INDIAN INSTITUTE OF INFORMATION TECHNOLOGY

Dharwad

DevOps & its Applications (CS457)

Jenkins Assignment 1

Under the Guidance of - **Dr. Uma S**

Submitted by -

18BCS006-Anjuru Lokesh
18BCS021-Avinash ch
18BCS025-Diddi Geethakrishna
18BCS028-Godina Pranav
18BCS035-Kancharla Gireesh Kumar
18BCS070-Pravalika

Setting up CI/CD Jenkins pipeline for kubernetes

Tech Stack:

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

System Requirements :

- System with above 2 gb ram and 15 gb storage
- So here we have used ec2 instances with 4gb (t2.medium) memory and 15gb ssd storage

Setting Up kubernetes Cluster :

The kubeadm utility was used to set up the kubernetes cluster. Creating a kubernetes cluster with two master and worker nodes

Commands to run on both nodes :

```
To update the system packages
   $ sudo apt-get update
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
```

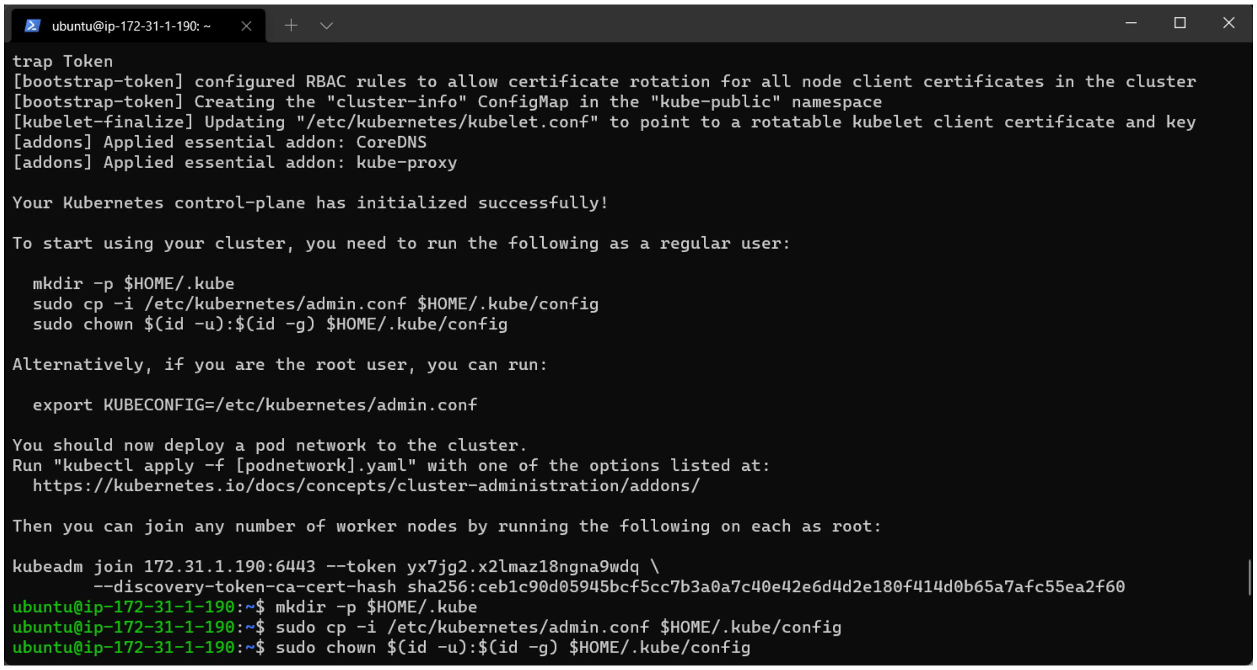
```
To add docker daemon :
$ cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
EOF
Enabling and starting docker :
$ sudo systemctl enable docker
$ sudo systemctl daemon-reload
$ sudo systemctl restart docker
```

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.g
pg] 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
```

Commands to run only on master node :

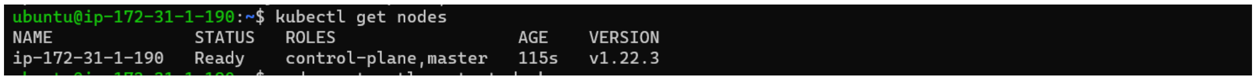
```
Initializing the master node using kubeadm
 $ sudo kubeadm init --pod-network-cidr 10.0.0.0/16
```



```
trap 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 172.31.1.190:6443 --token yx7jg2.x2lmaz18ngna9wdq \
        --discovery-token-ca-cert-hash sha256:ceb1c90d05945bcf5cc7b3a0a7c40e42e6d4d2e180f414d0b65a7afc55ea2f60
ubuntu@ip-172-31-1-190:~$ mkdir -p $HOME/.kube
ubuntu@ip-172-31-1-190:~$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
ubuntu@ip-172-31-1-190:~$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

```
$ 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



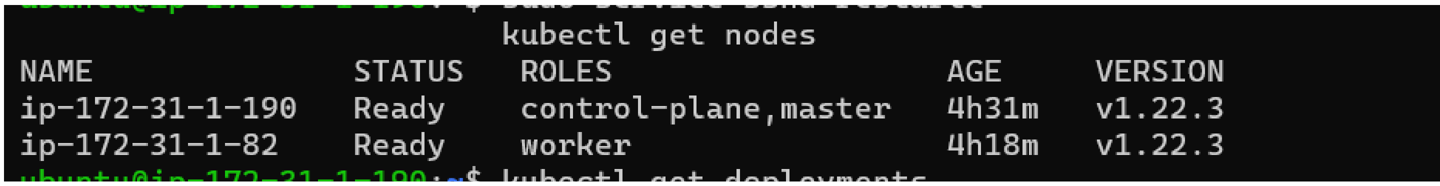
```
ubuntu@ip-172-31-1-190:~$ kubectl get nodes
NAME              STATUS   ROLES                  AGE    VERSION
ip-172-31-1-190   Ready    control-plane,master   115s   v1.22.3
```

Commands to run only on worker node

```
$ sudo kubeadm join 172.31.27.210:6443 --token ws6v8y.fd9o89anv7clc8nw
--discovery-token-ca-cert-hash
sha256:f6c8fdc0710b9296dd04ede995c7a13ea6a76c1bfcaac19ff5a10e6e625890fe
```

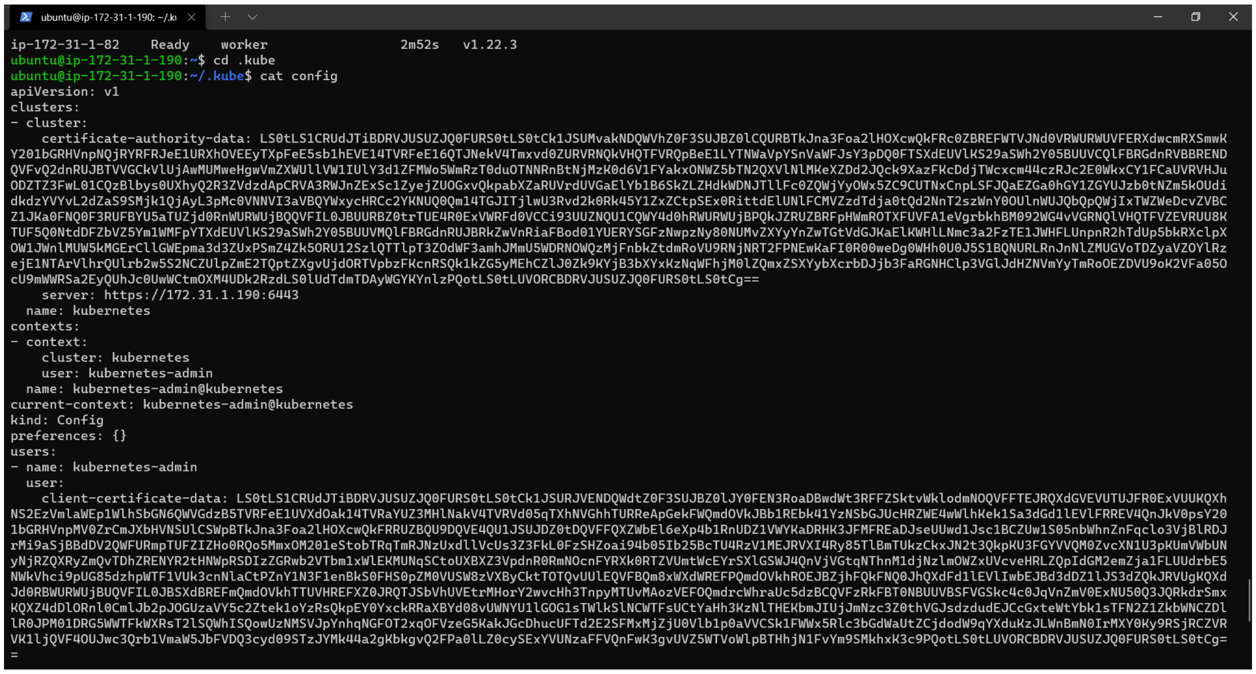
Commands to run only on master node

```
$ 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 to label it as
worker
$ kubectl label node ip-@(ip address nuder name section of node)
node-role.kubernetes.io/worker=worker
```



```
                          kubectl get nodes
NAME                 STATUS    ROLES                    AGE      VERSION
ip-172-31-1-190      Ready     control-plane,master     4h31m    v1.22.3
ip-172-31-1-82       Ready     worker                   4h18m    v1.22.3
```

```
to check configurations of kubernetes cluster
$ cd .kube
$ cat config
```



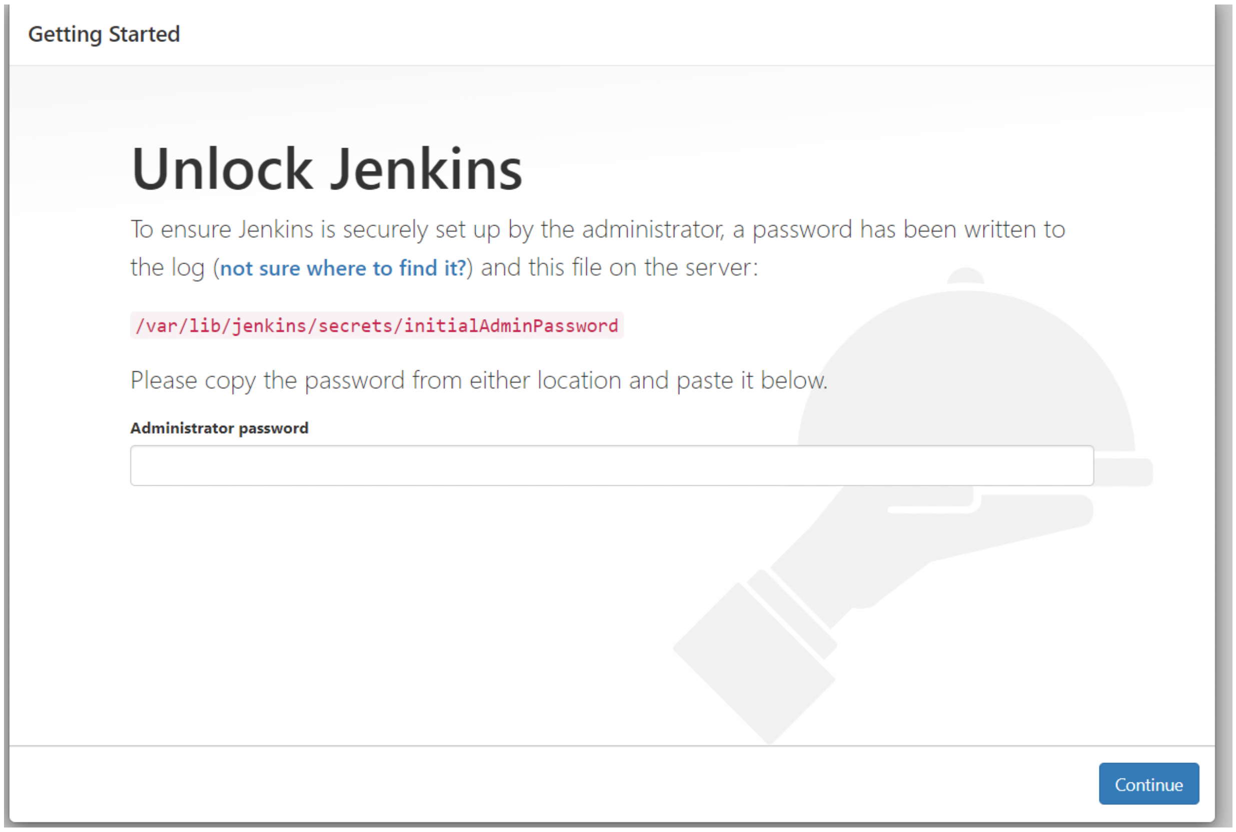
```
ip-172-31-1-82    Ready    worker                 2m52s   v1.22.3
ubuntu@ip-172-31-1-190:~$ cd .kube
ubuntu@ip-172-31-1-190:~/.kube$ cat config
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUMvakNDQWVhZ0F3SUJBZ0lCQURBTkJna3Foa2lHOXcwQkFRc0ZBREFWTVJNd0VRWURWUVFERXdwcmRXSmwK
Y201bGRHVnpNQjRYRFRJeE1URXhOVEEyTXpFeE5sb1hEVE14TVRFeE16QTJNekV4Tmxvd0ZURVRNQkVHQTFVRQpBeE1LYTNWaVpYSnVaWFJsY3pDQ0FTSXdEUVlKS29aSWh2Y05BUUVCQlFBRGdnRVBBREND
QVFvQ2dnRUJBTVVGCkVlUjAwMUMweHgwVmZXWUllVW1IUlY3d1ZFMWo5WmRzT0duOTNNRnBtNjMzK0d6V1FYakxONWZ5bTN2QXVlNlMKeXZDd2JQck9XazFKcDdjTWcxcm44czRJc2E0WkxCY1FCaUVRVHJu
ODZTZ3FwL01CQzBlbys0UXhyQ2R3ZVdzdApCRVA3RWJnZExSc1ZyejZUOGxvQkpabXZaRUVrdUVGaElYb1B6SkZLZHdkWDNJTllFc0ZQWjYyOWx5ZC9CUTNxCnpLSFJQaEZGa0hGY1ZGYUJzb0tNZm5kOUdi
dkdzYVYvL2dZaS9SMjk1QjAyL3pMc0VNNVI3aVBQYWxycHRCc2YKNUQ0Qm14TGJITjlwU3Rvd2k0Rk45Y1ZxZCtpSEx0RittdElUNlFCMVZzdTdja0tQd2NnT2szWnY0OUlnWUJQbQpQWjIxTWZWeDcvZVBC
Z1JKa0FNQ0F3RUFBYU5aTUZjd0RnWURWUjBQQVFIL0JBUURBZ0trTUE4R0ExVWRFd0VCCi93UUZNQU1CQWY4d0hRWURWUjBPQkJZRUZBRFpHWmROTXFUVFA1eVgrbkhBM092WG4vVGRNQlVHQTFVZEVRUU8K
TUF5Q0NtdDFZbVZ5Ym1WMFpYTXdEUVlKS29aSWh2Y05BUUVMQlFBRGdnRUJBRkZwVnRiaFBod01YUERYSGFzNwpzNy80NUMvZXYyYnZwTGtVdGJKaElKWHlLNmc3a2FzTE1JWHFLUnpnR2hTdUp5bkRXclpX
OW1JWnlMUW5kMGErCllGWEpma3d3ZUxPSmZ4Zk5ORU12SzlQTTlpT3ZOdWF3amhJMmU5WDRNOWQzMjFnbkZtdmRoVU9RNjNRT2FPNEwKaFI0R00weDg0WHh0U0J5S1BQNURLRnJnNlZMUGVoTDZyaVZOYlRz
ejE1NTArVlhrQUlrb2w5S2NCZUlpZmE2TQptZXgvUjdORTVpbzFKcnRSQk1kZG5yMEhCZlJ0Zk9KYjB3bXYxKzNqWFhjM0lZQmxZSXYybXcrbDJjb3FaRGNHClp3VGlJdHZNVmYyTmRoOEZDVU9oK2VFa05O
cU9mWWRSa2EyQUhJc0UwWCtmOXM4UDk2RzdLS0lUdTdmTDAyWGYKYnlzPQotLS0tLUVORCBDRVJUSUZJQ0FURS0tLS0tCg==
    server: https://172.31.1.190:6443
  name: kubernetes
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: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURJVENDQWdtZ0F3SUJBZ0lJY0FEN3RoaDBwdWt3RFFZSktvWklodmNOQVFFTEJRQXdGVEVUTUJFR0ExVUUKQXh
NS2EzVmlaWEp1WlhSbGN6QWVGdzB5TVRFeE1UVXdOak14TVRaYUZ3MHlNakV4TVRVd05qTXhNVGhhTURReApGekFWQmdOVkJBb1REbk41YzNSbGJUcHRZWE4wWlhKek1Sa3dGd1lEVlFRREV4QnJkV0psY20
1bGRHVnpMV0ZrCmJXbHVNSUlCSWpBTkJna3Foa2lHOXcwQkFRRUZBQU9DQVE4QU1JSUJDZ0tDQVFFQXZXbEl6eXp4b1RnUDZ1VWYKaDRHK3JFMFREaDJseUUwd1Jsc1BCZUw1S05nbWhnZnFqclo3VjBlRDJ
rMi9aSjBBdDV2QWFURmpTUFZIZHo0RQo5MmxOM201eStobTRqTmRJNzUxdllVcUs3Z3FkL0FzSHZoai94b05Ib25BcTU4RzV1MEJRVXI4Ry85TlBmTUkzCkxJN2t3QkpKU3FGYVVQM0ZvcXN1U3pKUmVWbUN
yNjRZQXRyZmQvTDhZRENYR2tHNWpRSDIzZGRwb2VTbm1xWlEKMUNqSCtoUXBXZ3VpdnR0RmNOcnFYRXk0RTZVUmtWcEYrSXlGSWJ4QnVjVGtqNThnM1djNzlmOWZxUVcveHRLZQpIdGM2emZja1FLUUdrbE5
NWkVhci9pUG85dzhpWTF1VUk3cnNlaCtPZnY1N3F1enBkS0FHS0pZM0VUSW8zVXByCktTOTQvUUlEQVFBQm8xWXdWREFPQmdOVkhROEJBZjhFQkFNQ0JhQXdFd1lEVlIwbEJBd3dDZ1lJS3dZQkJRVUgKQXd
Jd0RBWURWUjBUQVFIL0JBSXdBREFmQmdOVkhTTUVHREFXZ0JRQTJSbVhUVEtrMHorY2wvcHh3TnpyMTUvMAozVEFOQmdrcWhraUc5dzBCQVFzRkFBT0NBUUVBSFVGSkc4c0JqVnZmV0ExNU50Q3JQRkdrU214
KQXZ4dDlORnl0CmlJb2pJOGUzaVY5c2Ztek1oYzRsQkpEY0YxckRRaXBYd08vUWNYU1lGOG1sTWlkSlNCWTFsUCtYaHh3KzNlTHEKbmJIUjJmNzc3Z0thVGJsdzdudEJCcGxteWtYbk1sTFN2Z1ZkbWNCZDl
lR0JPM01DRG5WWTFkWXRsT2lSQWhISQowUzNMSVJpYnhqNGFOT2xqOFVzeG5KakJGcDhucUFTd2E2SFMxMjZjU0Vlb1p0aVVCSk1FWWx5Rlc3bGdWaUtZCjdodW9qYXduKzJLWnBmN0IrMXY0Ky9RSjRCZVR
VK1ljQVF4OUJwc3Qrb1VmaW5JbFVDQ3cyd09STzJYMk44a2gKbkgvQ2FPa0lLZ0cySExYVUNzaFFVQnFwK3gvUVZ5WTVoWlpBTHhjN1FvYm9SMkhxK3c9PQotLS0tLUVORCBDRVJUSUZJQ0FURS0tLS0tCg=
=
```

Setting up jenkins

On aws ec2 ubuntu

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
```



Getting Started

Unlock Jenkins

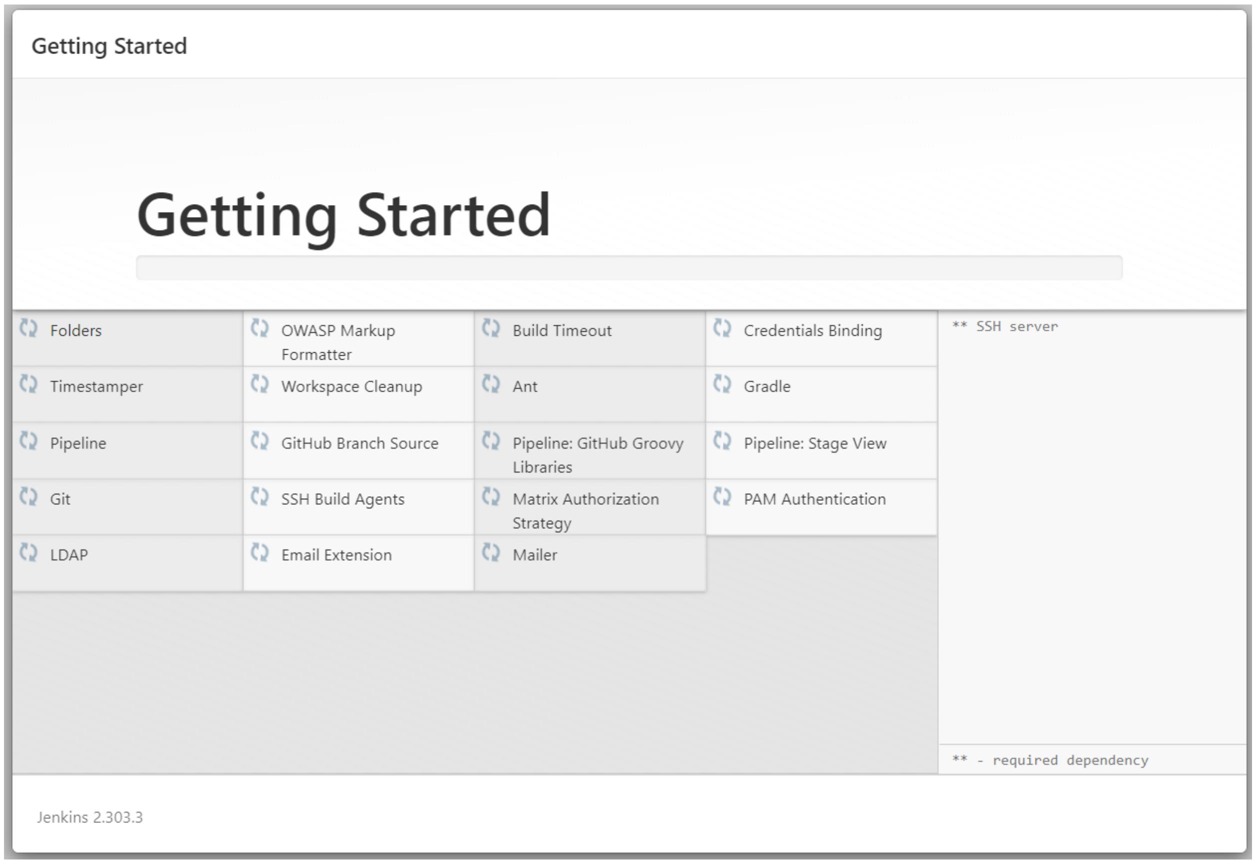
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

`/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 username and password -
```



Getting Started

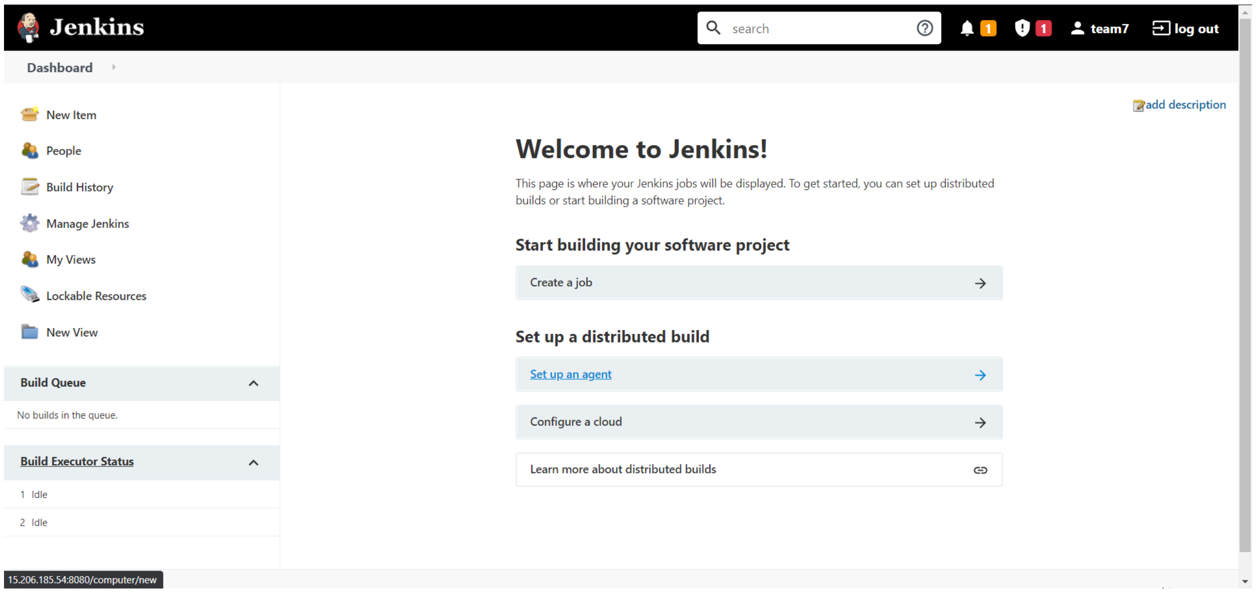
Getting Started

Folders	OWASP Markup Formatter	Build Timeout	Credentials Binding
Timestamper	Workspace Cleanup	Ant	Gradle
Pipeline	GitHub Branch Source	Pipeline: GitHub Groovy Libraries	Pipeline: Stage View
Git	SSH Build Agents	Matrix Authorization Strategy	PAM Authentication
LDAP	Email Extension	Mailer	

```
** SSH server
```

```
** - required dependency
```

Jenkins 2.303.3



Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job

Set up a distributed build

Set up an agent

Configure a cloud

Learn more about distributed builds

15.206.185.54:8080/computer/new

Installing docker on jenkins server
Same steps mentioned in earlier steps to install docker
To add current ubuntu usr and jenkins to docker group

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

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

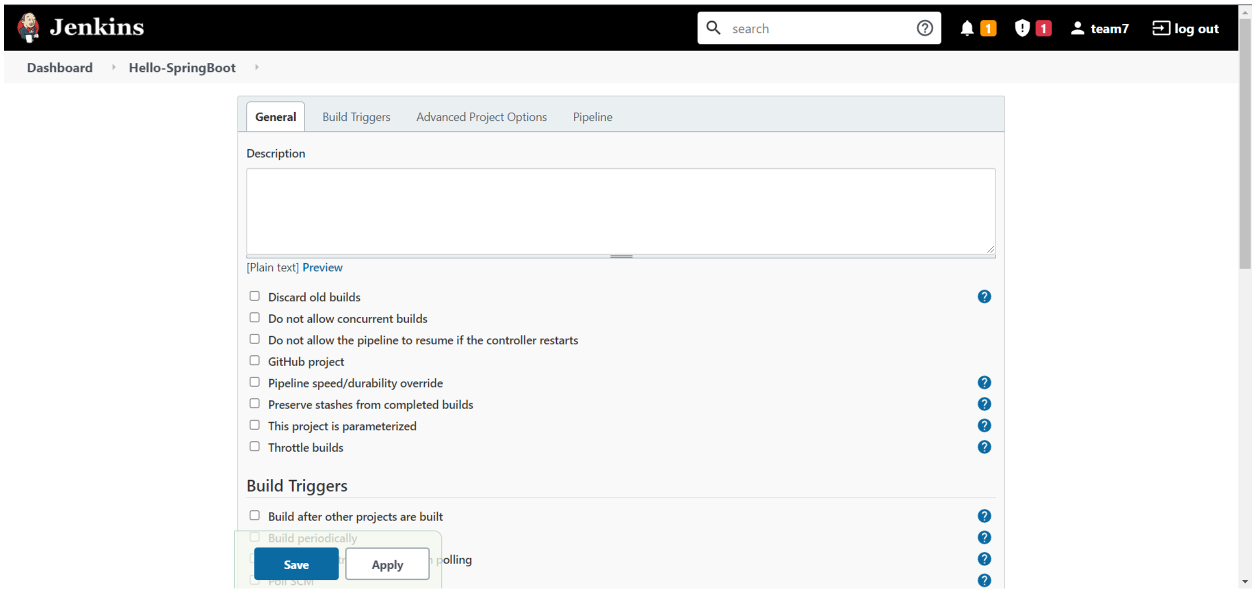
```
FROM openjdk:11
ARG JAR_FILE=build/libs/*.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
```

Deployment.yml file is present in the github repo.

Building the pipeline

Creating a pipeline

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



Jenkins | search | team7 | log out

Dashboard › Hello-SpringBoot ›

General | Build Triggers | Advanced Project Options | Pipeline

Description

[Plain text] Preview

- ☐ Discard old builds
- ☐ Do not allow concurrent builds
- ☐ Do not allow the pipeline to resume if the controller restarts
- ☐ GitHub project
- ☐ Pipeline speed/durability override
- ☐ Preserve stashes from completed builds
- ☐ This project is parameterized
- ☐ Throttle builds

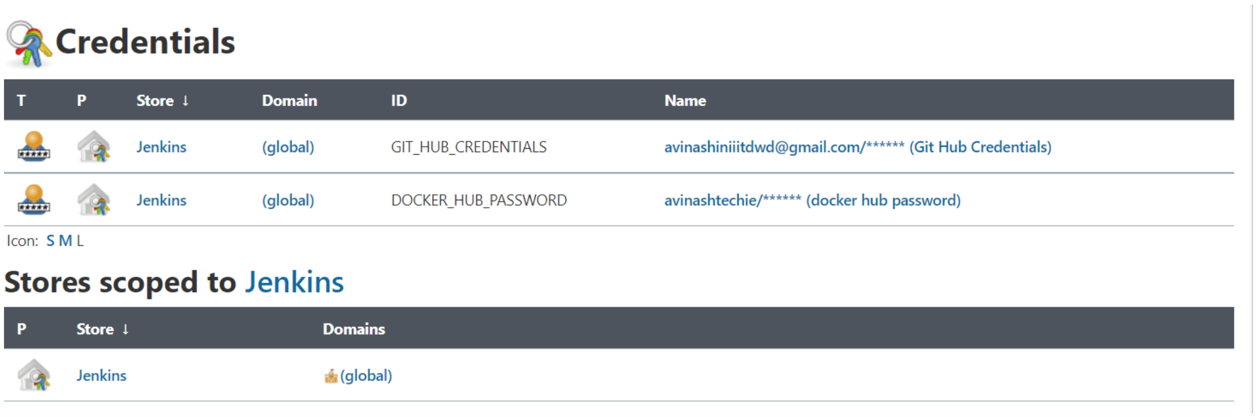
Build Triggers

- ☐ Build after other projects are built
- ☐ Build periodically

Save | Apply

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
setup github credentials
Go to jenkins dashboard -> manage jenkins -> manage credentials -> stores scoped to jenkins -> global -> add credentials Select username and password from select menu
Scope : select global scope

For username : Github username
For password : PAT github token
For ID : GIT_HUB_CREDENTIALS



Credentials

T	P	Store ↓	Domain	ID	Name
		Jenkins	(global)	GIT_HUB_CREDENTIALS	avinashiniiitdwd@gmail.com/****** (Git Hub Credentials)
		Jenkins	(global)	DOCKER_HUB_PASSWORD	avinashtechie/****** (docker hub password)

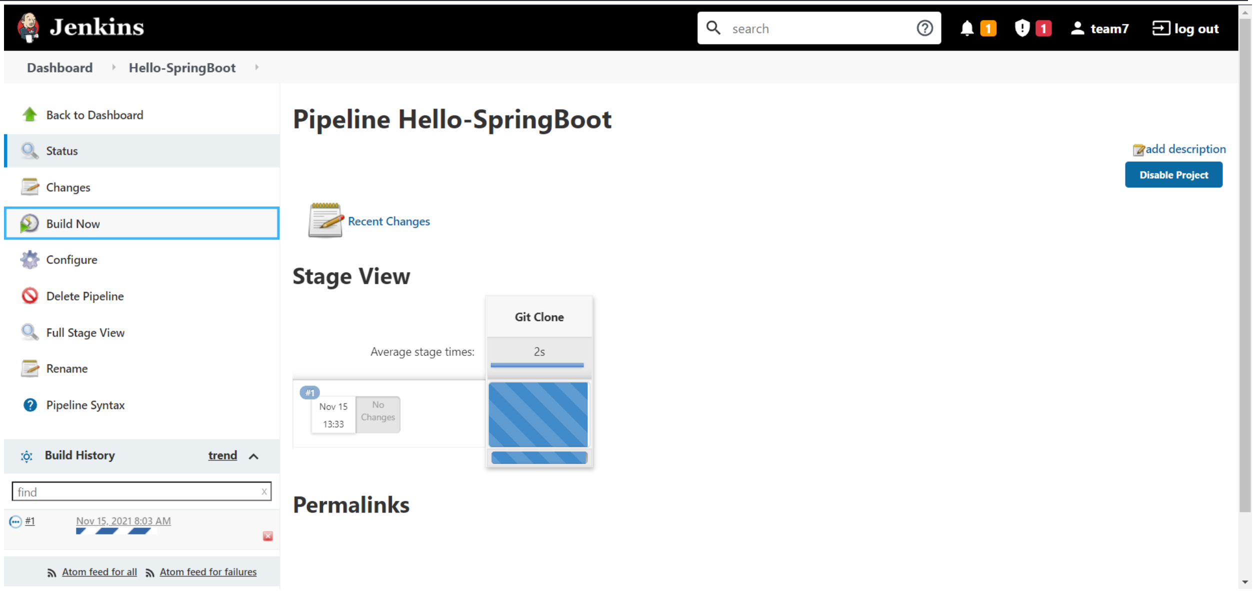
Icon: S M L

Stores scoped to Jenkins

P	Store ↓	Domains
	Jenkins	(global)

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/avinashtechlvr/springboot-with-docker.git'
 }

}
```



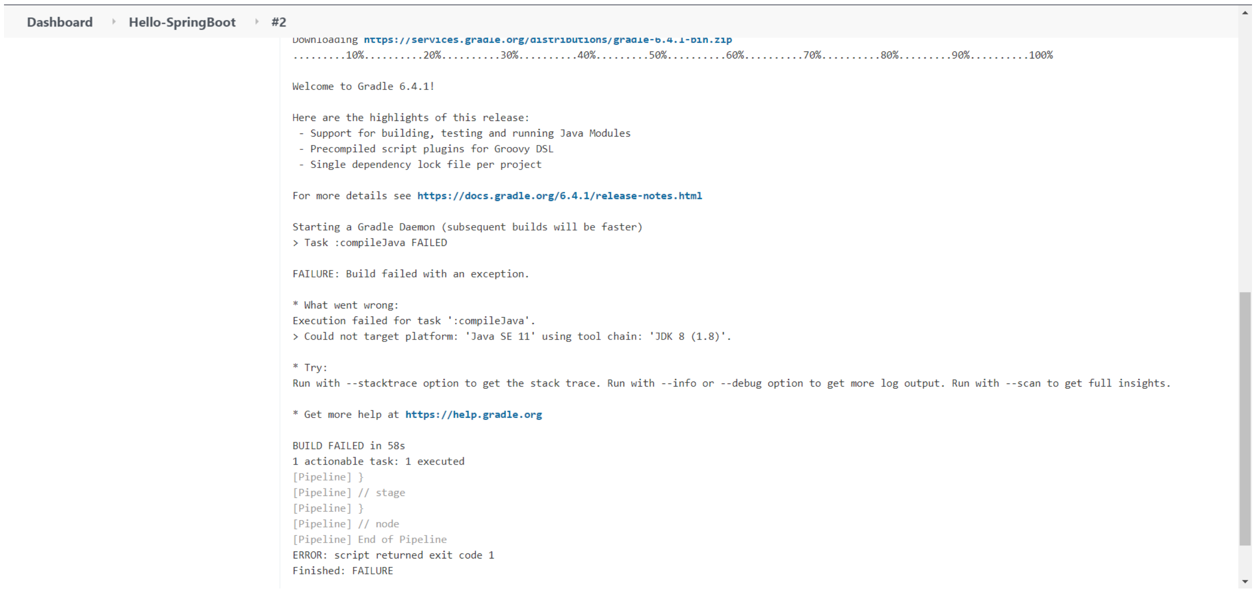
Pipeline Hello-SpringBoot

Stage View

Average stage times: 2s

#1 Nov 15, 2021 8:03 AM

Permalinks



Dashboard › Hello-SpringBoot › #2

```
Downloading 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 FAILED

FAILURE: Build failed with an exception.

* What went wrong:
Execution failed for task ':compileJava'.
> Could not target platform: 'Java SE 11' using tool chain: 'JDK 8 (1.8)'.

* Try:
Run with --stacktrace option to get the stack trace. Run with --info or --debug option to get more log output. Run with --scan to get full insights.

* Get more help at https://help.gradle.org

BUILD FAILED in 58s
1 actionable task: 1 executed
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
ERROR: script returned exit code 1
Finished: FAILURE
```

Building the spring boot application using gradle

setup gradle for building application :

Go to Jenkins dashboard -> new items -> enter item name:CICD-pipeline -> select pipeline -> click ok After creating a pipeline , select pipeline.

adding stage for building application using gradle

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

Build the docker image and tag it

Create a repository in the docker hub

```
We  have created a repository named team7.
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'
}
```

Adding docker login stage

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

```
adding stage for docker login for jenkins
credentials Id : ID of docker credentials
stage("Docker Login"){
withCredentials([string(credentialsId: 'DOCKER_HUB_PASSWORD', variable:
'PASSWORD')]) { sh 'docker login -u avinashtechie -p $PASSWORD' } }
```

Pushing the tagged docker image into docker hub

```
stage("Push Image to Docker Hub"){ sh 'docker push
avinashtechie/team7:docker-demo'}
```

add stage for login in to the kubernetes master node from jenkins server :
We do this by through ssh

add ssh pipeline plugin Go to jenkins dashboard -> manage plugins -> available -> in the search box type ssh pipeline steps -> select ssh pipeline plugin -> install without restart.

create password for kubernetes master server (ec2 ubuntu)

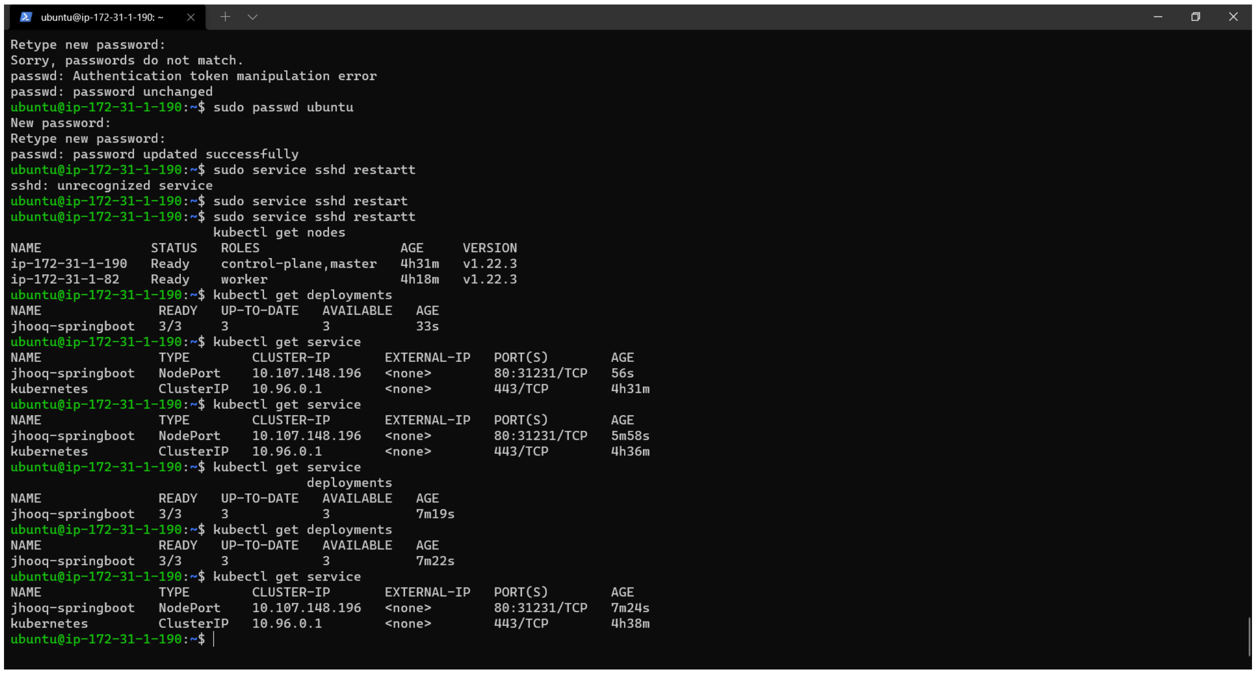
```
 $ sudo vi /etc/ssh/sshd_config Change the line passwordAuthentication from
"no" to "yes" Change the line permitRooLogin from "prohobit-password" to
"yes"
 $ sudo passwd ubuntu It will prompt to enter password.
$ sudo service sshd restart
```

```
adding the stage for this:
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 = 'public
ip' remote.user = 'ubuntu' remote.password = 'ubuntu' remote.allowAnyHosts
= true
}
```

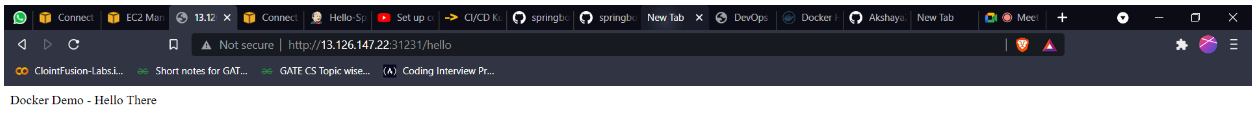
add stage for deploying the application on kubernetes

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
```



```
Retype new password:
Sorry, passwords do not match.
passwd: Authentication token manipulation error
passwd: password unchanged
ubuntu@ip-172-31-1-190:~$ sudo passwd ubuntu
New password:
Retype new password:
passwd: password updated successfully
ubuntu@ip-172-31-1-190:~$ sudo service sshd restartt
sshd: unrecognized service
ubuntu@ip-172-31-1-190:~$ sudo service sshd restart
ubuntu@ip-172-31-1-190:~$ sudo service sshd restartt
                          kubectl get nodes
NAME               STATUS   ROLES                  AGE     VERSION
ip-172-31-1-190    Ready    control-plane,master   4h31m   v1.22.3
ip-172-31-1-82     Ready    worker                 4h18m   v1.22.3
ubuntu@ip-172-31-1-190:~$ kubectl get deployments
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
jhooq-springboot   3/3     3            3           33s
ubuntu@ip-172-31-1-190:~$ kubectl get service
NAME               TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
jhooq-springboot   NodePort    10.107.148.196   <none>        80:31231/TCP   56s
kubernetes         ClusterIP   10.96.0.1        <none>        443/TCP        4h31m
ubuntu@ip-172-31-1-190:~$ kubectl get service
NAME               TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
jhooq-springboot   NodePort    10.107.148.196   <none>        80:31231/TCP   5m58s
kubernetes         ClusterIP   10.96.0.1        <none>        443/TCP        4h36m
ubuntu@ip-172-31-1-190:~$ kubectl get service
                                        deployments
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
jhooq-springboot   3/3     3            3           7m19s
ubuntu@ip-172-31-1-190:~$ kubectl get deployments
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
jhooq-springboot   3/3     3            3           7m22s
ubuntu@ip-172-31-1-190:~$ kubectl get service
NAME               TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
jhooq-springboot   NodePort    10.107.148.196   <none>        80:31231/TCP   7m24s
kubernetes         ClusterIP   10.96.0.1        <none>        443/TCP        4h38m
ubuntu@ip-172-31-1-190:~$ |
```



Not secure | http://13.126.147.22:31231/hello

Docker Demo - Hello There

Final Output of the deployed application