CREATE MACHINE 1 ON CONSOLE: with t2-medium

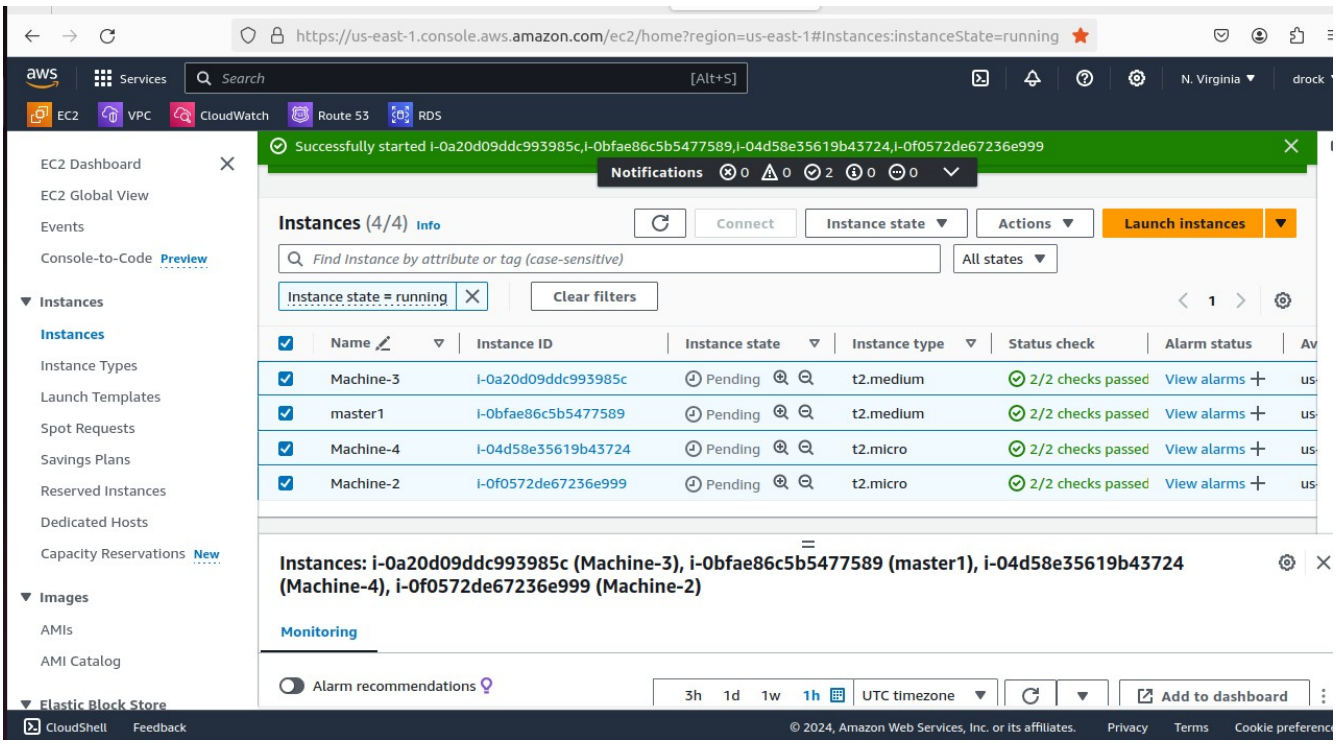
sudo nano main.tf

```
ubuntu@ip-172-31-87-230:~$ sudo apt update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu focal-security InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
All packages are up to date.
ubuntu@ip-172-31-87-230:~$ sudo nano main.tf
ubuntu@ip-172-31-87-230:~$
```

```
aws  Services  Search [Alt+S]  N. Virginia
EC2  VPC  CloudWatch  Route 53  RDS
  GNU nano 4.8                                   main.tf
provider "aws" {
  region     = "us-east-2"
  access_key = AKIAUUYXAGUJH64ZPN45
  secret_key = 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
}

resource "aws_instance" "K8s-master" {
  ami           = "ami-0cd59ecaf368e5ccf"
  instance_type = "t2.medium"
  key_name      = "newkey-virginia"

  tags = {
    Name = "Machine3"
  }
}

resource "aws_instance" "K8s-Slave1" {
  ami           = "ami-0cd59ecaf368e5ccf"
  instance_type = "t2.micro"
  key_name      = "newkey-virginia"
                                   [ Read 36 lines ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    M-U Undo      M-A Mark Text  M-] To Brack
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line M-E Redo      M-6 Copy Text  ^Q Where Was
```

```
provider "aws" {
  region     = "us-east-1"
  access_key = "AKIAQYFADVTUQ5VDNBX4"
  secret_key = "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"
}

resource "aws_instance" "K8s-master" {
  ami           = "ami-0cd59ecaf368e5ccf"
  instance_type = "t2.medium"
  key_name      = "newkey-virginia"

  tags = {
    Name = "Machine-3"
  }
}

resource "aws_instance" "K8s-Slave1" {
  ami           = "ami-0cd59ecaf368e5ccf"
  instance_type = "t2.micro"
```

```
  key_name      = "newkey-virginia"

  tags = {
    Name = "Machine-2"
  }
}

resource "aws_instance" "K8s-Slave2" {
  ami           = "ami-0cd59ecaf368e5ccf"
  instance_type = "t2.micro"
  key_name      = "newkey-virginia"

  tags = {
    Name = "Machine-4"
  }
}
```

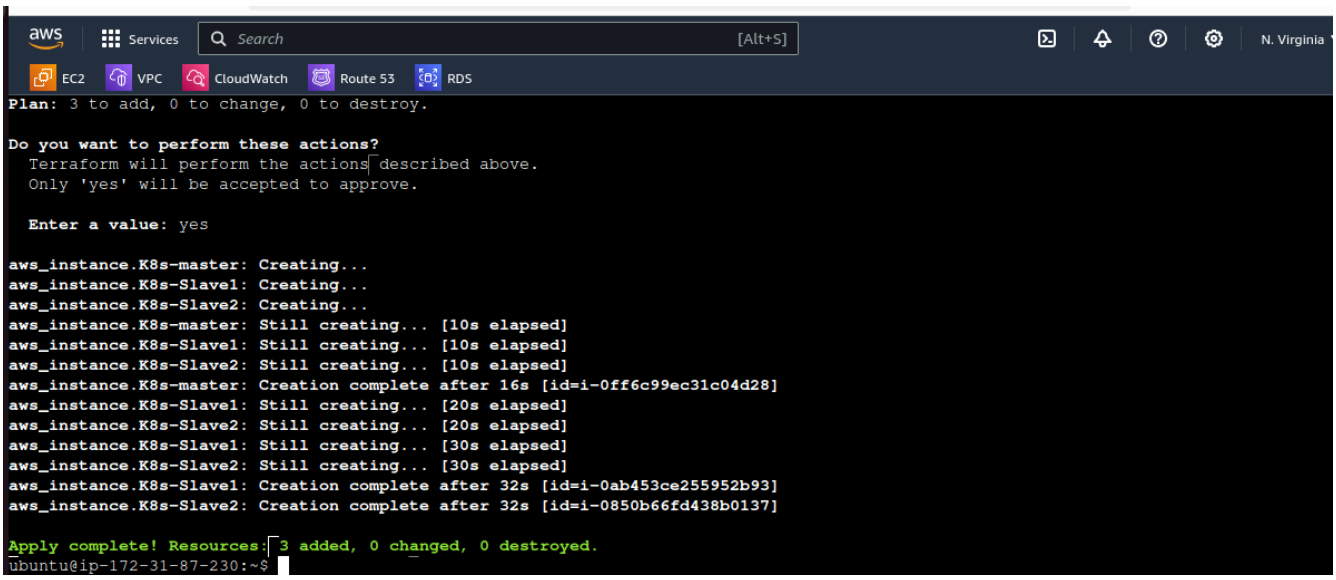


INSTALL TERRAFORM ON MACHINE1

```
sudo yum install -y yum-utils shadow-utils
sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
sudo yum -y install terraform
```

sudo apt update
sudo apt install -y gnupg software-properties-common curl
sudo mkdir -p /etc/apt/trusted.gpg.d
sudo curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /etc/apt/trusted.gpg.d/hashicorp.gpg
echo "deb [arch=amd64] https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list
sudo apt update
sudo apt install terraform

terraform init
terraform plan
terraform apply



INSTALL ANSIBLE IN MACHINE 1
sudo apt install software-properties-common
sudo apt-add-repository --yes --update ppa:ansible/ansible
sudo apt install ansible

amazon linux
sudo yum update
sudo yum install ansible

sudo yum update

sudo yum install ansible

```
  Verifying        : ansible-8.3.0-1.amzn2023.0.1.noarch

Installed:
  ansible-8.3.0-1.amzn2023.0.1.noarch        ansible-core-2.15.3-1.amzn2023.0.3.x86_64        sshpass-1.09-6.amzn2023.0

Complete!
[ec2-user@ip-172-31-84-186 ~]$ ansible --version
ansible [core 2.15.3]
  config file = None
  configured module search path = ['/home/ec2-user/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3.9/site-packages/ansible
  ansible collection location = /home/ec2-user/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.9.16 (main, Sep  8 2023, 00:00:00) [GCC 11.4.1 20230605 (Red Hat 11.4.1-2)] (/usr/bin/python3.9)
  jinja version = 3.1.2
  libyaml = True
[ec2-user@ip-172-31-84-186 ~]$
```

<u>Setup the ssh</u>

ssh-keygen
cd .ssh
ls
cat id_rsa.pub
copy to paste in the file .ssh/authorized_keys of other servers
sudo nano .ssh/authorized_keys

<u>Setup host</u>
cd
sudo nano */etc/*ansible/hosts

```
ubuntu@ip-172-31-87-230:~/.ssh$ cd
ubuntu@ip-172-31-87-230:~$ sudo nano /etc/ansible/hosts
```

i-0d44bbbaaa7669853 (Machine1)

PublicIPs: 54.210.116.17 PrivateIPs: 172.31.87.230

[master]
private ip of machine3

[slaves]
private ip of machine2 and 4

```
GNU nano 5.8                                   /etc/ansible/hosts
[master]
172.31.87.1

[slaves]
172.31.91.215
172.31.90.147

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo
```

i-0697b0e98fd874294 (machine1)

PublicIPs: 44.208.34.24 PrivateIPs: 172.31.84.186

ansible -m ping all

```
ubuntu@ip-172-31-87-230:~$ ansible -m ping all
172.31.85.58 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
172.31.83.163 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
172.31.91.210 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
ubuntu@ip-172-31-87-230:~$
```

Worker1: Jenkins, Java

Worker2: Docker, Kubernetes
Worker3: Java, Docker, Kubernetes
Worker4: Docker, Kubernetes

Infrastructure Creation and Configuration Management

Write 3 script to install as advised

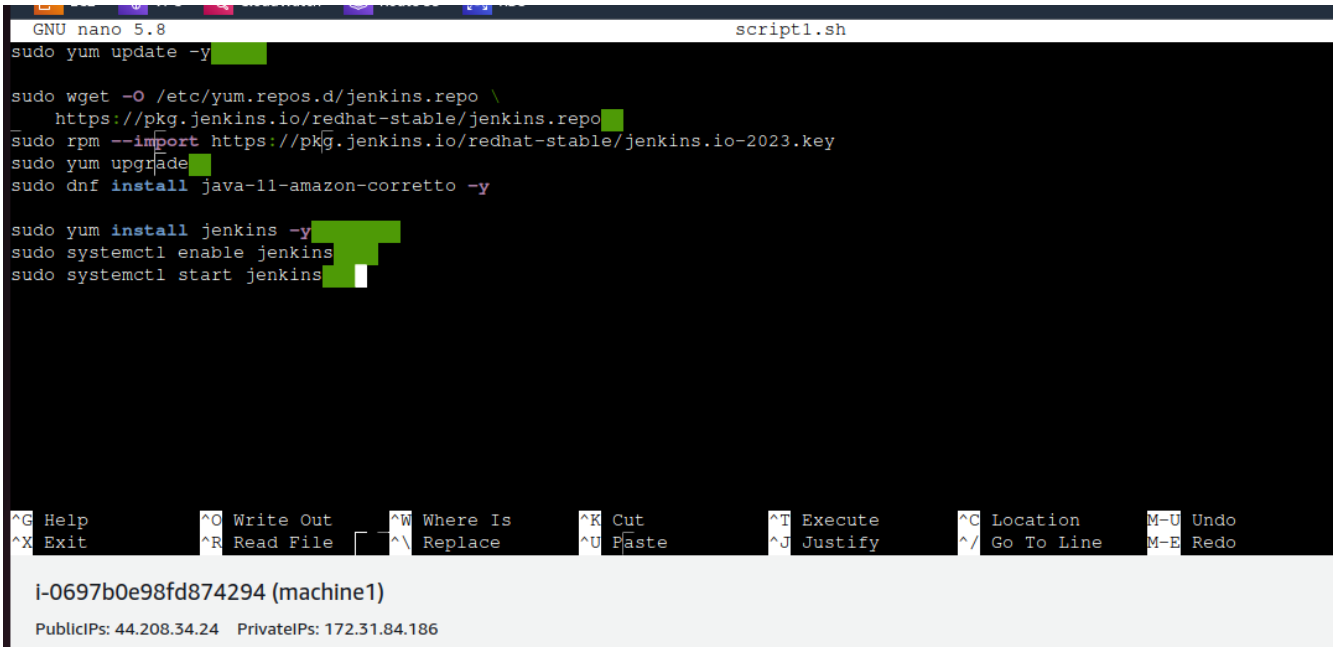
Script1 install Jenkins / Java to Machine 1
Script2 install Docker / Kubernetes to Machine 2 and 4
Script3 install Java/Docker/Kubernetes to Machine 3

sudo nano script1.sh

search google for : https://www.jenkins.io/doc/book/installing/linux/

```
sudo yum update –y

sudo wget -O /etc/yum.repos.d/jenkins.repo \
    https://pkg.jenkins.io/redhat-stable/jenkins.repo

sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
sudo yum upgrade

sudo dnf install java-11-amazon-corretto -y

sudo yum install jenkins -y
sudo systemctl enable jenkins
```

```
sudo systemctl start jenkins
```



for ubuntu: weekly release

```
sudo apt get upgrade
sudo apt install openjdk-11-jdk -y
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian/jenkins.io-2023.key
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins -y
```

aws Services Search [Alt+S]

EC2 VPC CloudWatch Route 53 RDS

```
  GNU nano 6.2                                                    script1.sh *
sudo apt get upgrade
sudo apt install openjdk-11-jdk -y
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian/jenkins.io-2023.key
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins -y

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/
```

i-0cfd8aabf2c09ac2d (machine1)

PublicIPs: 52.23.182.74 PrivateIPs: 172.31.94.177

```
  GNU nano 4.8                                                    script1.sh
sudo apt get upgrade
sudo apt install openjdk-11-jdk -y
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian/jenkins.io-2023.key
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins -y
^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E
```

i-0d44bbbaaa7669853 (Machine1)

PublicIPs: 54.210.116.17 PrivateIPs: 172.31.87.230

Script2
sudo nano script2.sh

sudo apt update

sudo apt install openjdk-11-jdk -y

sudo apt install docker.io -y

sudo apt update

sudo apt upgrade -y

sudo apt install -y curl apt-transport-https ca-certificates software-properties-common

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

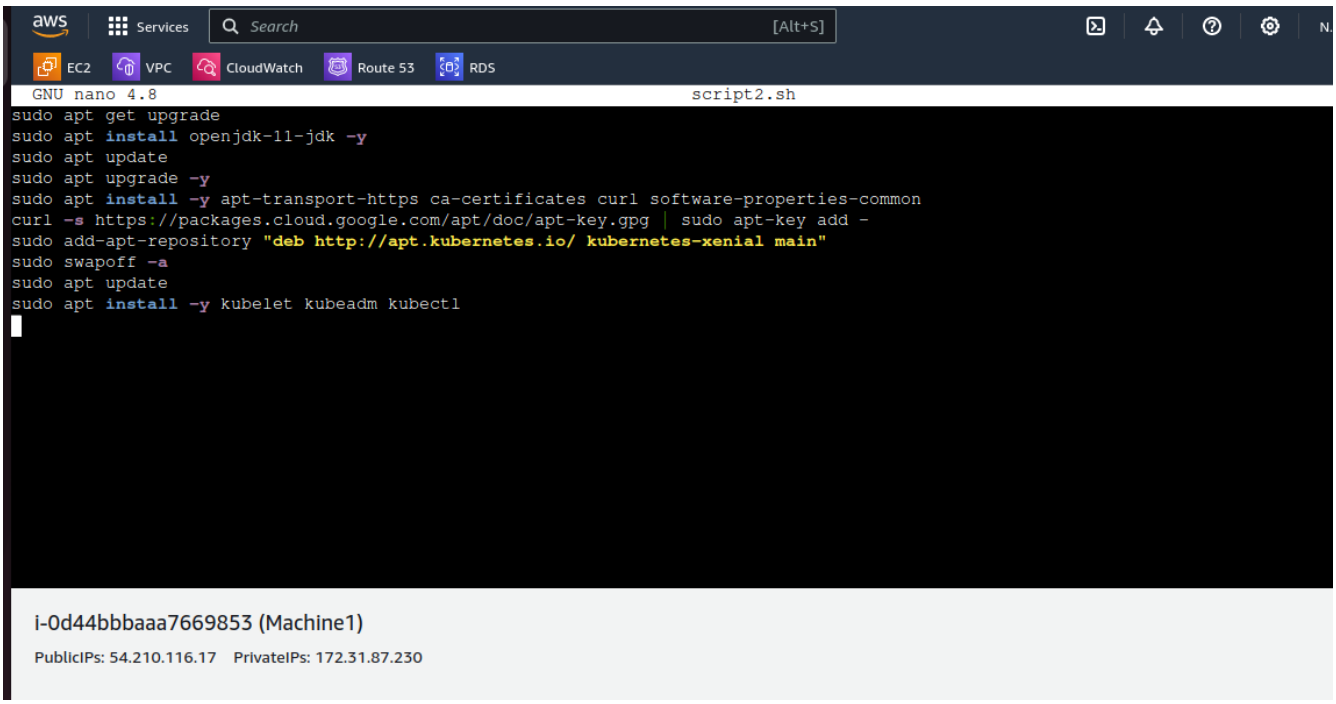
sudo add-apt-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"

sudo swapoff -a

sudo apt update

sudo apt install -y kubelet kubeadm kubectl

sudo apt get upgrade
sudo apt install openjdk-11-jdk -y
sudo apt update
sudo apt upgrade -y
sudo apt install -y apt-transport-https ca-certificates curl software-properties-common
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
sudo add-apt-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"
sudo swapoff -a
sudo apt update
sudo apt install -y kubelet kubeadm kubectl



Script3

sudo apt update
sudo apt upgrade -y
sudo apt install -y apt-transport-https ca-certificates curl software-properties-common
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
sudo add-apt-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"
sudo swapoff -a
sudo apt update
sudo apt install -y kubelet kubeadm kubectl

```
GNU nano 4.8                                    script3.sh
sudo apt update
sudo apt upgrade -y
sudo apt install -y apt-transport-https ca-certificates curl software-properties-common
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
sudo add-apt-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"
sudo swapoff -a
sudo apt update
sudo apt install -y kubelet kubeadm kubectl

                                    [ Read 8 lines ]
^G Get Help   ^O Write Out   ^W Where Is   ^K Cut Text   ^J Justify   ^C Cur Pos   M-U Undo   M-A Mar
^X Exit       ^R Read File   ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line M-E Redo   M-6 Copy
```

i-0d44bbbaaa7669853 (Machine1)

PublicIPs: 54.210.116.17 PrivateIPs: 172.31.87.230

```
ubuntu@ip-172-31-87-230:~$ ls
main.tf  script1.sh  script2.sh  script3.sh  terraform.tfstate  terraform.tfstate.backup
ubuntu@ip-172-31-87-230:~$
```

i-0d44bbbaaa7669853 (Machine1)

PublicIPs: 54.210.116.17 PrivateIPs: 172.31.87.230

```
ubuntu@ip-172-31-87-230:~$ sudo nano play.yml
ubuntu@ip-172-31-87-230:~$
```

i-0d44bbbaaa7669853 (Machine1)

PublicIPs: 54.210.116.17 PrivateIPs: 172.31.87.230

sudo nano play.yml

```
---
- name: Install Jenkins and Java on Machine1
  hosts: localhost
  become: true
  tasks:
    - name: Running script1
      script: script1.sh
```

```
- name: Install K8s and Java on Machine3
  hosts: master
  become: true
  tasks:
    - name: Running script2
      script: script2.sh

- name: Install K8s on Machine2 and 4
  hosts: slaves
  become: true
  tasks:
    - name: Running script3
      script: script3.sh
```

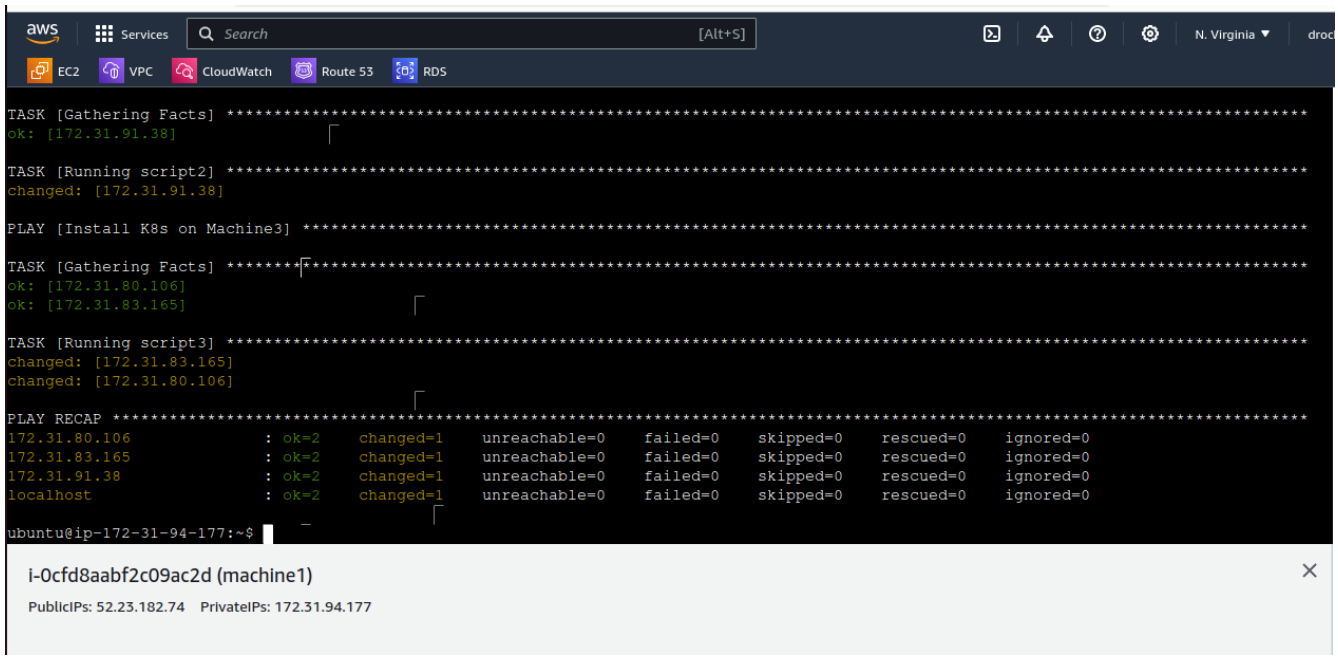
```
  GNU nano 4.8                                          play.yml
--
- name: Install Jenkins and Java on Machine1
  hosts: localhost
  become: true
  tasks:
    - name: Running script1
      script: script1.sh

- name: Install K8s and Java on Machine2
  hosts: master
  become: true
  tasks:
    - name: Running script2
      script: script2.sh

- name: Install K8s on Machine3
  hosts: slaves
  become: true
  tasks:
    - name: Running script3

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo       M-A Mar
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E Redo       M-6 Cop
```

i-0d44bbbaaa7669853 (Machine1)

PublicIPs: 54.210.116.17 PrivateIPs: 172.31.87.230

ansible-playbook play.yml --syntax-check
ansible-playbook play.yml –check

ansible-playbook play.yml



ON LOCALHOST

```
[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 172.31.93.16:6443 --token frghwy.ak4qwd9mm6nuyadm \
        --discovery-token-ca-cert-hash sha256:11c04936a95e0d817d729fa11e794b9c886b88f46af0fdb981df14945b56c76a
ubuntu@ip-172-31-93-16:~$
```

i-0a20d09ddc993985c (Machine-3)

PublicIPs: 54.224.45.40 PrivateIPs: 172.31.93.16

sudo kubeadm join 172.31.93.16:6443 --token frghwy.ak4qwd9mm6nuyadm \
--discovery-token-ca-cert-hash sha256:11c04936a95e0d817d729fa11e794b9c886b88f46af0fdb981df14945b56c76a on slaves

```
ubuntu@ip-172-31-17-68:~$ sudo kubeadm join 172.31.93.16:6443 --token frghwy.ak4qwd9mm6nuyadm \
>         --discovery-token-ca-cert-hash sha256:11c04936a95e0d817d729fa11e794b9c886b88f46af0fdb981df14945b56c76a
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

ubuntu@ip-172-31-17-68:~$
```

i-0f0572de67236e999 (Machine-2)

PublicIPs: 34.224.80.79 PrivateIPs: 172.31.17.68

```
ubuntu@ip-172-31-16-243:~$ sudo kubeadm join 172.31.93.16:6443 --token frghwy.ak4qwd9mm6nuyadm \
>         --discovery-token-ca-cert-hash sha256:11c04936a95e0d817d729fa11e794b9c886b88f46af0fdb981df14945b56c76a
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

ubuntu@ip-172-31-16-243:~$
```

i-04d58e35619b43724 (Machine-4)

PublicIPs: 34.227.190.135 PrivateIPs: 172.31.16.243

```
clusterrole.rbac.authorization.k8s.io/calico-cni-plugin created
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-cni-plugin created
daemonset.apps/calico-node created
deployment.apps/calico-kube-controllers created
ubuntu@ip-172-31-93-16:~$ kubectl get nodes
NAME               STATUS     ROLES           AGE     VERSION
ip-172-31-16-243   NotReady   <none>          5m24s   v1.28.8
ip-172-31-17-68    NotReady   <none>          6m30s   v1.28.8
ip-172-31-93-16    Ready      control-plane   7m13s   v1.28.8
ubuntu@ip-172-31-93-16:~$ kubectl get nodes
NAME               STATUS   ROLES           AGE     VERSION
ip-172-31-16-243   Ready    <none>          6m51s   v1.28.8
ip-172-31-17-68    Ready    <none>          7m57s   v1.28.8
ip-172-31-93-16    Ready    control-plane   8m40s   v1.28.8
ubuntu@ip-172-31-93-16:~$
```

i-0a20d09ddc993985c (Machine-3)

PublicIPs: 54.224.45.40 PrivateIPs: 172.31.93.16

On Master and Slave:

sudo apt-get update
sudo apt install docker.io -y
sudo apt-get install -y apt-transport-https ca-certificates curl gpg
sudo mkdir -p -m 755 /etc/apt/keyrings
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.28/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo systemctl enable --now kubelet

On Master:
sudo kubeadm init --apiserver-advertise-address=privateipofmaster

Note: You need to replace “private_ip_of_master” with the actual private ip of your kubernetes master.

Paste the Token on Slave

On Master

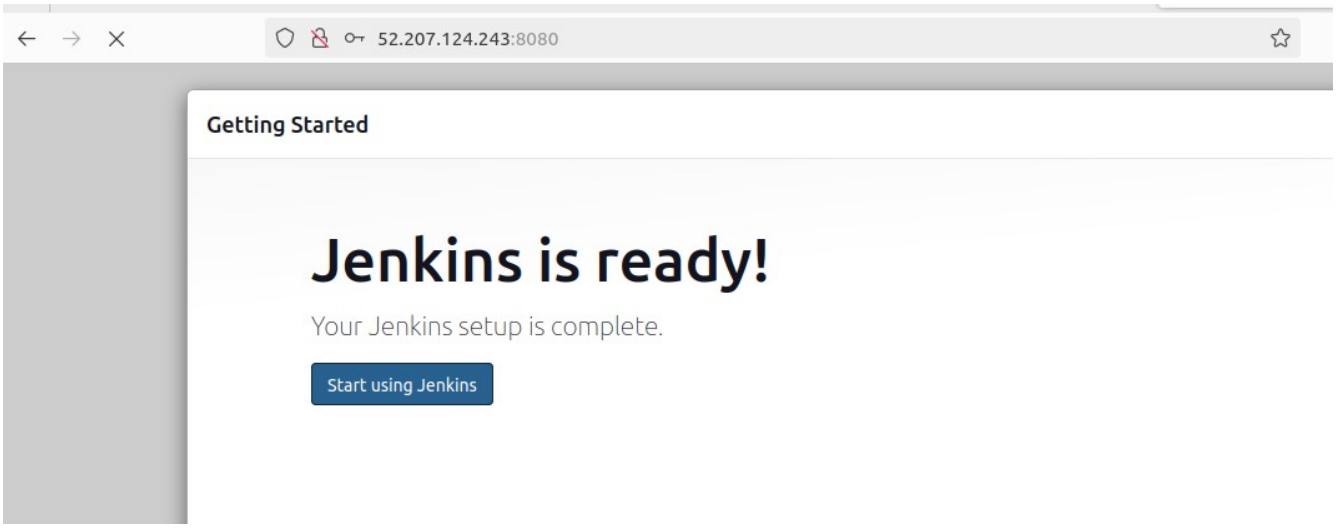
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

Installing Calico
curl https://raw.githubusercontent.com/projectcalico/calico/v3.27.2/manifests/calico.yaml -O

kubectl apply -f calico.yaml

kubectl get nodes

ON MACHINE ONE:



manage jenkins ----- nodes
add kubernetes master as the node
---new node---supply name-- pick Permanent Agent—create
Remote root directory-- *home/*ubuntu/jenkins
Launch method-- launch Agent via ssh
Host --- Private ip
credential: private key

52.207.124.243:8080/manage/computer/createItem

Dashboard > Manage Jenkins > Nodes >

Jenkins Credentials Provider: Jenkins

Username

ubuntu

Treat username as secret ?

Private Key

Enter directly

Key

Enter New Secret Below


```
4TQKZylOoEpT3FTdn2f76pZFV0j3GVfVwWG5YtFrkcTbIdkz0zV0ulrKF+8MvD0URK6rt0
Xw2jt9LQ7AxG+tAAAAF3VidW50dUBpcC0xNzItMzEtODAtMTUzAQID
-----END OPENSSH PRIVATE KEY-----
```


Not secure 3.86.158.196:8080/manage/computer/

Gmail YouTube Maps Gmail Utiva Online Course... Socrative Discord | Friends Home - Micros... All Bookmarks

Jenkins Search (CTRL+K) subby log out

Dashboard > Manage Jenkins > Nodes >

Nodes

Clouds

Build Queue

No builds in the queue.

Build Executor Status

Built-In Node

1 Idle

2 Idle

k8sm

1 Idle

Nodes

New Node Configure Monitors

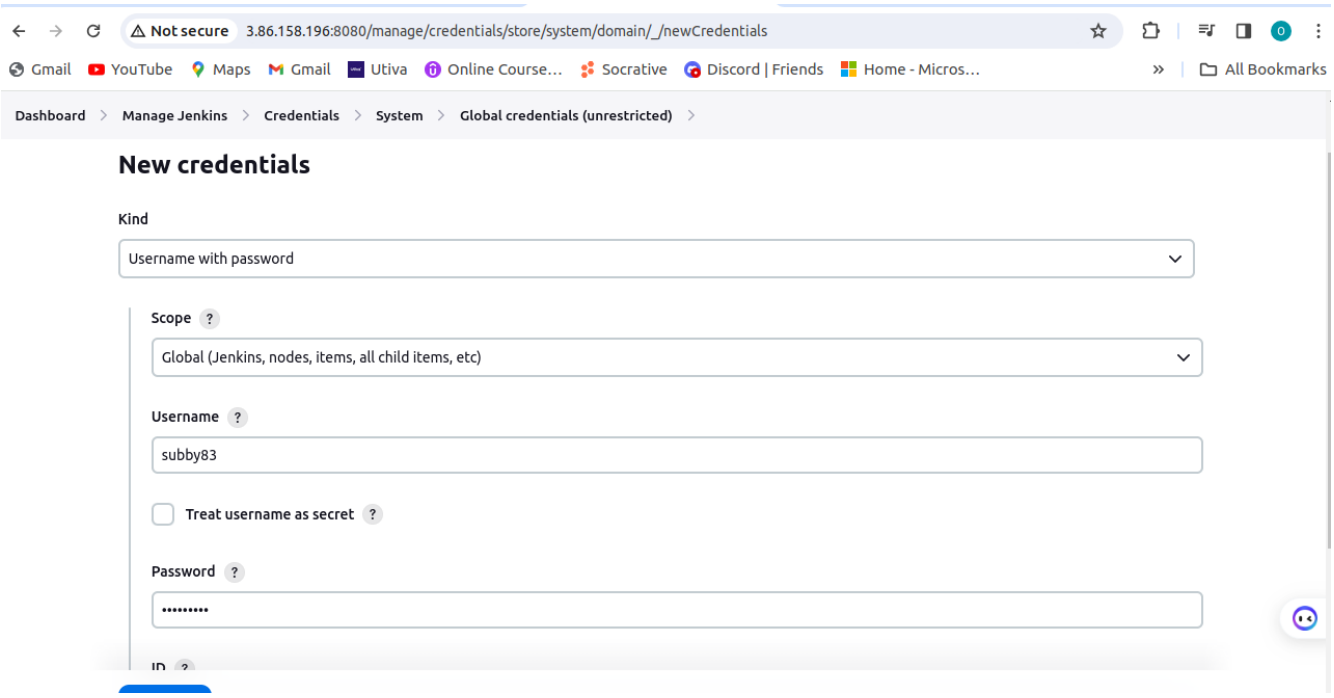
0.37 sec

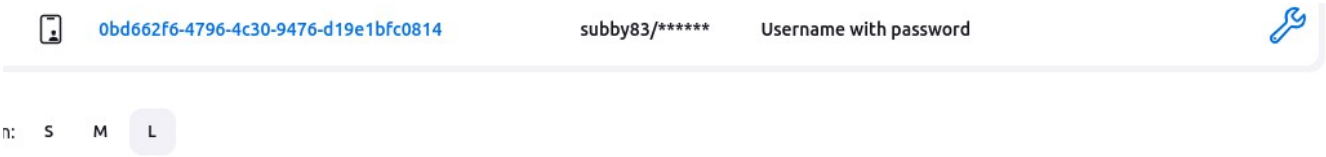
S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	15.11 GiB	0 B	15.11 GiB	0ms
	k8sm	Linux (amd64)	In sync	2.87 GiB	0 B	2.87 GiB	30ms
	Data obtained	0.38 sec	0.37 sec	0.36 sec	0.36 sec	0.37 sec	0.37 sec

Icon: S M L

Legend

CREATE CREDENTIAL FOR DOCKER HUB ACCOUNT
dashboard – manage jenkins – credentials
global – add credential
Put docker hub username and password --- create

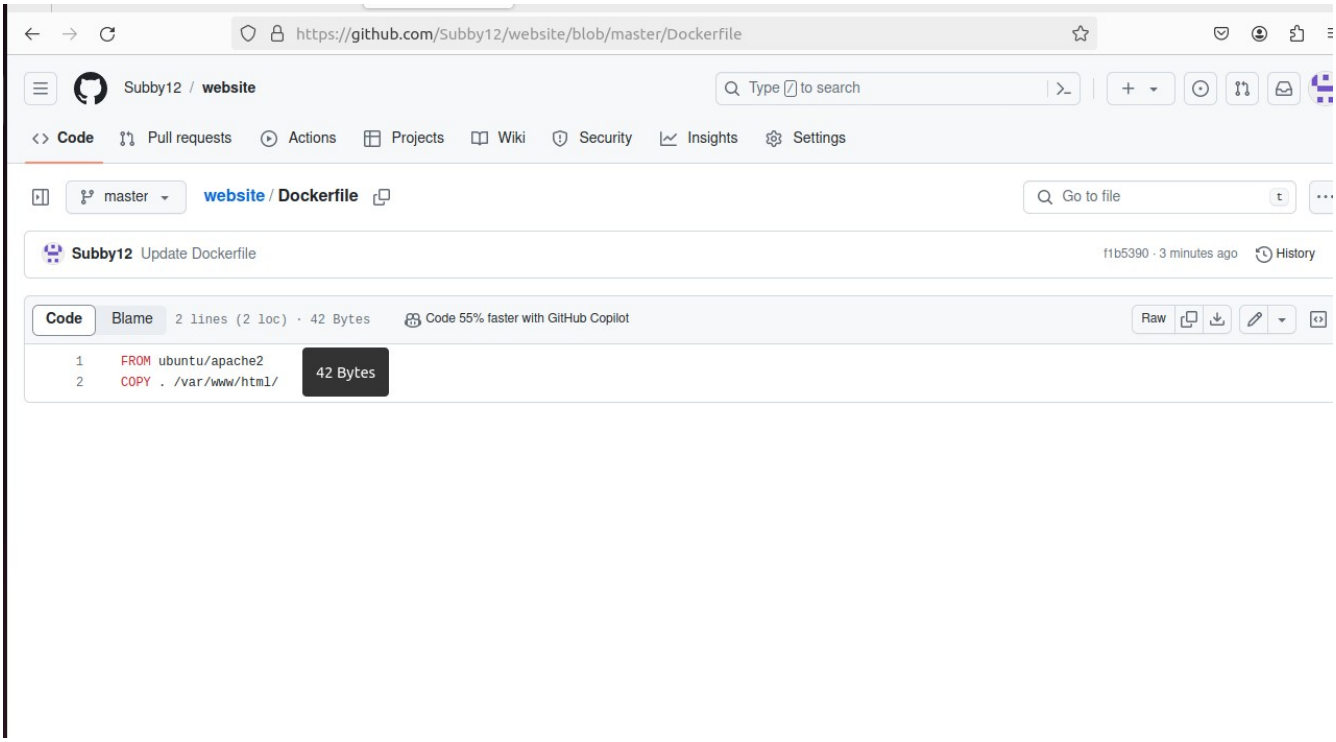




0bd662f6-4796-4c30-9476-d19e1bfc0814

CREATE REPOSITORY: to use in order to create the pipeline.

fork the repo given
create dockerfile and add: 2 lines



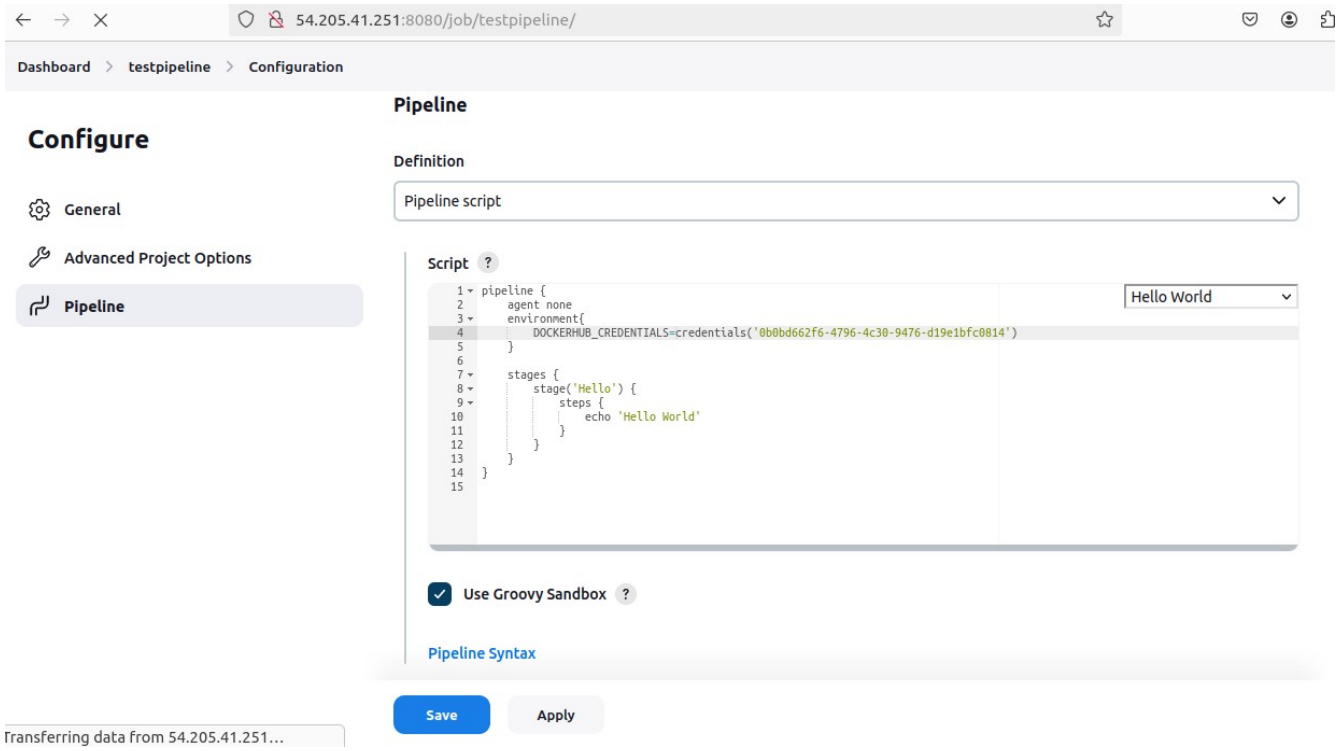
CREATE Jenkins file
Dashboard – new item – Name
Pick PIPELINE -----
pipeline script

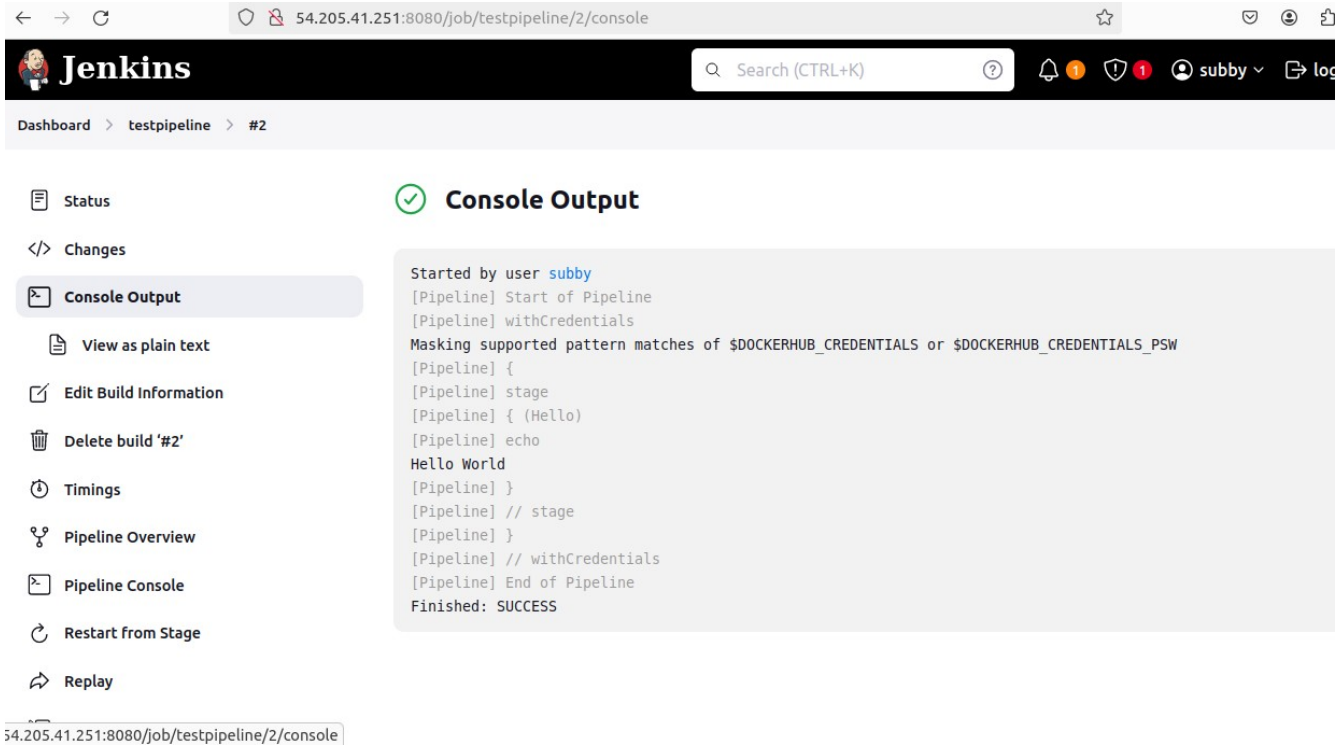
the stages

STAGE ONE: TESTINGl HELLO WORLD

```
pipeline {
    agent none
    environment{
        DOCKERHUB_CREDENTIALS=credentials('0bd662f6-4796-4c30-9476-d19e1bfc0814')
```

```
    }

    stages {
        stage('Hello') {
            steps {
                echo 'Hello World'
            }
        }
    }
}
```

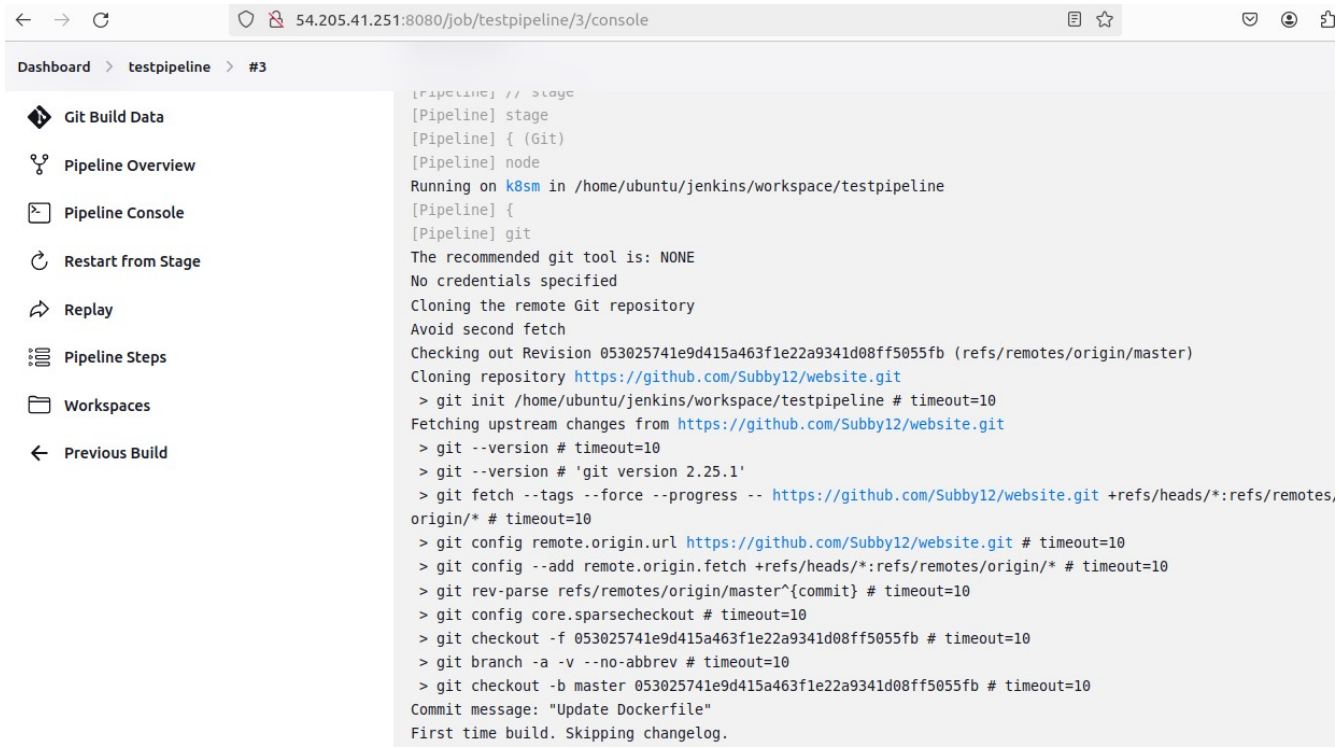




2ND STAGE – GIT

```
pipeline {
    agent none
    environment{
        DOCKERHUB_CREDENTIALS=credentials('0bd662f6-4796-4c30-9476-d19e1bfc0814')
    }

    stages {
        stage('Hello') {
            steps {
                echo 'Hello World'
            }
        }
        stage('Git') {
            agent{
```

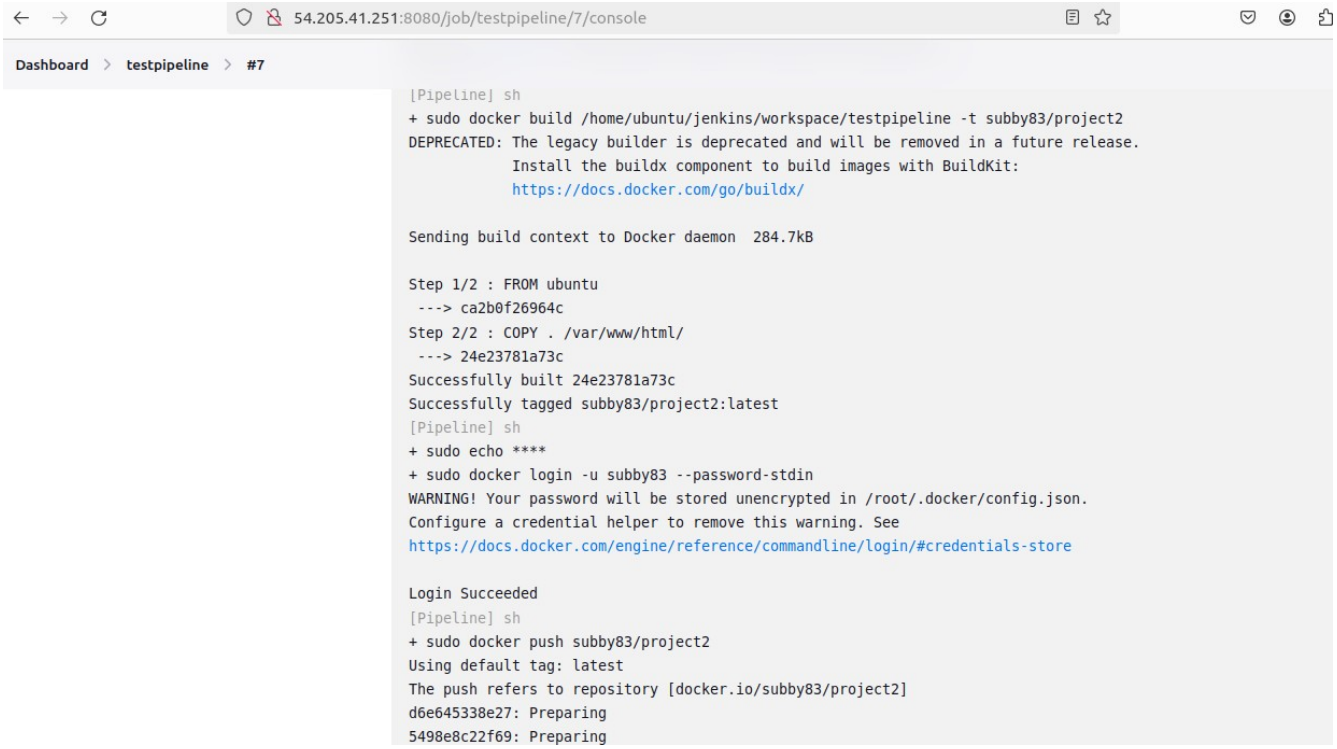
```
            label 'k8sm'
        }
        steps {
            git 'https://github.com/Subby12/website.git'
        }
    }


  }
}
```

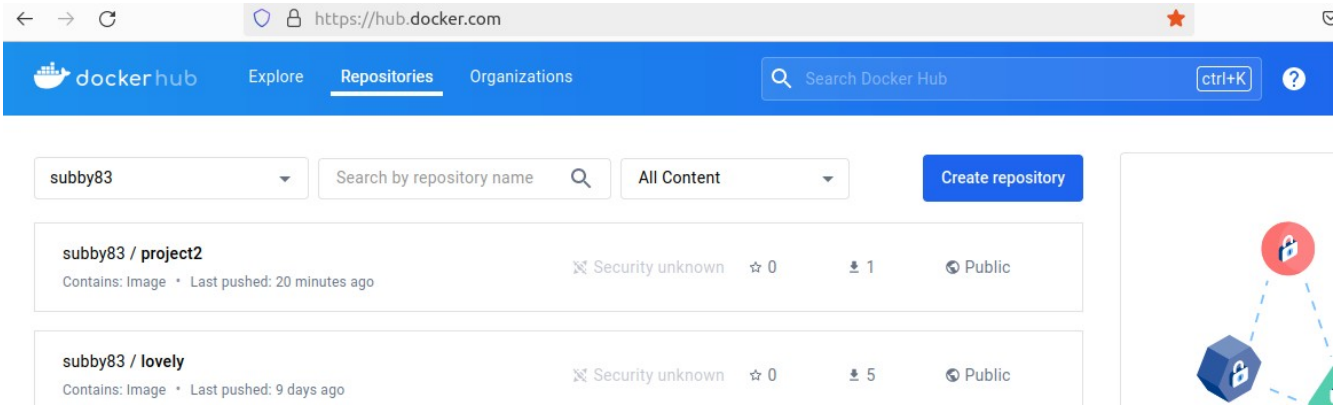


STAGE 3: DOCKER

```
pipeline {
    agent none
    environment{
        DOCKERHUB_CREDENTIALS=credentials('0bd662f6-4796-4c30-9476-d19e1bfc0814')
    }
```

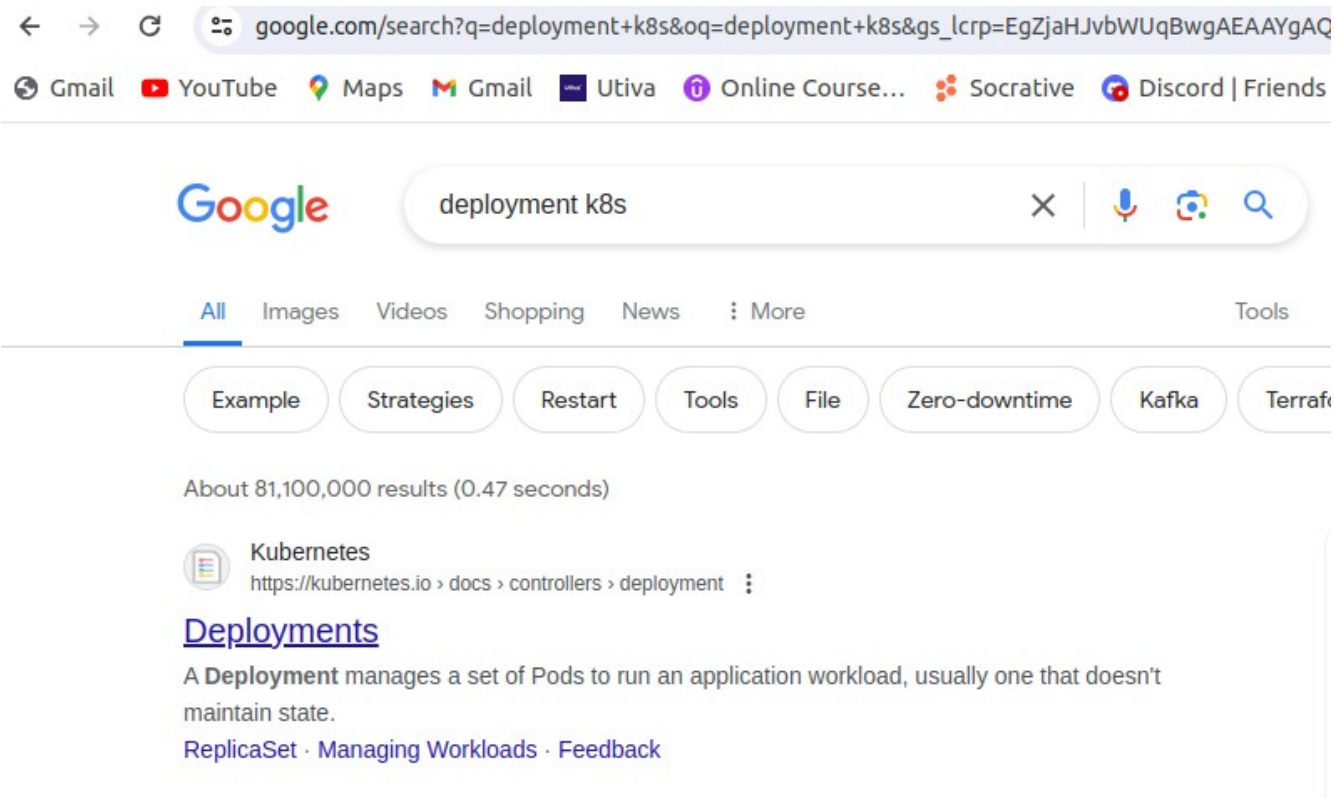
```
    stages {
        stage('Hello') {
            steps {
                echo 'Hello World'
            }
        }
        stage('Git') {
            agent{
                label 'k8sm'
            }
            steps {
                git 'https://github.com/Subby12/website.git'
            }
        }
        stage('Docker') {
            agent{
                label 'k8sm'
            }
            steps {
                sh 'sudo docker build /home/ubuntu/jenkins/workspace/testpipeline -t subby83/project2'
                sh 'sudo echo $DOCKERHUB_CREDENTIALS_PSW | sudo docker login -u
$DOCKERHUB_CREDENTIALS_USR --password-stdin'
                sh 'sudo docker push subby83/project2'
            }
        }
    }
}
```





4TH STAGE

search ‘deployment k8s’



We will create 2 more file: for: As per the requirement in the production server, you need to use the Kubernetes cluster and the containerized code from Docker Hub should be deployed with 2 replicas. Create a NodePort service and configure the same for port 30008.

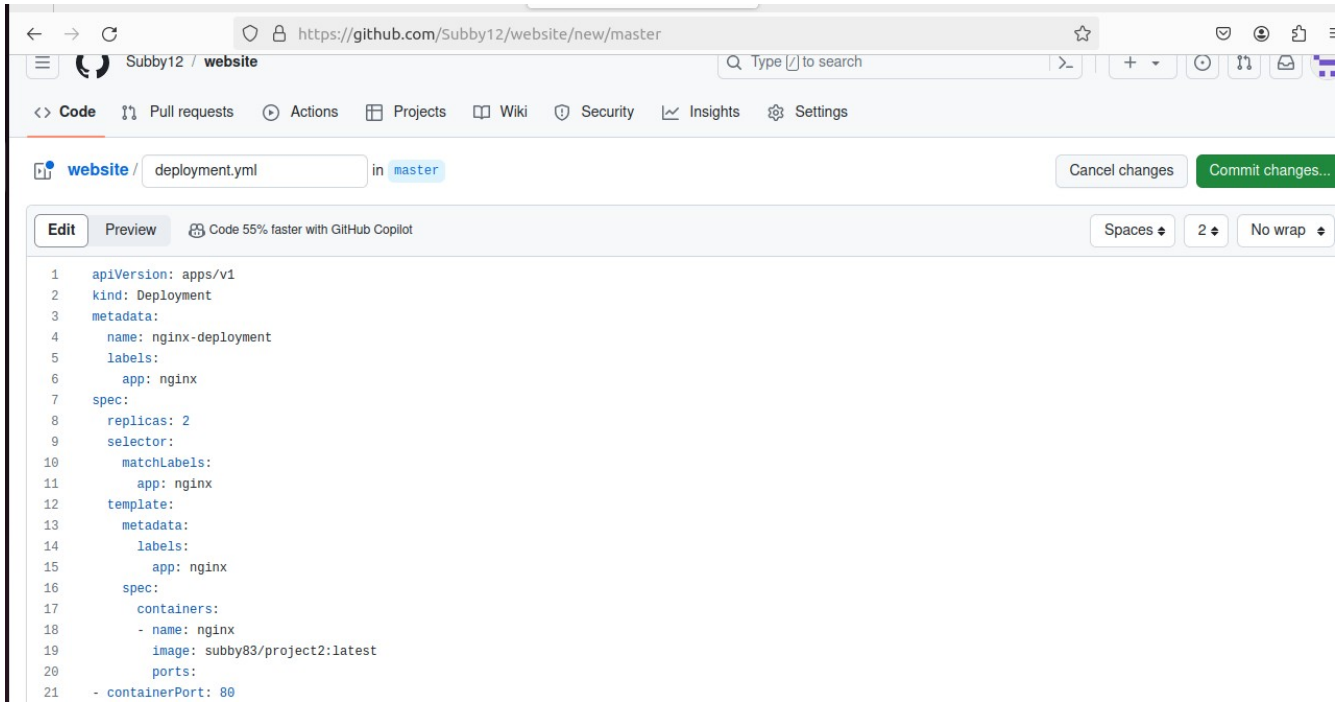
We will create 2 more files in the github:

1: deployment.yaml

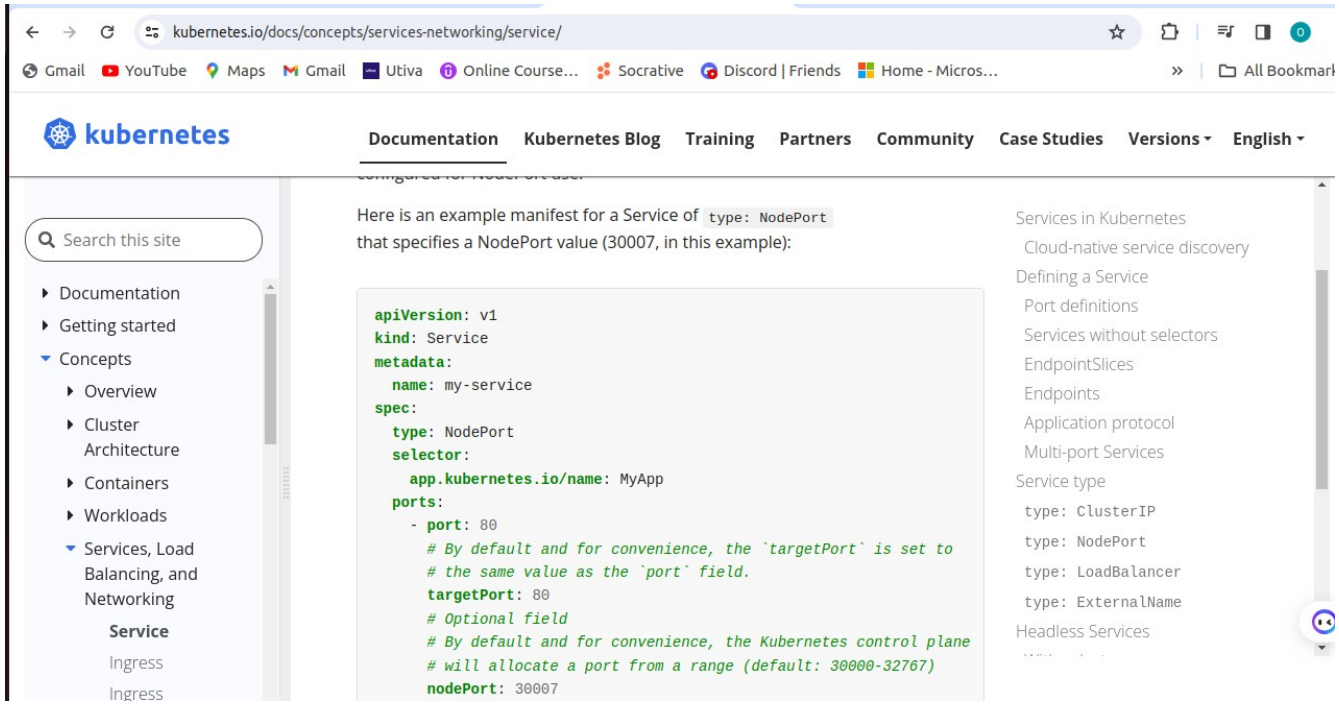
2: service.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
```

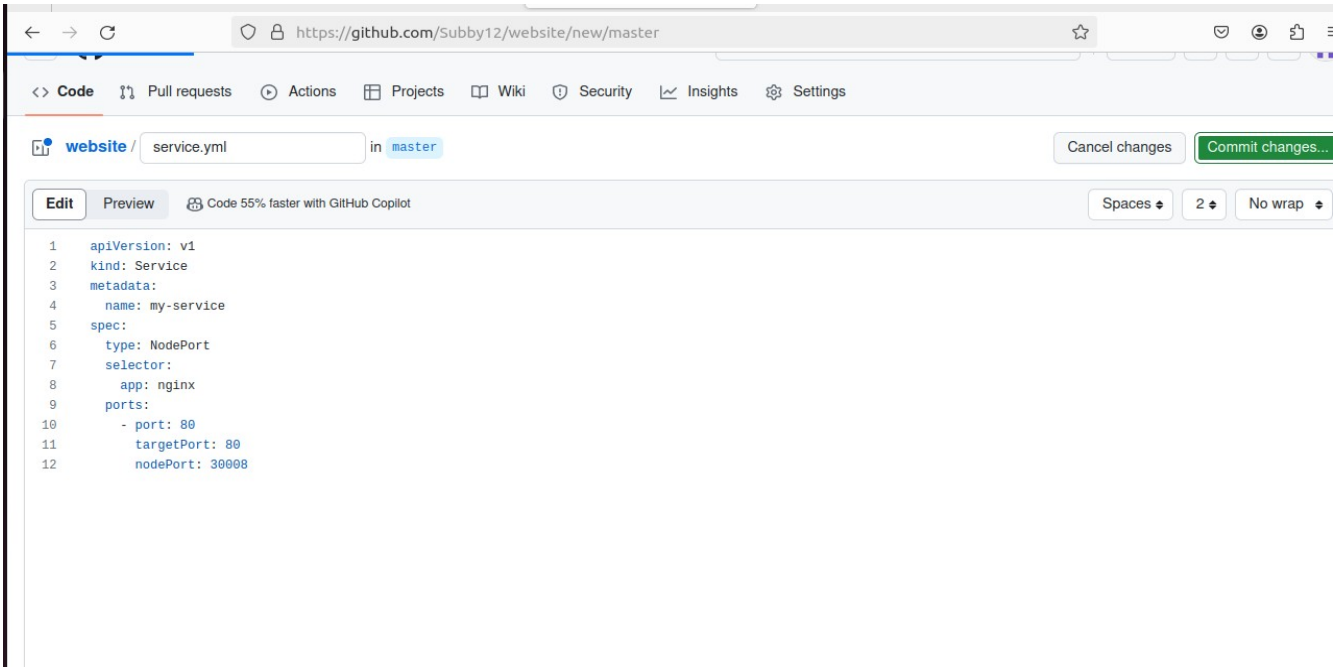
```
  matchLabels:
    app: nginx
template:
  metadata:
    labels:
      app: nginx
  spec:
    containers:
    - name: nginx
      image: subby83/project2:latest
      ports:
      - containerPort: 80
```



create service.yml file in git hub: (search service - node port and modify to taste)



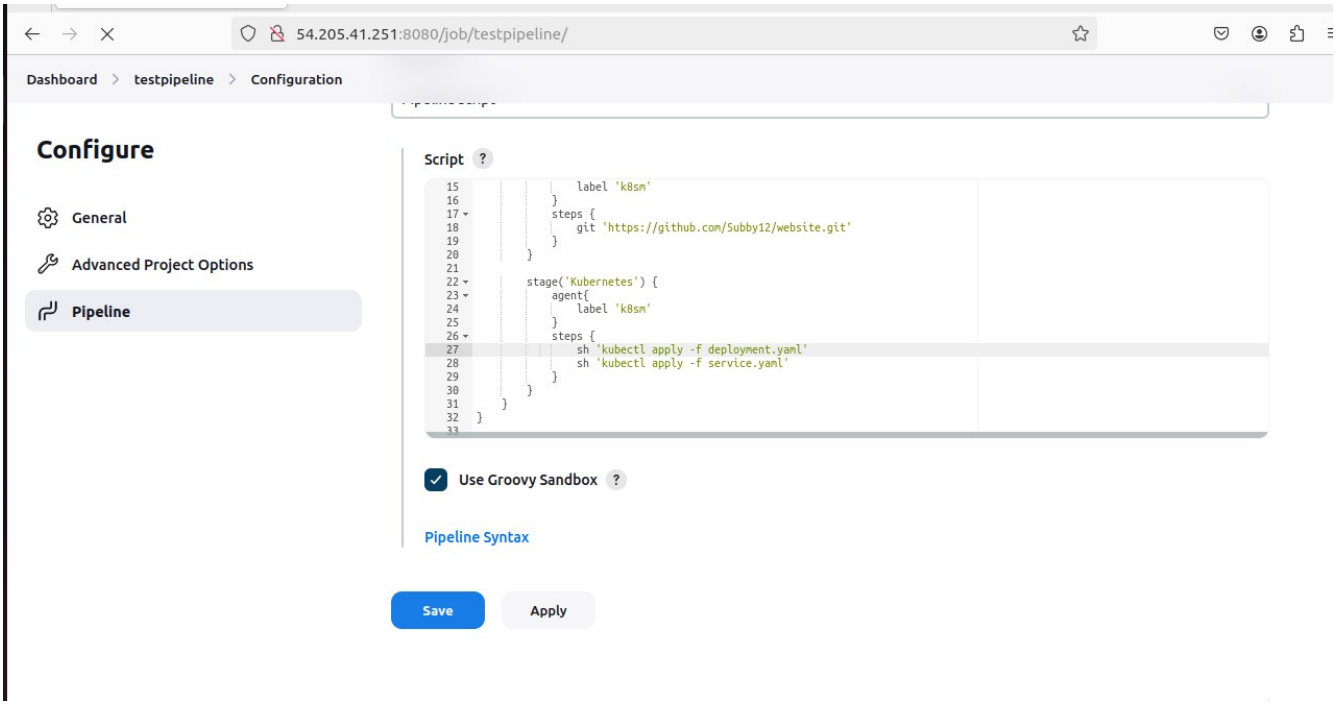

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  type: NodePort
  selector:
    app: nginx
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30008
```

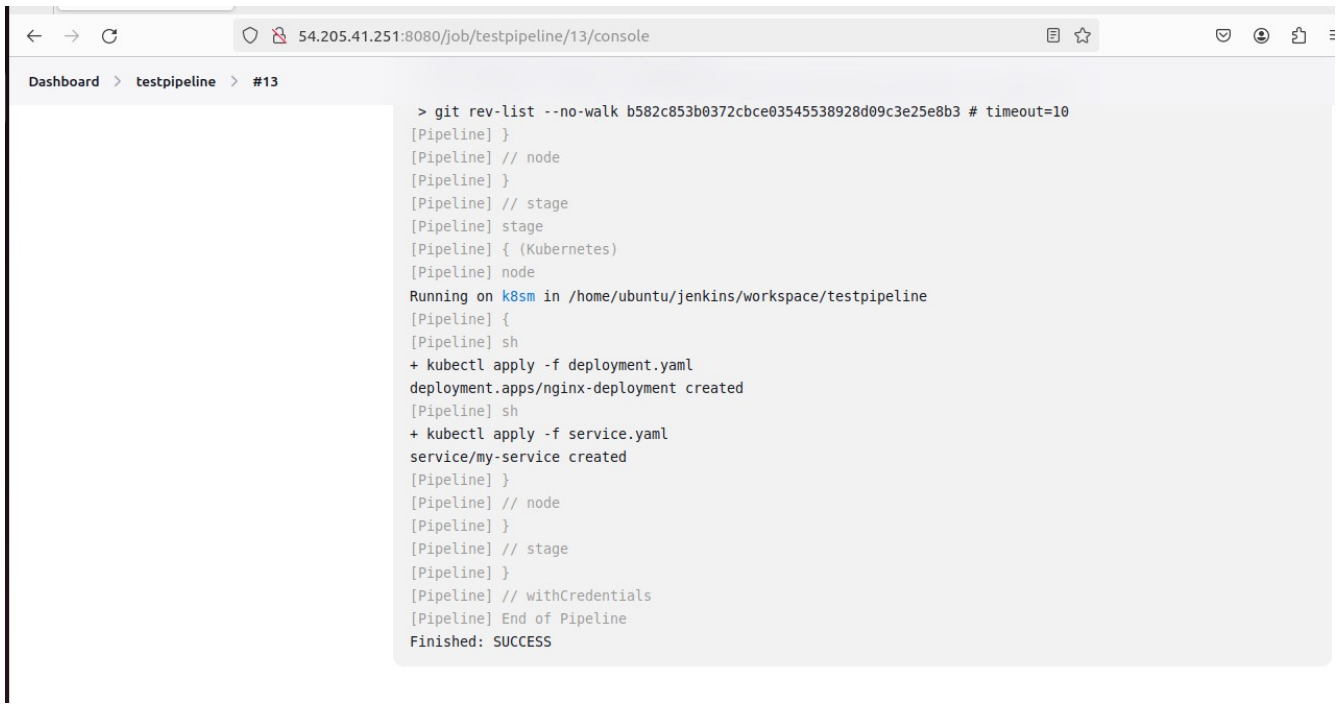


4TH STAGE

```
pipeline {
  agent none
  environment{
    DOCKERHUB_CREDENTIALS=credentials('0bd662f6-4796-4c30-9476-d19e1bfc0814')
  }

  stages {
    stage('Hello') {
      steps {
        echo 'Hello World'
      }
    }
    stage('Git') {
      agent{
        label 'k8sm'
      }
```

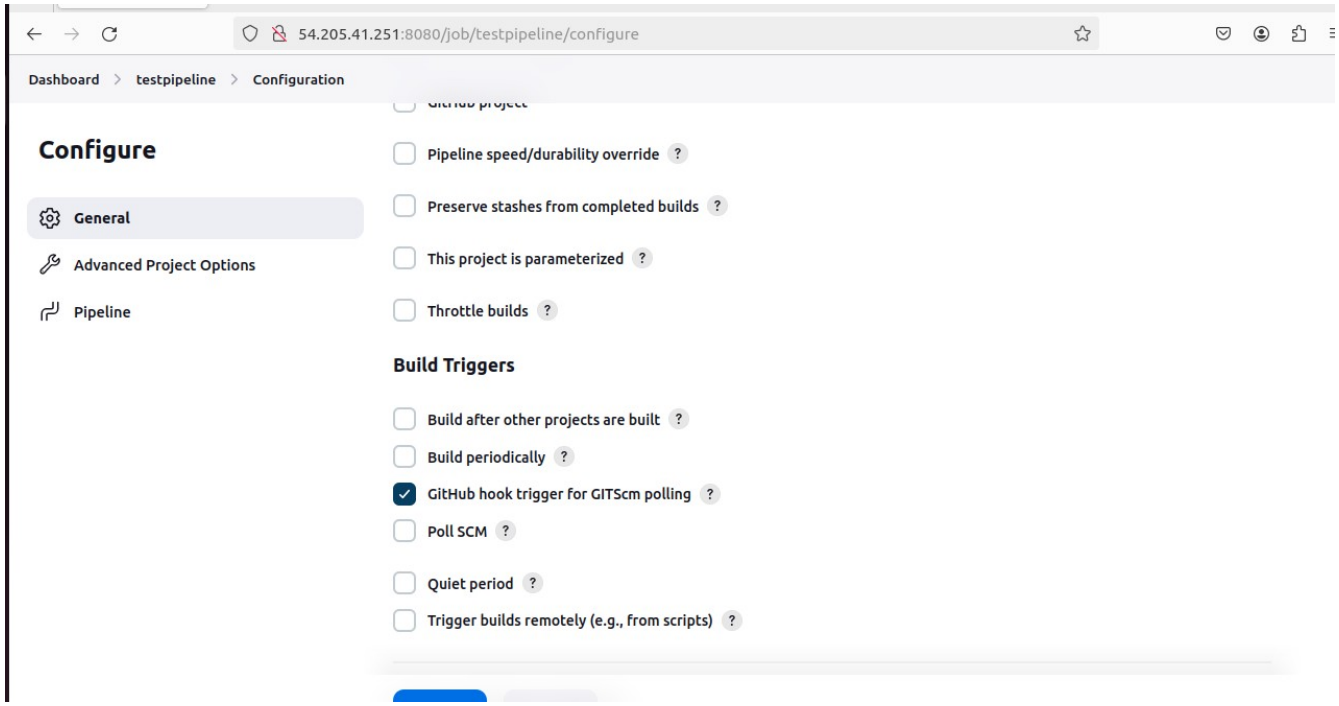
```
        steps {
            git 'https://github.com/Subby12/website.git'
        }
    }
    stage('Kubernetes') {
        agent{
            label 'k8sm'
        }
        steps {
            sh 'kubectl apply -f deployment.yaml'
            sh 'kubectl apply -f service.yaml'
        }
    }
  }
}
```





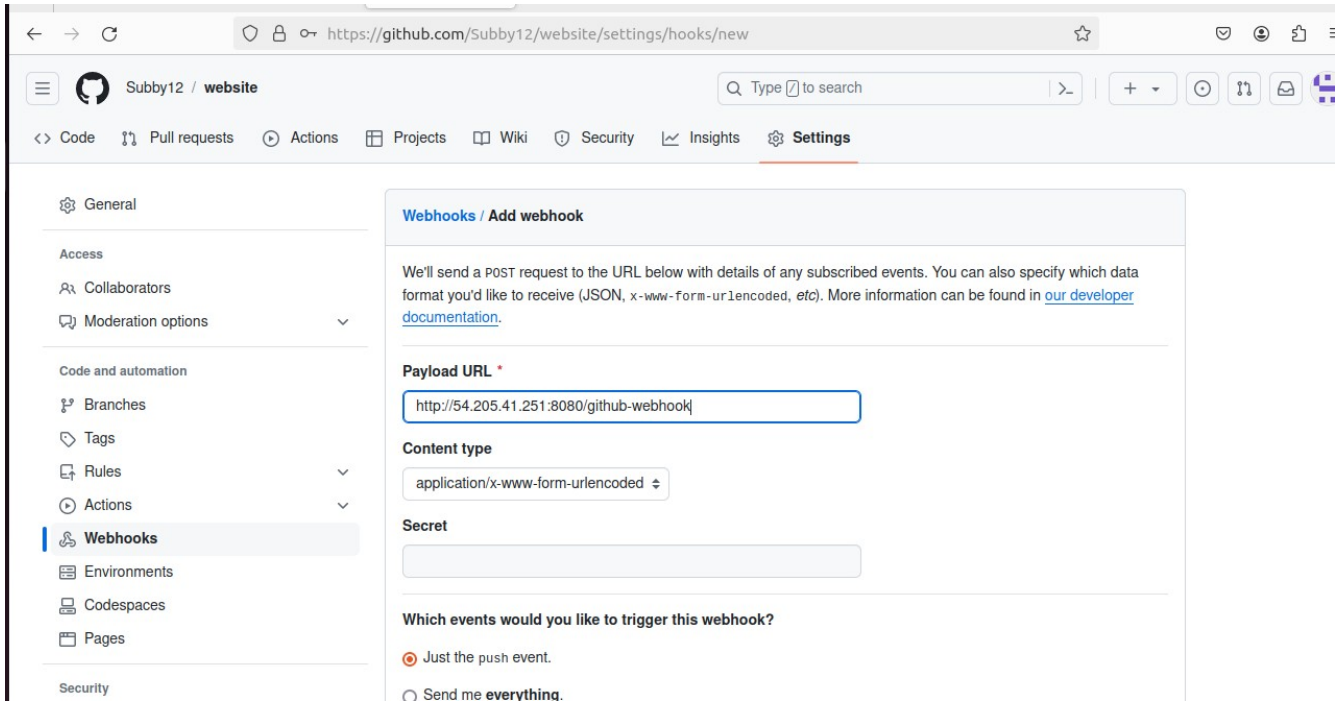
GITHUB HOOK

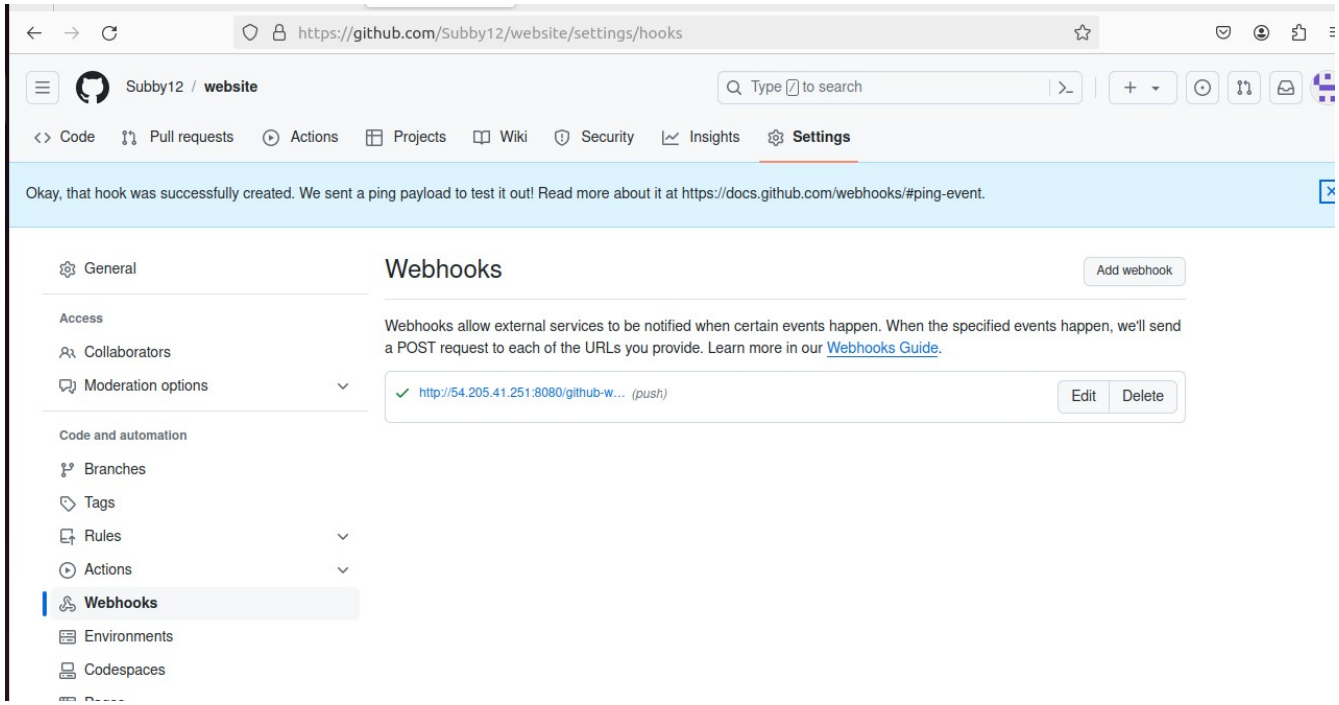
Dashboard – -job = testpipeline --- configuration –build trigger—github hook trigger fot GITSCM polling



ON GITHUB PAGE:

Settings: webhook: add webhook: payload url:: add webhook

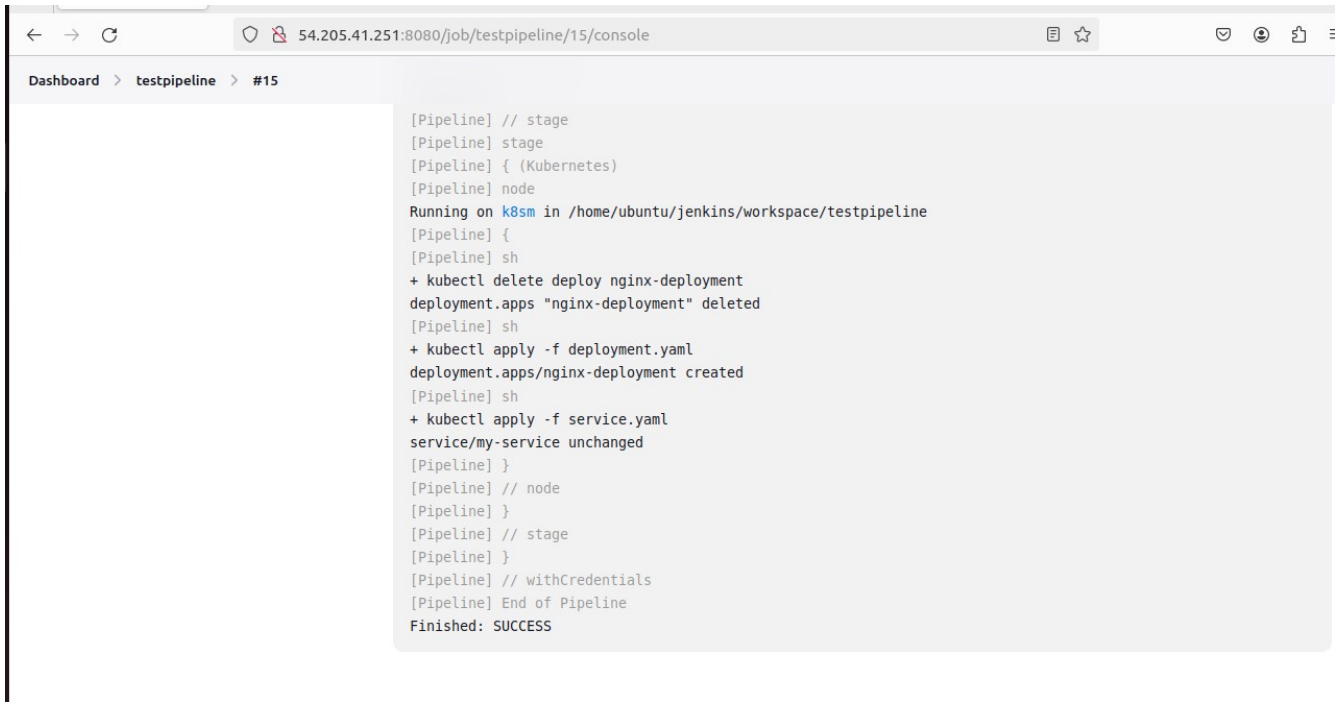




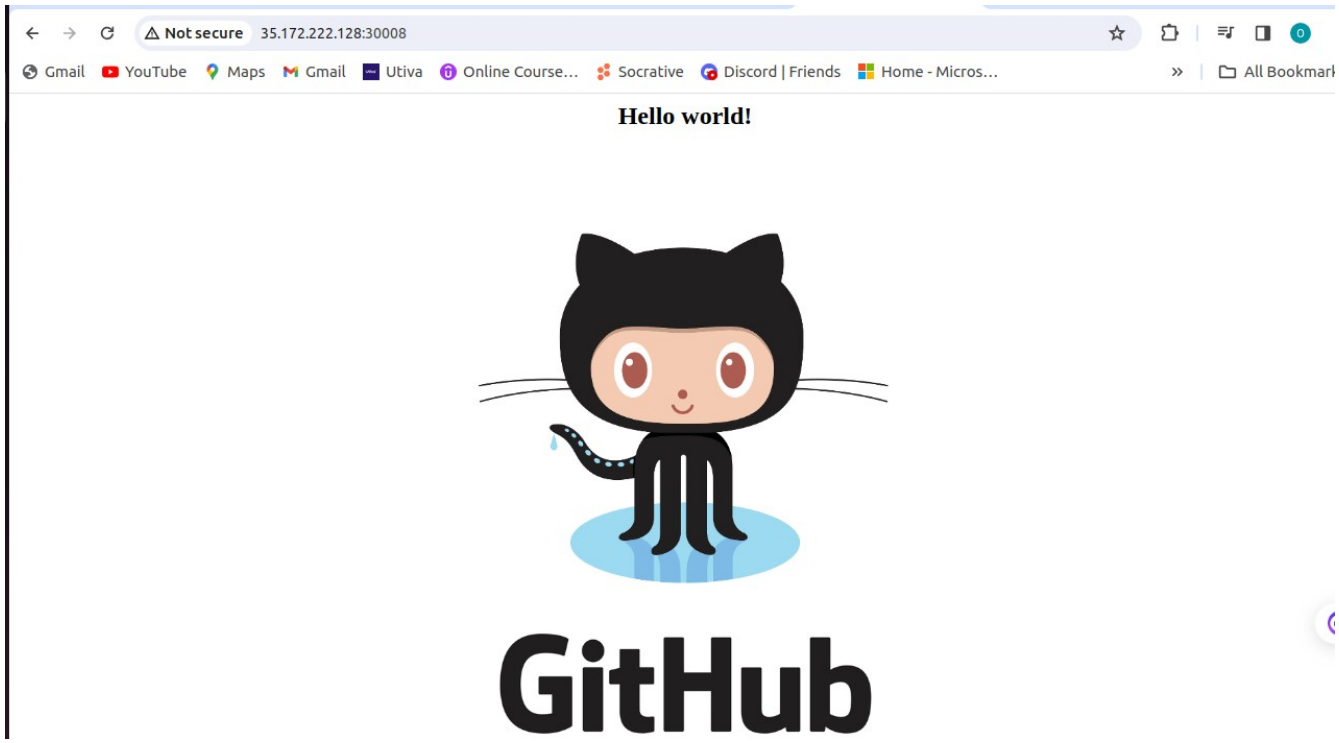
add to the code: to automate: this line: pipeline script

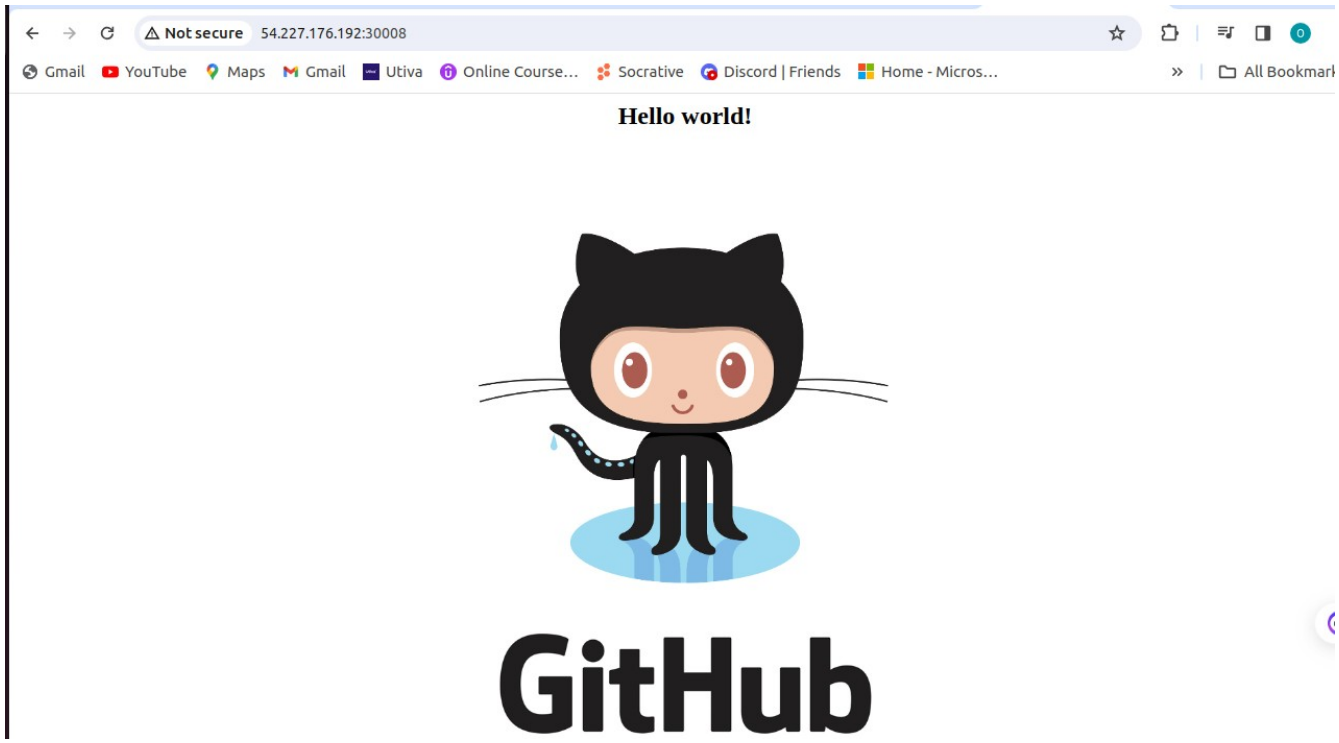
```
            sh 'kubectl delete deploy nginx-deployment'

pipeline {
    agent none
    environment{
        DOCKERHUB_CREDENTIALS=credentials('0bd662f6-4796-4c30-9476-d19e1bfc0814')
    }

    stages {
        stage('Hello') {
            steps {
                echo 'Hello World'
            }
        }
        stage('Git') {
            agent{
                label 'k8sm'
            }
            steps {
                git 'https://github.com/Subby12/website.git'
            }
        }
        stage('Docker') {
            agent{
                label 'k8sm'
            }
            steps {
                sh 'sudo docker build /home/ubuntu/jenkins/workspace/testpipeline -t subby83/project2'
                sh 'sudo echo $DOCKERHUB_CREDENTIALS_PSW | sudo docker login -u
$DOCKERHUB_CREDENTIALS_USR --password-stdin'
                sh 'sudo docker push subby83/project2'
            }
```

```
        }
        stage('Kubernetes') {
            agent{
                label 'k8sm'
            }
            steps {
                sh 'kubectl delete deploy nginx-deployment'
                sh 'kubectl apply -f deployment.yaml'
                sh 'kubectl apply -f service.yaml'
            }
        }
    }
}
```



IP address of the slaves:30008





Note:

label ‘k8sm’ =node name

git ‘url’