

## Project: Capstone 1

You have been hired as a Sr. DevOps Engineer in Abode Software. They want to implement DevOps Lifecycle in their company. You have been asked to implement this lifecycle as fast as possible. Abode Software is a product-based company and their product is available on this GitHub link.

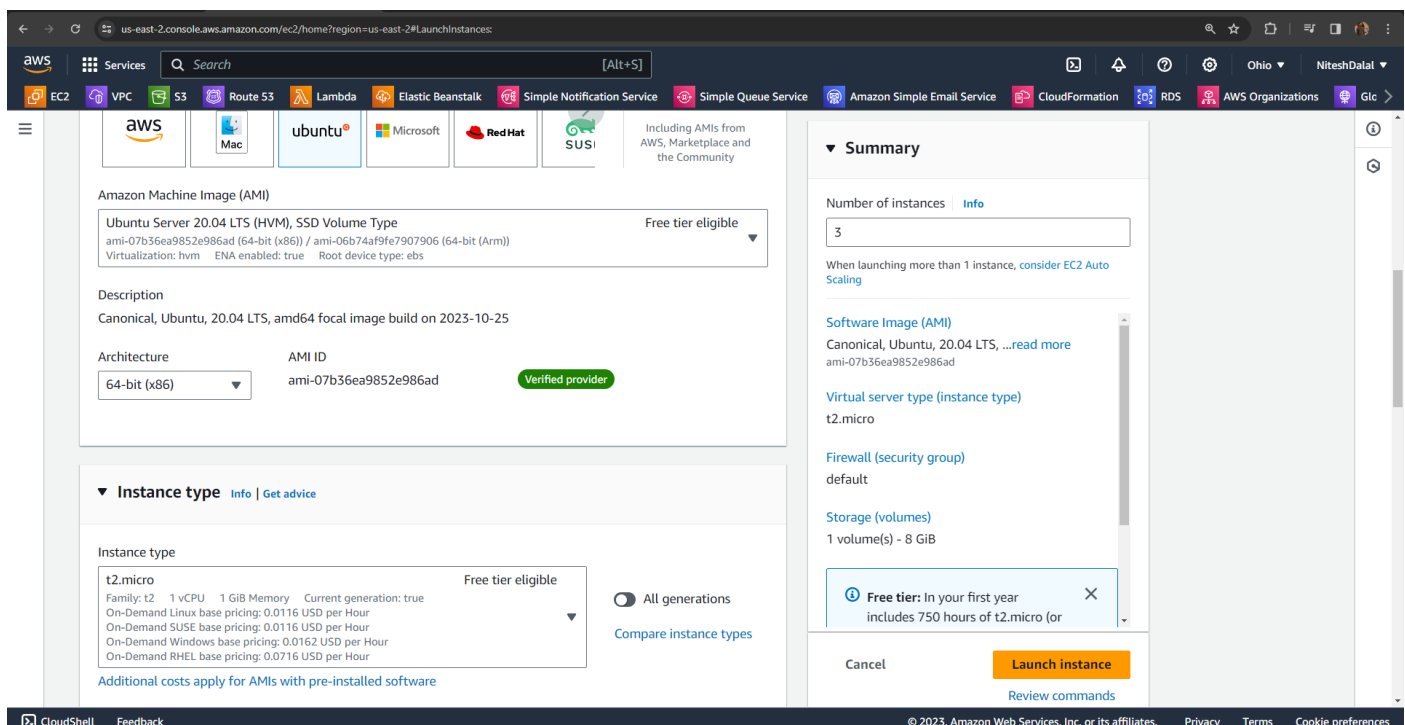
<https://github.com/hshar/webapp>

Following are the specifications of the lifecycle:

1. Install the necessary software on the machines using a configuration management tool
2. Git workflow has to be implemented
3. CodeBuild should automatically be triggered once a commit is made to master branch or develop branch. a. If a commit is made to master branch, test and push to prod b. If a commit is made to develop branch, just test the product, do not push to prod
4. The code should be containerized with the help of a Dockerfile. The Dockerfile should be built every time there is a push to GitHub. Use the following pre-built container for your application: hshar/webapp. The code should reside in '/var/www/html'
5. The above tasks should be defined in a Jenkins Pipeline with the following jobs:
  - a. Job1: Build
  - b. Job2: Test
  - c. Job3: Prod

**Solution:**

1. As per the requirement we need to have 3 EC2 instances. Also enabling it with all traffic enabled & with Ubuntu as AMI.





```

ubuntu@ip-172-31-7-114:~$ 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:sVbH9c25YSba74/M0l9pVmq0Bpxm2jFXl1xpR4XpCQQ ubuntu@ip-172-31-7-114
The key's randomart image is:
+---[RSA 3072]-----+
|      E+. o+O|
|      oo.oX+|
|      . . +++*=|
|      + + o* o|
|      S . B . |
|      . X o   |
|      = =, = |
|      o =, = |
|      .+o+o|
+---[SHA256]-----+

```

- Now from .ssh folder of master machine I will copy the id\_rsa.pub file content. And then paste it to .ssh folder of slave machine in file name -> authorized keys.

```

ubuntu@ip-172-31-7-114:~$ cd .ssh
ubuntu@ip-172-31-7-114:~/.ssh$ ls
authorized_keys  id_rsa  id_rsa.pub
ubuntu@ip-172-31-7-114:~/.ssh$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgCQSV1+SNDJmFl8Q6tWB2nFOXgaoAiLod2N1PKAWMv1WC5iexmp2VWA9yGRrPybVckgloggDyU3wSorWp23nlUetTlppy7TEHQS/VG/+iMc6z2E3e1GULejsDKrt/TgR8
+KQ3xJ5ZHvt16mOZqrJ8km2dsuBiQZi65EVusBK+mmuY08gxHSVrk6f7yVBsYwqULPTrIkJgwLjU/4F3hGXYB1ibJQqOiaqJigAVwN3oUFTjvox122cr1C+o2eavrdUuj7E72v/vo27r6KAGWxhJOJ54kB8rAH51+ytIo
onURRYIu2ErPQCQWkJSBwc+LMm4TIVgt+mffYqVVufCPKwUwwZ/L0= ubuntu@ip-172-31-7-114
ubuntu@ip-172-31-7-114:~/.ssh$

```

i-0d53ae463b7d3a633 (Project-M)  
PublicIPs: 18.218.78.102 PrivateIPs: 172.31.7.114

```

us-east-2.console.aws.amazon.com/ec2-instance-connect/ssh?region=us-east-2&connType=standard&instanceId=i-048d763d862dcd398&osUser=ubuntu&sshPort=22#/?
aws
Services
Search [Alt+S]
EC2 VPC S3 Route 53 Lambda Elastic Beanstalk Simple Notification Service Simple Queue Service Amazon Simple Email Service CloudFormation RDS AWS Organizations
GNU nano 4.8 authorized_keys Modified
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgCQSV1+SNDJmFl8Q6tWB2nFOXgaoAiLod2N1PKAWMv1WC5iexmp2VWA9yGRrPybVckgloggDyU3wSorWp23nlUetTlppy7TEHQS/VG/+iMc6z2E3e1GULejsDKrt/TgR8
+KQ3xJ5ZHvt16mOZqrJ8km2dsuBiQZi65EVusBK+mmuY08gxHSVrk6f7yVBsYwqULPTrIkJgwLjU/4F3hGXYB1ibJQqOiaqJigAVwN3oUFTjvox122cr1C+o2eavrdUuj7E72v/vo27r6KAGWxhJOJ54kB8rAH51+ytIo
onURRYIu2ErPQCQWkJSBwc+LMm4TIVgt+mffYqVVufCPKwUwwZ/L0= ubuntu@ip-172-31-7-114$
Get Help Write Out Where Is Cut Text Justify Cur Pos M-U Undo M-A Mark Text M-] To Bracket M-Q Previous
Exit Read File Replace Paste Text To Spell Go To Line M-E Redo M-6 Copy Text M-^ Where Was M-W Next

```

i-048d763d862dcd398 (Project-S1)  
PublicIPs: 18.119.105.185 PrivateIPs: 172.31.11.197

```

us-east-2.console.aws.amazon.com/ec2-instance-connect/ssh?region=us-east-2&connType=standard&instanceId=i-04a5c1a388f192b3f&osUser=ubuntu&sshPort=22#/?
aws
Services
Search [Alt+S]
EC2 VPC S3 Route 53 Lambda Elastic Beanstalk Simple Notification Service Simple Queue Service Amazon Simple Email Service CloudFormation RDS AWS Organizations
GNU nano 4.8 authorized_keys Modified
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgCQSV1+SNDJmFl8Q6tWB2nFOXgaoAiLod2N1PKAWMv1WC5iexmp2VWA9yGRrPybVckgloggDyU3wSorWp23nlUetTlppy7TEHQS/VG/+iMc6z2E3e1GULejsDKrt/TgR8
+KQ3xJ5ZHvt16mOZqrJ8km2dsuBiQZi65EVusBK+mmuY08gxHSVrk6f7yVBsYwqULPTrIkJgwLjU/4F3hGXYB1ibJQqOiaqJigAVwN3oUFTjvox122cr1C+o2eavrdUuj7E72v/vo27r6KAGWxhJOJ54kB8rAH51+ytIo
onURRYIu2ErPQCQWkJSBwc+LMm4TIVgt+mffYqVVufCPKwUwwZ/L0= ubuntu@ip-172-31-7-114$
Get Help Write Out Where Is Cut Text Justify Cur Pos M-U Undo M-A Mark Text M-] To Bracket M-Q Previous
Exit Read File Replace Paste Text To Spell Go To Line M-E Redo M-6 Copy Text M-^ Where Was M-W Next

```

i-04a5c1a388f192b3f (Project-S2)  
PublicIPs: 18.118.24.242 PrivateIPs: 172.31.2.126

- Now I will edit the host file for ansible slaves. Which is located in /etc/ansible/ location. Then mentioning the private IP's of both slaves in host file.

```
us-east-2.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-0d53ae463b7d3a6338&osUser=ubuntu&region=us-east-2&sshPort=22#/  
aws  
Services  
[Alt+S]  
EC2 VPC S3 Route 53 Lambda Elastic Beanstalk Simple Notification Service Simple Queue Service Amazon Simple Email Service CloudFormation RDS AWS Organizations Glc  
GNU nano 4.8 hosts Modified  
172.31.11.197  
172.31.2.126  
^G Get Help ^O Write Out ^W Where Is ^X Cut Text ^J Justify ^C Cur Pos ^U Undo ^A Mark Text ^] To Bracket ^_ Previous  
^X Exit ^R Read File ^N Replace ^V Paste Text ^K To Spell ^G Go To Line ^E Redo ^6 Copy Text ^C Where Was ^N Next  
i-0d53ae463b7d3a633 (Project-M)  
PublicIPs: 18.218.78.102 PrivateIPs: 172.31.7.114
```

## 7. Test the connection with hosts.

```
us-east-2.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-0d53ae463b7d3a6338&osUser=ubuntu&region=us-east-2&sshPort=22#/  
aws  
Services  
[Alt+S]  
EC2 VPC S3 Route 53 Lambda Elastic Beanstalk Simple Notification Service Simple Queue Service Amazon Simple Email Service CloudFormation RDS AWS Organizations Glc  
ubuntu@ip-172-31-7-114:/etc/ansible$ ansible -m ping all  
172.31.2.126 | SUCCESS => {  
  "ansible_facts": {  
    "discovered_interpreter_python": "/usr/bin/python3"  
  },  
  "changed": false,  
  "ping": "pong"  
}  
172.31.11.197 | SUCCESS => {  
  "ansible_facts": {  
    "discovered_interpreter_python": "/usr/bin/python3"  
  },  
  "changed": false,  
  "ping": "pong"  
}  
ubuntu@ip-172-31-7-114:/etc/ansible$  
i-0d53ae463b7d3a633 (Project-M)  
PublicIPs: 18.218.78.102 PrivateIPs: 172.31.7.114
```

## 8. Now let's create a playbook file for installation of required tools. For master I will create -> master.sh and for slave instances I will create slave.sh

```
us-east-2.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-0d53ae463b7d3a6338&osUser=ubuntu&region=us-east-2&sshPort=22#/  
aws  
Services  
[Alt+S]  
EC2 VPC S3 Route 53 Lambda Elastic Beanstalk Simple Notification Service Simple Queue Service Amazon Simple Email Service CloudFormation RDS AWS Organizations Glc  
GNU nano 4.8 master.sh Modified  
sudo apt update  
sudo apt install openjdk-11-jdk -y  
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \  
https://pkg.jenkins.io/debian/jenkins.io-2023.key  
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \  
https://pkg.jenkins.io/debian binary/ | sudo tee \  
/etc/apt/sources.list.d/jenkins.list > /dev/null  
sudo apt-get update  
sudo apt-get install jenkins -y  
i-0d53ae463b7d3a633 (Project-M)  
PublicIPs: 18.218.78.102 PrivateIPs: 172.31.7.114
```

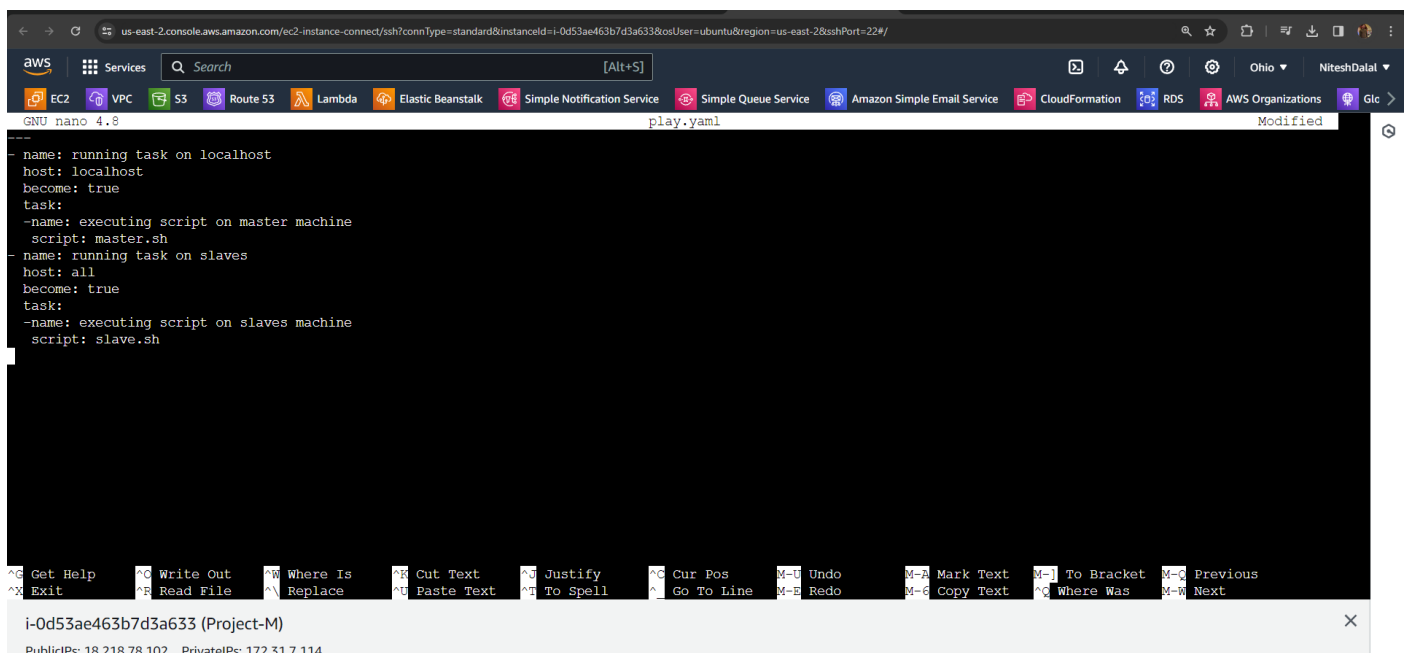


The screenshot shows the AWS Management Console interface with a terminal window open. The terminal is running the GNU nano 4.8 editor, editing a file named `slave.sh`. The user has entered the following commands:

```
sudo apt update
sudo apt install openjdk-11-jdk -y
sudo apt install docker.io -y
```

The terminal window title is `i-0d53ae463b7d3a633 (Project-M)`. The bottom status bar shows the instance's public and private IP addresses: `PublicIPs: 18.218.78.102 PrivateIPs: 172.31.7.114`.

9. Now I will create a playbook file for installation.



The screenshot shows the AWS Management Console interface with a terminal window open. The terminal is running the GNU nano 4.8 editor, editing a file named `play.yaml`. The user has entered the following content:

```
- name: running task on localhost
  host: localhost
  become: true
  task:
    - name: executing script on master machine
      script: master.sh
- name: running task on slaves
  host: all
  become: true
  task:
    - name: executing script on slaves machine
      script: slave.sh
```

The terminal window title is `i-0d53ae463b7d3a633 (Project-M)`. The bottom status bar shows the instance's public and private IP addresses: `PublicIPs: 18.218.78.102 PrivateIPs: 172.31.7.114`.

10. Let's do a syntax check.



The screenshot shows the AWS Management Console interface with a terminal window open. The terminal is running the `ansible-playbook` command to perform a syntax check on the `play.yaml` file:

```
ubuntu@ip-172-31-7-114:/$ ansible-playbook play.yaml --syntax-check
playbook: play.yaml
ubuntu@ip-172-31-7-114:/$
```

The terminal window title is `i-0d53ae463b7d3a633 (Project-M)`. The bottom status bar shows the instance's public and private IP addresses: `PublicIPs: 18.218.78.102 PrivateIPs: 172.31.7.114`.

11. Now doing a dry run of the code.

```
ubuntu@ip-172-31-7-114:/$ ansible-playbook play.yaml --check

PLAY [running tasks on localhost] *****

TASK [Gathering Facts] *****
ok: [localhost]

TASK [executing script on master machine] *****
skipping: [localhost]

PLAY [running tasks on slaves] *****

TASK [Gathering Facts] *****
ok: [172.31.11.197]
ok: [172.31.2.126]

TASK [executing script on slave machine] *****
skipping: [172.31.11.197]
skipping: [172.31.2.126]

PLAY RECAP *****
172.31.11.197      : ok=1    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
172.31.2.126      : ok=1    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
localhost         : ok=1    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0

ubuntu@ip-172-31-7-114:/$
```

i-Od53ae463b7d3a633 (Project-M)

PublicIPs: 18.218.78.102 PrivateIPs: 172.31.7.114

## 12. Finally executing the playbook file.

```
ubuntu@ip-172-31-25-104:~$ ansible-playbook play.yaml

PLAY [running tasks on localhost] *****

TASK [Gathering Facts] *****
ok: [localhost]

TASK [executing script on master machine] *****
changed: [localhost]

PLAY [running tasks on slaves] *****

TASK [Gathering Facts] *****
ok: [172.31.17.205]
ok: [172.31.30.176]

TASK [executing script on slave machine] *****
changed: [172.31.17.205]
changed: [172.31.30.176]

PLAY RECAP *****
172.31.17.205      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
172.31.30.176      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
localhost         : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

ubuntu@ip-172-31-25-104:~$
```

i-O68ab839ae8fd3689 (Project-M)

PublicIPs: 18.60.217.76 PrivateIPs: 172.31.25.104

## 13. Let's test it from slave machine.

```
ubuntu@ip-172-31-17-205:~$ java --version
openjdk 11.0.21 2023-10-17
OpenJDK Runtime Environment (build 11.0.21+9-post-Ubuntu-0ubuntu122.04)
OpenJDK 64-Bit Server VM (build 11.0.21+9-post-Ubuntu-0ubuntu122.04, mixed mode, sharing)
ubuntu@ip-172-31-17-205:~$ docker --version
Docker version 24.0.5, build 24.0.5-0ubuntu1~22.04.1
ubuntu@ip-172-31-17-205:~$
```

i-O8f2573b44790c0d6 (Project-S1)

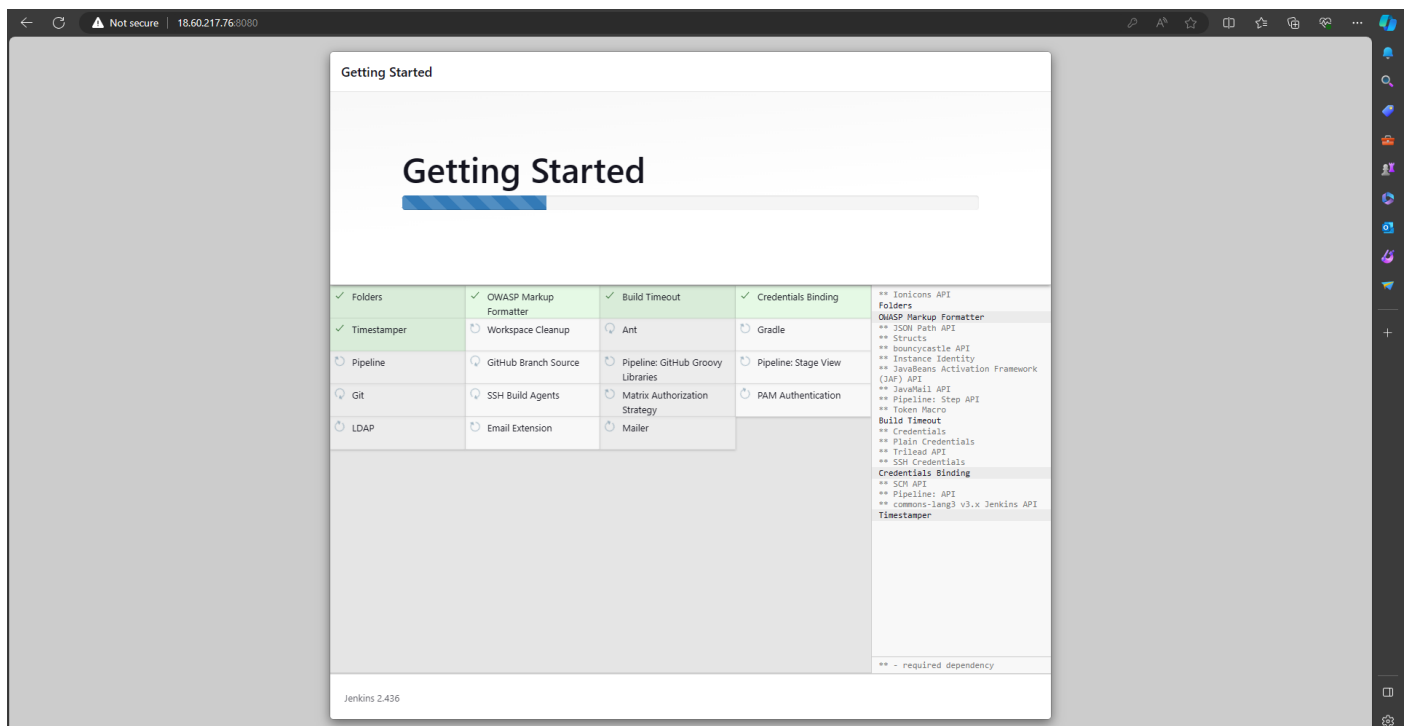
PublicIPs: 18.60.62.204 PrivateIPs: 172.31.17.205

## 14. Now next task is to setup Jenkins dashboard from master machine public IP. Pasted initial password & installed suggested plugins.

```
ubuntu@ip-172-31-25-104:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
a34952428b4e41f6884b9cadfed6f6d8
ubuntu@ip-172-31-25-104:~$ ^C
ubuntu@ip-172-31-25-104:~$
```

i-O68ab839ae8fd3689 (Project-M)

PublicIPs: 18.60.217.76 PrivateIPs: 172.31.25.104



15. Now next task is to add our slave machines as two nodes of Jenkins.

Dashboard > Manage Jenkins > Nodes > New node

## New node

Node name

Type

☒ Permanent Agent

Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.

Create

Remote root directory ?

Launch method ?

Launch agents via SSH

Host ?

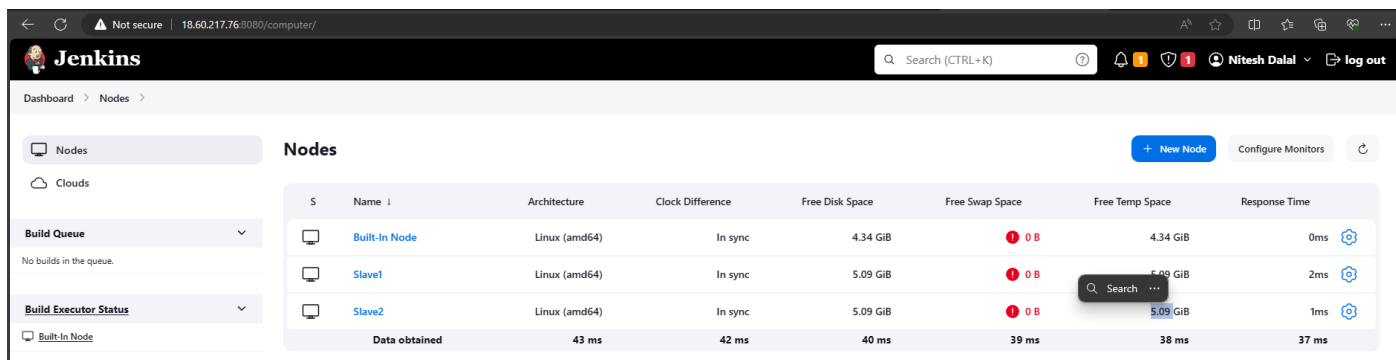
Credentials ?

+ Add

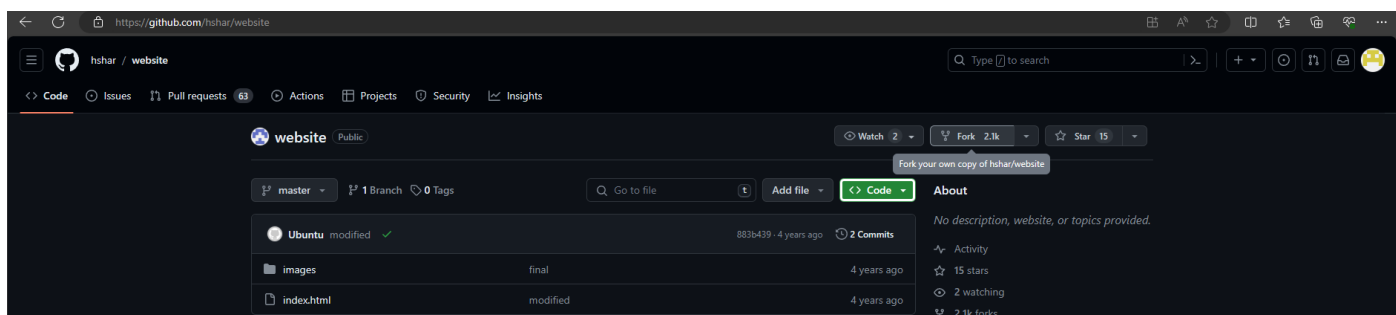
Host Key Verification Strategy ?

Advanced

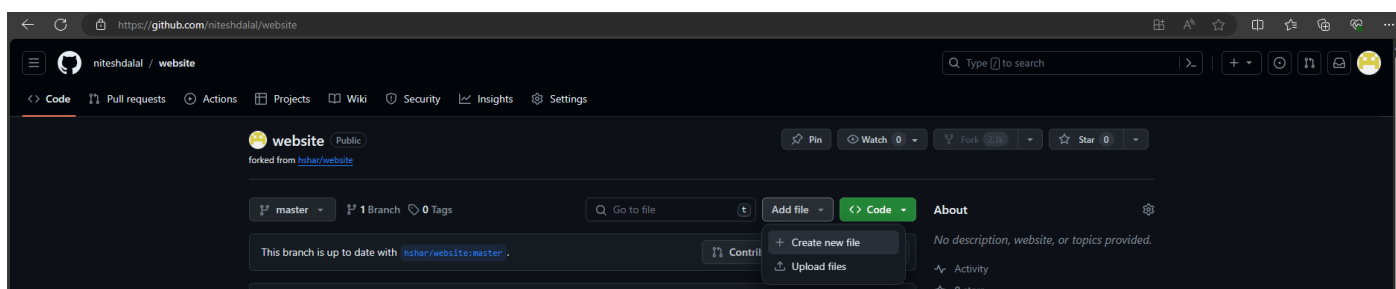
16. Repeat the same for slave2.



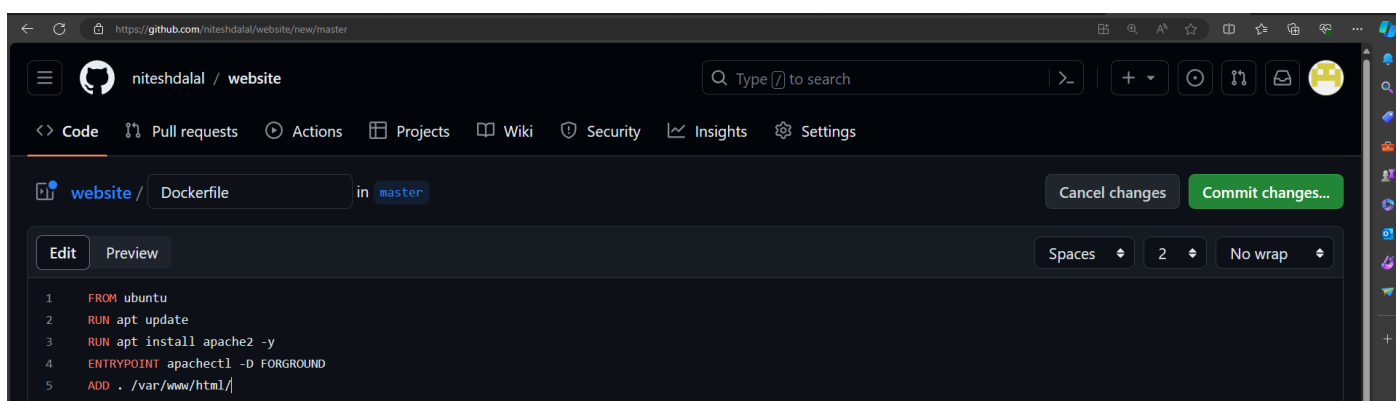
17. Now next we are going to create a repository on github.com by forking it from the project question url.



18. Now we will create a docker file on github.com from here.

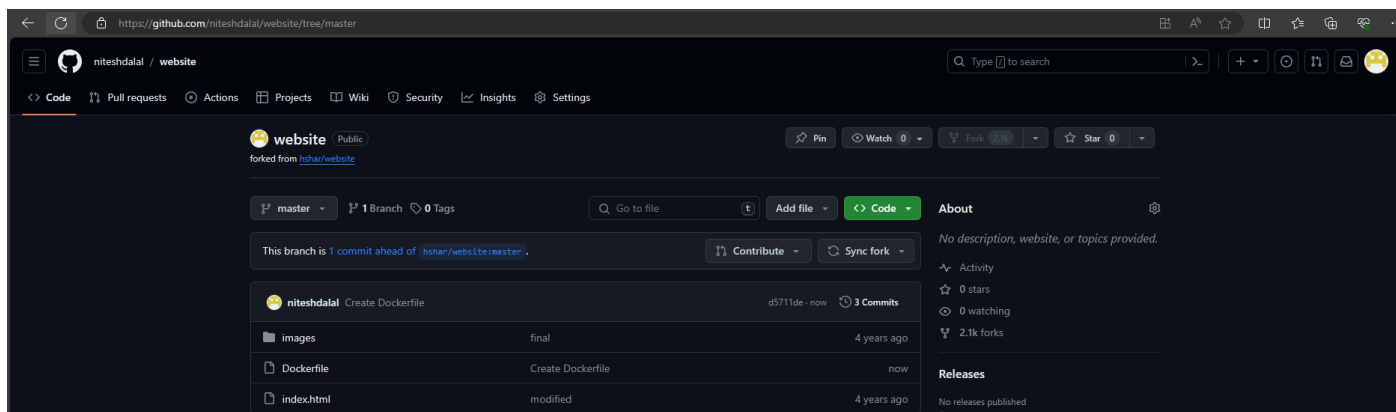


19. Write the code to install everything and copy the content in that specific folder.

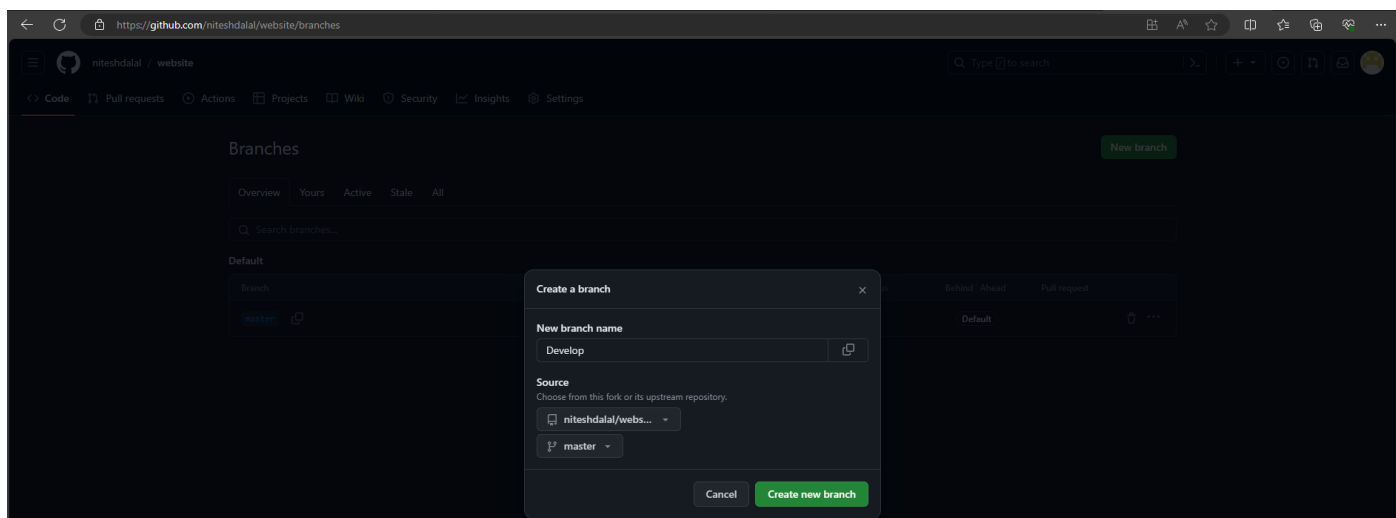


20. Commit the changes and it will create a docker file.

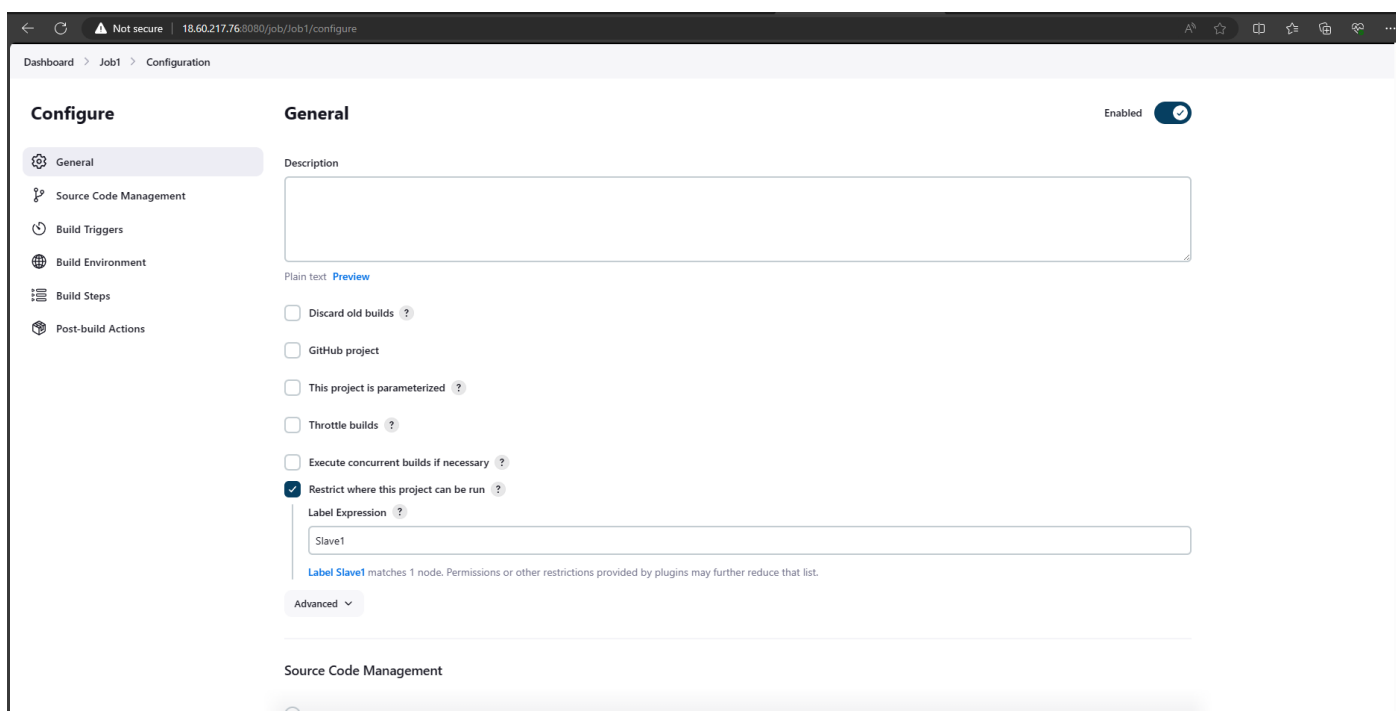




21. Now we need to create an additional branch as per question i.e. Develop.



22. Next we will create jobs in Jenkins dashboard.



Dashboard > Job1 > Configuration

## Configure

- General
- Source Code Management**
- Build Triggers
- Build Environment
- Build Steps
- Post-build Actions

### Source Code Management

☐ None

☒ **Git** ?

Repositories ?

Repository URL ?

Credentials ?

- none -

+ Add

Advanced

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

Add Branch

Dashboard > Job1 > Configuration

## Configure

- General
- Source Code Management
- Build Triggers**
- Build Environment
- Build Steps
- Post-build Actions

### Build Triggers

☐ Trigger builds remotely (e.g., from scripts) ?

☐ Build after other projects are built ?

☐ Build periodically ?

☒ **GitHub hook trigger for GITScm polling** ?

☐ Poll SCM ?

### Build Environment

☐ Delete workspace before build starts

☐ Use secret text(s) or file(s) ?

☐ Add timestamps to the Console Output

☐ Inspect build log for published build scans

☐ Terminate a build if it's stuck

☐ With Ant ?

### Build Steps

Execute shell ?

Command

See [the list of available environment variables](#)

```
sudo docker build /home/ubuntu/jenkins/workspace/Job1 -t j1
sudo docker run -itd -p 83:88 --name=c1 j1
```

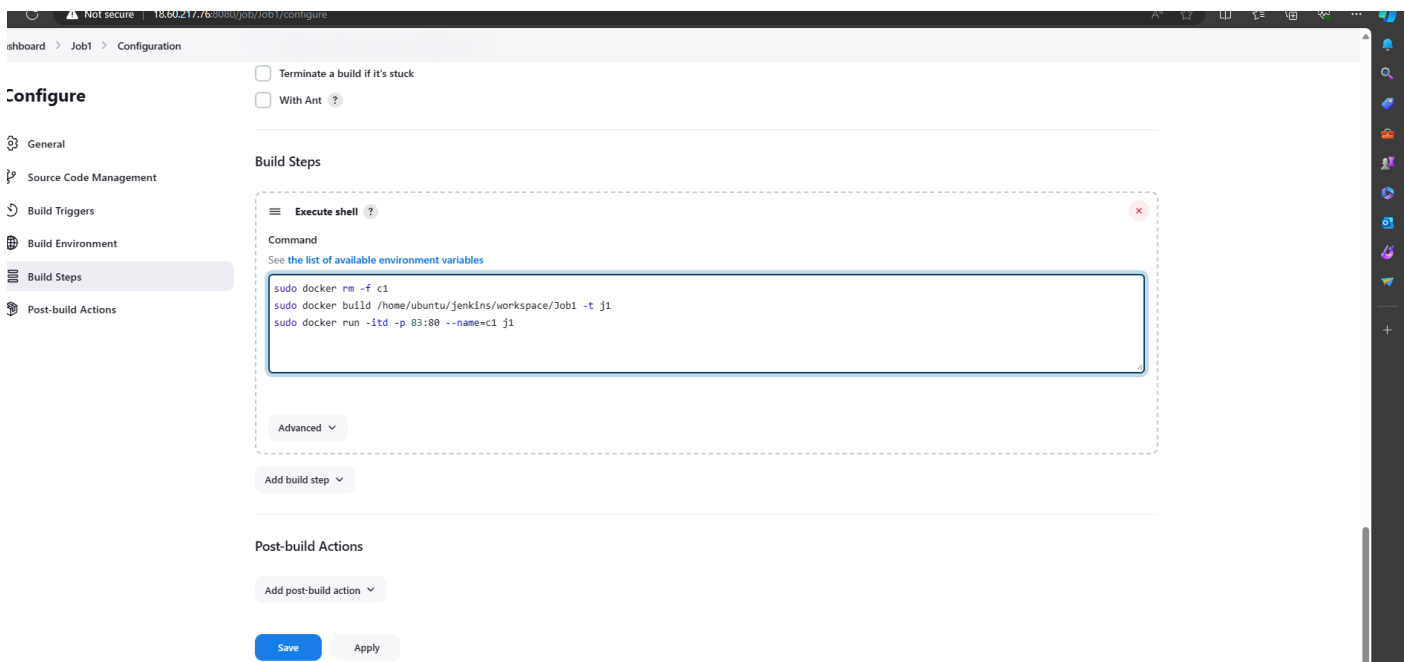
Save

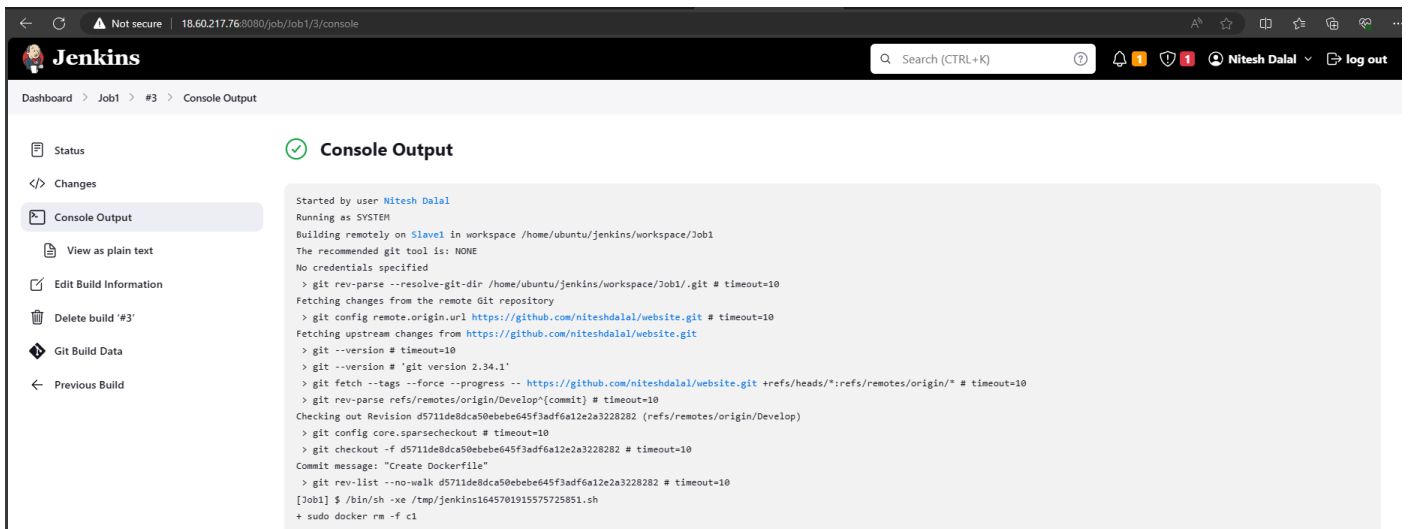
Apply

23. Now build the first job with Build now button of Jenkins dashboard and test it through the Slave1 public IP. As it is your testing environment.

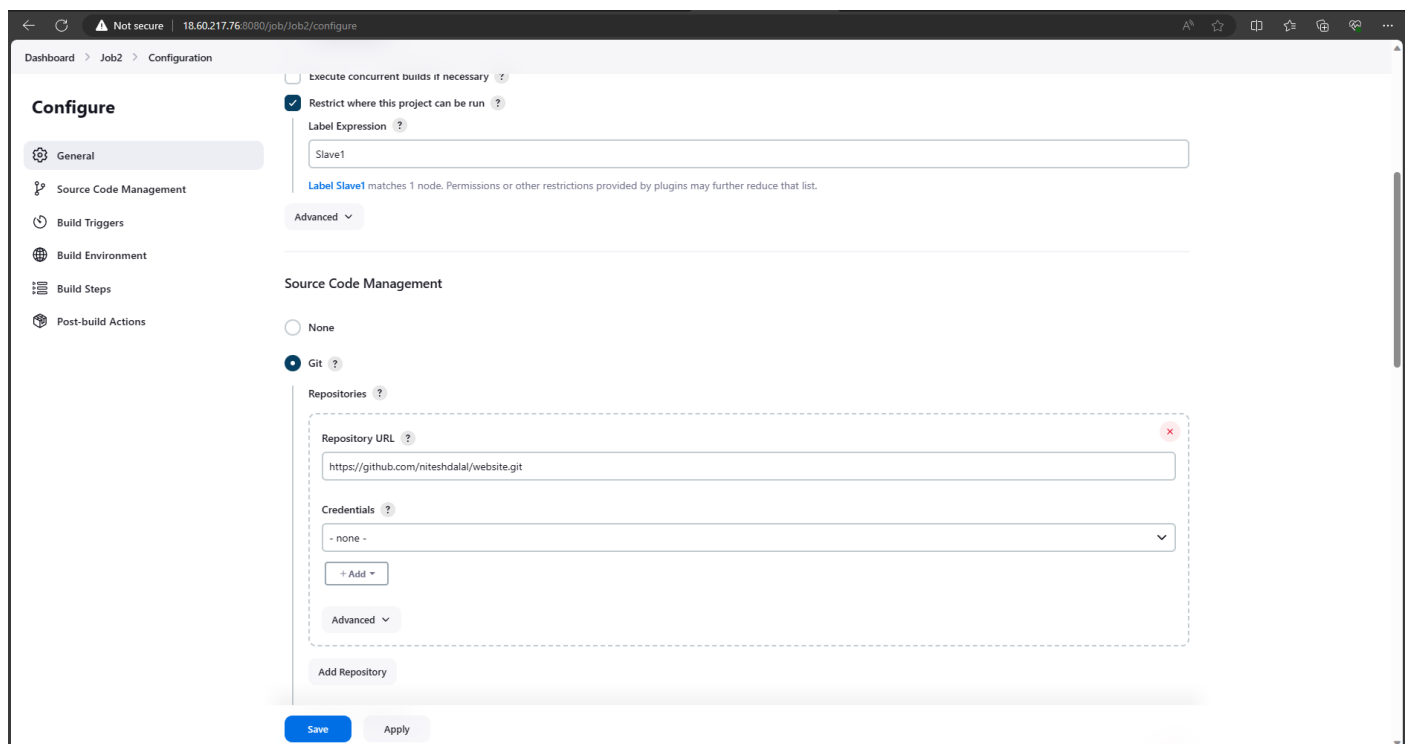


24. Once the first job is successfully build we need to re-configure the job in such a way to delete the existing container. To avoid the error on our next jobs creation.





25. Now next we will create Job for Slave1 but this time for master branch. Also we need to edit the image & container names.



**Configure**

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps**
- Post-build Actions

☐ With Ant ?

### Build Steps

**Execute shell** ?

Command

See [the list of available environment variables](#)

```

sudo docker rm -f c2
sudo docker build /home/ubuntu/jenkins/workspace/Job2 -t j2
sudo docker run -itd -p 82:80 --name=c2 j2

```

Advanced ▾

Add build step ▾

### Post-build Actions

Add post-build action ▾

**Save** **Apply**

26. After the first successful build we need to re configure the Job2 with execute shell commands as follow.

← ↻ ⚠ Not secure | 18.60.217.76:8080/job/Job2/configure

Dashboard > Job2 > Configuration

**Configure**

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps**
- Post-build Actions

☐ Terminate a build if it's stuck

☐ With Ant ?

### Build Steps

**Execute shell** ?

Command

See [the list of available environment variables](#)

```

sudo docker rm -f c2
sudo docker build /home/ubuntu/jenkins/workspace/Job2 -t j2
sudo docker run -itd -p 82:80 --name=c2 j2

```

Advanced ▾

Add build step ▾


### Post-build Actions

Add post-build action ▾

**Save** **Apply**

27. Let's create the final Job3 which will execute on production server i.e. Slave2 with master branch restriction.

← ↻ ⚠ Not secure | 18.60.217.76:8080/job/job3/configure

 Jenkins

Search (CTRL+K) 🔍

🔔 🛡️ 🇮🇳 Nitesh Dalal log out

Dashboard > Job3 > Configuration

Configure

General

Source Code Management

Build Triggers

Build Environment

Build Steps

Post-build Actions

General

Description

Plain text [Preview](#)

☐ Discard old builds ?

☐ GitHub project

☐ This project is parameterized ?

☐ Throttle builds ?

☐ Execute concurrent builds if necessary ?

☒ Restrict where this project can be run ?

Label Expression ?

Slave2


Label Slave2 matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.

Advanced ▾


Source Code Management

Save

Apply

Enabled 

← ↻ ⚠ Not secure | 18.60.217.76:8080/job/job3/configure

 Jenkins

Search (CTRL+K) 🔍

🔔 🛡️ 🇮🇳 Nitesh Dalal log out

Dashboard > Job3 > Configuration

Configure

General

Source Code Management

Build Triggers

Build Environment

Build Steps

Post-build Actions

Source Code Management

☐ None

☒ Git ?

Repositories ?

Repository URL ?

https://github.com/niteshdalal/website.git

Credentials ?

- none -

+ Add ▾

Advanced ▾

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

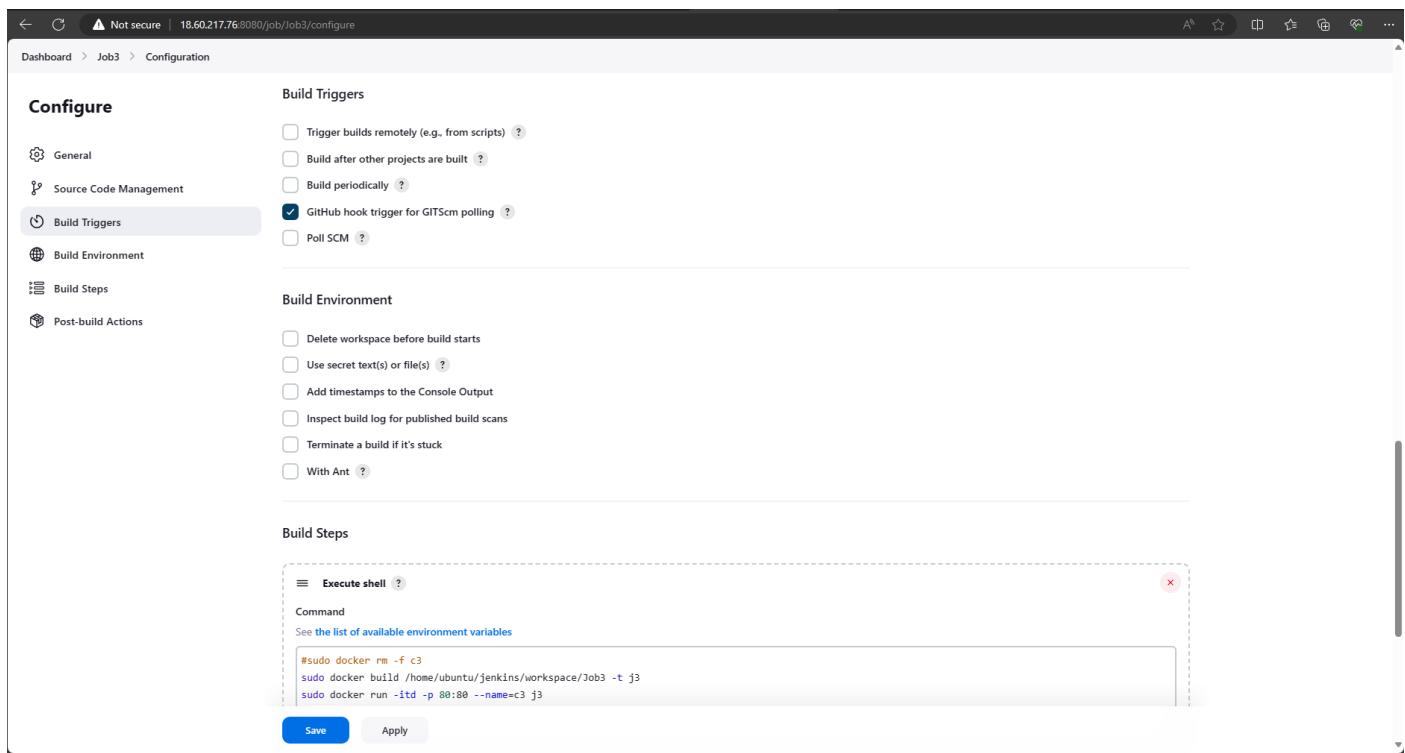
\*/master

Add Branch

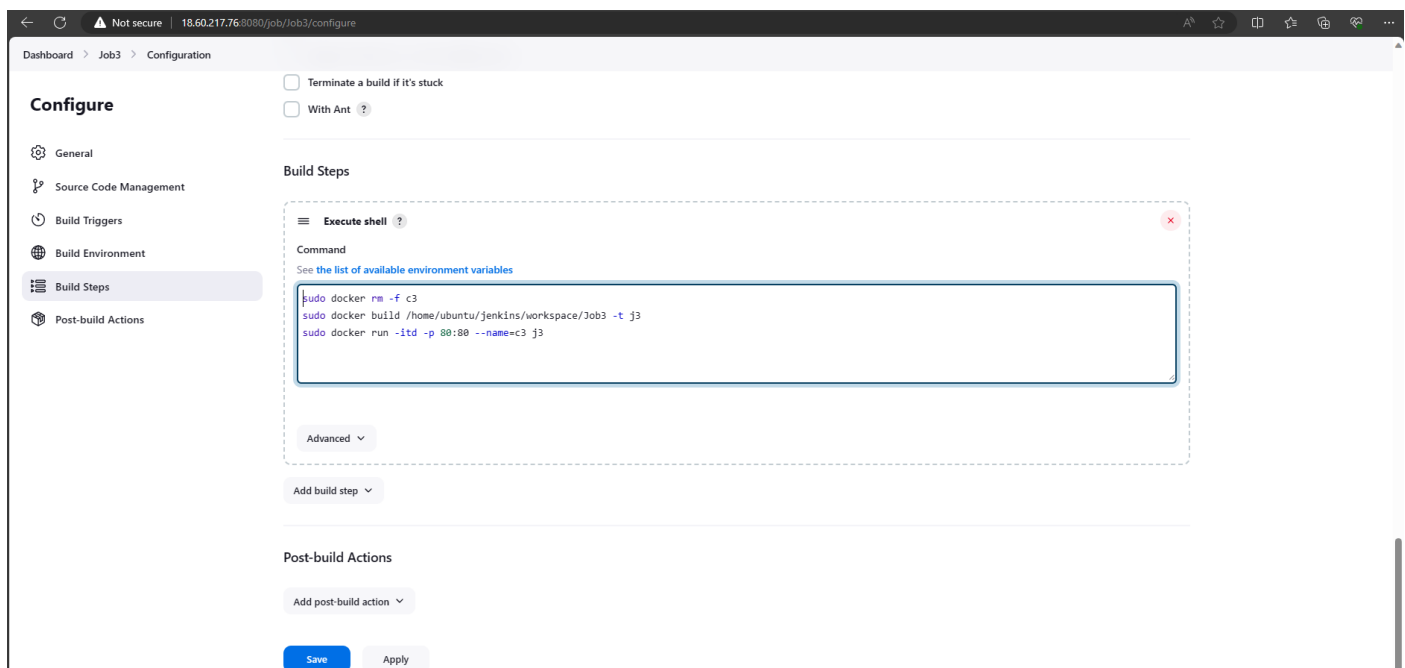
Repository browser ?

Save

Apply



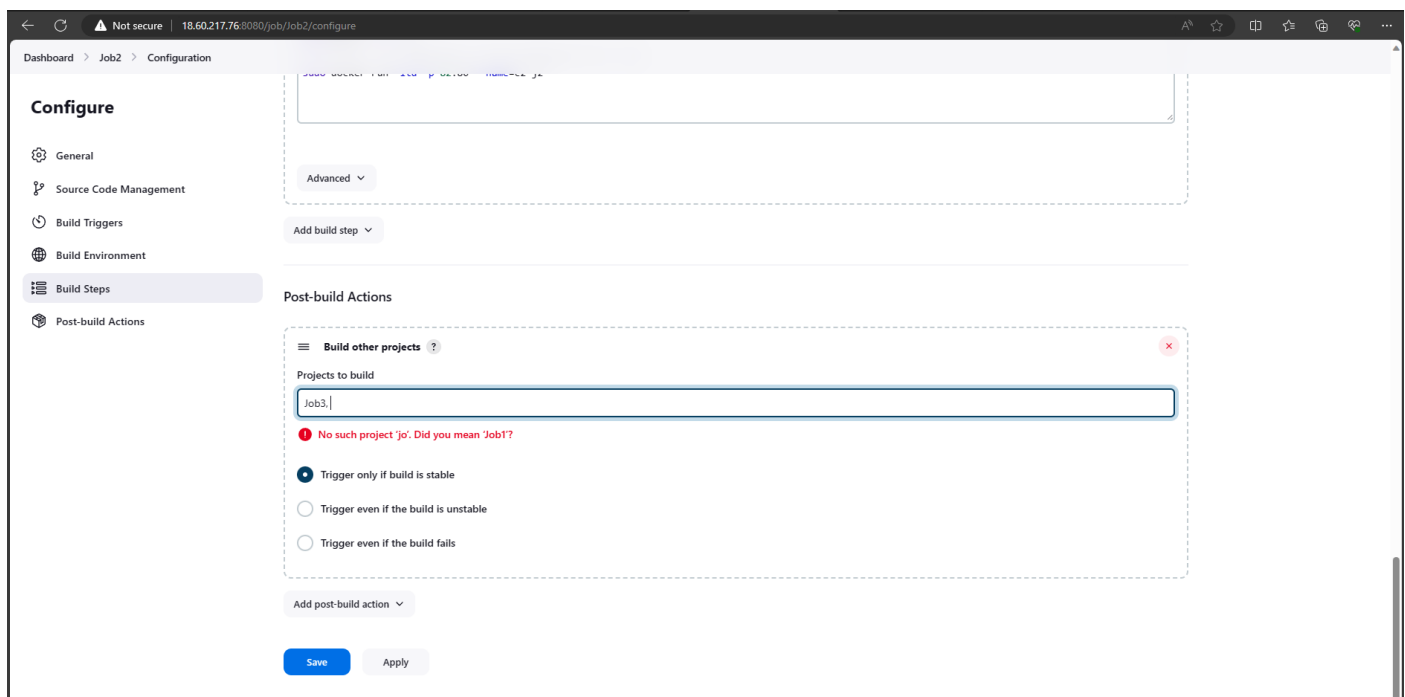
28. Reconfiguring the Job3 after first build.



29. See on slave2 machine it is running on port 80 correctly.

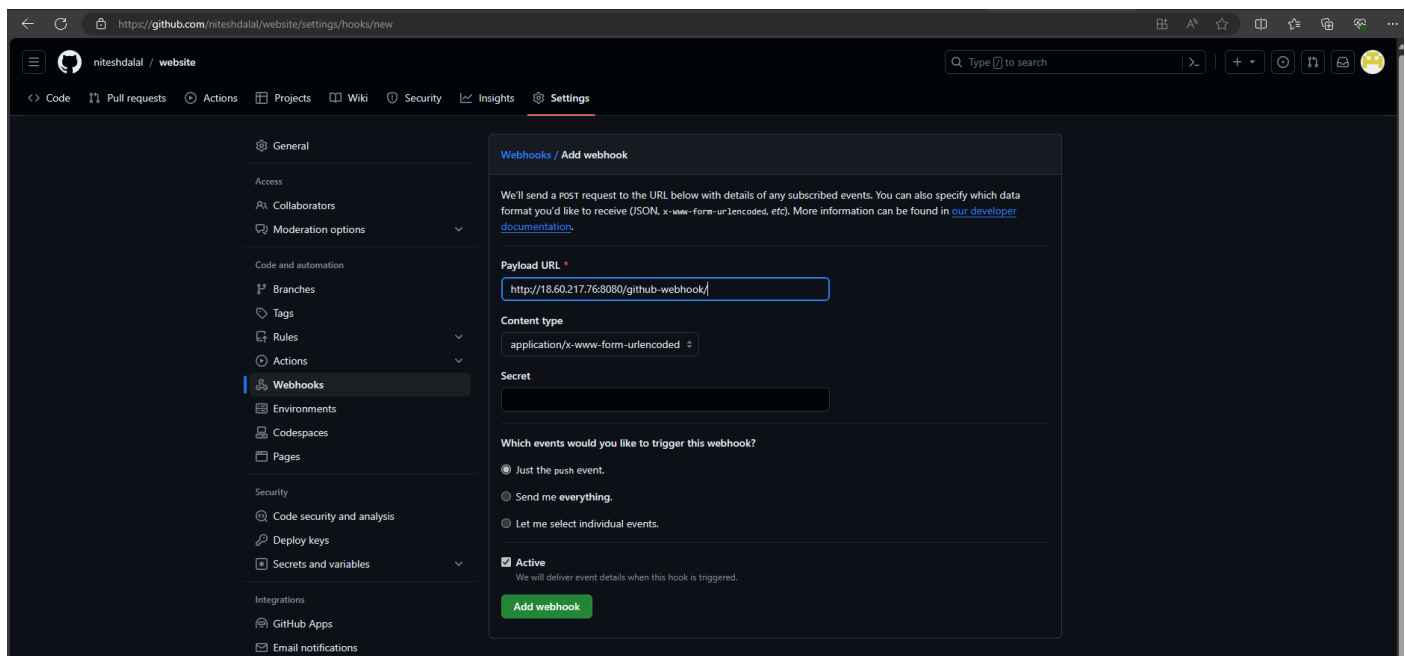


30. Now we need to re configure Job3 in a manner so that if Job2 is successful then Job3 should be automatically triggered. So, we need to enable post build actions.



31. Last thing remaining is to add webhook of github for Jenkins jobs.





32. That's all.