Jenkins CI/CD Pipeline Project With AWS

Problem Statement:

You have been Hired 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, 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.Code Build should automatically be triggered once commit is made to master branch or develop branch.

If commit is made to master branch, test and push to prod
If commit is made to develop branch, just test the product, do not push to prod

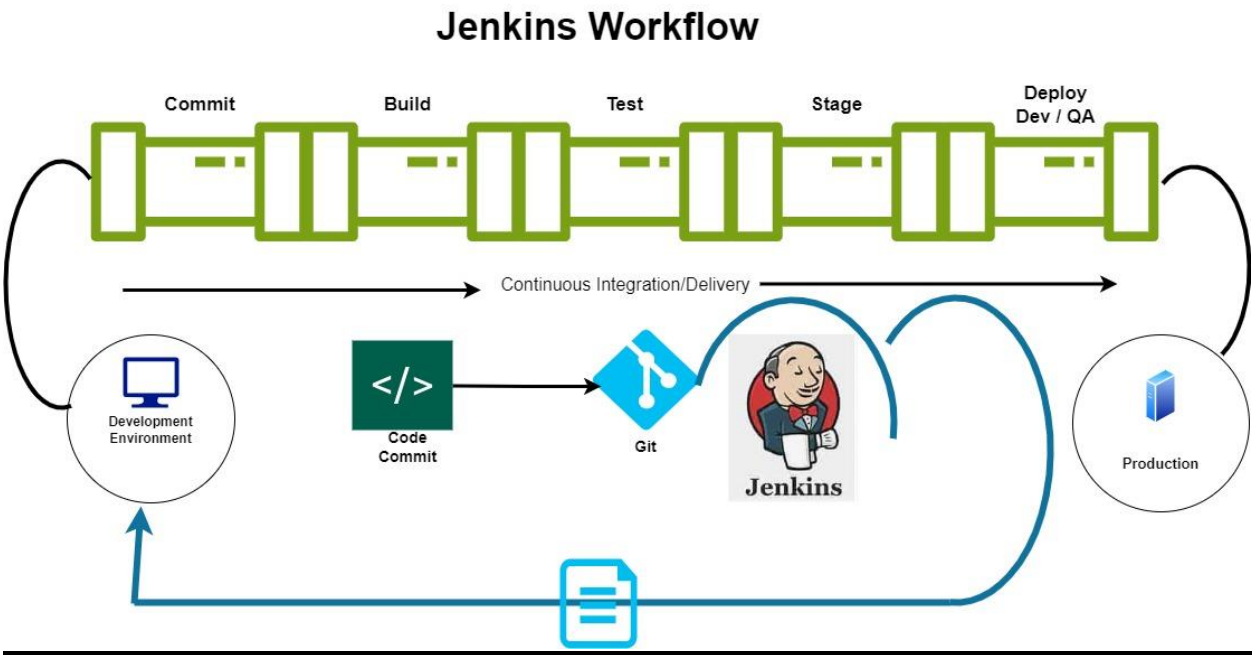
4.The Code should be containerized with the help of a Dockerfile. The Dockerfile should be built every time there is a push to Git-Hub. Use the following pre-built container for your application:

hshar/webapp

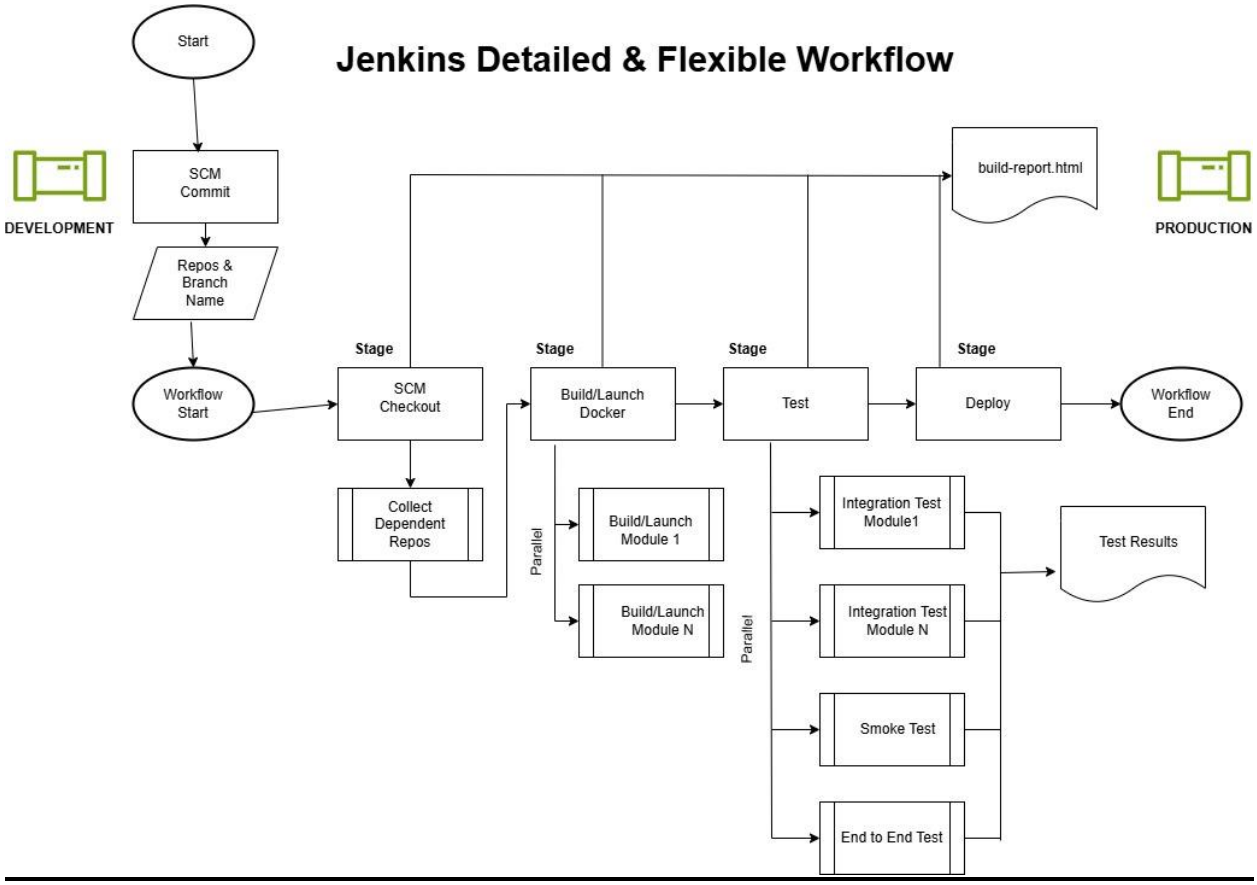
The code should reside in '/var/www/html'

5.The above tasks should be defined in a Jenkins Pipeline, with the following jobs:
Job1 : build
Job2: test
Job3 : prod

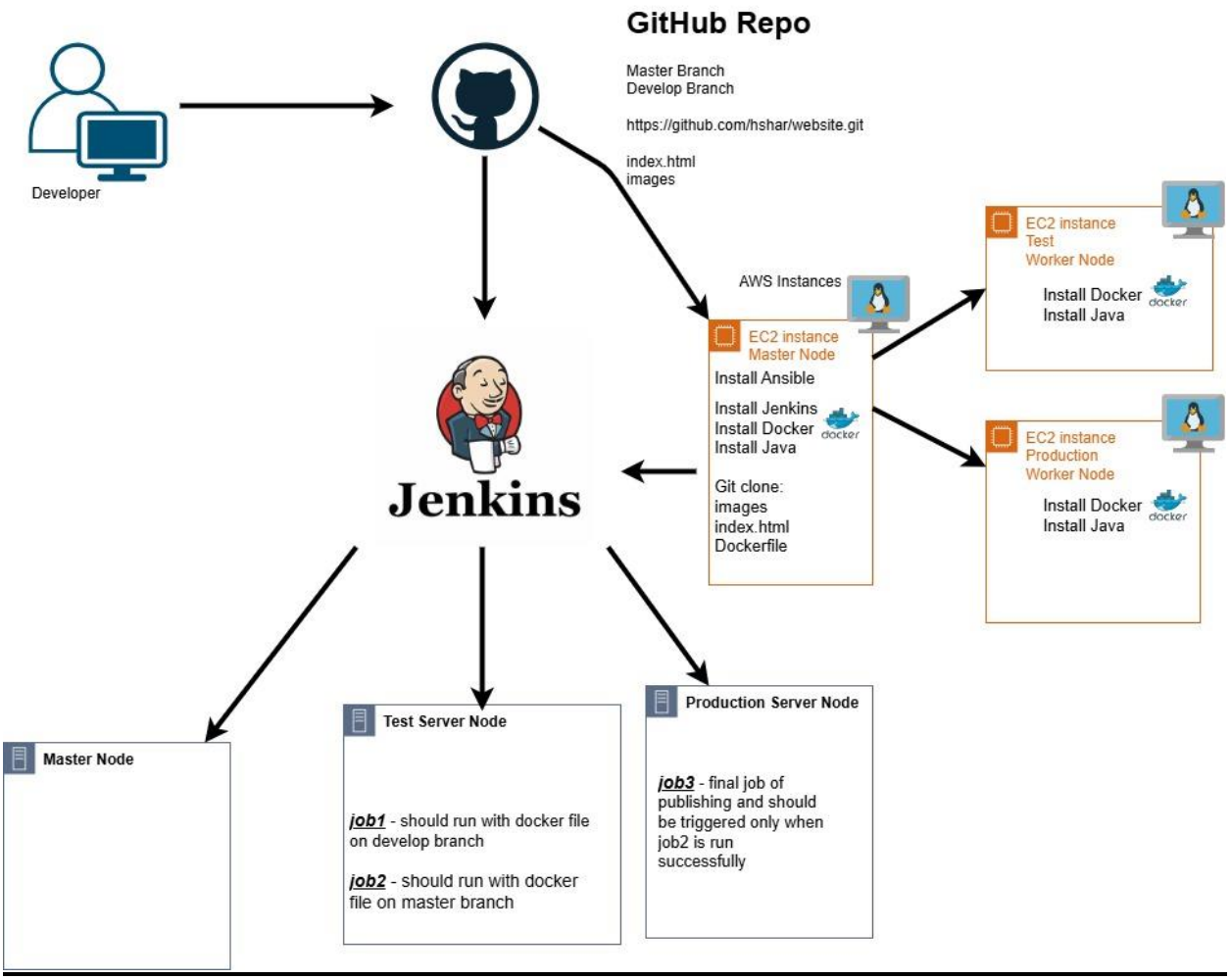
Jenkins High Level Workflow:



Jenkins Detailed Workflow:



This Project's Architecture:



Create AWS Instances

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...
master	i-04d45442fa33d6483	Running	t2.medium	2/2 checks passed	No alarms	us-east-2b	ec2-3-17-188-54.us-eas...	3.17.188.54
prod	i-0eccda53599f92078	Running	t2.micro	2/2 checks passed	No alarms	us-east-2c	ec2-3-17-27-190.us-eas...	3.17.27.190
test	i-054e27ce3d54a795c	Running	t2.micro	2/2 checks passed	No alarms	us-east-2c	ec2-3-16-109-221.us-e...	3.16.109.221

Step 1. Fork this github account, https://github.com/hshar/website.git into my GitHub account, [redacted]

Step 2. Launch 3 EC2 instances in AWS (master, test, prod) and update machines (sudo apt-get update)

. ssh connection into test and prod from the master. See commands below

cd
cd .ssh
ls
ssh-keygen
cat id_rsa.pub

On each worker node (test & prod):

cd .ssh
sudo nano authorized_key

Note: paste key save & exit

Step 3. Install Ansible on master

https://docs.ansible.com/ansible/latest/installation_guide/installation_distros.html#installing-ansible-on-ubuntu

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

```
When complete verify

ansible –version
which ansible
```

Ansible installed successfully by confirmation of pinging the test and prod worker nodes.

Step 5. Create script for ansible playbook on master node. I will call script install.yml

Go inside directory cd /etc/ansible
ls
sudo nano install.yml

This will install jenkins, java, and docker on the master node via a script, named jenkins.sh

It will also install java and docker on the test and prod nodes via a script, named docker.sh

```
GNU nano 4.8
---
 - hosts: localhost
   become: true
   name: install jenkins, java, and docker
   tasks:
   - name: master task
     script: jenkins.sh
 - hosts: test
   become: true
   name: install java and docker
   tasks:
   - name: test task
     script: docker.sh
 - hosts: prod
   become: true
   name: install java and docker
   tasks:
   - name: prod task
     script: docker.sh
```

ls

sudo nano hosts

Add

[test]
Private ip address

[prod]
Private ip address

```
  GNU nano 4.8                                                                  hosts
# This is the default ansible 'hosts' file.
#
# It should live in /etc/ansible/hosts
#
#   - Comments begin with the '#' character
#   - Blank lines are ignored
#   - Groups of hosts are delimited by [header] elements
#   - You can enter hostnames or ip addresses
#   - A hostname/ip can be a member of multiple groups

# Ex 1: Ungrouped hosts, specify before any group headers:

## green.example.com
## blue.example.com
## 192.168.100.1
## 192.168.100.10

# Ex 2: A collection of hosts belonging to the 'webservers' group:

## [webservers]
## alpha.example.org
## beta.example.org
## 192.168.1.100
## 192.168.1.110

# If you have multiple hosts following a pattern, you can specify
# them like this:

## www[001:006].example.com

# Ex 3: A collection of database servers in the 'dbservers' group:

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

# Here's another example of host ranges, this time there are no
# leading 0s:

## db-[99:101]-node.example.com

[test]
172.31.87.58

[prod]
172.31.92.147
```

Step 4. Write script to install jenkins, docker, and java on master instance.

ls

sudo nano jenkins.sh

go to https://www.jenkins.io/doc/book/installing/linux/

```
GNU nano 4.8                                                    jenkins.sh
curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo tee \
  /usr/share/keyrings/jenkins-keyring.asc > /dev/null
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 openjdk-11-jdk -y
sudo apt-get install docker.io -y
sudo apt-get install jenkins -y
```

Step 5. Create script to install docker and java on the test and prod machines

```
GNU nano 4.8                                                    docker.sh
sudo apt-get update
sudo apt-get install openjdk-11-jdk -y
sudo apt-get install docker.io -y
```

Step 6. Ping machines

ansible –m ping all

```
    ~/.ssh/known_hosts:4: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
172.31.38.138 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
yes
172.31.38.183 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
ubuntu@ip-172-31-19-149:/etc/ansible$
```

Step 7. Run ansible playbook

ansible–playbook install.yml

Ran successfully

```
PLAY RECAP *********************************************************************************************************
172.31.38.138              : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
172.31.38.183              : 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
```

Step 8. Set up Jenkins from master public ip 3.17.188.54:8080

Unlock passwork:

`/var/lib/jenkins/secrets/initialAdminPassword`

[redacted]

Install plugins

Complete login credentials

Step 9. Clone GitHub master branch to your GitHub account

```
ubuntu@ip-172-31-19-149:/etc/ansible$ cd
ubuntu@ip-172-31-19-149:~$ git clone https://github.com/[redacted].git
Cloning into 'website'...
remote: Enumerating objects: 8, done.
remote: Total 8 (delta 0), reused 0 (delta 0), pack-reused 8
Receiving objects: 100% (8/8), 82.69 KiB | 2.51 MiB/s, done.
Resolving deltas: 100% (1/1), done.
ubuntu@ip-172-31-19-149:~$ ls
website
ubuntu@ip-172-31-19-149:~$
```

Step 10. Create Dockerfile to create container

```
ubuntu@ip-172-31-19-149:~/website$ sudo nano Dockerfile
```

Dockerfile:

```
FROM ubuntu
RUN apt-get update
RUN apt-get install apache2 -y
ADD . /var/www/html
ENTRYPOINT apachectl -D FOREGROUND
```

Step 11. Add Dockerfile to GitHub and create develop branch & push files to develop branch and Dockerfile to master as shown in the below commands

git status

git add .

git commit –m “adding Dockerfile”

git branch

git branch develop

git branch

git push origin develop

git push origin master

Step 12. Jenkins

Manage Jenkins > Configure Global Security > Agents > Random

Apply then Save

Step 13. Configure nodes

Manage Jenkins > Manage Nodes and Clouds

Master has been synched

Select + New Node

Name ?

test

Description ?

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

In below host is private ip address of test instance

Jenkins

Search

Dashboard > Nodes >

Back to Dashboard

Manage Jenkins

New Node

Configure Clouds

Node Monitoring

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

Name ?

test

Description ?

Number of executors ?

1

Remote root directory ?

/home/ubuntu/jenkins/

Labels ?

Usage ?

Use this node as much as possible

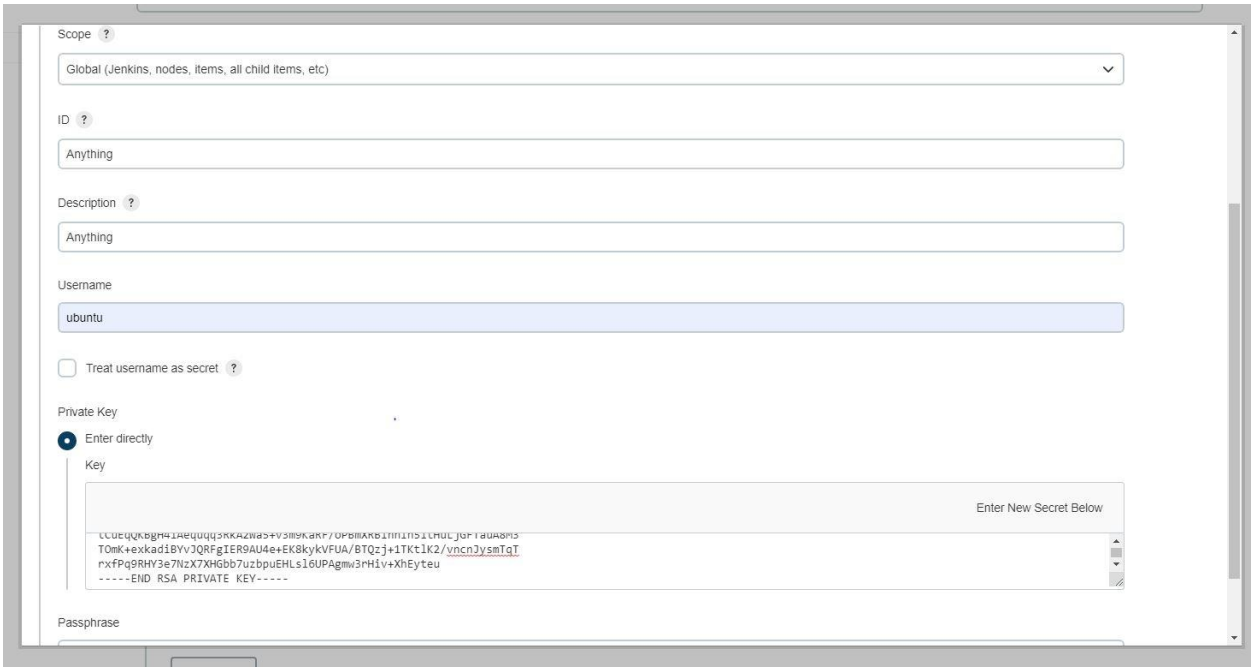
Launch method ?

Launch agents via SSH

Host ?

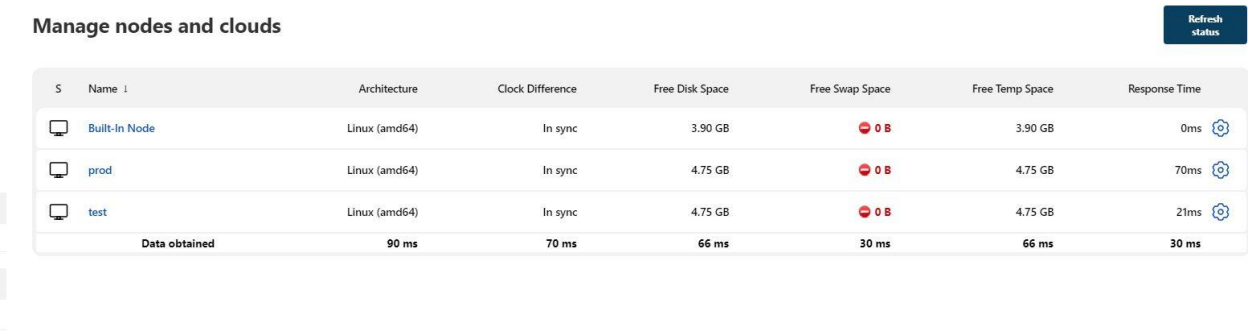
172.31.38.138

The below is a screenshot to configure credentials in the above screenshot. Select + Add button



Test node has been configured successfully. Do the same for prod. Prod has been configured successfully.

Built-in Node is master node.



Manage nodes and clouds

Refresh status

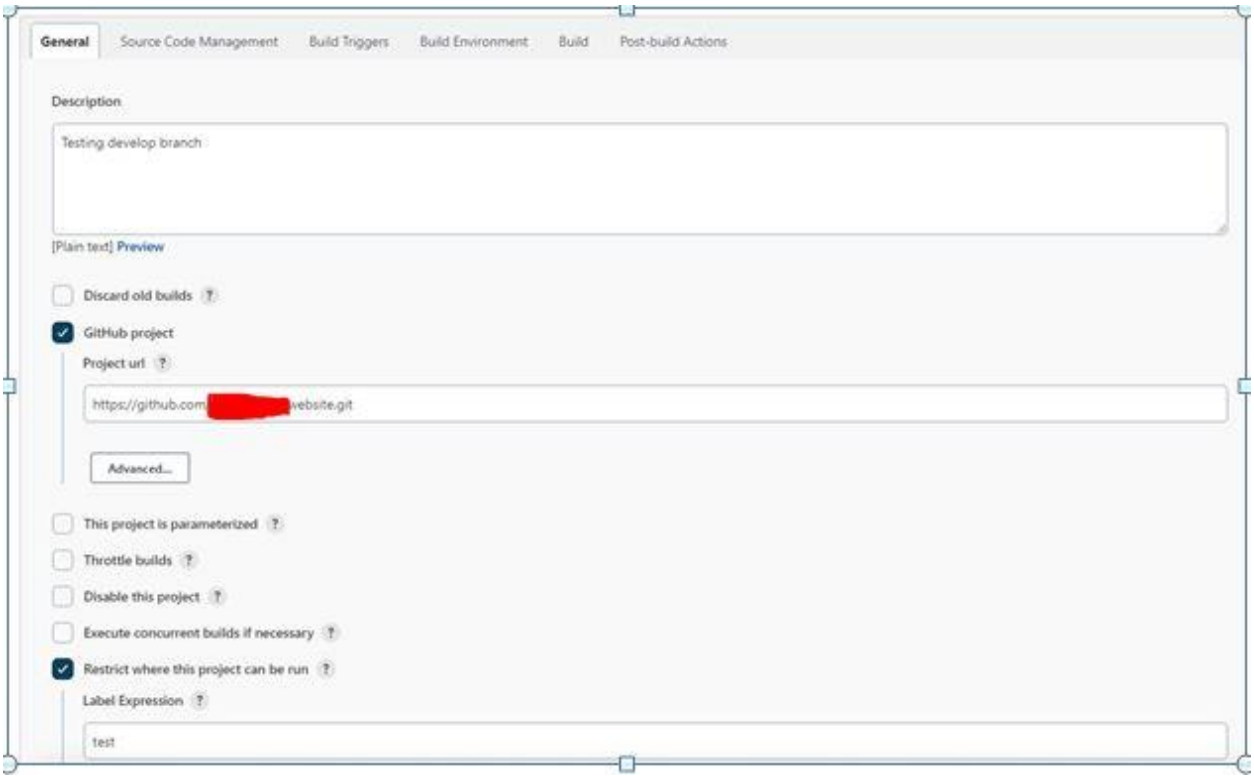
S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	3.90 GB	0 B	3.90 GB	0ms
	prod	Linux (amd64)	In sync	4.75 GB	0 B	4.75 GB	70ms
	test	Linux (amd64)	In sync	4.75 GB	0 B	4.75 GB	21ms
	Data obtained	90 ms	70 ms	66 ms	30 ms	66 ms	30 ms

Create Jobs

Step 14. Create job1 in Jenkins

Dashboard > + New Item

Job1 > Freestyle



General | Source Code Management | Build Triggers | Build Environment | Build | Post-build Actions

Description

Testing develop branch

[Plain text] Preview

Discard old builds ?

GitHub project

Project url ?

https://github.com[redacted]website.git

Advanced...

This project is parameterized ?

Throttle builds ?

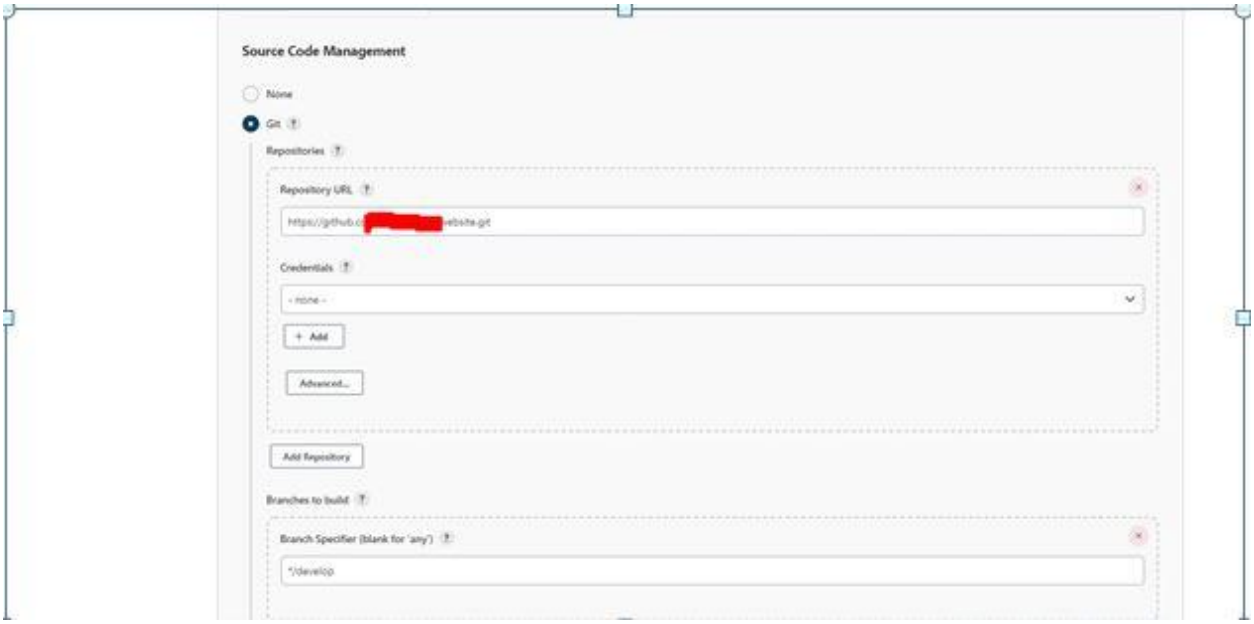
Disable this project ?

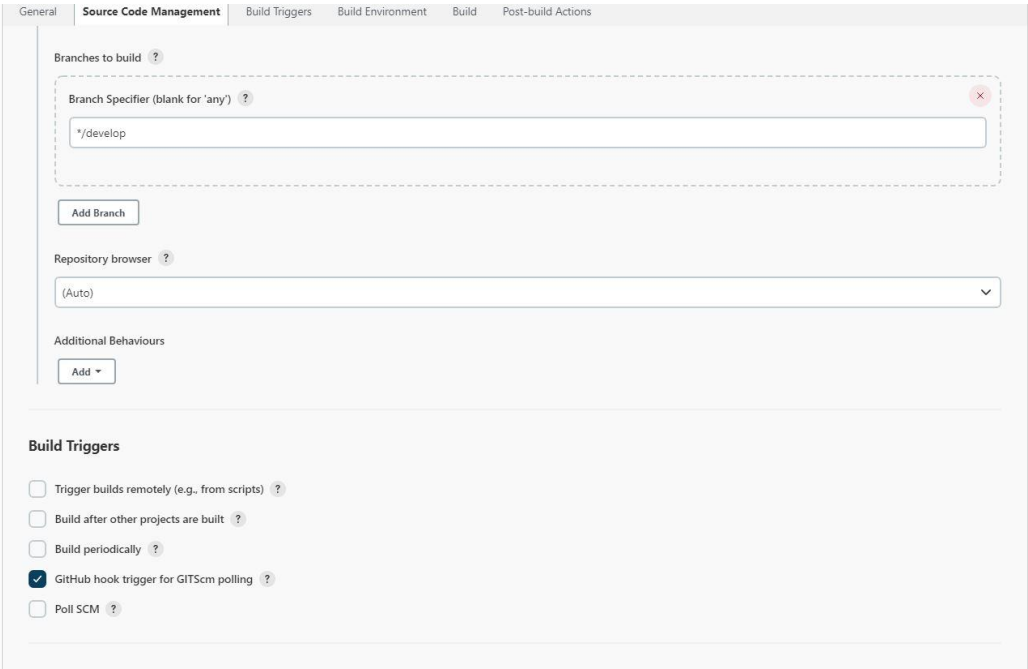
Execute concurrent builds if necessary ?

Restrict where this project can be run ?

Label Expression ?

test



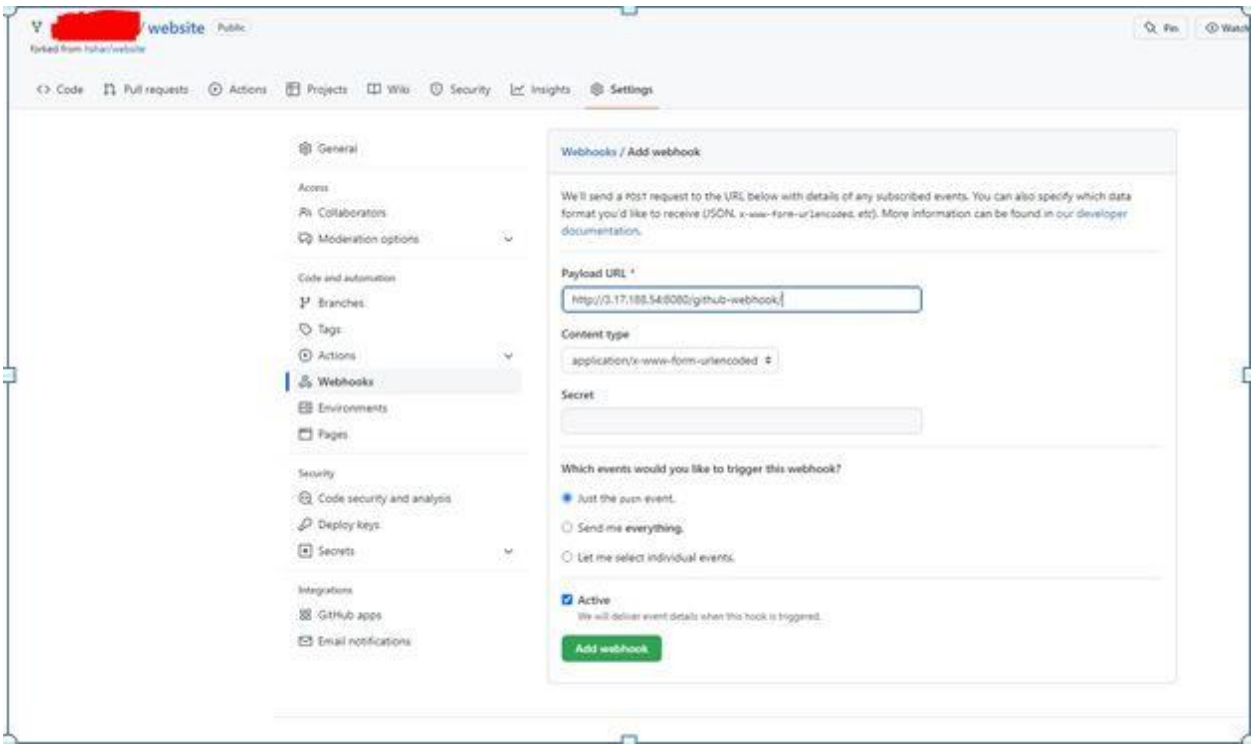


Generate webhook in GitHub

Settings > Webhook > Add Webhook

Under Payload URL: <master instance public ip address>:8080/git-webhook/

Select <Add Webhook>



- Build Now
- Job1 built successfully

Verify files are in test node in linux

```
ubuntu@ip-172-31-38-138:~$ cd jenkins
ubuntu@ip-172-31-38-138:~/jenkins$ ls
remoting  remoting.jar  workspace
ubuntu@ip-172-31-38-138:~/jenkins$ cd workspace
ubuntu@ip-172-31-38-138:~/jenkins/workspace$ ls
job1
ubuntu@ip-172-31-38-138:~/jenkins/workspace$ cd job1
ubuntu@ip-172-31-38-138:~/jenkins/workspace/job1$ ls
Dockerfile  images  index.html
ubuntu@ip-172-31-38-138:~/jenkins/workspace/job1$
```

pwd for path in build for next steps

Go back to Jenkins

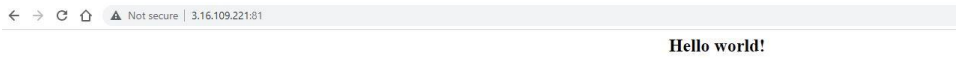
- Configure
- Build
- Execute shell

Build script for docker container:

Include line 1: Forcefully remove the Docker container identified as “c1”
Include line 2: Build container using path of job1 and provide a tag (-t developapp)
Include line 3: Run the container (itd - interactive, attach a terminal to the container’s shell detach) and provide the port to publish container (81:80)

- Build Now
- Project job1 ran successfully

Verify by testing:



Hello world!

GitHub

Step 15. Configure job2 in Jenkins – Testing for master branch

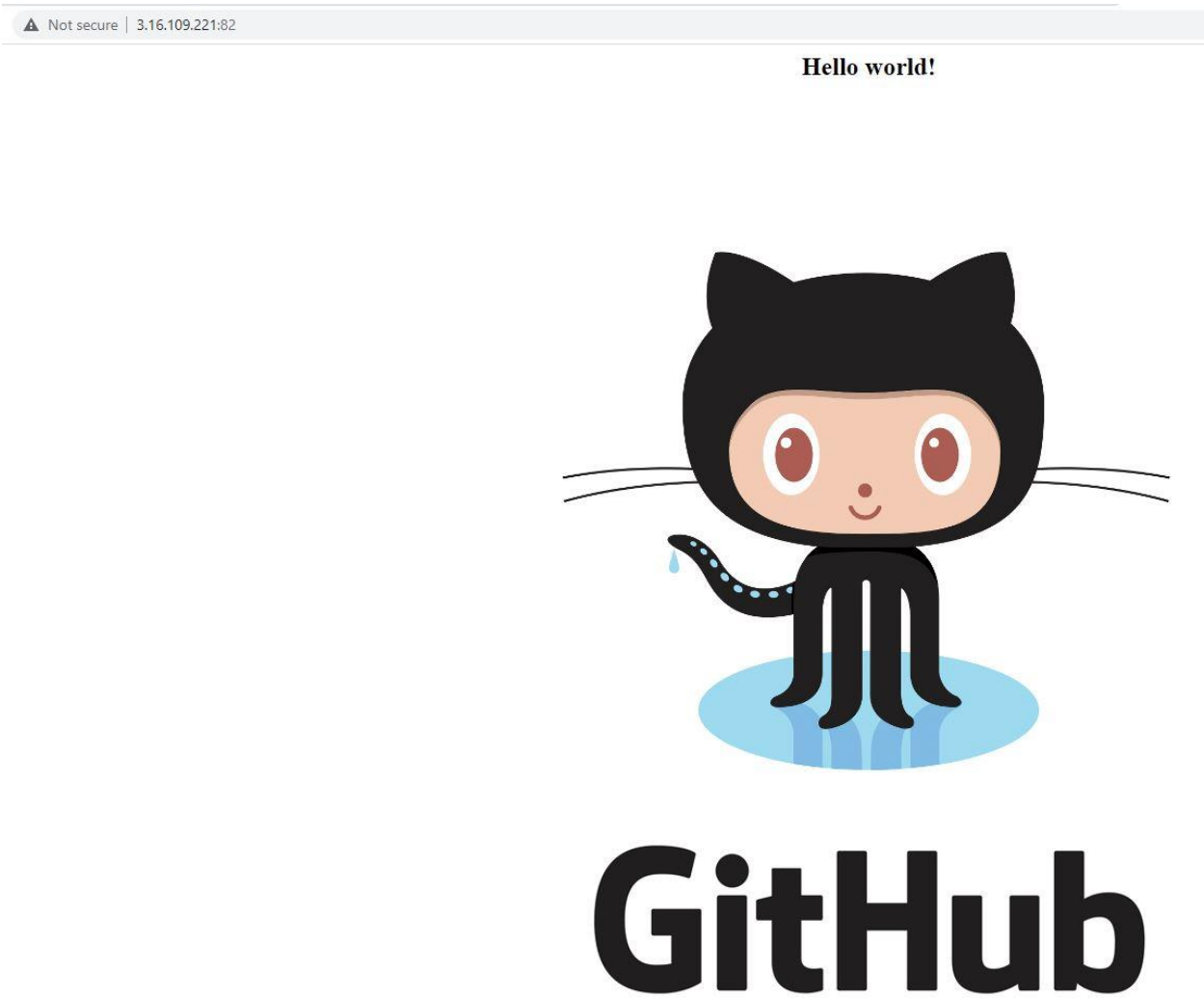
Use steps in screenshots above and change branch from develop to master under Source Code Management section.

Go back to configure

- Build
- Execute Shell

- Build Now
- Project job2 ran successfully

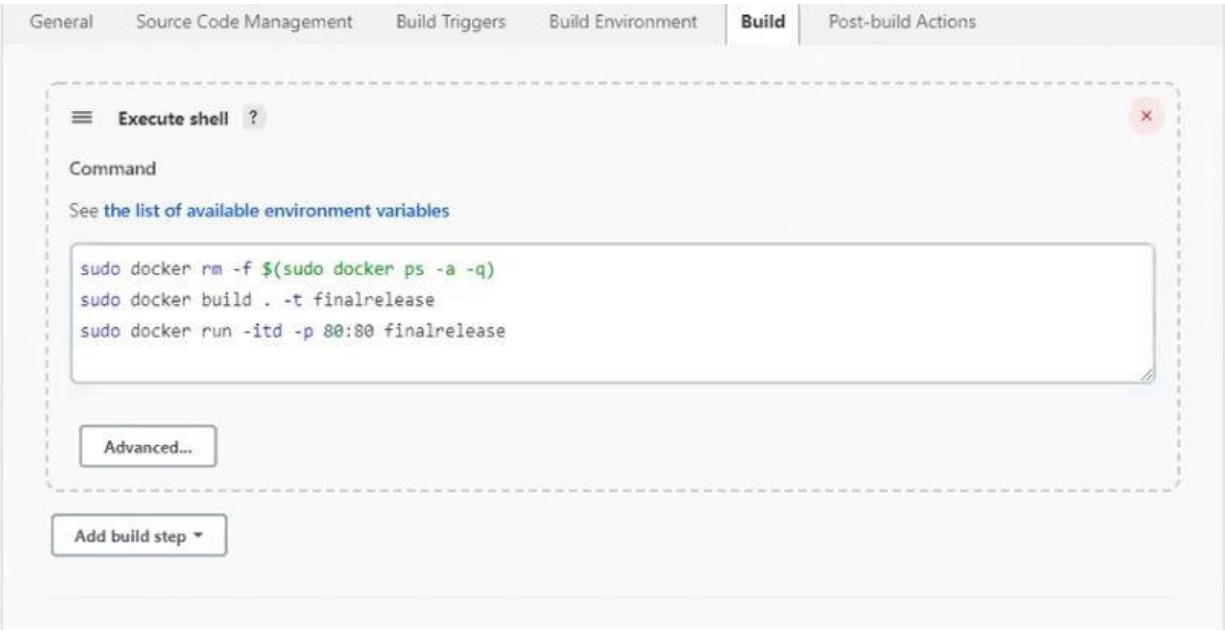
Verify by testing:



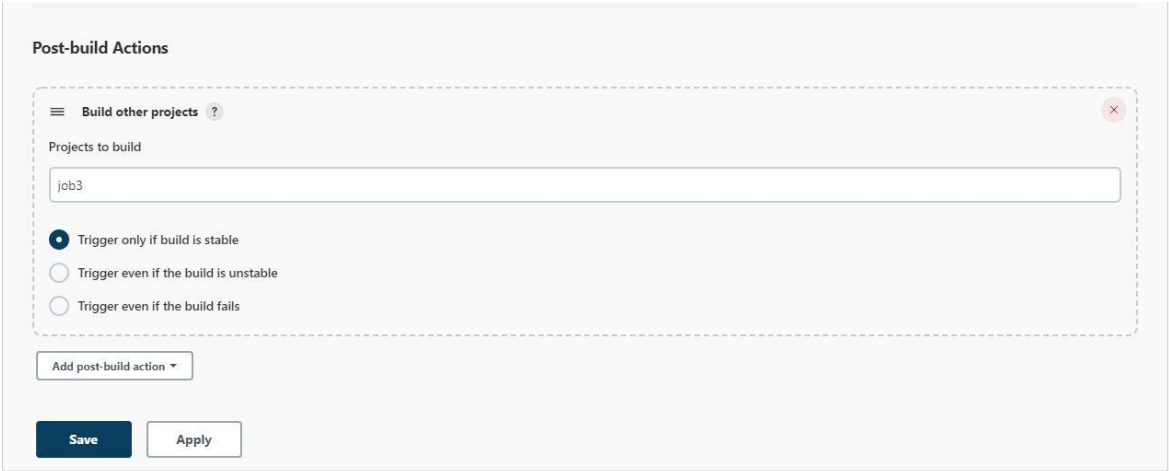
Step 16. Configure job3 in Jenkins – release product after successful master test at job2. Repeat steps in screenshots above

Differences in job3 configuration under General Tab:

Note: Do not select GitHub hook trigger for GITScm polling from Build Triggers as we want to publish only if job2 is successful on the test node.



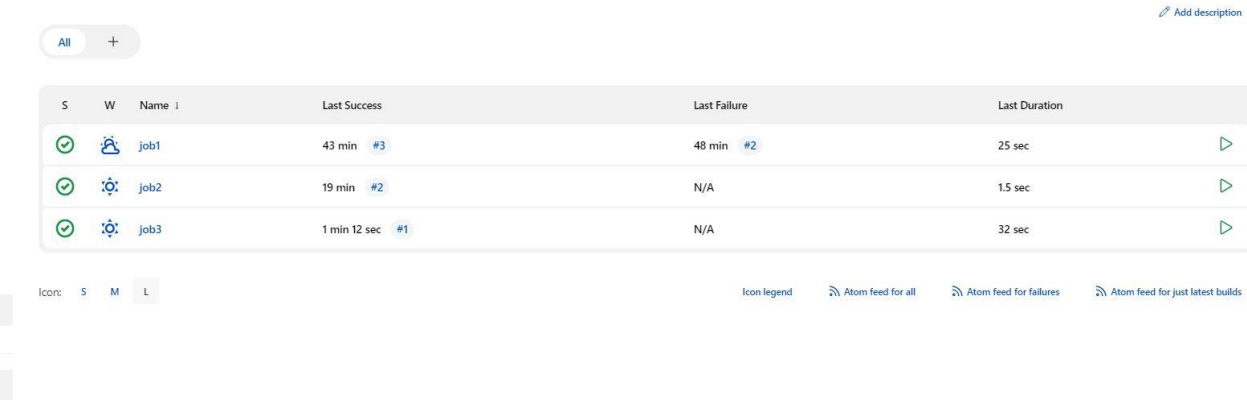
Now go into job2 and configure the Post Build Actions. So, only if job2 is stable do want job3 to publish the final release to prod.



Go back to job3 and select Build Now

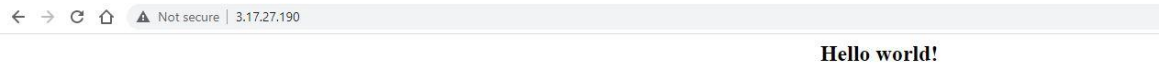
- Job3 built successfully

See below 3 jobs ran and completed successfully



Verify:

Public IP address of prod machine with port 80 or 82



Hello world!

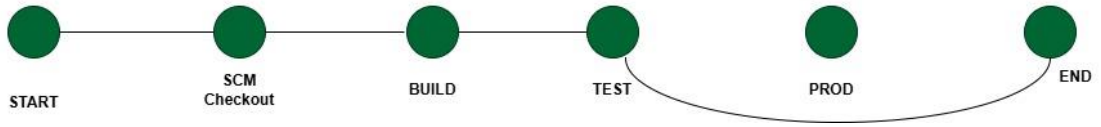
GitHub

What happens when we make code changes in GitHub:

After making a change to the code on the develop branch, the pipeline overview would look like the below overview. The code would be tested but not published to prod.

PIPELINE OVERVIEW

FOLLOWING A CODE CHANGE AND COMMIT ON THE DEVELOP BRANCH IN GITHUB



After making a change to the code on the master branch and conducting a commit, or making a change on the develop branch and merging to the master branch the pipeline overview would look like the below overview. The code would be published to prod as a commit has been made on the master branch and thus publishes the code to prod.

PIPELINE OVERVIEW

FOLLOWING A CODE CHANGE AND COMMIT ON THE MASTER BRANCH OR ON THE DEVELOP BRANCH AND MERGED TO THE MASTER BRANCH IN GITHUB