DEVOPS (CS457)

Assignment 2 : Task 1

Setting up Complete CI/CD Jenkins Pipeline for Kubernetes

Submitted to

Dr. Uma S

Submitted by Team 1

Sumith Sai Budde (18BCS101)
Syed Sufyan Ahmed(18BCS103)
Shaik Fharook (18BCS091)
Parvati Jayakumar (18BEC036)
P Chethan Krishna (18BEC040)
G Rithika (18BCS031)

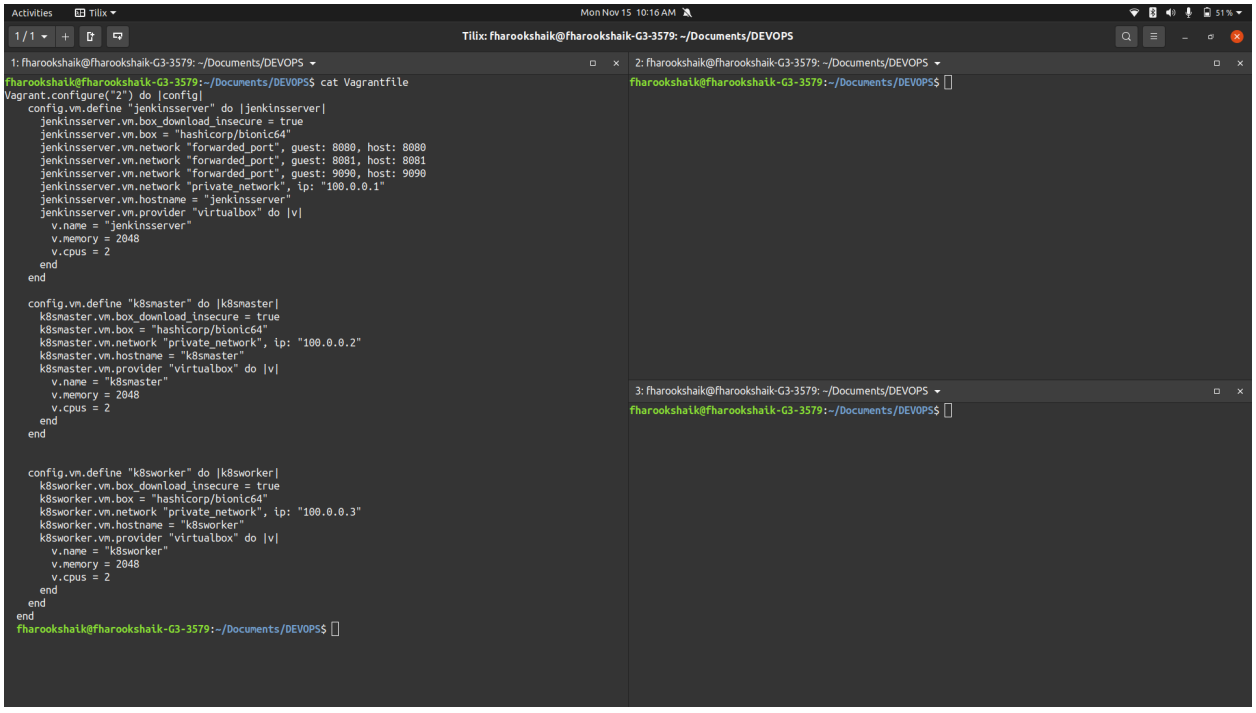
Software Requirements

- Vagrant
- Oracle VirtualBox 6.1.26 (for Ubuntu)
- Jenkins, Docker, Kubernetes, Java 11, Gradle, GitHub, Docker Hub (Configured in VM)

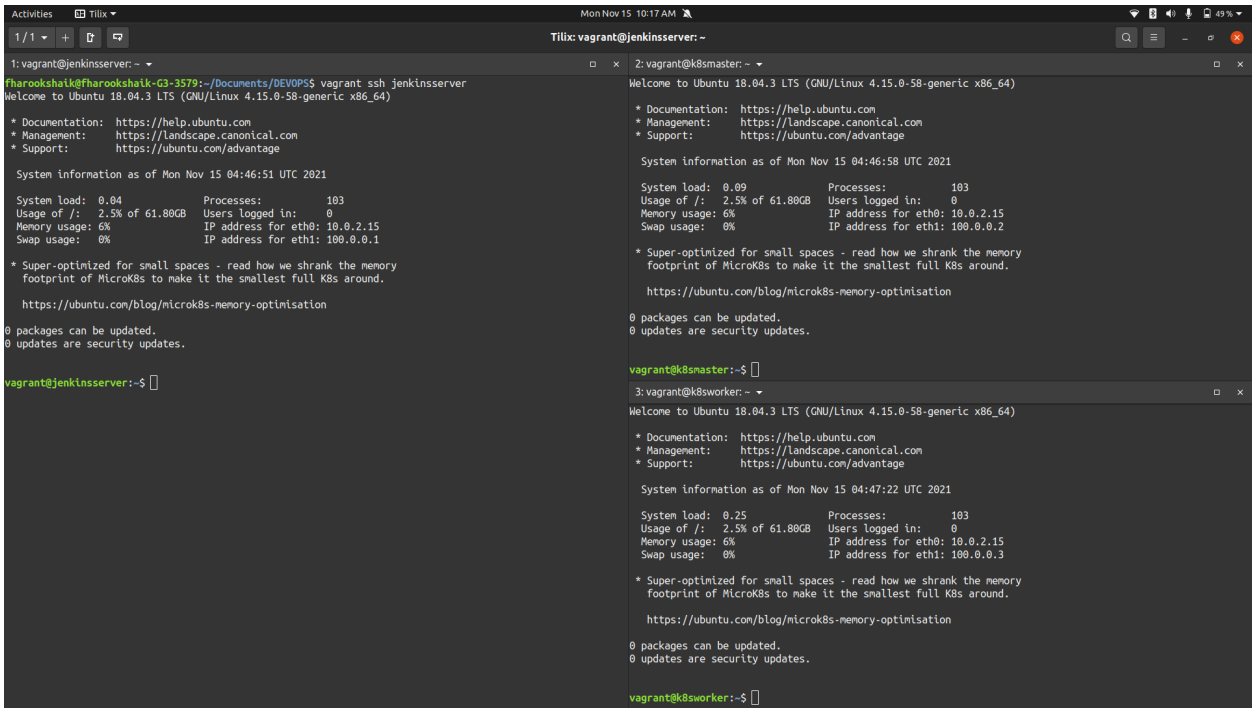
Setting Up Kubernetes Cluster Using kubespray

Step 1: Setting up pre-configured Virtual Machines using Vagrant

- We Create 3 VM's with its own predefined IP Address
    - jenkinsserver - 100.0.0.1 - 2 CPU with 2 GB RAM
    - k8smaster - 100.0.0.2 - 2 CPU with 2 GB RAM
    - k8sworker - 100.0.0.3 - 2 CPU with 2 GB RAM
- Using `vagrant up` we create and boot the preconfigured VM's in headless mode.



- Using vagrant ssh <vm_name> we can login to the respective VM.

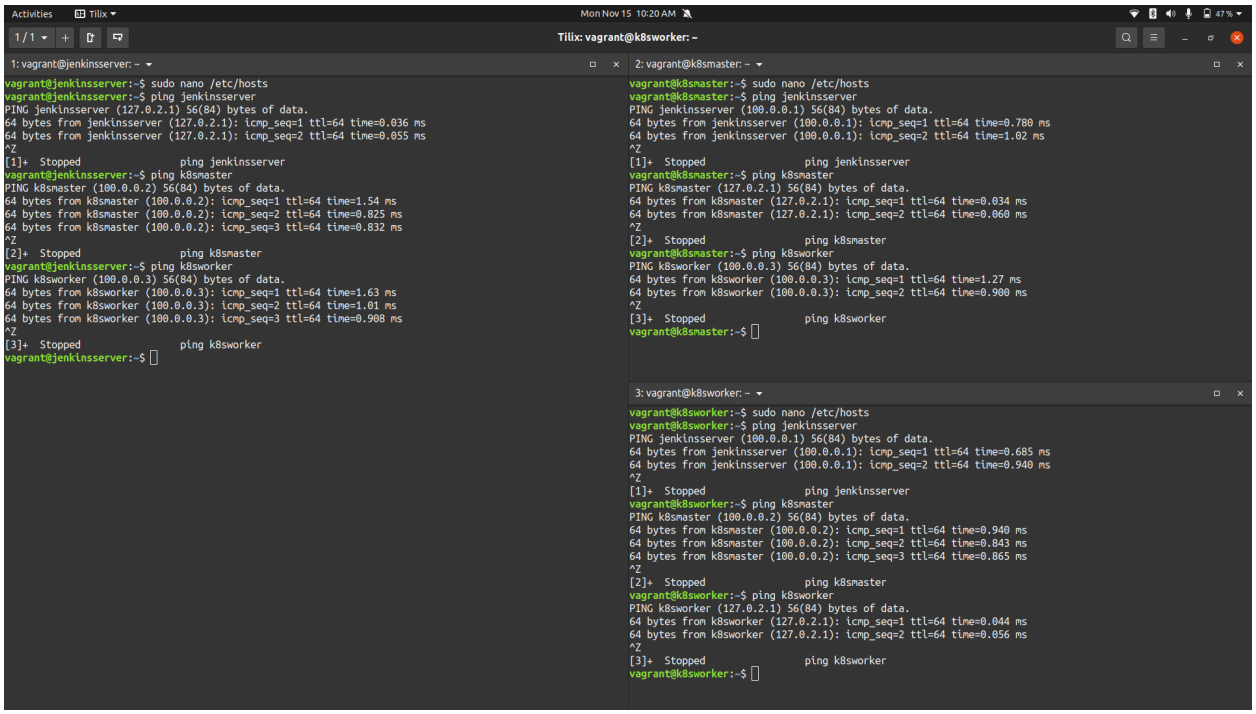


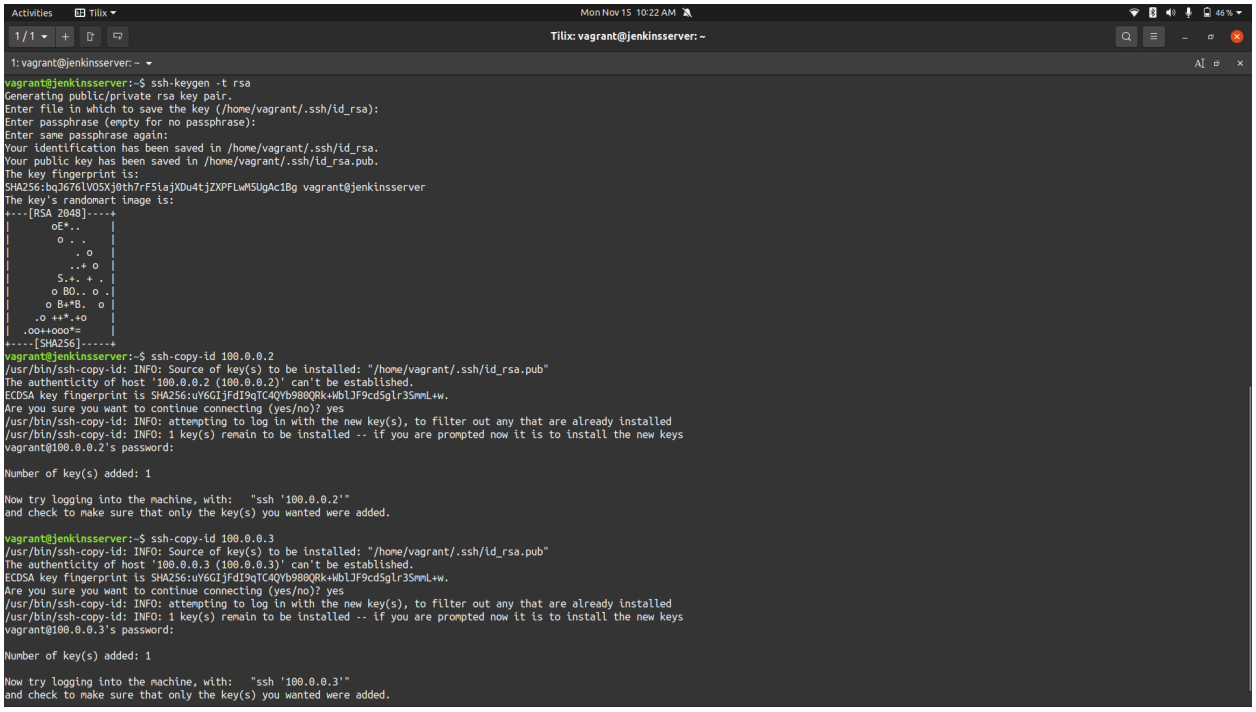
Step 2: Updating /etc/hosts on all nodes & Adding ssh key from jenkinsserver to other nodes

- The following code is appended to all the three nodes in `/etc/hosts`

```
100.0.0.1 jenkinsserver jenkinsserver
100.0.0.2 k8smaster k8smaster
100.0.0.3 k8sworker k8sworker
```

- An ssh key is generated in jenkinsserver and added to k8smaster & k8sworker to access the other nodes directly from jenkinsserver.


Activities
Tilix
Mon Nov 15 10:20 AM
47 %
1 / 1
Tilix: vagrant@k8sworker: ~
1: vagrant@jenkinsserver: ~
vagrant@jenkinsserver:~$ sudo nano /etc/hosts
vagrant@jenkinsserver:~$ ping jenkinsserver
PING jenkinsserver (127.0.2.1) 56(84) bytes of data.
64 bytes from jenkinsserver (127.0.2.1): icmp_seq=1 ttl=64 time=0.036 ms
64 bytes from jenkinsserver (127.0.2.1): icmp_seq=2 ttl=64 time=0.055 ms
^Z
[1]+ Stopped ping jenkinsserver
vagrant@jenkinsserver:~$ ping k8smaster
PING k8smaster (100.0.0.2) 56(84) bytes of data.
64 bytes from k8smaster (100.0.0.2): icmp_seq=1 ttl=64 time=1.54 ms
64 bytes from k8smaster (100.0.0.2): icmp_seq=2 ttl=64 time=0.825 ms
64 bytes from k8smaster (100.0.0.2): icmp_seq=3 ttl=64 time=0.832 ms
^Z
[2]+ Stopped ping k8smaster
vagrant@jenkinsserver:~$ ping k8sworker
PING k8sworker (100.0.0.3) 56(84) bytes of data.
64 bytes from k8sworker (100.0.0.3): icmp_seq=1 ttl=64 time=1.63 ms
64 bytes from k8sworker (100.0.0.3): icmp_seq=2 ttl=64 time=1.01 ms
64 bytes from k8sworker (100.0.0.3): icmp_seq=3 ttl=64 time=0.908 ms
^Z
[3]+ Stopped ping k8sworker
vagrant@jenkinsserver:~$
2: vagrant@k8smaster: ~
vagrant@k8smaster:~$ sudo nano /etc/hosts
vagrant@k8smaster:~$ ping jenkinsserver
PING jenkinsserver (100.0.0.1) 56(84) bytes of data.
64 bytes from jenkinsserver (100.0.0.1): icmp_seq=1 ttl=64 time=0.780 ms
64 bytes from jenkinsserver (100.0.0.1): icmp_seq=2 ttl=64 time=1.02 ms
^Z
[1]+ Stopped ping jenkinsserver
vagrant@k8smaster:~$ ping k8smaster
PING k8smaster (127.0.2.1) 56(84) bytes of data.
64 bytes from k8smaster (127.0.2.1): icmp_seq=1 ttl=64 time=0.034 ms
64 bytes from k8smaster (127.0.2.1): icmp_seq=2 ttl=64 time=0.060 ms
^Z
[2]+ Stopped ping k8smaster
vagrant@k8smaster:~$ ping k8sworker
PING k8sworker (100.0.0.3) 56(84) bytes of data.
64 bytes from k8sworker (100.0.0.3): icmp_seq=1 ttl=64 time=1.27 ms
64 bytes from k8sworker (100.0.0.3): icmp_seq=2 ttl=64 time=0.900 ms
^Z
[3]+ Stopped ping k8sworker
vagrant@k8smaster:~$
3: vagrant@k8sworker: ~
vagrant@k8sworker:~$ sudo nano /etc/hosts
vagrant@k8sworker:~$ ping jenkinsserver
PING jenkinsserver (100.0.0.1) 56(84) bytes of data.
64 bytes from jenkinsserver (100.0.0.1): icmp_seq=1 ttl=64 time=0.685 ms
64 bytes from jenkinsserver (100.0.0.1): icmp_seq=2 ttl=64 time=0.940 ms
^Z
[1]+ Stopped ping jenkinsserver
vagrant@k8sworker:~$ ping k8smaster
PING k8smaster (100.0.0.2) 56(84) bytes of data.
64 bytes from k8smaster (100.0.0.2): icmp_seq=1 ttl=64 time=0.940 ms
64 bytes from k8smaster (100.0.0.2): icmp_seq=2 ttl=64 time=0.843 ms
64 bytes from k8smaster (100.0.0.2): icmp_seq=3 ttl=64 time=0.865 ms
^Z
[2]+ Stopped ping k8smaster
vagrant@k8sworker:~$ ping k8sworker
PING k8sworker (127.0.2.1) 56(84) bytes of data.
64 bytes from k8sworker (127.0.2.1): icmp_seq=1 ttl=64 time=0.044 ms
64 bytes from k8sworker (127.0.2.1): icmp_seq=2 ttl=64 time=0.056 ms
^Z
[3]+ Stopped ping k8sworker
vagrant@k8sworker:~$

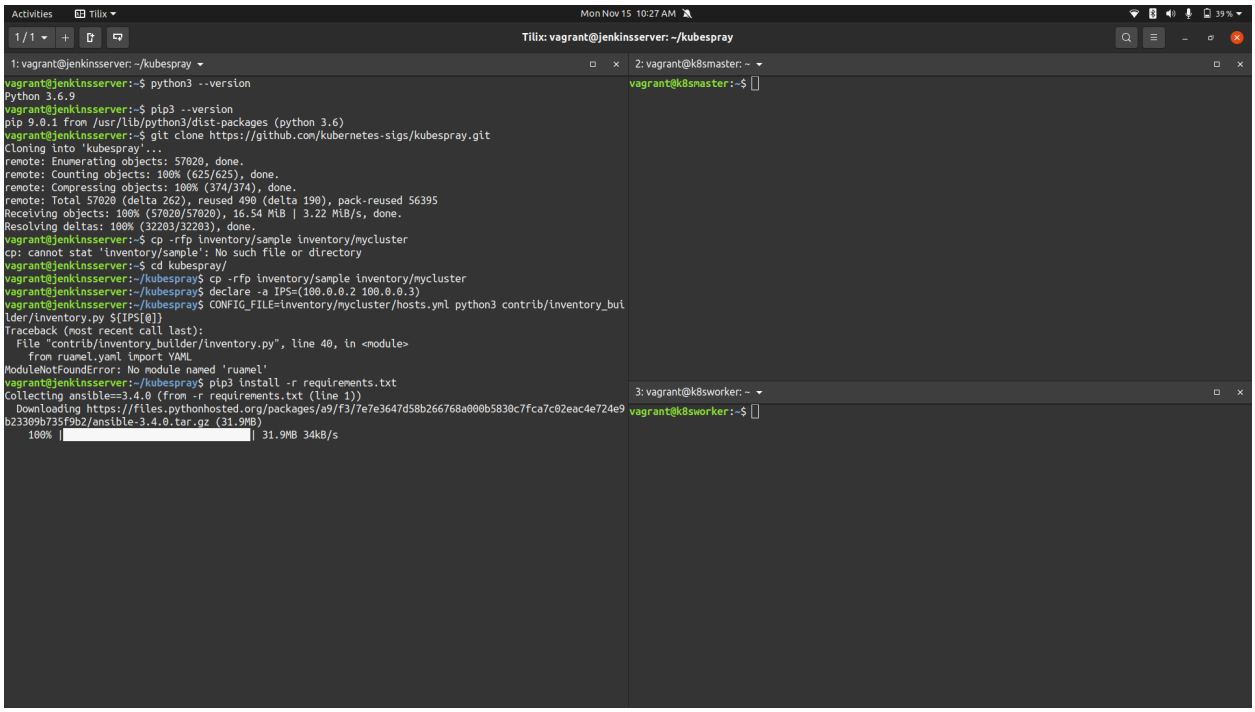

Activities
Tilix
Mon Nov 15 10:22 AM
46 %
1 / 1
Tilix: vagrant@jenkinsserver: ~
1: vagrant@jenkinsserver: ~
vagrant@jenkinsserver:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/vagrant/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/vagrant/.ssh/id_rsa.
Your public key has been saved in /home/vagrant/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:bqJ676lVO5Xj0th7rF5iajXDu4tjZXPFLwM5UgAc1Bg vagrant@jenkinsserver
The key's randomart image is:
+---[RSA 2048]----+
| oE*.. |
| o . . |
| . o |
| ..+ o |
| S.+. + . |
| o BO.. o .|
| o B+*B. o |
| .o ++*.+o |
| .oo++ooo*= |
+----[SHA256]-----+
vagrant@jenkinsserver:~$ ssh-copy-id 100.0.0.2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/vagrant/.ssh/id_rsa.pub"
The authenticity of host '100.0.0.2 (100.0.0.2)' can't be established.
ECDSA key fingerprint is SHA256:uY6GIjFdI9qTC4QYb980QRk+WblJF9cd5glr35mmL+w.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
vagrant@100.0.0.2's password:
Number of key(s) added: 1
Now try logging into the machine, with: "ssh '100.0.0.2'"
and check to make sure that only the key(s) you wanted were added.
vagrant@jenkinsserver:~$ ssh-copy-id 100.0.0.3
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/vagrant/.ssh/id_rsa.pub"
The authenticity of host '100.0.0.3 (100.0.0.3)' can't be established.
ECDSA key fingerprint is SHA256:uY6GIjFdI9qTC4QYb980QRk+WblJF9cd5glr35mmL+w.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
vagrant@100.0.0.3's password:
Number of key(s) added: 1
Now try logging into the machine, with: "ssh '100.0.0.3'"
and check to make sure that only the key(s) you wanted were added.

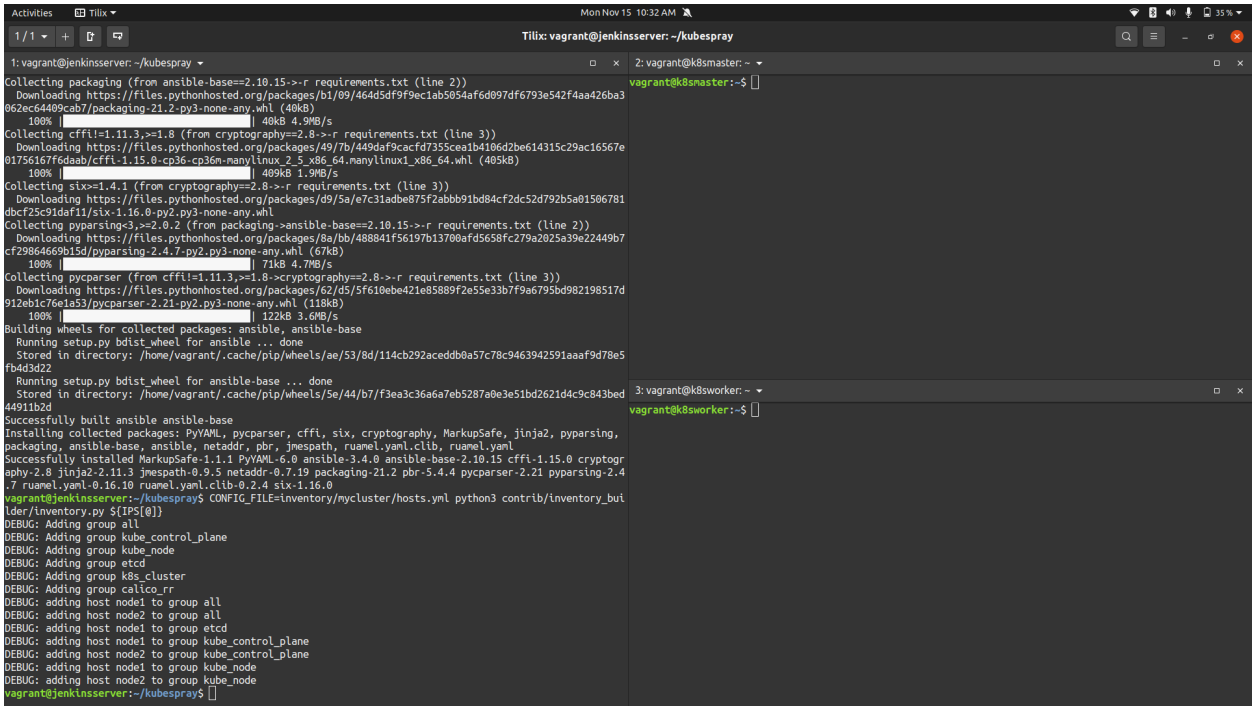
Step 3: Cloning kubespray git repository and setting up kubespray

- We setup the kubespray in jenkinsserver using the following commands
- Clone the git repository.

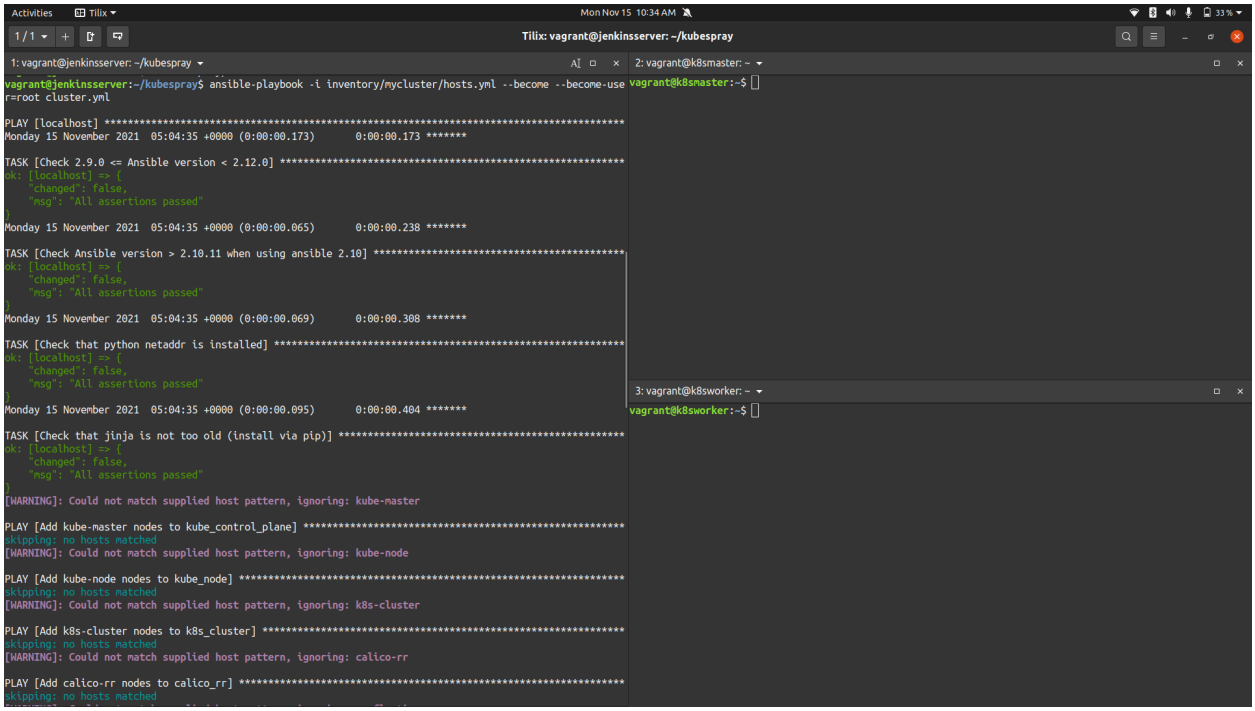
```
git clone https://github.com/kubernetes-sigs/kubespray.git
```

- Install the requirements using `sudo pip3 install -r requirements.txt`
- Copy the inventory file to current users using `cp -rfp inventory/sample inventory/mycluster`
- Declare the k8smaster k8sworker using `declare -a IPS=(100.0.0.2 100.0.0.3)`
- And prepare the host.yml for ansible using `CONFIG_FILE=inventory/mycluster/hosts.yml python3 contrib/inventory_builder/inventory.py ${IPS[@]}`

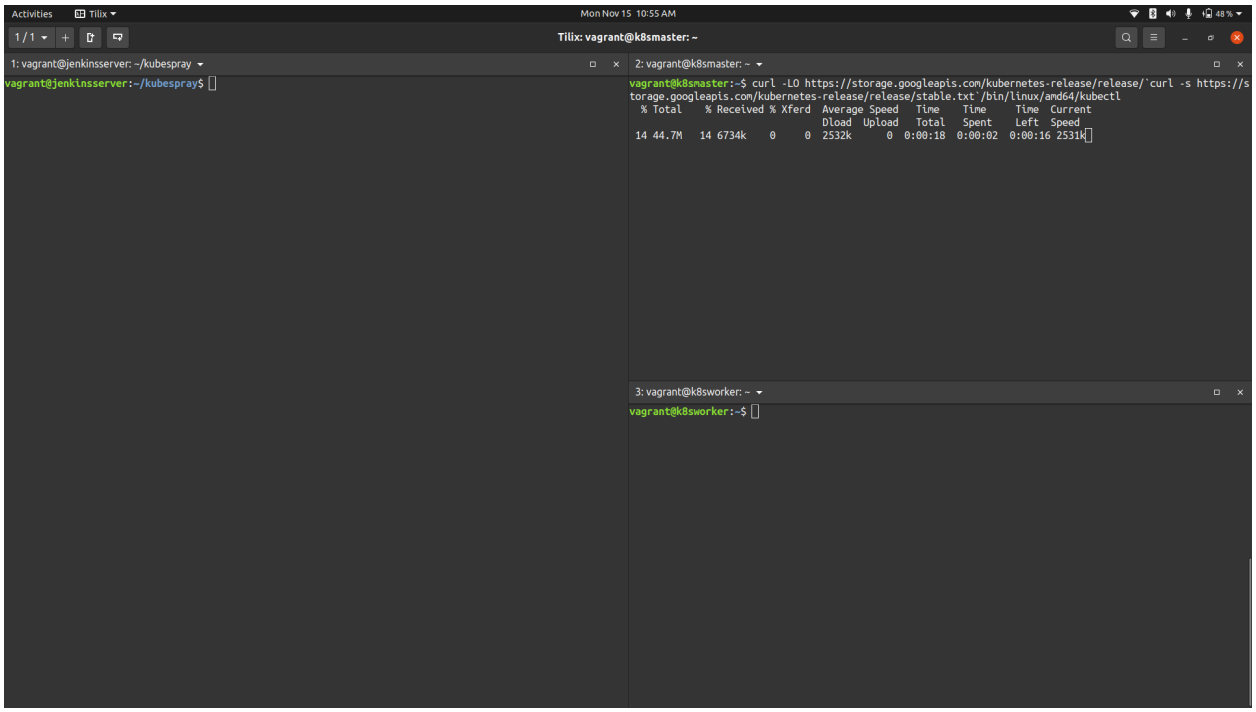


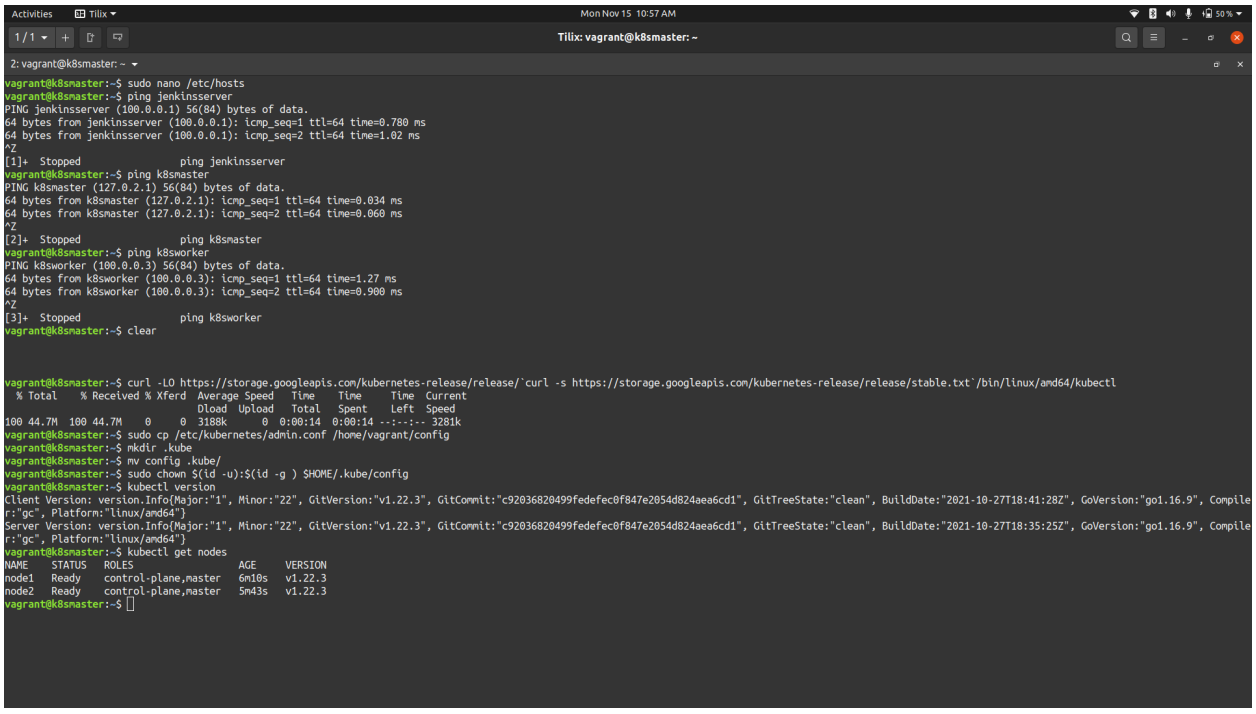


- Run the ansible playbook to setup the kubernetes cluster on k8smaster and k8sworker using kubespray from jenkinsserver



Step 4: Install kubectl on k8smaster

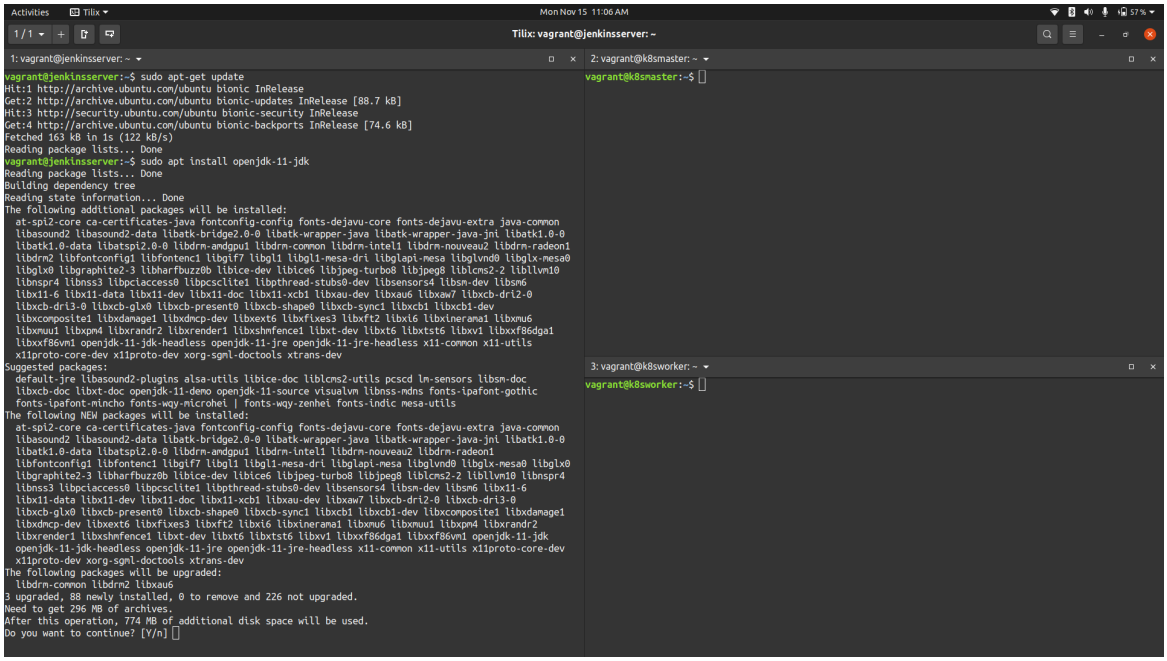




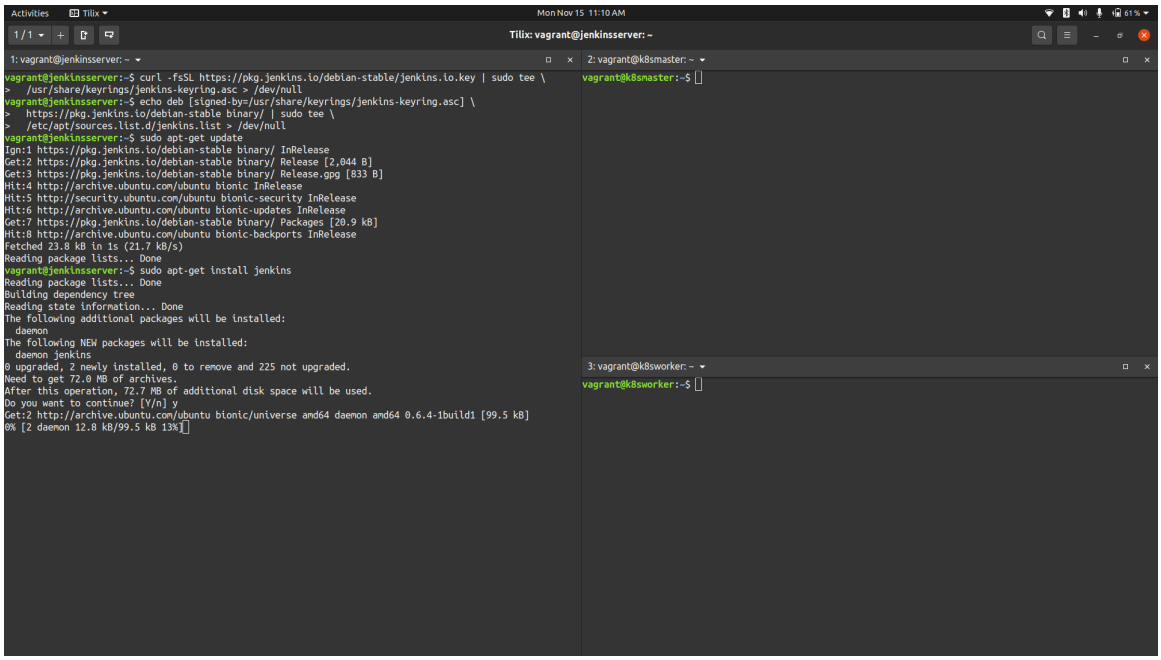
Setup CI/CD Jenkins Kubernetes Pipeline

Step 5: Install and configure Jenkins on jenkinsserver

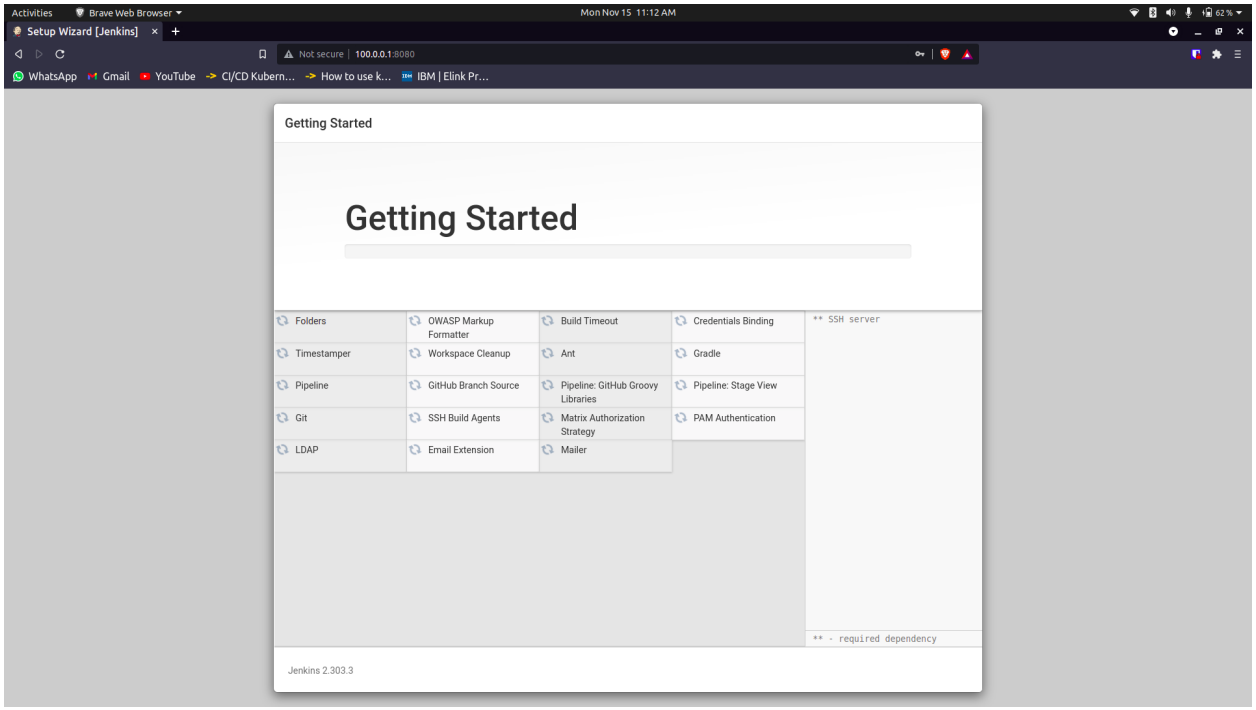
- Update the repositories and install openjdk-11-jdk

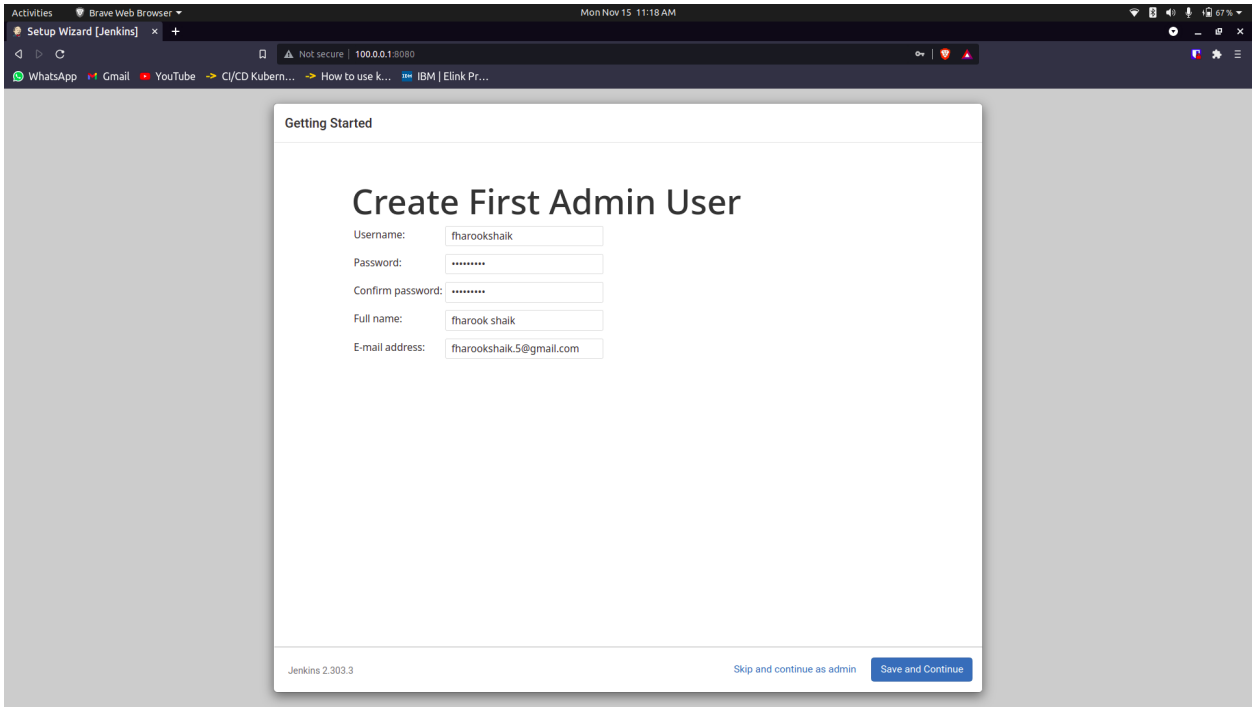


- Install Jenkins

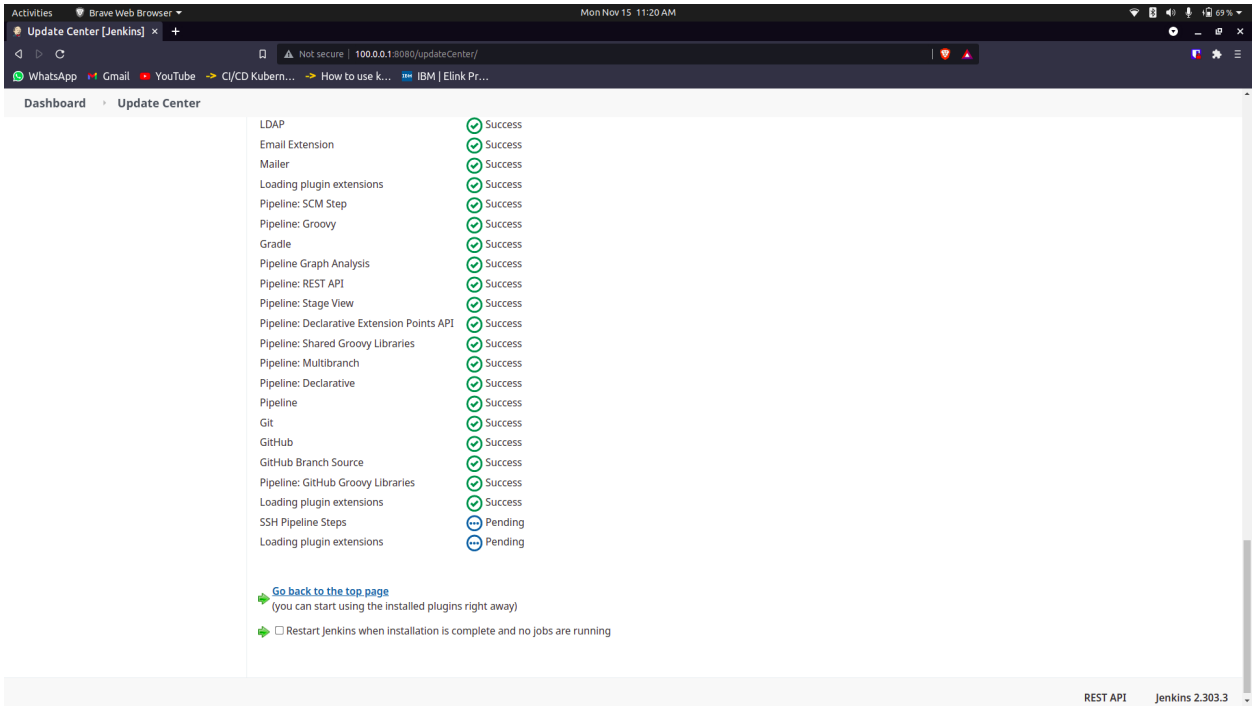


- Setup Jenkins with default configuration and create a user

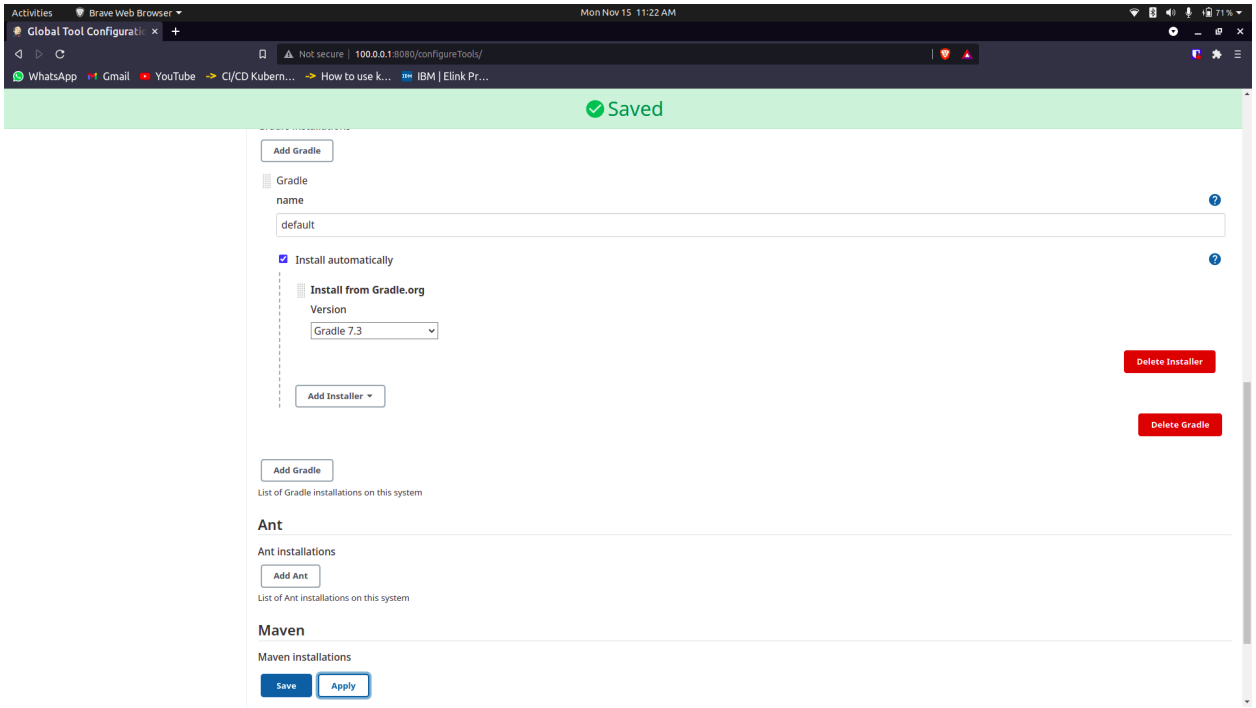




- Install SSH Pipeline Steps Plugin in Jenkins

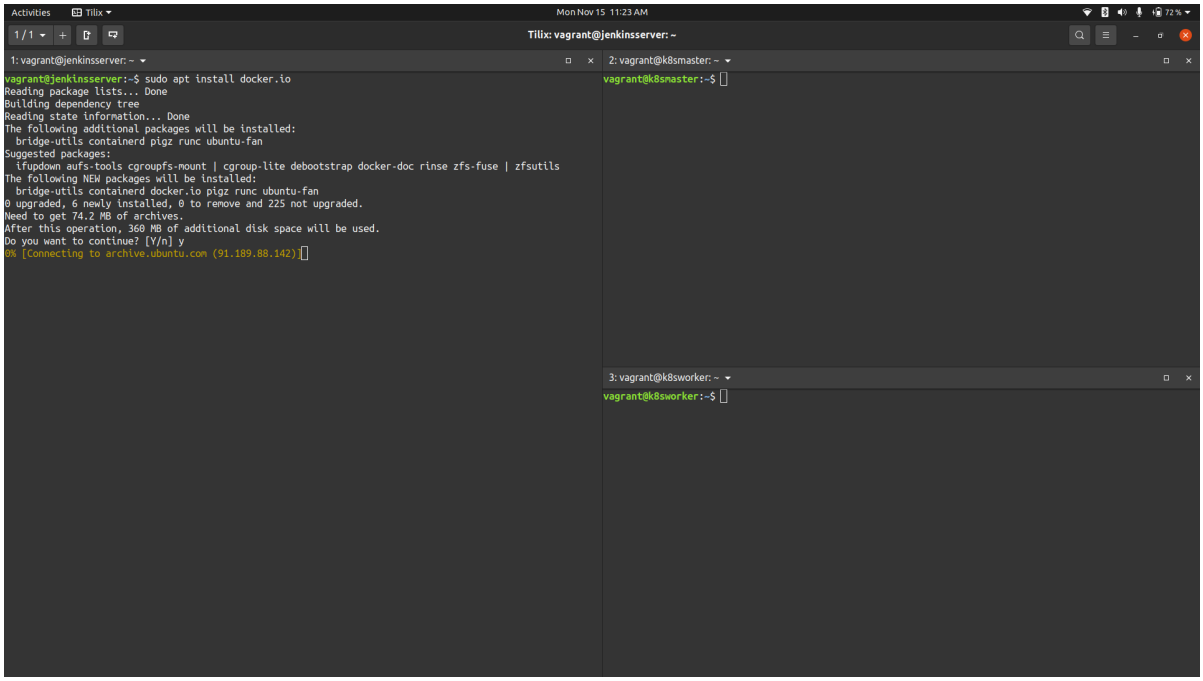


- As we're deploying a java spring boot application, we need few things to be set up like gradle.

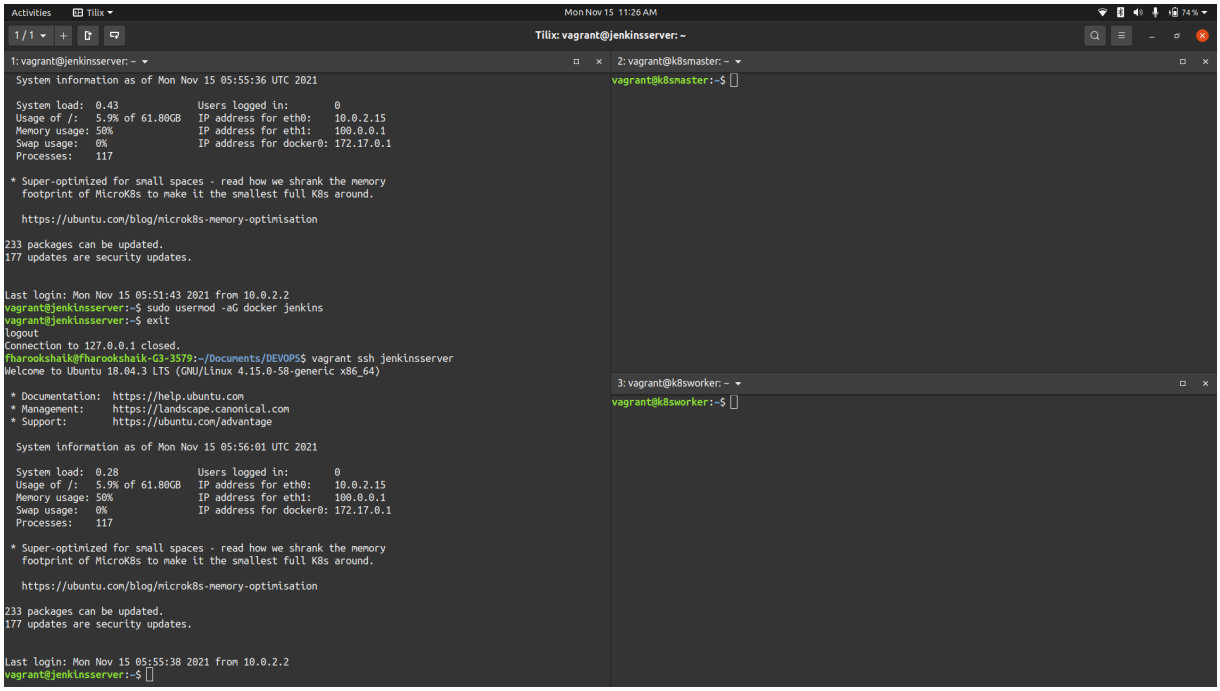


Step 6: Install and configure Docker in jenkinsserver

- Install docker io

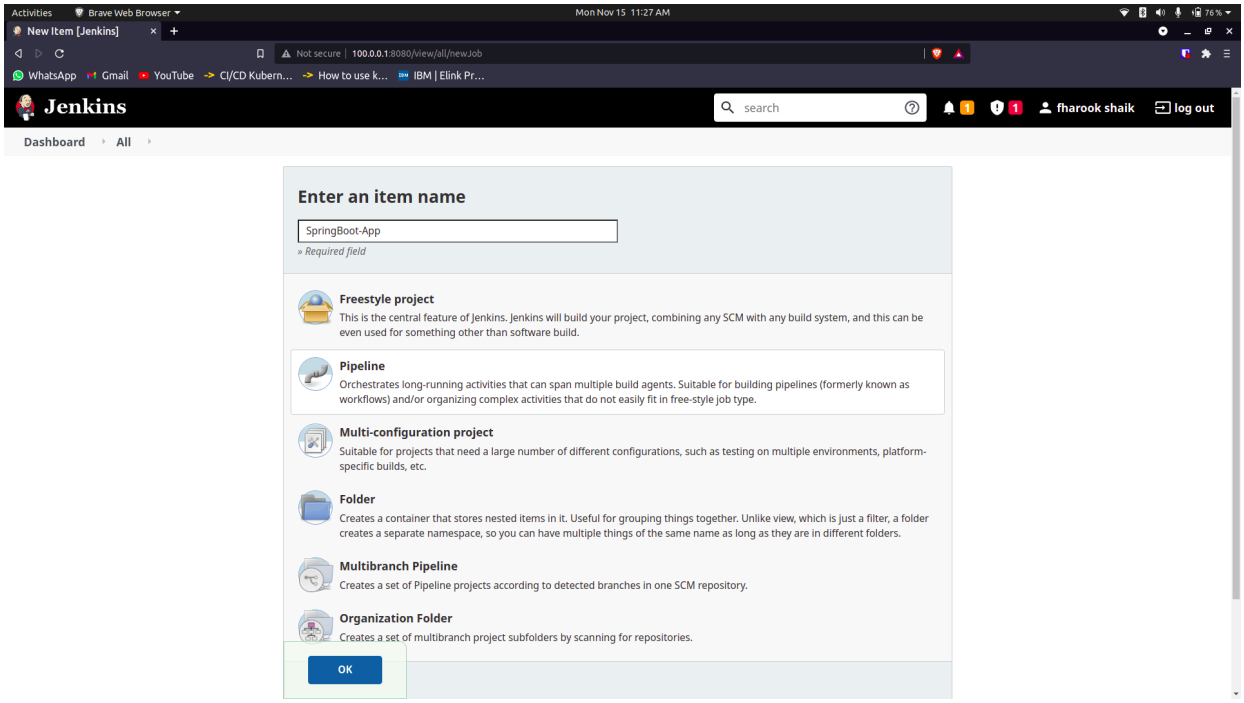


- Add docker & jenkins to usermod group

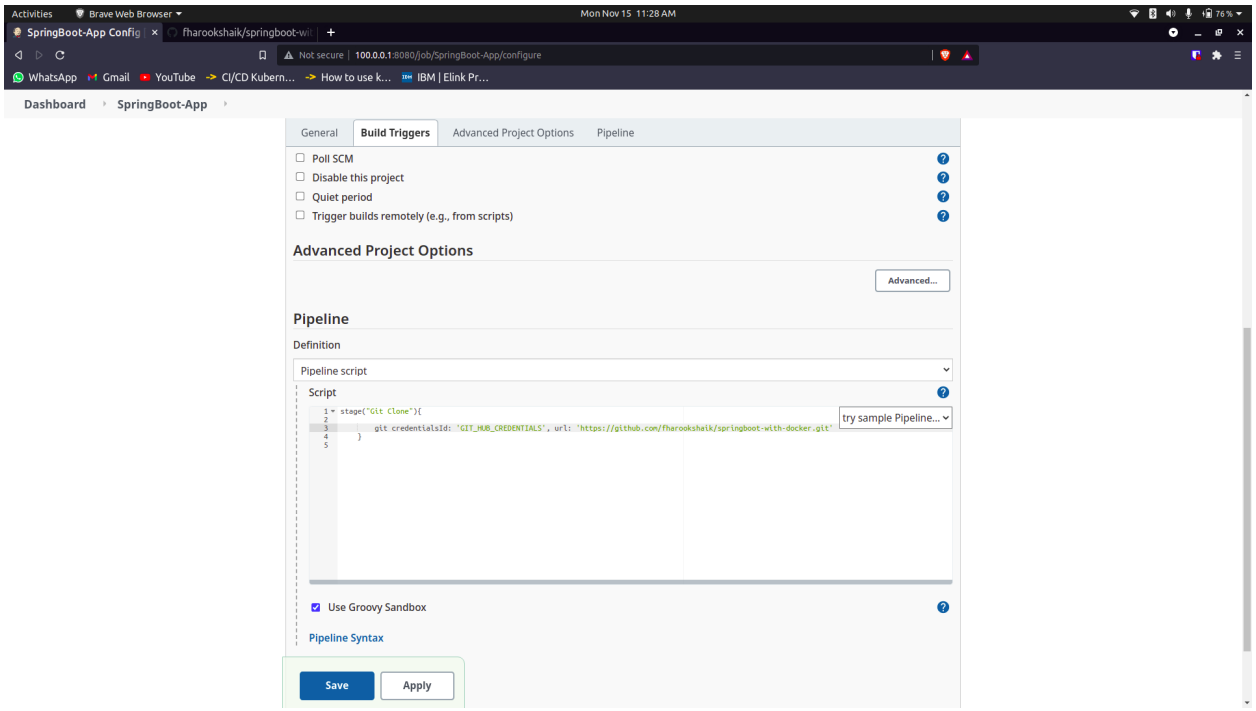


Step 7: Create a new Pipeline for our application in jenkins

- We name our application as **SpringBoot-App** and the type is a pipeline.



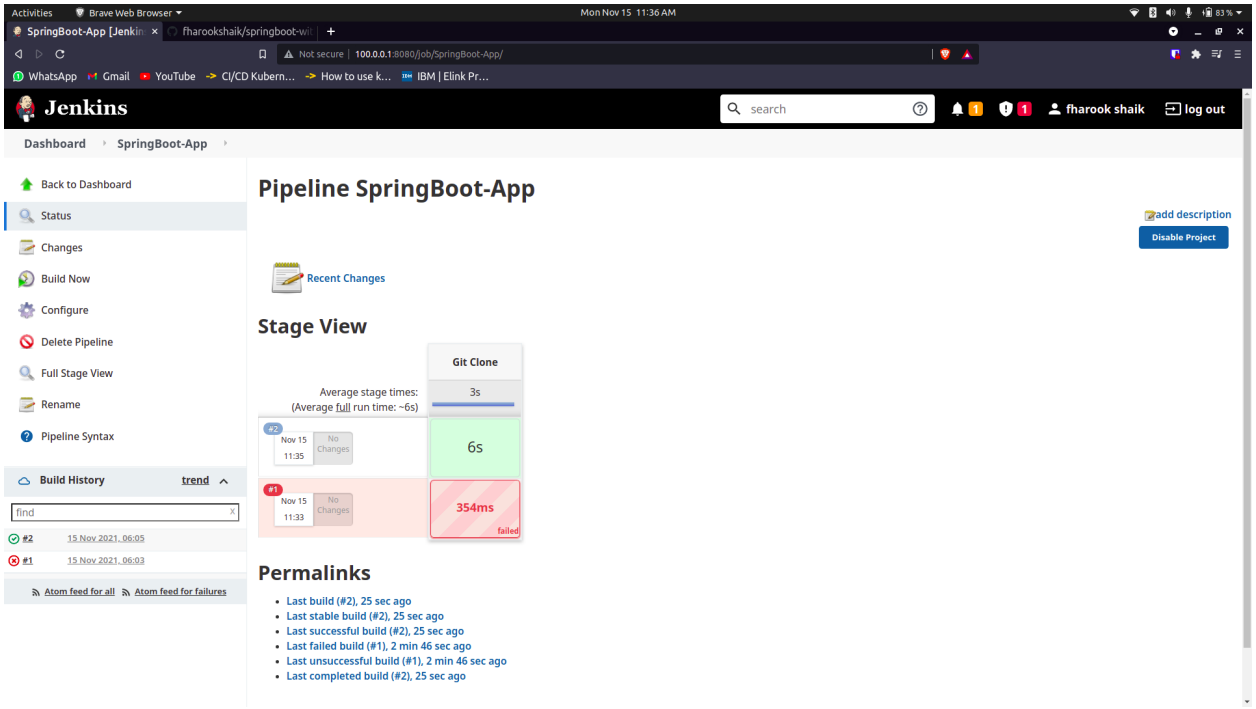
- Since the architecture of our application contains active pulling of repository from GitHub and push the docker file to DockerHub, We configure GitHub and Docker Hub Credentials in jenkins



Step 8: Building and Testing Pipeline

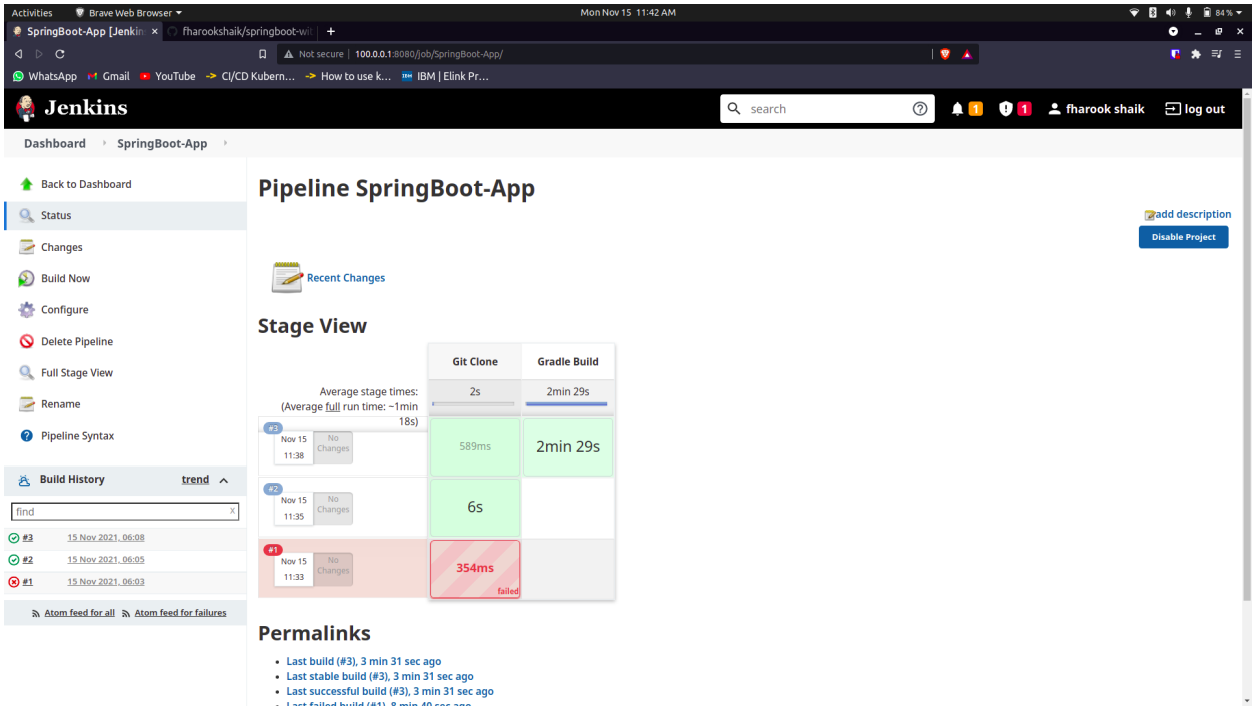
- The first step in the pipeline is to pull the Project repository from GitHub. With the credentials already being stored in Jenkins, we use the following command to automate this step.

```
stage("Git Clone"){

        git credentialsId: 'GIT_HUB_CREDENTIALS', url:
'https://github.com/fharookshaik/springboot-with-docker.git'
    }
```

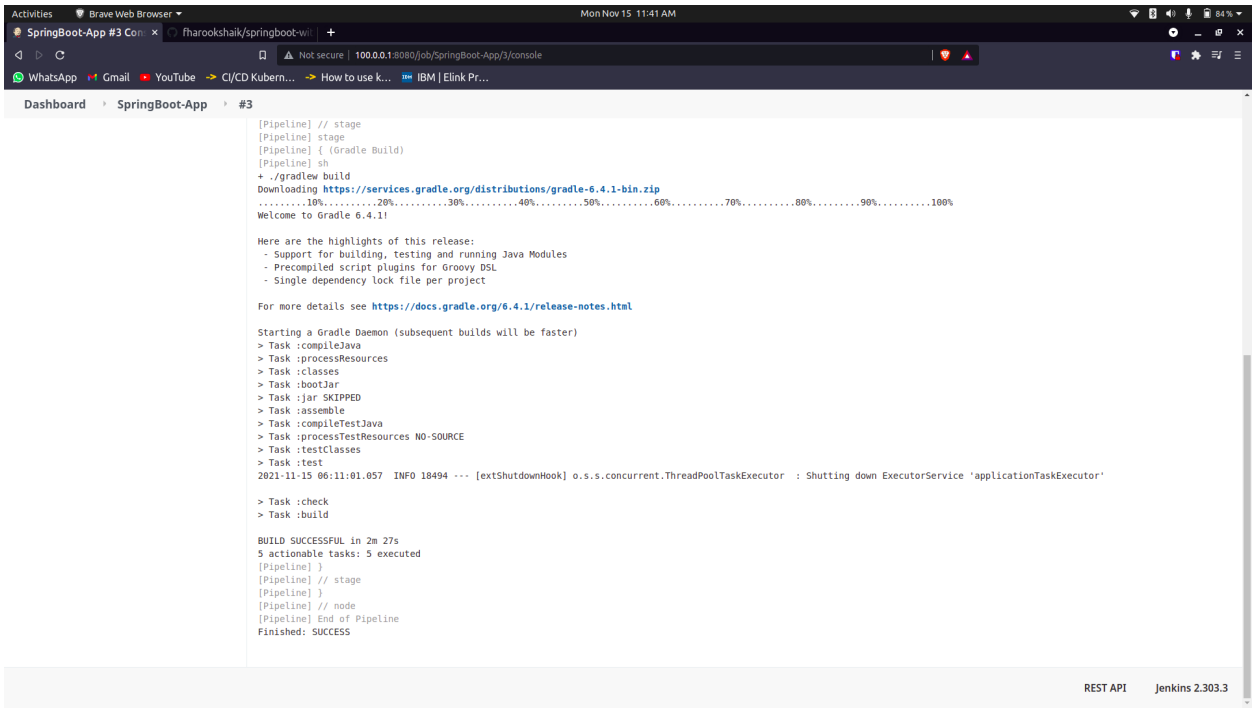


- The Second step in the pipeline is to build the project using gradle. As we've already configured the gradle in jenkins as well as in the project. The following command is used to add this step into the pipeline.

```
stage('Gradle Build') {

      sh './gradlew build'

   }
```

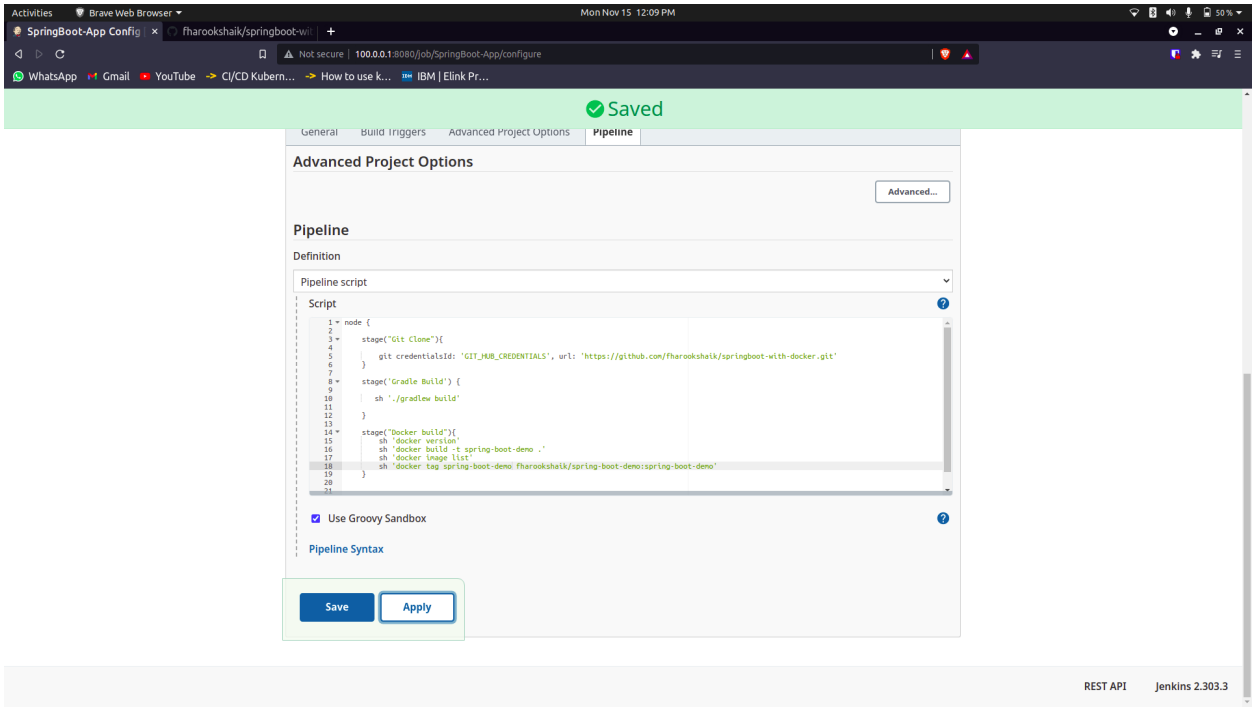


- The logs or output of the build can shown in the below picture



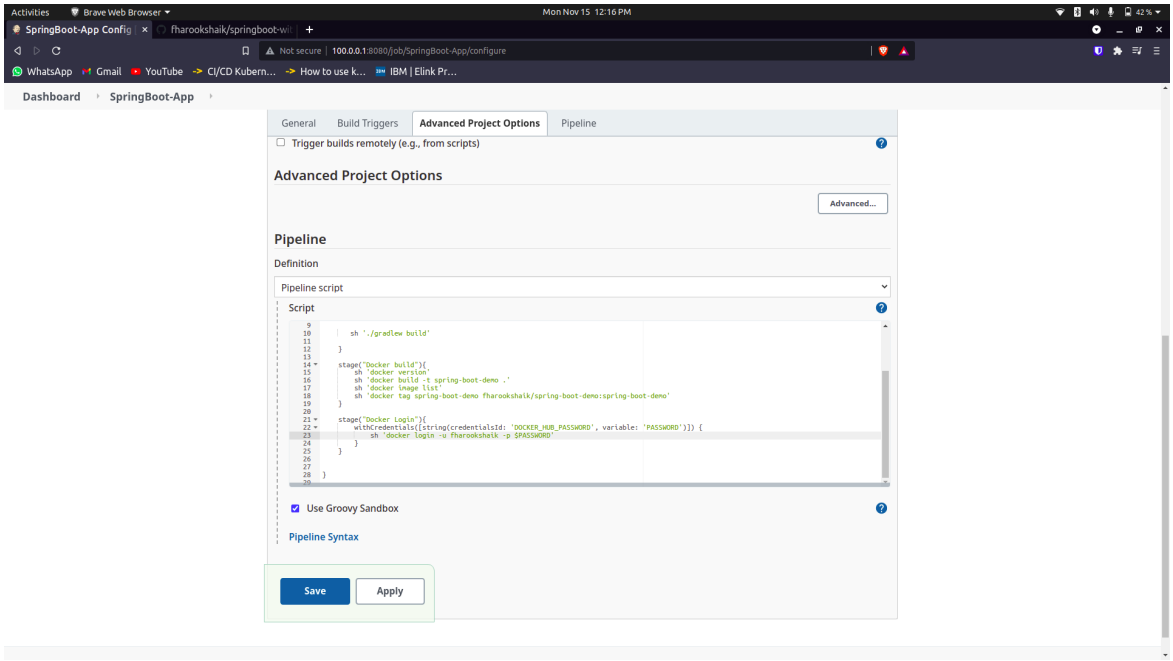
- The third step is to build a docker image of the application.

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



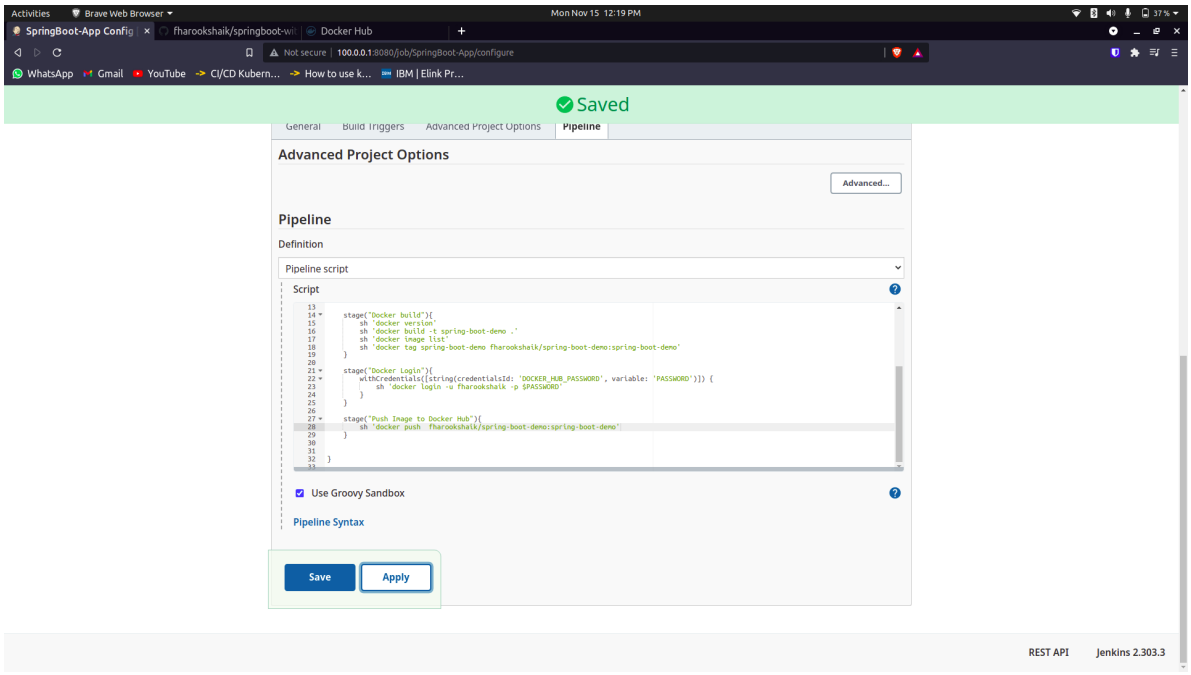
- The fourth step in the pipeline is to login into the Docker Hub for pushing the newly created docker image.

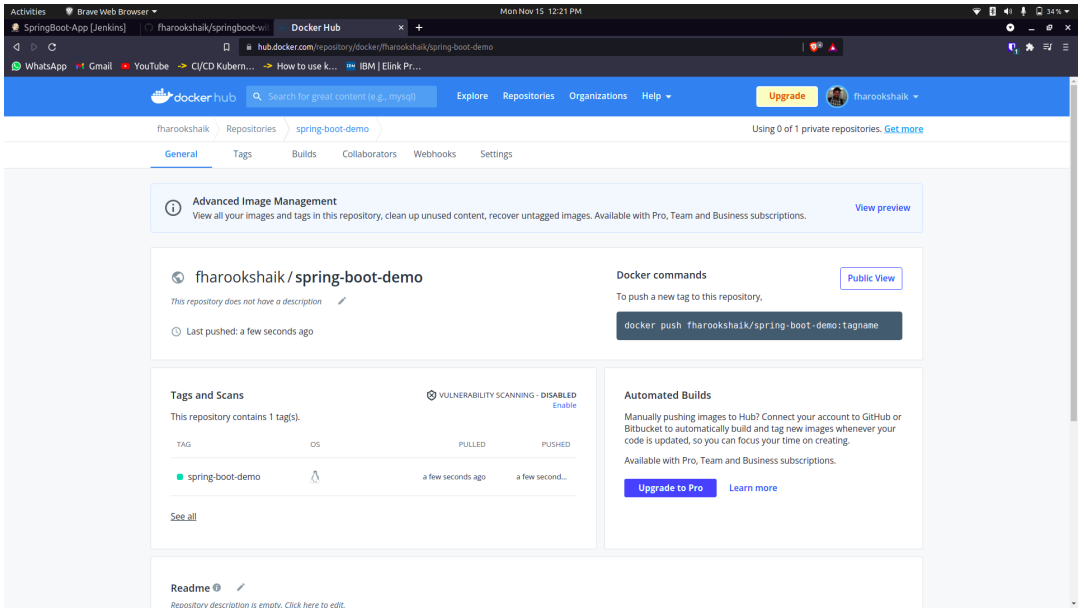
```
stage("Docker Login"){
        withCredentials([string(credentialsId: 'DOCKER_HUB_PASSWORD',
variable: 'PASSWORD')]) {
            sh 'docker login -u fharookshaik -p $PASSWORD'
        }
    }
```



- The fifth step in the pipeline is to push the docker image to Docker Hub with the stored credentials in Jenkins.

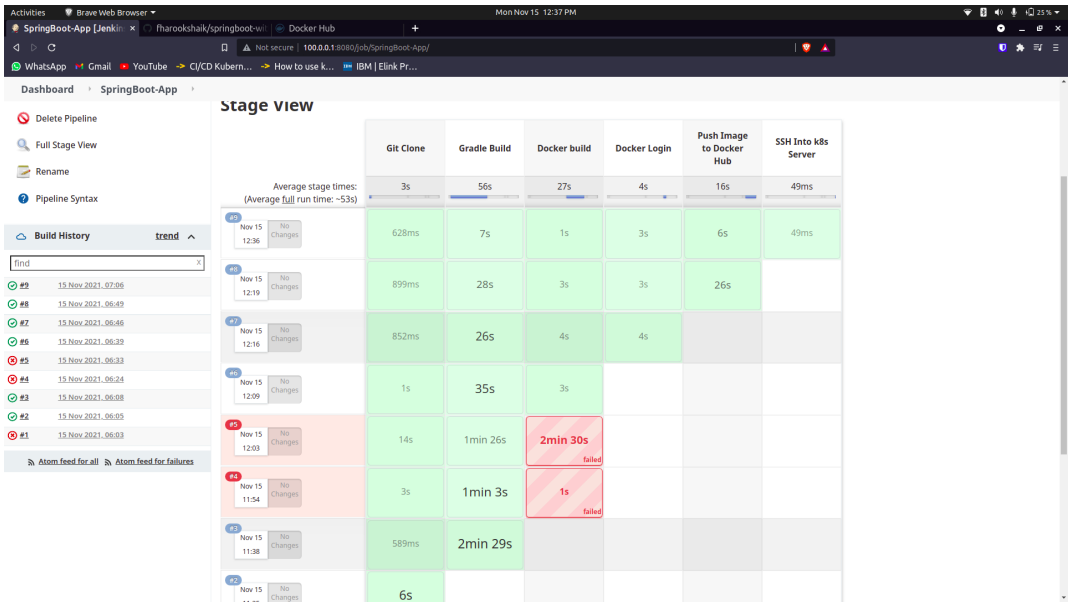
```
stage("Push Image to Docker Hub"){
        sh 'docker push  fharookshaik/spring-boot-demo:spring-boot-demo'
    }
```





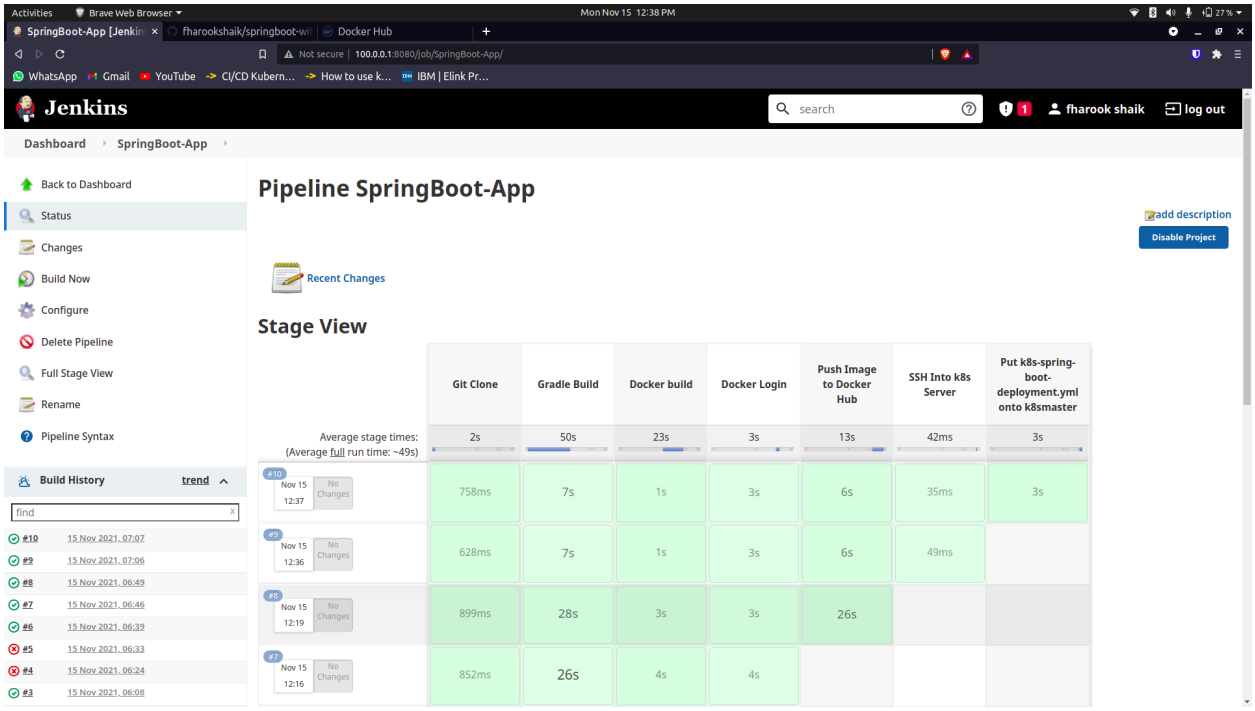
- The sixth step in the pipeline is to login into k8smaster from jenkinsserver via SSH.

```
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
}
```



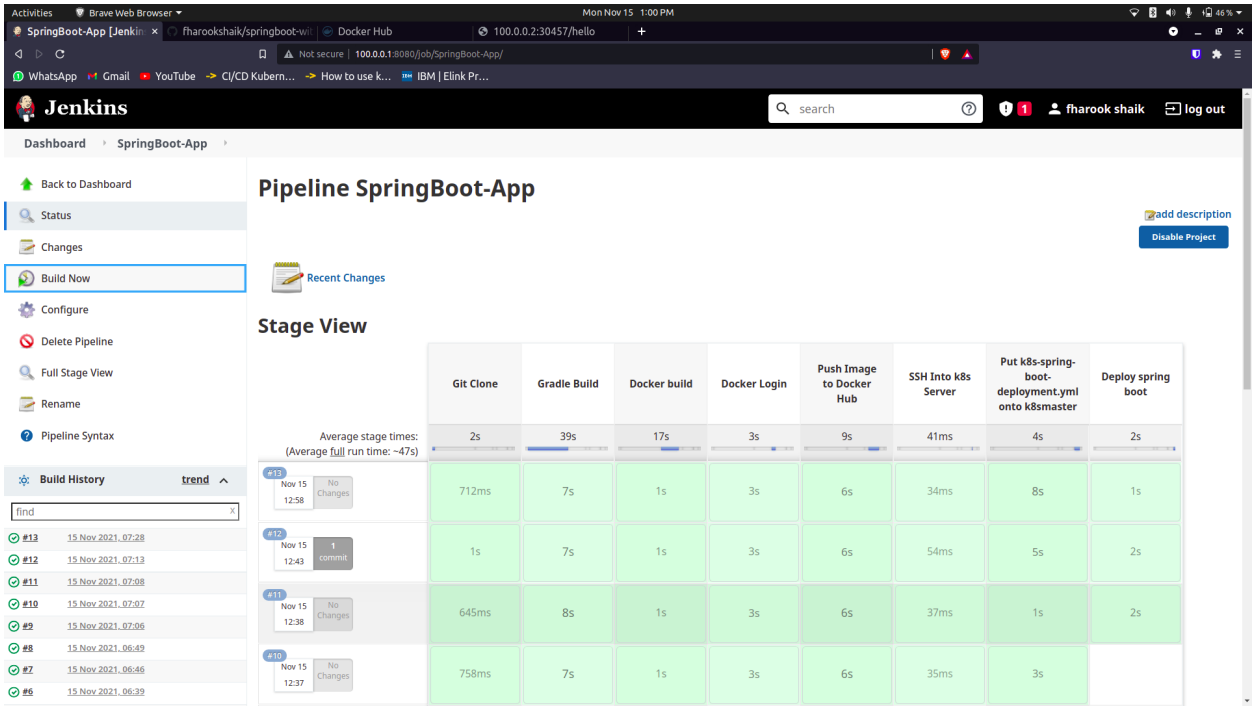
- The seventh step in the pipeline is to copy k8s-spring-boot-deployment.yml, a preconfigured kubernetes deployment file to k8smaster server.

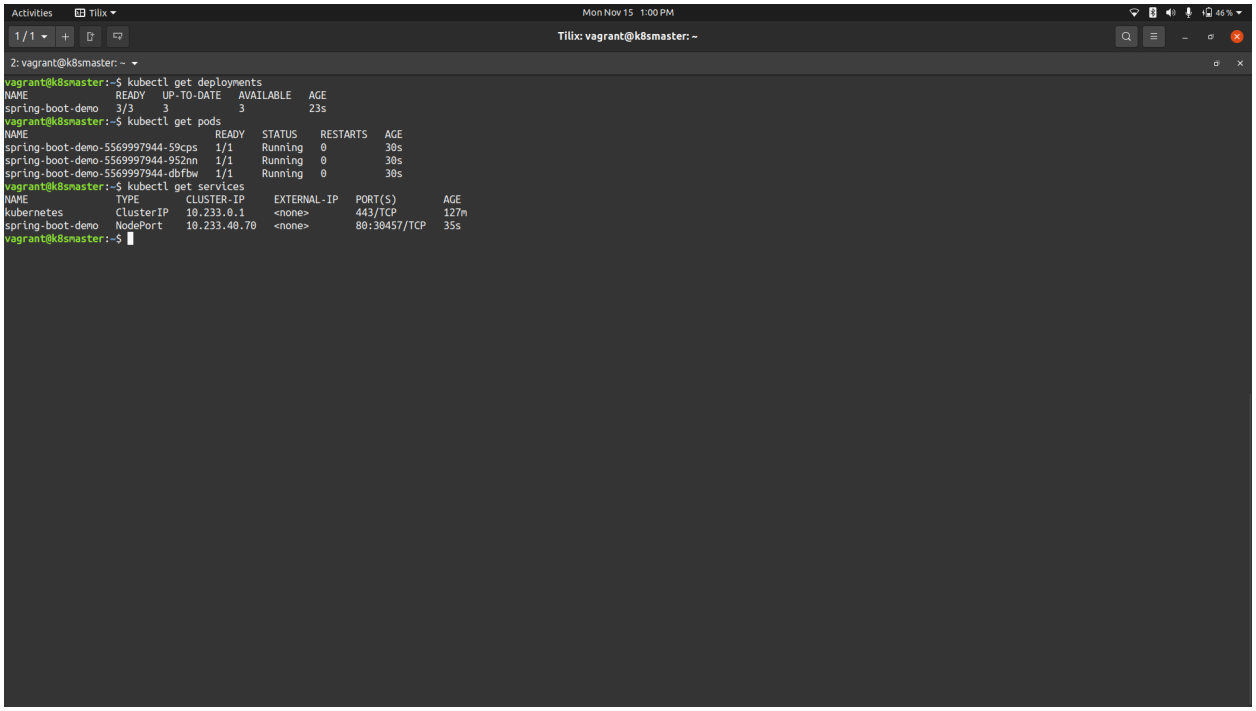
```
stage('Put k8s-spring-boot-deployment.yml onto k8smaster') {
            sshPut remote: remote, from: 'k8s-spring-boot-deployment.yml',
into: '.'
        }
```



- The final step in the pipeline is to deploy the spring boot application using kubectl in k8smaster.

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


Activities
Brave Web Browser
Mon Nov 15 1:00 PM
SpringBoot-App [Jenkins
fharookshaik/springboot-wit
Docker Hub
100.0.0.2:30457/hello
Not secure | 100.0.0.1:8080/job/SpringBoot-App/
WhatsApp
Gmail
YouTube
CI/CD Kubern...
How to use k...
IBM | Elink Pr...
Jenkins
search
fharook shaik
log out
Dashboard
SpringBoot-App
Back to Dashboard
Status
Changes
Build Now
Configure
Delete Pipeline
Full Stage View
Rename
Pipeline Syntax
Build History
trend
find
#13 15 Nov 2021, 07:28
#12 15 Nov 2021, 07:13
#11 15 Nov 2021, 07:08
#10 15 Nov 2021, 07:07
#9 15 Nov 2021, 07:06
#8 15 Nov 2021, 06:49
#7 15 Nov 2021, 06:46
#6 15 Nov 2021, 06:39
Pipeline SpringBoot-App
add description
Disable Project
Recent Changes
Stage View
Git Clone
Gradle Build
Docker build
Docker Login
Push Image to Docker Hub
SSH Into k8s Server
Put k8s-spring-boot-deployment.yml onto k8smaster
Deploy spring boot
Average stage times:
(Average full run time: ~47s)
2s
39s
17s
3s
9s
41ms
4s
2s
#13 Nov 15 12:58 No Changes
712ms
7s
1s
3s
6s
34ms
8s
1s
#12 Nov 15 12:43 1 commit
1s
7s
1s
3s
6s
54ms
5s
2s
#11 Nov 15 12:38 No Changes
645ms
8s
1s
3s
6s
37ms
1s
2s
#10 Nov 15 12:37 No Changes
758ms
7s
1s
3s
6s
35ms
3s


Activities
Tilix
Mon Nov 15 1:00 PM
1 / 1
Tilix: vagrant@k8smaster: ~
2: vagrant@k8smaster: ~
vagrant@k8smaster:~$ kubectl get deployments
NAME READY UP-TO-DATE AVAILABLE AGE
spring-boot-demo 3/3 3 3 23s
vagrant@k8smaster:~$ kubectl get pods
NAME READY STATUS RESTARTS AGE
spring-boot-demo-5569997944-59cps 1/1 Running 0 30s
spring-boot-demo-5569997944-952nn 1/1 Running 0 30s
spring-boot-demo-5569997944-dbfbw 1/1 Running 0 30s
vagrant@k8smaster:~$ kubectl get services
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
kubernetes ClusterIP 10.233.0.1 <none> 443/TCP 127m
spring-boot-demo NodePort 10.233.40.70 <none> 80:30457/TCP 35s
vagrant@k8smaster:~$

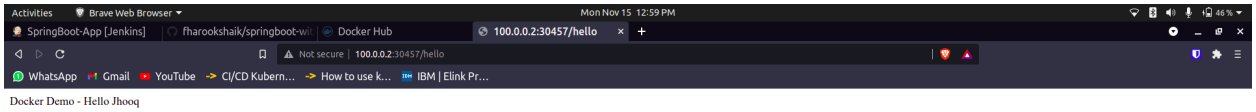
The total build command can be found below

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

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

    stage("Docker build"){
        sh 'docker version'
        sh 'docker build -t spring-boot-demo .'
        sh 'docker image list'
        sh 'docker tag spring-boot-demo
fharookshaik/spring-boot-demo:spring-boot-demo'
    }

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

    stage("Push Image to Docker Hub"){
        sh 'docker push  fharookshaik/spring-boot-demo:spring-boot-demo'
    }

    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"
        }
    }
}
```

Deployment

- The deployed spring boot application can be shown in http://100.0.0.2:30457/hello (Ports may vary for each deployment)



References

- https://jhooq.com/kubespray-12-steps-for-installing-a-production-ready-kubernetes-cluster/
- https://jhooq.com/ci-cd-jenkins-kubernetes/
- https://github.com/rahulwagh/springboot-with-docker