

## 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:~$
```



```
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"
}
```

```
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"
}
}
```

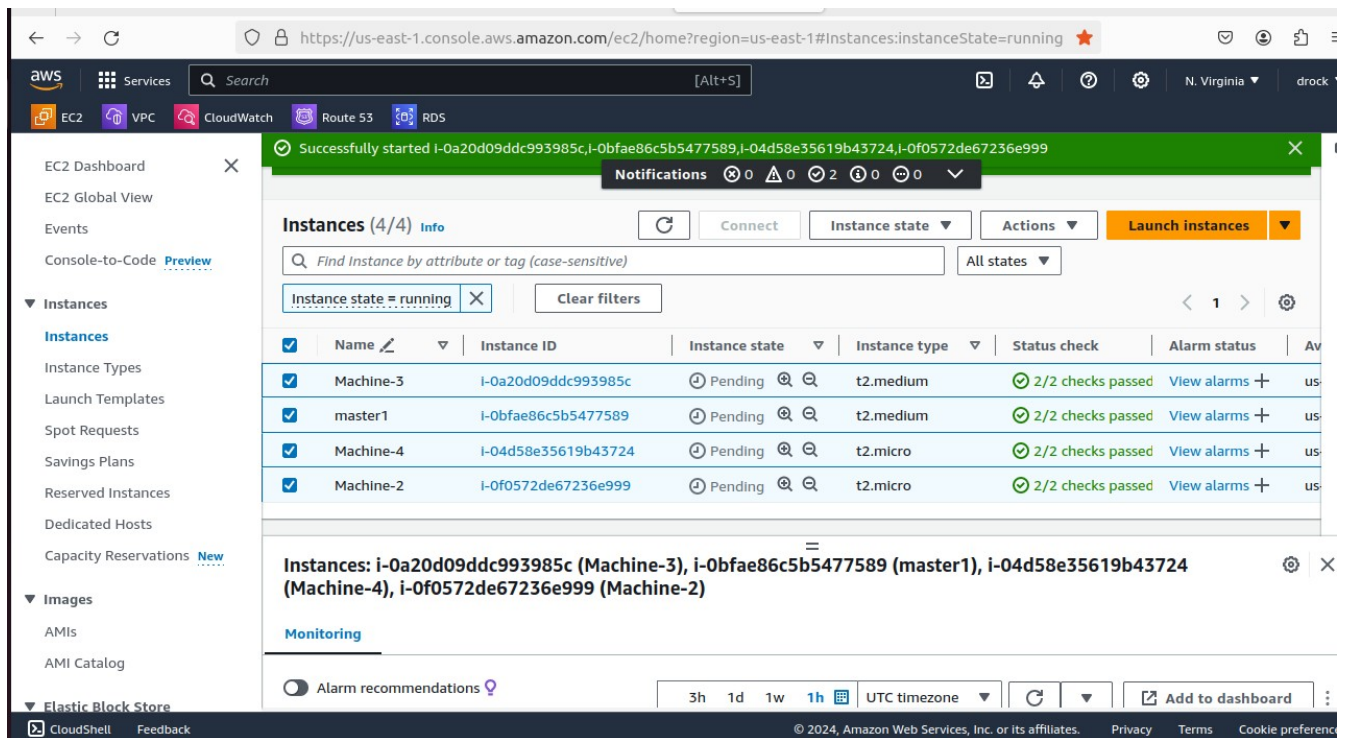
```
resource "aws_instance" "K8s-Slave1" {
  ami          = "ami-0cd59ecaf368e5ccf"
  instance_type = "t2.micro"
```

```
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
```

```

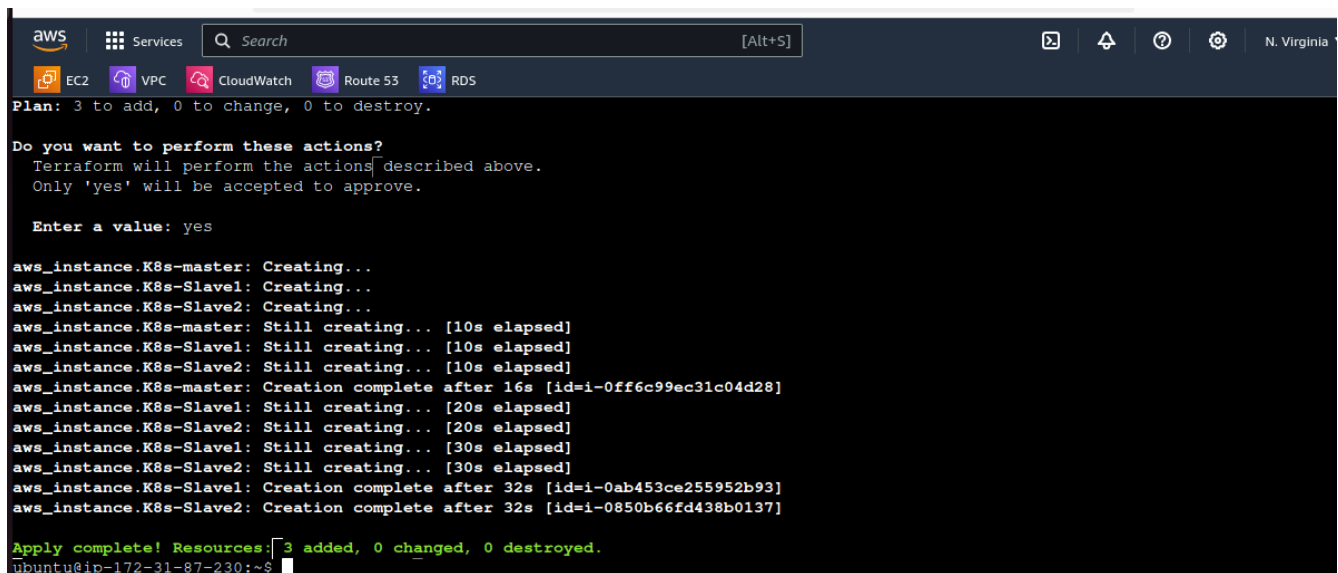
sudo apt update
sudo apt install -y gnupg software-properties-common curl
sudo mkdir -p /etc/apt/trusted.gpg.d
sudo curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o
/etc/apt/trusted.gpg.d/hashicorp.gpg
echo "deb [arch=amd64] https://apt.releases.hashicorp.com $(lsb_release -cs)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

```

```

terraform init
terraform plan
terraform apply

```



The screenshot shows a terminal window with the AWS CLI interface. The top bar displays 'aws' and 'Services' with a search bar. Below the bar, icons for EC2, VPC, CloudWatch, Route 53, and RDS are visible. The terminal output shows the Terraform plan and apply process. The plan indicates 3 resources to be added. The apply process prompts for confirmation, which is given as 'yes'. The output shows the creation of three AWS instances: 'aws\_instance.K8s-master', 'aws\_instance.K8s-Slave1', and 'aws\_instance.K8s-Slave2'. The master instance is created after 16s, and the slave instances are created after 32s. The final output shows 'Apply complete! Resources: 3 added, 0 changed, 0 destroyed.'

```

aws
Services Search [Alt+S]
EC2 VPC CloudWatch Route 53 RDS
Plan: 3 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_instance.K8s-master: Creating...
aws_instance.K8s-Slave1: Creating...
aws_instance.K8s-Slave2: Creating...
aws_instance.K8s-master: Still creating... [10s elapsed]
aws_instance.K8s-Slave1: Still creating... [10s elapsed]
aws_instance.K8s-Slave2: Still creating... [10s elapsed]
aws_instance.K8s-master: Creation complete after 16s [id=i-0ff6c99ec31c04d28]
aws_instance.K8s-Slave1: Still creating... [20s elapsed]
aws_instance.K8s-Slave2: Still creating... [20s elapsed]
aws_instance.K8s-Slave1: Still creating... [30s elapsed]
aws_instance.K8s-Slave2: Still creating... [30s elapsed]
aws_instance.K8s-Slave1: Creation complete after 32s [id=i-0ab453ce255952b93]
aws_instance.K8s-Slave2: Creation complete after 32s [id=i-0850b66fd438b0137]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
ubuntu@ip-172-31-87-230:~$

```

## 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.amzn2023.0.1.noarch

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 ~]$
```

## Setup the ssh

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

## Setup host

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-0d44bbbbaaa7669853 (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

Contact us: [support@intellipaat.com](mailto:support@intellipaat.com) / © Copyright Intellipaat / All rights reserved

DevOps Certification Training



**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
```

```
GNU nano 5.8 script1.sh
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
```

^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

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
```



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 -

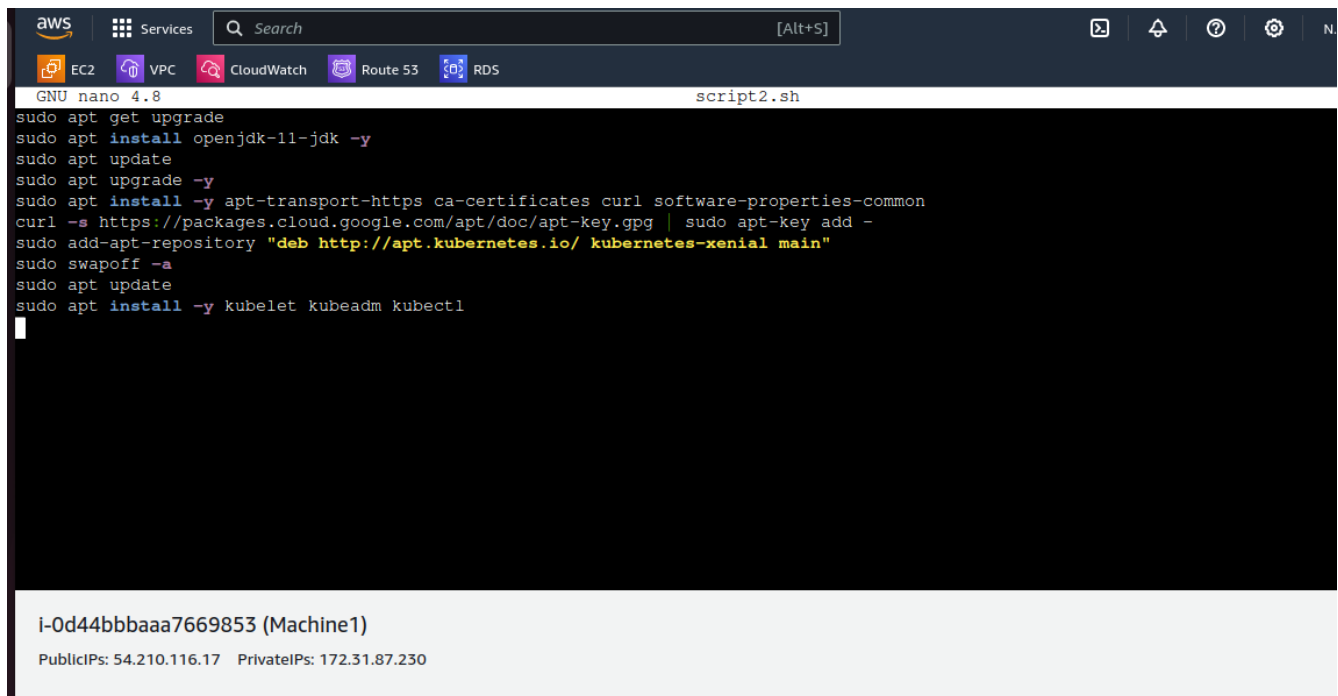
sudo add-apt-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"

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
```



```
aws | Services | Search [Alt+S]
EC2 VPC CloudWatch Route 53 RDS
GNU nano 4.8 script2.sh
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
```

i-0d44bbbbaa7669853 (Machine1)  
PublicIPs: 54.210.116.17 PrivateIPs: 172.31.87.230

### 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
```

```
aws Services Search [Alt+S]
EC2 VPC CloudWatch Route 53 RDS
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 ^C 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-Od44bbbbaaa7669853 (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-Od44bbbbaaa7669853 (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-Od44bbbbaaa7669853 (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-0d44bbbbaa7669853 (Machine1)
PublicIPs: 54.210.116.17  PrivateIPs: 172.31.87.230

```

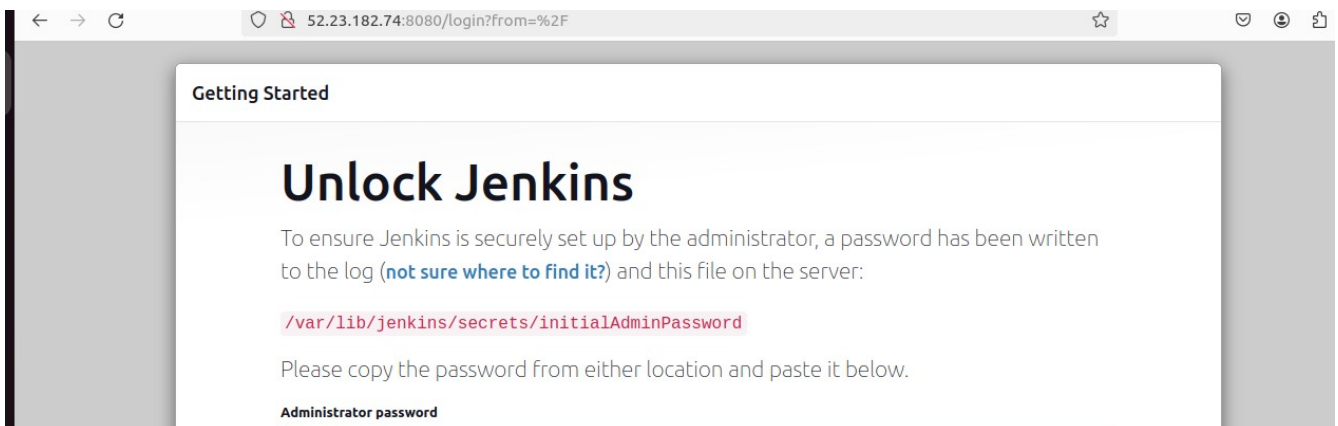
ansible-playbook play.yml --syntax-check  
 ansible-playbook play.yml -check

ansible-playbook play.yml

```
aws
Services
Search [Alt+S]
EC2 VPC CloudWatch Route 53 RDS
TASK [Gathering Facts] *****
ok: [172.31.91.38]
TASK [Running script2] *****
changed: [172.31.91.38]
PLAY [Install K8s on Machine3] *****
TASK [Gathering Facts] *****
ok: [172.31.80.106]
ok: [172.31.83.165]
TASK [Running script3] *****
changed: [172.31.83.165]
changed: [172.31.80.106]
PLAY RECAP *****
172.31.80.106 : ok=2 changed=1 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
172.31.83.165 : ok=2 changed=1 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
172.31.91.38 : ok=2 changed=1 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
localhost : ok=2 changed=1 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
ubuntu@ip-172-31-94-177:~$
I-Ocfd8aabf2c09ac2d (machine1)
PublicIPs: 52.23.182.74 PrivateIPs: 172.31.94.177
```

## ON LOCALHOST

```
ubuntu@ip-172-31-94-177:~$ java --version
openjdk 11.0.22 2024-01-16
OpenJDK Runtime Environment (build 11.0.22+7-post-Ubuntu-0ubuntu22.04.1)
OpenJDK 64-Bit Server VM (build 11.0.22+7-post-Ubuntu-0ubuntu22.04.1, mixed mode, sharing)
ubuntu@ip-172-31-94-177:~$
I-Ocfd8aabf2c09ac2d (machine1)
PublicIPs: 52.23.182.74 PrivateIPs: 172.31.94.177
```



```
← → ↻ https://us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-0a...
aws Services Search [Alt+S]
EC2 VPC CloudWatch Route 53 RDS
[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.ak4gwd9mm6nuyadm \
> --discovery-token-ca-cert-hash sha256:11c04936a95e0d817d729falle794b9c886b88f46af0fdb981df14945b56c76a
[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] 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 apiservert 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-node created
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
```

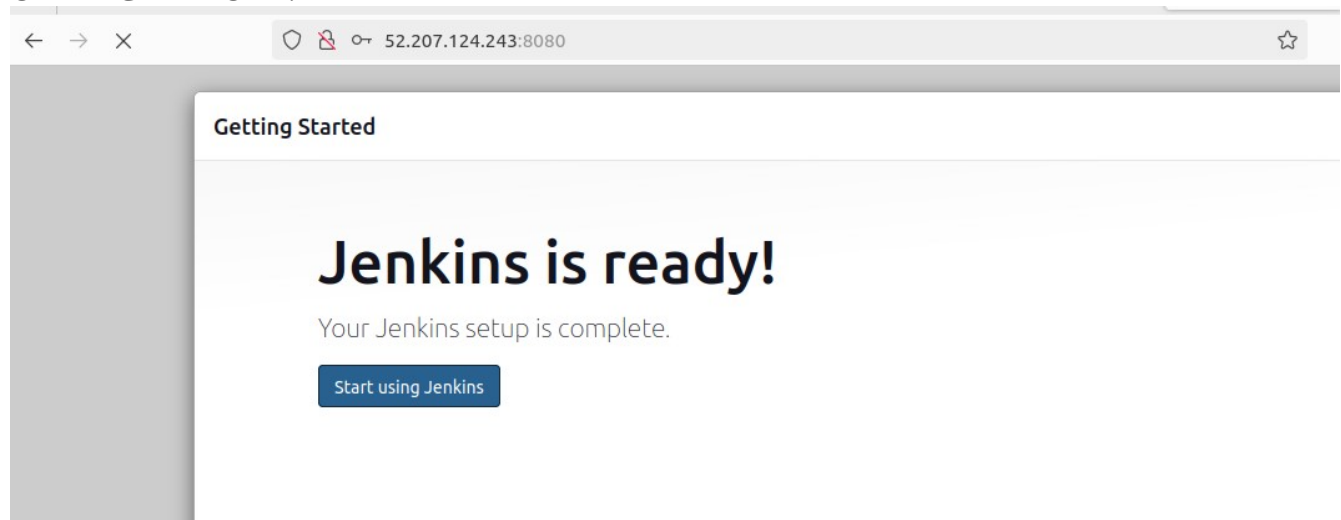
Installing Calico

```
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

F0y04acDSGmtC10tn29bCaytWSxvSKyMGfTEMHGNmRgdZK+Ug+wtD0nFXVZt0Y1CSQYYWW  
4TQKZyl0oEpT3FTdn2f76pZFV0j3GVfVwWG5YtFrkcTbIdkz0zV0ulrKF+8MvD0URK6rt0  
Xw2jt9LQ7AxG+tAAAF3VidW50dUBpcC0xNzItMzEt0DATMTUzAQID  
-----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...

Jenkins

Search (CTRL+K)

subby log out

Dashboard > Manage Jenkins > Nodes >

Nodes

Build Queue

Build Executor Status

Nodes

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

← → ↻ Not secure 3.86.158.196:8080/manage/credentials/store/system/domain/\_/newCredentials ☆ 📁 📄 0 ⋮

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

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

### New credentials

Kind

Username with password

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Username ?

subby83

☐ Treat username as secret ?

Password ?

\*\*\*\*\*

ID ?

0bd662f6-4796-4c30-9476-d19e1bfc0814 subby83/\*\*\*\*\* Username with password

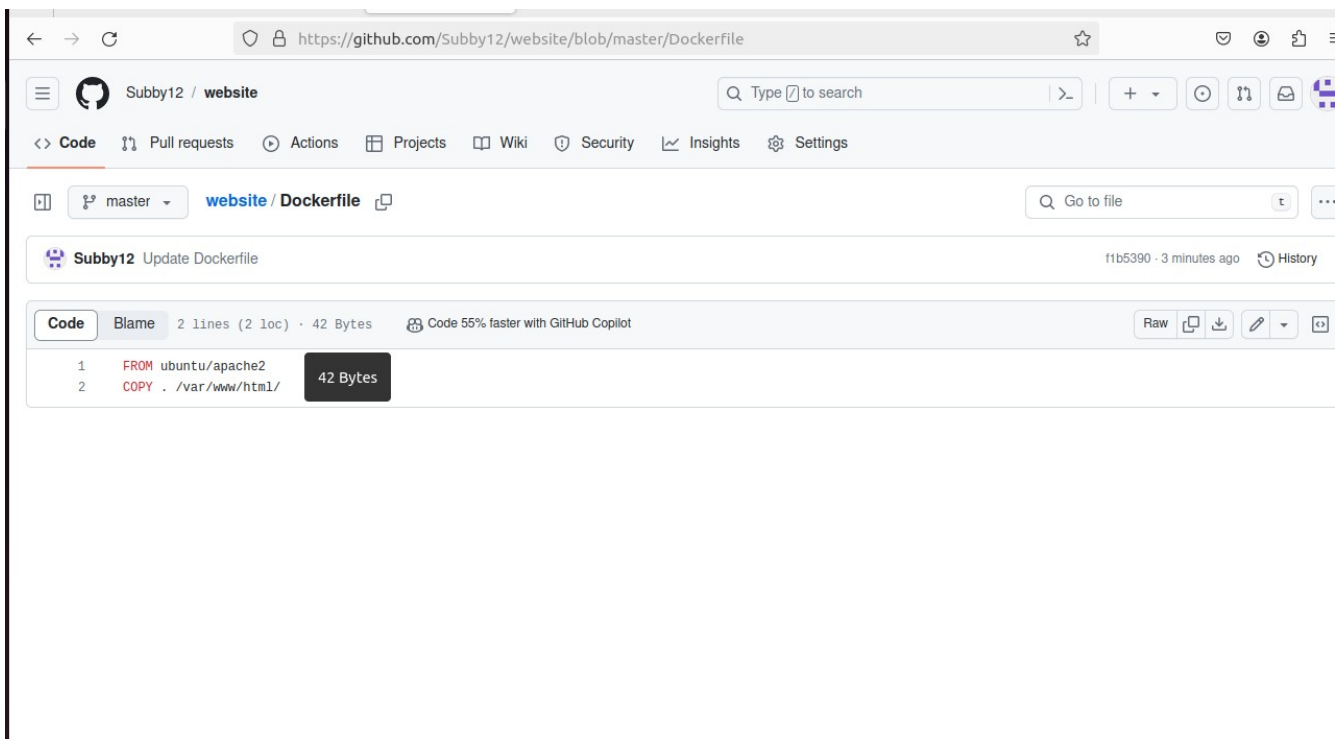
n: S M L

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: TESTINGI HELLO WORLD

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

```
}
```

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

```
}
```

← → ×

54.205.41.251:8080/job/testpipeline/

☆

🔒

👤

📄

Dashboard > testpipeline > Configuration

Configure

⚙️ General

🔧 Advanced Project Options

📄 Pipeline

Pipeline

Definition

Pipeline script

Script ?

1 pipeline {  
2 agent none  
3 environment{  
4 DOCKERHUB\_CREDENTIALS=credentials('0b0bd662f6-4796-4c30-9476-d19e1bfc0814')  
5 }  
6  
7 stages {  
8 stage('Hello') {  
9 steps {  
10 echo 'Hello World'  
11 }  
12 }  
13 }  
14 }  
15 }

Hello World

☒ Use Groovy Sandbox ?

[Pipeline Syntax](#)

Save

Apply

Transferring data from 54.205.41.251...

Jenkins

Search (CTRL+K)

1

1

subby ▾

log

Dashboard > testpipeline > #2

Status

Changes

Console Output

View as plain text

Edit Build Information

Delete build '#2'

Timings

Pipeline Overview

Pipeline Console

Restart from Stage

Replay

✓ Console Output

Started by user [subby](#)

[Pipeline] Start of Pipeline

[Pipeline] withCredentials

Masking supported pattern matches of \$DOCKERHUB\_CREDENTIALS or \$DOCKERHUB\_CREDENTIALS\_PSW

[Pipeline] {

[Pipeline] stage

[Pipeline] { (Hello)

[Pipeline] echo

Hello World

[Pipeline] }

[Pipeline] // stage

[Pipeline] }

[Pipeline] // withCredentials

[Pipeline] End of Pipeline

Finished: SUCCESS

54.205.41.251:8080/job/testpipeline/2/console

## 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'
    }
}
}
}

```

Dashboard > testpipeline > #3

```

[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Git)
[Pipeline] node
Running on k8sm in /home/ubuntu/jenkins/workspace/testpipeline
[Pipeline] {
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Avoid second fetch
Checking out Revision 053025741e9d415a463f1e22a9341d08ff5055fb (refs/remotes/origin/master)
Cloning repository https://github.com/Subby12/website.git
> git init /home/ubuntu/jenkins/workspace/testpipeline # timeout=10
Fetching upstream changes from https://github.com/Subby12/website.git
> git --version # timeout=10
> git --version # 'git version 2.25.1'
> git fetch --tags --force --progress -- https://github.com/Subby12/website.git +refs/heads/*:refs/remotes,
origin/* # timeout=10
> git config remote.origin.url https://github.com/Subby12/website.git # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> git config core.sparsecheckout # timeout=10
> git checkout -f 053025741e9d415a463f1e22a9341d08ff5055fb # timeout=10
> git branch -a -v --no-abbrev # timeout=10
> git checkout -b master 053025741e9d415a463f1e22a9341d08ff5055fb # timeout=10
Commit message: "Update Dockerfile"
First time build. Skipping changelog.

```

## 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'
    }
  }
}
```

```
Dashboard > testpipeline > #7


[Pipeline] sh
+ sudo docker build /home/ubuntu/jenkins/workspace/testpipeline -t subby83/project2
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon 284.7kB

Step 1/2 : FROM ubuntu
--> ca2b0f26964c
Step 2/2 : COPY . /var/www/html/
--> 24e23781a73c
Successfully built 24e23781a73c
Successfully tagged subby83/project2:latest
[Pipeline] sh
+ sudo echo ****
+ sudo docker login -u subby83 --password-stdin
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[Pipeline] sh
+ sudo docker push subby83/project2
Using default tag: latest
The push refers to repository [docker.io/subby83/project2]
d6e645338e27: Preparing
5498e8c22f69: Preparing
```

← → ↻ https://hub.docker.com

 **dockerhub** Explore Repositories Organizations  ctrl+K ?

subby83 ▼

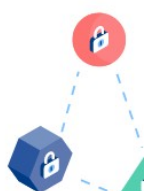
Search by repository name

All Content ▼

Create repository

**subby83 / project2**  
Contains: Image • Last pushed: 20 minutes ago  
Security unknown ☆ 0 ↓ 1 Public

**subby83 / lovely**  
Contains: Image • Last pushed: 9 days ago  
Security unknown ☆ 0 ↓ 5 Public



## 4TH STAGE

search 'deployment k8s'





deployment k8s



All

Images

Videos

Shopping

News

More

Tools

Example

Strategies

Restart

Tools

File

Zero-downtime

Kafka

Terraform

About 81,100,000 results (0.47 seconds)



Kubernetes

<https://kubernetes.io/docs/controllers/deployment>

## Deployments

A **Deployment** manages a set of Pods to run an application workload, usually one that doesn't maintain state.

[ReplicaSet](#) · [Managing Workloads](#) · [Feedback](#)

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:

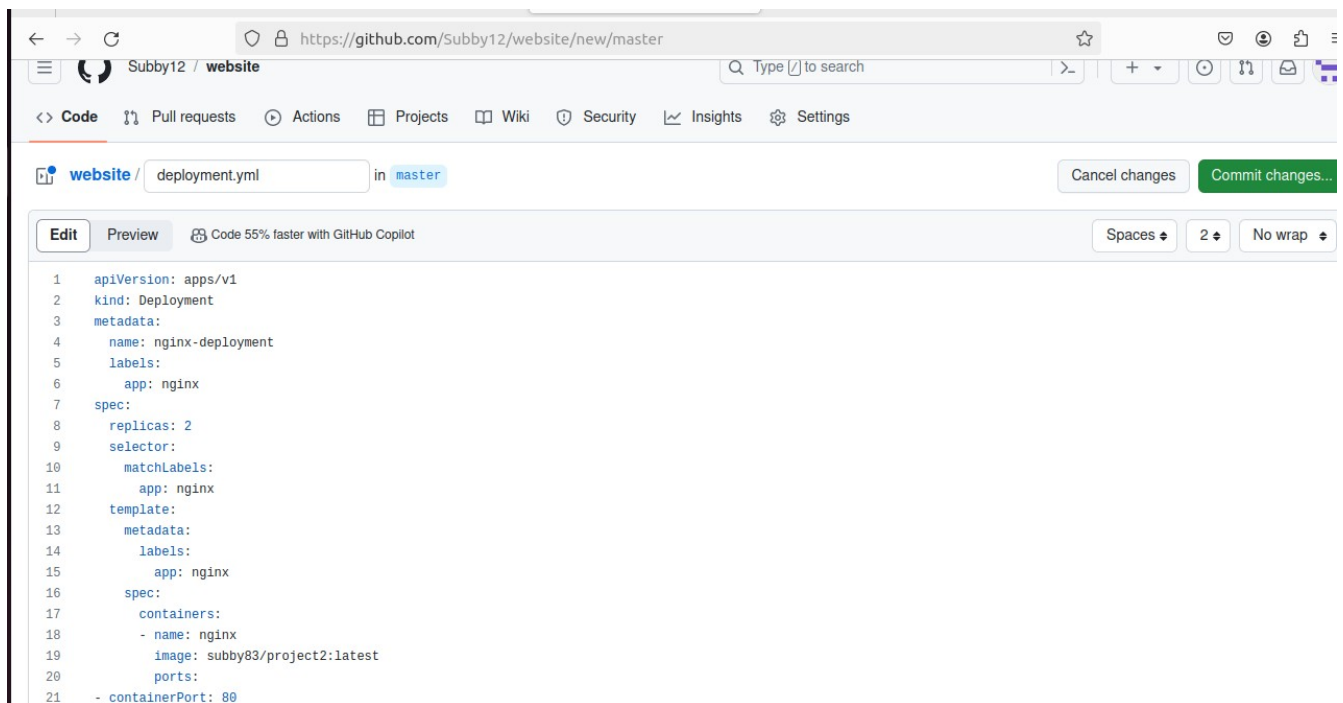
containers:

- name: nginx

image: subby83/project2:latest

ports:

- containerPort: 80



The screenshot shows a web browser window displaying a GitHub repository editor. The address bar shows the URL `https://github.com/Subby12/website/new/master`. The repository name is `Subby12 / website`. The file being edited is `deployment.yml` in the `master` branch. The editor has tabs for `Edit` and `Preview`, with a note that `Code 55% faster with GitHub Copilot`. The code is a Kubernetes Deployment manifest for an nginx application. The manifest includes metadata with labels and a spec with two replicas, a selector matching the labels, and a container named `nginx` using the `subby83/project2:latest` image and exposing port 80.

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: nginx-deployment
5    labels:
6      app: nginx
7  spec:
8    replicas: 2
9    selector:
10     matchLabels:
11       app: nginx
12   template:
13     metadata:
14       labels:
15         app: nginx
16     spec:
17       containers:
18         - name: nginx
19           image: subby83/project2:latest
20           ports:
21             - containerPort: 80
```

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

The screenshot shows the Kubernetes documentation page for Services. The browser address bar displays `kubernetes.io/docs/concepts/services-networking/service/`. The page header includes the Kubernetes logo and navigation links: Documentation, Kubernetes Blog, Training, Partners, Community, Case Studies, Versions, and English. A search bar is located on the left side of the page. The sidebar on the left contains a list of navigation links: Documentation, Getting started, Concepts, Overview, Cluster Architecture, Containers, Workloads, Services, Load Balancing, and Networking. The main content area features an example manifest for a Service of type NodePort. The manifest is as follows:

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  type: NodePort
  selector:
    app.kubernetes.io/name: MyApp
  ports:
    - port: 80
      # By default and for convenience, the `targetPort` is set to
      # the same value as the `port` field.
      targetPort: 80
      # Optional field
      # By default and for convenience, the Kubernetes control plane
      # will allocate a port from a range (default: 30000-32767)
      nodePort: 30007
```

On the right side of the page, there is a list of related topics: Services in Kubernetes, Cloud-native service discovery, Defining a Service, Port definitions, Services without selectors, EndpointSlices, Endpoints, Application protocol, Multi-port Services, Service type, type: ClusterIP, type: NodePort, type: LoadBalancer, type: ExternalName, and Headless Services.

apiVersion: v1

kind: Service

metadata:

name: my-service

spec:

type: NodePort

selector:

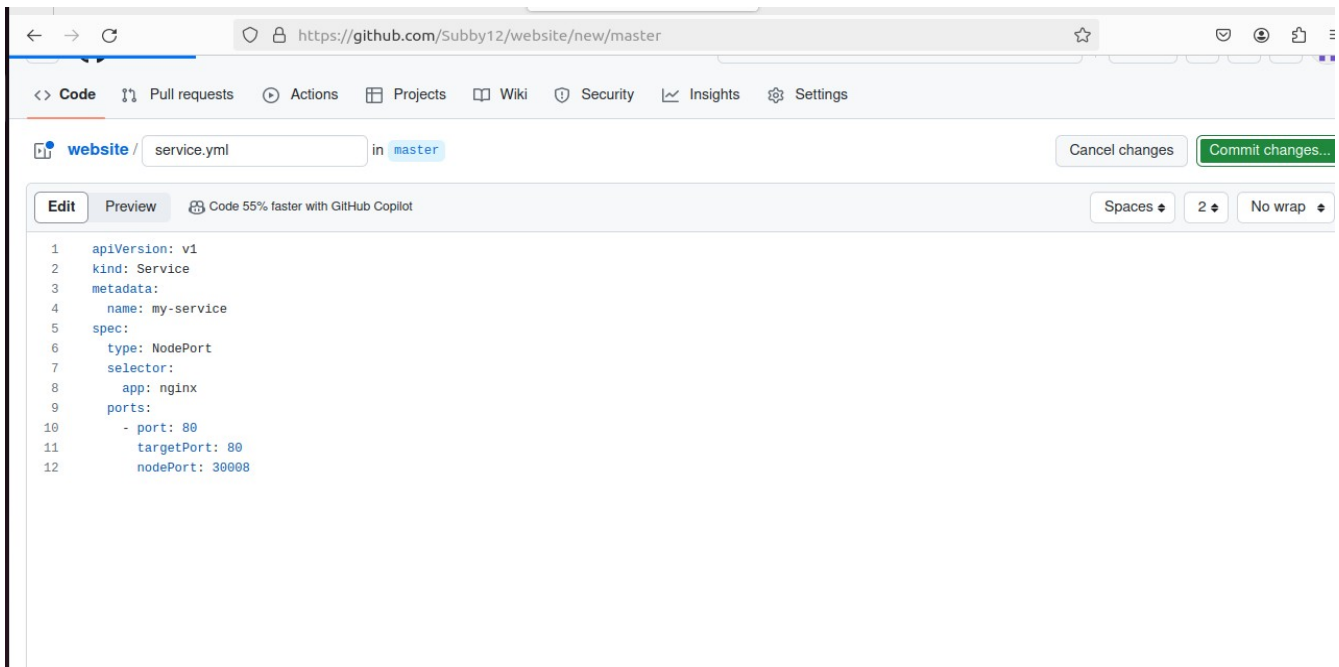
app: nginx

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'
    }
}
}
}

```

54.205.41.251:8080/job/testpipeline/

Dashboard > testpipeline > Configuration

## Configure

- General
- Advanced Project Options
- Pipeline**

**Script** ?

```

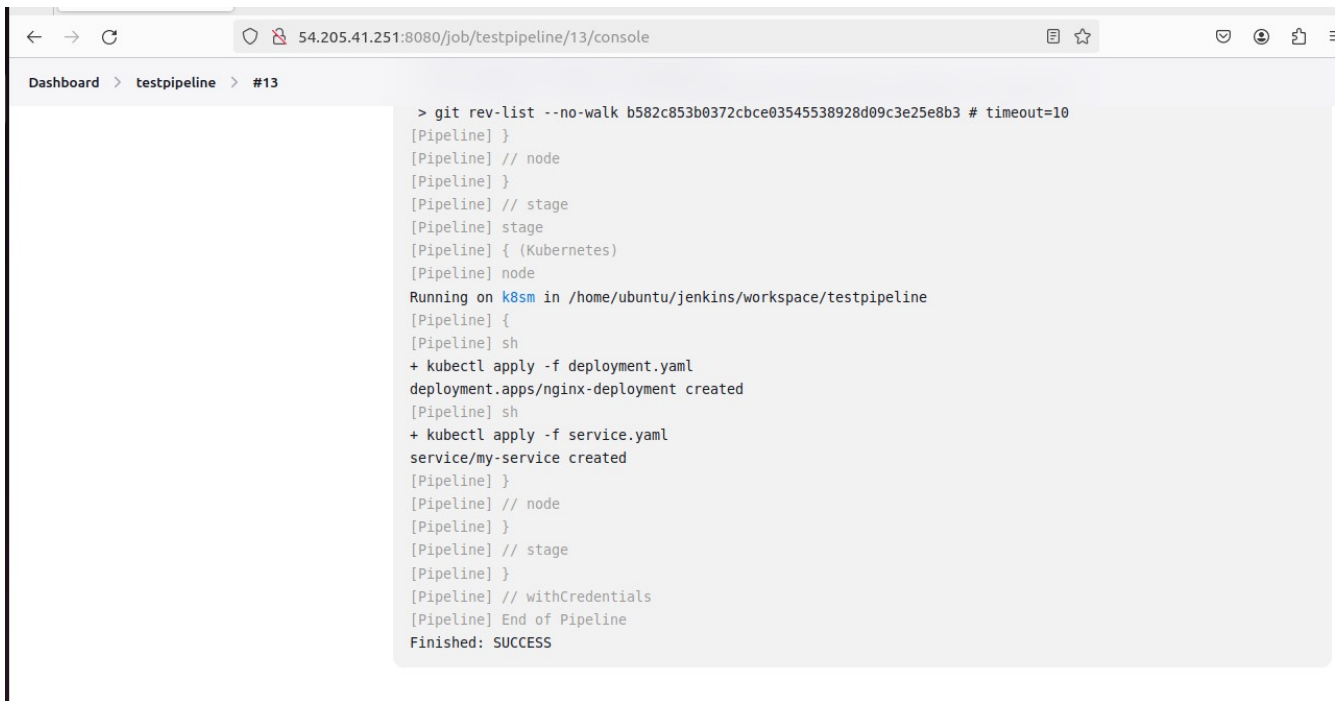
15     } label 'k8sm'
16     }
17     steps {
18     git 'https://github.com/Subby12/website.git'
19     }
20     }
21
22     stage('Kubernetes') {
23     agent{
24     label 'k8sm'
25     }
26     steps {
27     sh 'kubectl apply -f deployment.yaml'
28     sh 'kubectl apply -f service.yaml'
29     }
30     }
31 }
32 }
33 }

```

☒ Use Groovy Sandbox ?

[Pipeline Syntax](#)

**Save** **Apply**

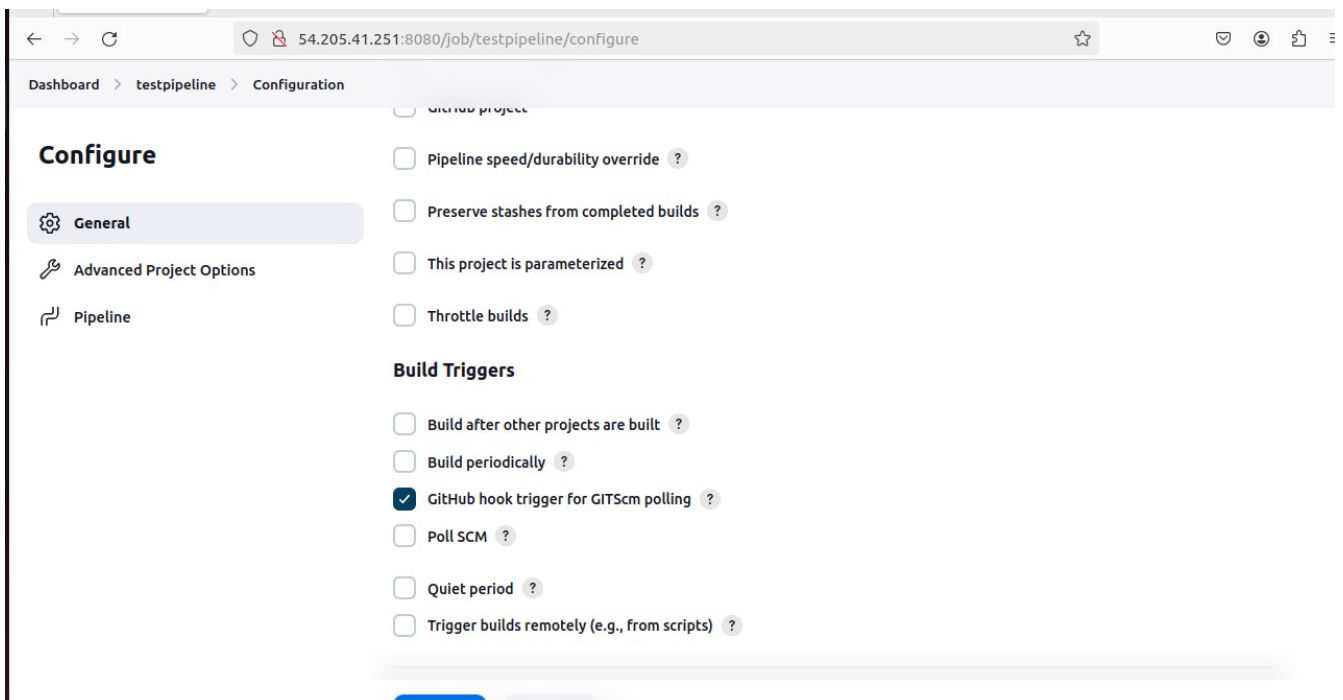


Dashboard > testpipeline > #13

```
> git rev-list --no-walk b582c853b0372cbce03545538928d09c3e25e8b3 # timeout=10
[Pipeline] }
[Pipeline] // node
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Kubernetes)
[Pipeline] node
Running on k8sm in /home/ubuntu/jenkins/workspace/testpipeline
[Pipeline] {
[Pipeline] sh
+ kubectl apply -f deployment.yaml
deployment.apps/nginx-deployment created
[Pipeline] sh
+ kubectl apply -f service.yaml
service/my-service created
[Pipeline] }
[Pipeline] // node
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withCredentials
[Pipeline] End of Pipeline
Finished: SUCCESS
```

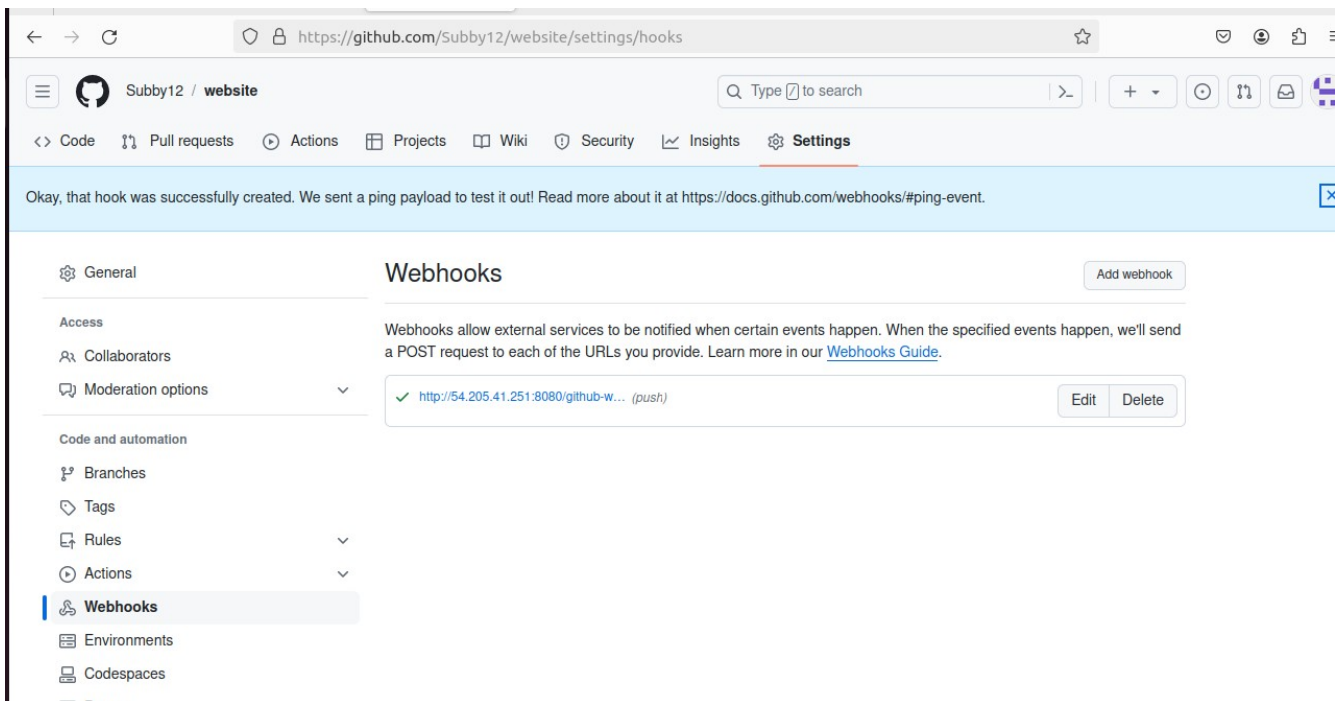
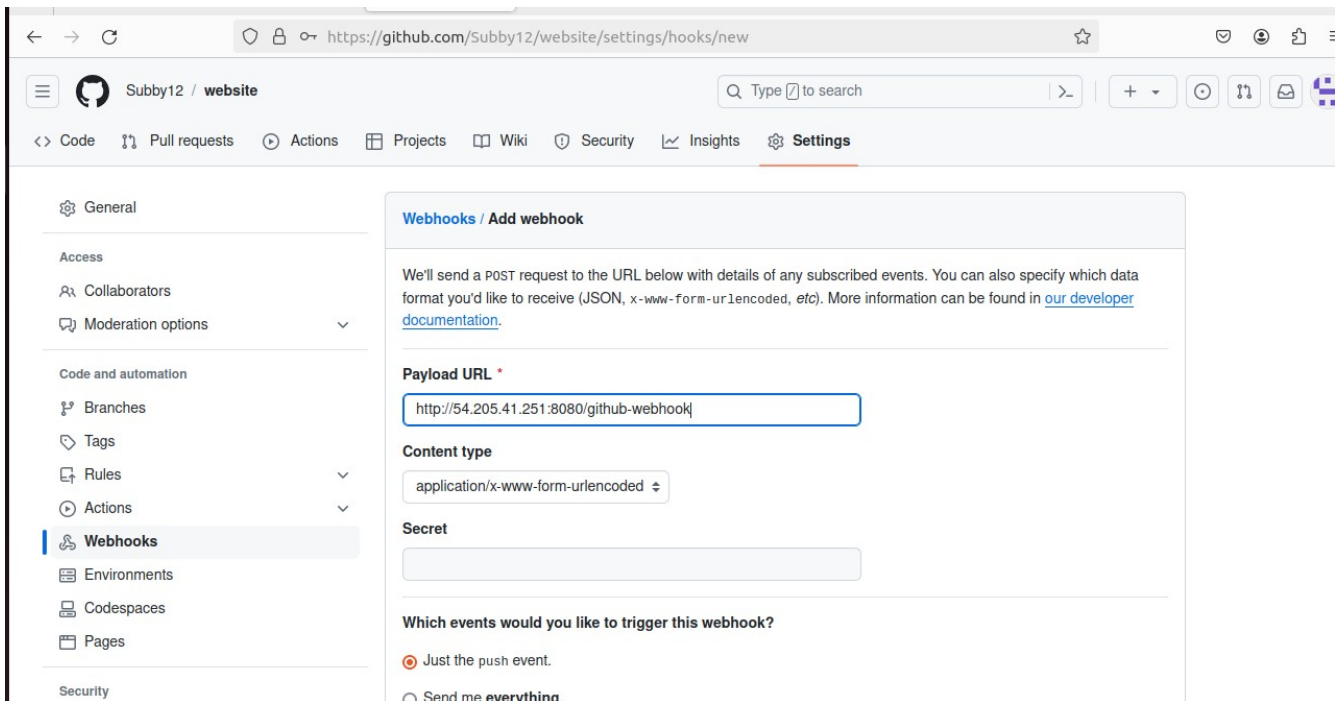
## GITHUB HOOK

Dashboard – -job = testpipeline --- configuration –build trigger—github hook trigger for 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'
    }
}
}
}
}

```

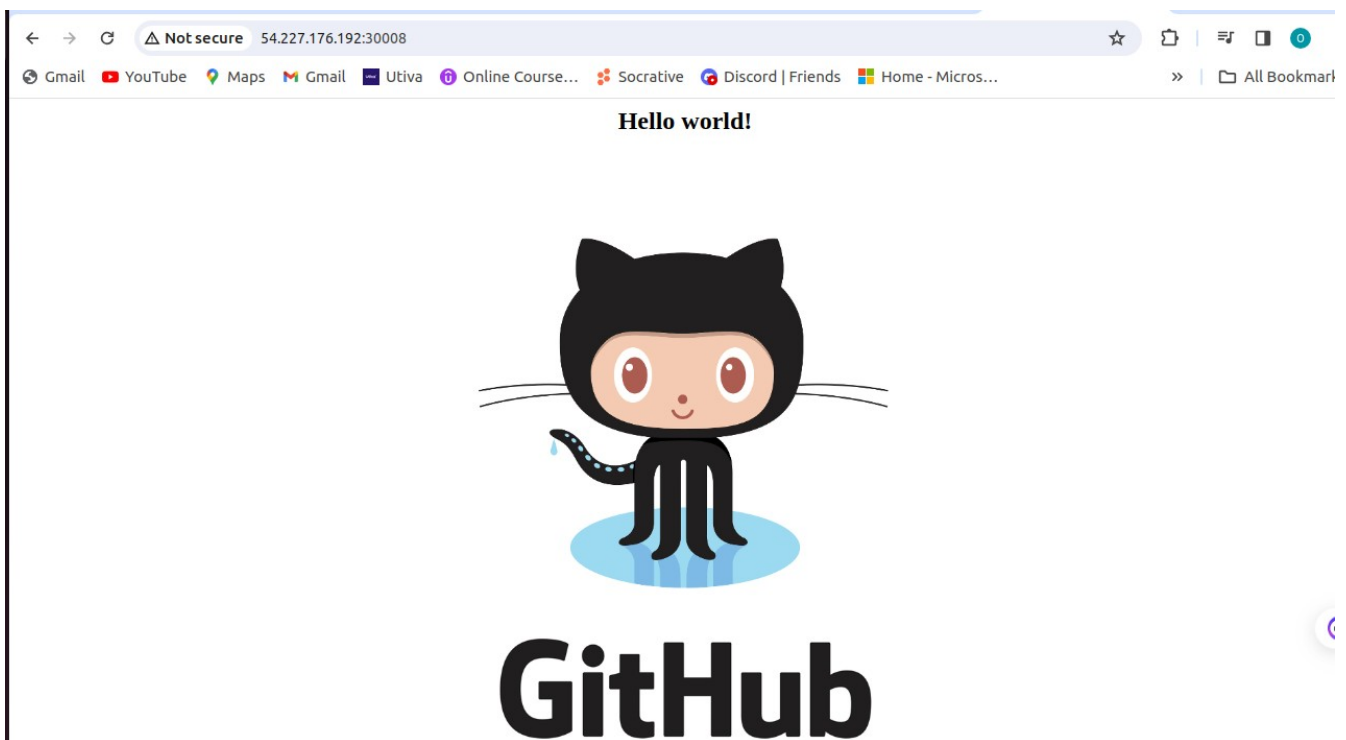
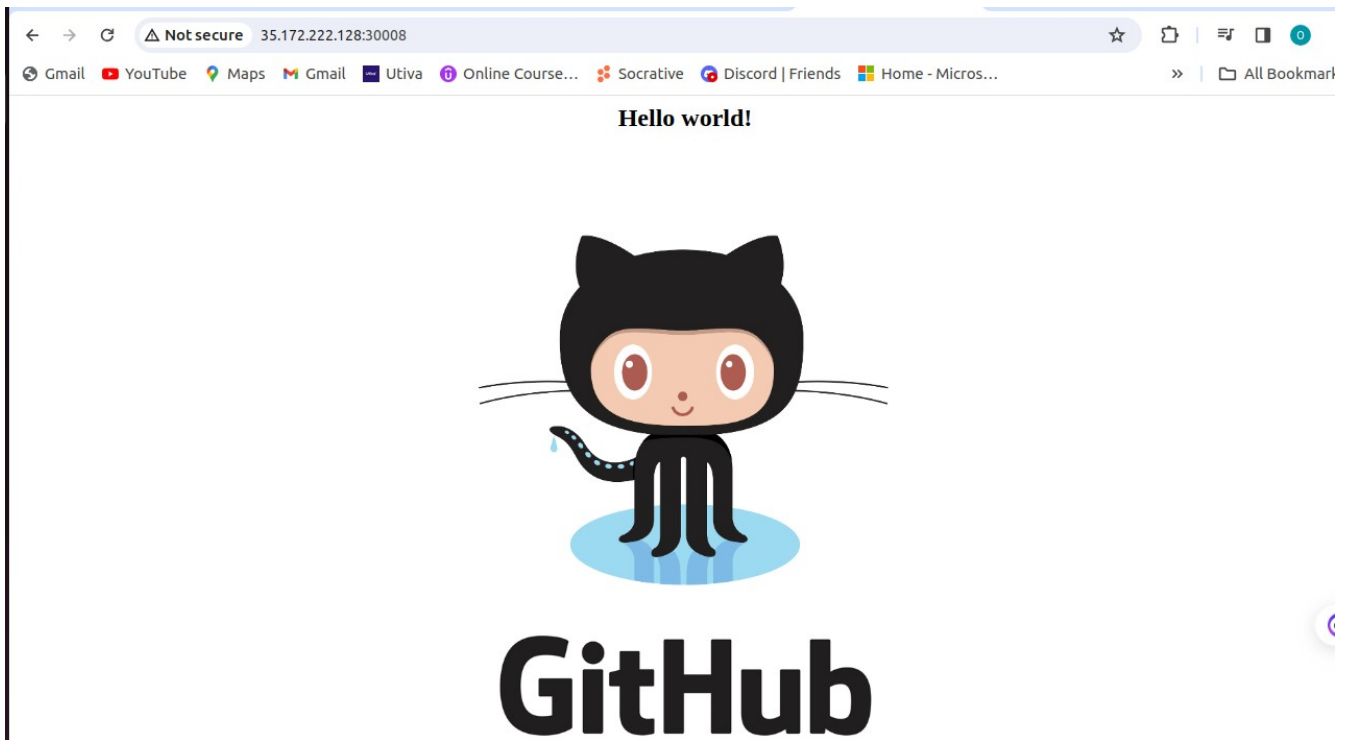
The screenshot shows a web browser window displaying the Jenkins console output for a pipeline named 'testpipeline' at job #15. The browser's address bar shows the URL '54.205.41.251:8080/job/testpipeline/15/console'. The console output is as follows:

```

[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Kubernetes)
[Pipeline] node
Running on k8sm in /home/ubuntu/jenkins/workspace/testpipeline
[Pipeline] {
[Pipeline] sh
+ kubectl delete deploy nginx-deployment
deployment.apps "nginx-deployment" deleted
[Pipeline] sh
+ kubectl apply -f deployment.yaml
deployment.apps/nginx-deployment created
[Pipeline] sh
+ kubectl apply -f service.yaml
service/my-service unchanged
[Pipeline] }
[Pipeline] // node
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withCredentials
[Pipeline] End of Pipeline
Finished: SUCCESS

```

IP address of the slaves:30008



Note:

label 'k8sm' =node name

git 'url'