Project : Capstone I

DevOps Certification Training

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/website.git

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

Step1:

We have to Fork the repository under our account.

Create 3 t2.micro instances with Ubuntu 22.0 OS.
I Named them :

MainVm1: Main_machine/Jenkins_Master

Test-Slave: Slave1/Test

Prod-Slave: Slave2/Prod

Install Ansible on MainVm

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

As you can see below the ansible is installed in our MainVm

```
ubuntu@ip-172-31-3-210:~$ 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, Nov 20 2023, 15:14:05) [GCC 11.4.0] (/usr/bin/python3)
  jinja version = 3.0.3
  libyaml = True
ubuntu@ip-172-31-3-210:~$
```

Note: step2 is ssh-keygen in MainVm and pasted on all the slaves in .ssh location withing authorized file present by default.

```
ubuntu@ip-172-31-3-210:~$ 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:beiAdA5KcCbHICAMubaPnowJDYBjASl5nAiVUdW+rPc ubuntu@ip-172-31-3-210
The key's randomart image is:
+---[RSA 3072]----+
|^@*=....         |
|XBB     .        |
|=+. o ..         |
|++ o =  .o       |
|o o . o.S.o      |
| +     oo.       |
|. +    ..        |
|+o..  . .        |
|++     . .E      |
+----[SHA256]-----+
ubuntu@ip-172-31-3-210:~$
```

Step3: We will paste the MainVm public key in authorized_keys of other two slave machine in order to make reacheability.

```
ubuntu@ip-172-31-3-210:~/.ssh$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCnL/oIeTWxn87aHISX2ojGlI6Bnv/gMnVpyvGoRbBNUQGlAjJ+r+jhNj8bd7CuzW7UiVi35K4UxKdgRK/nnbdMNZjMx1AKVQNEMTNf0pE5yrEYbzxJlt7zNjqBD/
DZ76HK1WESdW/rLnbVEYdT7wjWmZpvR2n+T/CsTH/ZW13qSx/y32+u2o+9BoDBsCLMsQoz2McLeGKPeKAVrA6tv431u2u5Uwpj3RBVEWlQBHeqOddoMXg/kNfKdXtmr5BPlEc15zCZfqR89tYcXmH3sUxWGlauC+oq
bNwLnL4uWlyso/hSd9UVZOTB+ayTXNqn/ma9Z+N0/woXouoKNZrpe5XCTL7LL3EgAtP7TepYp1rgtBqA2XBzlC85jzdD1I/S1sSlK+GjkqYrlY1F3qc4NdNctDRYdim7zqCu1CII8LTz70DZeVNVv/NFO2HK651cV1
NcxWsFL+IloatIlhDOadfIvaSzjtMZLiL8RiOM3eqs9JhTIKt28idRfvQq4+rSxmE= ubuntu@ip-172-31-3-210
ubuntu@ip-172-31-3-210:~/.ssh$
```

Step4:Edit your host file on master and ad ip addrees of your slave in group method.

```
GNU nano 6.2
[Slaves]
172.31.7.229
172.31.11.183
```

Step5: As you can see our slaveVm ping able and reacheable

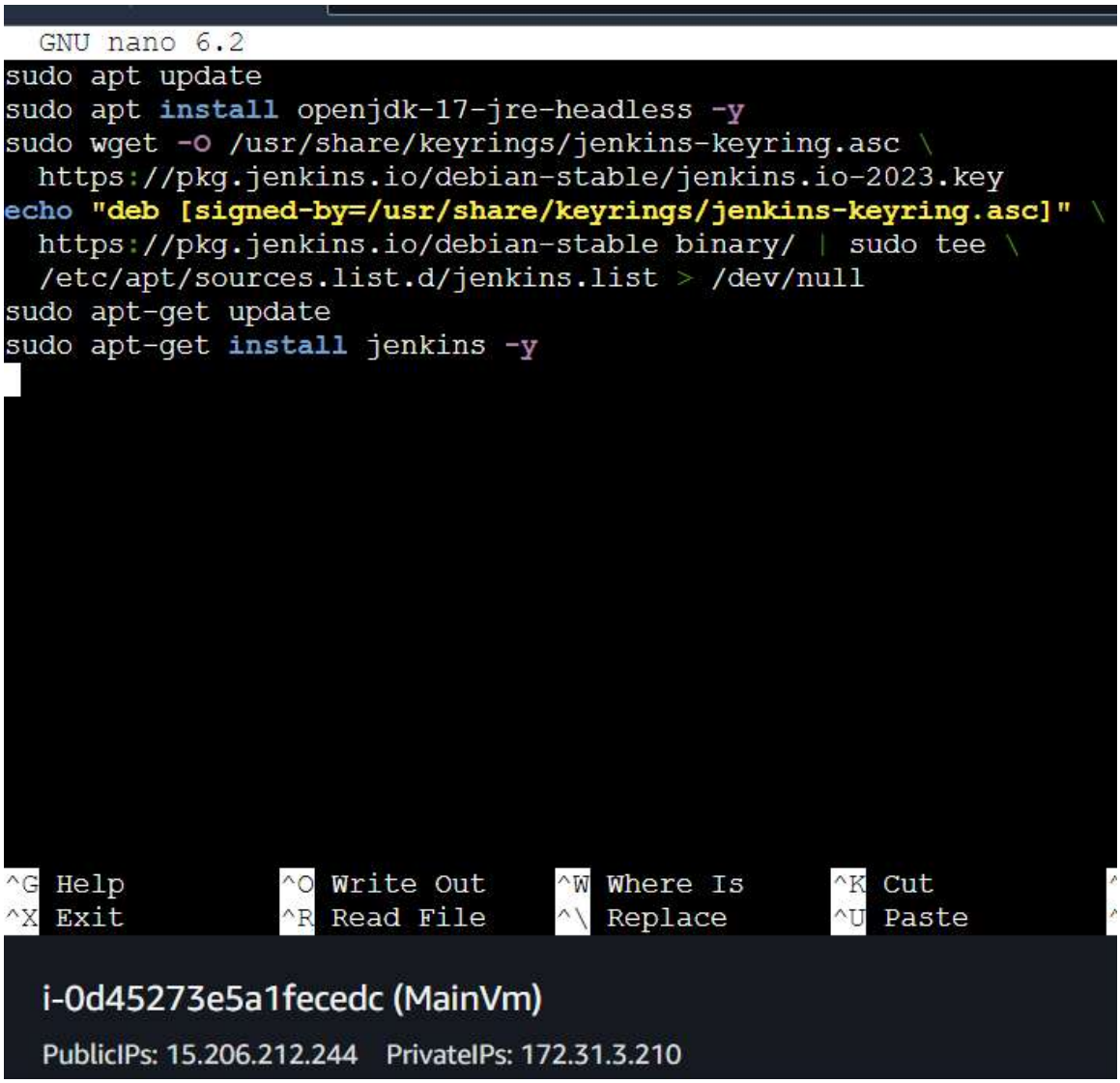
```
ubuntu@ip-172-31-3-210:~$ ansible -m ping all
172.31.7.229 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
172.31.11.183 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
ubuntu@ip-172-31-3-210:~$
```

Step6: Creation of playbook. In this playbook in we,ll write jenkins and java in MainVm. and Docker and java In SlavesVm

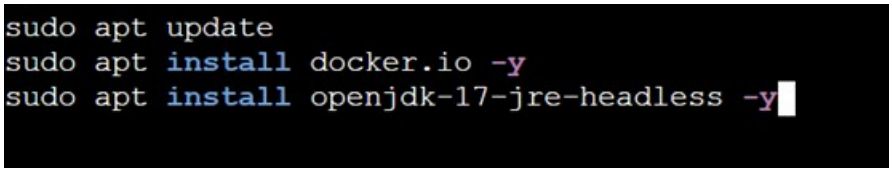
```
ubuntu@ip-172-31-3-210:/etc/ansible$ cat playbook.yml

---
#play1
- name: Task for MainVm
  hosts: localhost
  become: true
  tasks:
  - name: Install Jenkins and Java
    script: jenkins_master.sh

#play2
- name: Task for Slaves
  hosts: Slaves
  become: true
  tasks:
  - name: Install Docker and Java in both slaves
    script: slaves.sh
ubuntu@ip-172-31-3-210:/etc/ansible$
```

Step7: Create the Jenkins_master.sh and Slave.sh respectively because these are being called within playbook

Jenkins_master.sh



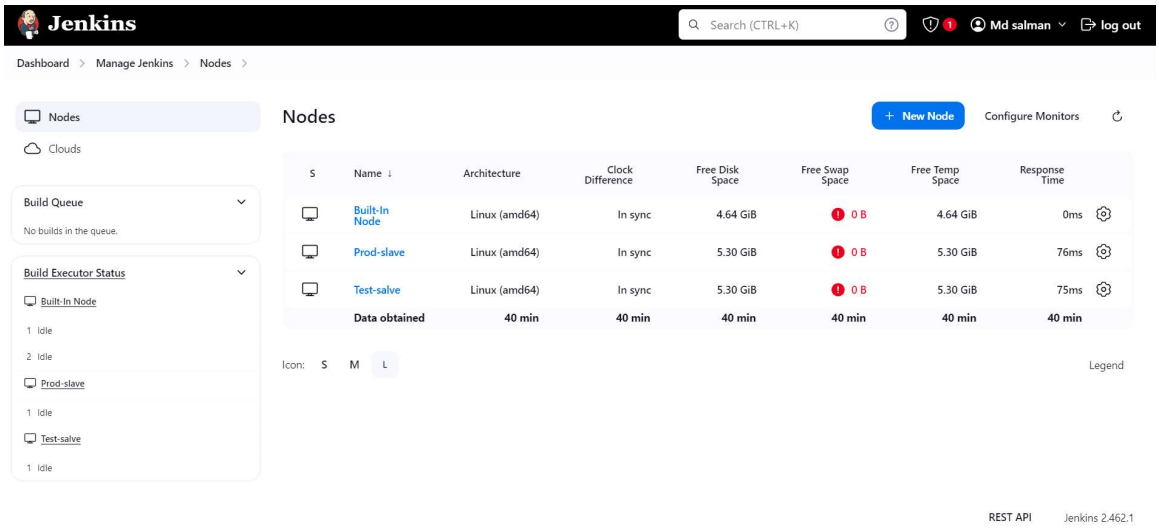
Step8: creation of slaves.sh



Step9: Jenkins and java, Docker and java created in Machines.

```
ubuntu@ip-172-31-3-210:/etc/ansible$ ansible-playbook playbook.yml

PLAY [Task for MainVm] ***********************************************************************************************************************************

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

TASK [Install Jenkins and Java] **************************************************************************************************************************
changed: [localhost]

PLAY [Task for Slaves] ***********************************************************************************************************************************

TASK [Gathering Facts] ***********************************************************************************************************************************
ok: [172.31.7.229]
ok: [172.31.11.183]

TASK [Install Docker and Java in both slaves] ************************************************************************************************************
changed: [172.31.7.229]
changed: [172.31.11.183]

PLAY RECAP ***********************************************************************************************************************************************
172.31.11.183              : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
172.31.7.229               : 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
```

Step10: Now we,ll create the nodes in order to make connection .



Node 1:This is My Test-slave where Dockerfile will be tested coming from develop branch (job1) and master branch (job2)

Node2: This prod-slave will run (job3) which is going to pull all the data from github and master branch create an image from Dockerfile and deploy it in the container.

Step11:We will clone the Git repository which is provide by Intellipaat

```
ubuntu@ip-172-31-3-210:~$ ls
project1
ubuntu@ip-172-31-3-210:~$ cd project1
ubuntu@ip-172-31-3-210:~/project1$ ls
Dockerfile  images  index.html
ubuntu@ip-172-31-3-210:~/project1$ cat Dockerfile

FROM ubuntu
RUN apt update
RUN apt install apache2 -y
ADD . /var/www/html/
ENTRYPOINT apachectl -D FOREGROUND
ubuntu@ip-172-31-3-210:~/project1$
```

As you can see we have fork the repo with the name project1.

and we created the Dockerfile which will create the image

Note: In Dockerfile we replacing index.html file with apache2 server in order to always up and run the image.

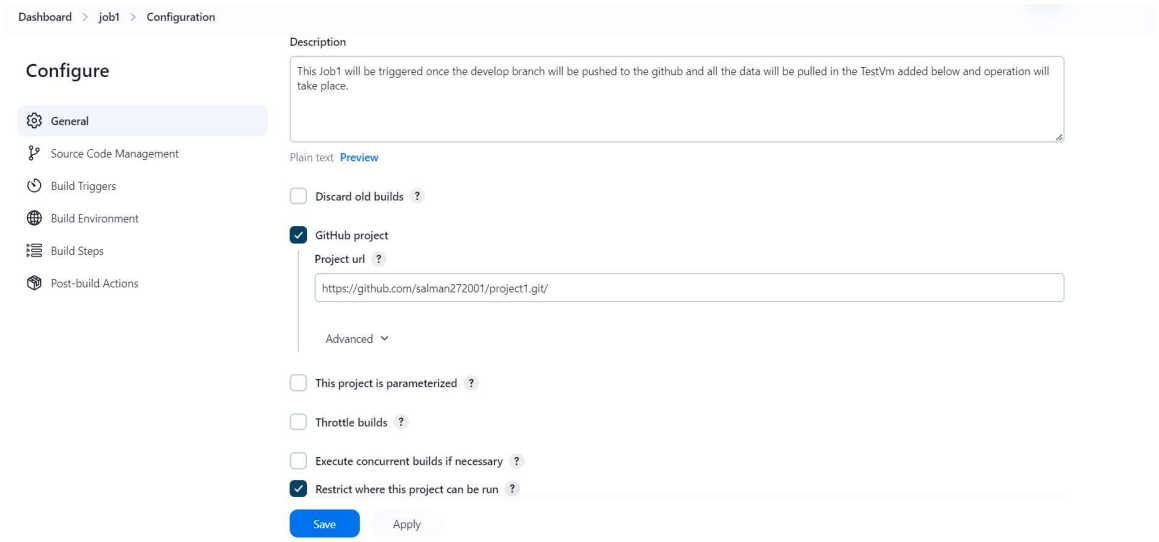
Note: but in the background it is using apache2 server to display the image.

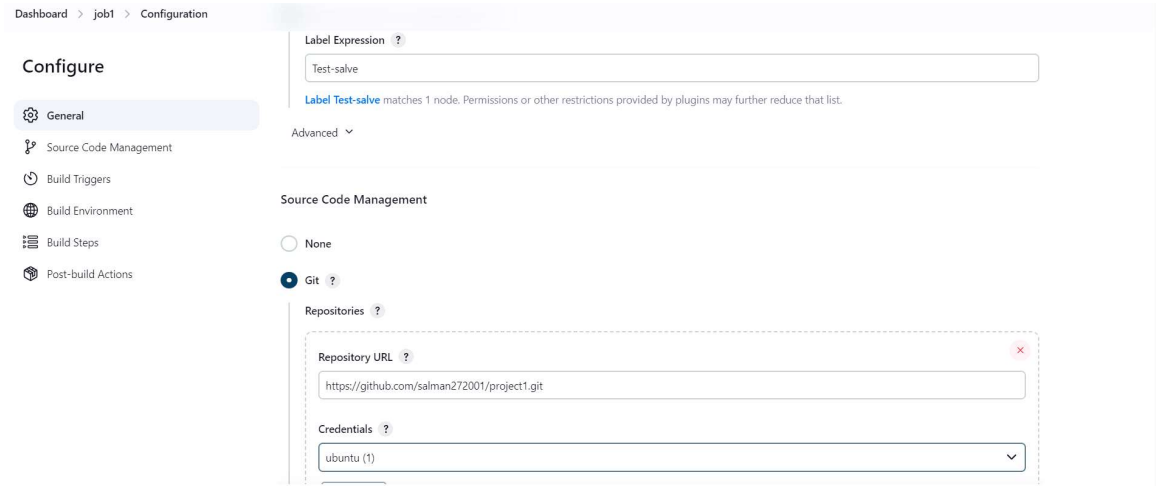
Step12: we will create the job1
This Job1 will be triggered once the develop branch will be pushed to the github and all the data will be pulled in the TestVm added below and operation will take place.

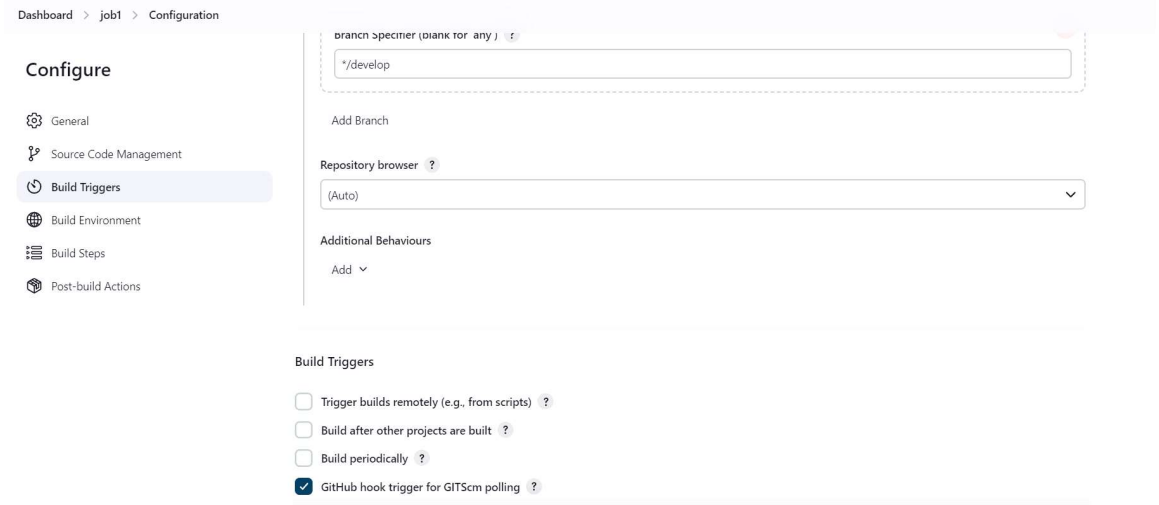
Pass github > Test machine > Develop > Select "github hook trigger GITdcm Polling" > Build Step: sudo docker build /home/ubuntu/Jenkins/workspace/jobname/ -t image_name
Apply and Save the job

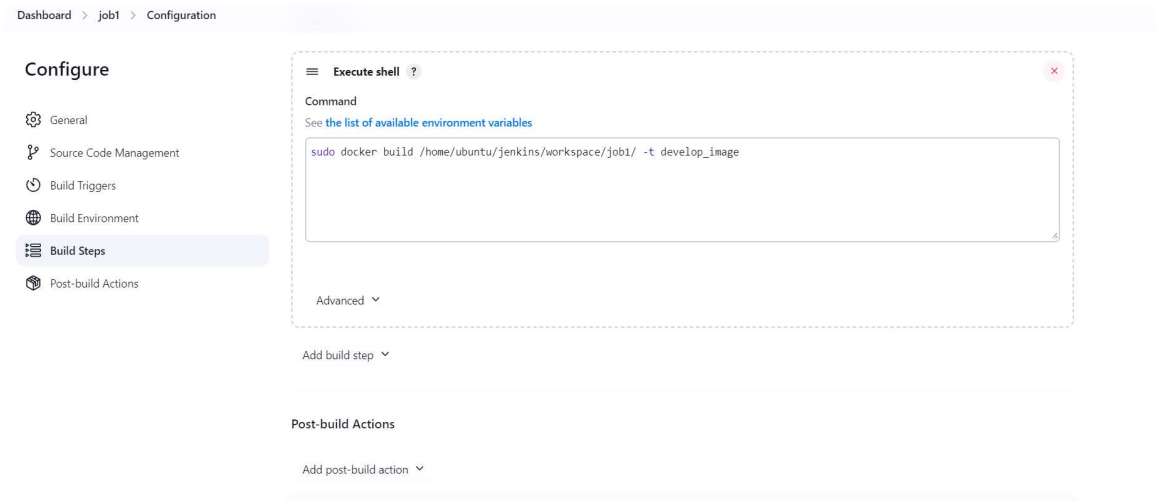
You will go to the githu repo>settings > add webhook trigger.

***Jenkins job1 configuration.**


Dashboard > job1 > Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
Description
This Job1 will be triggered once the develop branch will be pushed to the github and all the data will be pulled in the TestVm added below and operation will take place.
Plain text Preview
Discard old builds
GitHub project
Project url
https://github.com/salman272001/project1.git/
Advanced
This project is parameterized
Throttle builds
Execute concurrent builds if necessary
Restrict where this project can be run
Save
Apply

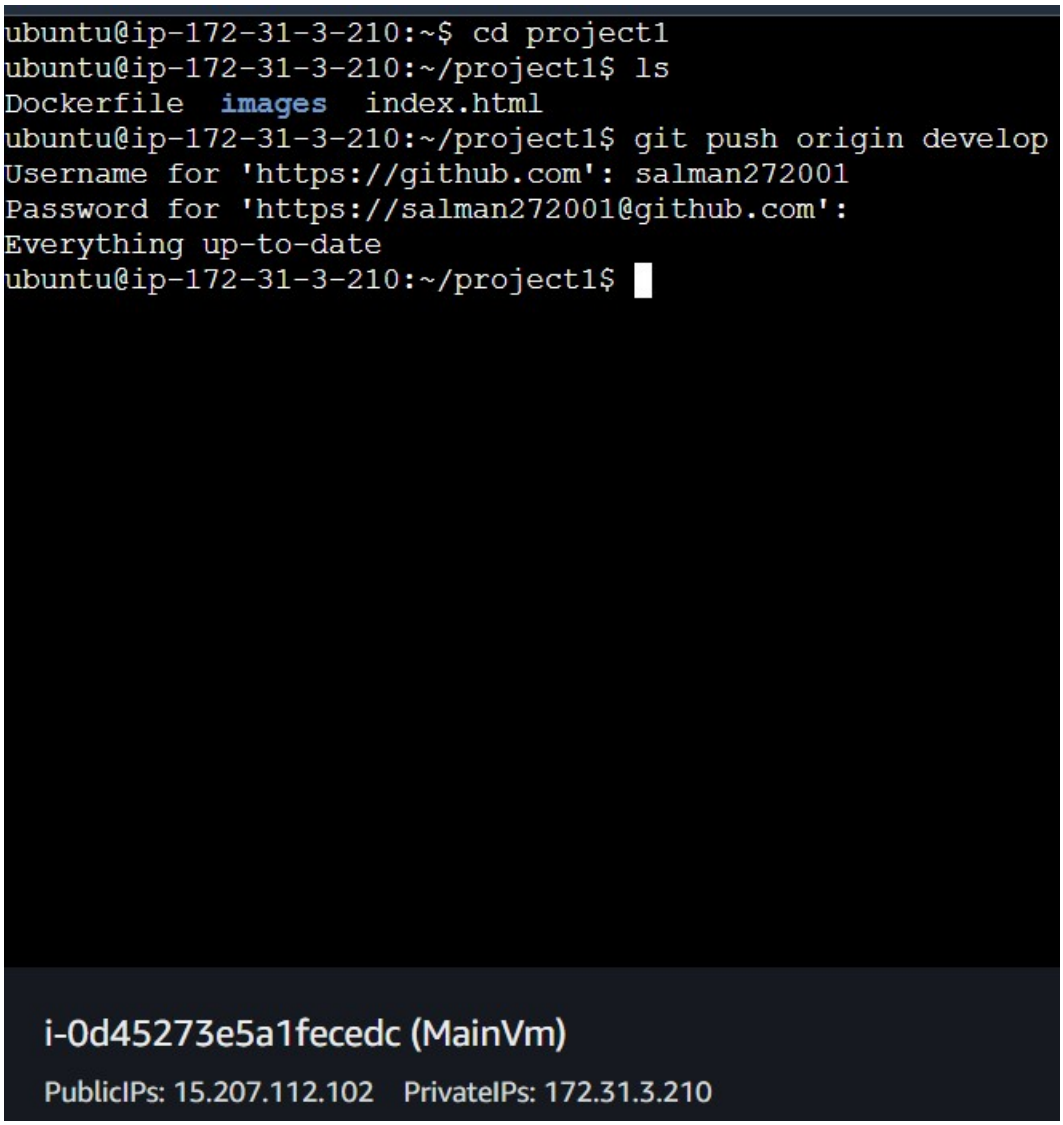

Dashboard > job1 > Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
Label Expression
Test-salve
Label Test-salve matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.
Advanced
Source Code Management
None
Git
Repositories
Repository URL
https://github.com/salman272001/project1.git
Credentials
ubuntu (1)

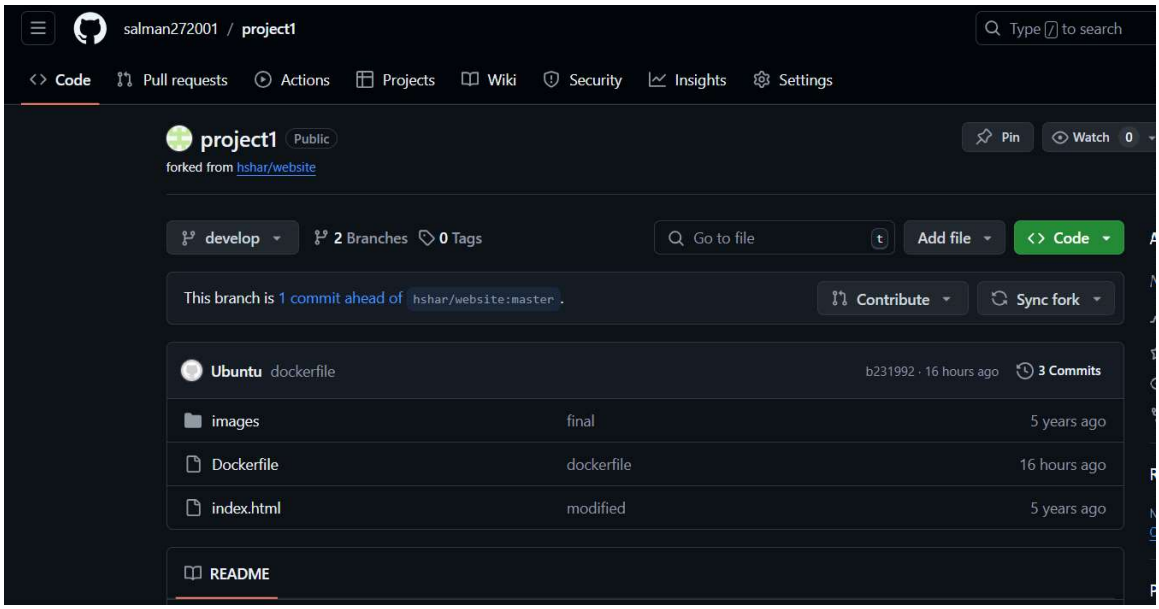

Dashboard > job1 > Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
*/develop
Add Branch
Repository browser
(Auto)
Additional Behaviours
Add
Build Triggers
Trigger builds remotely (e.g., from scripts)
Build after other projects are built
Build periodically
GitHub hook trigger for GITScm polling



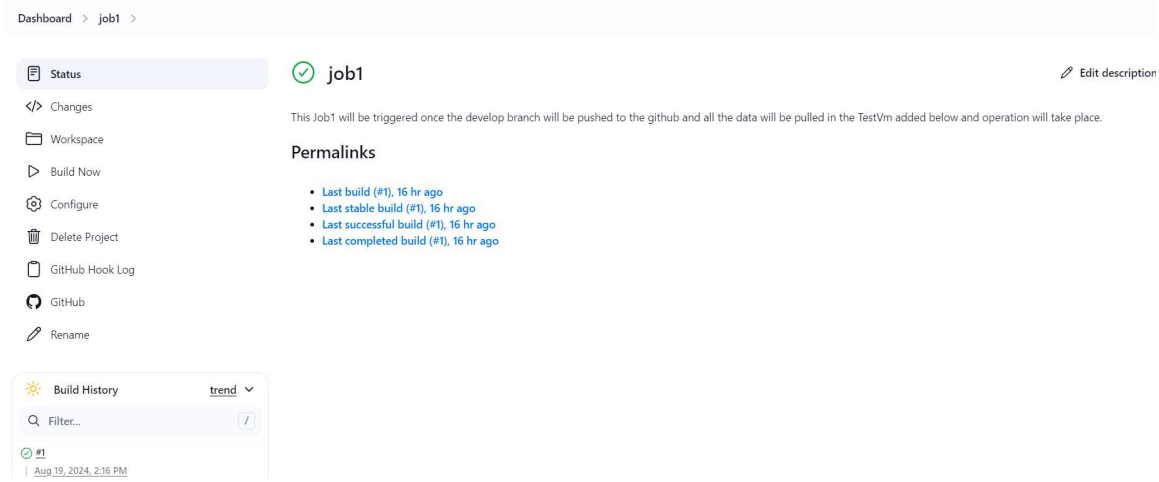
Note: when we push the develop branch in github Job1 will be triggered and the files will move to Test-slaveVm as we configured in jenkins dashboard.

Step13:pushing the develop branch in github.

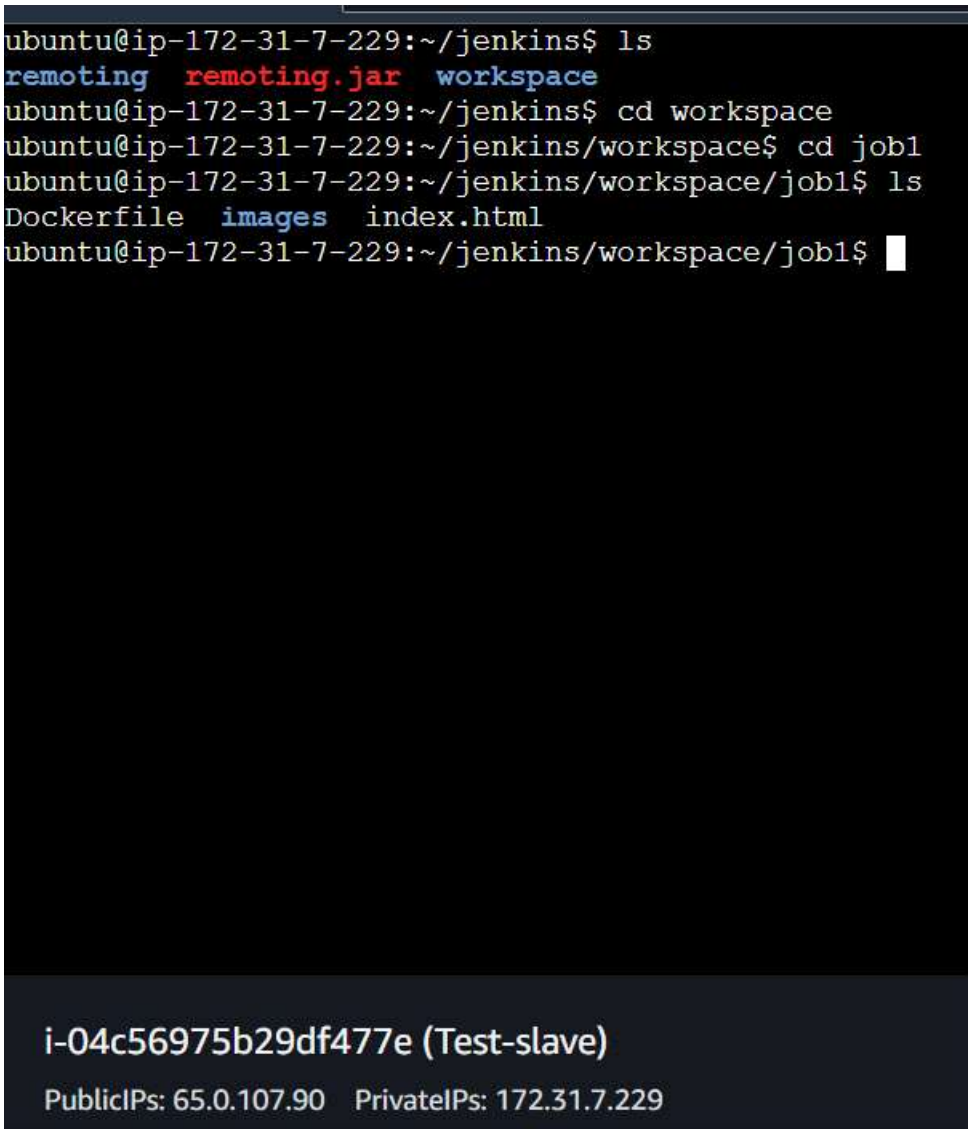

ubuntu@ip-172-31-3-210:~$ cd project1
ubuntu@ip-172-31-3-210:~/project1$ ls
Dockerfile images index.html
ubuntu@ip-172-31-3-210:~/project1$ git push origin develop
Username for 'https://github.com': salman272001
Password for 'https://salman272001@github.com':
Everything up-to-date
ubuntu@ip-172-31-3-210:~/project1$
i-0d45273e5a1fecedc (MainVm)
PublicIPs: 15.207.112.102 PrivateIPs: 172.31.3.210



As you can see the develop branch is pushed to the github our files are present there. and also our job1 ran succcessfully it triggered.



Step14: checking in Test-slave Vm the moved successfully and docker image created.




```
develop_image          latest    8f1531a164fa   17 hours ago   224MB
ubuntu                 latest    edbfe74c41f8   2 weeks ago    78.1MB
ubuntu@ip-172-31-7-229:~/jenkins/workspace/job1$
```

Step15:

Creation of job2 and job3

Job2 Upstream

Pass github > Test machine > Master > Select “github hook trigger GITdcm Polling” > Build Step: sudo docker build /home/ubuntu/Jenkins/workspace/jobname/ -t image_name
Apply and Save the job

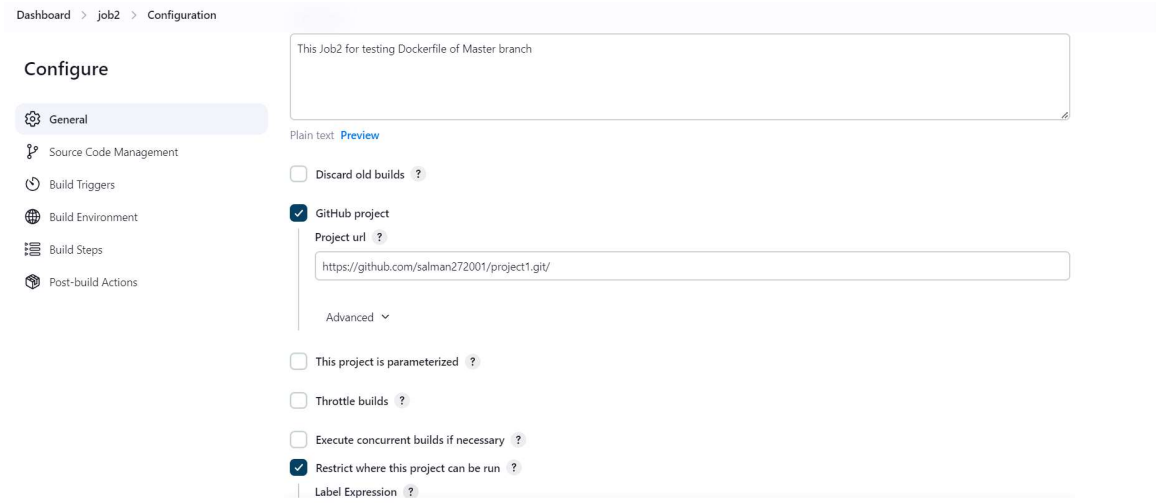
Job3 Downstream

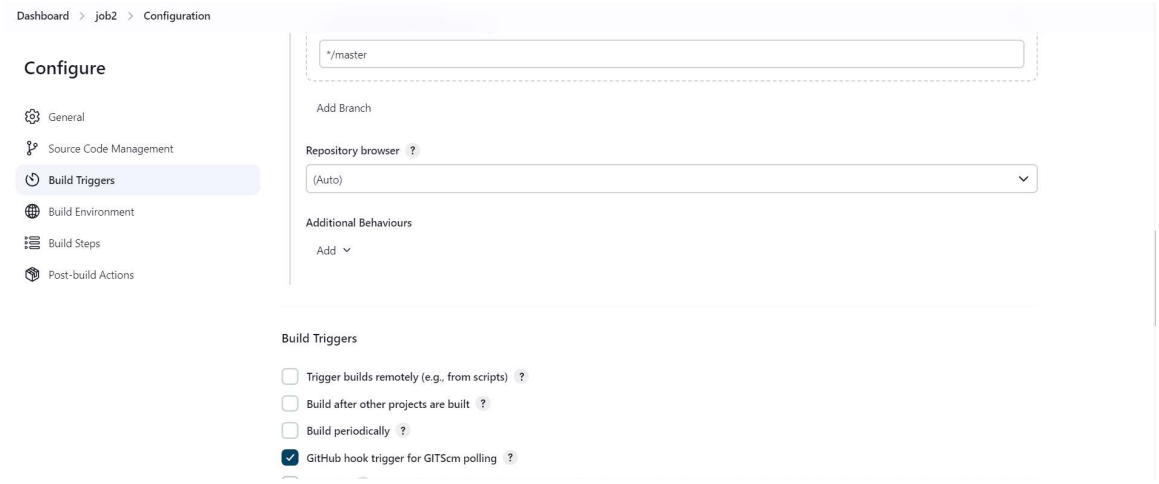
Pass github > Prod machine > Build Step:

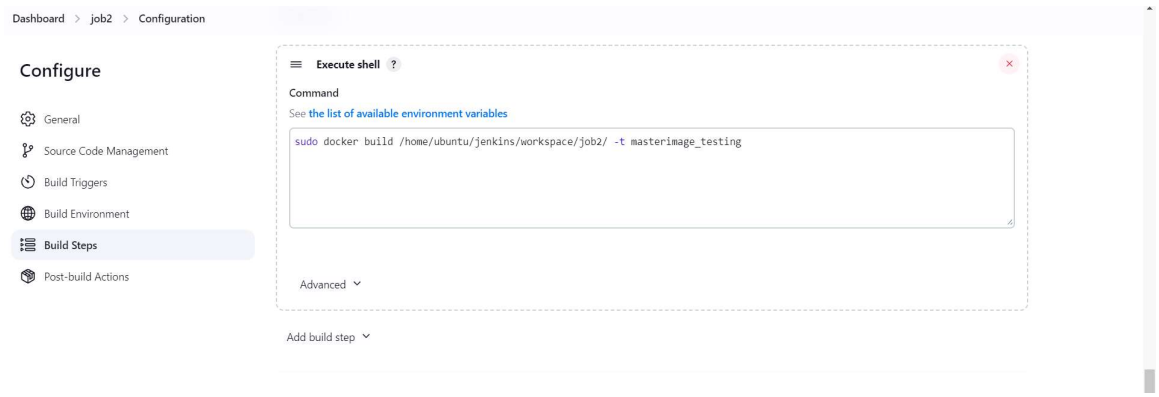
sudo docker build /home/ubuntu/Jenkins/workspace/jobname/ -t image_name

sudo docker run -itd –name container_name -p 80:80 image_name
Apply and Save the job

You will go to job 2 > Configure> Post build action and add Job 3 as “build other projects”

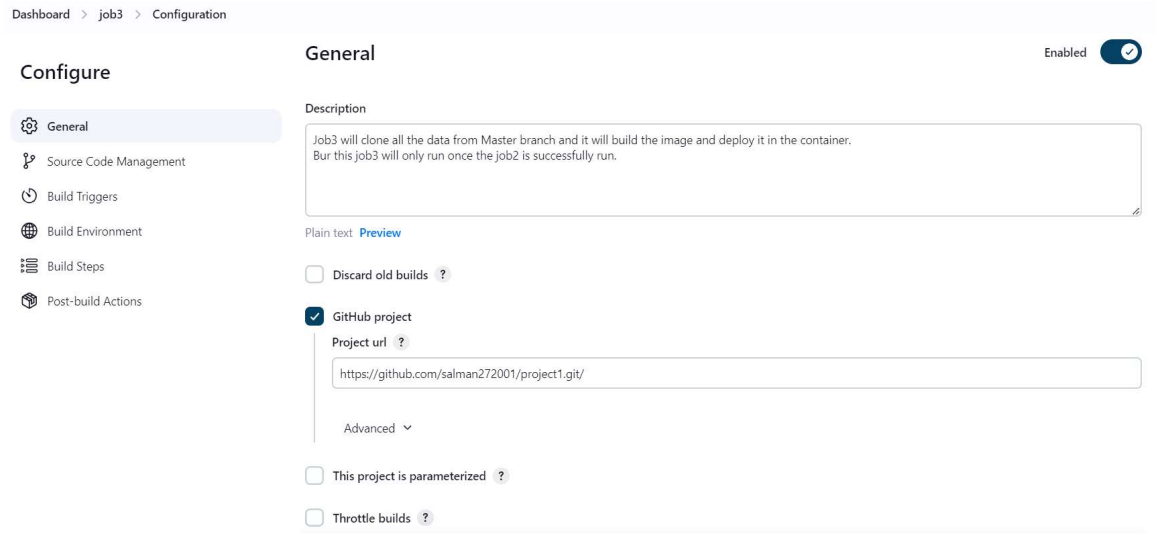


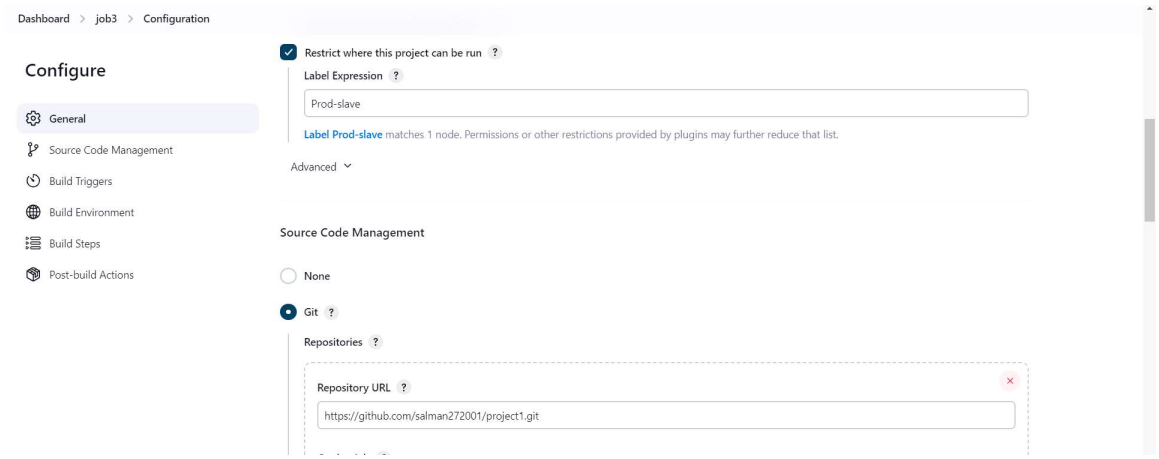


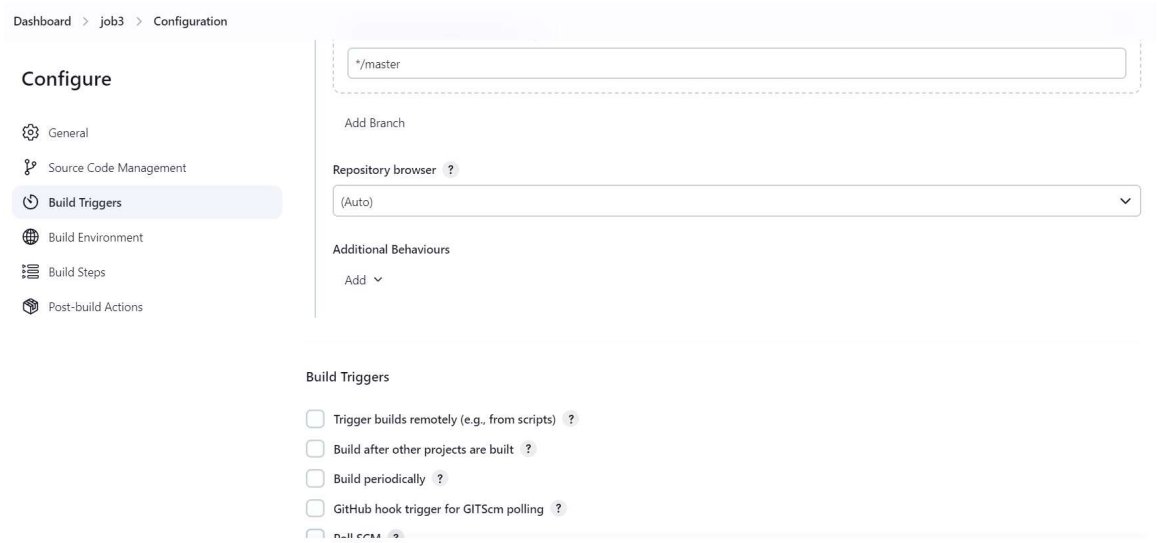


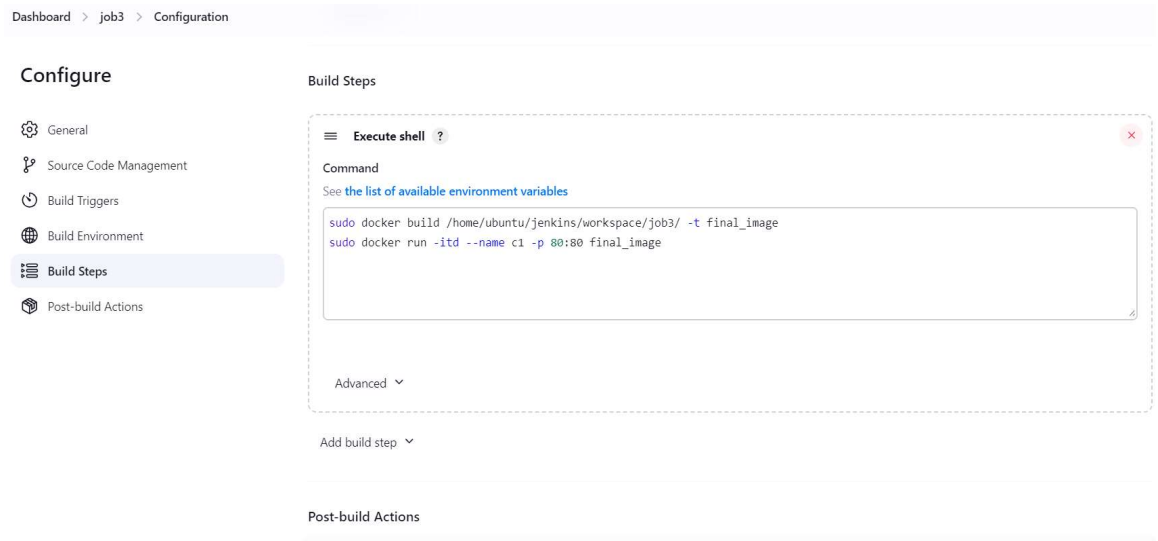
Note:This Above Job2 is for Testing Dockerfile of Master branch

And now we will create Job3





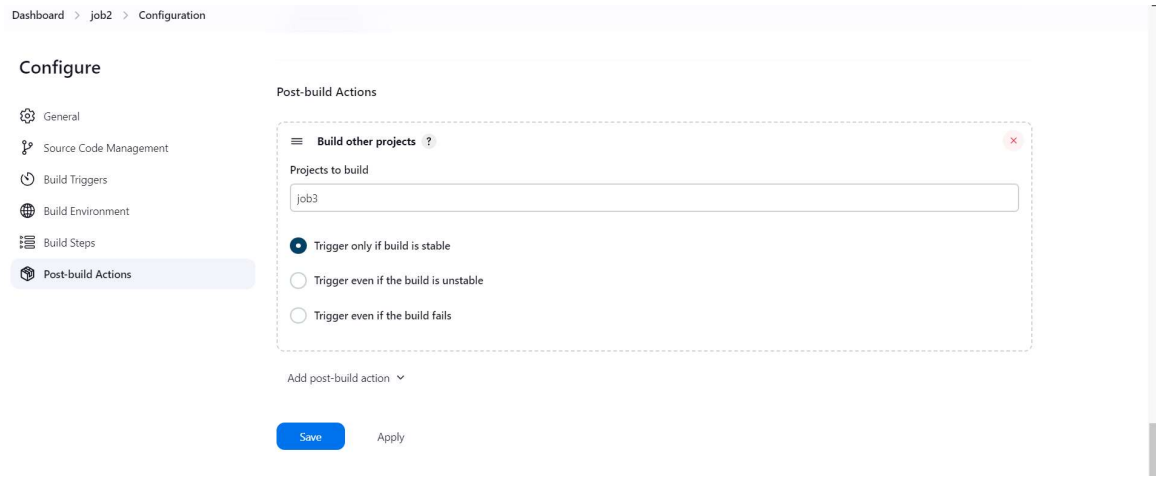




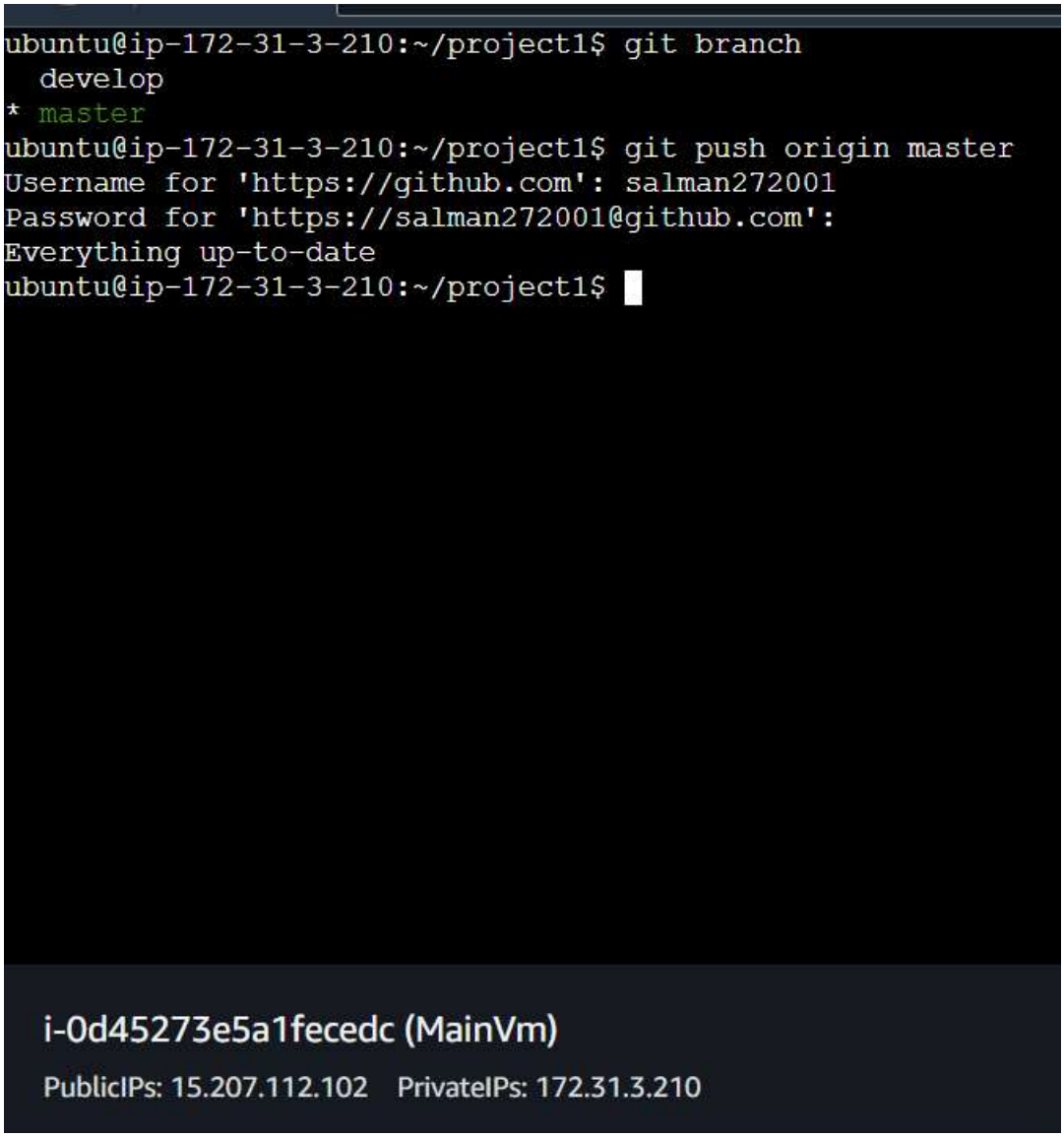
***We created the job2 and job3**

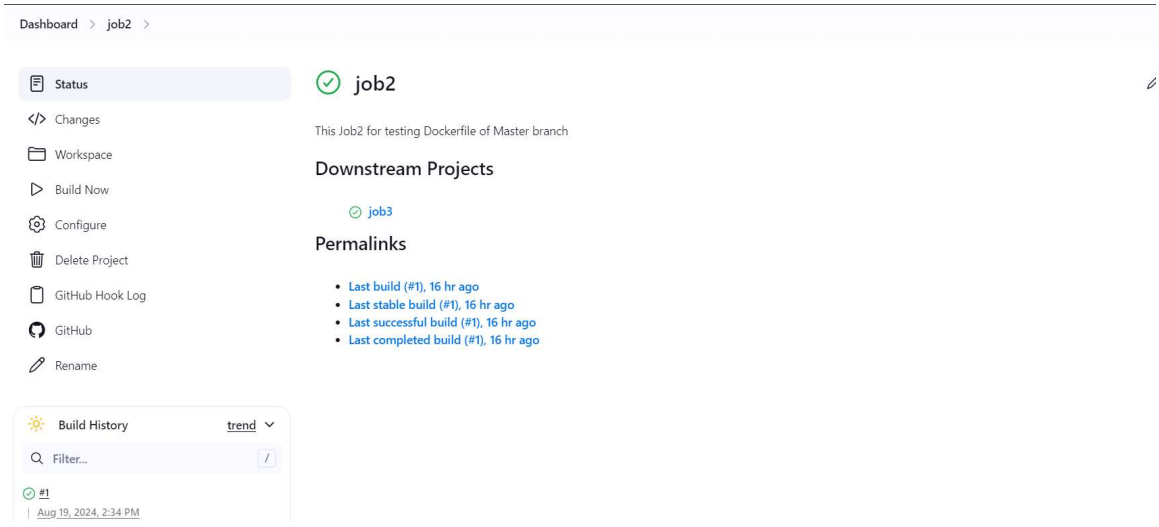
Note:job3 will only run once if the job2 is successfully run.

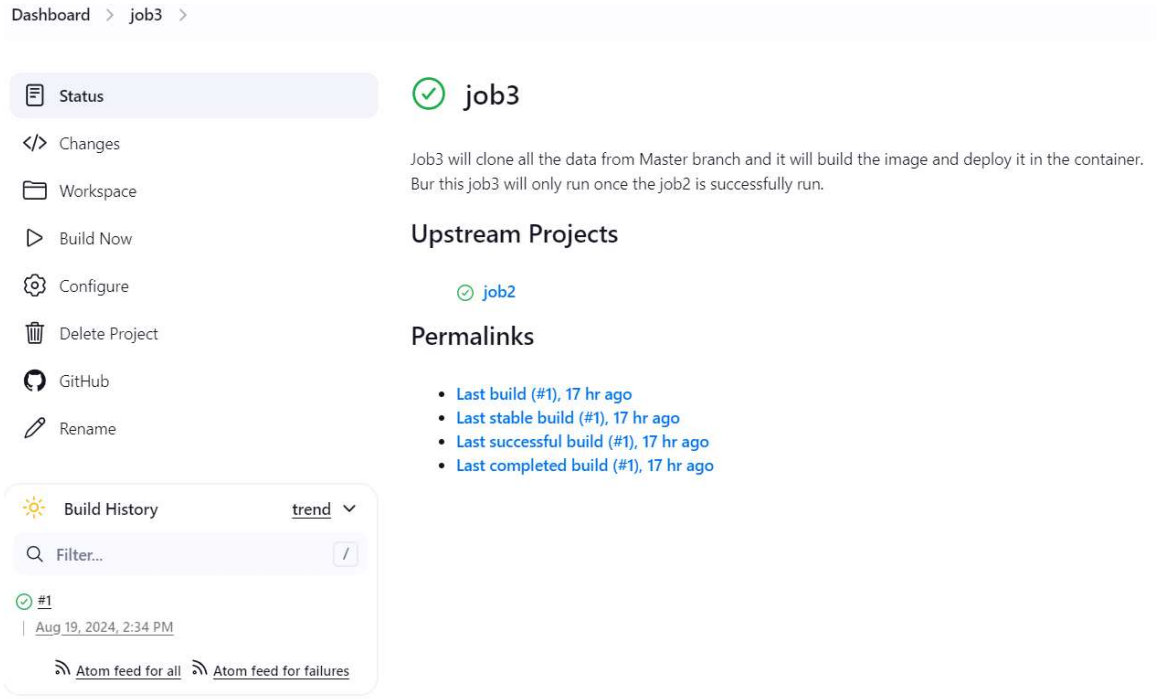
Step16: Now in job2 we will add the post build action in order to run the job3



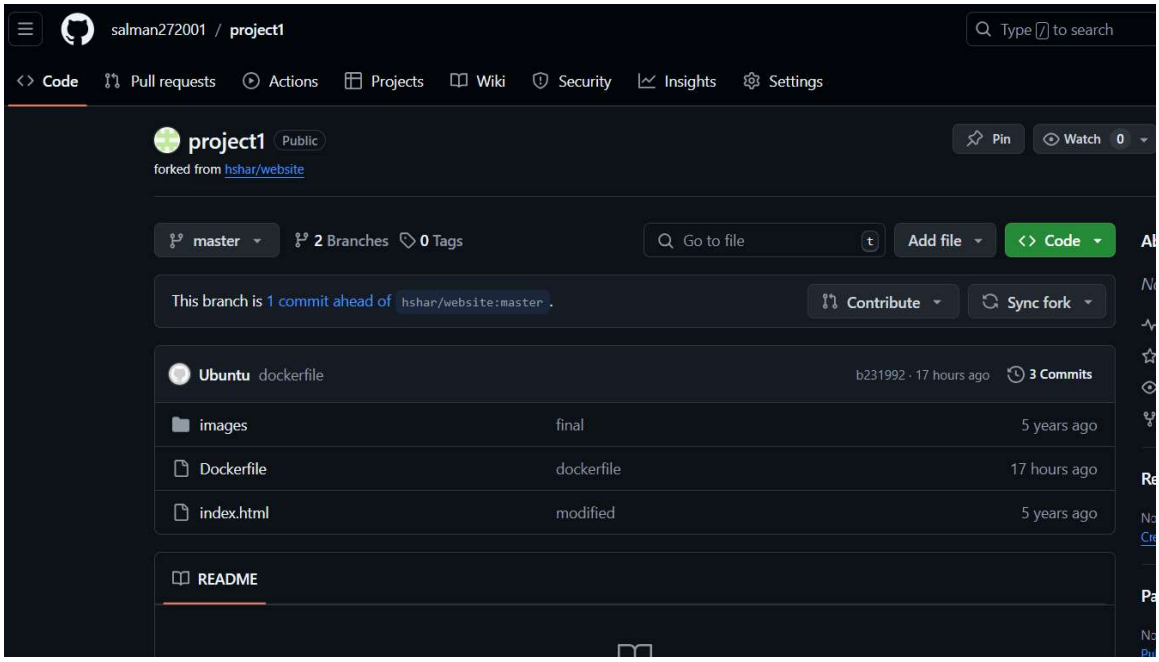
Step17: Now we will push the master branch in github and job3 will clone all the data in Prod-slave Vm and it will deploy the image in container.


ubuntu@ip-172-31-3-210:~/project1$ git branch
develop
* master
ubuntu@ip-172-31-3-210:~/project1$ git push origin master
Username for 'https://github.com': salman272001
Password for 'https://salman272001@github.com':
Everything up-to-date
ubuntu@ip-172-31-3-210:~/project1$
i-0d45273e5a1fecedc (MainVm)
PublicIPs: 15.207.112.102 PrivateIPs: 172.31.3.210



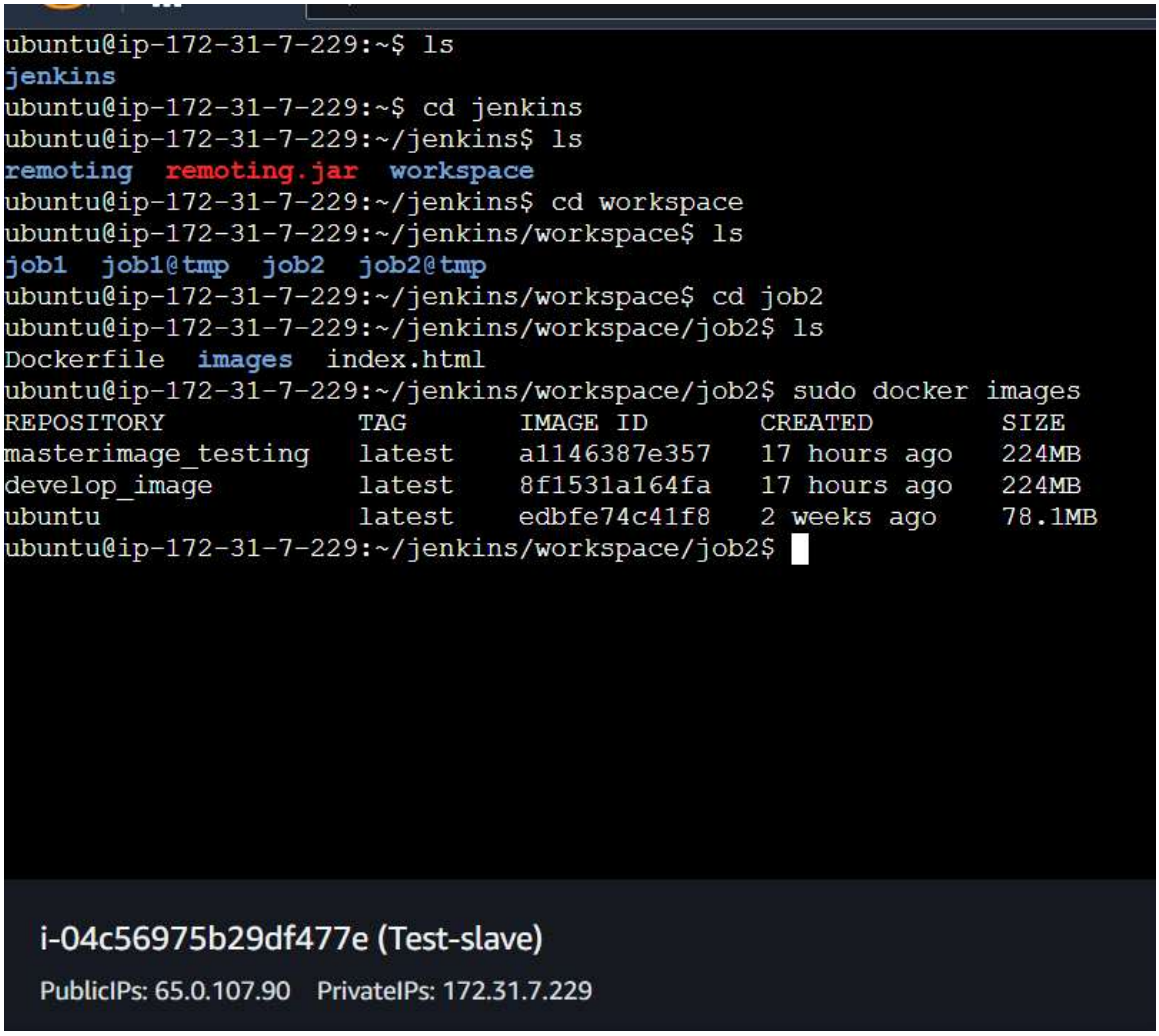


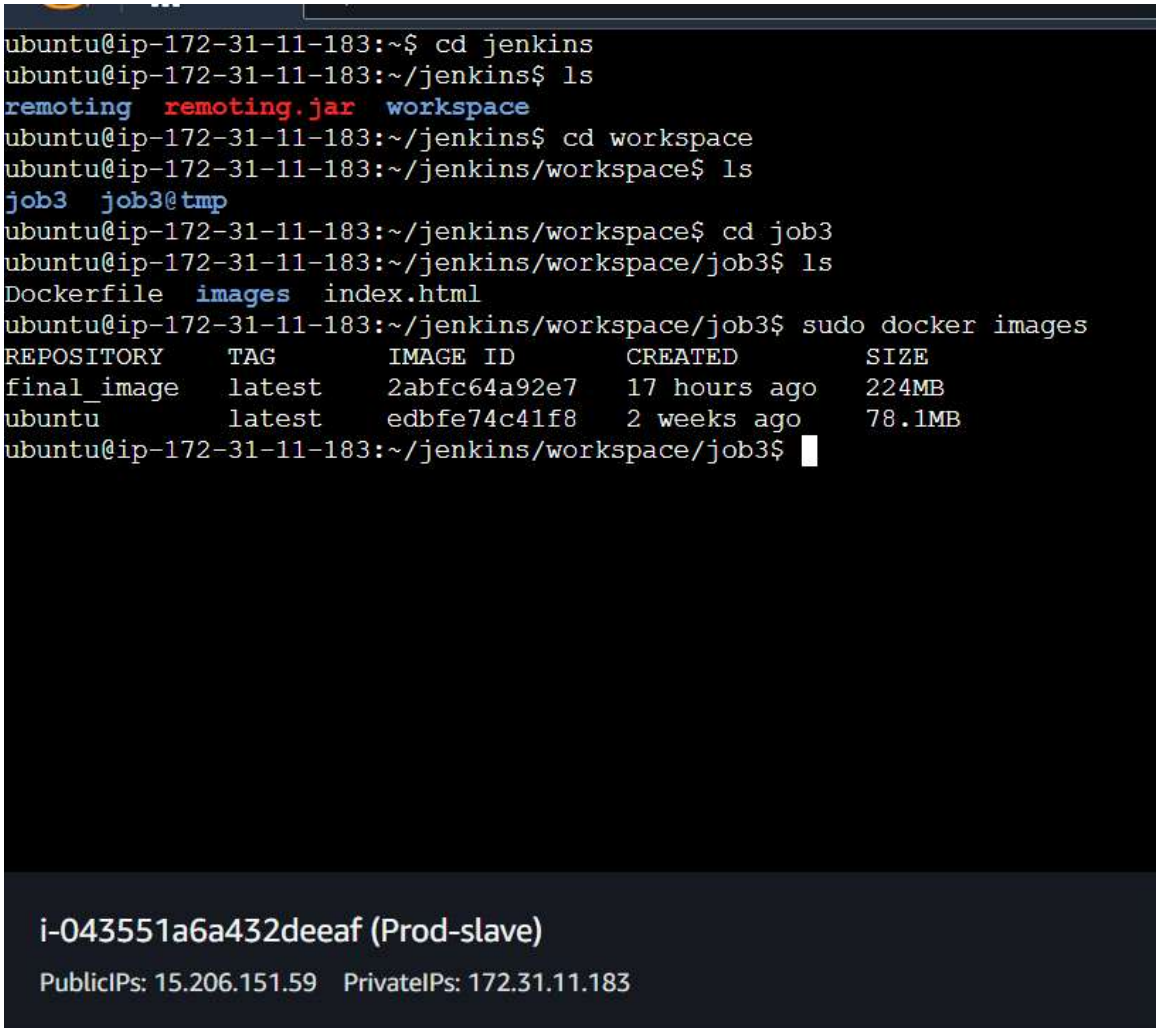
***As you can see job2 and job3 successfully run.**



***As you can see The master branch pushed and Our files are available in github dashboard**

Step18: checking in Test-slave and Prod-slave Vm are files cloned or not.


ubuntu@ip-172-31-7-229:~$ ls
jenkins
ubuntu@ip-172-31-7-229:~$ cd jenkins
ubuntu@ip-172-31-7-229:~/jenkins$ ls
remoting remoting.jar workspace
ubuntu@ip-172-31-7-229:~/jenkins$ cd workspace
ubuntu@ip-172-31-7-229:~/jenkins/workspace$ ls
job1 job1@tmp job2 job2@tmp
ubuntu@ip-172-31-7-229:~/jenkins/workspace$ cd job2
ubuntu@ip-172-31-7-229:~/jenkins/workspace/job2$ ls
Dockerfile images index.html
ubuntu@ip-172-31-7-229:~/jenkins/workspace/job2$ sudo docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
masterimage_testing latest a1146387e357 17 hours ago 224MB
develop_image latest 8f1531a164fa 17 hours ago 224MB
ubuntu latest edbfe74c41f8 2 weeks ago 78.1MB
ubuntu@ip-172-31-7-229:~/jenkins/workspace/job2$
i-04c56975b29df477e (Test-slave)
PublicIPs: 65.0.107.90 PrivateIPs: 172.31.7.229



***As you can see above the respective file successfully cloned in the Vm,s**

Step19: Copy the Ip address of Prod-slace and check on you local.

***And the image is successfully displayed on local**