Capstone-project-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/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:

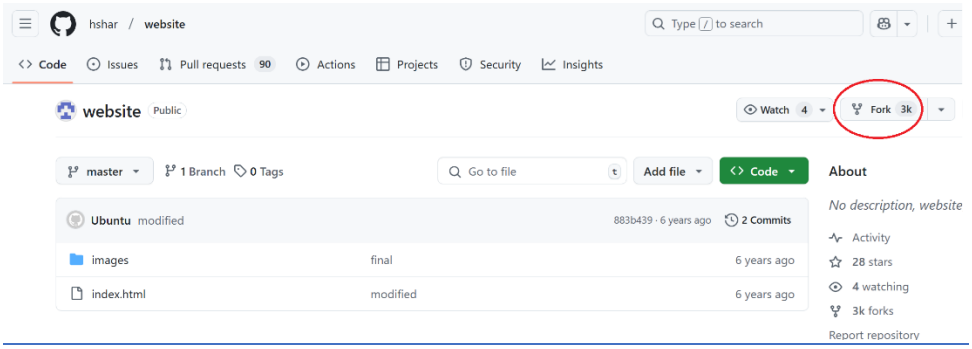
a. Job1 : build

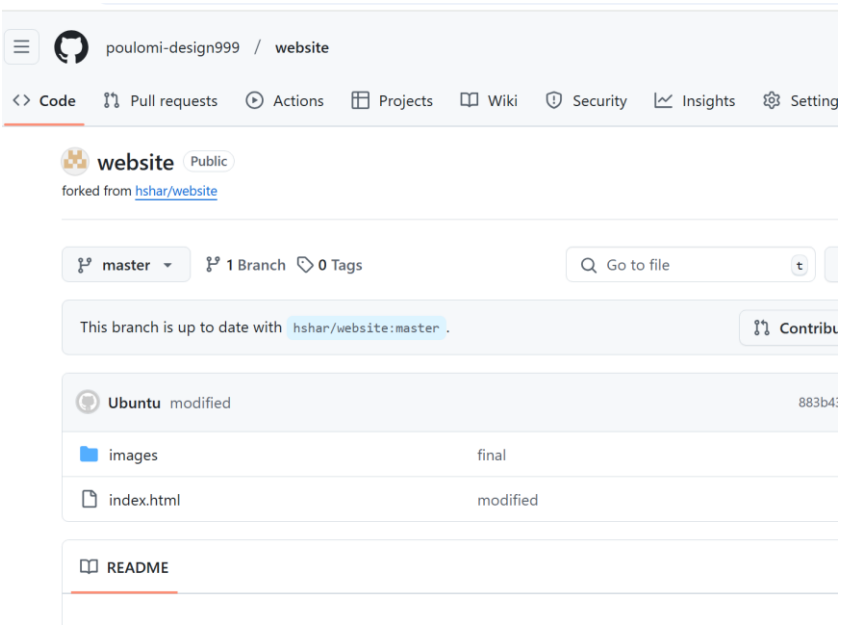
b. Job2 : test

c. Job3 : prod

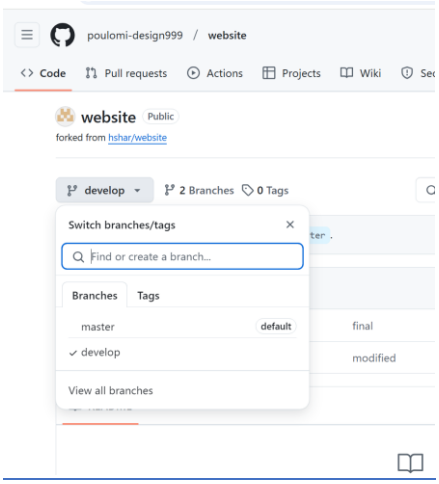
Solution:

1) **Fork the github link https://github.com/hshar/website.git**

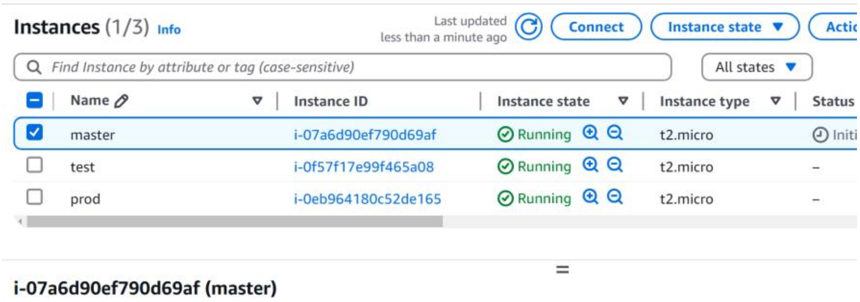




[Create a develop branch :](#)



2) **[Create 3 EC2 [test, prod and master] > connect and update all of them](#)**



3) **Install ansible on master > cd .ssh > ssh-keygen > key is generated> view the file > copy the contents**
$ sudo apt update
$ sudo apt install software-properties-common
$ sudo add-apt-repository --yes --update ppa:ansible/ansible
$ sudo apt install ansible

```
ubuntu@ip-172-31-4-35:~$ cd .ssh
ubuntu@ip-172-31-4-35:~/.ssh$ ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_ed25519
Your public key has been saved in /home/ubuntu/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:4EaUlpD2BrG6roWzvuOhaEaojYgq5ABc48W2MZvIiU8 ubuntu@ip-172-31-4-35
The key's randomart image is:
+--[ED25519 256]--+
|     ++.o        |
|    o+O+         |
|. =.B+O          |
|.o E *o.         |
|o +  .o S        |
|o+ o .           |
|@+o              |
|OXo              |
|&=+              |
+----[SHA256]-----+
ubuntu@ip-172-31-4-35:~/.ssh$ sudo cat /home/ubuntu/.ssh/id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAINMaVWeWStWr7W8w/qymIH3Aop4mYHtjsjeKO/Df/Rh1 ubuntu@ip-172-31-4-35
ubuntu@ip-172-31-4-35:~/.ssh$
```

4) **Go to test and prod > cd.ssh > ls > vi authorized_keys > paste the content of the key generated in master**

```
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAINMaVWeWStWr7W8w/qymIH3Aop4mYHtjsjeKO/Df/Rh1 ubuntu@ip-172-31-4-35
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC63el4F/k3VhQjRUF98omlJB/WN86ZjDoiIb0gKpJmkpNTNPMta90/C/GRnXe8CPBFKYgrQPdPffT9hS
xbz6kZCp6WZ0wKmMql90JLXvJueJDwuLvEBJmeopW59d+h53UBo7qru3k0LdWh22BjA2zwHAmUWdEcWBNzd9aGQ323ZVdJgcR2ammb7zyEpeDryVqCmQ8J
EueEp/UmsRdCD7p2ewUEKHQS4HVHyOKAM4v7+VV0XuZFOsoohUTIiJwWMF9zFiwE0EiVFJ/+SVZlUJhR molly
~
~
~
-- INSERT --
```

```
ubuntu@ip-172-31-8-1:~$ cd .ssh
ubuntu@ip-172-31-8-1:~/.ssh$ ls
authorized_keys
ubuntu@ip-172-31-8-1:~/.ssh$ vi authorized_keys
ubuntu@ip-172-31-8-1:~/.ssh$
```

5) **Go to master server > cd /etc/ansible > ls > vi hosts**

```
ubuntu@ip-172-31-4-35:~/.ssh$ cd
ubuntu@ip-172-31-4-35:~$ cd /etc/ansible
ubuntu@ip-172-31-4-35:/etc/ansible$ ls
ansible.cfg  hosts  roles
ubuntu@ip-172-31-4-35:/etc/ansible$ vi hosts
```

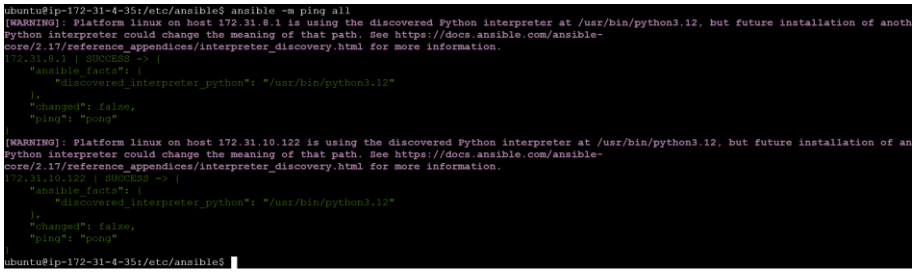
i-0d50e93fdc45520e7 (master)

6) **Copy the private IP of slave servers [test and prod]**

```
## [dbservers]
##
## db01.intranet.mydomain.net
## db02.intranet.mydomain.net
## 10.25.1.56
## 10.25.1.57


# Ex4: Multiple hosts arranged into groups

## [Debian]
## alpha.example.org
## beta.example.org

## [openSUSE]
## green.example.com
## blue.example.com
[slave]
172.31.10.122
172.31.8.1
-- INSERT --
```

i-0d50e93fdc45520e7 (master)

7) **Check the connection between master and slave servers**
ansible -m ping all



8) **Now we create a playbook vi play.yml in master with the following code . according to it the script master.sh and slaves.sh will run when we play this playbook.**

```
---
- name: task for master mc
  hosts: localhost
  become: true
  tasks:
  - name: master script
    script: master.sh
- name: task for slave mc
  hosts: slave
  become: true
  tasks:
  - name: slaves script
    script: slaves.sh
```

9) **We create master.sh and slaves.sh [script file] which will run in playbook. Inside this script all the software dependencies will be downloaded as per the problem statement.**

master.sh : install java & install jenkins

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

slaves.sh : install java & docker

```
sudo apt update
sudo apt install openjdk-17-jre -y
sudo apt install docker.io -y
```

10) Check the playbook syntax

```
ubuntu@ip-172-31-4-35:~$ ansible-playbook play.yml --syntax-check

playbook: play.yml
ubuntu@ip-172-31-4-35:~$
```

11) Check for errors , dry run playbook

```
ubuntu@ip-172-31-4-35:~$ ansible-playbook play.yml --check
```

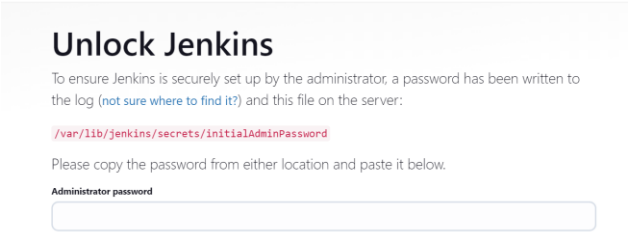
```
PLAY RECAP *********************************************************************************************************
172.31.10.122              : ok=1    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
172.31.8.1                 : ok=1    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
localhost                  : ok=1    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
```

12) Execute playbook

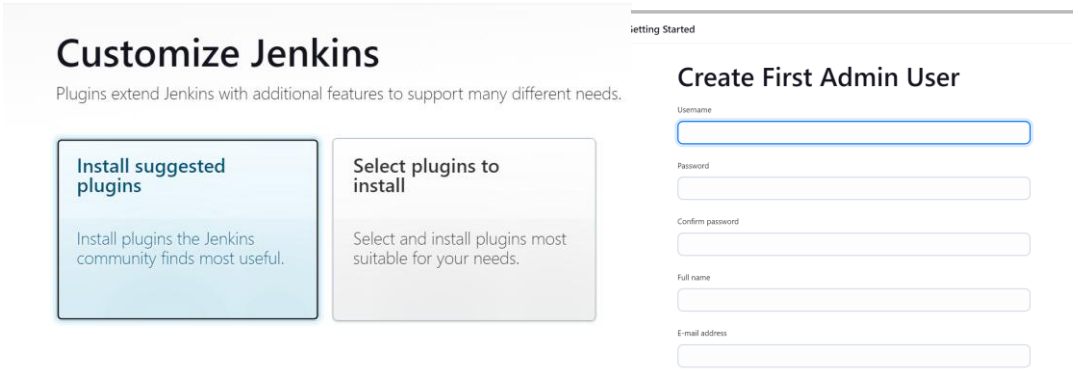
```
ubuntu@ip-172-31-4-35:~$ ansible-playbook play.yml
```

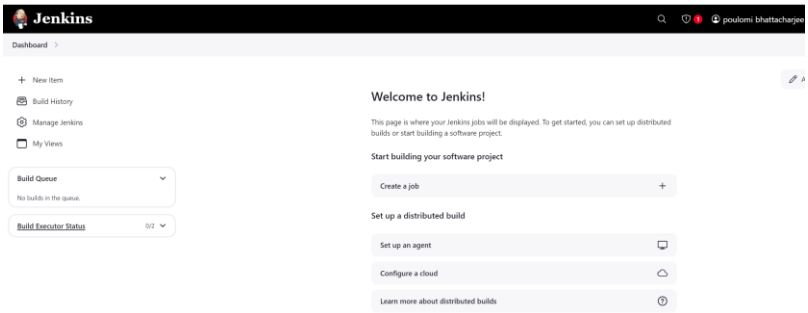
Access Jenkins

1) Copy and paste the public IP:8080 (master) on the browser and we can see unlock Jenkins page

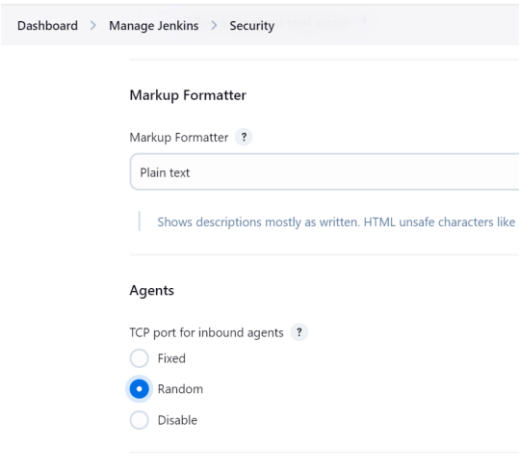


2) Unlock Jenkins > download plugins > create a user > we can access Jenkins dashboard

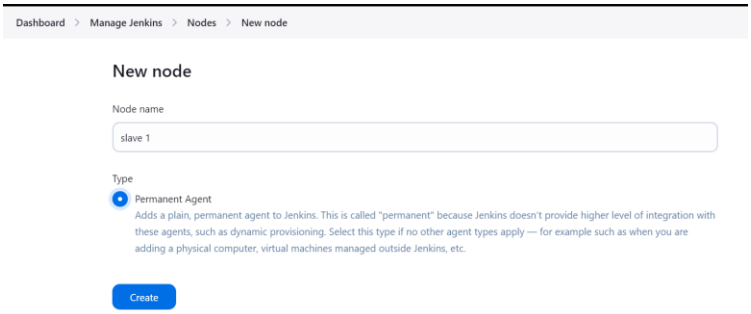




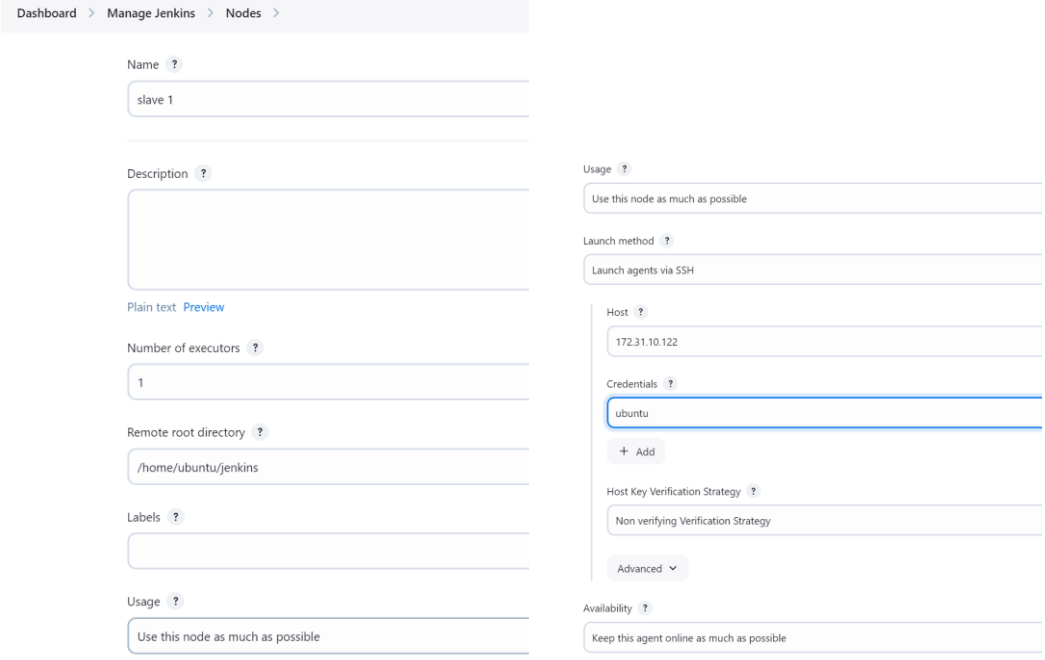
3) Manage > security > agents > random > save



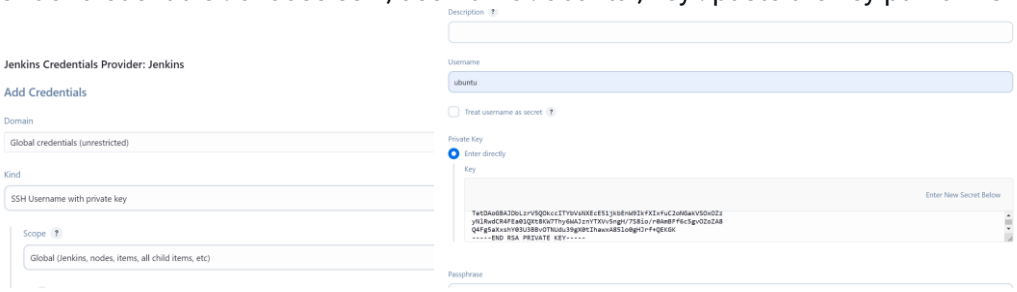
4) Create slave 1 : manage > nodes > give name > create



5) Give root dir : /home/ubuntu/Jenkins > launch method: ssh > host : private IP of slave

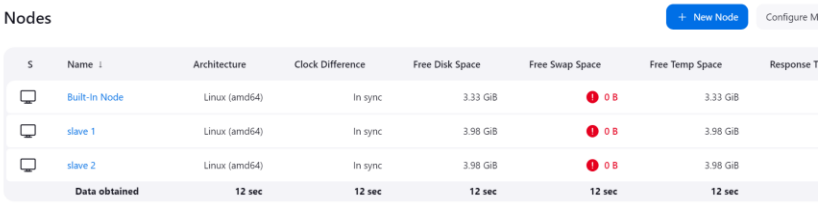


6) Under credentials : choose ssh , username : ubuntu , key : paste the key pair of EC2



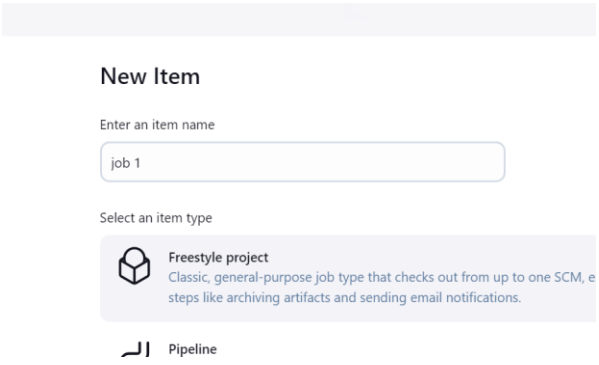
Slave 1 created.

7) **Create slave 2 .**

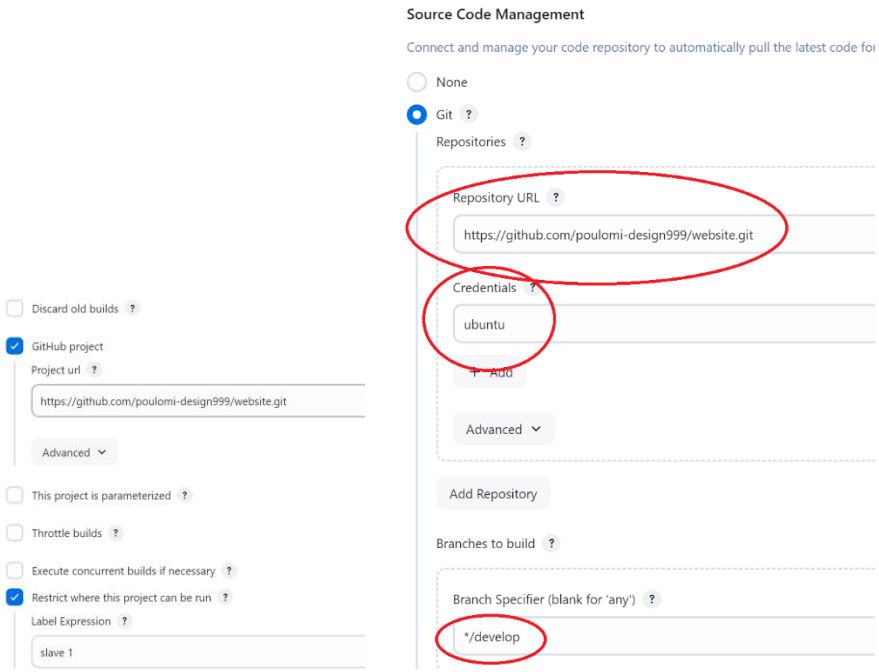


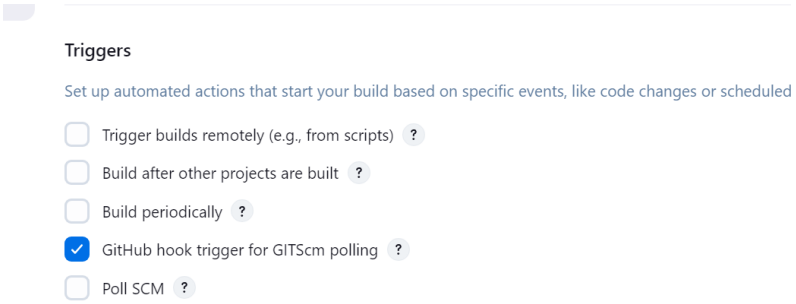
Create jobs

1)



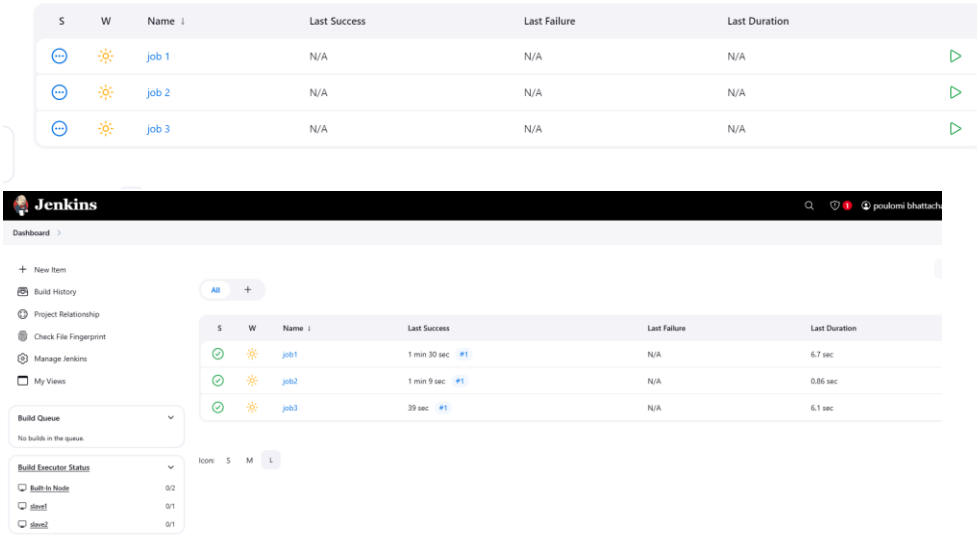
2)





3) Create job 2 : branch = */master [all same as job 1]

4) Create job 3 : branch = */master , label :slave 2 [all same as job 1]

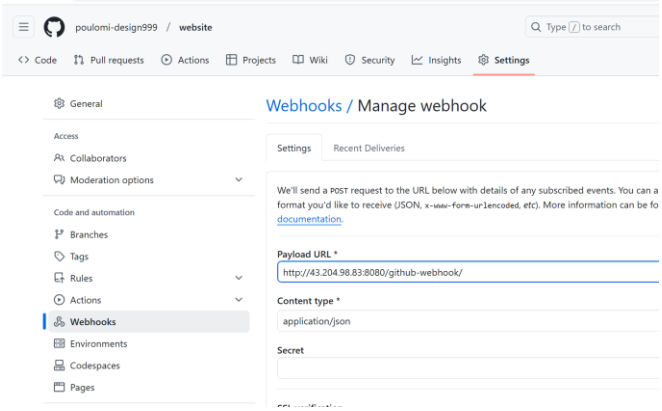


Github

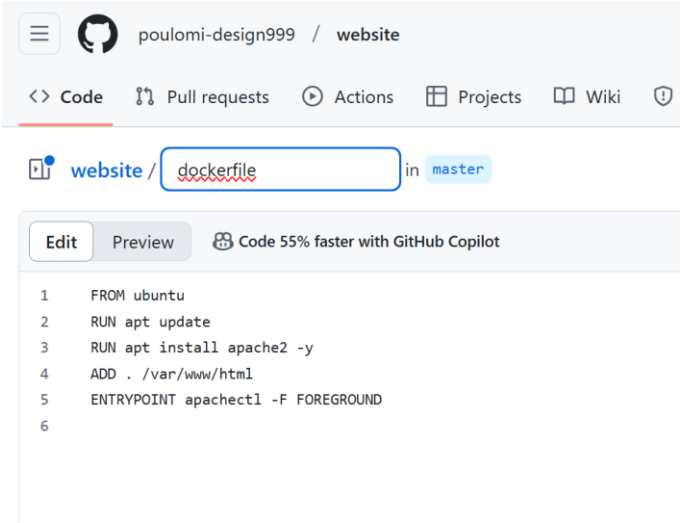
1) Create a webhook in the repo

Go to the repo > settings > webhook >

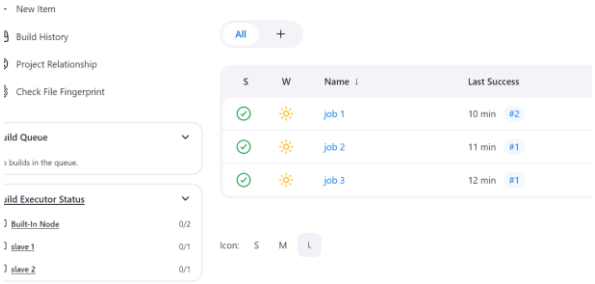
under URL : mention Jenkins URL/github-webhook/



2) **Create a docker file in your repo**

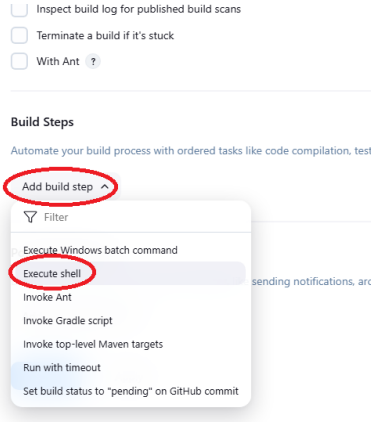


3) **We can see that after commit the jobs ran successfully automatically.**



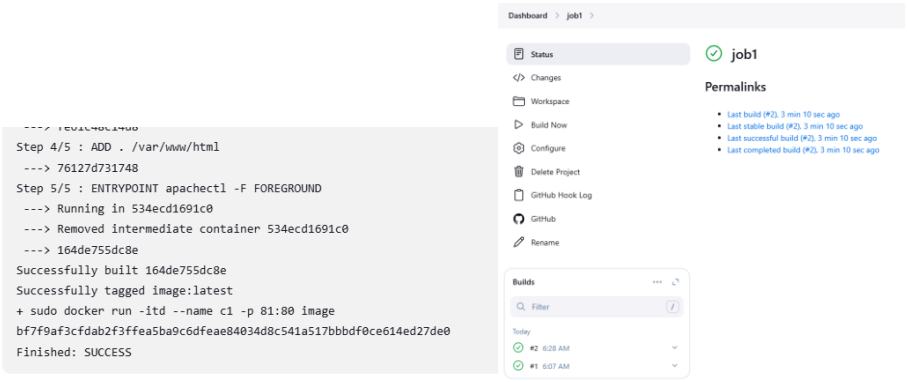
4) **Dockerize the docker file**

Job 1 > configure > add build step > execute shell



5) **Write the code to build image from dockerfile and containerize and run on a specific port.**

6) **Build job1 and we can see it was successful**



7) **For job 2 & 3 follow the same steps to build image from dockerfile and containerize and run on a specific port.**
8) **Paste the public IP of test machine and see the default apache2 page**