

Project2

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

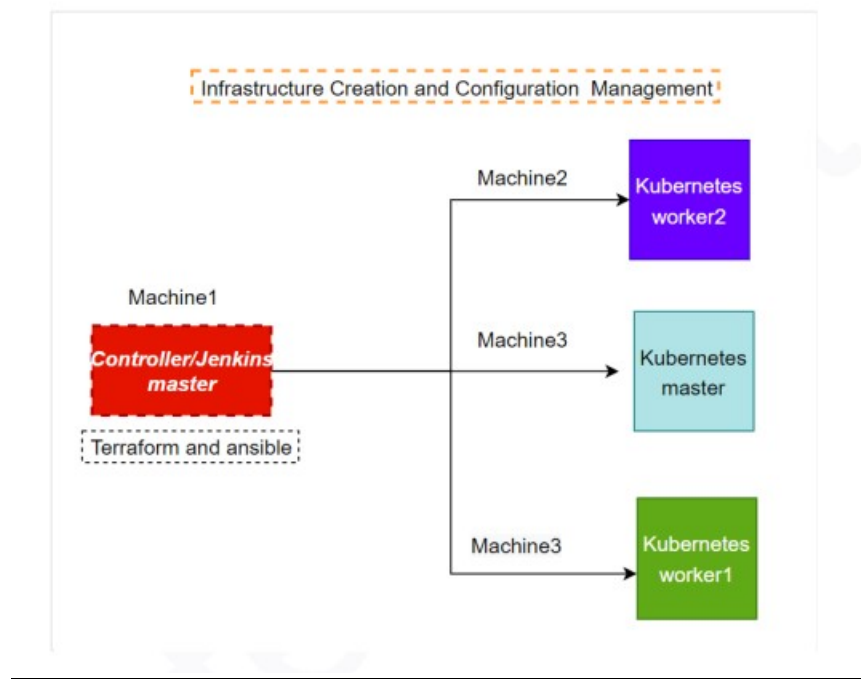
Softwares to be installed on the respective machines using configuration management.

Worker1: Jenkins, Java

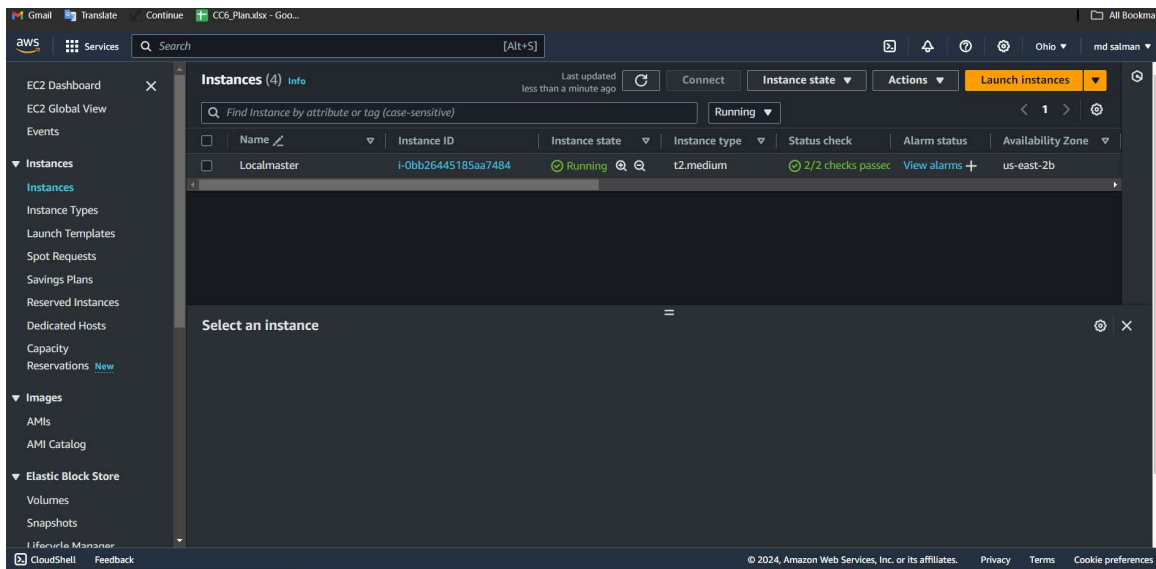
Worker2: Docker, Kubernetes

Worker3: Java, Docker, Kubernetes

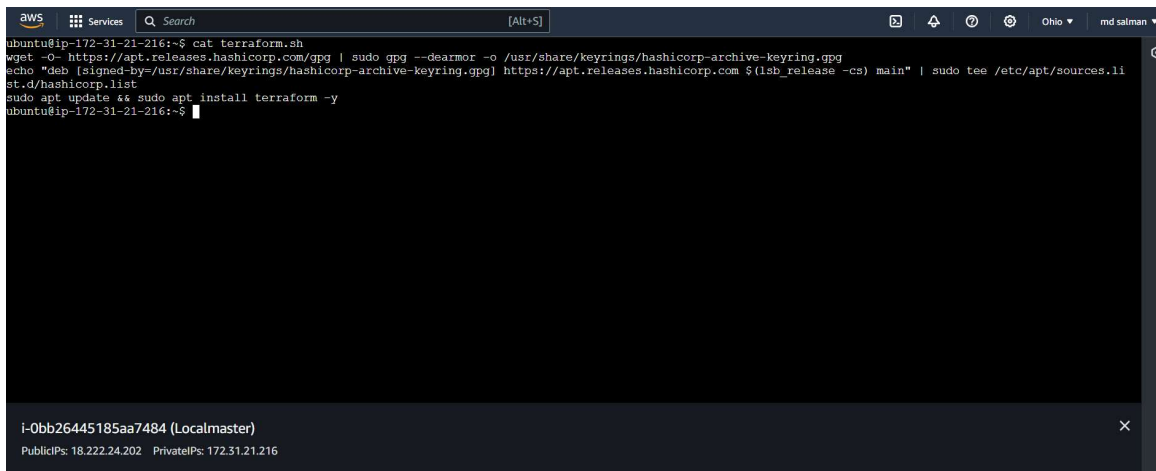
Worker4: Docker, Kubernetes



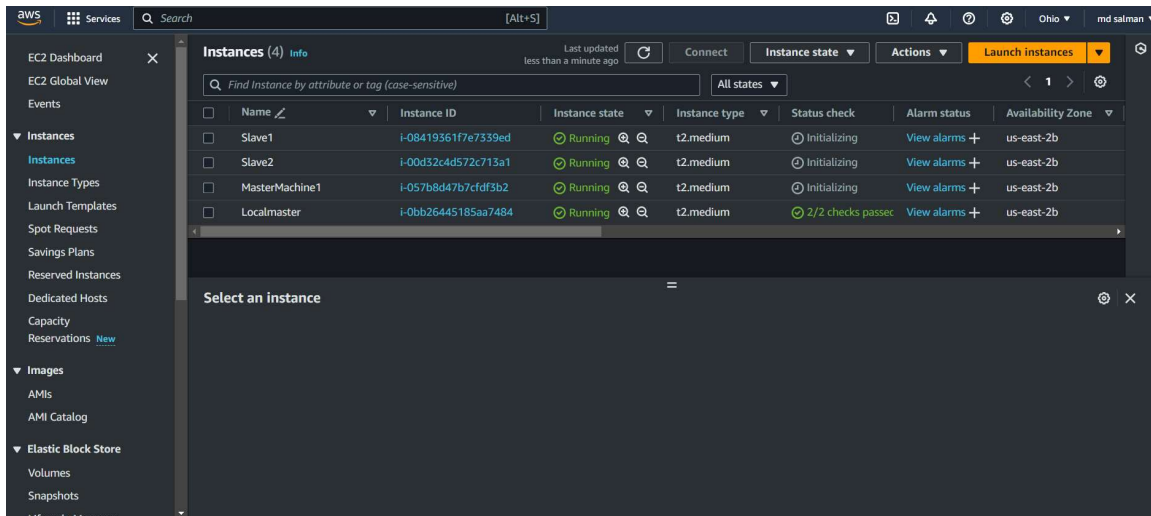
Step1: I have launched the ubuntu (OS) as local master which is my worker1



Step2: I have installed terraform and launched other 3 worker machines with the help of terraform.



As you can see my 3 workers machine up and running. I named them worker2:Master1 and worker3:slave1 and worker4:slave2.



Step3: I have installed ansible and java in local master machine which is worker1. In order to make connection with other worker nodes.

```
ubuntu@ip-172-31-21-216:~$ ansible --version
ansible [core 2.16.10]
  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, Jul 29 2024, 16:56:48) [GCC 11.4.0] (/usr/bin/python3)
  jinja version = 3.0.3
  libyaml = True
ubuntu@ip-172-31-21-216:~$ java --version
openjdk 17.0.12 2024-07-16
OpenJDK Runtime Environment (build 17.0.12+7-Ubuntu-1ubuntu222.04)
OpenJDK 64-Bit Server VM (build 17.0.12+7-Ubuntu-1ubuntu222.04, mixed mode, sharing)
ubuntu@ip-172-31-21-216:~$
```

i-0bb26445185aa7484 (Localmaster)
PublicIPs: 3.21.19.132 PrivateIPs: 172.31.21.216

Step4: ssh key-gen in Local master worker1 and nodes ip address attachment in ansible hosts and also

```
ubuntu@ip-172-31-21-216:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):
/home/ubuntu/.ssh/id_rsa already exists.
Overwrite (y/n)? y
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:+4km5K6YgS6bQnpwqs61xn19J2bt1Zb1+WEPSDAmgnE ubuntu@ip-172-31-21-216
The key's randomart image is:
+---[RSA 3072]-----+
|
| . E
| +
| . . S + . .
|.O. . . + o o *|
|++o..o o o .*+|
|B.o*..O..+ * ooo+|
|Q*+..ooo. * +. o|
+---[SHA256]-----+
ubuntu@ip-172-31-21-216:~$
```

i-0bb26445185aa7484 (Localmaster)
PublicIPs: 18.222.24.202 PrivateIPs: 172.31.21.216

```
GNU nano 6.2 hosts
[master]
172.31.17.247
[slaves]
172.31.22.177
172.31.30.92

# This is the default ansible 'hosts' file.
#
# It should live in /etc/ansible/hosts
#
# - Comments begin with the '#' character

[ Read 72 lines ]
^C Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^G Location   M-U Undo      M-A S
^X Exit      ^R Read File  ^_ Replace    ^U Paste      ^J Justify    ^/ Go To Line  M-E Redo      M-C C
```

i-0bb26445185aa7484 (Localmaster)
PublicIPs: 18.222.24.202 PrivateIPs: 172.31.21.216

Step5:And pasted the public key of local master in other worker nodes.

```
ubuntu@ip-172-31-21-216:~/.ssh$ ls
authorized_keys  id_rsa  id_rsa.pub  known_hosts  known_hosts.old
ubuntu@ip-172-31-21-216:~/.ssh$ id_rsa.pub
id_rsa.pub: command not found
ubuntu@ip-172-31-21-216:~/.ssh$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgQCvKzIDHxmTw8UWf3OG5tmWMybIzpo6/8iFz8qhwoLKgHaqPWRHCw9rgd512Ky/Q+QWbAKSpJNmva6Fka3Y4OLgeiREsLFA
bzBFYPD4mf+ZMpEmMF306dAQk0mKF8nDFH2CIEd/5qciLUOUFMEBKP8zCsGGmf2beKD1Snu5Wa8YYN1BkeVvoEvDvJQJFmuZrKLe1XtEYaL9F6d0Cav6Lp/HCVrHH21q7sue
LF1C9hjvg12xE9CwNoR0TY6k0D4YmXnjS/EKsdCBGj+cUNhCWx7oGQrwnxLmG5MdyTTWFwSE3RRqwt61Y/fEbOCczLr4GYCCdvKQ3AkmsnE5RE41ED63rhewH5acqVR/tS
3wBcBG4+chUM8GGpbtFEi6Cp+uaZqE1ThJBQfF/o2GSM1lpz3DM7VcCV8Bc= ubuntu@ip-172-31-21-216
ubuntu@ip-172-31-21-216:~/.ssh$
```

i-Obb26445185aa7484 (Localmaster)
PublicIPs: 18.222.24.202 PrivateIPs: 172.31.21.216

And as you can see we successfully able to make the connections with other Machines.

```
ubuntu@ip-172-31-21-216:/etc/ansible$ ansible -m ping all
172.31.22.177 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
172.31.30.92 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
172.31.17.247 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
ubuntu@ip-172-31-21-216:/etc/ansible$
```

i-0bb26445185aa7484 (Localmaster)

PublicIPs: 18.222.24.202 PrivateIPs: 172.31.21.216

Step6: And now I am gonna write the playbook and install the software packages in their servers.

```
ubuntu@ip-172-31-21-216:/etc/ansible$ cat playbook.yml
---

- name: installing jenkins,java,kubernetes in master1
  become: true
  hosts: localhost
  tasks:
    - name: run script1
      script: script1.sh
- name: installing java,kubernetes in master1
  become: true
  hosts: master
  tasks:
    - name: run script2
      script: script2.sh
- name: installing kubernetes in slaves1 and 2
  become: true
  hosts: slaves
  tasks:
    - name: run script3
      script: script3.sh
ubuntu@ip-172-31-21-216:/etc/ansible$
```

i-0bb26445185aa7484 (Localmaster)

PublicIPs: 18.222.24.202 PrivateIPs: 172.31.21.216

Step7:In playbook I am gonna write 3 scripts In script1 we have jenkins and java and kubernetes. Script 1 is for local master which is my worker1.


```

ubuntu@ip-172-31-21-216:/etc/ansible$ cat script1.sh
sudo apt update
sudo apt install openjdk-17-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

sudo swapoff -a

# 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

## Install CRI-O Runtime
sudo apt-get update -y
sudo apt-get install -y software-properties-common curl apt-transport-https ca-certificates gpg

sudo curl -fsSL https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/cri-o-apt-keyring.gpg
echo "deb [signed-by=/etc/apt/keyrings/cri-o-apt-keyring.gpg] https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/ /*" | sudo tee /etc/apt/sources.list.d/cri-o.list

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

sudo systemctl daemon-reload
sudo systemctl enable cri-o --now
sudo systemctl start cri-o.service

```

Script2 is for master1 which is my worker2. java, and kubernetes

```

ubuntu@ip-172-31-21-216:/etc/ansible$ cat script2.sh
sudo apt update
sudo apt install openjdk-17-jdk -y
sudo swapoff -a

# 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

## Install CRI-O Runtime
sudo apt-get update -y
sudo apt-get install -y software-properties-common curl apt-transport-https ca-certificates gpg

sudo curl -fsSL https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/cri-o-apt-keyring.gpg
echo "deb [signed-by=/etc/apt/keyrings/cri-o-apt-keyring.gpg] https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/ /*" | sudo tee /etc/apt/sources.list.d/cri-o.list

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

sudo systemctl daemon-reload
sudo systemctl enable cri-o --now
sudo systemctl start cri-o.service

```

Script3 is for slaves1 and 2 (kubernetes.)

```
ubuntu@ip-172-31-21-216:/etc/ansible$ cat script3.sh
sudo apt-get update

sudo swapoff -a

# 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

## Install CRI-O Runtime
sudo apt-get update -y
sudo apt-get install -y software-properties-common curl apt-transport-https ca-certificates gpg

i-0bb26445185aa7484 (Localmaster)
PublicPs: 18.222.24.202 PrivatePs: 172.31.21.216
```

As you can see we successfully installed the required software packages and dependencies.

```
ubuntu@ip-172-31-21-216:~$ jenkins --version
2.475
ubuntu@ip-172-31-21-216:~$ java --version
openjdk 17.0.12 2024-07-16
OpenJDK Runtime Environment (build 17.0.12+7-Ubuntu-1ubuntu222.04)
OpenJDK 64-Bit Server VM (build 17.0.12+7-Ubuntu-1ubuntu222.04, mixed mode, sharing)
ubuntu@ip-172-31-21-216:~$ kubectl version --client
Client Version: v1.29.0
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
ubuntu@ip-172-31-21-216:~$ █
```

i-0bb26445185aa7484 (Localmaster)

PublicIPs: 18.117.117.58 PrivateIPs: 172.31.21.216

```
ubuntu@ip-172-31-17-247:~$ java --version
openjdk 17.0.12 2024-07-16
OpenJDK Runtime Environment (build 17.0.12+7-Ubuntu-1ubuntu222.04)
OpenJDK 64-Bit Server VM (build 17.0.12+7-Ubuntu-1ubuntu222.04, mixed mode, sharing)
ubuntu@ip-172-31-17-247:~$ kubectl version --client
Client Version: v1.29.0
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
ubuntu@ip-172-31-17-247:~$ █
```

i-057b8d47b7cfd3b2 (MasterMachine1)

PublicIPs: 18.224.151.236 PrivateIPs: 172.31.17.247

```
ubuntu@ip-172-31-22-177:~$ kubectl version --client
Client Version: v1.29.0
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
ubuntu@ip-172-31-22-177:~$
```

i-08419361f7e7339ed (Slave1)

PublicIPs: 52.14.217.240 PrivateIPs: 172.31.22.177

```
ubuntu@ip-172-31-30-92:~$ kubectl version --client
Client Version: v1.29.0
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
ubuntu@ip-172-31-30-92:~$
```

i-00d32c4d572c713a1 (Slave2)

PublicIPs: 3.15.216.60 PrivateIPs: 172.31.30.92

Step8: I created node in jenkins and connected with master1 which is my worker2.

Dashboard > Manage Jenkins > Nodes >

Nodes

Clouds

Build Queue

No builds in the queue.

Build Executor Status

Built-In Node 0/2

Master1 0/1

Nodes

+ New Node

Configure Monitors



S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	2.77 GiB	 0 B	2.77 GiB	0ms 
	Master1	Linux (amd64)	In sync	2.27 GiB	 0 B	2.27 GiB	59ms 
Data obtained		37 min	37 min	37 min	37 min	37 min	37 min

Icons: S M L

Legend









```
ubuntu@ip-172-31-17-247:~$ ls
jenkins
ubuntu@ip-172-31-17-247:~$
```

i-057b8d47b7cfd3b2 (MasterMachine1)
PublicIPs: 18.224.151.236 PrivateIPs: 172.31.17.247

Step9:Added the docker credentials for pushing the image in Docker-Hub

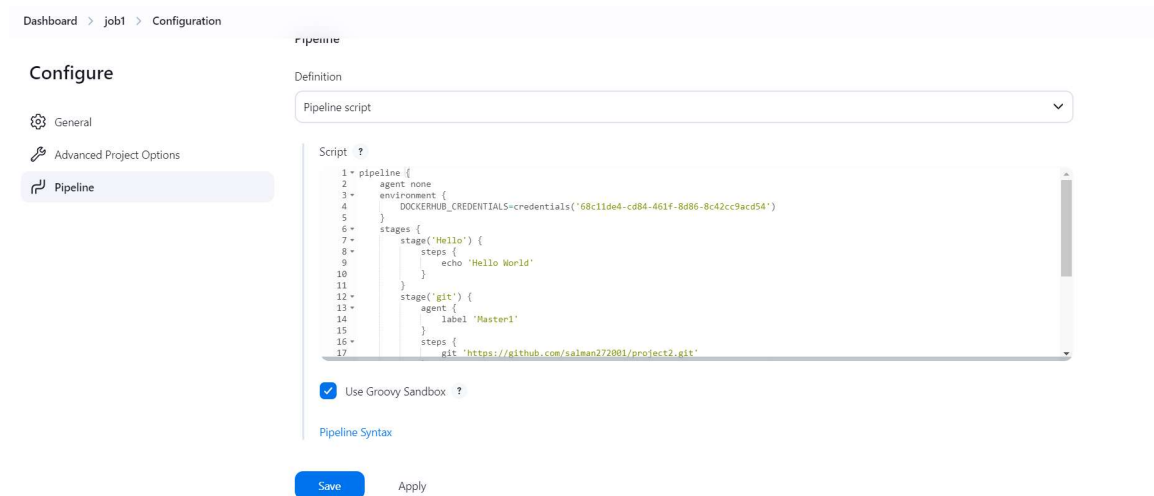
Global credentials (unrestricted) [+ Add Credentials](#)

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
 sshkey	jenkins (key)	SSH Username with private key	key 
 mykey	jenkins	SSH Username with private key	
 privatekey	ubuntu	SSH Username with private key	
 68c11de4-cd84-461f-8d86-8c42cc9acd54	salman272001/*****	Username with password	

Icon: S M L

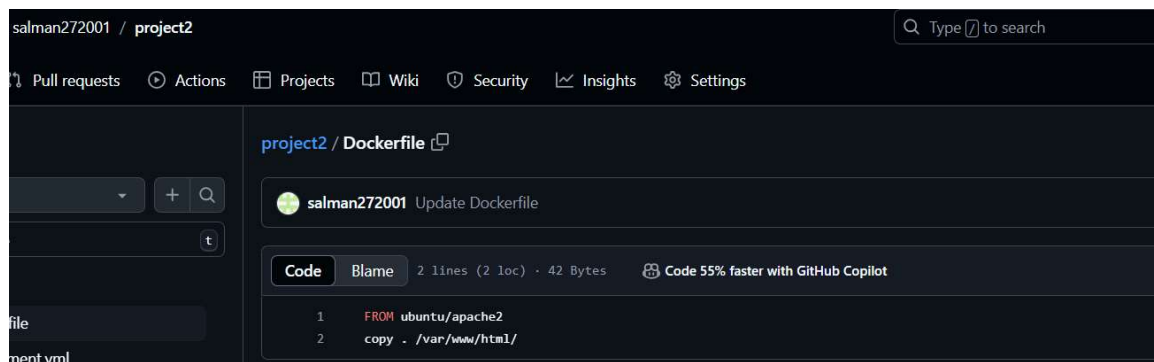
Step10: Pipeline creation builded the first stage (Hello world) in pipeline.



The screenshot shows the Jenkins Pipeline Configuration page. The left sidebar has a 'Pipeline' tab selected. The main area shows the 'Definition' dropdown set to 'Pipeline script'. Below this is a text area for the pipeline script, which contains a Groovy script defining a pipeline with two stages: 'Hello' and 'git'. The 'Hello' stage has a step 'echo' with the text 'Hello World'. The 'git' stage has a step 'git' with the repository URL 'https://github.com/salman272001/project2.git'. Below the script area is a checkbox for 'Use Groovy Sandbox' which is checked. At the bottom are 'Save' and 'Apply' buttons.

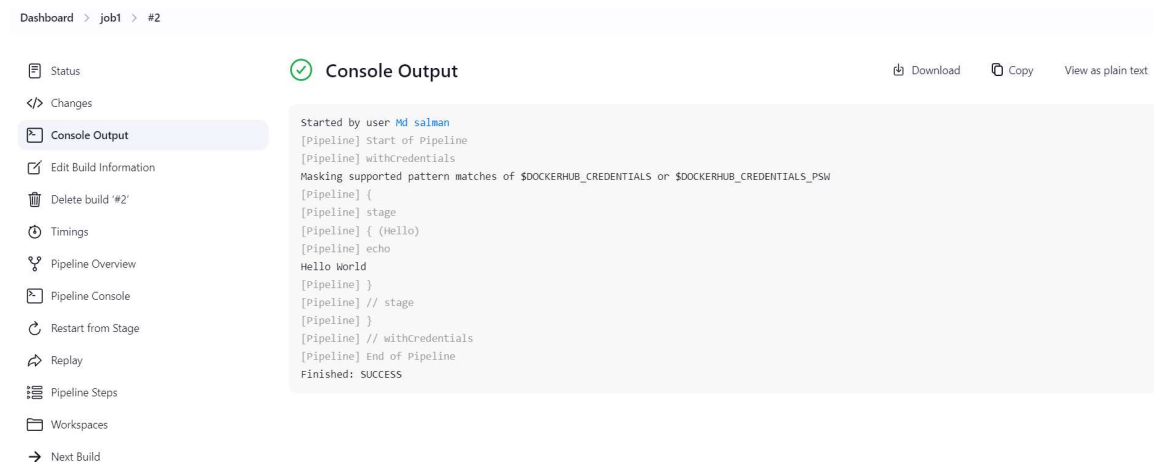
```
1 pipeline {
2   agent none
3   environment {
4     DOCKERHUB_CREDENTIALS=credentials('68c11de4-cd84-461f-8d86-8c42cc9acd54')
5   }
6   stages {
7     stage('Hello') {
8       steps {
9         echo 'Hello World'
10      }
11    }
12    stage('git') {
13      agent {
14        label 'Master1'
15      }
16      steps {
17        git 'https://github.com/salman272001/project2.git'
```

Created the Dockerfile in git-hub and it succesfully build.



The screenshot shows the GitHub repository page for 'salman272001 / project2'. The 'Dockerfile' file is selected, showing its content. The file has 2 lines (2 loc) and 42 Bytes. The content of the Dockerfile is:

```
1 FROM ubuntu/apache2
2 copy . /var/www/html/
```



The screenshot shows the Jenkins Console Output for a build. The output shows the pipeline execution steps and the successful completion of the 'Hello World' stage.

```
Started by user Md salman
[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
```

Dashboard > job1 > #5

Status

Changes

Console Output

Edit Build Information

Delete build '#5'

Timings

Git Build Data

Pipeline Overview

Pipeline Console

Restart from Stage

Replay

Pipeline Steps

Workspaces

Previous Build

Next Build

Console Output

Download Copy View as plain

```
Started by user Md salman
[Pipeline] Start of Pipeline
[Pipeline] withCredentials
Masking supported pattern matches of $DOCKERHUB_CREDENTIALS on $DOCKERHUB_CREDENTIALS_PSW
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Hello)
[Pipeline] echo
Hello world
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (git)
[Pipeline] node
Running on Master1 in /home/ubuntu/jenkins/workspace/job1
[Pipeline] {
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/salman272801/project2.git
> git init /home/ubuntu/jenkins/workspace/job1 # timeout=10
Fetching upstream changes from https://github.com/salman272801/project2.git
> git --version # timeout=10
> git --version # 'git version 2.34.1'
> git fetch --tags --force --progress -- https://github.com/salman272801/project2.git +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
Checking out Revision 883b439379efd36978896f7c57089e5cb546a9a (refs/remotes/origin/master)
> git config remote.origin.url https://github.com/salman272801/project2.git # timeout=10
```

Step11:Now its time to Build and push the docker Image in Stage2 and we have validate the credentials in pipeline script check out below.

Dashboard > job1 > Configuration

Configure

General

Advanced Project Options

Pipeline

Definition

Pipeline script

```
Script ?
20 + stage('docker') {
21 +   agent {
22 +     label 'Master1'
23 +   }
24 +   steps {
25 +     sh 'sudo docker build /home/ubuntu/jenkins/workspace/job1 -t salman272801/project2'
26 +     sh 'sudo echo $DOCKERHUB_CREDENTIALS_PSW | sudo docker login -u $DOCKERHUB_CREDENTIALS_USR --password -stdin'
27 +     sh 'sudo docker push salman272801/project2'
28 +   }
29 + }
30 + stage('kubernetes') {
31 +   agent {
32 +     label 'Master1'
33 +   }
34 +   steps {
35 +     sh 'kubectl apply -f deployment.yml'
36 +     sh 'kubectl apply -f service.yml'
```

☒ Use Groovy Sandbox ?

Pipeline Syntax

Save Apply

And it successfully build and pushed in Docker-hub.

Dashboard > job1 > #10

Status Console Output Download Copy View as plain text

Changes

Console Output

Edit Build Information

Delete build #10

Timings

Git Build Data

Pipeline Overview

Pipeline Console

Replay

Pipeline Steps

Workspaces

Previous Build

Next Build

```

Started by user Md salman
[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] stage
[Pipeline] { (git)
[Pipeline] node
Running on Master1 in /home/ubuntu/jenkins/workspace/job1
[Pipeline] {
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
Fetching changes from the remote Git repository
> git rev-parse --resolve-git-dir /home/ubuntu/jenkins/workspace/job1/.git # timeout=10
Checking out Revision 9c8d496f98f7b60bf72ab555fb3efb5a994019d (refs/remotes/origin/master)
Commit message: "Update Dockerfile"
> git config remote.origin.url https://github.com/salman272001/project2.git # timeout=10

```

```

+ sudo docker login -u salman272001 --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 salman272001/project2
Using default tag: latest
The push refers to repository [docker.io/salman272001/project2]
5462c7bef063: Preparing
deb9dc140b41: Preparing
e5974be2b52b: Preparing
a30a5965a4f7: Preparing
a30a5965a4f7: Mounted from ubuntu/apache2
deb9dc140b41: Mounted from ubuntu/apache2
e5974be2b52b: Mounted from ubuntu/apache2
5462c7bef063: Pushed
latest: digest: sha256:44bae9f5ae0b758b7267b221228be3bcd43a59465bcd804cec8221565834f4 size: 1158
[Pipeline] }
[Pipeline] // node
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withCredentials
[Pipeline] End of Pipeline
Finished: SUCCESS

```

salman272001 Search by repository name All Content Create repository

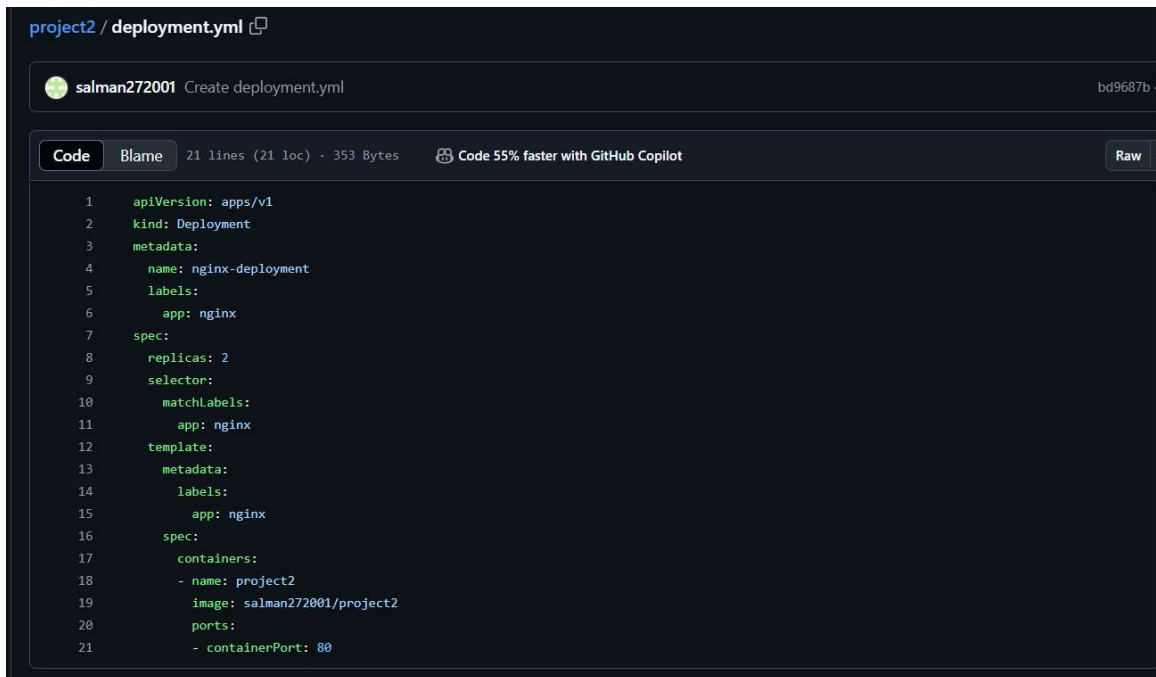
salman272001 / project2

Contains: Image Last pushed: 4 days ago

0 12 Public Scout inactive

Step12: Now I am gonna create the deployment.yml with 2 replicas in order to display the image in other workers. And this our stage3 which is

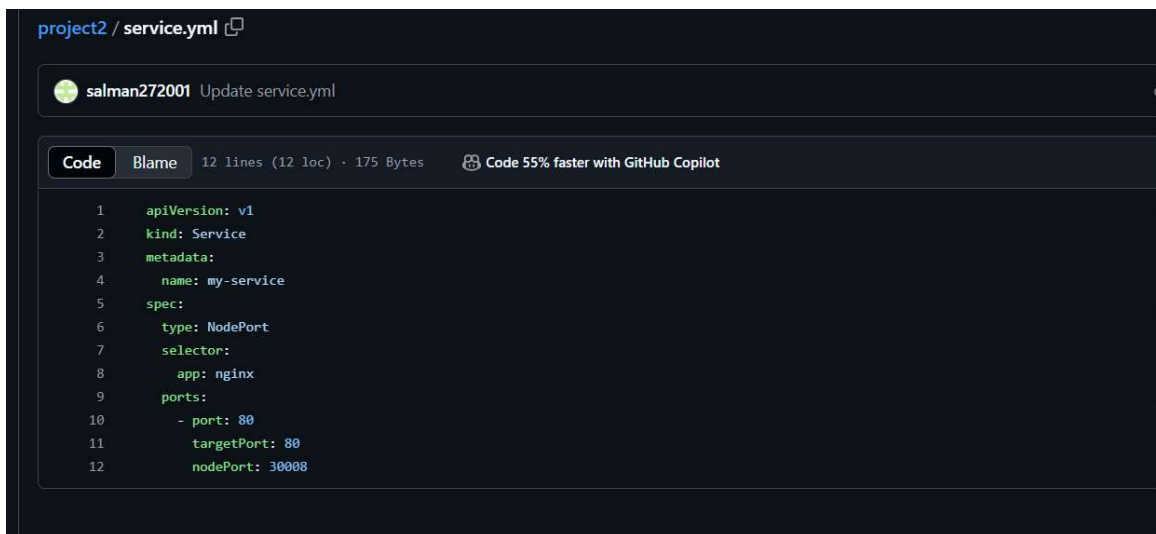
kubernetes stage.



The screenshot shows a GitHub repository interface for a file named `project2 / deployment.yml`. The file was created by user `salman272001`. The code is displayed in a dark-themed editor with line numbers 1 through 21. The YAML content defines a Kubernetes Deployment for an application named `nginx-deployment`. It specifies two replicas and a selector matching the `app: nginx` label. The deployment template includes a container named `project2` using the `salman272001/project2` image, with port 80 exposed on the container.

```
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: project2
19        image: salman272001/project2
20        ports:
21        - containerPort: 80
```

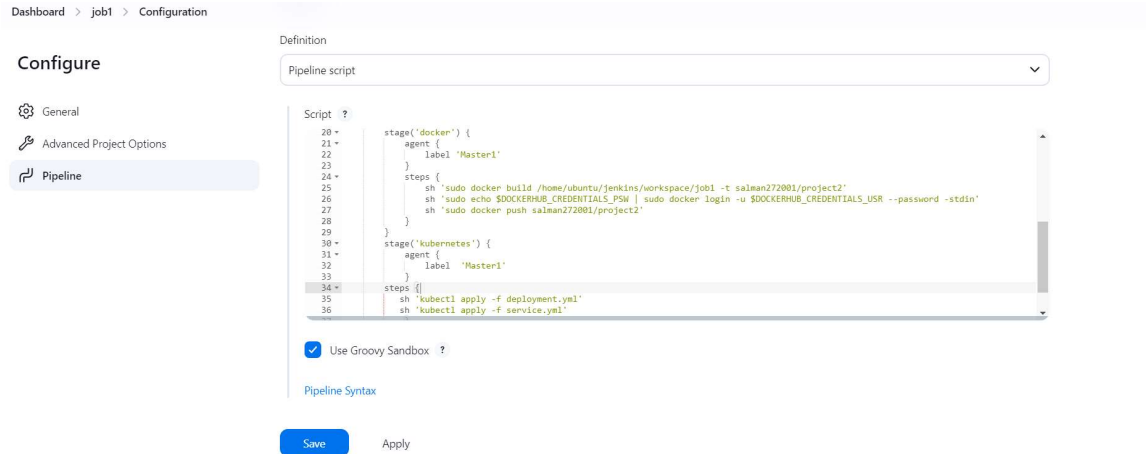
And we'll also create the service.yml for our node on port no.300008.



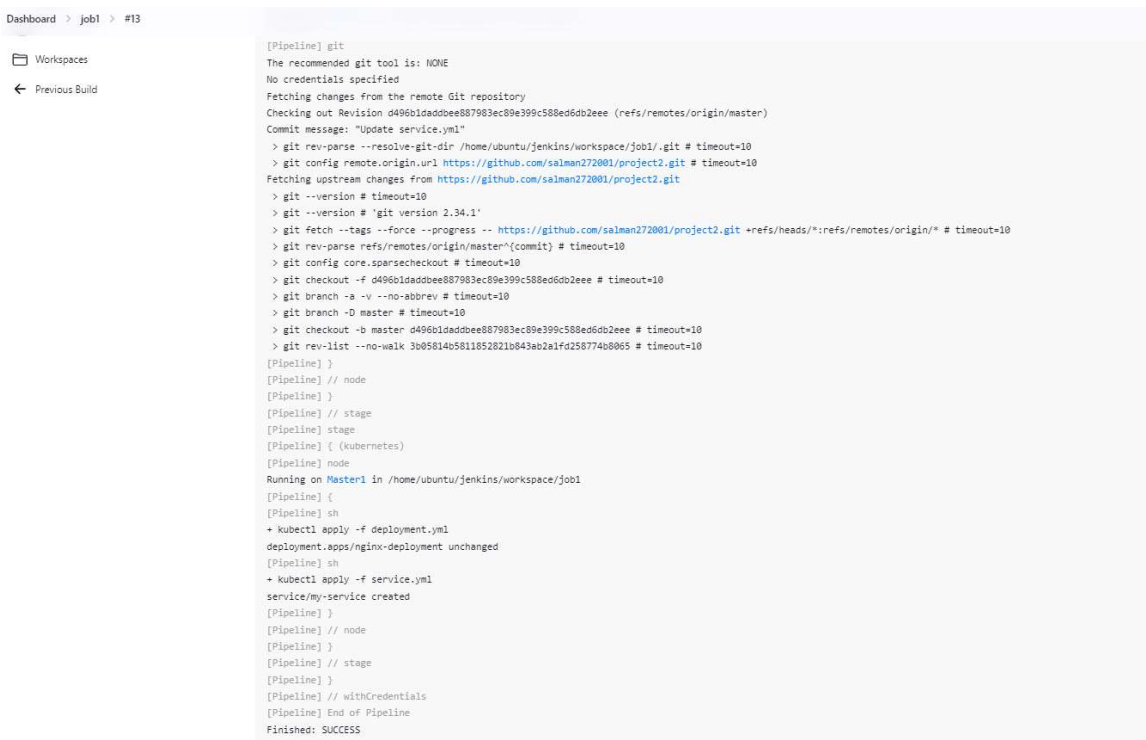
The screenshot shows a GitHub repository interface for a file named `project2 / service.yml`. The file was updated by user `salman272001`. The code is displayed in a dark-themed editor with line numbers 1 through 12. The YAML content defines a Kubernetes Service of type `NodePort` named `my-service`. It uses the same `app: nginx` selector as the deployment and exposes port 80 on the container, which is mapped to node port 30008 on the host.

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: my-service
5  spec:
6    type: NodePort
7    selector:
8      app: nginx
9    ports:
10     - port: 80
11       targetPort: 80
12     nodePort: 30008
```

Step13: Now its time to write kubernetes command in pipeline script. and click on build.



And as you can see our deployment and service.yml successfully ran.



And the image successfully displayed in all the workers except local master. Master1 which is Worker 2.



Hello world!



GitHub

Slave1 which is worker3.



Hello world!



GitHub

Slave2 which is worker4.



