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 branchor 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

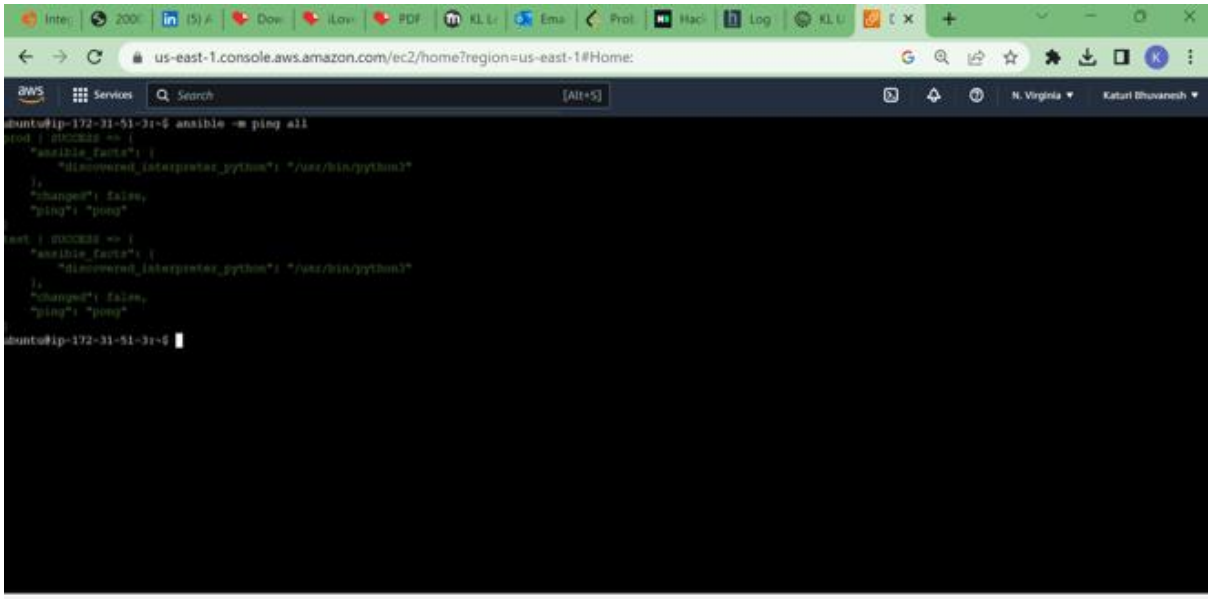
Project -01

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”.



2. Creating Ansible playbook:

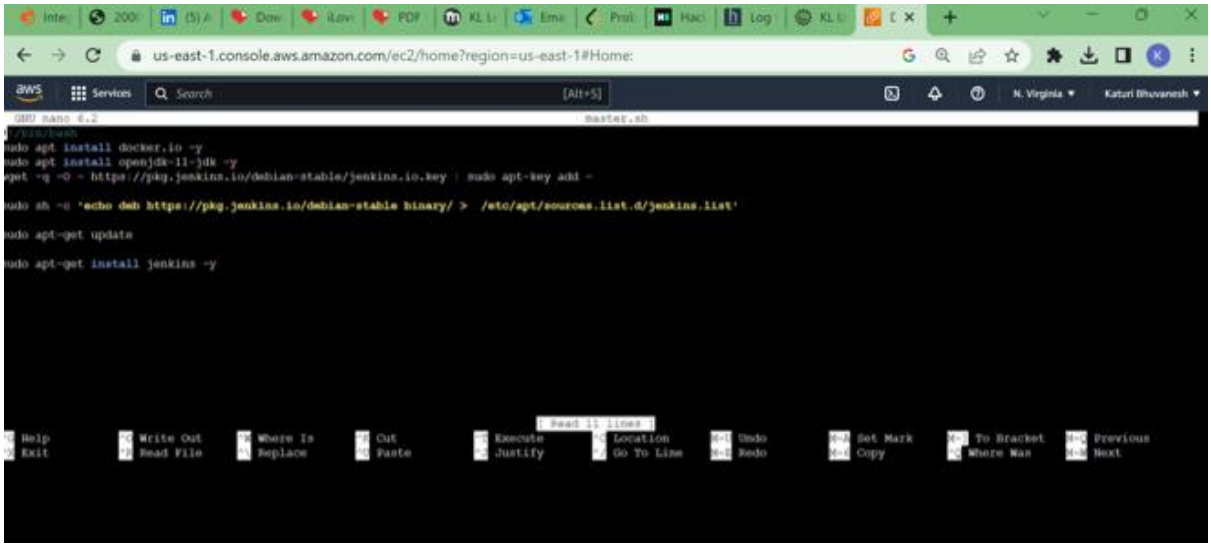
. Ansible playbook is created which will deploy all the necessary software's on servers.

. 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.

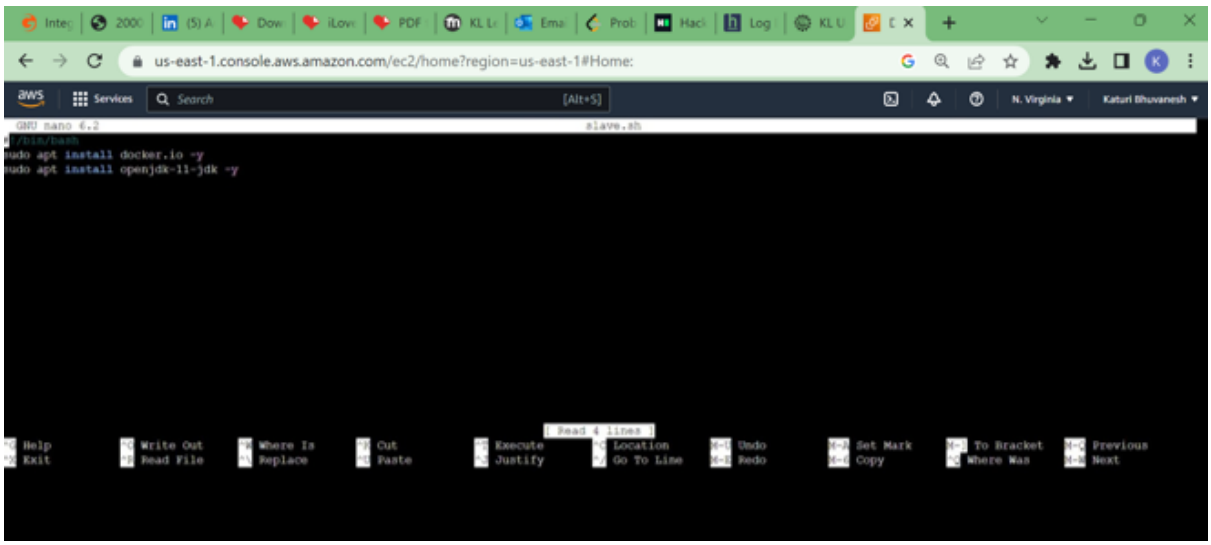
.This file will deploy docker and Jenkins on master server.



4. Creating slave.sh script file:

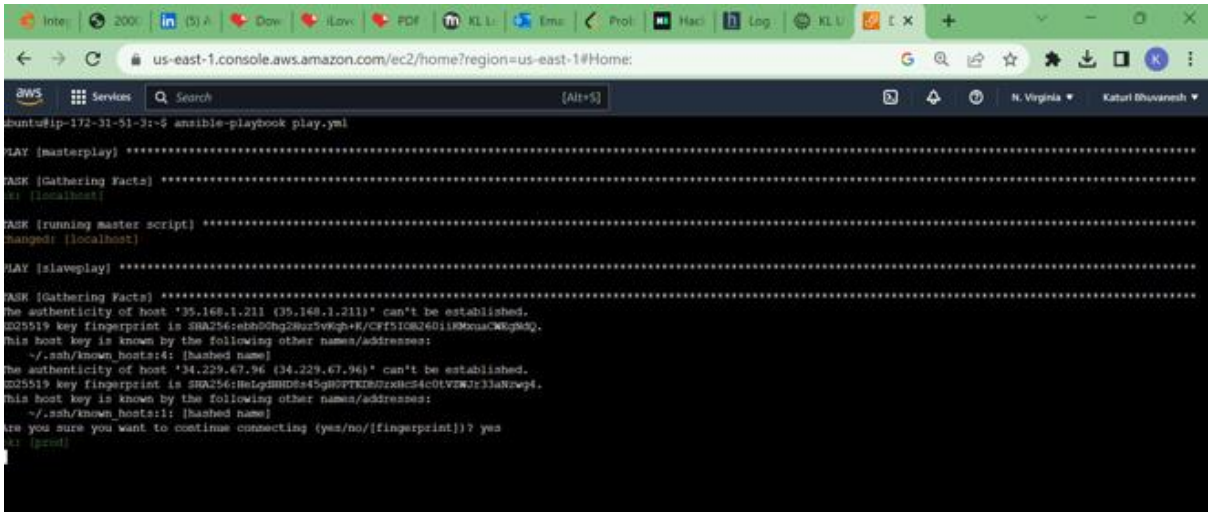
.Create script file for slaves which will deploy all the necessary software's on slaves.

.By running this script file it will install Docker and Java on slave servers.



5. Running Ansible Playbook:

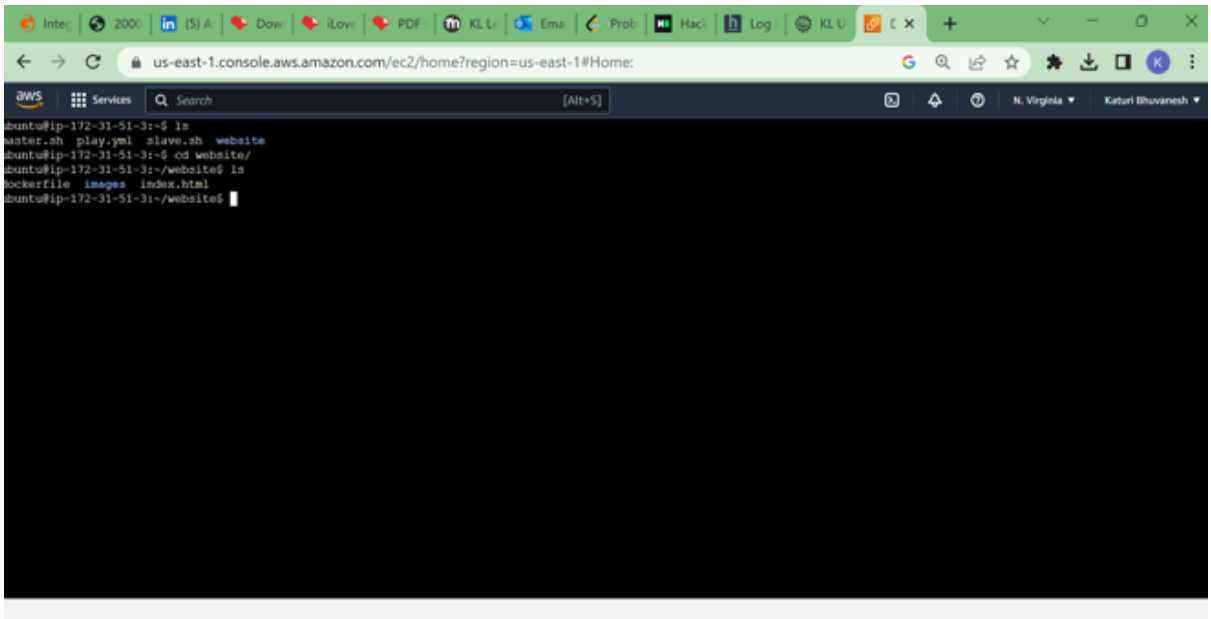
. Thus by running this playbook file we will configure all the servers with all the necessary software's.



6. Cloning project to master server:

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

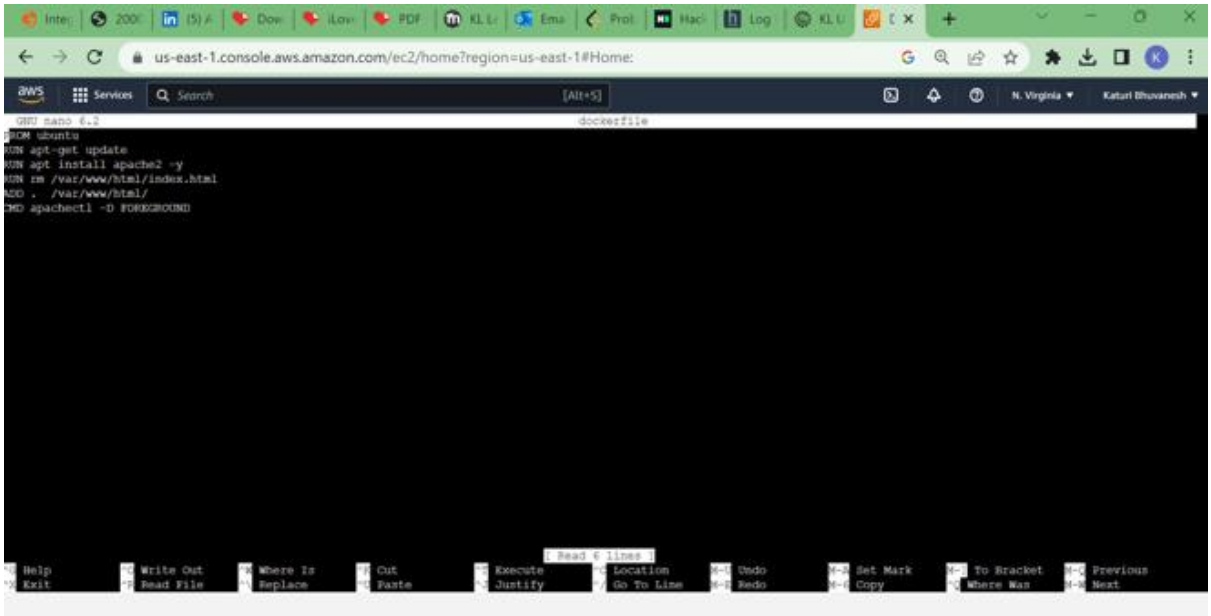
. Thus necessary index file for website and images needed for website is cloned to master server.



7. Creating Dockerfile:

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

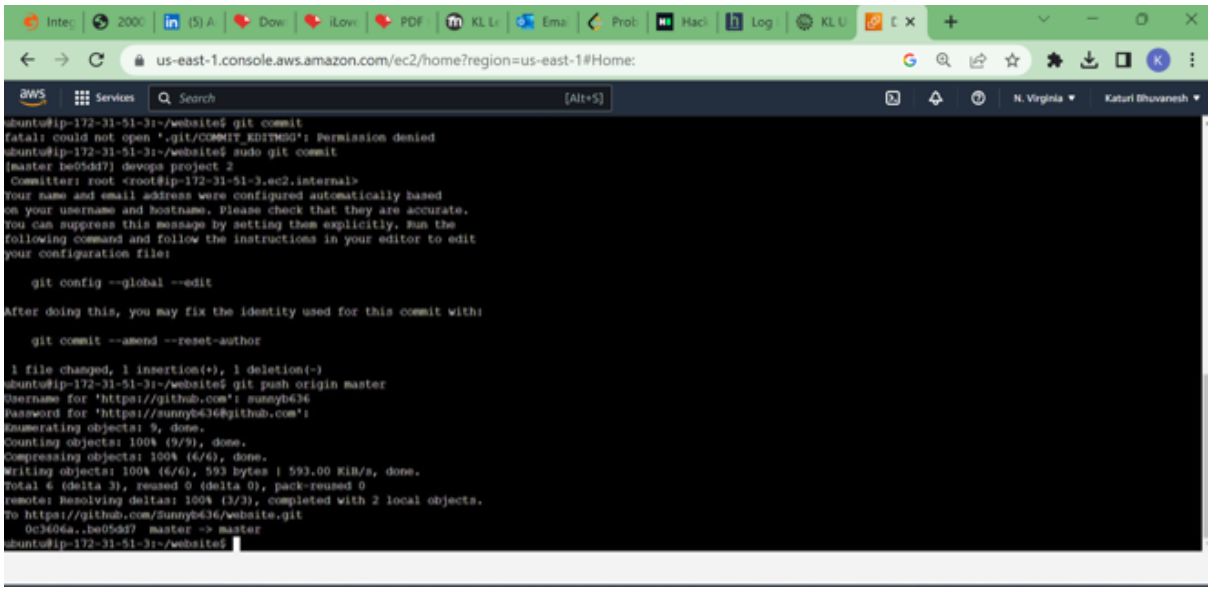
. It will install the apache2 webserver on this container and copies the required files to /var/www/html/ location.



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>".



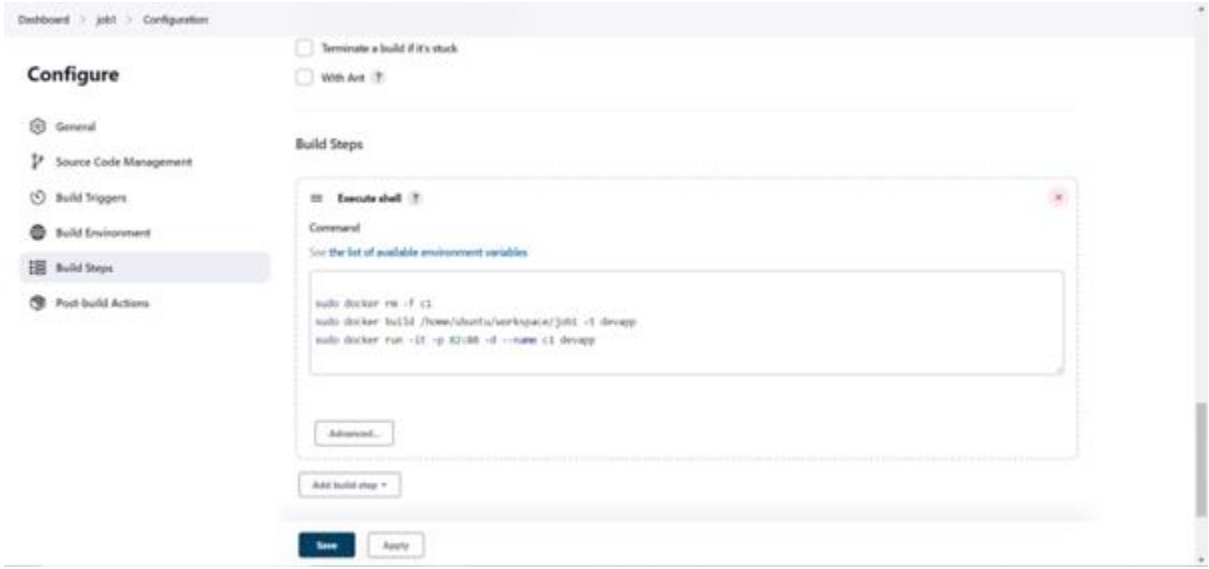
9. Creating job on jenkins cluster:

. 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.

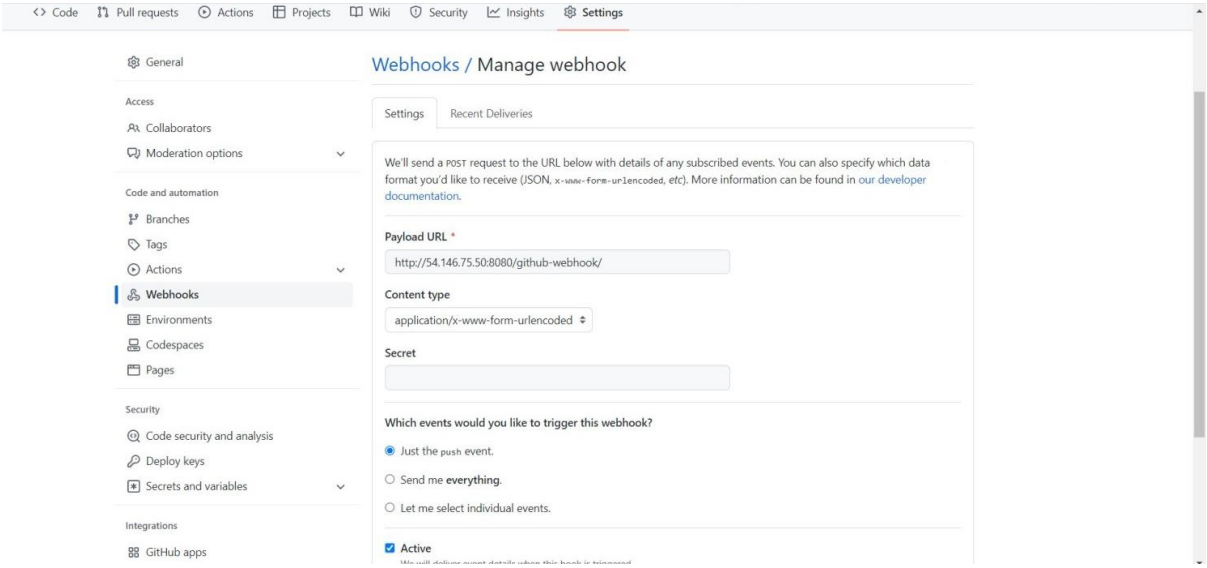
. Also provided build steps with which it will create a docker image and run the container on port 82.



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.

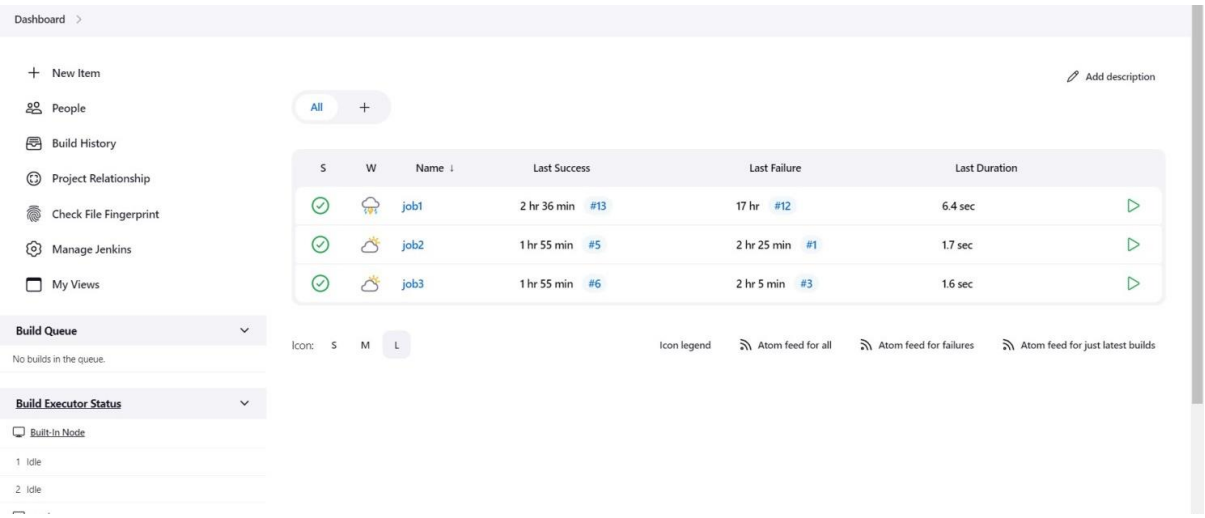


11. Creating 2 more jobs on Jenkins:

.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.



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