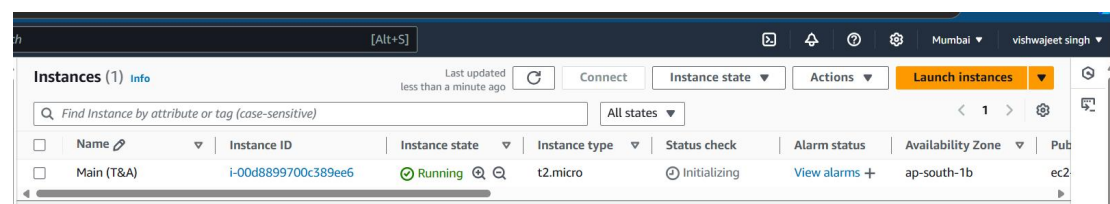


---you are hired as a DevOps engineer for Analytics Pvt Ltd. This company is a product based organization which uses Docker for their containerization needs within the company. The final product received a lot of traction in the first few weeks of launch. Now with the increasing demand, the organization needs to have a platform for automating deployment, scaling, and operations of application containers across clusters of hosts, As a DevOps engineer, you need implement a DevOps life cycle, such that all the requirements are implemented without any change in the Docker containers in the testing environment. Up until now, this organization used to follow a monolithic architecture with just 2 developers. The product is present on <https://github.com/hshar/website.git>

Following are the specifications of life-cycle:

1. Git workflow should be implemented. Since the company follows monolithic architecture of Development you need to take care of version control. The release should happen only on 25<sup>th</sup> of every month.
2. Code build should be triggered once the commits are made in the master Branch.
3. The code should be containerized with the help of the Docker file, The Dockerfile should be built every time if there is a push to Git-Hub. Create a custom Docker image using a Dockerfile.
4. 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.
5. Create a Jenkins pipeline script to accomplish the above task.
6. For configuration management of the infrastructure, you need to deploy the configuration on the servers to install necessary software and configurations.
7. Using Terraform accomplish the task of infrastructure creation in the AWS cloud provider.

**Step1: lets launch instance and name it as main. We will install ansible and terraform on this instance.**



We named instance as Main(T&A), T stands for terraform and A stands for ansible. Now lets install terraform on this instance, we will use this to launch different machine/architecture.

## Step2: Install Terraform


As we are using Ubuntu type instance. Check official documentation for installing the terraform.

<https://developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli>

```
sudo apt-get update && sudo apt-get install -y gnupg software-properties-common
wget -O- https://apt.releases.hashicorp.com/gpg | \
gpg --dearmor | \
sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg > /dev/null
gpg --no-default-keyring \
--keyring /usr/share/keyrings/hashicorp-archive-keyring.gpg \
--fingerprint
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] \
https://apt.releases.hashicorp.com $(lsb_release -cs) main" | \
sudo tee /etc/apt/sources.list.d/hashicorp.list
sudo apt update
sudo apt-get install terraform
```

Use above command one by one for installing terraform

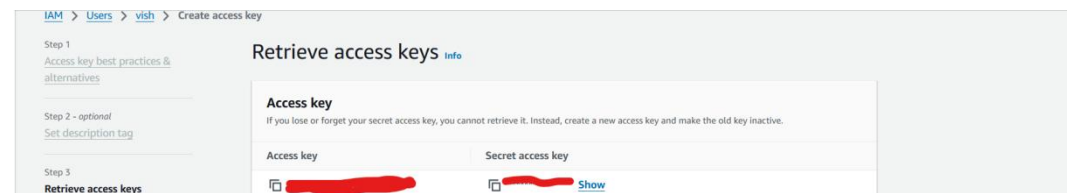
```
ubuntu@ip-172-31-10-26:~$ terraform --version
Terraform v1.9.8
on linux_amd64
ubuntu@ip-172-31-10-26:~$
```



We have installed terraform successfully

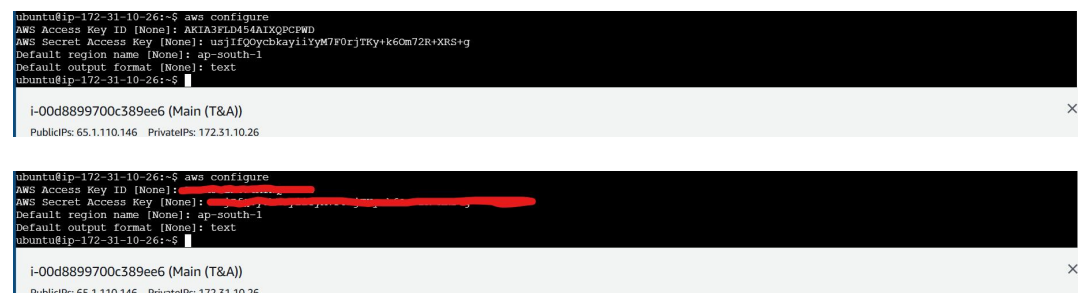
## Step3: lets create directory and create .tf file. We will use this to launch three additional instance and which will be our kubernetes master and 2 slave instance where we will run the container

We will set aws configuration with the help of iam user



Run command aws configure to add the configuration

```
ubuntu@ip-172-31-10-26:~$ aws configure
AWS Access Key ID [None]: AKIA3FLD454AIXQPCPWD
AWS Secret Access Key [None]: usjIfQyCbkayiiyM7f0rjTKy+k6Om72R+XRS+g
Default region name [None]: ap-south-1
Default output format [None]: text
ubuntu@ip-172-31-10-26:~$
```



Now we have setup the aws configuration, we will use this in our terraform file

```
provider "aws" {
  profile = "default"
  region = "ap-south-1"
}
resource "aws_instance" "Kubernetes_Master" {
  ami = "ami-09b0a86a2c84101e1"
```

```

instance_type = "t2.medium"
subnet_id = "subnet-0b4518708d215c36a"
key_name = "capstone"
tags = {
    Name = "k8s_master"
}
}
resource "aws_instance" "Kubernetes_Slave1" {
    ami      = "ami-09b0a86a2c84101e1"
    instance_type = "t2.micro"
    subnet_id = "subnet-0b4518708d215c36a"
    key_name = "capstone"
    tags = {
        Name = "k8s_slave1"
    }
}
resource "aws_instance" "Kubernetes_Slave2" {
    ami      = "ami-09b0a86a2c84101e1"
    instance_type = "t2.micro"
    subnet_id = "subnet-0b4518708d215c36a"
    key_name = "capstone"
    tags = {
        Name = "k8s_slave2"
    }
}

```

We will paste the above code in myec2.tf file and we will launch

```

ubuntu@ip-172-31-10-26:~$ ls
capstone_tfcode/
ubuntu@ip-172-31-10-26:~$ cd capstone_tfcode
ubuntu@ip-172-31-10-26:~/capstone_tfcode$ ls
ubuntu@ip-172-31-10-26:~/capstone_tfcode$ nano myec2.tf
ubuntu@ip-172-31-10-26:~/capstone_tfcode$

```

i-00d8899700c389ee6 (Main (T&A))

PublicIPs: 65.1.110.146 PrivateIPs: 172.31.10.26

Run terraform init , it will download required plugged in.

```

ubuntu@ip-172-31-10-26:~/capstone_tfcode$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.73.0...
- Installed hashicorp/aws v5.73.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
ubuntu@ip-172-31-10-26:~/capstone_tfcode$

```

i-00d8899700c389ee6 (Main (T&A))

PublicIPs: 65.1.110.146 PrivateIPs: 172.31.10.26

Run terraform apply to deploy it

```

aws_instance.Kubernetes_Slave1: Creating...
aws_instance.Kubernetes_Master: Creating...
aws_instance.Kubernetes_Slave2: Creating...
aws_instance.Kubernetes_Slave1: Still creating... [10s elapsed]
aws_instance.Kubernetes_Master: Still creating... [10s elapsed]
aws_instance.Kubernetes_Slave2: Still creating... [10s elapsed]
aws_instance.Kubernetes_Master: Creation complete after 13s [id=i-0209da0b0d848baf8]
aws_instance.Kubernetes_Slave1: Still creating... [20s elapsed]
aws_instance.Kubernetes_Slave2: Still creating... [20s elapsed]
aws_instance.Kubernetes_Slave1: Still creating... [30s elapsed]
aws_instance.Kubernetes_Slave2: Still creating... [30s elapsed]
aws_instance.Kubernetes_Master: Creation complete after 32s [id=i-021df9035df94ae22]
aws_instance.Kubernetes_Slave2: Creation complete after 32s [id=i-05c653c8af0d28c86]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
ubuntu@ip-172-31-10-26:~/capstone_tfcode$

```

i-00d8899700c389ee6 (Main (T&A))

PublicIPs: 65.1.110.146 PrivateIPs: 172.31.10.26

Instances (4) Info							
Find Instance by attribute or tag (case-sensitive)				All states			
<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
<input type="checkbox"/>	k8s_slave1	i-021df9035df9d4e22	Running	t2.micro	Initializing	View alarms +	ap-south-1a
<input type="checkbox"/>	k8s_slave2	i-05c653c8af0d28c86	Running	t2.micro	Initializing	View alarms +	ap-south-1a
<input type="checkbox"/>	k8s_master	i-0209da0b0d848baf8	Running	t2.medium	Initializing	View alarms +	ap-south-1a
<input type="checkbox"/>	Main (T&A)	i-00d8899700c389ee6	Running	t2.micro	2/2 checks passed	View alarms +	ap-south-1b

We have successfully launched the three instance. Now we will work installing ansible on main machine which will help us to install required.

#### Step 4 : connect with main machine and install ansible on it

```
sudo apt update
sudo apt install software-properties-common
sudo add-apt-repository --yes --update ppa:ansible/ansible
sudo apt install ansible
```

Lets install by running above command one by one to install ansible

```
ubuntu@ip-172-31-10-26:~$ ansible --version
ansible [core 2.17.5]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/home/ubuntu/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /home/ubuntu/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.10.12 (main, Sep 11 2024, 15:47:36) [GCC 11.4.0] (/usr/bin/python3)
  Jinja2 version = 3.0.3
  libyaml = True
ubuntu@ip-172-31-10-26:~$
```

-----> we will set password-less ssh connection between the instances, Generate public key with the command “ssh-keygen -t rsa”

After running the above command we will have public key, cat the public key and copy it

```
ubuntu@ip-172-31-10-26:~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCSgQ0LvcY130KjblN2XYd3zugdFS46z7phEpd+ztZ2MtmEuMWCzpc/z41d7MhVx20WUImqQt4F4ORRldKRF6VE2dhgFShsP9Wwp/lsFRYelCWEVNV53ZALR2P8yWmXNIA2RwL5f18KEL0jJ5dgPTVVFd3hkvTA1BHFzdh+4+Gfe6c527h8ozNduOuFrFX2YK4NLVMMXmoneZcqpPR8SJDJLana32cweI8WBDcIiUWMxERaWnocz+R6DmM4coerniazFBRbKYwF46dAyc4+kHMO2ALawRV4Wf9MNOJyHdR41KNZb+wxm19HOGIwn/QB9Psp4EoP2dRt90wyJjWmDARetw1WwAhMQ86vt3j1DR5+g6kemvU0940WfYrgWd13fBlXJkmWcb8W+KtoqTRF6B6Qeo3N0tEFAjHG4RVBqDq2yZFVpsE6+hhho9j1lWt8CJy47wC4jzwsE4NO0zUICRl8/z2DA0le2yRyguu85k4wUf6neY9+q1BZVM=
ubuntu@ip-172-31-10-26:~$
```

Now this copied key will be pasted in authorized\_keys of instance

```
ubuntu@ip-172-31-47-55:~$ nano .ssh/authorized_keys
ubuntu@ip-172-31-47-55:~$ cat .ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCSgQ0LvcY130KjblN2XYd3zugdFS46z7phEpd+ztZ2MtmEuMWCzpc/z41d7MhVx20WUImqQt4F4ORRldKRF6VE2dhgFShsP9Wwp/lsFRYelCWEVNV53ZALR2P8yWmXNIA2RwL5f18KEL0jJ5dgPTVVFd3hkvTA1BHFzdh+4+Gfe6c527h8ozNduOuFrFX2YK4NLVMMXmoneZcqpPR8SJDJLana32cweI8WBDcIiUWMxERaWnocz+R6DmM4coerniazFBRbKYwF46dAyc4+kHMO2ALawRV4Wf9MNOJyHdR41KNZb+wxm19HOGIwn/QB9Psp4EoP2dRt90wyJjWmDARetw1WwAhMQ86vt3j1DR5+g6kemvU0940WfYrgWd13fBlXJkmWcb8W+KtoqTRF6B6Qeo3N0tEFAjHG4RVBqDq2yZFVpsE6+hhho9j1lWt8CJy47wC4jzwsE4NO0zUICRl8/z2DA0le2yRyguu85k4wUf6neY9+q1BZVM=
ubuntu@ip-172-31-10-26:~$
```

```
ubuntu@ip-172-31-47-55:~$ nano .ssh/authorized_keys
ubuntu@ip-172-31-47-55:~$ cat .ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCSgQ0LvcY130KjblN2XYd3zugdFS46z7phEpd+ztZ2MtmEuMWCzpc/z41d7MhVx20WUImqQt4F4ORRldKRF6VE2dhgFShsP9Wwp/lsFRYelCWEVNV53ZALR2P8yWmXNIA2RwL5f18KEL0jJ5dgPTVVFd3hkvTA1BHFzdh+4+Gfe6c527h8ozNduOuFrFX2YK4NLVMMXmoneZcqpPR8SJDJLana32cweI8WBDcIiUWMxERaWnocz+R6DmM4coerniazFBRbKYwF46dAyc4+kHMO2ALawRV4Wf9MNOJyHdR41KNZb+wxm19HOGIwn/QB9Psp4EoP2dRt90wyJjWmDARetw1WwAhMQ86vt3j1DR5+g6kemvU0940WfYrgWd13fBlXJkmWcb8W+KtoqTRF6B6Qeo3N0tEFAjHG4RVBqDq2yZFVpsE6+hhho9j1lWt8CJy47wC4jzwsE4NO0zUICRl8/z2DA0le2yRyguu85k4wUf6neY9+q1BZVM=
ubuntu@ip-172-31-10-26:~$
```

```
ubuntu@ip-172-31-47-55:~$ nano .ssh/authorized_keys
ubuntu@ip-172-31-47-55:~$ cat .ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCSgQ0LvcY130KjblN2XYd3zugdFS46z7phEpd+ztZ2MtmEuMWCzpc/z41d7MhVx20WUImqQt4F4ORRldKRF6VE2dhgFShsP9Wwp/lsFRYelCWEVNV53ZALR2P8yWmXNIA2RwL5f18KEL0jJ5dgPTVVFd3hkvTA1BHFzdh+4+Gfe6c527h8ozNduOuFrFX2YK4NLVMMXmoneZcqpPR8SJDJLana32cweI8WBDcIiUWMxERaWnocz+R6DmM4coerniazFBRbKYwF46dAyc4+kHMO2ALawRV4Wf9MNOJyHdR41KNZb+wxm19HOGIwn/QB9Psp4EoP2dRt90wyJjWmDARetw1WwAhMQ86vt3j1DR5+g6kemvU0940WfYrgWd13fBlXJkmWcb8W+KtoqTRF6B6Qeo3N0tEFAjHG4RVBqDq2yZFVpsE6+hhho9j1lWt8CJy47wC4jzwsE4NO0zUICRl8/z2DA0le2yRyguu85k4wUf6neY9+q1BZVM=
ubuntu@ip-172-31-10-26:~$
```

```
ubuntu@ip-172-31-32-127:~$ nano .ssh/authorized_keys
ubuntu@ip-172-31-32-127:~$ cat .ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCSFRRTtIIPkLE8j2TPT07Gd135UzFk4iyJ8herV4CmCT3V007CqWJTJdggcfVMJZ+qnLarP05v0P8A0nKaoUbHnXhQ7ect1pnK6r9jNJC4Nh+x5CW2ccOXHxTo+4up
8gaohITvuDL1zorpyZyMzodwCOTRS2DT2KfjVRIOsaPw3d+/aSAvmP81sHRDlNs5gc4H97RcsFAUVPiDbFyp8os3RWatbvU4s6aM3/JVMu4kk/vdcI2enaXt0wAIzjJ0BVlyupp57TeoM+G6Tc5vtGhvfrrNTIwpJUrP
FV44s9Qc6vZwZep2LaeX1q1rzh8Vase1KerDAlF2BOwJ2i0ao0 capstone
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCSFRRTtIIPkLE8j2TPT07Gd135UzFk4iyJ8herV4CmCT3V007CqWJTJdggcfVMJZ+qnLarP05v0P8A0nKaoUbHnXhQ7ect1pnK6r9jNJC4Nh+x5CW2ccOXHxTo+4up
8gaohITvuDL1zorpyZyMzodwCOTRS2DT2KfjVRIOsaPw3d+/aSAvmP81sHRDlNs5gc4H97RcsFAUVPiDbFyp8os3RWatbvU4s6aM3/JVMu4kk/vdcI2enaXt0wAIzjJ0BVlyupp57TeoM+G6Tc5vtGhvfrrNTIwpJUrP
FV44s9Qc6vZwZep2LaeX1q1rzh8Vase1KerDAlF2BOwJ2i0ao0 capstone
RV4Wt9MNOJyHdR41KN2b+wxm19HOGIwn/QBSPsp4EoP2dRt90wyJjWMBdArEtclWMAhWQ86vt3jflDR5+g6kemvU0940WfYrgWDL3fBlXJmWcbBW+KtqqlRF6B6Qeo3NOTBfAjHG4RVBqDq2yZFVpsE6+hhHo9j1bW
t6Ccsy47wC4jzws84NO0zuICRiu0/z2DA01e2yYgyw85k4wOf6neY9+q1B2VM= ubuntu@ip-172-31-10-26
ubuntu@ip-172-31-32-127:~$
```

i-05c653c8af0d28c86 (k8s\_slave2)  
PublicIPs: 3.111.149.23 PrivateIPs: 172.31.32.127

Next we will add the private ip of the instances to complete the setup of ansible

Cd to /etc/ansible/hosts

```
[master]
172.31.47.55

[slave1]
172.31.34.103

[slave2]
172.31.32.127
```

i-00d8899700c389ee6 (Main (T&A))  
PublicIPs: 3.109.121.248 PrivateIPs: 172.31.10.26

### Step 5: as per above project we will install below tools on nodes

- Main (T&A) - Jenkins and java ( for automation)
- K8\_master- Kubernetes, docker, java
- K8slave1 - docker kubernetes
- K8slave2- docker kubernetes

We will be creating K8s cluster of 3 nodes.

Lets create script so we can bash it using absile, according to above requirement, we will need three script

### Main.sh (jenkins and java)

```
#!/bin/bash
sudo apt update
sudo apt install fontconfig openjdk-17-jre -y
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" \
https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
/etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt update
sudo apt install jenkins -y
```

```
ubuntu@ip-172-31-8-213:~$ nano main.sh
ubuntu@ip-172-31-8-213:~$
```

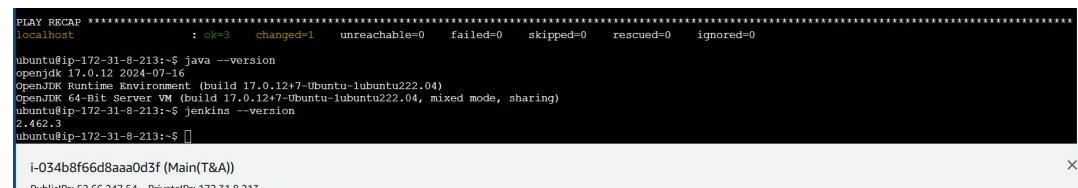
i-034b8f66d8aaa0d3f (Main(T&A))  
PublicIPs: 52.66.247.54 PrivateIPs: 172.31.8.213

Lets create playbook which will run aand install these script file

```
---
- name: Installation of Java and Jenkins
  hosts: localhost
  become: yes
  tasks:
    - name: Clean apt cache
      apt:
        autoclean: yes
        become: yes

    - name: Run the main.sh script
      script: main.sh
      become: yes
```

Now lets run it



We can see java and jenkins has been installed sucessfully on our main file. Now lets setup k8s cluser

**Step 6: create script file on master and add the below command and bash it. We are doing this manually as k8s will not be installed via ansible**

```
#!/bin/bash
set -euxo pipefail
```

```
# Kuernetes Variable Declaration
```

```
KUBERNETES_VERSION="1.29.0-1.1"
```

```
# disable swap
sudo swapoff -a
```

```
# keeps the swaf off during reboot
(crontab -l 2>/dev/null; echo "@reboot /sbin/swapoff -a") | crontab - || true
sudo apt-get update -y
```

```
# Install CRI-O Runtime
```

```
OS="xUbuntu_22.04"
```

```
VERSION="1.28"
```

```
# Create the .conf file to load the modules at bootup
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
```

```
br_netfilter
EOF
```

```
sudo modprobe overlay
sudo modprobe br_netfilter
```

```
# sysctl params required by setup, params persist across reboots
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward = 1
EOF
```

```
# Apply sysctl params without reboot
sudo sysctl --system
```

```
cat <<EOF | sudo tee /etc/apt/sources.list.d/devel:kubic:libcontainers:stable.list
deb https://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable/$OS/ /
EOF
cat <<EOF | sudo tee /etc/apt/sources.list.d/devel:kubic:libcontainers:stable:cri-o:$VERSION.list
deb http://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable:/cri-
o:$VERSION/$OS/ /
EOF
```

```
curl -L https://download.opensuse.org/repositories/devel:kubic:libcontainers:stable:cri-
o:$VERSION/$OS/Release.key | sudo apt-key --keyring /etc/apt/trusted.gpg.d/libcontainers.gpg add -
curl -L
https://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable/$OS/Release.key |
sudo apt-key --keyring /etc/apt/trusted.gpg.d/libcontainers.gpg add -
```

```
sudo apt-get update
sudo apt-get install cri-o cri-o-runc -y
```

```
sudo systemctl daemon-reload
sudo systemctl enable crio --now
```

```
echo "CRI runtime installed successfully"
```

```
# Install kubelet, kubectl and Kubeadm
```

```
sudo apt-get update -y
sudo apt-get install -y apt-transport-https ca-certificates curl gpg
```

```
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/deb/Release.key | sudo gpg --dearmor -o
/etc/apt/keyrings/kubernetes-1-28-apt-keyring.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-1-28-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.28/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes-1.28.list
```

```
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.29/deb/Release.key | sudo gpg --dearmor -o
/etc/apt/keyrings/kubernetes-1-29-apt-keyring.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-1-29-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.29/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes-1.29.list
```

```
sudo apt-get update -y
sudo apt-get install -y kubelet="$KUBERNETES_VERSION" kubectl="$KUBERNETES_VERSION"
kubeadm="$KUBERNETES_VERSION"
sudo apt-get update -y
```

```
sudo apt-mark hold kubelet kubeadm kubectl
```

```
sudo apt-get install -y jq
```

```
local_ip="$(ip --json addr show eth0 | jq -r '[0].addr_info[] | select(.family == "inet") | .local)'"
cat > /etc/default/kubelet << EOF
KUBELET_EXTRA_ARGS=--node-ip=$local_ip
EOF
```

```
sudo bash install.sh
```

```
-----
## Execute ONLY on "Master Node" ( one-by-one ) :
```

```
#!/bin/bash
```

```
# Setup for Control Plane (Master) servers
```

```
set -euxo pipefail
```

```
# If you need public access to API server using the servers Public IP adress, change PUBLIC_IP_ACCESS
to true.
```

```
PUBLIC_IP_ACCESS="true"
```

```
NODENAME=$(hostname -s)
```

```
POD_CIDR="192.168.0.0/16"
```

```
# Pull required images
```

```
sudo kubeadm config images pull
```

```
# Initialize kubeadm based on PUBLIC_IP_ACCESS
```

```
if [[ "$PUBLIC_IP_ACCESS" == "false" ]]; then
```

```
    MASTER_PRIVATE_IP=$(ip addr show eth0 | awk '/inet / {print $2}' | cut -d/ -f1)
    sudo kubeadm init --apiserver-advertise-address="$MASTER_PRIVATE_IP" --apiserver-cert-extra-
sans="$MASTER_PRIVATE_IP" --pod-network-cidr="$POD_CIDR" --node-name "$NODENAME" --
ignore-preflight-errors Swap
```

```
elif [[ "$PUBLIC_IP_ACCESS" == "true" ]]; then
```

```
    MASTER_PUBLIC_IP=$(curl ifconfig.me && echo "")
    sudo kubeadm init --control-plane-endpoint="$MASTER_PUBLIC_IP" --apiserver-cert-extra-
sans="$MASTER_PUBLIC_IP" --pod-network-cidr="$POD_CIDR" --node-name "$NODENAME" --ignore-
preflight-errors Swap
```

```
else
```

```
    echo "Error: MASTER_PUBLIC_IP has an invalid value: $PUBLIC_IP_ACCESS"
    exit 1
fi
```

```
# Configure kubeconfig
```

```
mkdir -p "$HOME"/.kube
```

```
sudo cp -i /etc/kubernetes/admin.conf "$HOME"/.kube/config
```

```
sudo chown "$(id -u)": "$(id -g)" "$HOME"/.kube/config
```



# Install Calico Network Plugin Network

kubectl apply -f <https://docs.projectcalico.org/manifests/calico.yaml>

```
ubuntu@ip-172-31-38-48:~$ sudo nano install.sh
ubuntu@ip-172-31-38-48:~$ sudo bash install.sh
+ KUBERNETES_VERSION=1.29.0-1.1
+ sudo swapoff -a
+ crontab -l

i-03fb281f6ea68cd90 (K8s_master)
PublicIPs: 3.108.196.157 PrivateIPs: 172.31.38.48

No VM guests are running outdated hypervisor (qemu) binaries on this host.
++ ip -j --json --getid show eth0
++ jq -r '[.[] | select(.family == "inet") | .local'
+ local_ip=172.31.38.48
+ cat
ubuntu@ip-172-31-38-48:~$ []

i-03fb281f6ea68cd90 (K8s_master)
PublicIPs: 3.108.196.157 PrivateIPs: 172.31.38.48
```

Similarly we will run these commands on slave nodes also

**Step7: Now we have to run few command manually, run below command on master node one by one to setup k8s**

## Execute ONLY on "Master Node" ( one-by-one ) :

#!/bin/bash

# Setup for Control Plane (Master) servers

set -euxo pipefail

# If you need public access to API server using the servers Public IP adress, change PUBLIC\_IP\_ACCESS to true.

PUBLIC\_IP\_ACCESS="true"

NODENAME=\$(hostname -s)

POD\_CIDR="192.168.0.0/16"

# Pull required images

sudo kubeadm config images pull

# Initialize kubeadm based on PUBLIC\_IP\_ACCESS

if [[ "\$PUBLIC\_IP\_ACCESS" == "false" ]]; then

MASTER\_PRIVATE\_IP=\$(ip addr show eth0 | awk '/inet / {print \$2}' | cut -d/ -f1)  
sudo kubeadm init --apiserver-advertise-address="\$MASTER\_PRIVATE\_IP" --apiserver-cert-extra-sans="\$MASTER\_PRIVATE\_IP" --pod-network-cidr="\$POD\_CIDR" --node-name "\$NODENAME" --ignore-preflight-errors Swap

elif [[ "\$PUBLIC\_IP\_ACCESS" == "true" ]]; then

MASTER\_PUBLIC\_IP=\$(curl ifconfig.me && echo "")  
sudo kubeadm init --control-plane-endpoint="\$MASTER\_PUBLIC\_IP" --apiserver-cert-extra-sans="\$MASTER\_PUBLIC\_IP" --pod-network-cidr="\$POD\_CIDR" --node-name "\$NODENAME" --ignore-preflight-errors Swap

else

echo "Error: MASTER\_PUBLIC\_IP has an invalid value: \$PUBLIC\_IP\_ACCESS"  
exit 1

fi

## # Configure kubeconfig

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

## # Install Calico Network Plugin Network

kubectl apply -f <https://docs.projectcalico.org/manifests/calico.yaml>

Once we run all the commands we will have join commands

```
kubeadm join 3.108.196.157:6443 --token b3wkl1.oa6x3p8ejcktxy0 \
--discovery-token-ca-cert-hash sha256:06f510e9c107a89dbf292f396f26d2845b7e0e9f27a7854b87e7948329fd3b0
ubuntu@ip-172-31-38-48:~$ mkdir -p "$HOME"/.kube
sudo cp -i /etc/kubernetes/admin.conf "$HOME"/.kube/config
sudo chown "$(id -u)": "$(id -g)" "$HOME"/.kube/config
+ mkdir -p /home/ubuntu/.kube

i-03fb281f6ea68cd90 (K8s_master)
PublicIPs: 3.108.196.157 PrivateIPs: 172.31.38.48
```

We will run this join commands on slave/worker node

Now go to worker node and run below command  
sudo kubeadm reset pre-flight checks

After that we will

```
ubuntu@ip-172-31-34-193:~$ sudo su
root@ip-172-31-34-193:/home/ubuntu# kubeadm join 3.108.196.157:6443 --token b3wkl1.oa6x3p8ejcktxy0 \
--discovery-token-ca-cert-hash sha256:06f510e9c107a89dbf292f396f26d2845b7e0e9f27a7854b87e7948329fd3b0 --v=5
i1026 14:36:37.63541 6292 join.go:113] [preflight] found NodeName empty; using OS hostname as NodeName
i1026 14:36:37.635709 6292 initconfiguration.go:122] detected and using CRI socket: unix:///var/run/crio/crio.sock
[preflight] Running pre-flight checks
i1026 14:36:37.635821 6292 preflight.go:93] [preflight] Running general checks

i-0155b5b6ecc29200a (K8s_slave1)
PublicIPs: 13.235.23.121 PrivateIPs: 172.31.34.193

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.
root@ip-172-31-34-193:/home/ubuntu# []

i-0155b5b6ecc29200a (K8s_slave1)
PublicIPs: 13.235.23.121 PrivateIPs: 172.31.34.193

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.
root@ip-172-31-36-52:/home/ubuntu#

i-001c20c8e01d87451 (K8s_slave2)
PublicIPs: 15.206.145.43 PrivateIPs: 172.31.36.52
```

```
Last login: Sat Oct 26 13:59:35 2024 from 13.233.177.5
ubuntu@ip-172-31-38-48:~$ kubectl get nodes
NAME STATUS ROLES AGE VERSION
ip-172-31-34-193 Ready <none> 17m v1.29.0
ip-172-31-36-52 Ready <none> 76s v1.29.0
ip-172-31-38-48 Ready control-plane 36m v1.29.0
ubuntu@ip-172-31-38-48:~$

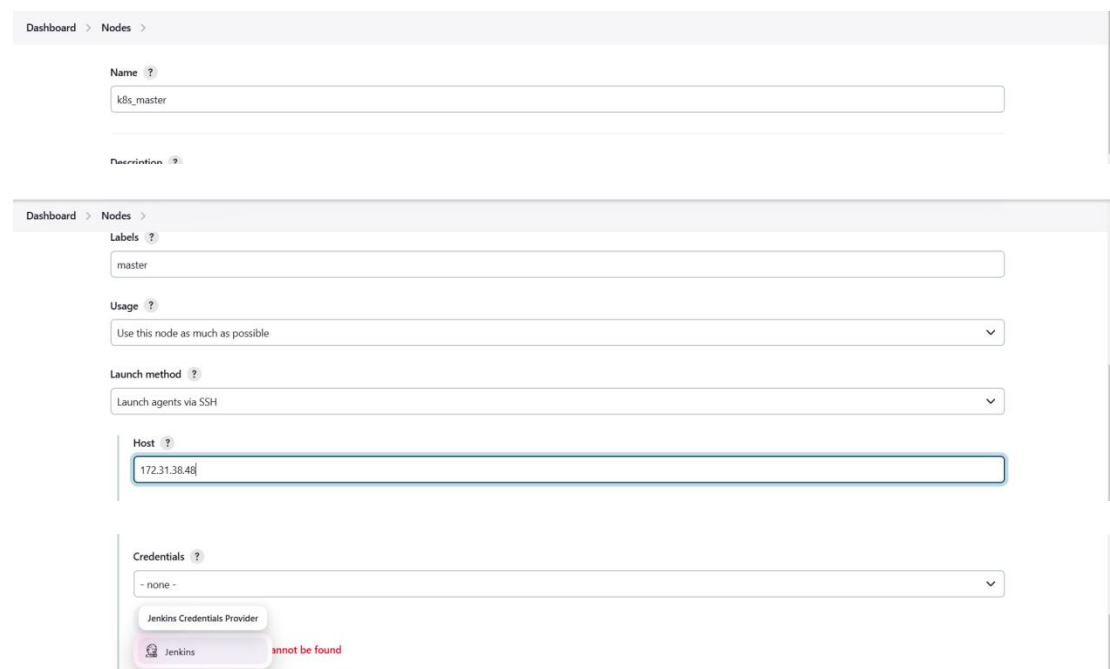
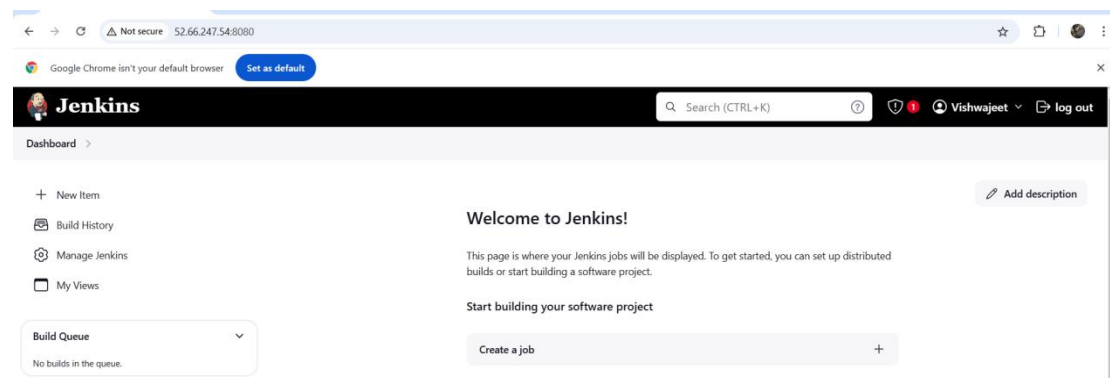
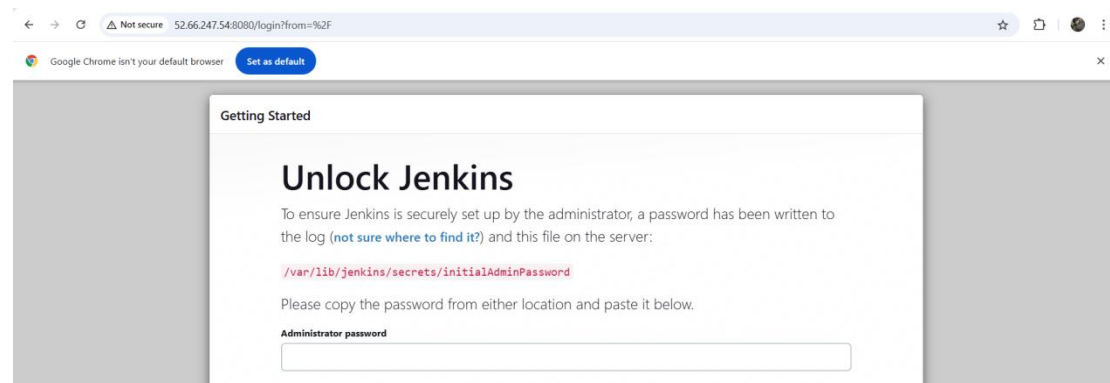
i-03fb281f6ea68cd90 (K8s_master)
PublicIPs: 3.108.196.157 PrivateIPs: 172.31.38.48
```

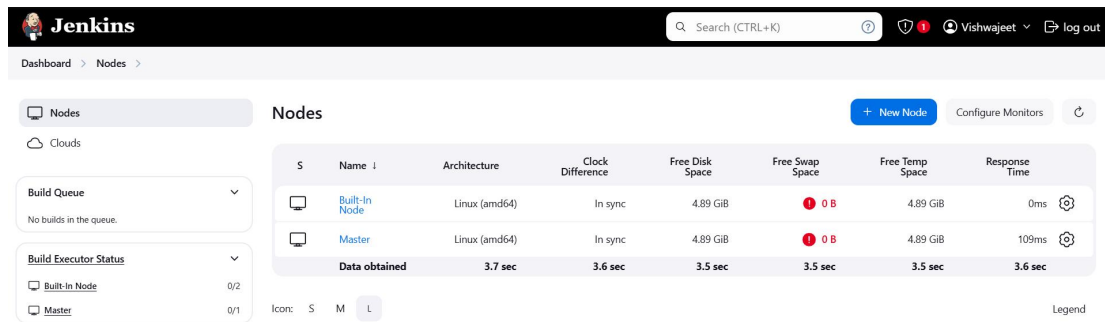
<input type="checkbox"/>	K8s_slave2	i-001c20c8e01d87451	<span>Running</span>	t2.medium	<span>2/2 checks passed</span>	<a href="#">View alarms</a>	ap-south-1a	ec2
<input checked="" type="checkbox"/>	Main(T&A)	i-034b8f66d8aaa0d3f	<span>Running</span>	t2.micro	<span>2/2 checks passed</span>	<a href="#">View alarms</a>	ap-south-1b	ec2

i-034b8f66d8aaa0d3f (Main(T&A))								
<input type="checkbox"/> i-034b8f66d8aaa0d3f		52.66.247.54   <a href="#">open address</a>			<input type="checkbox"/> 172.31.8.213			

## Step7: lets set the jenkins





The screenshot shows the Jenkins 'Nodes' page. On the left, there are sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (Built-In Node: 0/2, Master: 0/1). The main area displays a table of nodes:

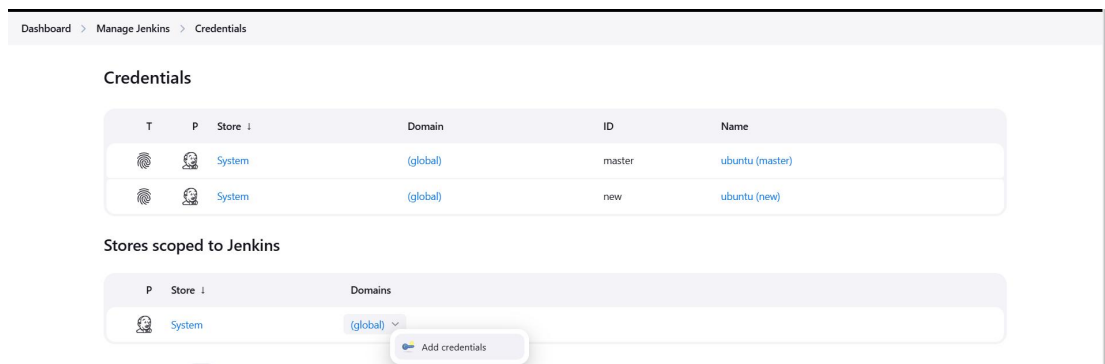
S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	4.89 GiB	1 0 B	4.89 GiB	0ms
	Master	Linux (amd64)	In sync	4.89 GiB	1 0 B	4.89 GiB	109ms
Data obtained				3.7 sec	3.6 sec	3.5 sec	3.6 sec

Buttons for '+ New Node', 'Configure Monitors', and a refresh icon are at the top right. A 'Legend' link is at the bottom right.

Node has been added successfully.

### Step 8: lets add dockerhub cred

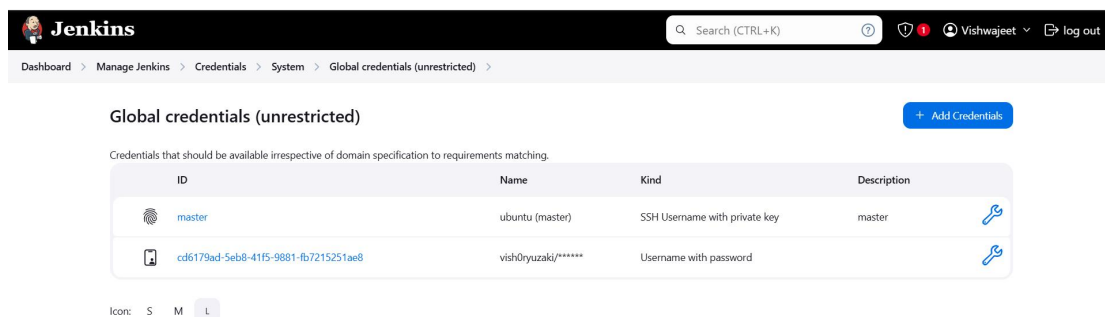
Go to manage jenkins and then credentials



The screenshot shows the Jenkins 'Credentials' page. It lists two credentials:

T	P	Store	Domain	ID	Name
		System	(global)	master	ubuntu (master)
		System	(global)	new	ubuntu (new)

Below this, the 'Stores scoped to Jenkins' section shows a dropdown for 'System' and a domain dropdown set to '(global)', with an 'Add credentials' button.



The screenshot shows the 'Global credentials (unrestricted)' page in Jenkins. It contains a table of credentials:

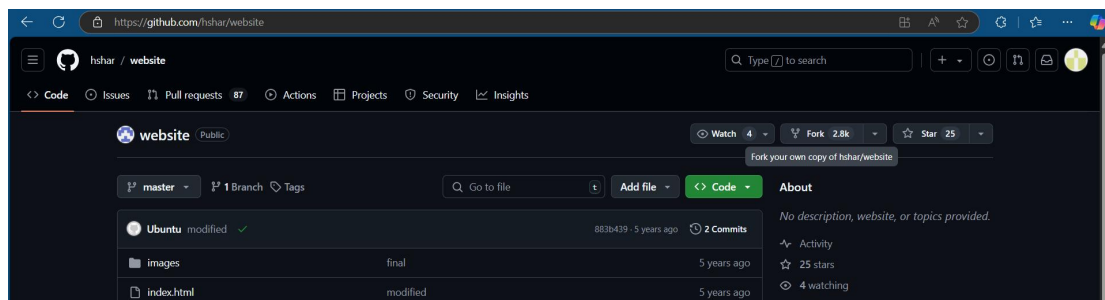
ID	Name	Kind	Description
	master	ubuntu (master)	SSH Username with private key
	cd6179ad-5eb8-41f5-9881-fb7215251ae8	vish0ryuzaki/*****	Username with password

Buttons for '+ Add Credentials' and 'Add Credentials' are visible. A legend at the bottom shows 'Icon: S M L'.

Added the credentails

### Step 9: lets fork the repo, go to below repo

<https://github.com/hshar/website.git>



The screenshot shows the GitHub repository page for 'hshar/website'. The repository is public and has 4 watchers, 2.8k forks, and 25 stars. The file list shows:

- Ubuntu modified ✓ (883b439 - 5 years ago, 2 Commits)
- images final (5 years ago)
- index.html modified (5 years ago)

The 'About' section states: 'No description, website, or topics provided.'

### Create a new fork

A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks.](#)

*Required fields are marked with an asterisk (\*).*

Owner \* Vish0ruyzaki / Repository name \* website  
website is available.

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

☒ Copy the master branch only  
 Contribute back to hshar/website by adding your own branch. [Learn more.](#)

1 You are creating a fork in your personal account.

Create fork

website

Public

forked from hshar/website

Pin

Watch 0

Fork

Star 0

master

1 Branch

Tags

Go to file

Add file

Code

About

Activity

0 stars

0 watching

0 forks

Releases

This branch is up to date with hshar/website:master.

Contribute

Sync fork

Ubuntu

modified

883b439 · 5 years ago

2 Commits

images

final

5 years ago

index.html

modified

5 years ago

## Step 10: lets create dockerfile inside the forked repo

FROM ubuntu/apache2  
COPY . /var/www/html

website / Dockerfile in master

Cancel changes Commit changes...

Edit Preview Code 55% faster with GitHub Copilot

Spaces 2 No wrap

```

1 FROM ubuntu/apache2
2 COPY . /var/www/html

```

Vish0ruyzaki Create Dockerfile

7b85a6 · now History

This branch is 1 commit ahead of hshar/website:master.

Contribute Sync fork

Name	Last commit message	Last commit date
images	final	5 years ago
Dockerfile	Create Dockerfile	now
index.html	modified	5 years ago

## Step 11: lets create the job and pipeline

Dashboard > Testing > Configuration

### Configure

- General
- Advanced Project Options
- Pipeline**

Pipeline script

```

1 pipeline {
2   agent none
3   environment {
4     DOCKERHUB_CREDENTIALS=credentials('dockerhub')
5   }
6   stages {
7     stage('Hello') {
8       steps {
9         echo 'Hello World'
10      }
11    }
12    stage('Git') {
13      agent {
14        label 'Master'
15      }
16      steps {
17        git 'https://github.com/Vish00uyzaki/webSite.git'
18      }
19    }
20  }
21 }

```

☒ Use Groovy Sandbox

[Pipeline Syntax](#)

**Save** **Apply**

Lets run and check

Dashboard > Testing >

**Status** **Testing** [Add description](#)

[Changes](#)

**Build Now**

[Configure](#)

[Delete Pipeline](#)

[Stages](#)

[Rename](#)

[Pipeline Syntax](#)

**Permalinks**

- Last build (#1), 9 min 32 sec ago
- Last stable build (#1), 9 min 32 sec ago
- Last successful build (#1), 9 min 32 sec ago
- Last completed build (#1), 9 min 32 sec ago

**Builds**

Filter

Today

- #2 4:24 AM
- #1 4:15 AM

```

First time build. Skipping changelog.
[Pipeline] }
[Pipeline] // node
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withCredentials
[Pipeline] End of Pipeline
Finished: SUCCESS

```

```

ubuntu@ip-172-31-39-39:~$ ls
install.sh jenkins
ubuntu@ip-172-31-39-39:~$ cd jenkins
ubuntu@ip-172-31-39-39:~/jenkins$ ls
remoting remoting.jar workspace
ubuntu@ip-172-31-39-39:~/jenkins$ cd workspace
ubuntu@ip-172-31-39-39:~/jenkins/workspace$ ls
testpipeline
ubuntu@ip-172-31-39-39:~/jenkins/workspace$ cd testpipeline
ubuntu@ip-172-31-39-39:~/jenkins/workspace/testpipeline$ ls
Dockerfile images index.html
ubuntu@ip-172-31-39-39:~/jenkins/workspace/testpipeline$

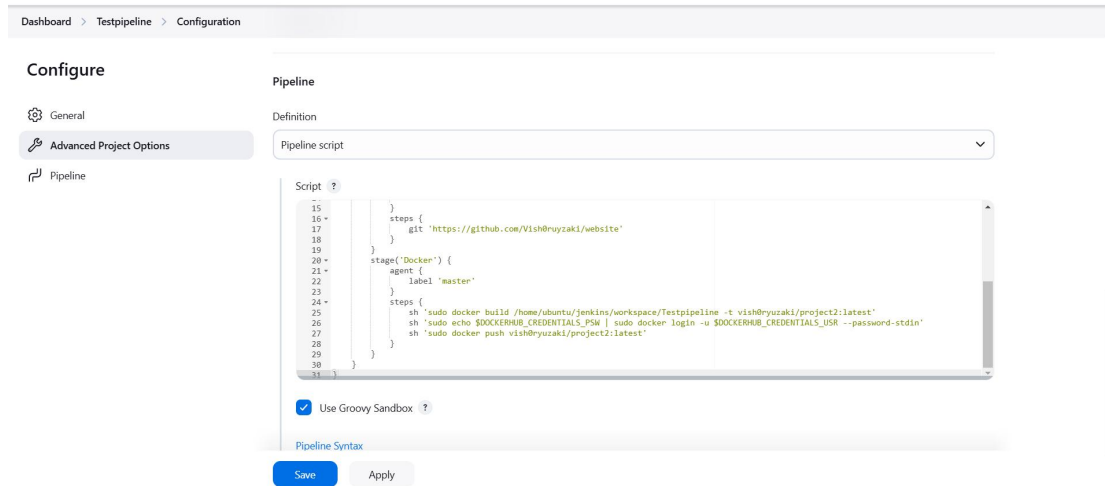
```

i-0916d8df61fb3bc9e (master)

PublicIPs: 3.111.157.223 PrivateIPs: 172.31.39.39

We can see the files has been imported from github to the master instance

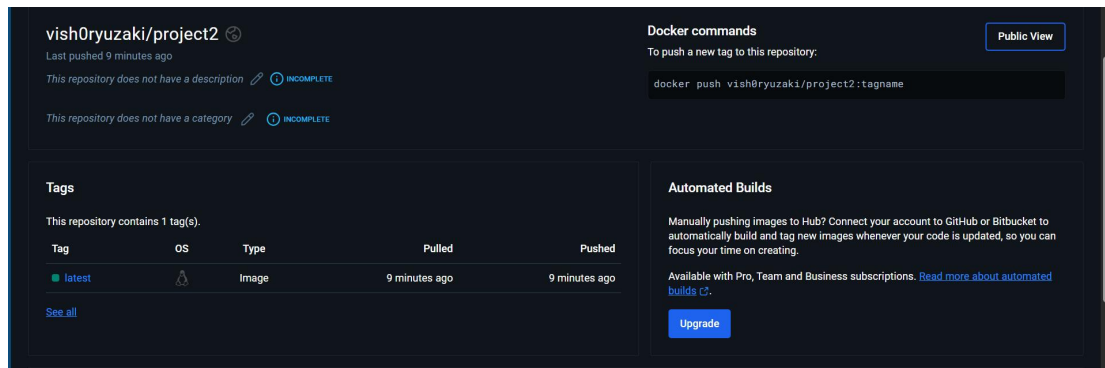
Now we will create image and push it to docker hub



```

pipeline {
    agent none
    environment {
        DOCKERHUB_CREDENTIALS=credentials('cd6179ad-5eb8-41f5-9881-fb7215251ae8')
    }
    stages {
        stage('Hello') {
            steps {
                echo 'Hello World'
            }
        }
        stage('Git') {
            agent {
                label 'master'
            }
            steps {
                git 'https://github.com/Vish0ryuzaki/website'
            }
        }
        stage('Docker') {
            agent {
                label 'master'
            }
            steps {
                sh 'sudo docker build /home/ubuntu/jenkins/workspace/Testpipeline -t
vish0ryuzaki/project2:latest'
                sh 'sudo echo $DOCKERHUB_CREDENTIALS_PSW | sudo docker login -u
$DOCKERHUB_CREDENTIALS_USR --password-stdin'
                sh 'sudo docker push vish0ryuzaki/project2:latest'
            }
        }
    }
}

```

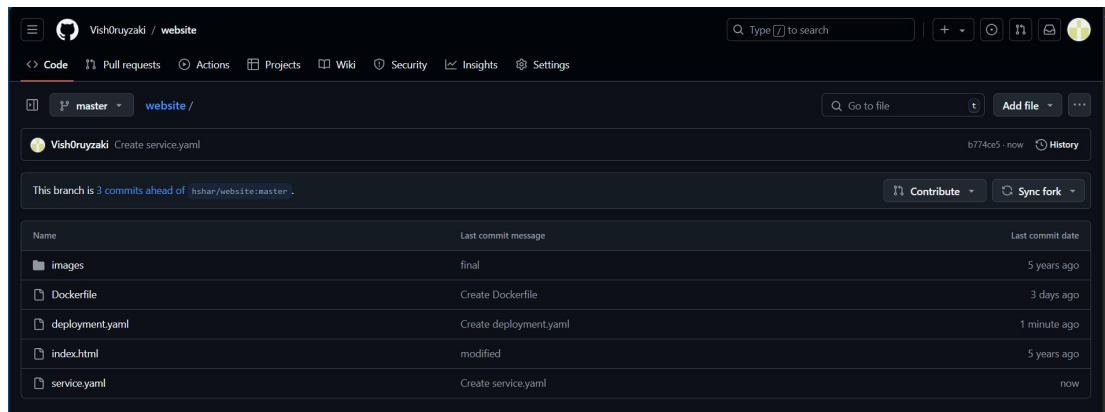


**Step10: now lets create the yaml file to deply the container and nodeport. We will add this on github**

```
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: vish0ryuzaki/project2:latest
          ports:
            - containerPort: 80
```

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





Step11: now we will create next pipeline script which will be below

```

pipeline {
  agent none
  environment {
    DOCKERHUB_CREDENTIALS=credentials('cd6179ad-5eb8-41f5-9881-fb7215251ae8')
  }
  stages {
    stage('Hello') {
      steps {
        echo 'Hello World'
      }
    }
    stage('Git') {
      agent {
        label 'master'
      }
      steps {
        git 'https://github.com/Vish0ruyzaki/website'
      }
    }
    stage('Docker') {
      agent {
        label 'master'
      }
      steps {
        sh 'sudo docker build /home/ubuntu/jenkins/workspace/Testpipeline -t
vish0ryuzaki/project2:latest'
        sh 'sudo echo $DOCKERHUB_CREDENTIALS_PSW | sudo docker login -u
$DOCKERHUB_CREDENTIALS_USR --password-stdin'
        sh 'sudo docker push vish0ryuzaki/project2:latest'
      }
    }
    stage('K8s') {
      agent {
        label 'master'
      }
      steps {
        sh 'kubectl apply -f deployment.yaml'
        sh 'kubectl apply -f service.yaml'
      }
    }
  }
}

```

}

Dashboard > Testpipeline > Configuration

### Configure

- General
- Advanced Project Options**
- Pipeline

**Pipeline**

Definition

Pipeline script

```
Script ?
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
}

sh "sudo docker build /home/ubuntu/jenkins/workspace/Testpipeline -t vish@ryuzaki/project2:latest"
sh "sudo echo $DOCKERHUB_CREDENTIALS_PSW | sudo docker login -u $DOCKERHUB_CREDENTIALS_USR --password-stdin"
sh "sudo docker push vish@ryuzaki/project2:latest"

stage('K8s') {
  agent {
    label 'master'
  }
  steps {
    sh 'kubectl apply -f deployment.yaml'
    sh 'kubectl apply -f service.yaml'
  }
}
```

☒ Use Groovy Sandbox ?

Lets run it

Jenkins

Search (CTRL+K)

Dashboard > Testpipeline

### Status

Changes

Build Now

Configure

Delete Pipeline

Stages

Rename

Pipeline Syntax

### Testpipeline

Add description

#### Permalinks

- Last build (#14), 1 min 18 sec ago
- Last stable build (#14), 1 min 18 sec ago
- Last successful build (#14), 1 min 18 sec ago
- Last failed build (#13), 5 min 54 sec ago
- Last unsuccessful build (#13), 5 min 54 sec ago
- Last completed build (#14), 1 min 18 sec ago

#### Builds

Filter

Today

#14 5:35 AM

Now lets check if we can see the project on the slave ip

Instances (1/4) info

Last updated about 2 hours ago

Connect

Instance state

Actions

Launch instances

Find Instance by attribute or tag (case-sensitive)

All states

Instance state = running

Clear filters

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Pub
master	i-0916d8df61fb3bc9e	Running	t2.medium	2/2 checks passed	View alarms +	ap-south-1a	ec2
<input checked="" type="checkbox"/> slave1	i-039c50d409406ca46	Running	t2.medium	2/2 checks passed	View alarms +	ap-south-1a	ec2
slave2	i-0c37d61785f2432a0	Running	t2.medium	2/2 checks passed	View alarms +	ap-south-1a	ec2
Main (T&A)	i-0b32ab6976f7f7134	Running	t2.micro	2/2 checks passed	View alarms +	ap-south-1b	ec2

### i-039c50d409406ca46 (slave1)

Details

Status and alarms

Monitoring

Security

Networking

Storage

Tags

#### Instance summary

Instance ID

i-039c50d409406ca46

Public IPv4 address copied

43.205.145.10 | open address

Private IPv4 addresses

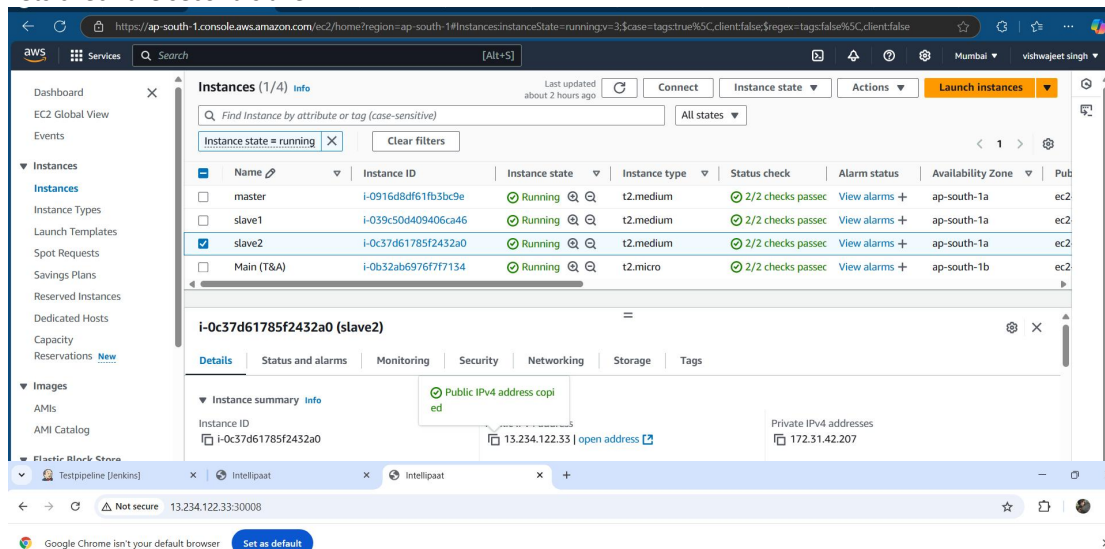
172.31.36.221



Hello world!



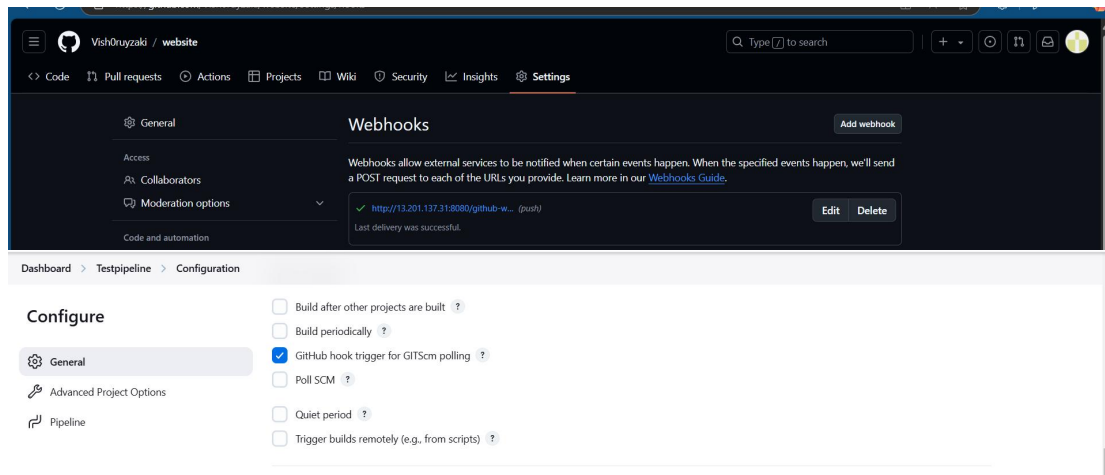
Lets check the second slave



Hello world!



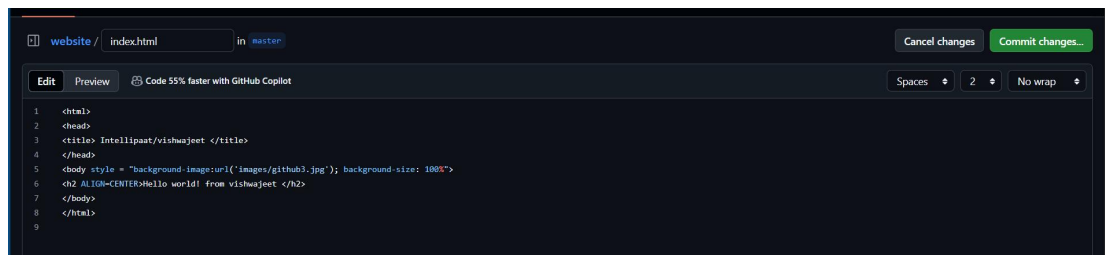
Step12: to automate the pipeline, we will steup webhook on the gitub



Now we will modify the jenkins scrips as well because the port will be occupied and previous pods needs to be deleted in order to create new one.. hence we will and below command

Sh 'kubectl delete deploy nginx-deployment'

Now we will make chnages on git to run the pipeline automatctially



We have chnged the heaader and message

**Commit changes** ✕

**Commit message**

Update index.html

**Extended description**

Add an optional extended description..

☒ Commit directly to the master branch

☐ Create a **new branch** for this commit and start a pull request [Learn more about pull requests](#)

Cancel Commit changes

We can see below pipeline has been running on its own

# Builds

Today

✓

#17

6:06 AM

✗

⌵

✓

#16

5:49 AM

⌵

✓

#15

5:44 AM

⌵

Dashboard > Testpipeline > #17

```

[Pipeline] stage
[Pipeline] { (K8s)
[Pipeline] node
Running on master 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

```

Now lets check this on browser

Testpipeline #17 Console (Jenki...
IntelliPaat/vishwajeet
IntelliPaat/vishwajeet

Not secure 43.205.145.10:30008

Google Chrome isn't your default browser Set as default

Hello world! from vishwajeet



