**CAPSTONE PROJECT**

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 Softwares 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:

Install all the machine using configuration tool.

1. Git Workflow has to be implemented
2. 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

1. 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'
2. The above tasks should be defined in a Jenkins Pipeline, with the following Jobs Job

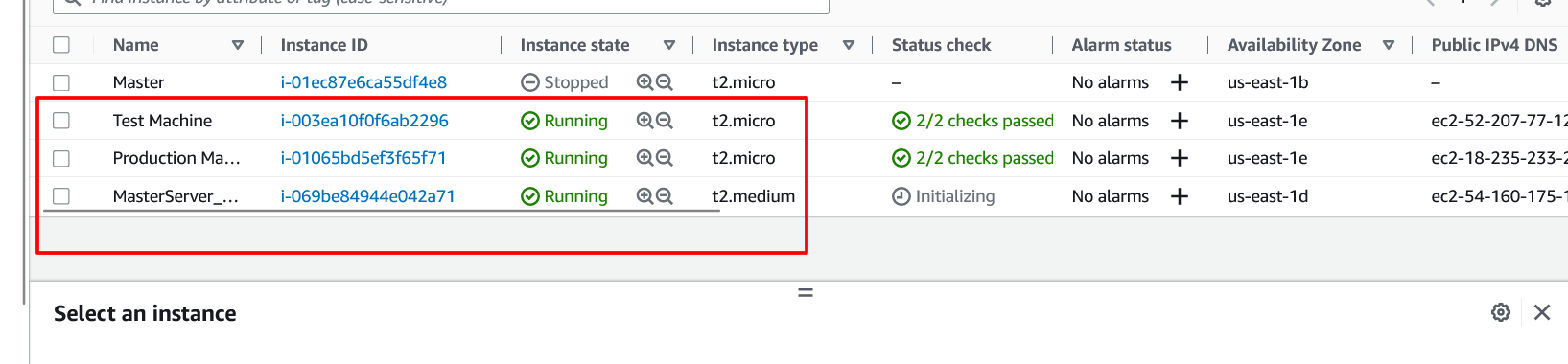
1 - Building Website Job   
2 - Testing Website Job  
3 - Push to Production

1. Since you are setting up the server for the first time, ensure the following file exists on both Test and Prod server in /home/ubuntu/config-management/status.txt. This file will be used by a third-party tool.

This should basically have the info whether apache is installed on the system or not The content of this file, should be based on whether git is installed or not. If apache is installed => Apache is Installed on this System" If apache is not installed => "Apache is not installed on this System" Architectural Advice: Create 3 servers on AWS "t2.micro" Server 1 - should have Jenkins Master, Puppet Master and Nagios Installed Server 2 - Testing Server, Jenkins Slave Server 3 - Prod Server, Jenkins Slave

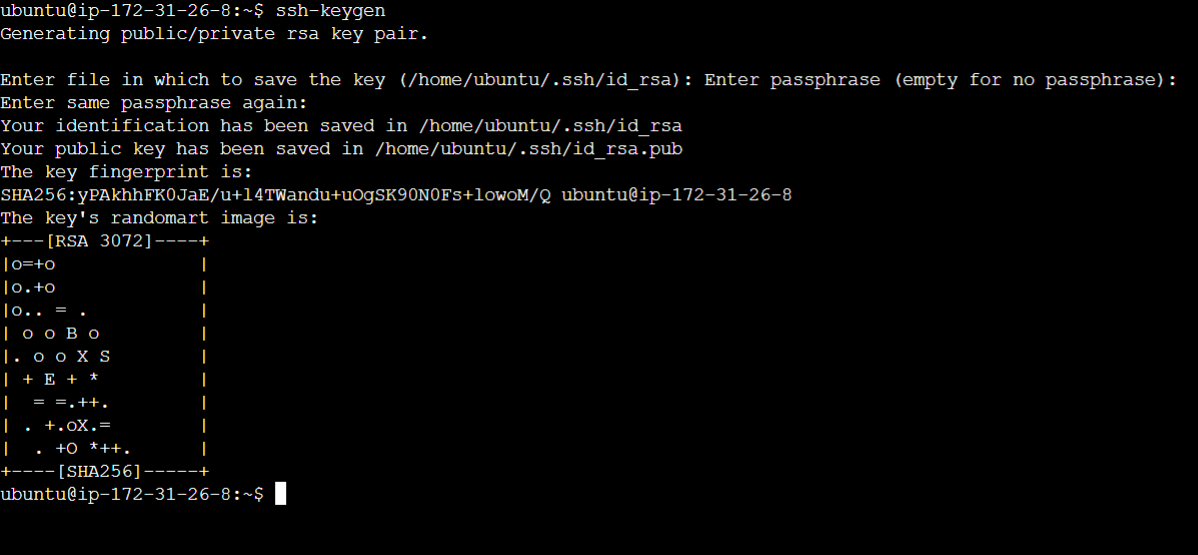
**SOLUTION**:

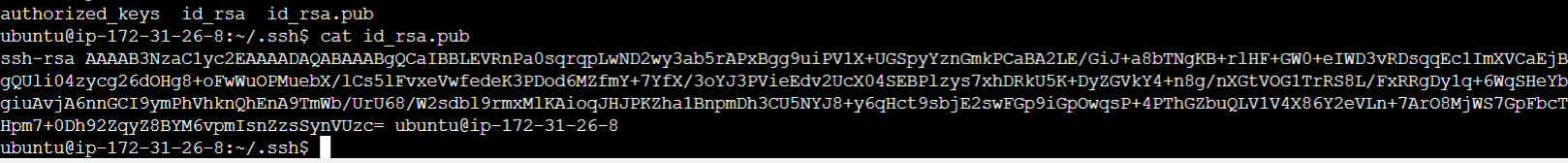
1. Created 3 Virtual machines, one master and two client nodes Test and Production.

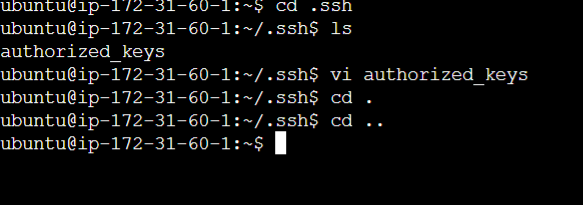


1. Created password less ssh between master and two client machines  
   commands: ***ssh-keygen*** (enter 3 times)

***cd .ssh*** and cat public key and copy the content and go to client machines and paste the content in *authorized\_key file*







1. Installed ansible in the master node using ansible installation script

*#!/bin/bash*

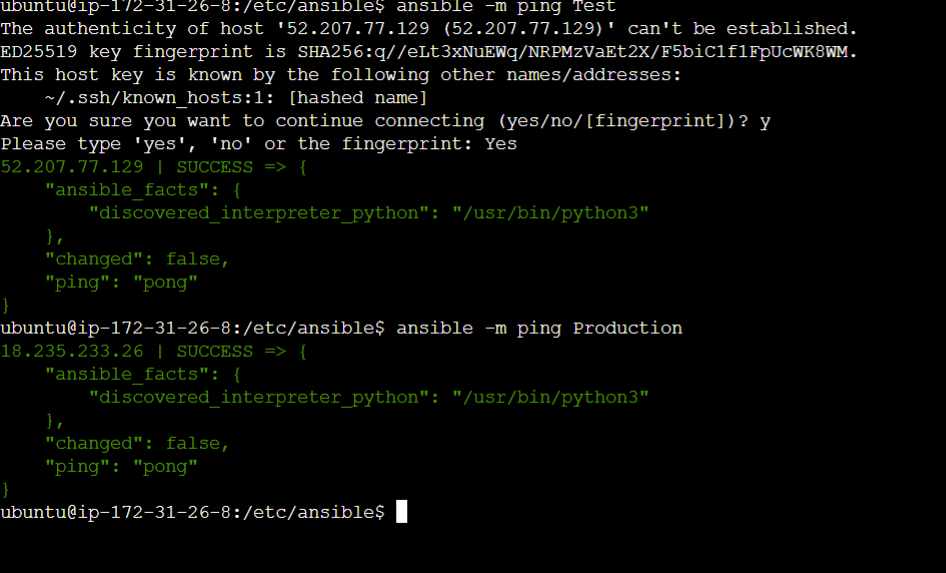
*sudo apt update*

*sudo apt install software-properties-common*

*sudo add-apt-repository --yes --update ppa:ansible/ansible*

*sudo apt install ansible*

1. Used **ssh** and **private Ips** of client users to connect from master node. Also checked the connection status from client nodes.



1. cd /etc/ansible/ to go to ansible directory and configure a file to install Jenkins, Docker, and Java in the nodes.

---

- hosts: localhost

become: true

name: Installing java and Jenkins

tasks:

- name: master task

script: jenkins.sh

- hosts: test

become: true

name: Installing docker and java

tasks:

- name: test tasks

script: docker.sh

- hosts: production

become: true

name: Installing docker and java

tasks:

- name: production tasks

script: docker.sh

1. Create two files jenekins.sh and doker.sh files to install dependency for this project.

#!/bin/bash

sudo apt-get update

sudo apt-get install docker.io -y

#!/bin/bash

sudo apt-get update

sudo apt-get install git-all -y

sudo apt-get install default-jdk -y

curl -fsSL https://pkg.jenkins.io/debian/jenkins.io.key | sudo tee \

/usr/share/keyrings/jenkins-keyring.asc > /dev/null

echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \

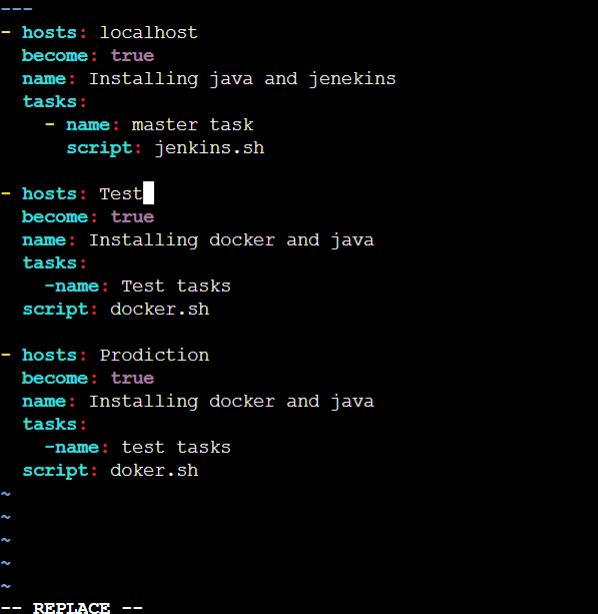
https://pkg.jenkins.io/debian binary/ | sudo tee \

/etc/apt/sources.list.d/jenkins.list > /dev/null

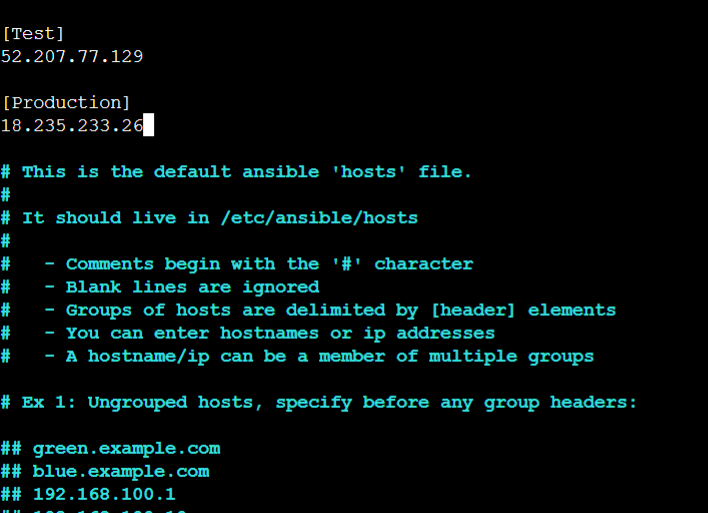
sudo apt-get update

sudo apt-get install jenkins -y

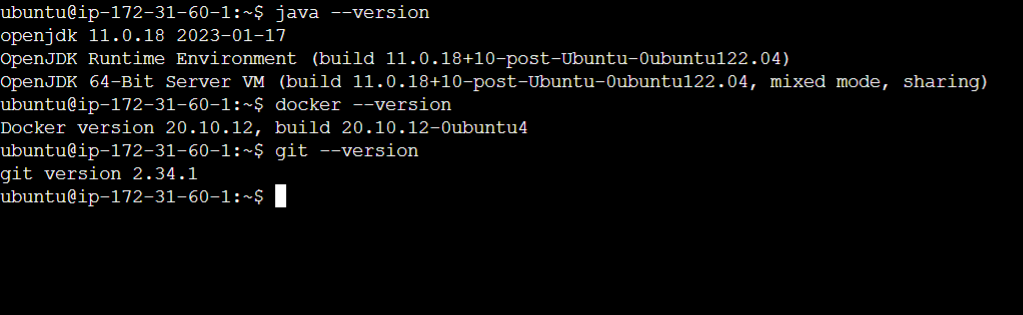
1. Then wrote the ansible playbook



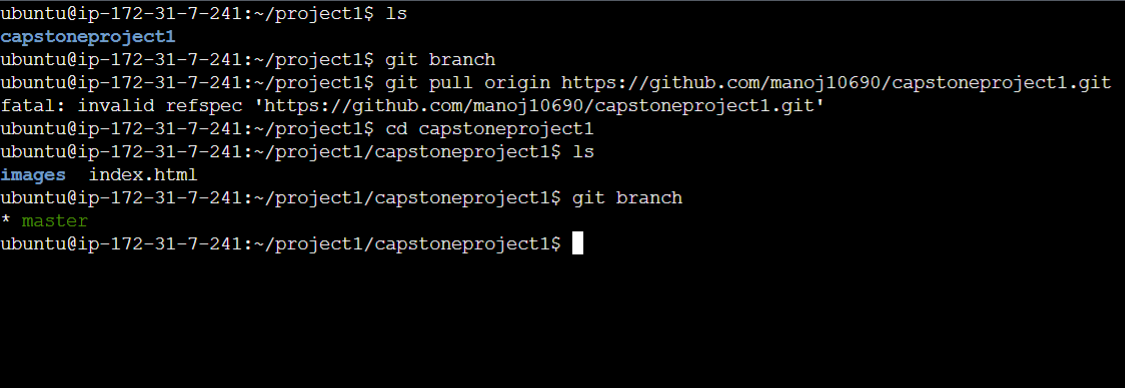
1. Opened the host files using ***sudo vi hosts*** and added the client machines Ips in the host record.



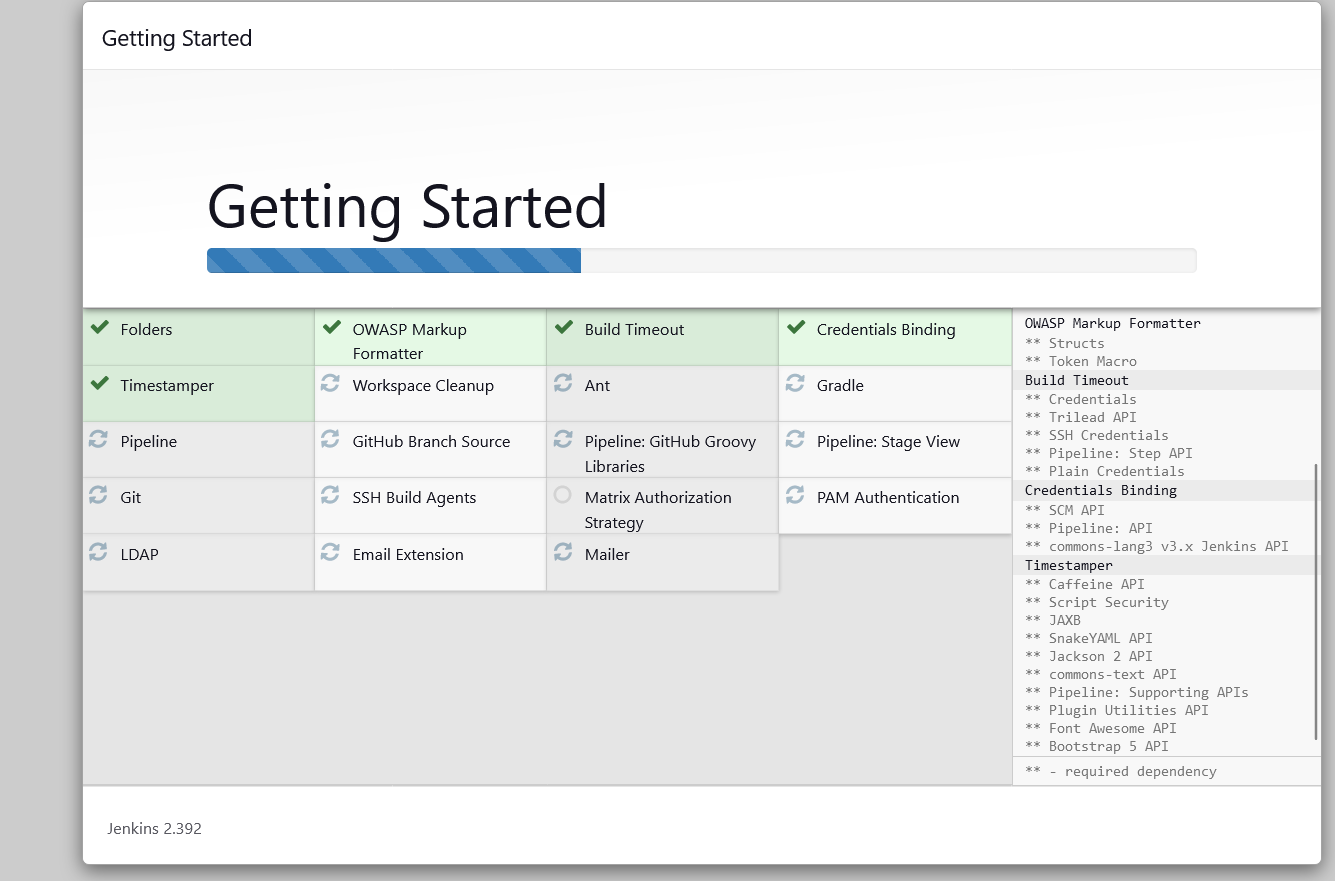
1. Ran the command (***ansible-playbook install.yaml***) and installed the dependencies and checked them on the nodes.

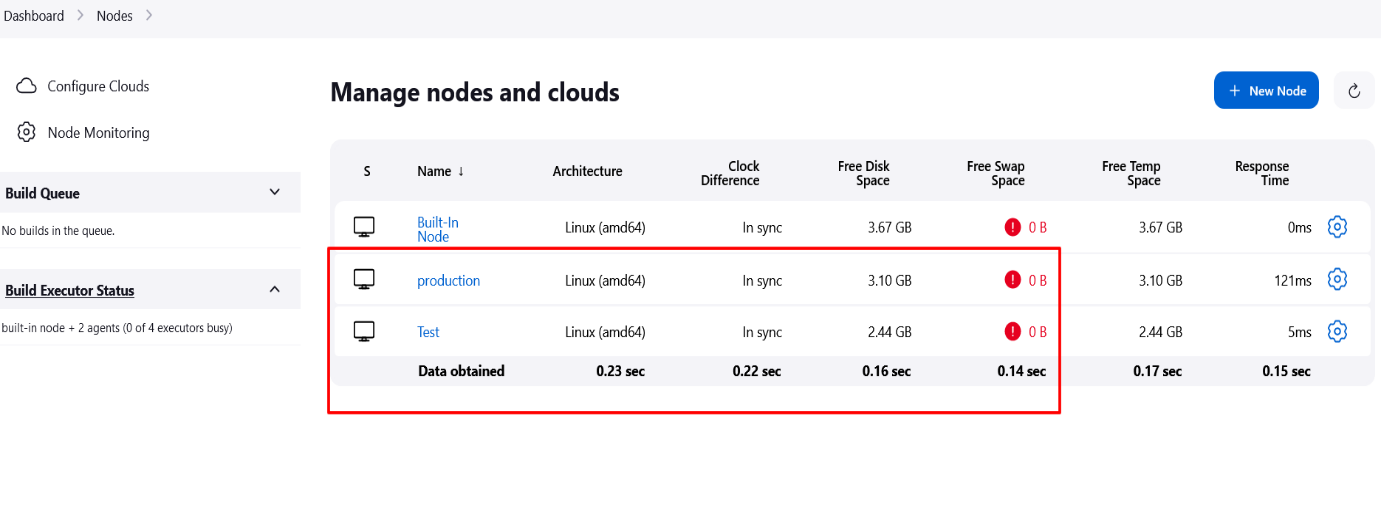


1. Once the nodes and all the dependencies are set up, I opened the github link in provided in the project and forked it with my Github account. Once I cloned the project link, I cloned the project with the local git repository (*git clone <* [*https://github.com/manoj10690/capstoneproject1.git*](https://github.com/manoj10690/capstoneproject1.git)>

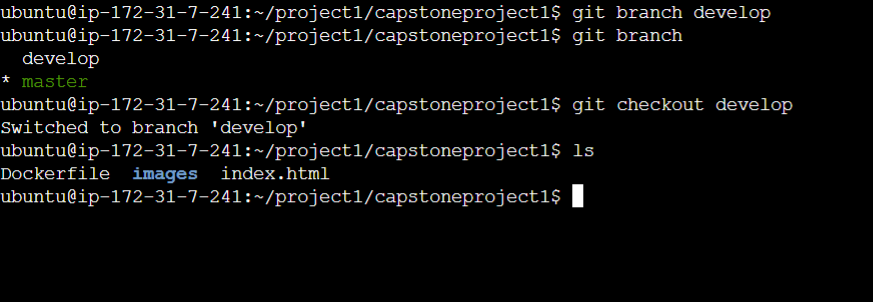


1. After setting up the git repository in the master node, I set up the Jenkins server and created two nodes “Test and Production” for this project.





1. After setting up the Jenkins server, created a developer branch and then I created docker file as per the requirement and committed the file to git.



***sudo vi Dockerfile***

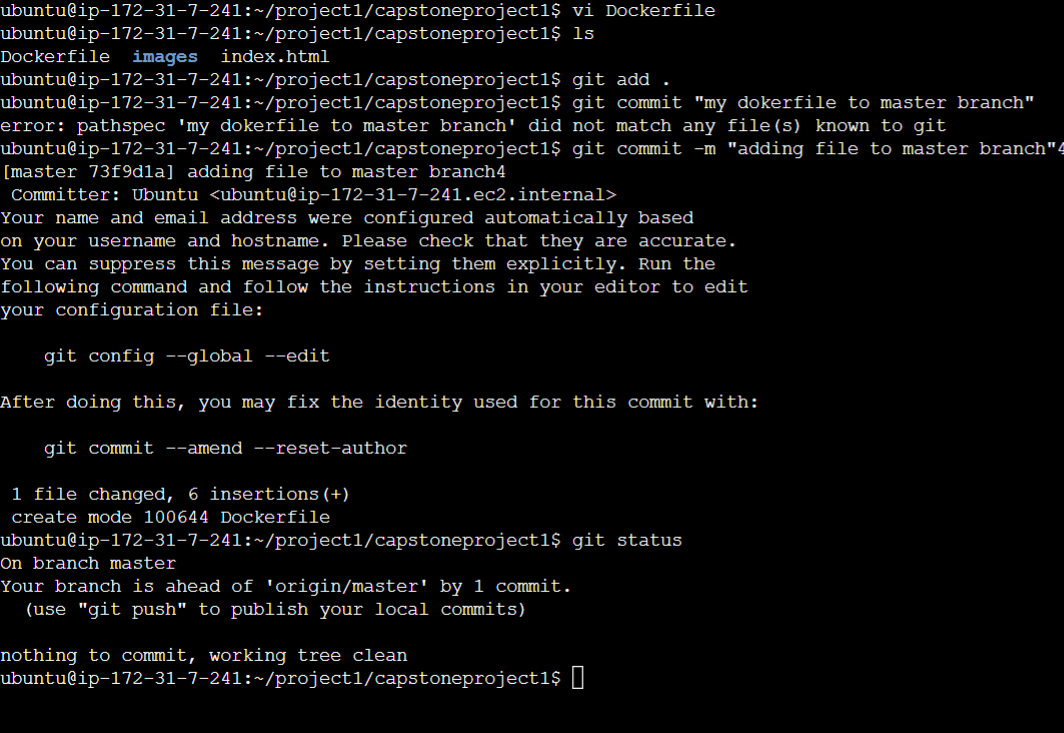
*FROM ubuntu*

*RUN apt update*

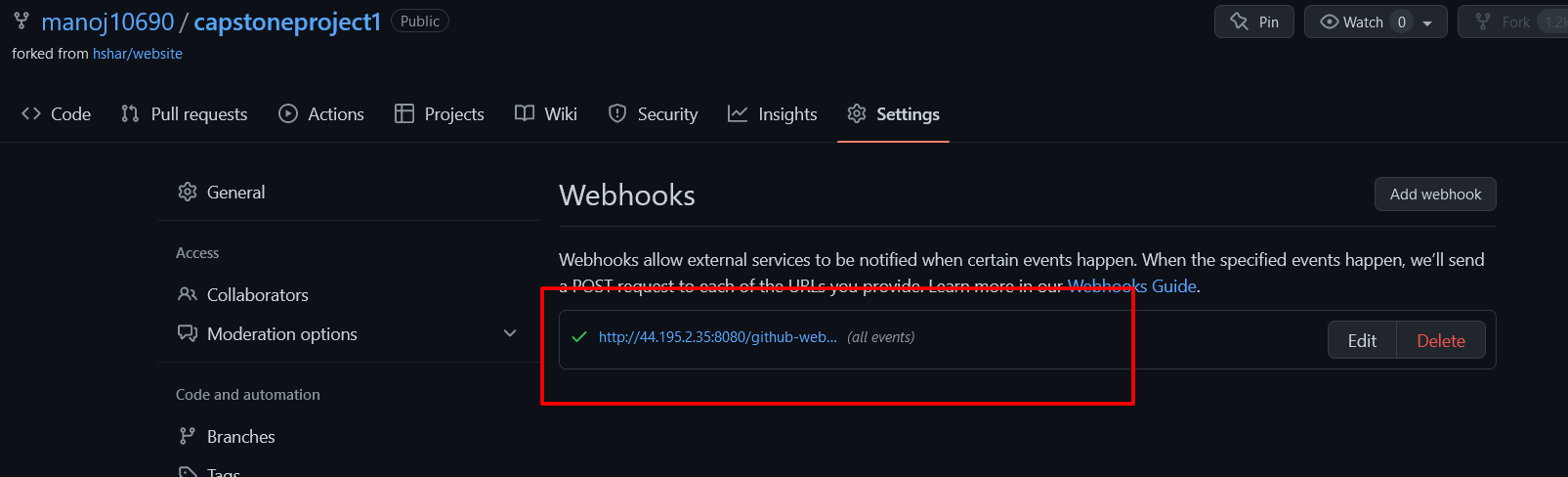
*RUN apt install apache2 -y*

*ADD . /var/www/html*

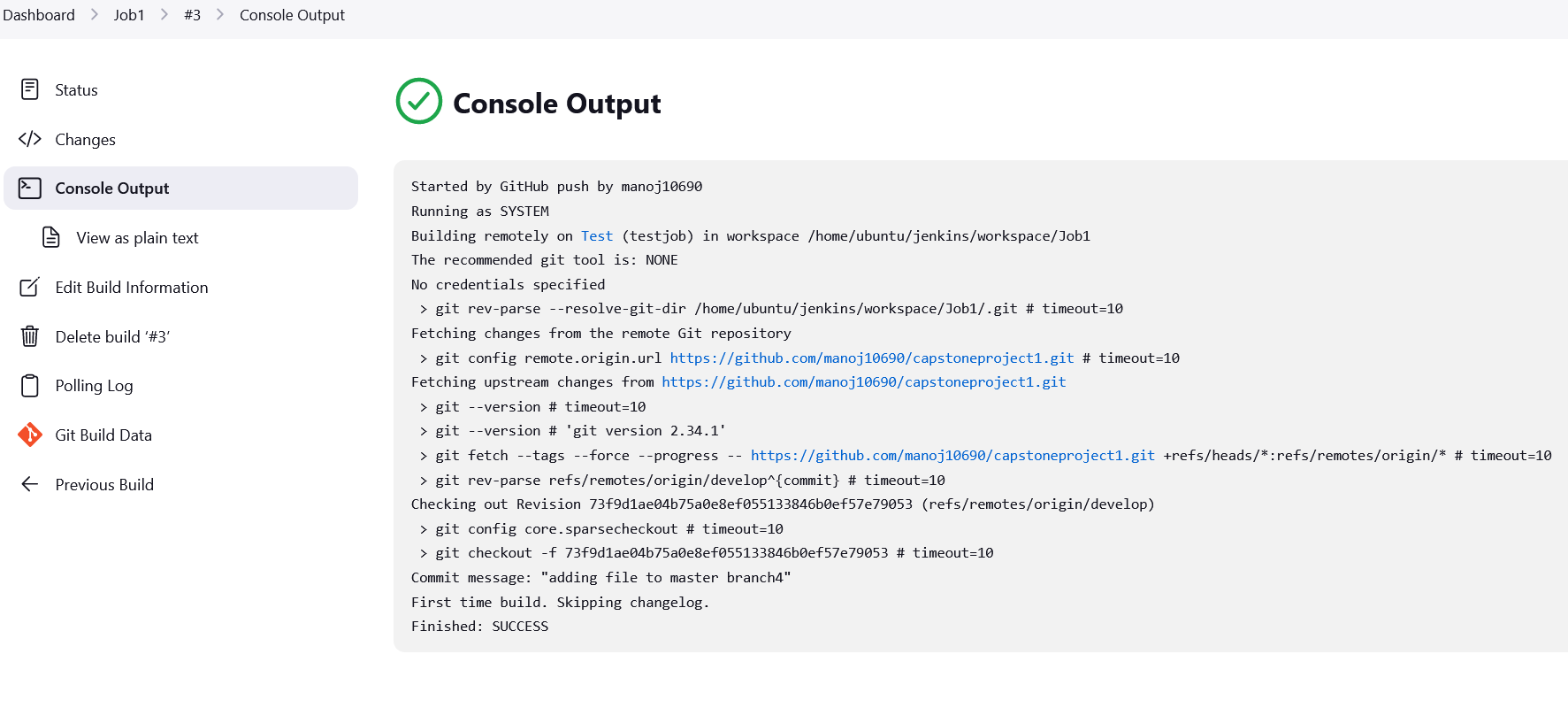
*ENTRYPOINT apachectl -D FOREGROUND*



1. After creating the Docker file, I created the webhook for the purpose of running the tests on test server.



1. Once the branches were set up and ready, I created 1st job (**Job1**) to test and build the container using the Docker image and test the job and it was successful.



1. After successfully running the first job by pushing the changes on the **developer branch**, configured the **Job1** to create container using docker file. Provided the commands and tested the results and it was successful as shown in the screenshots.

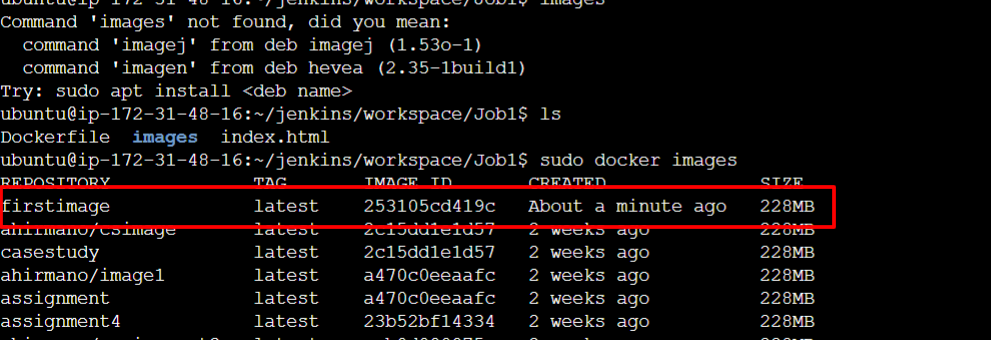
Commands used in Execute shell:

sudo docker rm -f $(sudo docker ps -a -q)

sudo docker build . -t masterapp

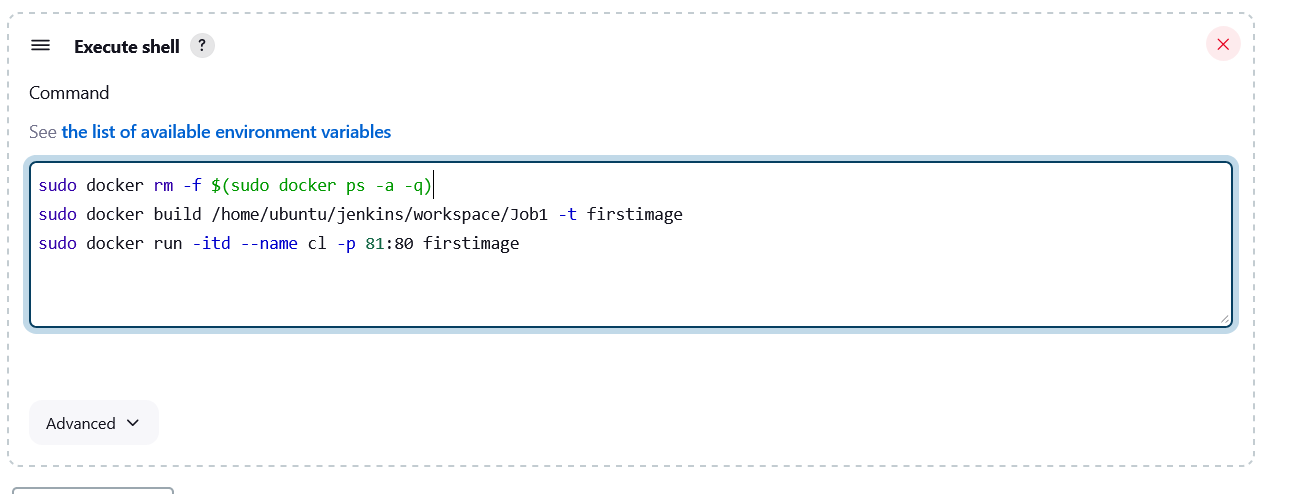
sudo docker run -itd --name websitecontainer -p 82:80 masterapp



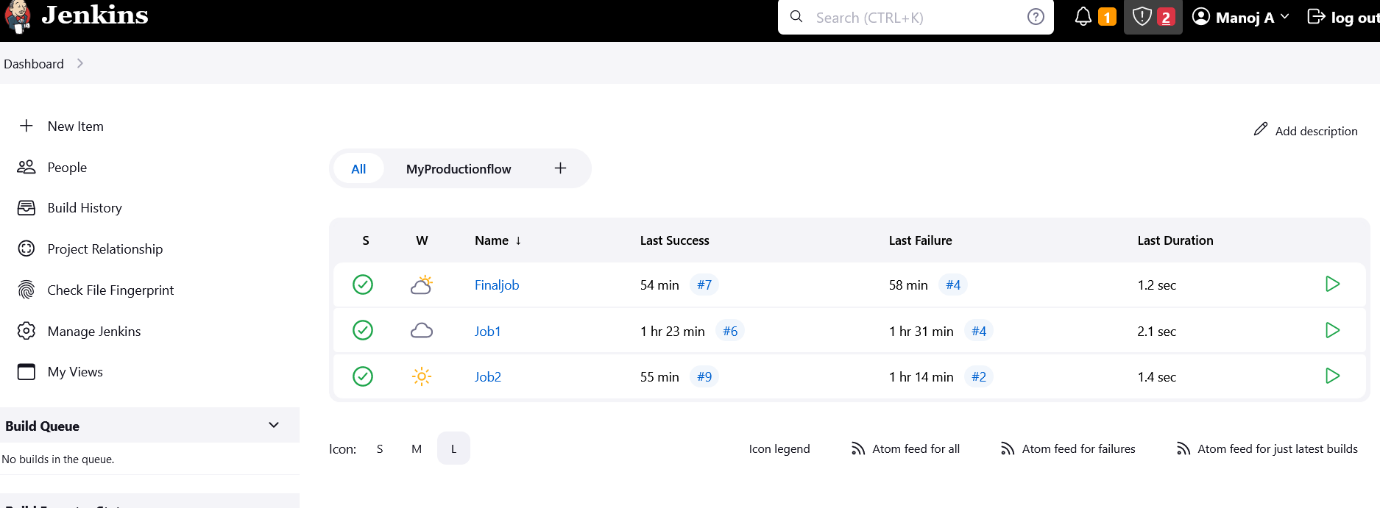


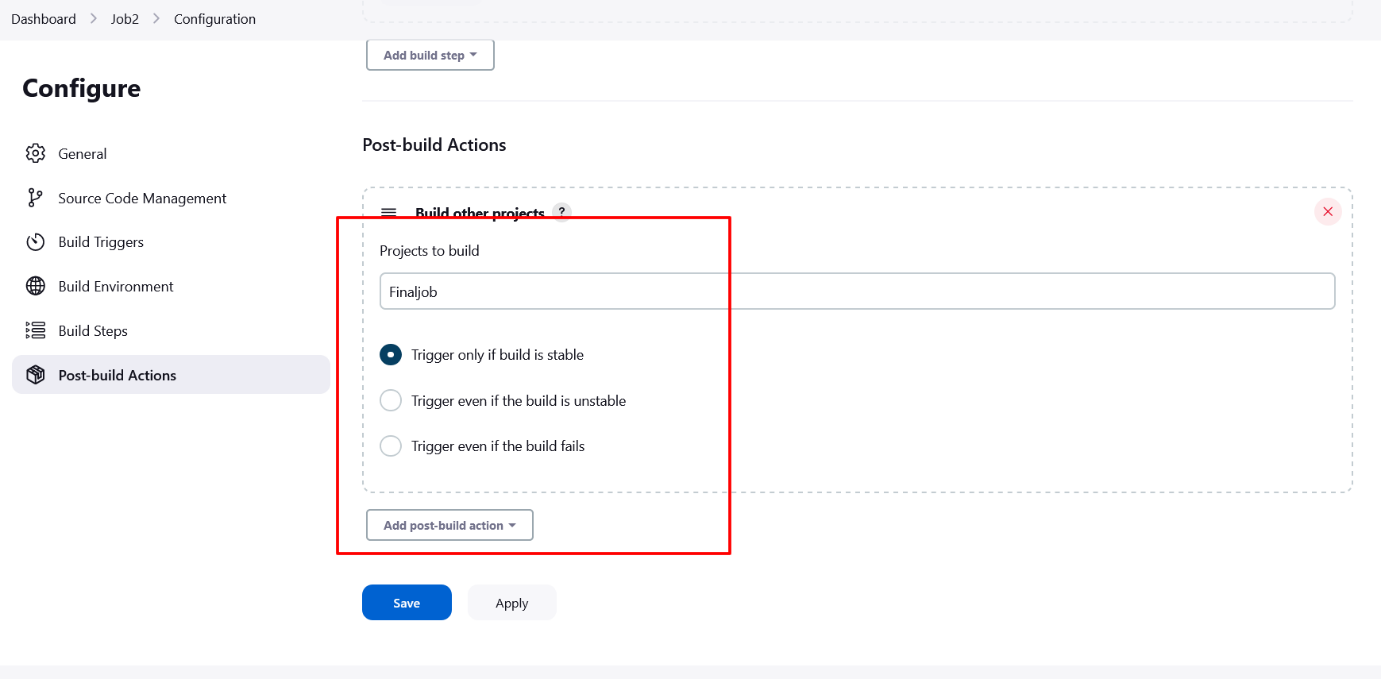


1. After creating the first container, changed the shell commands to run the container again without any errors.

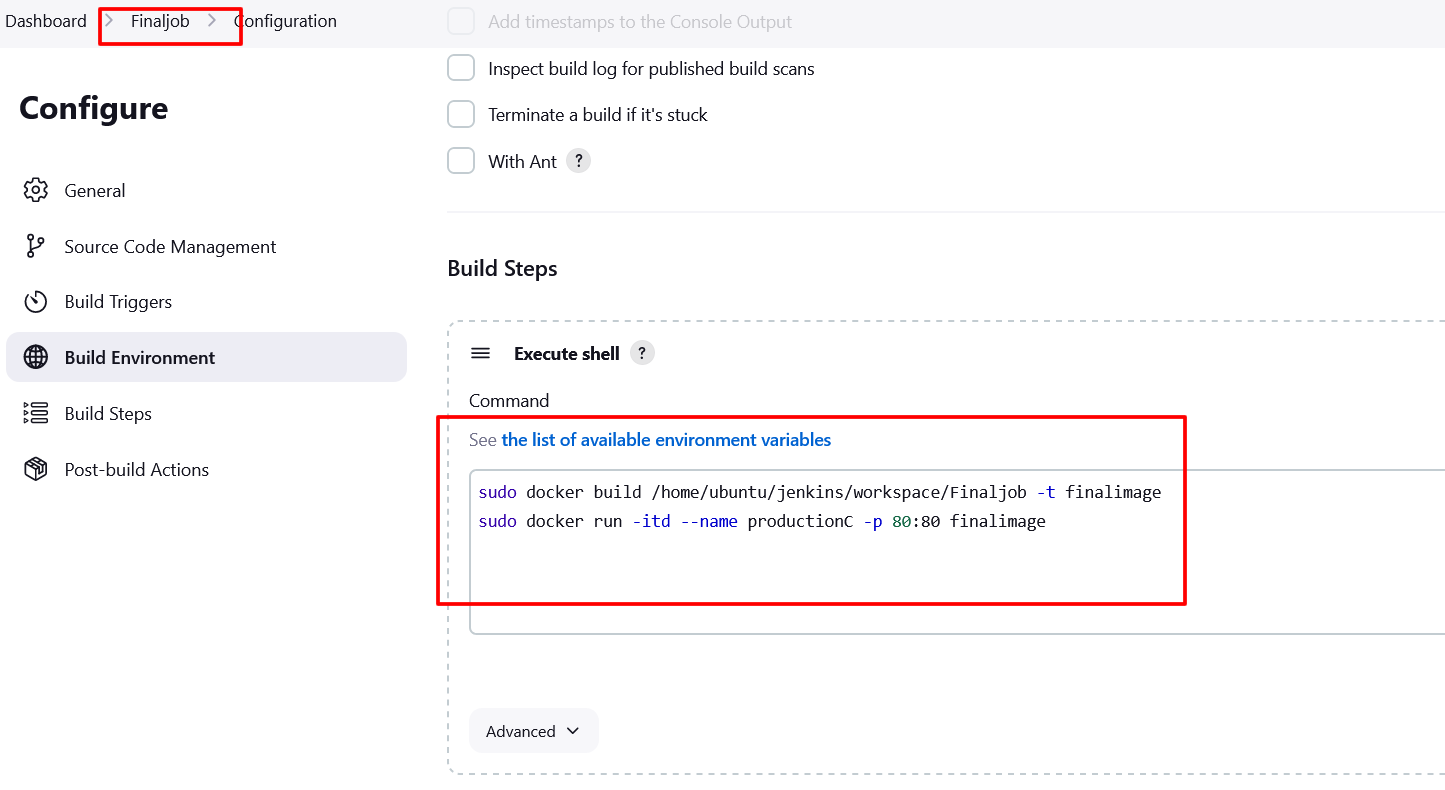


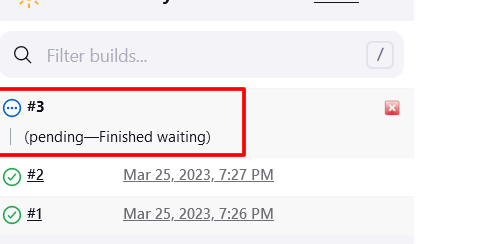
1. After checking the Job1, created another two jobs Job1 and Finaljob to complete the project. If the job2 on test server is successful, it should trigger the Finaljob to be completed on the production server.

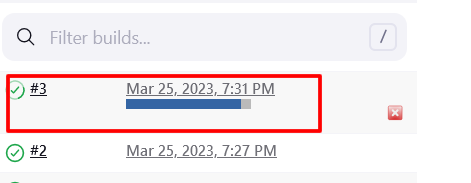
****

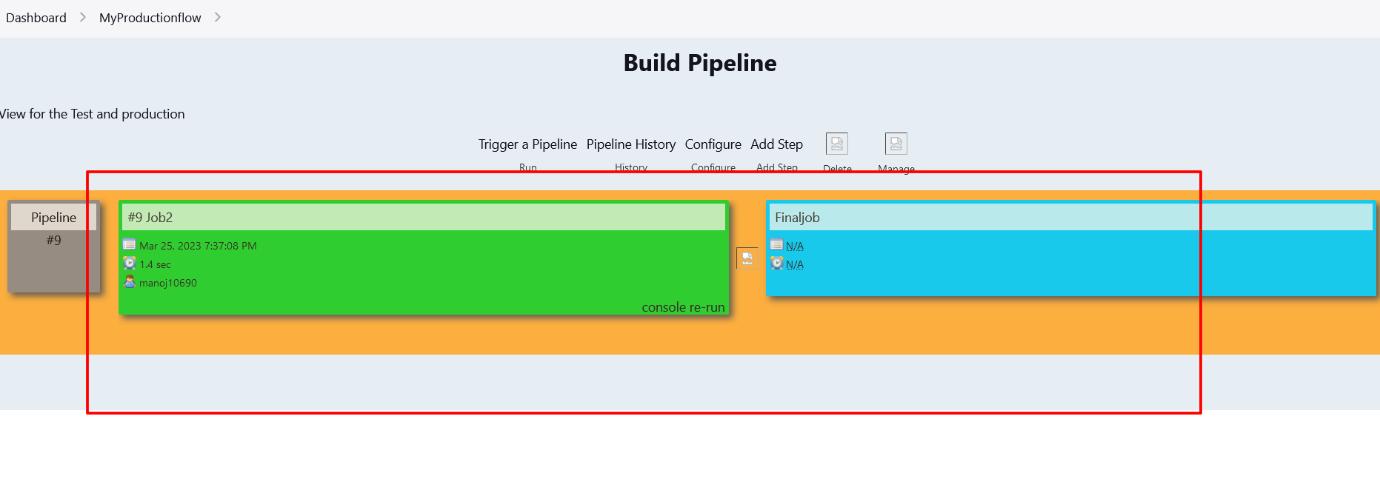
****

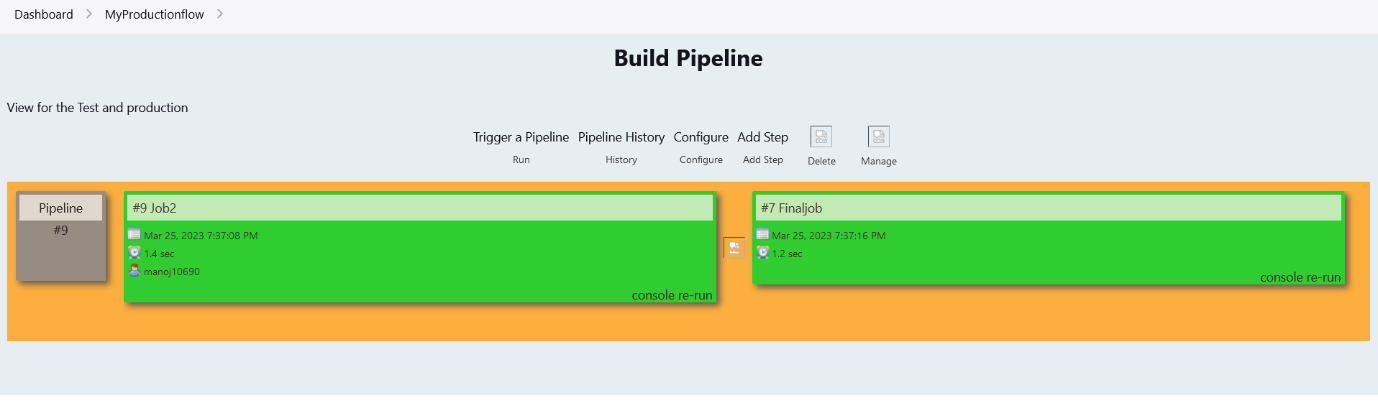
1. Build job and added Post build trigger and script to run the container on the Production environment to run the job only if the Job2 is successful. The project seems to be working as intended.

****

****

****

****

****

**Conclusion**: By doing this project, I understand that by implementing the git workflow and then creating the Jenkins CI pipelines, it can help to eliminate the errors and the production environment can run without any issues. If there are any issues, the test server would recognize it and the job will not be carried forward, we can work on the error and once the issue is fixed, we can run the job again.