Project : Capstone II

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 to implement a DevOps lifecycle 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 the lifecycle:

1. Git workflow should be implemented. Since the company follows a monolithic architecture of development, you need to take care of version control. The release should happen only on the 25th of every month.

2. CodeBuild should be triggered once the commits are made in the master branch.

3. The code should be containerized with the help of the Dockerfile. The Dockerfile should be built every time if there is a push to GitHub. 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.

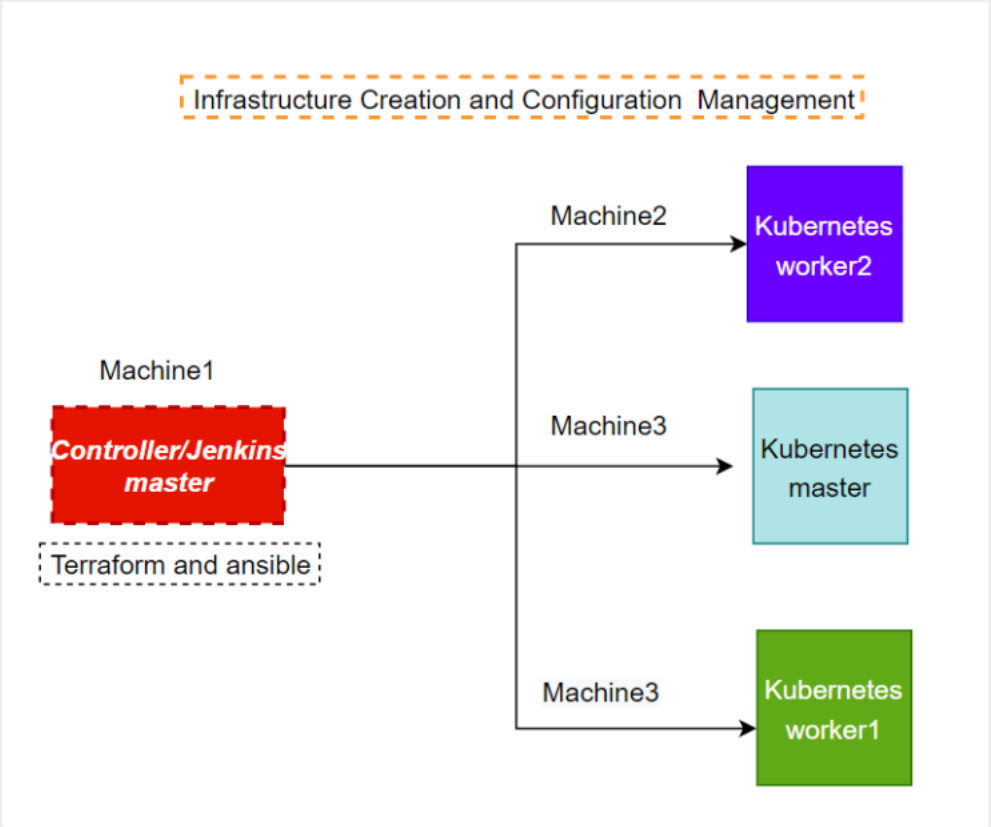
7. Using Terraform, accomplish the task of infrastructure creation in the AWS cloud provider.
Architectural Advice: Softwares to be installed on the respective machines using configuration management.

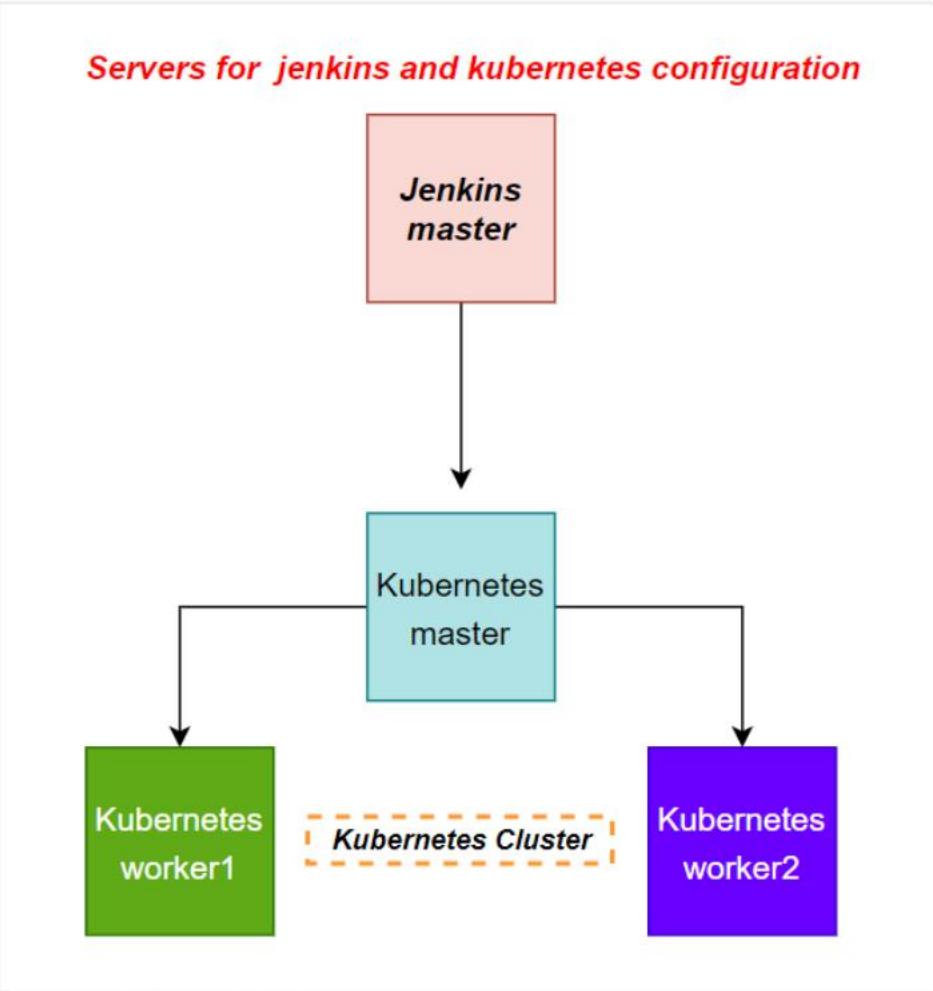
Worker1: Jenkins, Java

Worker2: Docker, Kubernetes

Worker3: Java, Docker, Kubernetes

Worker4: Docker, Kubernetes


Infrastructure Creation and Configuration Management
Machine2
Kubernetes worker2
Machine1
Controller/Jenkins master
Machine3
Kubernetes master
Terraform and ansible
Machine3
Kubernetes worker1


Servers for jenkins and kubernetes configuration
Jenkins master
Kubernetes master
Kubernetes worker1
Kubernetes Cluster
Kubernetes worker2

Terraform installation

- Created a EC2 instance manually in aws console
- Navigated into the EC2 instance and installed terraform through a shellscript

```
ubuntu@worker1:~$ vi terraforminstall.sh
ubuntu@worker1:~$ bash terraforminstall.sh
```

i-00b8547948a9cfa59 (Worker1)

PublicIPs: 3.141.13.91 PrivateIPs: 172.31.26.221

```
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
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 -y
~
~
~
~
~
~
~
~
"terraforminstall.sh" 12L, 548B
```

Architecture creation

After installation of terraform created a directory named infracreation

- Navigated inside the directory and created a main.tf file with a hcl script to create three EC2 instances

```
ubuntu@worker1:~$ mkdir infracreation
ubuntu@worker1:~$ cd infracreation
ubuntu@worker1:~/infracreation$ vi main.tf
```

```
provider "aws"{
    secret_key= ""
    access_key= ""
    region= "eu-east-2"
}

resource "aws_instance" "k8master"{
    ami= "ami-05fb0b8c1424f266b"
    instance_type= "t2.medium"
    key_name= "venkatohio"
    provider= "aws.worker2"
    tags= {
        Name= "worker2"
    }
}
resource "aws_instance" "k8slave"{
    ami= "ami-05fb0b8c1424f266b"
    instance_type= "t2.medium"
    key_name= "venkatohio"
    provider= "aws.worker3"
```

```
    tags= {
        Name= "worker3"
    }
}
resource "aws_instance" "k8slave"{
    ami= "ami-05fb0b8c1424f266b"
    instance_type= "t2.medium"
    key_name= "venkatohio"
    provider= "aws.worker4"
    tags= {
        Name= "worker4"
    }
}
```

- Then initialized terraform

```
ubuntu@worker1:~/infracreation$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.31.0...
- Installed hashicorp/aws v5.31.0 (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@worker1:~/infracreation$
```

- Performed terraform plan

```
ubuntu@worker1:~/infracreation$ terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_instance.this2 will be created
  + resource "aws_instance" "this2" {
      + ami                                  = "ami-05fb0b8c1424f266b"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
```

- And then applied the changes by terraform apply command

```
aws_instance.this3: Creating...
aws_instance.this4: Creating...
aws_instance.this2: Creating...
aws_instance.this3: Still creating... [10s elapsed]
aws_instance.this4: Still creating... [10s elapsed]
aws_instance.this2: Still creating... [10s elapsed]
aws_instance.this3: Still creating... [20s elapsed]
aws_instance.this4: Still creating... [20s elapsed]
aws_instance.this2: Still creating... [20s elapsed]
aws_instance.this3: Still creating... [30s elapsed]
aws_instance.this4: Still creating... [30s elapsed]
aws_instance.this2: Still creating... [30s elapsed]
aws_instance.this3: Creation complete after 31s [id=i-07edd73dde8245ebf]
aws_instance.this4: Creation complete after 31s [id=i-03b429699eedc391e]
aws_instance.this2: Still creating... [40s elapsed]
aws_instance.this2: Still creating... [50s elapsed]
aws_instance.this2: Creation complete after 52s [id=i-0b31db3012c176c15]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
ubuntu@worker1:~/infracreation$
```

Ansible configuration

- Then installed ansible in the server through a ansibleinstall.sh shell script file

```
  GNU nano 6.2                                          ansibleinstall.sh
sudo apt update
sudo apt install software-properties-common
sudo apt-add-repository ppa: ansible/ansible
sudo apt install ansible

                                                        [ Read 4 lines ]
^G Help      ^O Write Out   ^W Where Is    ^K Cut       ^T Execute     ^C Location    M-U Und
^X Exit      ^R Read File   ^\ Replace     ^U Paste     ^J Justify     ^/ Go To Line  M-E Red
```

Ansible is successfully installed

```
ubuntu@worker1:~$ ansible --version
ansible 2.10.8
  config file = None
  configured module search path = ['/home/ubuntu/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0]
ubuntu@worker1:~$
```

- Ssh key is generated

```
ubuntu@worker1:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_rsa
Your public key has been saved in /home/ubuntu/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:rSszP1TSJq9shqpp9O+E2D2pfoOyS920KCG421+fwCE ubuntu@worker1
The key's randomart image is:
+---[RSA 3072]----+
|                 |
|                 |
|          .      |
|.        o.+     |
|o . E o S*.      |
| o.= O +...      |
|..+.=.@+..       |
| +ooo+B+*o       |
|..*B==oO=.       |
+----[SHA256]-----+
```

- Ssh private key is copied

```
ubuntu@worker1:~$ sudo cat /home/ubuntu/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCxVi0IcrTODX1Kaqs427G+1Zh5PnV/EICAQDk+aJKqEswZDR4NCQiFPYhPdTNjAsUCs7qR4berZwxAAdS7tvZ2o3oiiki20uud0imjN
hULp0okVKWcATOUm66IVw/3IWinmb70vKOzc5e+Hl2Htw3HMbF7Cvc1+xx/QYFE8IXUJ9yIQ5C03BYARqi++aB5r63DP4VGj5Qb67nHEzA9IM9zjIaFILIE/7wV1wbyEX6vPCP7Ruitp+
L/6IewxGP0HgClT+ew09rTCcB4jaB9FpZvGewqPI2O0VJ3q7n0pwgHXv8nTWdNnqZDTQ8r0ulDTQdQDnxkNA+x10Oq5kX90QBpS2dzeb0uQ0ho9KOh+GSnNeIzV6kWxID+eF7t+Kv5QyZ
HUmsmGJ0sXRpVsILsMqd00DN53Lcr27Lu0Y6J7cqQgQl/mXsek1NNUTpmfS3AO8lIKtRe0DwRhB1jbx5qiL2L0qRe+RZkfkLoXsG27GalzvTs58Let6IDjIh4fsCdUHM= ubuntu@work
er1
ubuntu@worker1:~$
```

- Navigated to the server worker2 and opened authorized keys file and pasted ssh key generated in worker 1 to establish connection between the two

Similarly same steps performed in worker 3 and worker4

```
  GNU nano 6.2                                          ./.ssh/authorized_keys *
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCWFKI/ORqXVvGT4Pdb/6/VH4BmPA9uryj2vGHFYASYrhbo0V+2s1n0btYcRcAaD
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCxVi0IcrTODX1Kaqs427G+1Zh5PnV/EICAQDk+aJKqEswZDR4NQiFPYhPdTNjA

^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-0b31db3012c176c15 (worker2)

PublicIPs: 18.116.61.112 PrivateIPs: 172.31.20.162

Navigated back to the main server ie., worker1 and created a file named inventory and added private IPs of the worker 2 3 & 4 servers

- Choosen worker3 as a kubernates master and jenkins slave node
- Remaining two servers as kubernates slaves

```
  GNU nano 6.2                                                  inventory
[k8Master]
worker3 ansible_host=172.31.20.131
[k8slaves]
worker2 ansible_host=172.31.20.162
worker4 ansible_host=172.31.19.234

                                                  [ Read 5 lines ]
^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-
```

i-00b8547948a9cfa59 (Worker1)

PublicIPs: 3.141.13.91 PrivateIPs: 172.31.26.221

- With that ansible cluster setup is completed. Ensured connection between all servers by perfoming ansible ping

```
ubuntu@worker1:~$ sudo nano inventory
ubuntu@worker1:~$ sudo nano inventory
ubuntu@worker1:~$ ansible -i inventory all -m ping
The authenticity of host '172.31.20.131 (172.31.20.131)' can't be established.
ED25519 key fingerprint is SHA256:xlB5RiGAGABPN0R0DbVSz2KIO59if0V27dyio4nbkU0.
This key is not known by any other names
```

```
The authenticity of host '172.31.19.234 (172.31.19.234)' can't be established.
ED25519 key fingerprint is SHA256:Zkel/1vl4HezAbYz0UxAClDr8T0CtPQfBwmuDVsGypA.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? worker2 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
yes
worker3 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
yes
worker4 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
ubuntu@worker1:~$
```

Configuration deployments

- After that a shell script named localhost.sh is created to install java and jenkins in localhost(worker1)

```
  GNU nano 6.2                                                    localhost.sh
sudo apt update
sudo apt install openjdk-11-jdk -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-get update
sudo apt-get install jenkins -y
```

- Similarly another shellscript worker3.sh is created to install java docker and kubernates in worker3

```
  GNU nano 6.2                                          worker3.sh
sudo apt update
sudo apt install openjdk-11-jdk -y
sudo apt update
sudo apt install docker.io -y
sudo apt update -y

sudo apt install curl apt-transport-https -y
curl -fsSL  https://packages.cloud.google.com/apt/doc/apt-key.gpg|sudo gpg --dearmor -o /etc/apt/trusted.gpg.d/k8s.gpg
echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list

sudo apt update -y
sudo apt install kubelet kubeadm kubectl -y
sudo apt-mark hold kubelet kubeadm kubectl

sudo swapoff -a
sudo sed -i '/ swap / s/^(.*)$/#\1/g' /etc/fstab
sudo tee /etc/modules-load.d/k8s.conf <<EOF
                                                [ Read 46 lines ]
^G Help      ^O Write Out   ^W Where Is    ^K Cut       ^T Execute     ^C Location    M-U Undo      M-A Set Mark   M-] To Bracket
^X Exit      ^R Read File   ^\ Replace     ^U Paste     ^J Justify     ^/ Go To Line  M-E Redo      M-6 Copy       ^Q Where Was
```

- Finally last shellscript is created to install docker and kubernates on worker2 and worker4

```
  GNU nano 6.2                                          slaves.sh *
sudo apt update
sudo apt install docker.io -y
sudo apt update -y

sudo apt install curl apt-transport-https -y
curl -fsSL  https://packages.cloud.google.com/apt/doc/apt-key.gpg|sudo gpg --dearmor -o /etc/apt/trusted.gpg.d/k8s.gpg
echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list

sudo apt update -y
sudo apt install kubelet kubeadm kubectl -y
sudo apt-mark hold kubelet kubeadm kubectl

sudo swapoff -a
sudo sed -i '/ swap / s/^(.*)$/#\1/g' /etc/fstab

sudo tee /etc/modules-load.d/k8s.conf <<EOF
overlay
```

- An ansible playbook is created to ececute these shellscripts in their respective servers/hosts

```
---
- hosts: localhost
  name: installing jenkins and java
  become: yes
  tasks:
    - name: executing localhost.sh
      script: localhost.sh
- hosts: k8Master
  name: installing java docker and kubernates
  become: yes
  tasks:
    - name: executing worker3.sh
      script: worker3.sh
- hosts: k8slaves
  name: installing docker and kubernates
  become: yes
  tasks:
    - name: executing slaves.sh
      script: slaves.sh
~
"playbook.yml" 19L, 427B
```

- The playbook is executed successfully resulting the installations in the target host servers

```
        "No VM guests are running outdated hypervisor (qemu) binaries on this host.",
        "kubelet set on hold.",
        "kubeadm set on hold.",
        "kubectl set on hold.",
        "net.bridge.bridge-nf-call-iptables=1",
        "",
        "net.bridge.bridge-nf-call-iptables = 1",
        "Synchronizing state of docker.service with SysV service script with /lib/systemd/systemd-sysv-install.",
        "Executing: /lib/systemd/systemd-sysv-install enable docker"
    ]
}
META: ran handlers
META: ran handlers

PLAY RECAP *********************************************************************************************************************
localhost                  : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
worker2                    : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
worker3                    : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
worker4                    : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

ubuntu@worker1:~$
```

Kubernates cluster configuration

Navigated to the kubernates master and initialised kubeadm

```
ubuntu@ip-172-31-20-131:~$ sudo kubeadm init --apiserver-advertise-address=172.31.20.131 --pod-network-cidr=10.244.0.0/16
I0107 06:18:34.581405   50266 version.go:256] remote version is much newer: v1.29.0; falling back to: stable-1.28
[init] Using Kubernetes version: v1.28.5
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
```

Generated kubeadm token is copied

```
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 172.31.20.131:6443 --token fe9ej1.gd40acezkaqa0st9 \
        --discovery-token-ca-cert-hash sha256:7428b54a21823b234b8af394bdb4863597695b345ae2e4655f5773731f1ff59d
ubuntu@ip-172-31-20-131:~$
```

kubeadm join 172.31.20.131:6443 --token fe9ej1.gd40acezkaqa0st9 \

--discovery-token-ca-cert-hash sha256:7428b54a21823b234b8af394bdb4863597695b345ae2e4655f5773731f1ff59d

- Kubeadm token is ran in the kubernates slaves joining them to the cluster
- Kubernates cluster is successfully configured

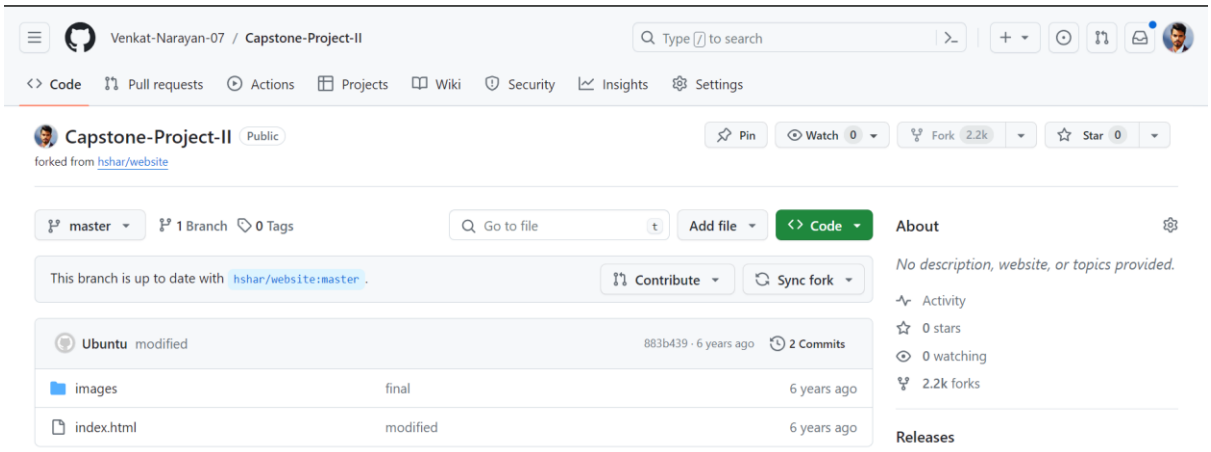
```
ubuntu@ip-172-31-20-131:~$ kubectl get nodes
NAME                STATUS   ROLES           AGE     VERSION
ip-172-31-19-234    Ready    <none>          3m59s   v1.28.2
ip-172-31-20-131    Ready    control-plane   26m     v1.28.2
ip-172-31-20-162    Ready    <none>          3m56s   v1.28.2
ubuntu@ip-172-31-20-131:~$
```

i-07edd73dde8245ebf (worker3)

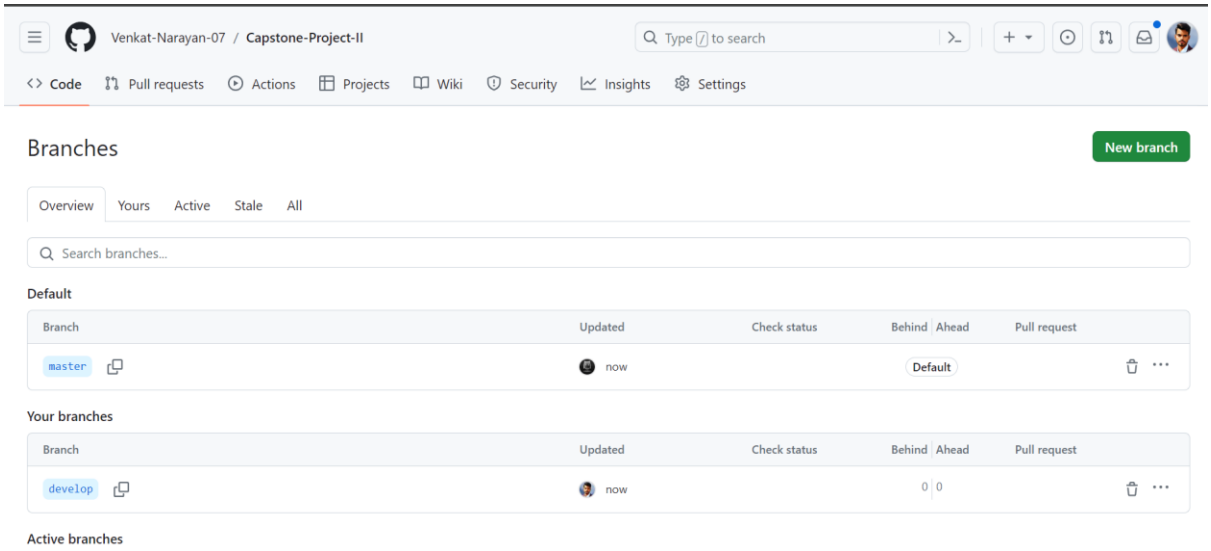
PublicIPs: 13.58.198.22 PrivateIPs: 172.31.20.131

Github configuration

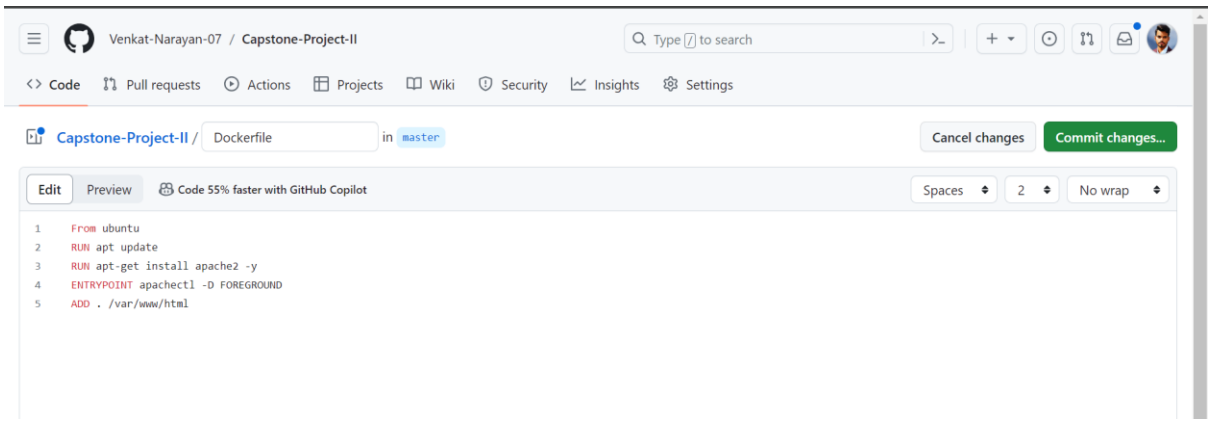
- Then navigated to the git repo provided and forked to my github account



- Created a new branch named develop



- Then created a new file named dockerfile to copy the entire repo and create a custom image out of it



- Similarly created a new file named hshar-deployment.yml to deploy hshar webpage from the image created by dockerfile with 3 replicas and create and attach nodeport service on port 30008

Venkat-Narayan-07 Update hshar-deployment.yml

Code Blame 21 lines (21 loc) · 352 Bytes Code 55% faster with GitHub Copilot

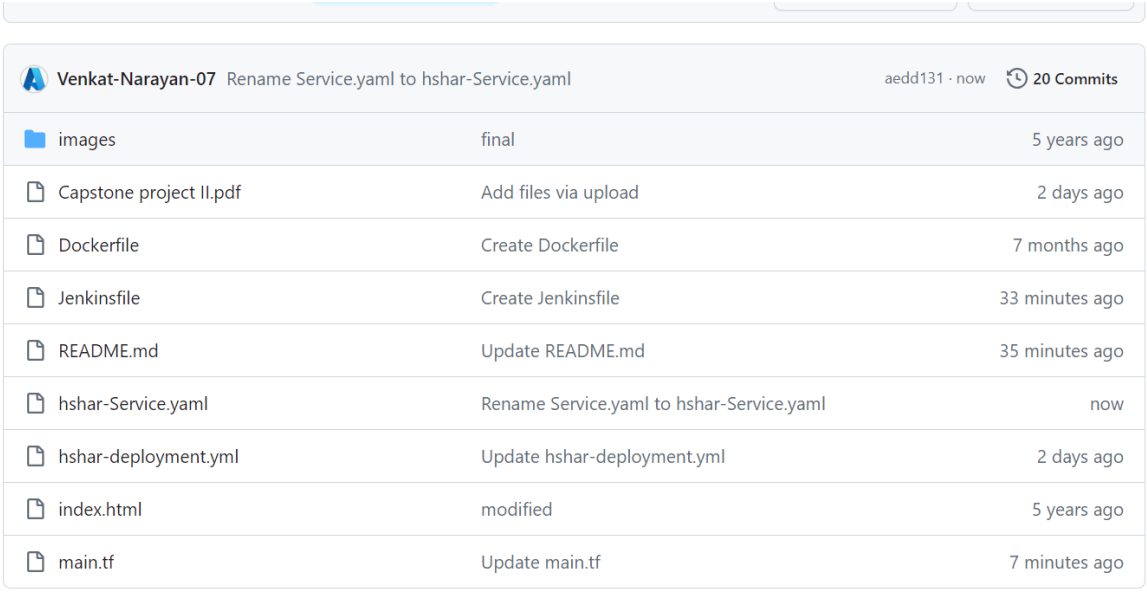
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hshar-deployment
  labels:
    web: hshar
spec:
  replicas: 1
  selector:
    matchLabels:
      web: hshar
  template:
    metadata:
      labels:
        web: hshar
    spec:
      containers:
      - name: hshar
        image: venkatnarayan16/hshar
        ports:
          - containerPort: 80
```

Capstone-Project-II / hshar-Service.yaml in master

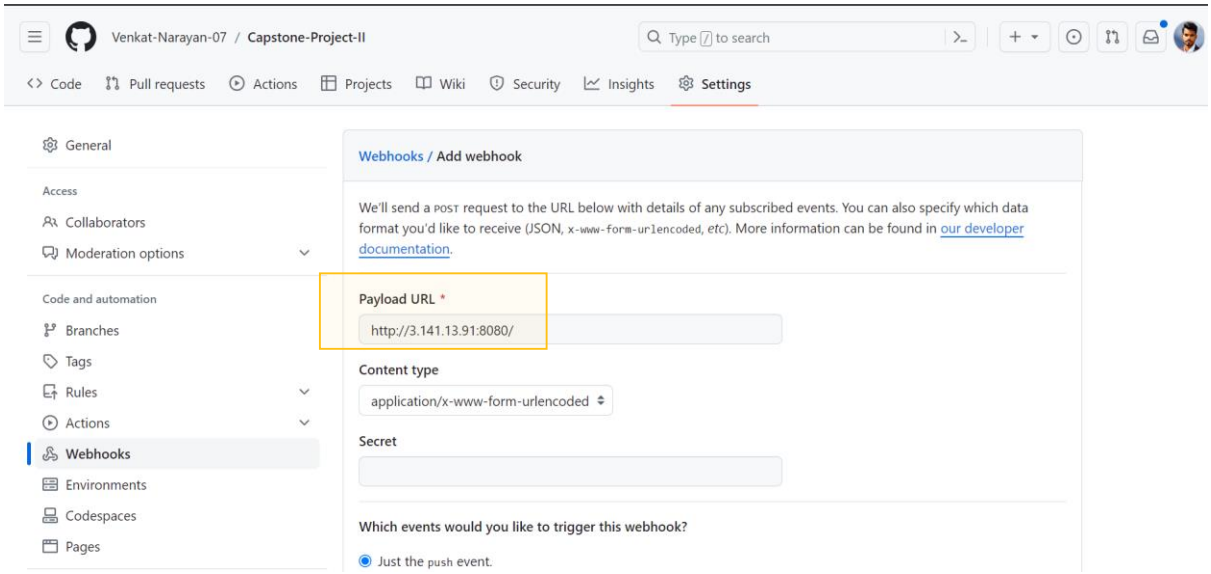
Edit Preview Code 55% faster with GitHub Copilot

```yaml
apiVersion: v1
kind: Service
metadata:
  name: hshar-service
spec:
  type: NodePort
  selector:
    web: hshar
  ports:
    - port: 80
      nodePort: 30008
```

- With that our git repo is ready

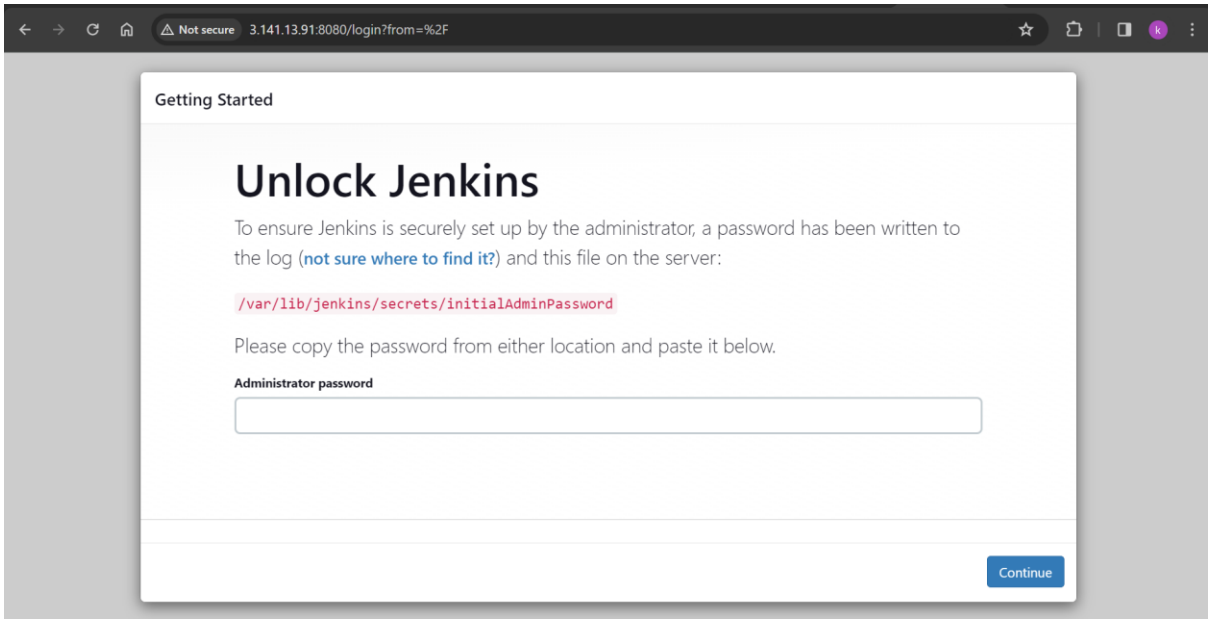


- Finally in the repo settings, a webhook is created to get triggered whenever there is a push to the repo



Jenkins configuration

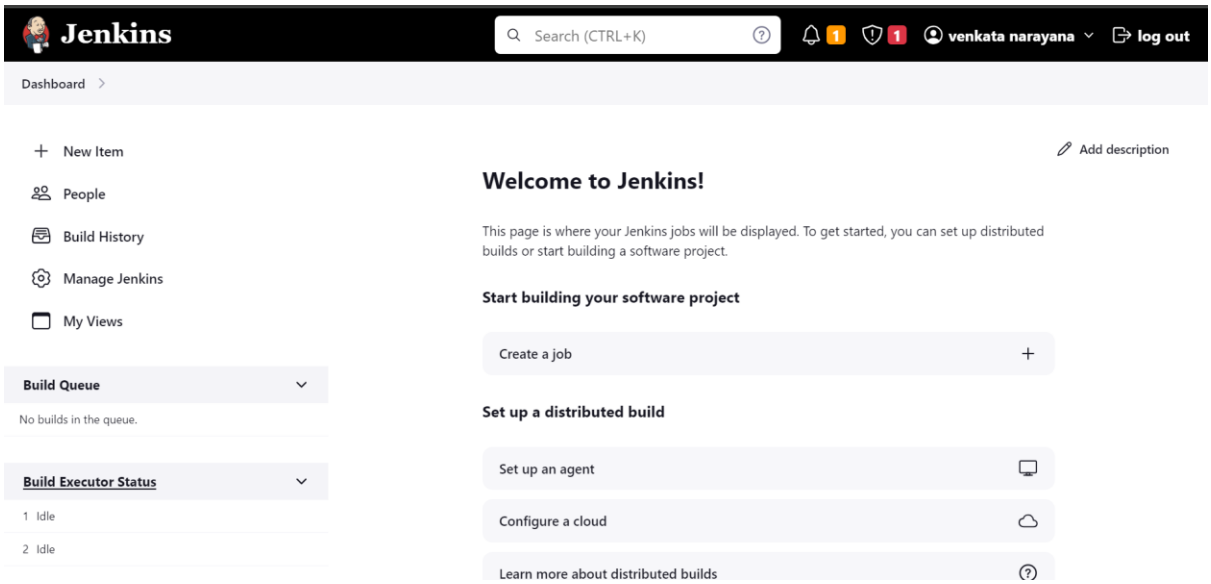
- Afterthat navigated to the jenkins dashboard and signed in



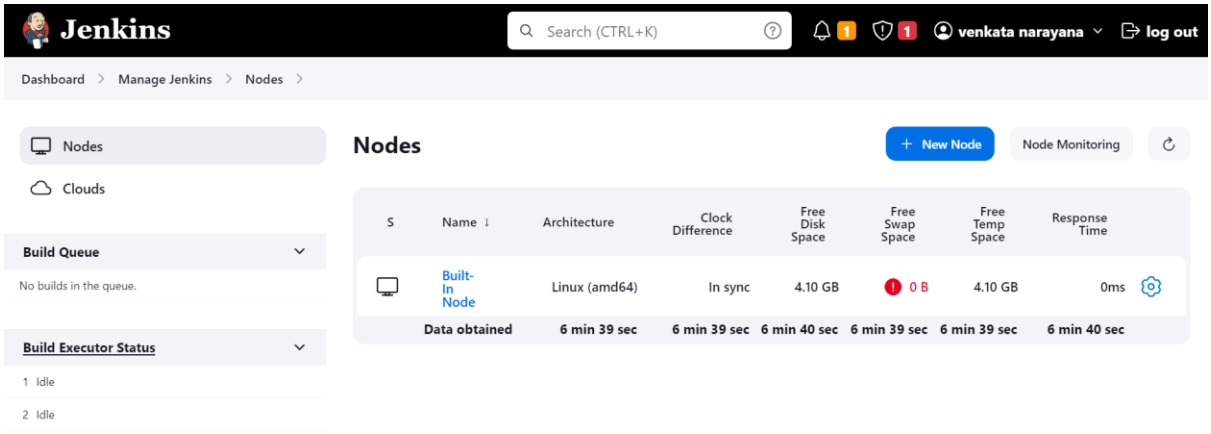

```
ubuntu@worker1:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
1104584b404c4f5998b0fa1460455790
ubuntu@worker1:~$
```

i-00b8547948a9cfa59 (Worker1)

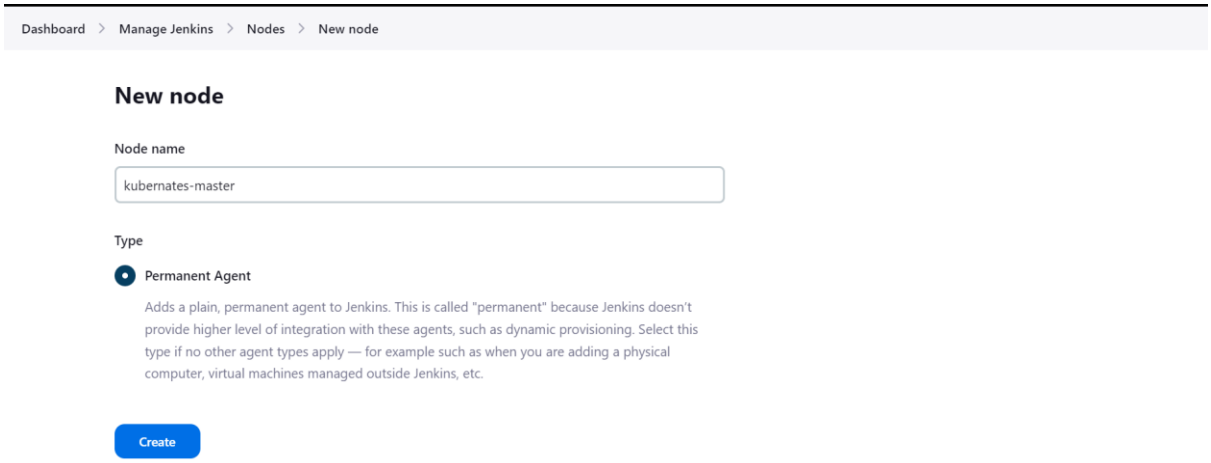
PublicIPs: 3.141.13.91 PrivateIPs: 172.31.26.221



- In the jenkins dashboard navigated to manage jenkins and nodes and added a new node



- Named the node as kubernates master and provided private IP of the worker3 server



Dashboard > Manage Jenkins > Nodes >

Name ?

kubernates-master

Description ?

Plain text Preview

Number of executors ?

1

Remote root directory ?

/home/ubuntu/jenkins

Labels ?

Usage ?

Use this node as much as possible

Launch method ?

Launch agents via SSH

Host ?

172.31.20.131

Credentials ?

ubuntu

+ Add

Host Key Verification Strategy ?

Non verifying Verification Strategy ?

Advanced

Availability ?

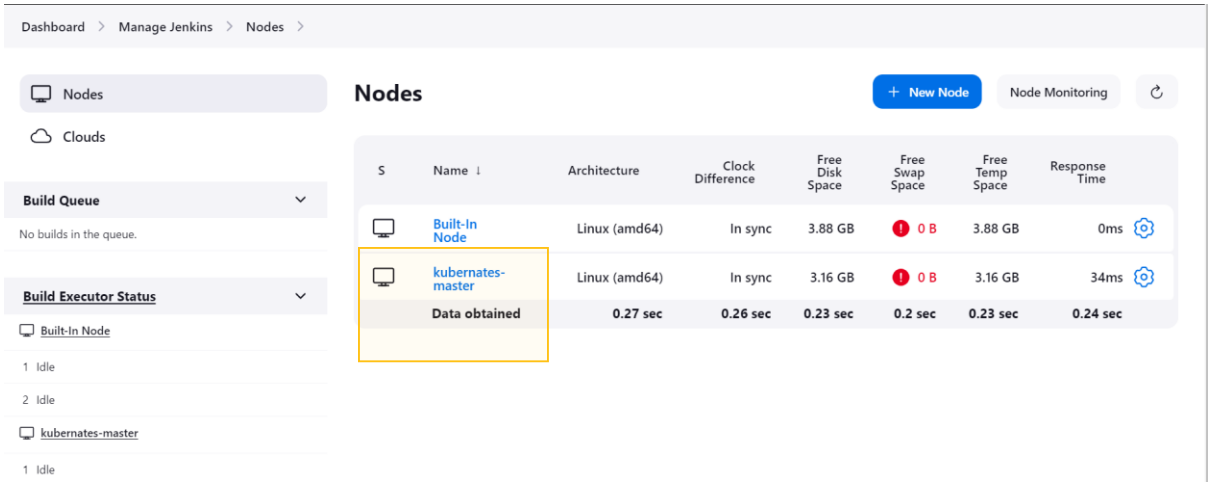
Keep this agent online as much as possible ?

Node Properties

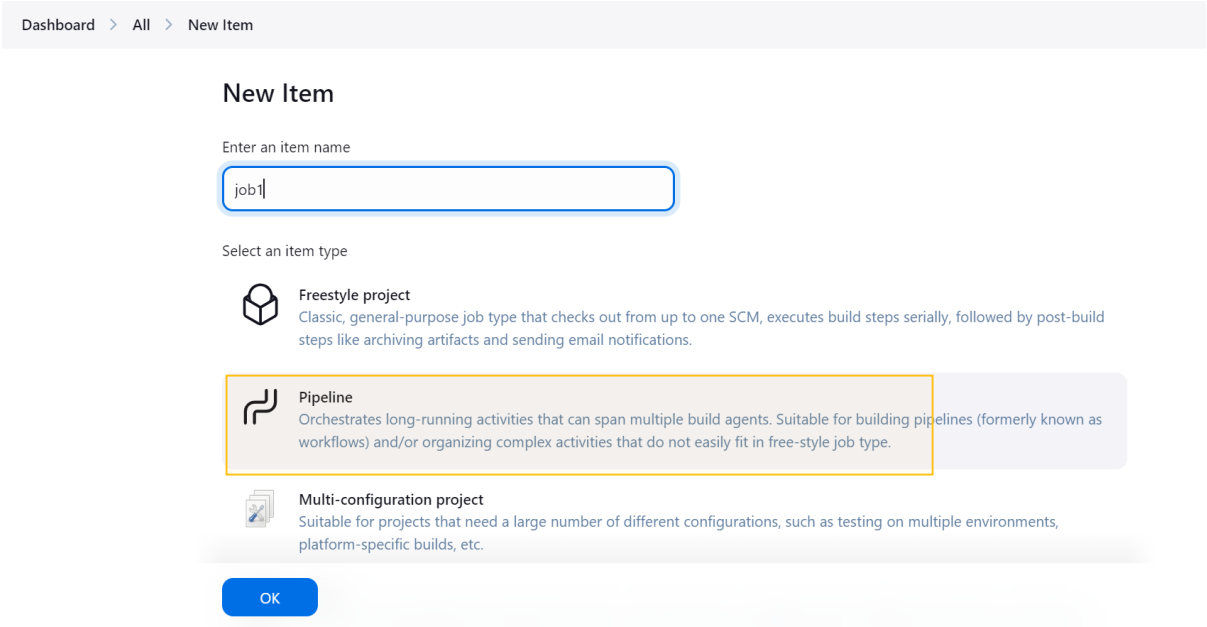
Disable deferred wipeout on this node ?

Save

- Worker3 is successfully added as a jenkins node



- Then created a job1



- Job1 is configured to the worker3 and git repo and the master branch is specified in source code management.
- Enabled webhook to trigger build whenever there is a push to master branch

Dashboard > job1 > Configuration

Configure

- General
- Advanced Project Options
- Pipeline

General

Enabled

Description

End-to-end Jenkins CI/CD pipeline using Docker and Kubernates

Plain text Preview

- [ ] Discard old builds ?
- [ ] Do not allow concurrent builds
- [ ] Do not allow the pipeline to resume if the controller restarts
- [ ] GitHub project
- [ ] Pipeline speed/durability override ?

Configure

- General
- Advanced Project Options
- Pipeline

Build Triggers

- [ ] Build after other projects are built ?
- [ ] Build periodically ?
- [x] GitHub hook trigger for GITScm polling ?
- [ ] Poll SCM ?
- [ ] Quiet period ?
- [ ] Trigger builds remotely (e.g., from scripts) ?

Advanced Project Options

Advanced

Pipeline

Configure

- General
- Advanced Project Options
- Pipeline

Pipeline script from SCM

SCM ?

Git ?

Repositories ?

Repository URL ?

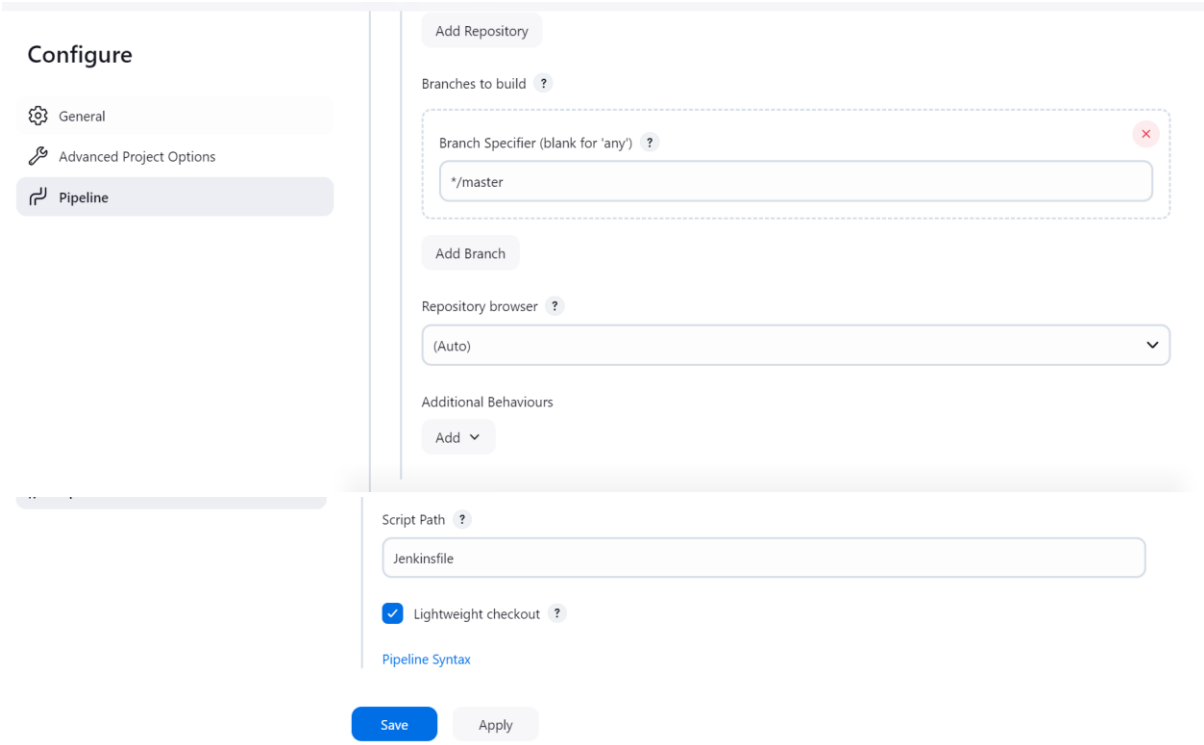
https://github.com/Venkat-Narayan-07/Capstone-Project-II.git

Credentials ?

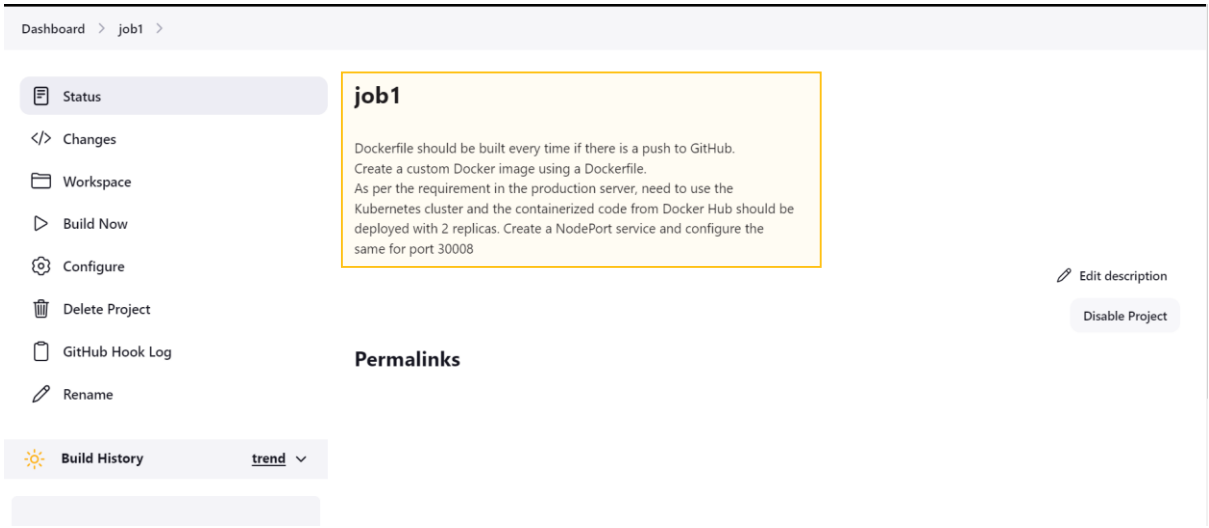
ubuntu

+ Add

Advanced



- Saved the job1 to the jenkins dashboard
- Then manually ran builtnow to test the job.



- Job1 is successfully completed built operation

Changes
Console Output
Edit Build Information
Delete build '#2'
Timings
Git Build Data
Pipeline Overview
Pipeline Console
Restart from Stage
Replay
Pipeline Steps
Workspaces
Previous Build

```
Started by user venkat
Obtained Jenkinsfile from git https://github.com/Venkat-Narayan-07/Capstone-Project-II.git
[Pipeline] Start of Pipeline
[Pipeline] withCredentials
Masking supported pattern matches of $DOCKERHUB_CREDENTIALS or $DOCKERHUB_CREDENTIALS_PSW
[Pipeline] {
[Pipeline] stage
[Pipeline] { (git)
[Pipeline] node
Running on K8master in /home/ubuntu/jenkins/workspace/job1
[Pipeline] {
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
using credential d0fe9a57-068a-40a6-a956-da9c9eebd01b
Fetching changes from the remote Git repository
Checking out Revision f0366e8a941f6431578e03b754f060392417c288 (refs/remotes/origin/master)
Commit message: "Update Jenkinsfile"
 > git rev-parse --resolve-git-dir /home/ubuntu/jenkins/workspace/job1/.git # timeout=10
 > git config remote.origin.url https://github.com/Venkat-Narayan-07/Capstone-Project-II.git # timeout=10
Fetching upstream changes from https://github.com/Venkat-Narayan-07/Capstone-Project-II.git
 > git --version # timeout=10
```

```
 > git --version # 'git version 2.34.1'
using GIT_SSH to set credentials
Verifying host key using known hosts file
You're using 'Known hosts file' strategy to verify ssh host keys, but your known_hosts file does not exist,
please go to 'Manage Jenkins' -> 'Security' -> 'Git Host Key Verification Configuration' and configure host
key verification.
 > git fetch --tags --force --progress -- https://github.com/Venkat-Narayan-07/Capstone-Project-II.git
+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 f0366e8a941f6431578e03b754f060392417c288 # timeout=10
 > git rev-list --no-walk 9b9872b0b7643e67c98cf71affda75fae018be30 # timeout=10
[Pipeline] withEnv
[Pipeline] {
[Pipeline] script
[Pipeline] {
[Pipeline] git
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
Fetching changes from the remote Git repository
Checking out Revision f0366e8a941f6431578e03b754f060392417c288 (refs/remotes/origin/master)
Commit message: "Update Jenkinsfile"
[Pipeline] }
```

```
[Pipeline] // script
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (docker)
[Pipeline] node
Running on K8master in /home/ubuntu/jenkins/workspace/job1
[Pipeline] {
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
using credential d0fe9a57-068a-40a6-a956-da9c9eebd01b
Fetching changes from the remote Git repository
 > git rev-parse --resolve-git-dir /home/ubuntu/jenkins/workspace/job1/.git # timeout=10
 > git config remote.origin.url https://github.com/Venkat-Narayan-07/Capstone-Project-II.git # timeout=10
Fetching upstream changes from https://github.com/Venkat-Narayan-07/Capstone-Project-II.git
 > git --version # timeout=10
 > git --version # 'git version 2.34.1'
 > git fetch --tags --force --progress -- https://github.com/Venkat-Narayan-07/Capstone-Project-II.git
+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 f0366e8a941f6431578e03b754f060392417c288 # timeout=10
 > git branch -a -v --no-abbrev # timeout=10
 > git branch -D master # timeout=10
 > git checkout -b master f0366e8a941f6431578e03b754f060392417c288 # timeout=10
 > git rev-parse --resolve-git-dir /home/ubuntu/jenkins/workspace/job1/.git # timeout=10
 > git config remote.origin.url https://github.com/Venkat-Narayan-07/Capstone-Project-II.git # timeout=10
Fetching upstream changes from https://github.com/Venkat-Narayan-07/Capstone-Project-II.git
 > git --version # timeout=10
 > git --version # 'git version 2.34.1'
using GIT_SSH to set credentials
Verifying host key using known hosts file
You're using 'Known hosts file' strategy to verify ssh host keys, but your known_hosts file does not exist,
please go to 'Manage Jenkins' -> 'Security' -> 'Git Host Key Verification Configuration' and configure host
key verification.
 > git fetch --tags --force --progress -- https://github.com/Venkat-Narayan-07/Capstone-Project-II.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
Checking out Revision f0366e8a941f6431578e03b754f060392417c288 (refs/remotes/origin/master)
Commit message: "Update Jenkinsfile"
[Pipeline] withEnv
[Pipeline] {
[Pipeline] script
[Pipeline] {
```

```
[Pipeline] sh
+ sudo docker build /home/ubuntu/jenkins/workspace/prtwebsite/ -t venkatnarayan16/hshar
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  146.9kB

Step 1/5 : From ubuntu
 ---> 35a88802559d
Step 2/5 : RUN apt update
 ---> Using cache
 ---> 402de2009e6d
Step 3/5 : RUN apt-get install apache2 -y
 ---> Using cache
 ---> bcd2c3035568
Step 4/5 : ENTRYPOINT apachectl -D FOREGROUND
 ---> Using cache
 ---> 7be6d0ba8b1c
Step 5/5 : ADD . /var/www/html
 ---> Using cache
 ---> 0acb334df88d
Successfully built 0acb334df88d
Successfully tagged venkatnarayan16/hshar:latest
```

```
Successfully tagged venkatnarayan16/hshar:latest
[Pipeline] sh
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
 > git config core.sparsecheckout # timeout=10
 > git checkout -f f0366e8a941f6431578e03b754f060392417c288 # timeout=10
+ sudo echo ****
+ sudo docker login -u venkatnarayan16 --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 venkatnarayan16/hshar
Using default tag: latest
The push refers to repository [docker.io/venkatnarayan16/hshar]
2f46104f539a: Preparing
2f71b4f6ef9f: Preparing
325d33a349f4: Preparing
a30a5965a4f7: Preparing
325d33a349f4: Layer already exists
2f71b4f6ef9f: Layer already exists
a30a5965a4f7: Layer already exists
2f46104f539a: Layer already exists
```

```
latest: digest: sha256:14bc098d76d4a3b69ec7b5e5877ef5468245f335103d354898a0003874034a05 size: 1162
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (k8s)
[Pipeline] node
Running on K8master in /home/ubuntu/jenkins/workspace/job1
[Pipeline] {
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
using credential d0fe9a57-068a-40a6-a956-da9c9eebd01b
Fetching changes from the remote Git repository
Checking out Revision f0366e8a941f6431578e03b754f060392417c288 (refs/remotes/origin/master)
Commit message: "Update Jenkinsfile"
[Pipeline] withEnv
[Pipeline] {
[Pipeline] script
```

```
[Pipeline] {
[Pipeline] sh
+ kubectl apply -f hshar-deployment.yml
deployment.apps/hshar-deployment unchanged
[Pipeline] sh
 > git rev-parse --resolve-git-dir /home/ubuntu/jenkins/workspace/job1/.git # timeout=10
 > git config remote.origin.url https://github.com/Venkat-Narayan-07/Capstone-Project-II.git # timeout=10
Fetching upstream changes from https://github.com/Venkat-Narayan-07/Capstone-Project-II.git
 > git --version # timeout=10
 > git --version # 'git version 2.34.1'
using GIT_SSH to set credentials
Verifying host key using known hosts file
You're using 'Known hosts file' strategy to verify ssh host keys, but your known_hosts file does not exist,
please go to 'Manage Jenkins' -> 'Security' -> 'Git Host Key Verification Configuration' and configure host
key verification.
 > git fetch --tags --force --progress -- https://github.com/Venkat-Narayan-07/Capstone-Project-II.git
+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 f0366e8a941f6431578e03b754f060392417c288 # timeout=10
+ kubectl apply -f hshar-Service.yaml
service/hshar-service unchanged
[Pipeline] }
```

```
[Pipeline] // script
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withCredentials
[Pipeline] End of Pipeline
Finished: SUCCESS
```

- Whenever there is a push made to the master branch, Jenkins job gets triggered and automatically containerizes the code and deploys replicas of the webpage using docker and Kubernetes respectively.

- Now with the k8s slave machines Ip and on port 30008 we can access the webpage