



INDIAN INSTITUTE OF  
INFORMATION  
TECHNOLOGY

## 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)

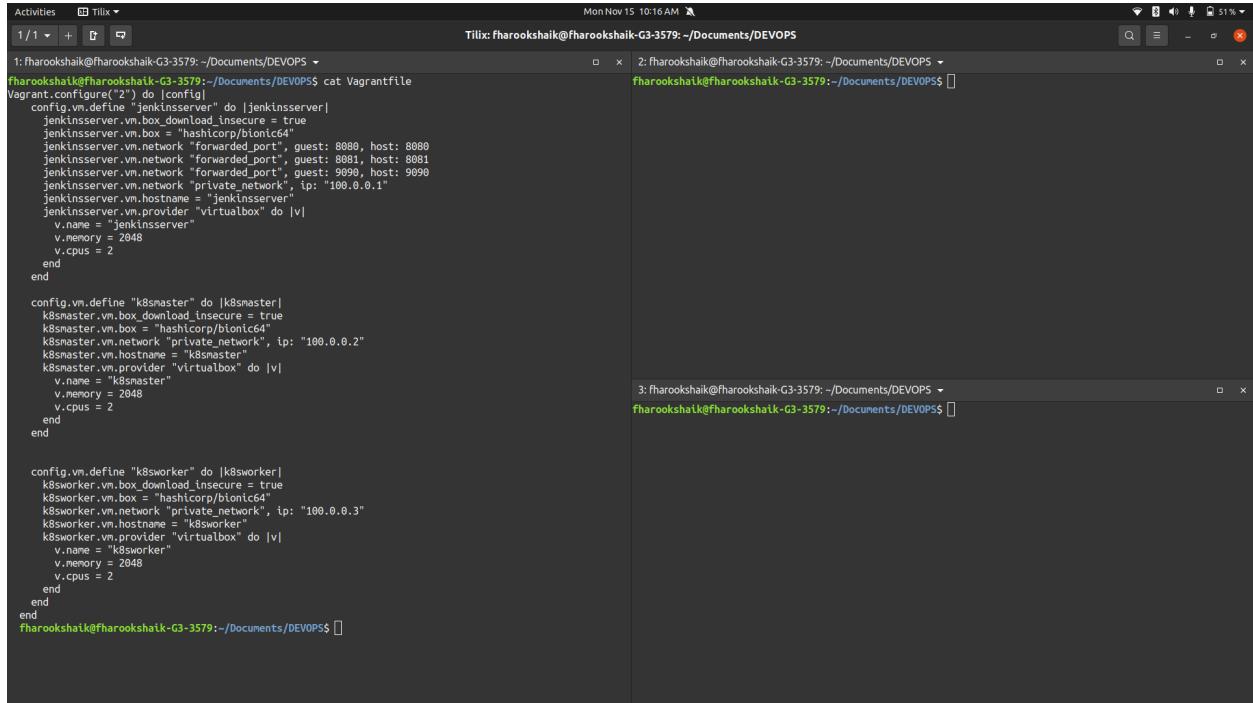
## Software Requirements

- 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.



The screenshot shows a Linux desktop environment with three terminal windows open in a tiled arrangement. All three terminals have the same title bar: "Tilix: fharookshaik@fharookshaik-G3-3579: ~/Documents/DEVOPS\$".  
Terminal 1 (Left):  
Content: "1: fharookshaik@fharookshaik-G3-3579:~/Documents/DEVOPS ~", followed by the command "cat Vagrantfile".  
Terminal 2 (Middle):  
Content: "2: fharookshaik@fharookshaik-G3-3579:~/Documents/DEVOPS ~", followed by the command "cat Vagrantfile".  
Terminal 3 (Right):  
Content: "3: fharookshaik@fharookshaik-G3-3579:~/Documents/DEVOPS ~", followed by the command "cat Vagrantfile".  
The Vagrantfile content is identical across all three terminals and includes definitions for three VMs: jenkinsserver, k8smaster, and k8sworker, each with specific configurations like box type, memory, and network settings.

```
fharookshaik@fharookshaik-G3-3579:~/Documents/DEVOPS$ cat Vagrantfile
Vagrant.configure('2') do |config|
  config.vm.define("jenkinsserver") do |jenkinsserver|
    jenkinsserver.vm.box = "hashicorp/bionic64"
    jenkinsserver.vm.network "forwarded_port", guest: 8080, host: 8080
    jenkinsserver.vm.network "forwarded_port", guest: 8081, host: 8081
    jenkinsserver.vm.network "forwarded_port", guest: 9090, host: 9090
    jenkinsserver.vm.network "private_network", ip: "100.0.0.1"
    jenkinsserver.vm.hostname = "jenkinsserver"
    jenkinsserver.vm.provider "virtualbox" do |v|
      v.name = "jenkinsserver"
      v.memory = 2048
      v.cpus = 2
    end
  end

  config.vm.define "k8smaster" do |k8smaster|
    k8smaster.vm.box_download.insecure = true
    k8smaster.vm.box = "hashicorp/bionic64"
    k8smaster.vm.network "private_network", ip: "100.0.0.2"
    k8smaster.vm.hostname = "k8smaster"
    k8smaster.vm.provider "virtualbox" do |v|
      v.name = "k8smaster"
      v.memory = 2048
      v.cpus = 2
    end
  end

  config.vm.define "k8sworker" do |k8sworker|
    k8sworker.vm.box_download.insecure = true
    k8sworker.vm.box = "hashicorp/bionic64"
    k8sworker.vm.network "private_network", ip: "100.0.0.3"
    k8sworker.vm.hostname = "k8sworker"
    k8sworker.vm.provider "virtualbox" do |v|
      v.name = "k8sworker"
      v.memory = 2048
      v.cpus = 2
    end
  end

```

- Using vagrant ssh <vm\_name> we can login to the respective VM.

```

Activities   Tilix
1/1 +  Tilix
Mon Nov 15 10:17 AM
Tilix: vagrant@jenkinsserver: ~
1:vagrant@jenkinsserver: ~
fharookshah@fharookshah-G3-3579:~/Documents/DEVOPS$ vagrant ssh jenkinsserver
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-58-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

System information as of Mon Nov 15 04:46:51 UTC 2021

System load: 0.04 Processes: 103
Usage of /: 2.5% of 61.80GB Users logged in: 0
Memory usage: 6% IP address for eth0: 10.0.2.15
Swap usage: 0% IP address for eth1: 100.0.0.1

* Super-optimized for small spaces - read how we shrank the memory
footprint of MicroK8s to make it the smallest full K8s around.

https://ubuntu.com/blog/microk8s-memory-optimisation

0 packages can be updated.
0 updates are security updates.

vagrant@jenkinsserver:~$ 

3:vagrant@k8smaster: ~
Tilix: vagrant@k8smaster: ~
2:vagrant@k8smaster: ~
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-58-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

System information as of Mon Nov 15 04:46:58 UTC 2021

System load: 0.09 Processes: 103
Usage of /: 2.5% of 61.80GB Users logged in: 0
Memory usage: 6% IP address for eth0: 10.0.2.15
Swap usage: 0% IP address for eth1: 100.0.0.2

* Super-optimized for small spaces - read how we shrank the memory
footprint of MicroK8s to make it the smallest full K8s around.

https://ubuntu.com/blog/microk8s-memory-optimisation

0 packages can be updated.
0 updates are security updates.

vagrant@k8smaster:~$ 

vagrant@k8sworker: ~
Tilix: vagrant@k8sworker: ~
4:vagrant@k8sworker: ~
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-58-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

System information as of Mon Nov 15 04:47:22 UTC 2021

System load: 0.25 Processes: 103
Usage of /: 2.5% of 61.80GB Users logged in: 0
Memory usage: 6% IP address for eth0: 10.0.2.15
Swap usage: 0% IP address for eth1: 100.0.0.3

* Super-optimized for small spaces - read how we shrank the memory
footprint of MicroK8s to make it the smallest full K8s around.

https://ubuntu.com/blog/microk8s-memory-optimisation

0 packages can be updated.
0 updates are security updates.

vagrant@k8sworker:~$ 

```

## 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
1 / | + -  ↻  47% ▾
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:~$ [ ]
```

```

Tilix: 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=0.788 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:~$ [ ]
```

```

Tilix: 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
1 / | + -  ↻  46% ▾
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's randomart image is:
+---[RSA 2048]---+
|          oE*.. |
|          o . . |
|          . o     |
|          ..+ o    |
|          S+.+ .   |
|          o o o .  |
|          o B+B+ o |
|          .o 4+*.o  |
|          .o+o+o+o  |
+---[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.
EDSA key fingerprint is SHA256:uY6CIjFdI9qTC4Qy9b80QRk+wbIJF9cdSglr3SmmL+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.
EDSA key fingerprint is SHA256:uY6CIjFdI9qTC4Qy9b80QRk+wbIJF9cdSglr3SmmL+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

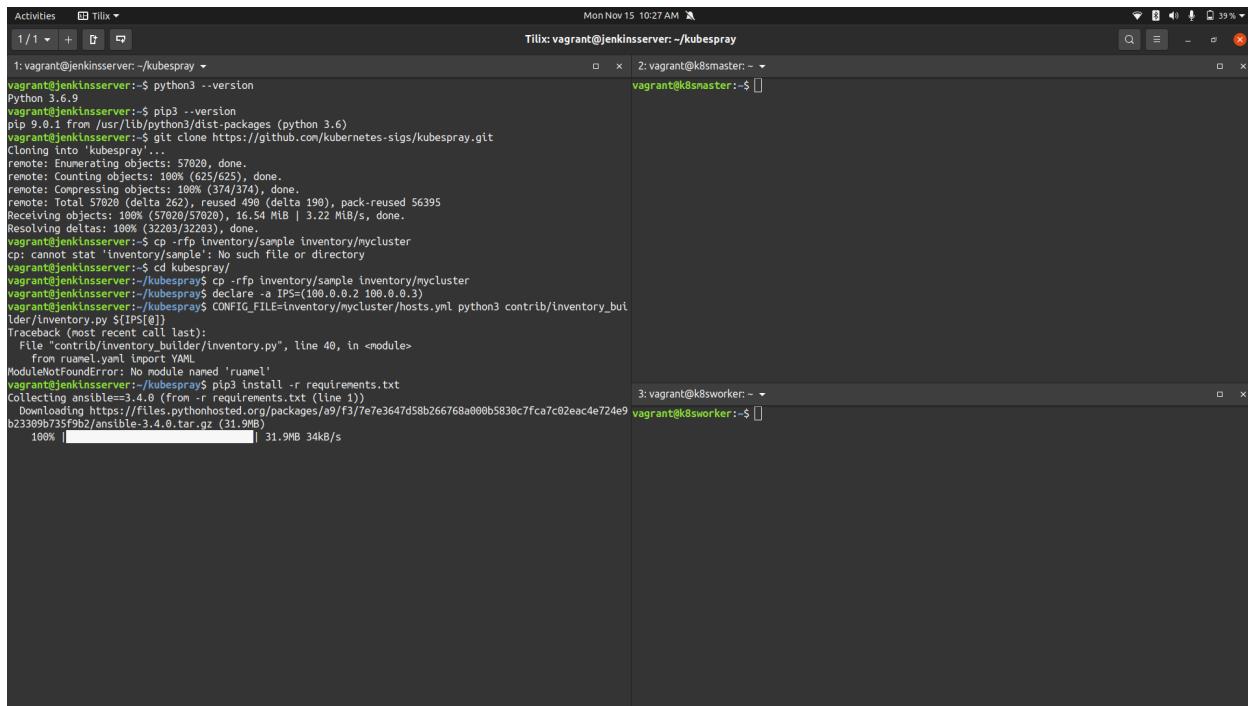
Now try logging into the machine, with: "ssh '100.0.0.3'"
and check to make sure that only the key(s) you wanted were added.
```

## 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[@]}`



The screenshot shows three terminal windows side-by-side:

- Terminal 1 (jenkinsserver):** Shows the initial steps of cloning the kubespray repository and installing requirements. It includes pip3 version check, cloning, and requirement installation.
- Terminal 2 (k8smaster):** Shows the configuration of the inventory file. It includes copying the sample inventory, changing to the kubespray directory, and running the declare command to set the IP addresses.
- Terminal 3 (k8sworker):** Shows the preparation of the host.yml file. It includes running the inventory\_builder/inventory.py script with the provided IP list.

```

Activities   Tilix   Mon Nov 15 10:32 AM
1 / 1 + 2  2: vagrant@jenkinsserver: ~/kubespray
vagrant@jenkinsserver: ~/kubespray
Collecting packaging (from ansible-base==2.10.15-> r requirements.txt (line 2))
  Downloading https://files.pythonhosted.org/packages/b1/09/464d5df9f9ec1ab5054af6d097df6793e542f4aa426ba3
  062ec64409cab7/packaging-21.2-py3-none-any.whl (40KB)
    100% |██████████| 40KB 4.9MB/s
Collecting cffi!=1.11.3,>=1.8 (from cryptography==2.8-> r requirements.txt (line 3))
  Downloading https://files.pythonhosted.org/packages/d9/5a/e7c31adeb875f2abbb91bd84cf2dc52d792b5a01596781
  d5fc25c91da11/six-1.16.0-py2.py3-none-any.whl (40KB)
Collecting pyrsistent<3,>=2.0.2 (from packaging->ansible-base==2.10.15-> r requirements.txt (line 2))
  Downloading https://files.pythonhosted.org/packages/8a/bb/488841f56197b13700a0fd568fc279a2025a39e22449b7
  cf29864669b15d/pyrsistent-2.4.7-py2.py3-none-any.whl (67KB)
    100% |██████████| 71kB 4.7MB/s
Collecting pycparser (from cffi!=1.11.3,>=1.8-> r requirements.txt (line 3))
  Downloading https://files.pythonhosted.org/packages/62/05/5f610be421e85889f2e55e3b7f9a6795bd982198517d
  912eb1c761a53/pycparser-2.21-py2.py3-none-any.whl (118B)
    100% |██████████| 122kB 3.0MB/s
Building wheels for collected packages: ansible-base
  Running setup.py bdist_wheel for ansible ... done
  Stored in directory: /home/vagrant/.cache/pip/wheels/ae/53/8d/114cb292aceddb0a57c78c9463942591aaaf9d78e5
Fbd4d2d2
  Running setup.py bdist_wheel for ansible-base ... done
  Stored in directory: /home/vagrant/.cache/pip/wheels/5e/44/b7/f3ea3c36a6a7eb5287ae3e51bd2621d4c9c843bed
44911b2d
Successfully built ansible-base
Installing collected packages: PyYAML, pycparser, cffi, six, cryptography, MarkupSafe, jinja2, pyrsistent, ansible-base, ansible, netaddr, pbr, jmespath, ruamel.yaml.lib, ruamel.yaml
Successfully installed MarkupSafe-1.1.1 PyYAML-6.0 ansible-3.4.0 ansible-base-2.10.15 cffi-1.15.0 cryptography-2.8 jinja2-2.11.3 jmespath-0.9.3 netaddr-0.7.19 packaging-21.2 pbr-5.4.4 pycparser-2.21 pyrsistent-2.4.7 ruamel.yaml.lib-0.2.4 six-1.16.0
vagrant@jenkinsserver:~/kubespray$ CONFIG_FILE=inventory/mycluster/hosts.yml python3 contrib/inventory_builder/inventory.py $([`ps`])
DEBUG: Adding group all
DEBUG: Adding group kube_control_plane
DEBUG: Adding group kube_node
DEBUG: Adding group etcd
DEBUG: Adding group k8s_cluster
DEBUG: Adding group calico_rr
DEBUG: adding host node1 to group all
DEBUG: adding host node2 to group all
DEBUG: adding host node1 to group etcd
DEBUG: adding host node1 to group kube_control_plane
DEBUG: adding host node2 to group kube_control_plane
DEBUG: adding host node1 to group kube_node
DEBUG: adding host node2 to group kube_node
vagrant@jenkinsserver:~/kubespray$ 
```

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

```

Activities   Tilix   Mon Nov 15 10:34 AM
1 / 1 + 2  2: vagrant@jenkinsserver: ~/kubespray
vagrant@jenkinsserver: ~/kubespray
vagrant@jenkinsserver: ~/kubespray$ ansible-playbook -i inventory/mycluster/hosts.yml --become --become-user vagrant@k8smaster:-
r=root cluster.yml
PLAY [localhost] *****
Monday 15 November 2021  05:04:35 +0000 (0:00:00.173)      0:00:00.173 *****
TASK [Check Ansible version < 2.12.0] *****
ok: [localhost] => [
  "changed": false,
  "msg": "All assertions passed"
]
Monday 15 November 2021  05:04:35 +0000 (0:00:00.065)      0:00:00.238 *****
TASK [Check Ansible version > 2.10.11 when using ansible 2.10] *****
ok: [localhost] => [
  "changed": false,
  "msg": "All assertions passed"
]
Monday 15 November 2021  05:04:35 +0000 (0:00:00.069)      0:00:00.308 *****
TASK [Check that python netaddr is installed] *****
ok: [localhost] => [
  "changed": false,
  "msg": "All assertions passed"
]
Monday 15 November 2021  05:04:35 +0000 (0:00:00.095)      0:00:00.404 *****
TASK [Check that jinja is not too old (install via pip)] *****
ok: [localhost] => [
  "changed": false,
  "msg": "All assertions passed"
]
[WARNING]: Could not match supplied host pattern, ignoring: kube-master
PLAY [Add kube-master nodes to kube_control_plane] *****
skipping: no hosts matched
[WARNING]: Could not match supplied host pattern, ignoring: kube-node
PLAY [Add kube-node nodes to kube_node] *****
skipping: no hosts matched
[WARNING]: Could not match supplied host pattern, ignoring: k8s-cluster
PLAY [Add k8s-cluster nodes to k8s_cluster] *****
skipping: no hosts matched
[WARNING]: Could not match supplied host pattern, ignoring: calico-rr
PLAY [Add calico-rr nodes to calico_rr] *****
skipping: no hosts matched
vagrant@jenkinsserver: ~/kubespray$ 
```

## Step 4: Install kubectl on k8smaster

Mon Nov 15 10:55 AM  
Tilix: vagrant@k8smaster:~  
1: vagrant@jenkinserver: ~/kubespray ~  
vagrant@jenkinserver:~/kubespray\$ [ ]  
2: vagrant@k8smaster:~ -  
vagrant@k8smaster:~\$ curl -LO https://storage.googleapis.com/kubernetes-release/release/'curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt' /bin/linux/amd64/kubectl  
% Total % Received % Xferd Average Speed Time Time Current  
Dload Upload Total Spent Left Speed  
14 44.7M 14 6734k 0 0 2532k 0 0:00:18 0:00:02 0:00:16 2531k[ ]  
3: vagrant@k8worker:~ -  
vagrant@k8worker:~\$ [ ]

Mon Nov 15 10:57 AM  
Tilix: vagrant@k8smaster:~  
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 (127.0.0.1) 56(84) bytes of data.  
64 bytes from k8smaster (127.0.0.1): icmp\_seq=1 ttl=64 time=0.034 ms  
64 bytes from k8smaster (127.0.0.1): icmp\_seq=2 ttl=64 time=0.060 ms  
^Z  
[2]+ Stopped ping k8smaster  
vagrant@k8smaster:~\$ ping k8worker  
PING k8worker (100.0.0.3) 56(84) bytes of data.  
64 bytes from k8worker (100.0.0.3): icmp\_seq=1 ttl=64 time=1.27 ms  
64 bytes from k8worker (100.0.0.3): icmp\_seq=2 ttl=64 time=0.998 ms  
^Z  
[3]+ Stopped ping k8worker  
vagrant@k8smaster:~\$ clear  
  
vagrant@k8smaster:~\$ curl -LO https://storage.googleapis.com/kubernetes-release/release/'curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt' /bin/linux/amd64/kubectl  
% Total % Received % Xferd Average Speed Time Time Current  
Dload Upload Total Spent Left Speed  
100 44.7M 100 44.7M 0 0 3188k 0 0:00:14 0:00:14 0:00:14 3281k  
vagrant@k8smaster:~\$ sudo cp ./bin/kubernetes/admin.conf /home/vagrant/config  
vagrant@k8smaster:~\$ mkdir .kube  
vagrant@k8smaster:~\$ mv config kube/  
vagrant@k8smaster:~\$ sudo chown \$(id -u):\$(id -g ) \$HOME/.kube/config  
vagrant@k8smaster:~\$ kubectl version  
Client Version: version.Info{Major:"1", Minor:"22", GitVersion:"v1.22.3", GitCommit:"c92036820499fedefec0f847e2054d824aea6cd1", GitTreeState:"clean", BuildDate:"2021-10-27T18:41:28Z", GoVersion:"go1.16.9", Compiler:"gc", Platform:"linux/amd64"}  
Server Version: version.Info{Major:"1", Minor:"22", GitVersion:"v1.22.3", GitCommit:"c92036820499fedefec0f847e2054d824aea6cd1", GitTreeState:"clean", BuildDate:"2021-10-27T18:35:25Z", GoVersion:"go1.16.9", Compiler:"gc", Platform:"linux/amd64"}  
vagrant@k8smaster:~\$ kubectl get nodes  
NAME STATUS ROLES AGE VERSION  
node1 Ready control-plane,master 6m10s v1.22.3  
node2 Ready control-plane,master 5m43s v1.22.3  
vagrant@k8smaster:~\$ [ ]

# Setup CI/CD Jenkins Kubernetes Pipeline

## Step 5: Install and configure Jenkins on jenkinsserver

- Update the repositories and install openjdk-11-jdk

Terminal 1 (jenkinsserver):  
vagrant@jenkinsserver:~\$ sudo apt-get update  
Get:1 http://archive.ubuntu.com/ubuntu bionic InRelease [88.7 kB]  
Get:2 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [74.6 kB]  
Fetched 163 kB in 1s (122 kB/s)  
Reading package lists... Done  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following additional packages will be installed:  
at-spi2-core ca-certificates-java fontconfig-config fonts-dejavu-core fonts-dejavu-extra java-common libasound2-data libatk-bridge0.0 libatk-wrapper-java libatk-wrapper-java-jni libatk1.0-0 libatk1.0-data libatspi2.0 libatk1.0 libatk1.0-i18n libatk1.0-intel libatk1.0-radeon1 libatk1.0-tx libatkfont1 libfontenc1 libglf7 libgl1-mesa-dri libglapi-mesa libglvns1 libglx1 libglxv1 libglxv1-mesa0 libglv0 libgrahpite2.3 libharfbuzz0b libice-dev libice6 libjpegb6 libjpegb8 liblcms2-2 liblomm10 liblspn4 liblx11-data liblx11-dev liblx11-doc liblx11-xcb liblxau-dev liblxaw liblxaw7 libxcb-cdr12-0 libxcb-cdr3-0 libxcb-glx0 libxcb-present libxcb-shape0 libxcb-sync1 libxf12 libxix6 libxinerama1 libxmu6 libxmuu6 libxrandr2 libxrender1 libxshmfence1 libxt-dev libxt6 libxtst libxv1 libxf86dg1 libxxf86uni openjdk-11-jre openjdk-11-jre-headless x11-common x11-utils x11proto-core-dev x11proto-dockutil xtrans-dev  
Suggested packages:  
default-jre libasound2-plugins alsu-utils libice-doc liblxns2-utils pccm lmsensors libm4n-doc libxcb-doc libxt-doc openjdk-11-doc visualvm libnss-mdns fonts-ipafont-gothic fonts-ipafont-mincho fonts-way-microhei fonts-way-zhenhei fonts-indic mesa-utils  
The following packages will be upgraded:  
libasound2 libasound2-data libatk-bride2.0 libatk-wrapper-java libatk-wrapper-java-jni libatk1.0-0 libatk1.0-data libatspi2.0 libatk1.0 libatk1.0-i18n libatk1.0-intel libatk1.0-radeon1 libfontconfig1 libglf7 libgl1 libgl1-mesa-dri libglapi-mesa libglvnd8 libglxv1 libglxv1-mesa0 libglv0 libgrahpite2.3 libharfbuzz0b libice-dev libice6 libjpegb6 libjpegb8 liblcms2-2 liblomm10 liblspn4 liblx11-data liblx11-dev liblx11-doc liblx11-xcb liblxau-dev liblxaw liblxaw7 libxcb-cdr12-0 libxcb-cdr3-0 libxcb-glx0 libxcb-present libxcb-shape0 libxcb-sync1 libxf12 libxix6 libxinerama1 libxmu6 libxmuu6 libxrandr2 libxrender1 libxshmfence1 libxt-dev libxt6 libxtst libxv1 libxf86dg1 libxxf86uni openjdk-11-jdk openjdk-11-jdk-headless openjdk-11-jre openjdk-11-jre-headless x11-common x11-utils x11proto-core-dev  
x11proto-core-dev xorg-sgml-doctools xtrans-dev  
The following packages will be upgraded:  
libdmz-common libdmz2  
3 upgraded, 0 newly installed, 0 to remove and 226 not upgraded.  
Need to get 796 MB of archives.  
After this operation, 774 MB of additional disk space will be used.  
Do you want to continue? [Y/n]

Terminal 2 (k8smaster):  
vagrant@k8smaster:~\$

Terminal 3 (k8sworker):  
vagrant@k8sworker:~\$

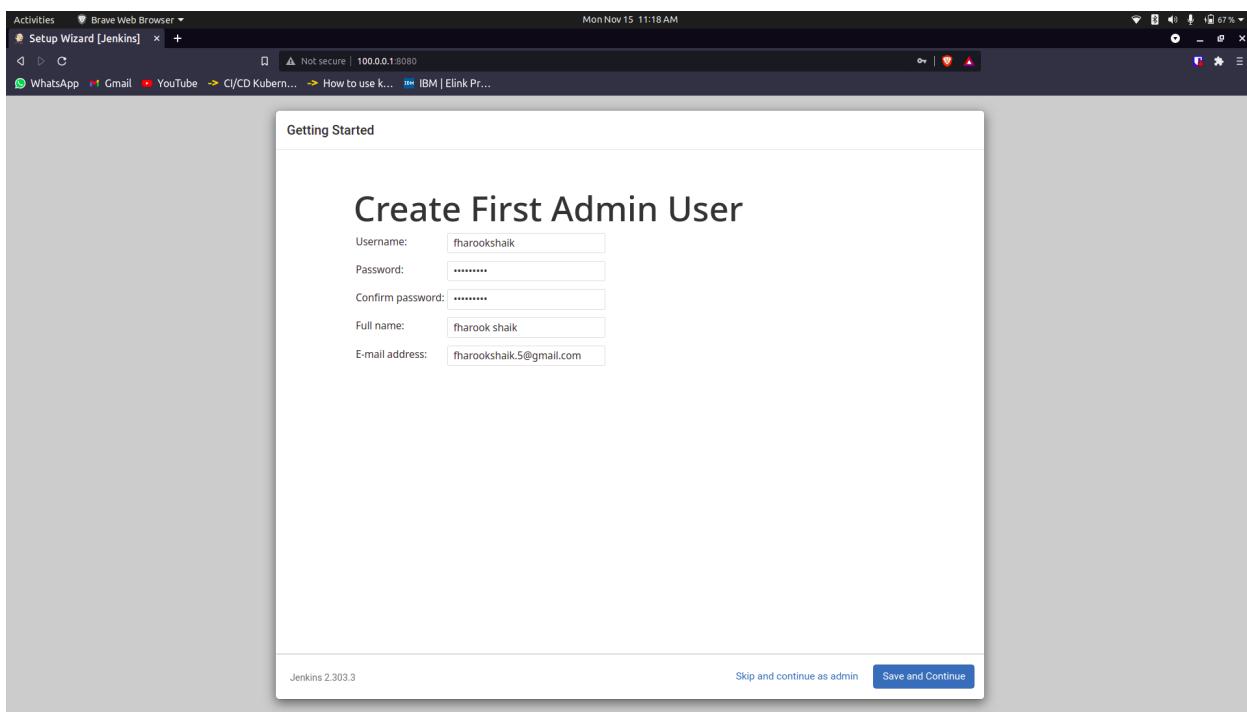
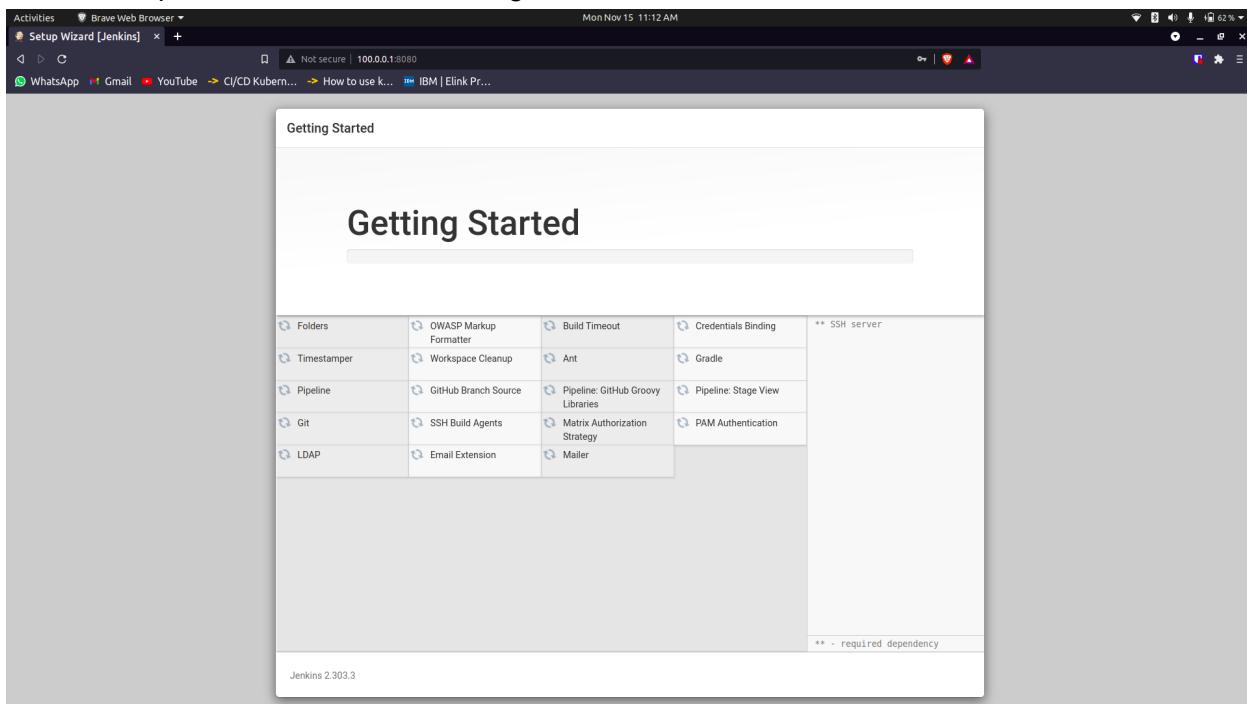
- Install Jenkins

Terminal 1 (jenkinsserver):  
vagrant@jenkinsserver:~\$ curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo tee /etc/apt/trusted.gpg.d/jenkins.io.key  
> /usr/share/keyrings/jenkins-keyring.asc > /dev/null  
vagrant@jenkinsserver:~\$ echo deb [signed-by='https://pkg.jenkins.io/debian-stable/binary/'] https://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list  
> /etc/apt/sources.list.d/jenkins.list > /dev/null  
vagrant@jenkinsserver:~\$ sudo apt-get update  
Ign:1 https://pkg.jenkins.io/debian-stable binary/ InRelease  
Get:2 https://pkg.jenkins.io/debian-stable binary/ Release [2,044 B]  
Get:3 https://pkg.jenkins.io/debian-stable binary/ Release.gpg [833 B]  
Hit:4 http://archive.ubuntu.com/ubuntu bionic InRelease  
Hit:5 http://security.ubuntu.com/ubuntu bionic-security InRelease  
Hit:6 http://archive.ubuntu.com/ubuntu bionic-updates InRelease  
Get:7 http://archive.ubuntu.com/ubuntu bionic-backports InRelease  
Hit:8 http://archive.ubuntu.com/ubuntu bionic-updates InRelease  
Fetched 23.8 kB in 1s (21.7 kB/s)  
Reading package lists... Done  
vagrant@jenkinsserver:~\$ sudo apt-get install jenkins  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following additional packages will be installed:  
daemon  
The following NEW packages will be installed:  
daemon jenkins  
0 upgraded, 2 newly installed, 0 to remove and 225 not upgraded.  
Need to get 72.8 kB of archives.  
After this operation, 72.7 kB of additional disk space will be used.  
Do you want to continue? [Y/n] y  
Get:2 http://archive.ubuntu.com/ubuntu bionic/universe amd64 daemon amd64 0.6.4-1build1 [99.5 kB]  
0% [2 daemon 12.8 kB/99.5 kB 13%]

Terminal 2 (k8smaster):  
vagrant@k8smaster:~\$

Terminal 3 (k8sworker):  
vagrant@k8sworker:~\$

- Setup Jenkins with default configuration and create a user



- Install SSH Pipeline Steps Plugin in Jenkins

The screenshot shows the Jenkins Update Center page. At the top, it says "Activities" and "Brave Web Browser". The title bar says "Update Center [Jenkins]". The URL is "Not secure | 100.0.0.1:8080/updateCenter/". Below the title, there's a breadcrumb trail: WhatsApp > YouTube > CI/CD Kubern... > How to use k... > IBM | Elink Pr... The main content area is titled "Dashboard > Update Center". It lists various Jenkins components and their status: LDAP (Success), Email Extension (Success), Mailer (Success), Loading plugin extensions (Success), Pipeline: SCM Step (Success), Pipeline: Groovy (Success), Gradle (Success), Pipeline Graph Analysis (Success), Pipeline: REST API (Success), Pipeline: Stage View (Success), Pipeline: Declarative Extension Points API (Success), Pipeline: Shared Groovy Libraries (Success), Pipeline: Multibranch (Success), Pipeline: Declarative (Success), Pipeline (Success), Git (Success), GitHub (Success), GitHub Branch Source (Success), Pipeline: GitHub Groovy Libraries (Success), Loading plugin extensions (Success), SSH Pipeline Steps (Pending), and Loading plugin extensions (Pending). At the bottom, there are links to "Go back to the top page" and "Restart Jenkins when installation is complete and no jobs are running". The footer shows "REST API" and "Jenkins 2.303.3".

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

The screenshot shows the Jenkins Global Tool Configuration page. At the top, it says "Activities" and "Brave Web Browser". The title bar says "Global Tool Configuration [Jenkins]". The URL is "Not secure | 100.0.0.1:8080/configureTools/". Below the title, there's a breadcrumb trail: WhatsApp > YouTube > CI/CD Kubern... > How to use k... > IBM | Elink Pr... The main content area has a green header bar with the text "Saved". It shows a configuration for "Gradle":

- "name": "default"
- "Install automatically" checkbox is checked
- "Install from Gradle.org" section:
  - "Version": "Gradle 7.3"
  - "Add Installer" button
  - "Delete Installer" button

Below this, there are sections for "Ant" and "Maven". The "Ant" section has "Ant installations" and "Add Ant" buttons. The "Maven" section has "Maven installations" and "Save" and "Apply" buttons. The footer shows "Delete Gradle".

## Step 6: Install and configure Docker in jenkinsserver

- Install docker io

The screenshot shows three terminal windows side-by-side. The leftmost window (Tilix: vagrant@jenkinsserver) displays the command `sudo apt install docker.io` being run, along with its output showing package dependencies and upgrade information. The middle window (Tilix: vagrant@k8smaster) and the rightmost window (Tilix: vagrant@k8worker) both show a blank command line prompt.

```
vagrant@jenkinsserver:~$ sudo apt install docker.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  libltdl-utils libcontainerd-pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroups-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  libltdl-utils libcontainerd-pigz runc ubuntu-fan
0 upgraded, 6 newly installed, 0 to remove and 225 not upgraded.
Need to get 74.2 MB of additional disk space will be used.
Do you want to continue? (Y/n) y
[Connecting to archive.ubuntu.com (91.189.88.142)]
```

- Add docker & jenkins to usermod group

The screenshot shows three terminal windows. The leftmost window (Tilix: vagrant@jenkinsserver) shows the command `sudo usermod -aG docker jenkins` being run, followed by a logout message. The middle window (Tilix: vagrant@k8smaster) and the rightmost window (Tilix: vagrant@k8worker) both show a blank command line prompt.

```
vagrant@jenkinsserver:~$ sudo usermod -aG docker jenkins
vagrant@jenkinsserver:~$ exit
Connection to 127.0.0.1 closed.
fharookshah@fharookshah-G3-3579:~/Documents/DEVOPS$ vagrant ssh jenkinsserver
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-58-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

System information as of Mon Nov 15 05:56:01 UTC 2021

System load: 0.28      Users logged in:   0
Usage of /: 5.9% of 61.80GB  IP address for eth0:  10.0.2.15
Memory usage: 50%
Swap usage:  0%          IP address for eth1:  100.0.0.1
Processes:   117          IP address for docker0: 172.17.0.1

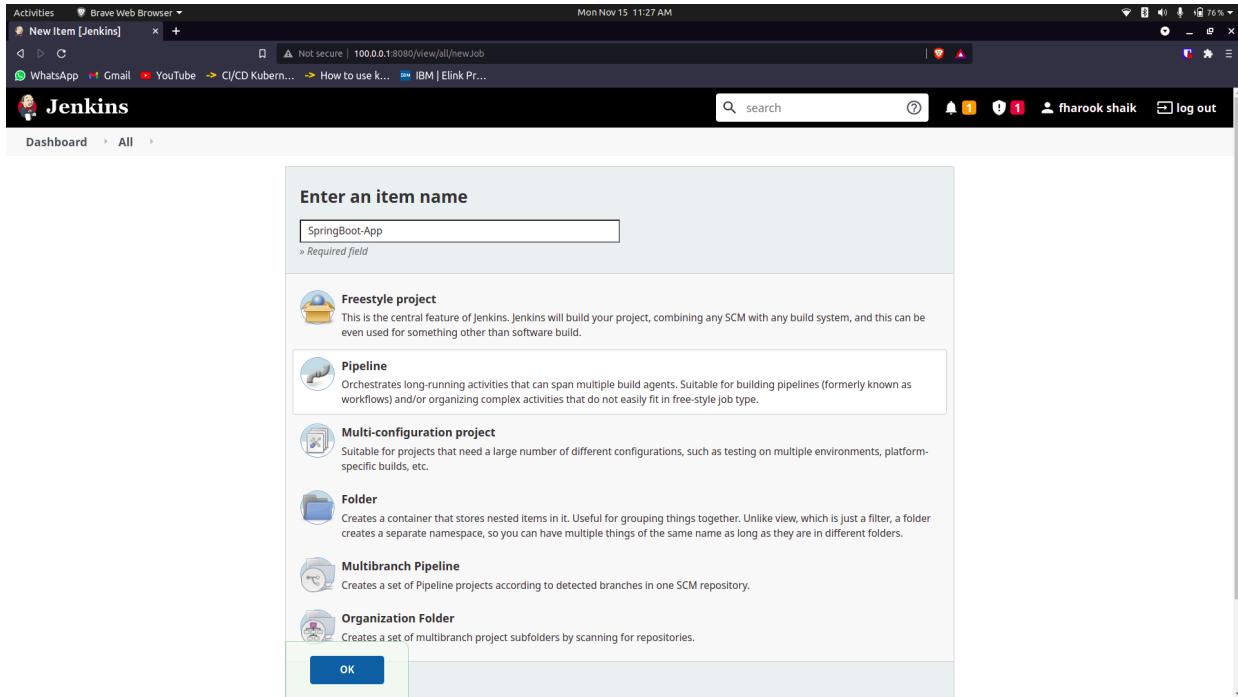
* Super-optimized for small spaces - read how we shrank the memory
  footprint of MicroK8s to make it the smallest full K8s around.
https://ubuntu.com/blog/microk8s-memory-optimisation

233 packages can be updated.
177 updates are security updates.

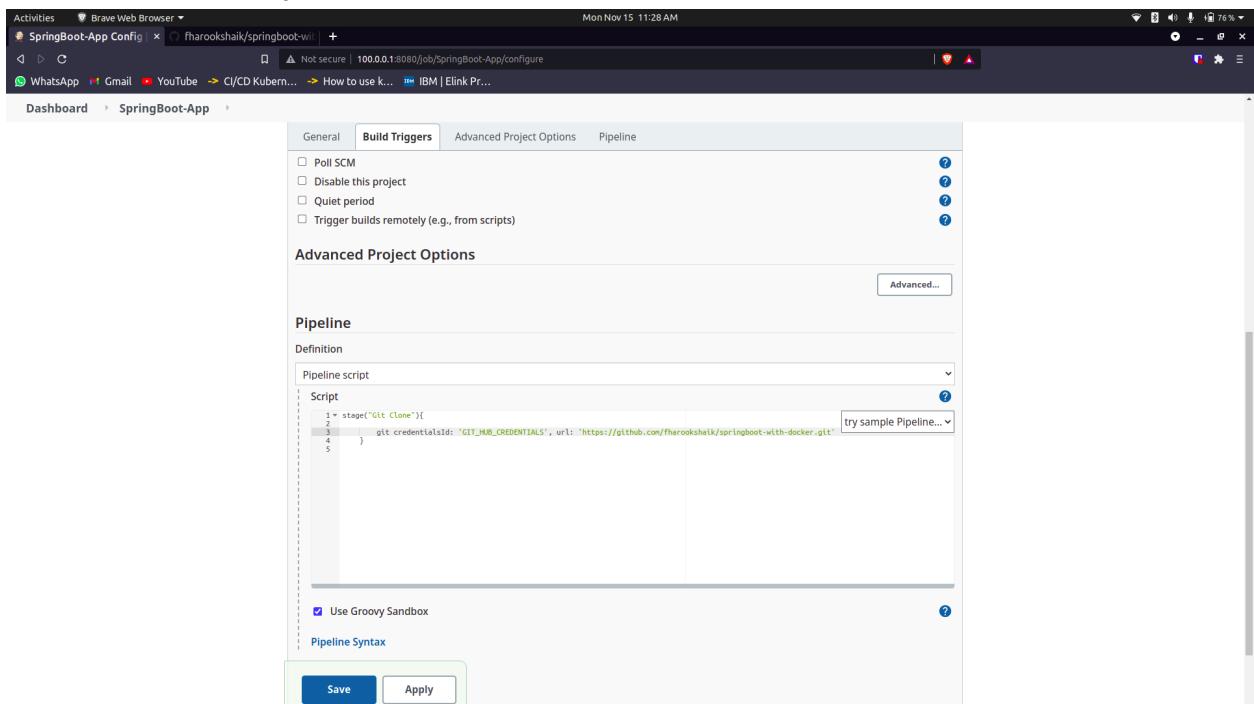
Last login: Mon Nov 15 05:51:43 2021 from 10.0.2.2
vagrant@jenkinsserver:~$
```

## 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 screenshot shows a Jenkins Pipeline interface for a project named "SpringBoot-App". The main view displays the "Stage View" for the "Pipeline SpringBoot-App". The "Git Clone" stage is listed with an average stage time of 3s. Two builds are shown: Build #2 was successful (Nov 15 11:35, No Changes) and Build #1 failed (Nov 15 11:33, No Changes). The pipeline history sidebar shows two builds: #2 (Nov 15 2021, 06:05) and #1 (Nov 15 2021, 06:03).

- 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'

}
```

**Pipeline SpringBoot-App**

**Stage View**

Git Clone	Gradle Build
2s	2min 29s
589ms	2min 29s
6s	
354ms	failed

**Build History**

- #3 15 Nov 2021, 06:08
- #2 15 Nov 2021, 06:05
- #1 15 Nov 2021, 06:03

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

```

[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Gradle Build)
[Pipeline] sh
+ ./gradlew build
> Task :gradleBuild
> Task :download https://services.gradle.org/distributions/gradle-6.4.1-bin.zip
.....10%.....20%.....30%.....40%.....50%.....60%.....70%.....80%.....90%.....100%
Welcome to Gradle 6.4.1!
Here are the highlights of this release:
- Support for building, testing and running Java Modules
- Precompiled script plugins for Groovy DSL
- Single dependency lock file per project
For more details see: https://docs.gradle.org/6.4.1/release-notes.html

Starting a Gradle Daemon (subsequent builds will be faster)
> Task :compileJava
> Task :processResources
> Task :mainClasses
> Task :bootJar
> Task :jar SKIPPED
> Task :assemble
> Task :compileTestJava
> Task :processTestResources NO-SOURCE
> Task :testClasses
> Task :test
2021-11-15 06:11:01.057 INFO 18494 --- [extShutdownHook] o.s.s.concurrent.ThreadPoolTaskExecutor : Shutting down ExecutorService 'applicationTaskExecutor'

> Task :check
> Task :build

BUILD SUCCESSFUL in 2m 27s
5 actionable tasks: 5 executed
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

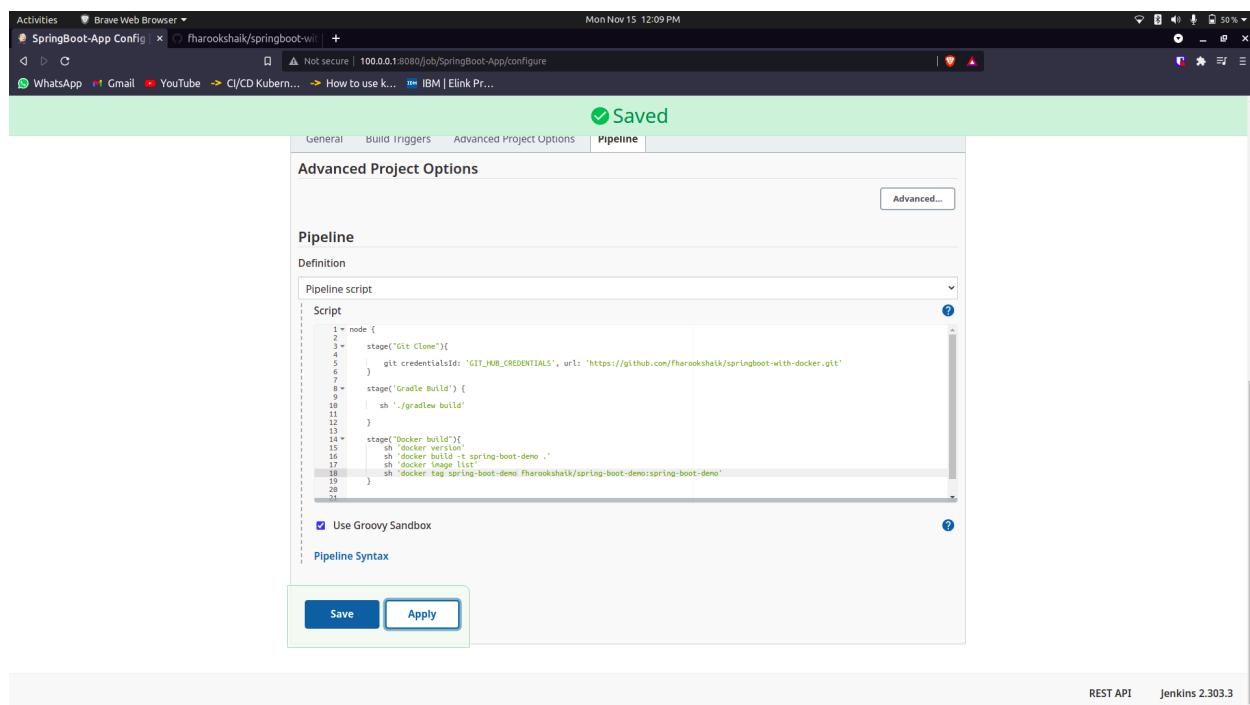
```

- 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 screenshot shows the Jenkins Pipeline configuration interface. The 'Advanced Project Options' tab is selected. The pipeline script is defined as follows:

```

stage("Build"){
    sh './gradlew build'
}
stage("Docker build"){
    sh 'docker build -t spring-boot-demo .'
    sh 'docker image ls'
    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'
}

```

Use Groovy Sandbox

**Save** **Apply**

- 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'
}

```

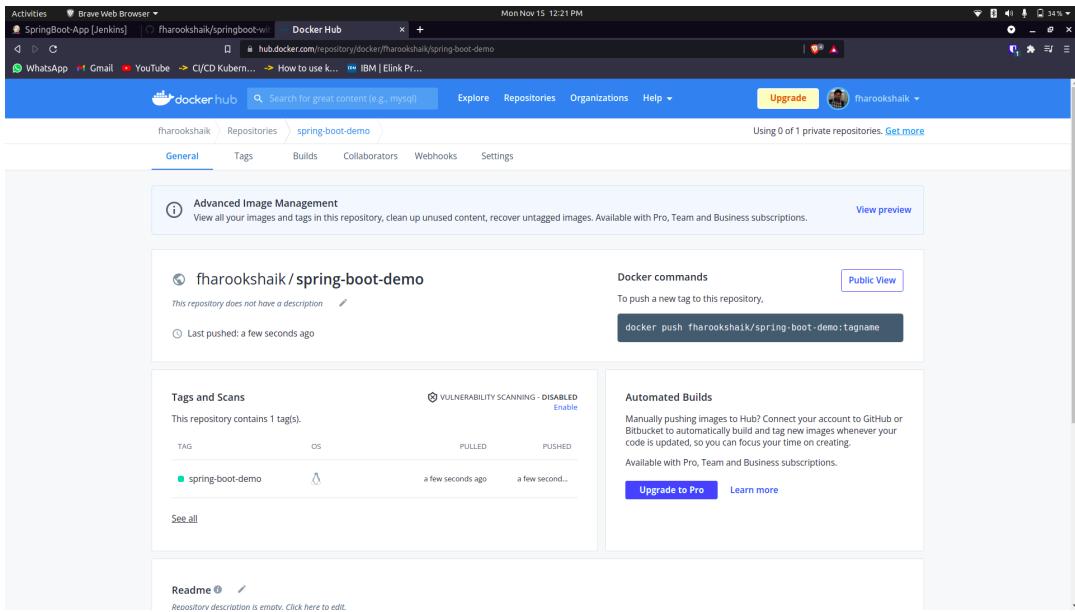
The screenshot shows the Jenkins Pipeline configuration interface. The 'Pipeline' tab is selected. The pipeline script is identical to the one shown in the previous screenshot:

```

stage("Build"){
    sh './gradlew build'
}
stage("Docker build"){
    sh 'docker build -t spring-boot-demo .'
    sh 'docker image ls'
    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'
}

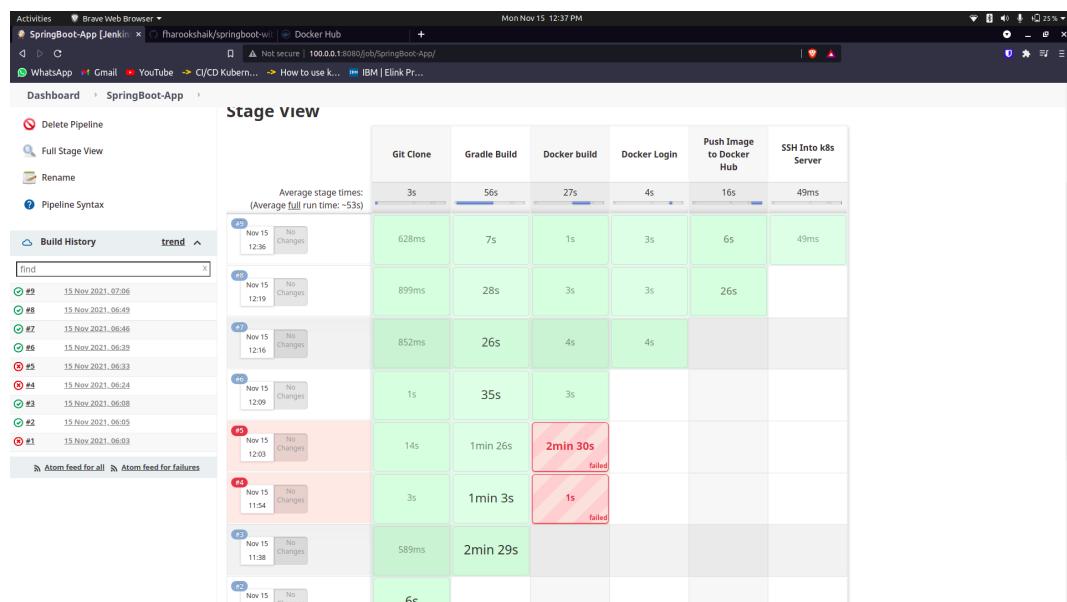
```

**Save** **Apply**



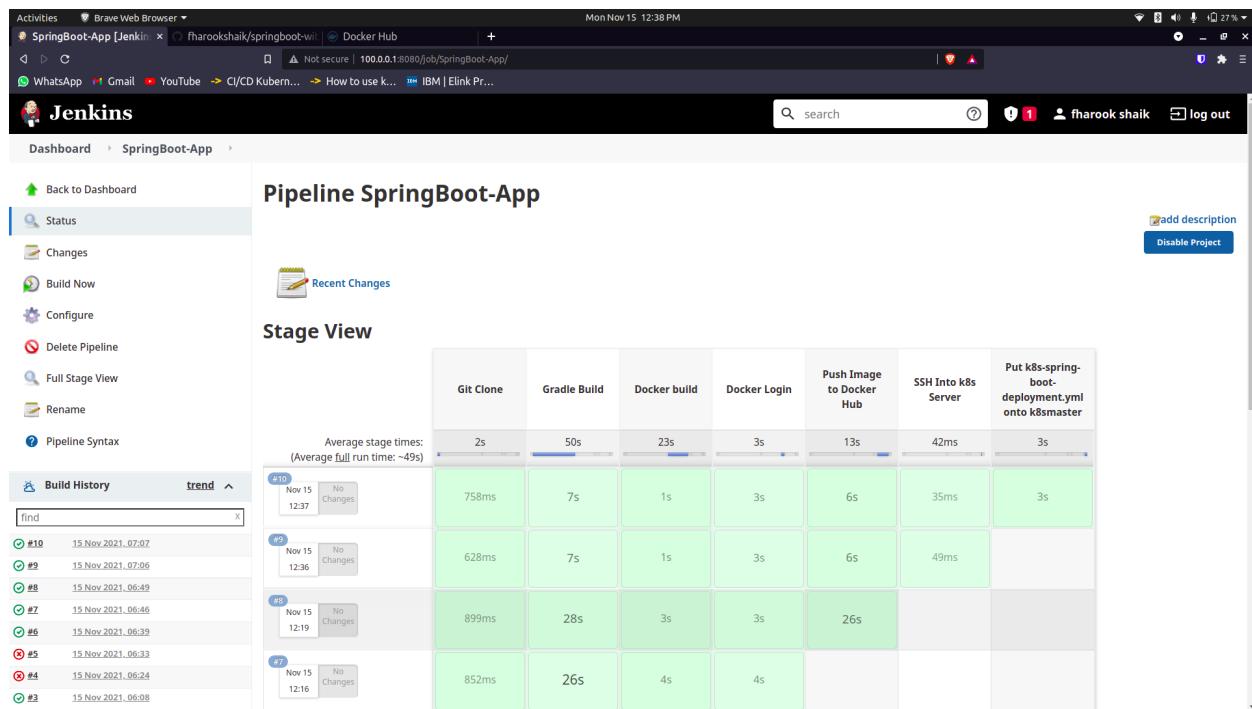
- The sixth step in the pipeline is to login into k8smaster from jenkinsserver via SSH.

```
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 ▾

SpringBoot-App [Jenkins] x fharookshaik/springboot-wii Docker Hub 100.0.0.2:30457/hello +

Not secure | 100.0.0.1:8080/job/SpringBoot-App/

WhatsApp Gmail YouTube CI/CD Kuber... How to use k... IBM | Elink Pr...

Jenkins

Dashboard > SpringBoot-App >

[Back to Dashboard](#)

Status Changes Build Now [Recent Changes](#) [Add description](#) [Disable Project](#)

**Pipeline SpringBoot-App**

**Stage View**

	Git Clone	Gradle Build	Docker build	Docker Login	PushImage 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	

**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

Activities ▾ Tiliix ▾

Mon Nov 15 1:00 PM

Tiliix: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-556997944-59cps   1/1     Running   0          30s
spring-boot-demo-556997944-952hn   1/1     Running   0          30s
spring-boot-demo-556997944-dfb7s   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

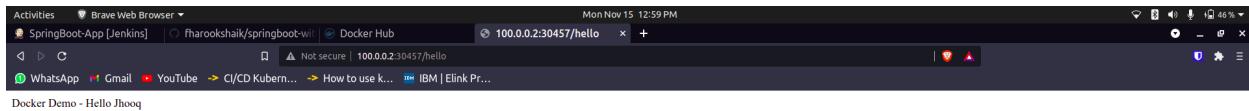
        stage('Put k8s-spring-boot-deployment.yml onto k8smaster') {
            sshPut remote: remote, from: 'k8s-spring-boot-deployment.yml',
into: '.'
        }
    }
}
```

```
        }

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