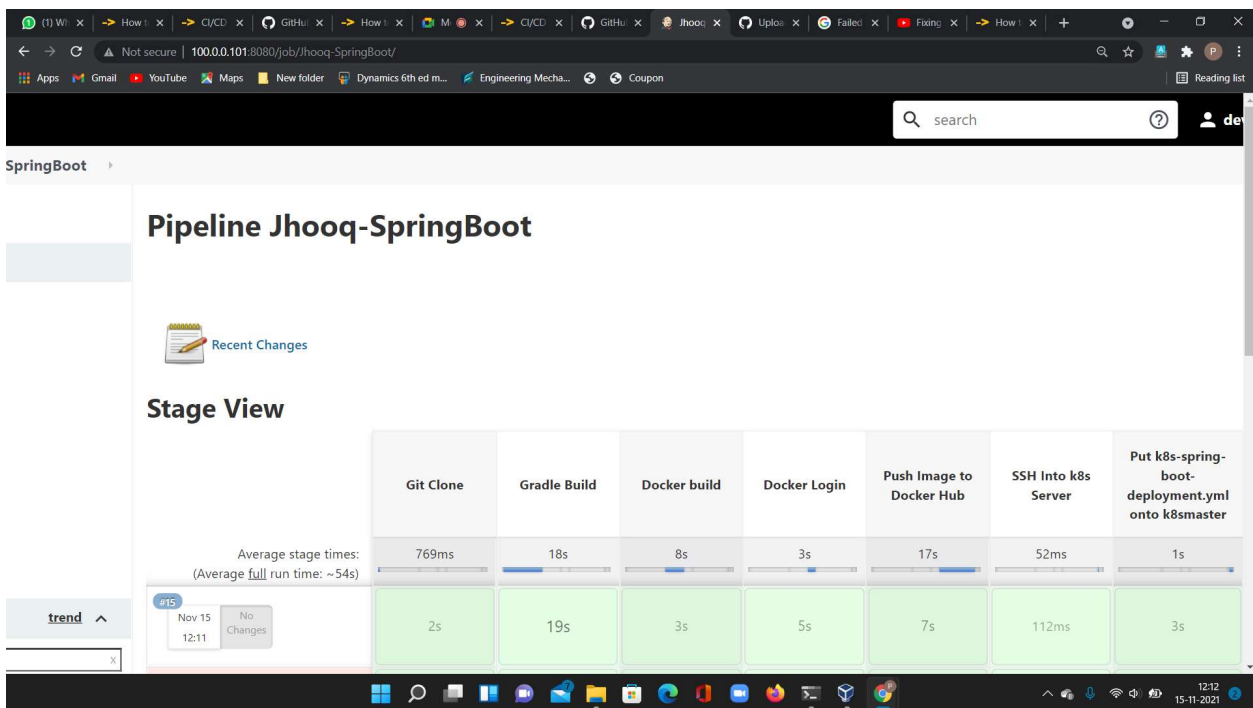
Code

Pipeline(stage6 with internal stages)

```
stage("SSH Into k8s Server") { def remote = [:] remote.name = 'K8S
master' remote.host = '100.0.0.2' remote.user = 'vagrant' remote.password
= 'vagrant' remote.allowAnyHosts = true
    stage('Put k8s-spring-boot-deployment.yml onto k8smaster') { sshPut
remote: remote, from: 'k8s-spring-boot-deployment.yml', into: '.' }
    stage('Deploy spring boot') { sshCommand remote: remote,
command: "kubectl apply -f k8s-spring-boot-deployment.yml" } }
```



Pipeline:



After, Application is successfully deployed on to kubernetes through Jenkins CI/CD pipeline.

