

EPAM(Cloud DevOps) PROJECT

Name: Katuri Bhuvanesh

Id: 2000031664

Sec: 13

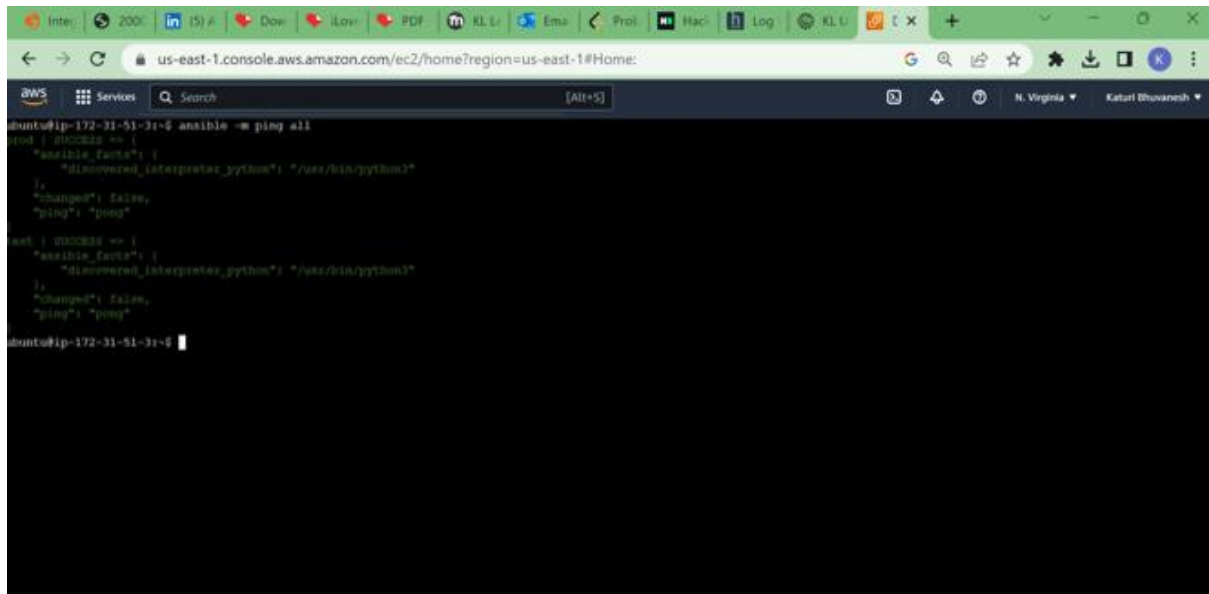
Automated CI/CD pipeline using Jenkins, Docker, and Ansible

Project :- we have a static website and we have to deploy this website using DevOps tools. Following devops lifecycle has to follow:

1. Git Workflow has to be implemented
2. Code Build should automatically be triggered once commit is made to master branch or develop branch.
3. 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

Project -01

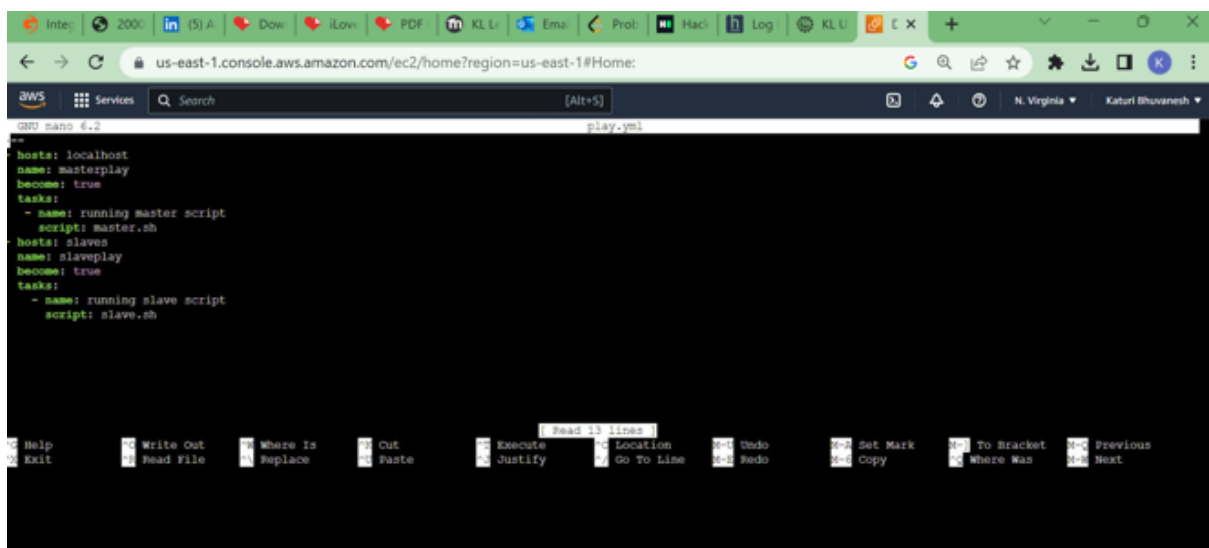
1. Configuring servers using Ansible:
 - . Launch 3 EC2 instances from AWS.
 - . Establish Ansible structure on these servers such that one server becomes "Master" and other two becomes "slaves".
 - . Slaves connected to the master using ssh keys generated through the command "ssh-keygen".



The screenshot shows the AWS Management Console interface with a terminal window open. The terminal displays the execution of an Ansible command: `ansible -m ping all`. The output shows that the command was successful for the host `ec2-172-31-51-31-1`, with the `ansible_facts` dictionary containing `discovered_interpreter_python` set to `/usr/bin/python3`. The terminal prompt is `ec2-172-31-51-31-1`.

2. Creating Ansible playbook:

. Ansible playbook is created which will deploy all the necessary software's on servers.



The screenshot shows the AWS Management Console interface with a terminal window open. The terminal displays the content of an Ansible playbook file named `play.yml`. The playbook defines two plays: one for `localhost` (named `masterplay`) and one for `slaves` (named `slaveplay`). Both plays have the `become: true` option and a task to run a script. The `localhost` play runs `master.sh`, and the `slaves` play runs `slave.sh`. The terminal prompt is `ec2-172-31-51-31-1`.

. we have included two script files `master.sh` and `slave.sh` in playbook thus, we have to create these two files.

3. Creating master.sh script file:

. Create `master.sh` script file which will deploy all the necessary software's on master.

. This file will deploy docker and Jenkins on master server.

The screenshot shows the AWS Management Console interface for an EC2 instance in the us-east-1 region. A terminal window is open, displaying the following commands and their outputs:

```
GNU nano 2.9.3 master.sh
#!/bin/bash
sudo apt install docker.io -y
sudo apt install openjdk-11-jdk -y
wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -
sudo sh -e 'echo deb https://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
sudo apt-get update
sudo apt-get install jenkins -y
```

The terminal window includes a menu bar at the bottom with options like Help, Exit, Write Out, Read File, Where Is, Replace, Cut, Paste, Execute, Justify, Location, Go To Line, Undo, Redo, Set Mark, Copy, To Bracket, Where Was, Previous, and Next.

4. Creating slave.sh script file:

- .Create script file for slaves which will deploy all the necessary software's on slaves.
- .By running this script file it will install Docker and Java on slave servers.

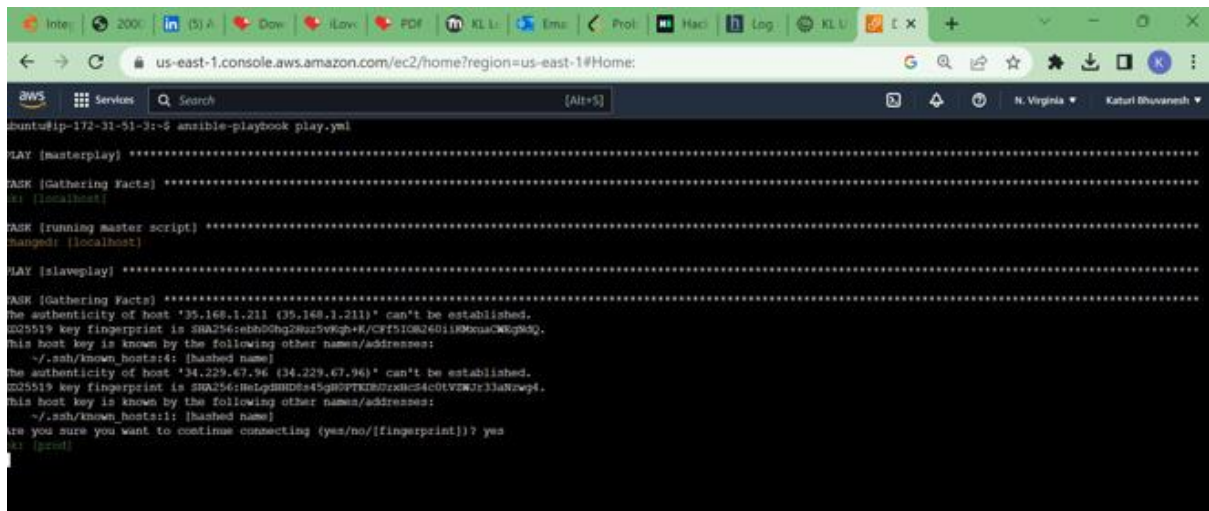
The screenshot shows the AWS Management Console interface for an EC2 instance in the us-east-1 region. A terminal window is open, displaying the following commands and their outputs:

```
GNU nano 2.9.3 slave.sh
#!/bin/bash
sudo apt install docker.io -y
sudo apt install openjdk-11-jdk -y
```

The terminal window includes a menu bar at the bottom with options like Help, Exit, Write Out, Read File, Where Is, Replace, Cut, Paste, Execute, Justify, Location, Go To Line, Undo, Redo, Set Mark, Copy, To Bracket, Where Was, Previous, and Next.

5. Running Ansible Playbook:

- . Thus by running this playbook file we will configure all the servers with all the necessary software's.



```
aws-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Home:
AWS Services Search [Alt+S]
buntu@ip-172-31-51-31:~$ ansible-playbook play.yml

PLAY (masterplay) *****
TASK [Gathering Facts] *****
ok: [localhost]

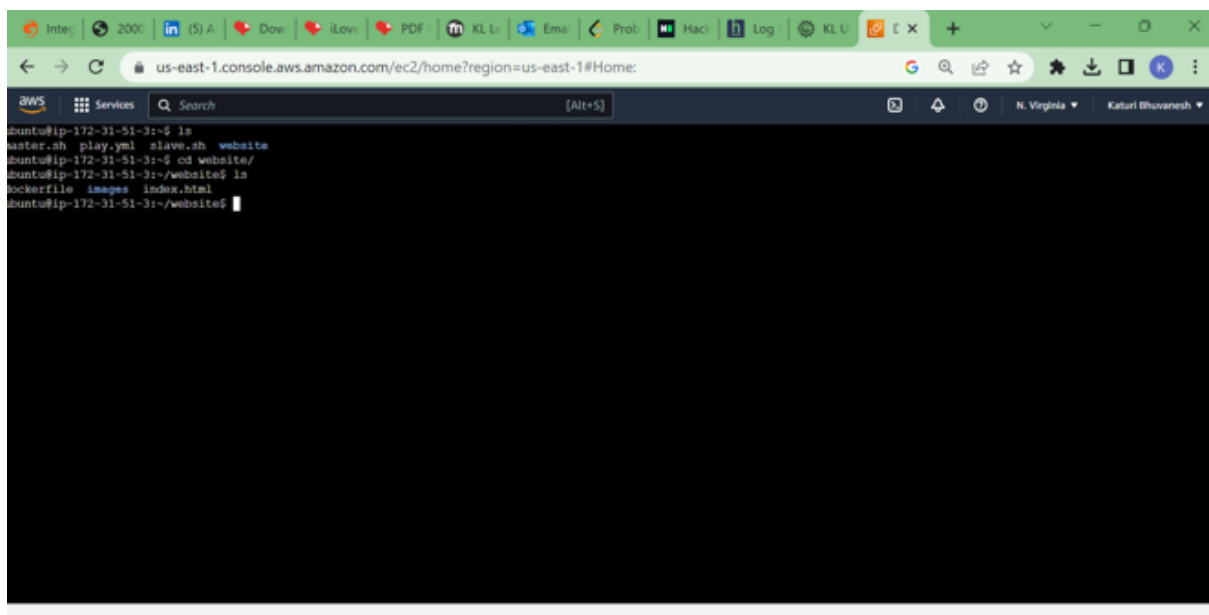
TASK [running master script] *****
changed: [localhost]

PLAY [slaveplay] *****
TASK [Gathering Facts] *****
The authenticity of host '35.168.1.211 (35.168.1.211)' can't be established.
ECDSA key fingerprint is SHA256:stbDhg2waz3vWgh+K/CFt108J4011806uuCKWg3Mg2.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:4: [hashed name]
The authenticity of host '34.229.67.96 (34.229.67.96)' can't be established.
ECDSA key fingerprint is SHA256:HeLgBND6e45gH0PTK0B/zxHcS4C0tV2WJr33aKzwp4.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:11: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
yes: [prod]
```

6. Cloning project to master server:

. As we have static website clone this repository to master server using command "git clone <repo link>".

. Thus necessary index file for website and images needed for website is cloned to master server.

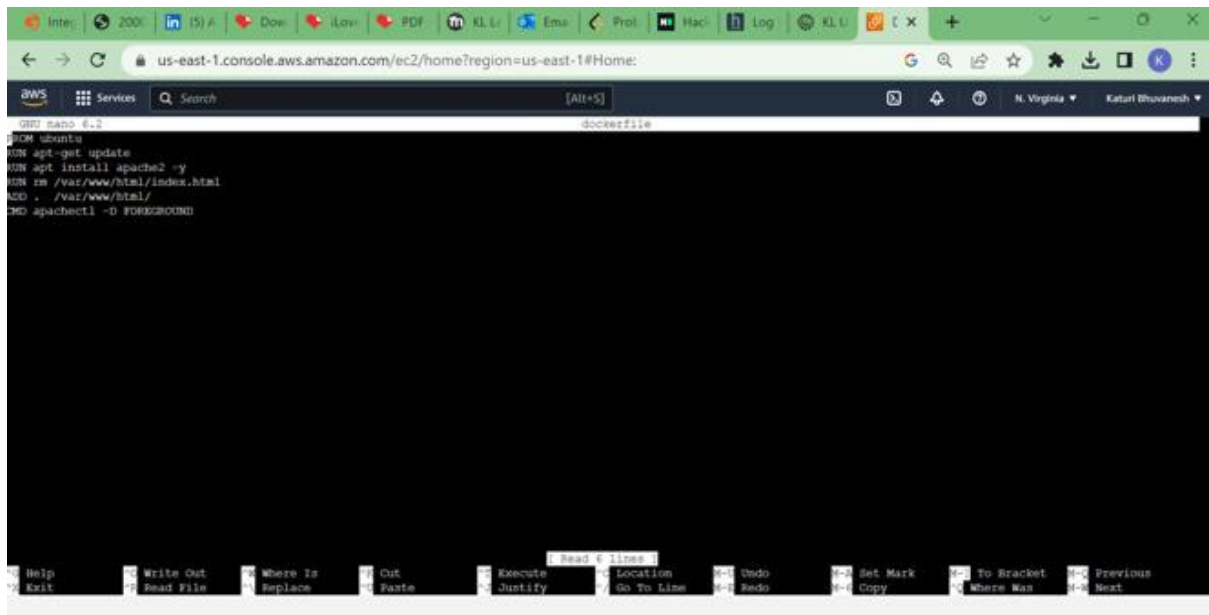


```
aws-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Home:
AWS Services Search [Alt+S]
buntu@ip-172-31-51-31:~$ ls
master.sh  play.yml  slave.sh  website
buntu@ip-172-31-51-31:~$ cd website/
buntu@ip-172-31-51-31:~/website$ ls
Dockerfile  images  index.html
buntu@ip-172-31-51-31:~/website$
```

7. Creating Dockerfile:

.A dockerfile has been created, creating a container with an ubuntu image.

. It will install the apache2 webserver on this container and copies the required files to /var/www/html/ location.



The screenshot shows the AWS Management Console for the us-east-1 region. A terminal window is open, displaying the following commands and their output:

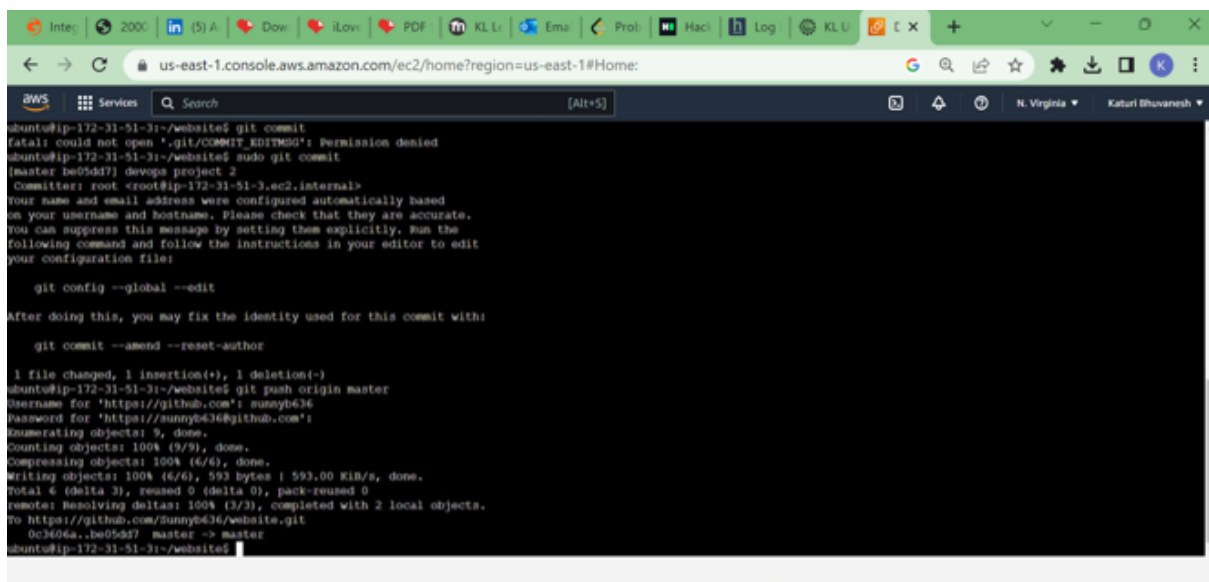
```
FROM ubuntu
RUN apt-get update
RUN apt install apache2 -y
RUN rm /var/www/html/index.html
ADD . /var/www/html/
CMD apache2ctl -D FOREGROUND
```

The terminal window has a menu bar at the bottom with options: Help, Exit, Write Out, Read File, Where Is, Replace, Cut, Paste, Execute, Justify, Location, Go To Line, Undo, Redo, Set Mark, Copy, To Bracket, Where Was, Previous, Next.

8. Pushing Dockerfile to github:

. A new branch with name "develop" is created for testing purpose. command used is "git branch <branch name>".

. Docker file and all project files are pushed to github using command "git add .", "git commit .", "git push origin <branch name>".



The screenshot shows the AWS Management Console for the us-east-1 region. A terminal window is open, displaying the following commands and their output:

```
shuntu@ip-172-31-51-31:~/website$ git commit
fatal: could not open '.git/COMMIT_EDITMSG': Permission denied
shuntu@ip-172-31-51-31:~/website$ sudo git commit
[master be05dd7] devops project 2
Committer: root <root@ip-172-31-51-31.ec2.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

1 file changed, 1 insertion(+), 1 deletion(-)
shuntu@ip-172-31-51-31:~/website$ git push origin master
Username for 'https://github.com': sunnyb636
Password for 'https://sunnyb636@github.com':
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 593 bytes | 593.00 KiB/s, done.
Total 6 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To https://github.com/sunnyb636/website.git
   0c3406a..be05dd7  master -> master
shuntu@ip-172-31-51-31:~/website$
```

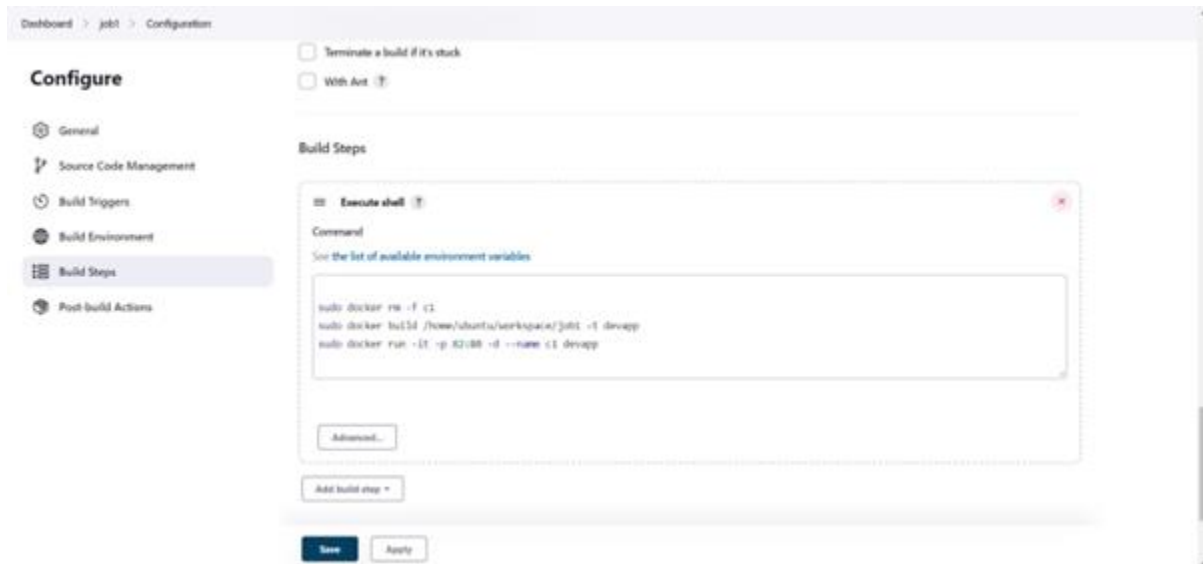
9. Creating job on jenkins cluster:

. As we have already set-up jenkins cluster using ansible we can create job which will deploy code on test server.

. Both nodes are attached to Jenkins with the names "test" and "prod".

.New job is created with the name "job1", so it will deploy the application on the test server.

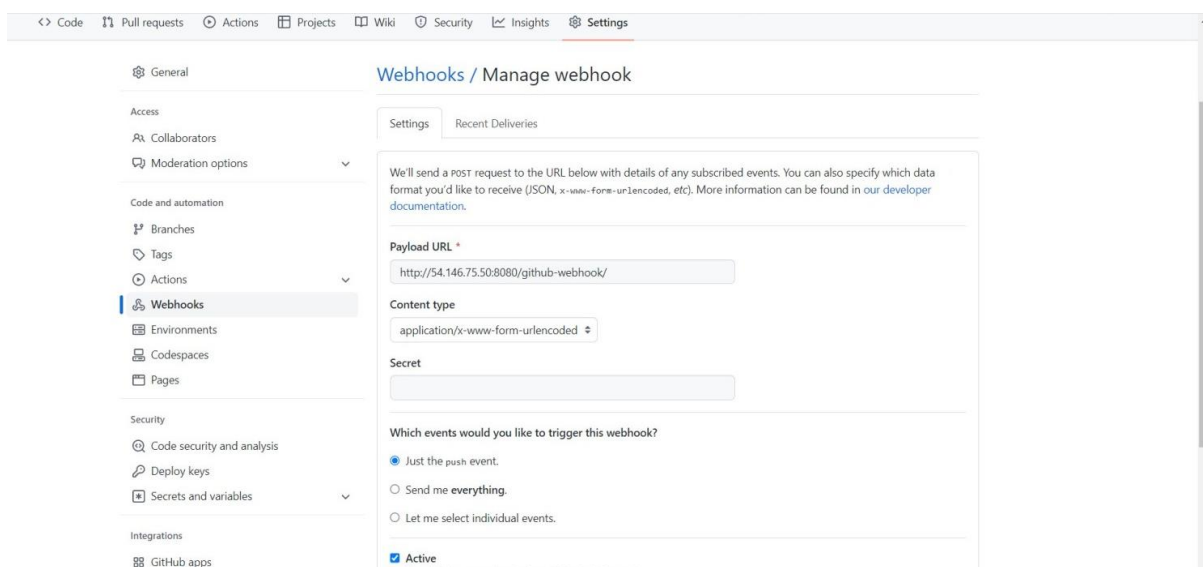
. Also provided build steps with which it will create a docker image and run the container on port 82.



10. Adding Git-Hub webhook:

. While creating the job I also enabled the webhook option and created a webhook on a git repository.

. This will trigger the job for every push to git repository.

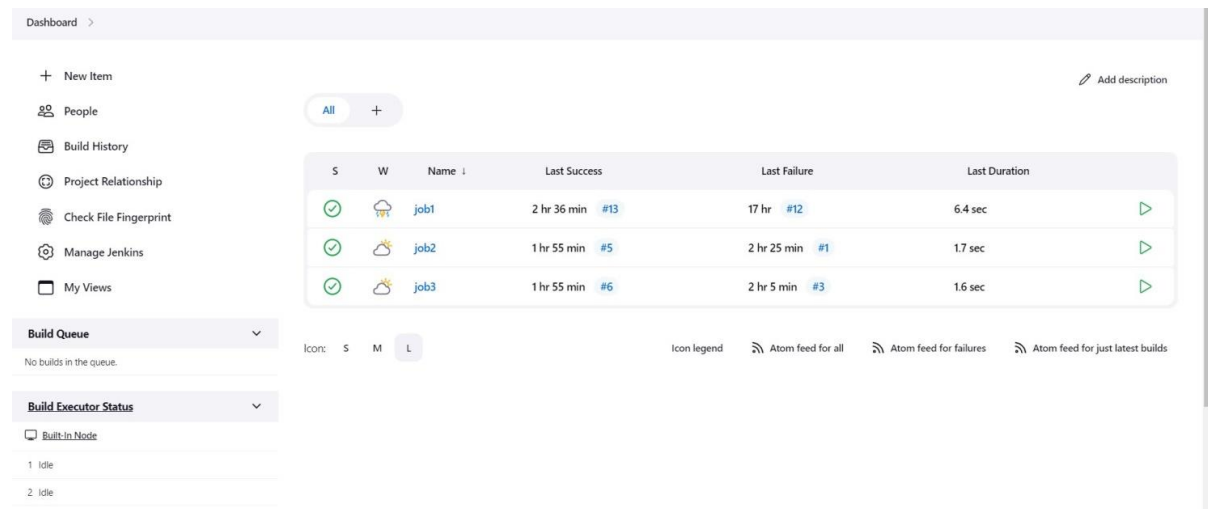


11. Creating 2 more jobs on Jenkins:

.Similar to job 1 two new jobs have been created with the names "job2" and "job3" .

.When a push is made on the master branch of the test server it will trigger Job 2 which will test the website and further trigger job 3.

.When job 2 runs successfully it will further trigger job 3 which will deploy the application on production server.



The screenshot shows the Jenkins Dashboard. On the left is a sidebar with navigation links: New Item, People, Build History, Project Relationship, Check File Fingerprint, Manage Jenkins, and My Views. The main area displays a table of build jobs. The table has columns for status (S), weather icon (W), name, last success, last failure, and last duration. Three jobs are listed: job1, job2, and job3. job1 has a success status, a sun icon, and a duration of 6.4 sec. job2 has a success status, a sun icon, and a duration of 1.7 sec. job3 has a success status, a sun icon, and a duration of 1.6 sec. Below the table, there are sections for 'Build Queue' (showing no builds) and 'Build Executor Status' (showing two idle executors).

S	W	Name	Last Success	Last Failure	Last Duration
✓	☀️	job1	2 hr 36 min #13	17 hr #12	6.4 sec
✓	☀️	job2	1 hr 55 min #5	2 hr 25 min #1	1.7 sec
✓	☀️	job3	1 hr 55 min #6	2 hr 5 min #3	1.6 sec

12. Website deployed:

.Finally, job 3 will deploy a website on port 84.

.And website looks like this:



"Hello world! welcome to advance human race of all universe" Greetings from Sunny



GitHub