DEVOPS TASK

TASK DETAILS: SYSTEM PROVISIONING

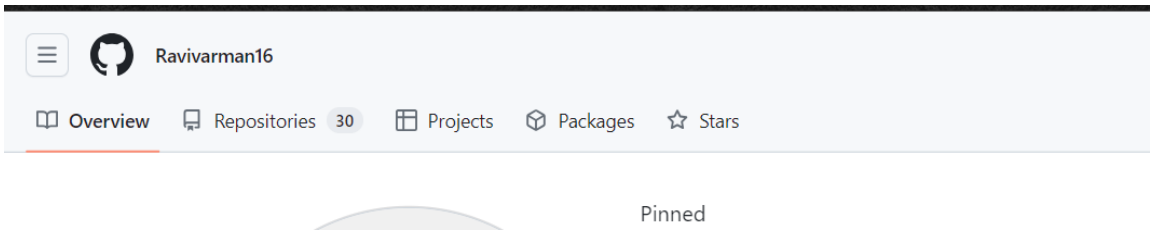
- Create a small web application to display “Hello GUVI GEEK”
- Push this application to a specific folder of the given GitHub repository.
- Create a job in Jenkins and make a build this application.
- On Every Commit, it has to Build a pipeline of the Application.
- If needed Use Pipeline as a Code, for Building the entire pipeline.
- Using Jenkins and Docker Plugin, create an Image of the developed web application.
- Push the Image Finally to the Docker Hub and send the URL image of the Docker.

Tools Needed: GIT, Jenkins, Docker Hub, AWS EC2 instance(ubuntu)

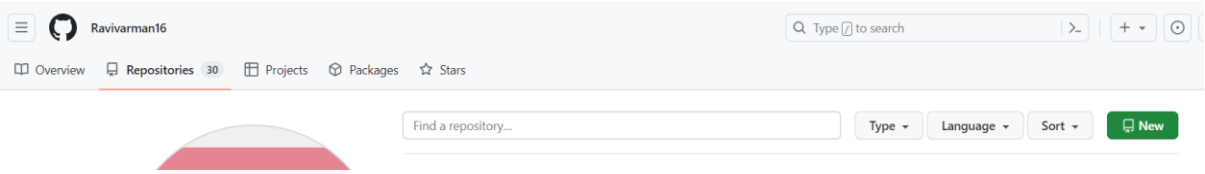
SOLUTION:

Creating GitHub repository:

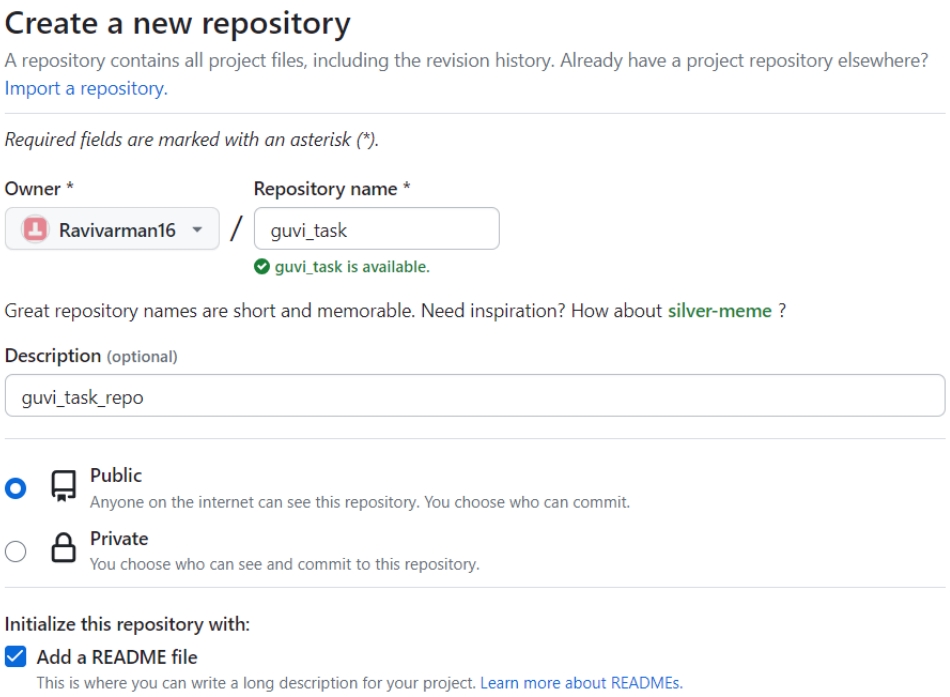
- First, I am login into my github repository:



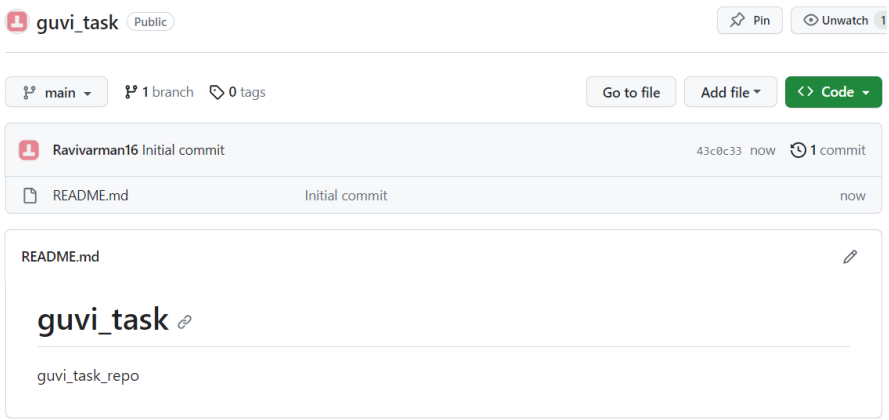
- Then under repositories, click that one where we can able to find the new option on the right side, click that one to create a new repository for this task:



- Then I am naming this repository as **guvi_task** and making this repository as **public** so that everyone can see this repository, finally adding **readme file** and click create repository:



- We can able to see the repository has been created successfully.



****

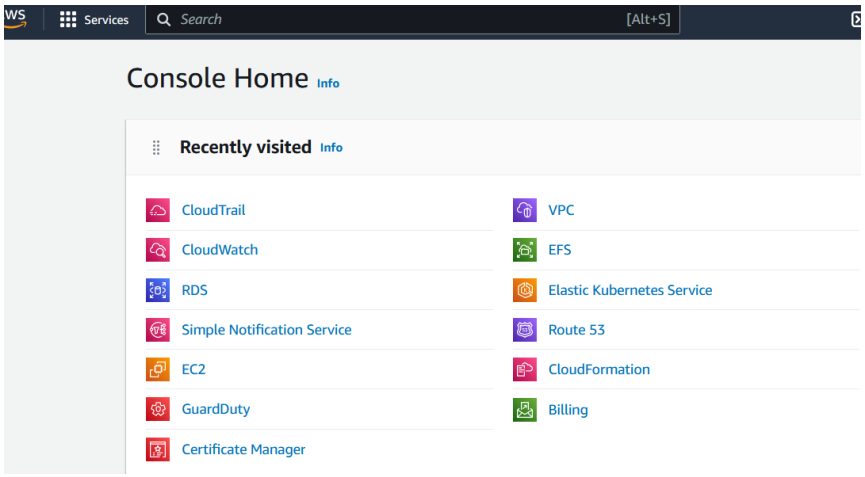
Creating a required EC2 instances for this task:

Required instances:

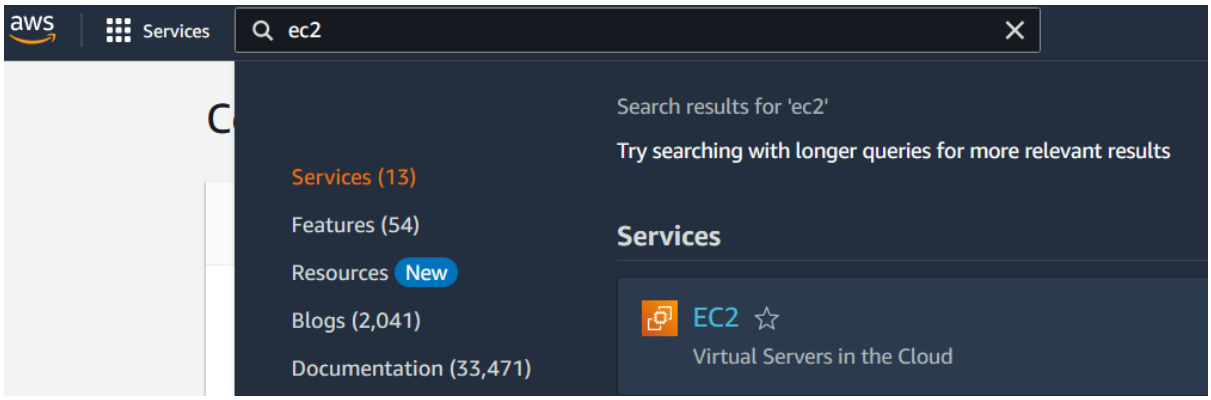
- **Ansible_master**
- **Version_control_server**
- **Jenkins_master_server**
- **Testing_server(testing)**
- **Production_server(live)**

Steps:

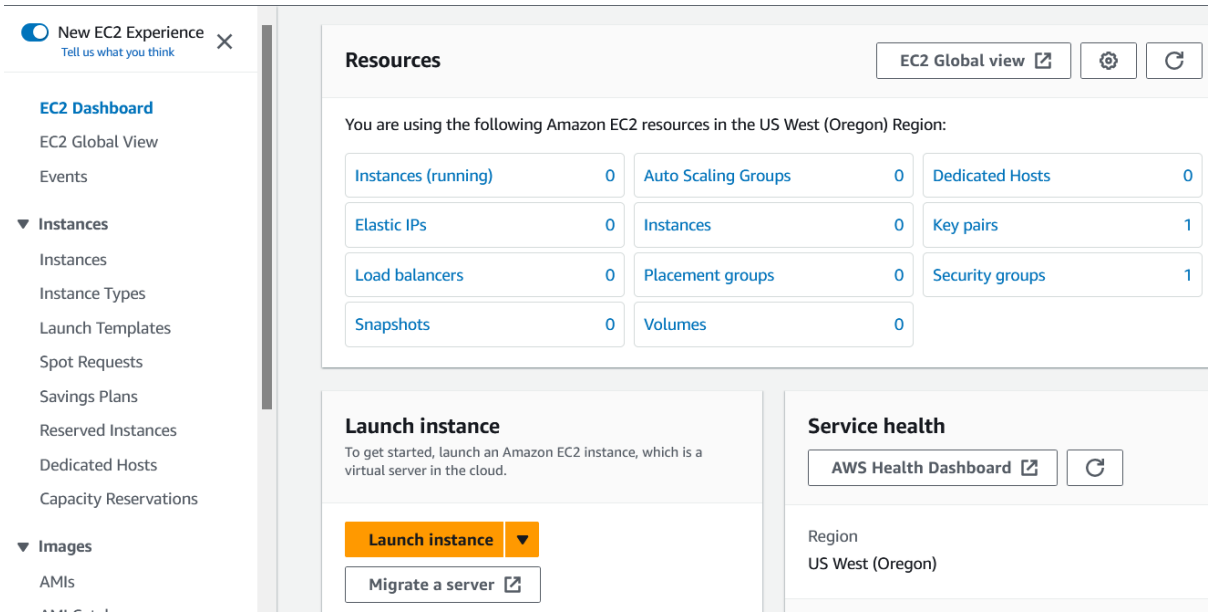
- First, we need to login into AWS management console:



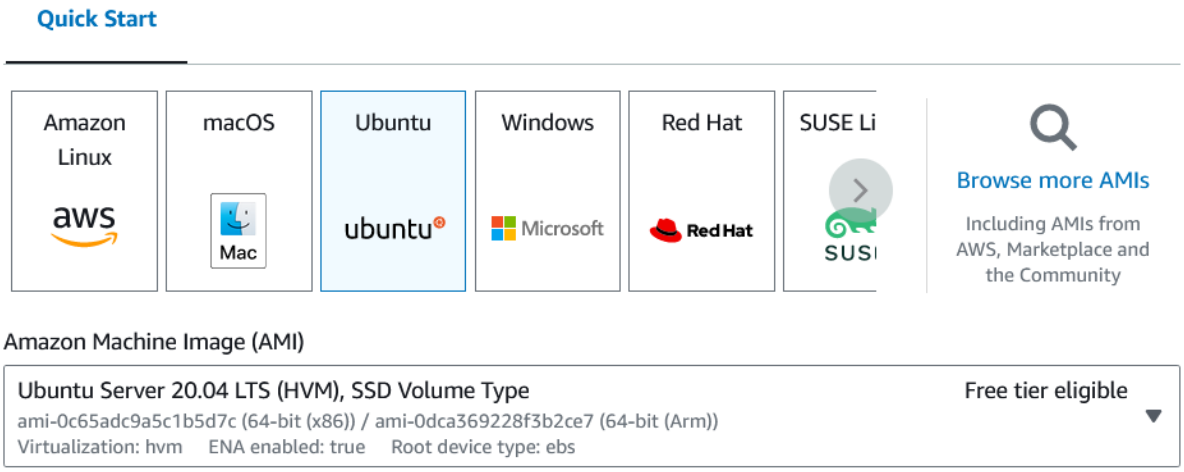
- Then we need to search EC2 on the service session, and click that



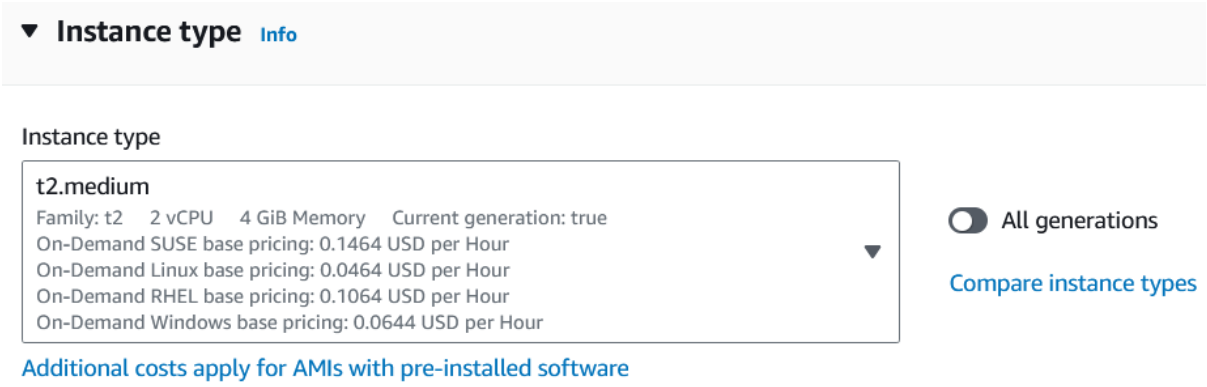
- Then on the EC2 management console click launch instances,



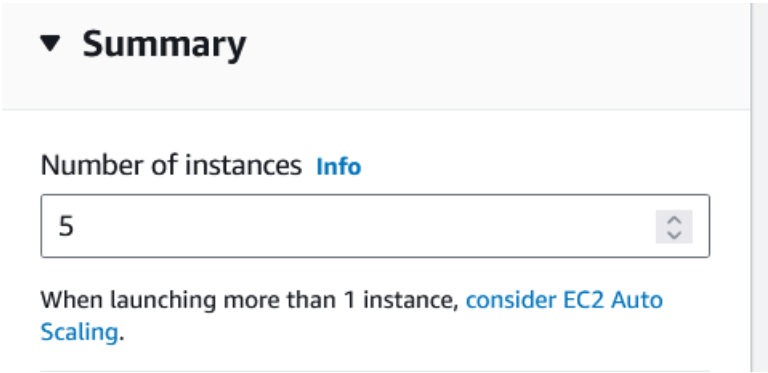
- Then I am selecting the OS as **Ubuntu 20.04** for all the servers:



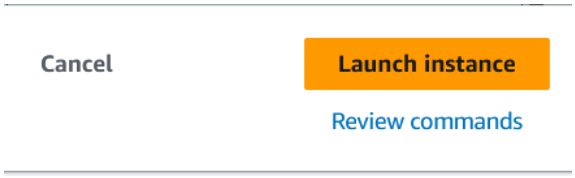
- Then for the instance power and ram supporting I am selecting **t2.medium** for all the servers:



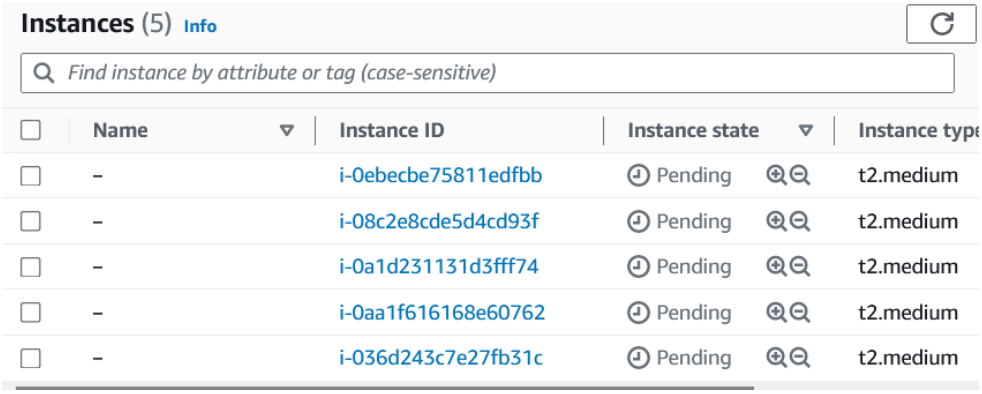
- Here we need 5 servers or instances for that we selecting number 5, so at the time of creating instances I will create 5 instances with the same specifications:



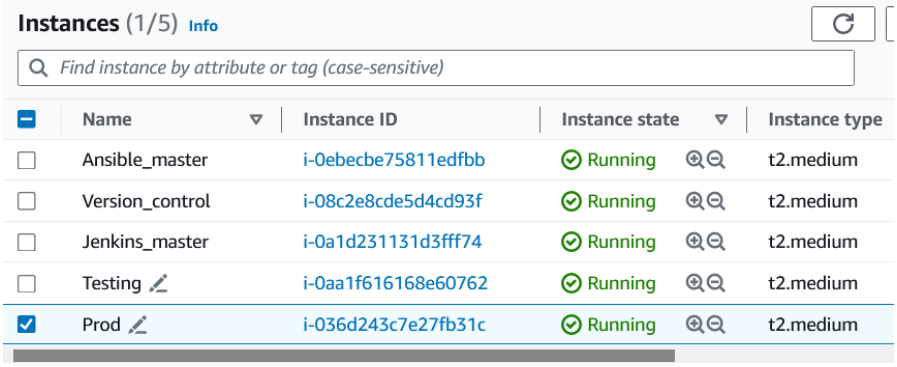
- Review the configurations, and click create instances:



- The instances have been created successfully,



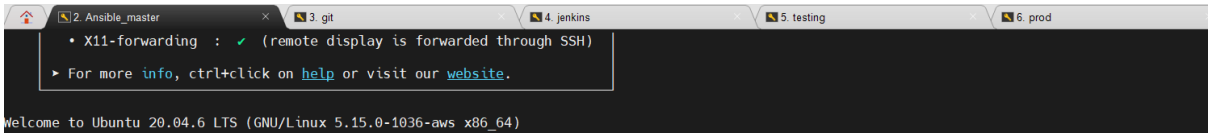
- Naming the instances according to the task specifications:



****

Ansible cluster configuration: [for installing required software's]

- Connecting instances via instance connect or with putty:



Then on all the instances I am making some changes for ansible cluster configuration:

- Creating password for root user: **passwd root**

```
root@ip-172-31-17-156:/home/ubuntu# passwd root
New password:
Retype new password:
passwd: password updated successfully
```

- Go inside the directory: **vi /etc/ssh/sshd_config**

Under this folder:

Remove the hash near port 22

```
Include /etc/ssh/sshd_config.d/*.conf

Port 22
```

Under authentication: **PermitRootLogin into yes**

```
# Authentication:

#LoginGraceTime 2m
#PermitRootLogin prohibit-password
#StrictModes yes
```

```
#LoginGraceTime 2m
PermitRootLogin yes
```

Password authentication changing it into yes

```
# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication no
```

```
# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication yes
```

Then restarting the sshd for to read all the changes that we made:

```
root@ip-172-31-28-44:/home/ubuntu# systemctl restart sshd
root@ip-172-31-28-44:/home/ubuntu#
```

Note: this must be performed on all the servers or instance except ansible master instance:

Installing ansible on ansible master instance:

- Creating shell file for installing ansible on ansible master instance and executing it.

```
root@ip-172-31-17-156:/home/ubuntu# touch ansible.sh
root@ip-172-31-17-156:/home/ubuntu# vi ansible.sh
root@ip-172-31-17-156:/home/ubuntu# chmod ansible.sh
chmod: missing operand after 'ansible.sh'
Try 'chmod --help' for more information.
root@ip-172-31-17-156:/home/ubuntu# chmod 744 ansible.sh
root@ip-172-31-17-156:/home/ubuntu# ./ansible.sh
Hit:1 http://us-west-2.ec2.archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://us-west-2.ec2.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:3 http://us-west-2.ec2.archive.ubuntu.com/ubuntu focal-backports InRelease
0% [Waiting for headers]
```

Ansible has been installed successfully.

```
ansible [core 2.12.10]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/root/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /root/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.8.10 (default, May 26 2023, 14:05:08) [GCC 9.4.0]
  jinja version = 2.10.1
  libyaml = True
```

Connecting the instance with ansible master for cluster configuration:

```
[git]
172.31.17.250

[jenkins]
172.31.28.44

[test]
172.31.18.41

[prod]
172.31.23.18
~
```

- Making changes under the directory: **vi /etc/ansible/hosts**

I adding four servers to this ansible master instance:

- Then we need to generate the key for connecting the instances: ssh-keygen

```
root@ip-172-31-17-156:/home/ubuntu# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:S3r8ITThw5QPABvH+YBhk8E6CVyYQHNUnf5kFGMs4Jo root@ip-172-31-17-156
The key's randomart image is:
+---[RSA 3072]----+
|=o==OX=.o+.      |
|.+o.==+=.+.      |
| . o...oB        |
|  + o  =.*       |
|   E    S .      |
|       = =       |
|      . = .      |
|       . o .     |
|          .      |
+----[SHA256]-----+
root@ip-172-31-17-156:/home/ubuntu#
```

- After generating the key, I am connecting instance via **ssh-copy-id root@ipaddress**

  Here root is the user of the instance which we had created the password before:

  Once entering command, it will ask us whether we need to connect or not with it. Just enter yes.

  Then it will ask the password of the user, enter the password, we can able to see the instance is connected.

  Like this we need to connect the other instances:

```
root@ip-172-31-17-156:/home/ubuntu# ssh-copy-id root@172.31.17.250
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa.pub"
The authenticity of host '172.31.17.250 (172.31.17.250)' can't be established.
ECDSA key fingerprint is SHA256:u6SulhrSf+PXcRwx0fUKf8LlBy1b4FAPwGz+osVZV8s.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
root@172.31.17.250's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'root@172.31.17.250'"
and check to make sure that only the key(s) you wanted were added.

root@ip-172-31-17-156:/home/ubuntu#
```

Once connecting all the instances, we check whether the instances are connected or not, with the help of the command **ansible all -m ping**

```
root@ip-172-31-17-156:/home/ubuntu# ansible all -m ping
172.31.28.44 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
172.31.17.250 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
172.31.18.41 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
172.31.23.18 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
root@ip-172-31-17-156:/home/ubuntu#
```

Installing the necessary packages on the instances via ansible:

Version control instance : Git

Jenkins master instance : Java, Jenkins

Testing server : Java, Docker

Prod server : Java, Docker

- Creating necessary script for installing packages on the instances:

```
root@ip-172-31-17-156:/home/ubuntu# ls
ansible.sh  docker.sh  git.sh  java.sh  jenkins.sh
root@ip-172-31-17-156:/home/ubuntu#
```

Then creating a main playbook file for installing the packages. After creating the main playbook file, we can check the syntax with the help of the command **ansible-playbook package.yml --syntax-check**

```
root@ip-172-31-17-156:/home/ubuntu# vi package.yml
root@ip-172-31-17-156:/home/ubuntu# ansible-playbook package.yml --syntax-check

playbook: package.yml
root@ip-172-31-17-156:/home/ubuntu#
```

Once the syntax is verified by ansible, we can execute the command to install the packages: **ansible-playbook package.yml**

```
PLAY RECAP *********************************************************
172.31.17.250              : ok=9    changed=3    unreachable=0
172.31.18.41               : ok=9    changed=3    unreachable=0
172.31.23.18               : ok=9    changed=3    unreachable=0
172.31.28.44               : ok=9    changed=3    unreachable=0
```

The packages have been installed by ansible. Now we need to check the respective servers:

Version_control_server:

```
root@ip-172-31-17-250:/home/ubuntu# git --version
git version 2.25.1
root@ip-172-31-17-250:/home/ubuntu# java --version
openjdk 11.0.20.1 2023-08-24
OpenJDK Runtime Environment (build 11.0.20.1+1-post-Ubuntu-0ubuntu120.04)
OpenJDK 64-Bit Server VM (build 11.0.20.1+1-post-Ubuntu-0ubuntu120.04, mixed mode, sharing)
root@ip-172-31-17-250:/home/ubuntu#
```

Jenkins_master:

```
root@ip-172-31-28-44:/home/ubuntu# java --version
openjdk 11.0.20.1 2023-08-24
OpenJDK Runtime Environment (build 11.0.20.1+1-post-Ubuntu-0ubuntu120.04)
OpenJDK 64-Bit Server VM (build 11.0.20.1+1-post-Ubuntu-0ubuntu120.04, mixed mode, sharing)
root@ip-172-31-28-44:/home/ubuntu# jenkins --version
2.414.2
root@ip-172-31-28-44:/home/ubuntu#
```

Testing_server:

```
root@ip-172-31-18-41:/home/ubuntu# java --version
openjdk 11.0.20.1 2023-08-24
OpenJDK Runtime Environment (build 11.0.20.1+1-post-Ubuntu-0ubuntu120.04)
OpenJDK 64-Bit Server VM (build 11.0.20.1+1-post-Ubuntu-0ubuntu120.04, mixed mode, sharing)
root@ip-172-31-18-41:/home/ubuntu# docker -v
Docker version 24.0.5, build 24.0.5-0ubuntu1~20.04.1
root@ip-172-31-18-41:/home/ubuntu#
```

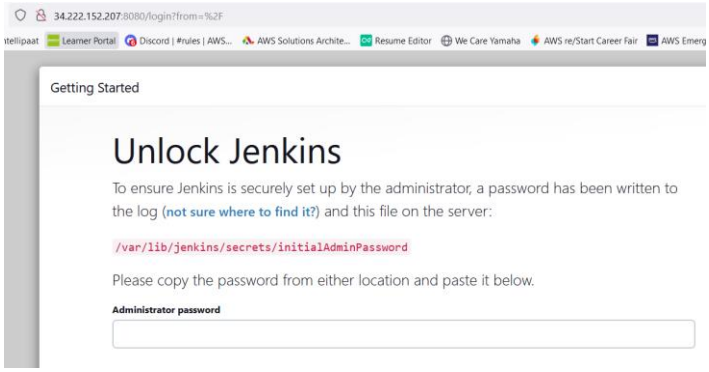
Prod_server:

```
root@ip-172-31-23-18:/home/ubuntu# java --version
openjdk 11.0.20.1 2023-08-24
OpenJDK Runtime Environment (build 11.0.20.1+1-post-Ubuntu-0ubuntu120.04)
OpenJDK 64-Bit Server VM (build 11.0.20.1+1-post-Ubuntu-0ubuntu120.04, mixed mode, sharing)
root@ip-172-31-23-18:/home/ubuntu# docker -v
Docker version 24.0.5, build 24.0.5-0ubuntu1~20.04.1
root@ip-172-31-23-18:/home/ubuntu#
```

****

Configuring Jenkins:

- Copy and paste the public ip address of the Jenkins master instance and paste it on the browser:

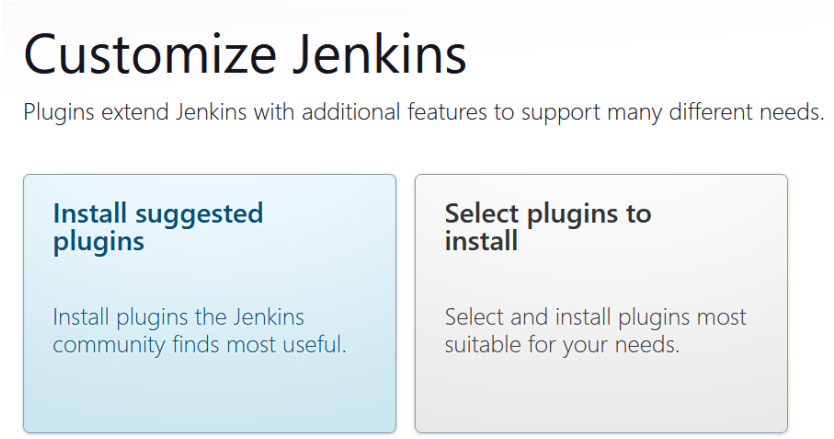


- Now we need to the password for further continuing process, copy and paste the directory given by Jenkins, and paste it on the server, like **sudo cat /var/lib/jenkins/secrets/initialAdminPassword**

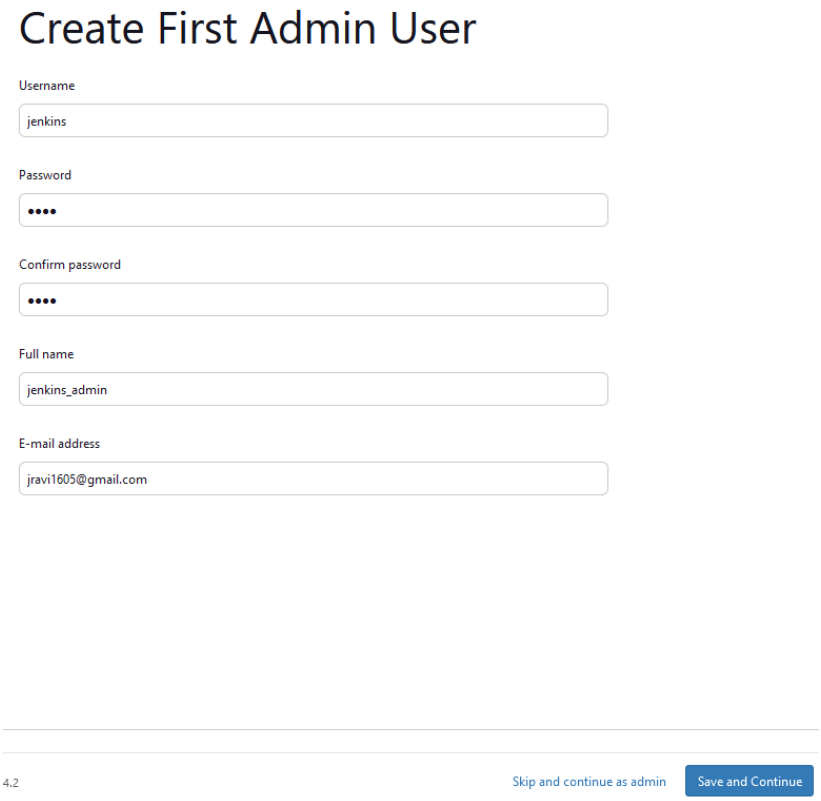
```
root@ip-172-31-28-44:/home/ubuntu# sudo cat /var/lib/jenkins/secrets/initialAdminPassword
3cfa67514724489a89605ebee6304148
root@ip-172-31-28-44:/home/ubuntu#
```

Copy and paste it and click continue.

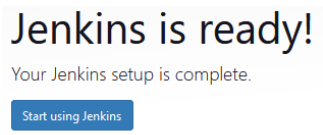
- After that we need to select the installation methods of plugins, here I am selecting **install suggested plugins methods**, which will install the default plugins.



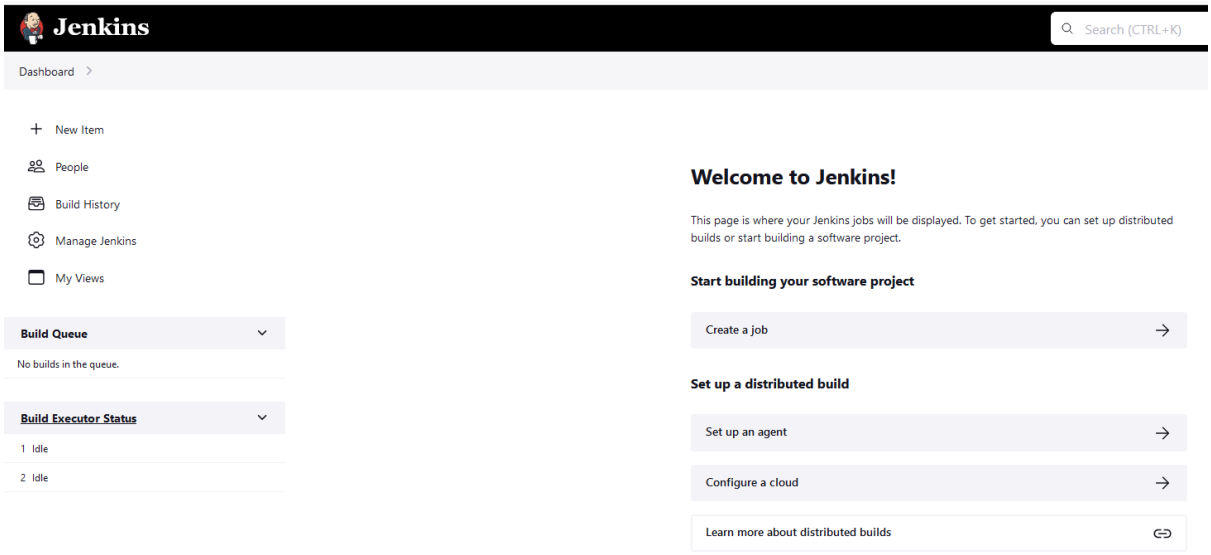
- Entering the username details and password, after **that click save and continue.**



- After that we could able to jenkins is ready, click **start using jenkins option:**

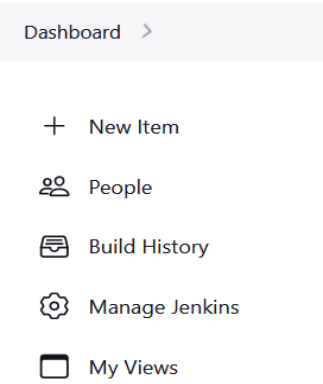


We could able to see the jenkins dashboard:



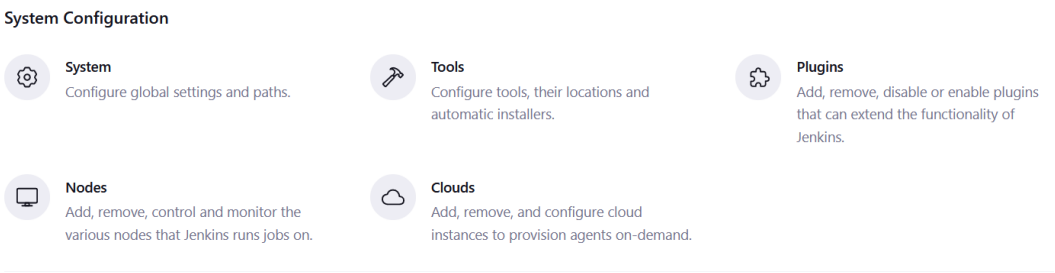
****

Jenkins master_slave configuration: (connecting testing and prod server)

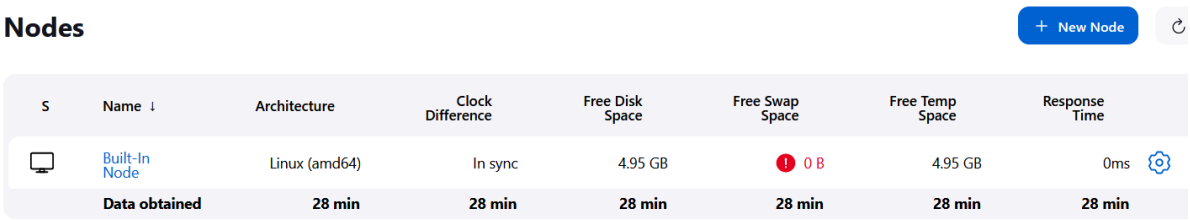


- On the left side we could able to see **manage Jenkins** option: click that one

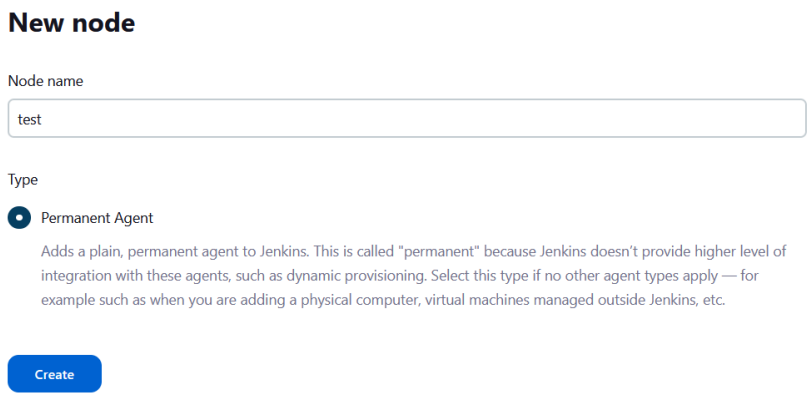
- Then on the manage Jenkins, under **system configuration**, select **nodes**

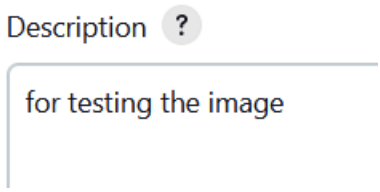


- We could able to see the node page, on that on the right side we could able to see **new node** option click that:

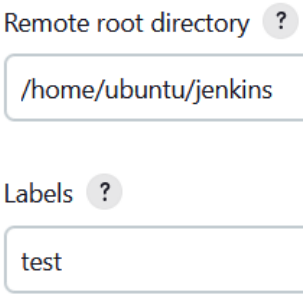


- Then enter the new name for the node, and select **permanent agent,** select create:

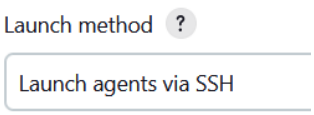




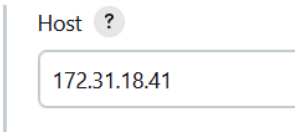
- Entering the description according to the task:



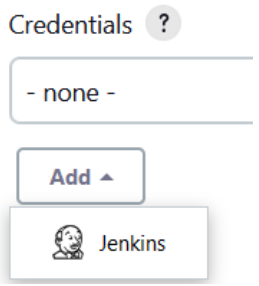
- Entering remote directory location where we can see the task details or job details:



- under launch method: **selecting the option launch agents via ssh:**



entering **host ip address** for executing the task there:

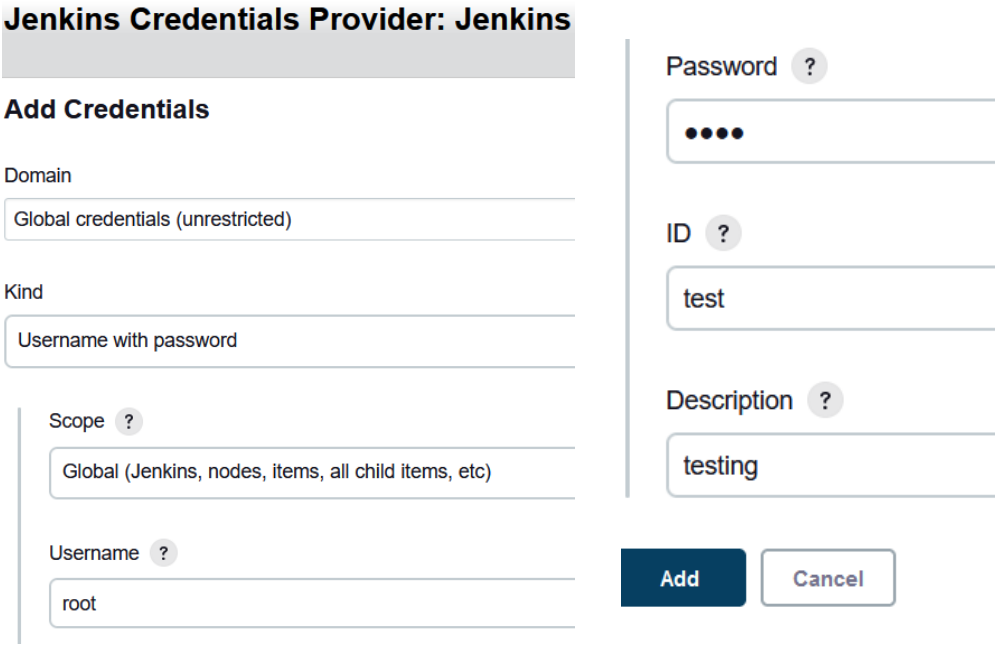


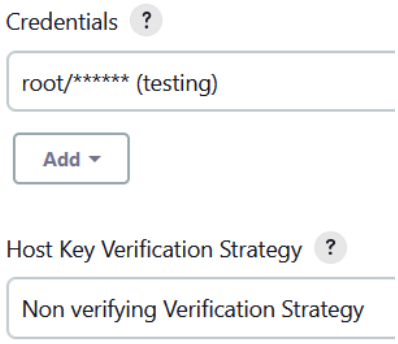
under credentials I **am adding credentials of test server:**

under kind select username and password:

entering **username and password of that user:**

click add:

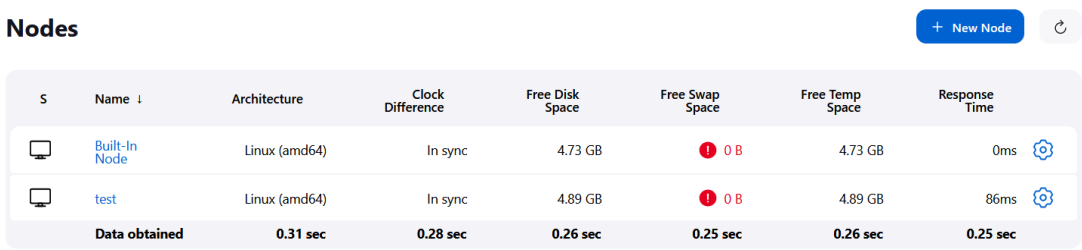




- Then adding the credentials just we created.

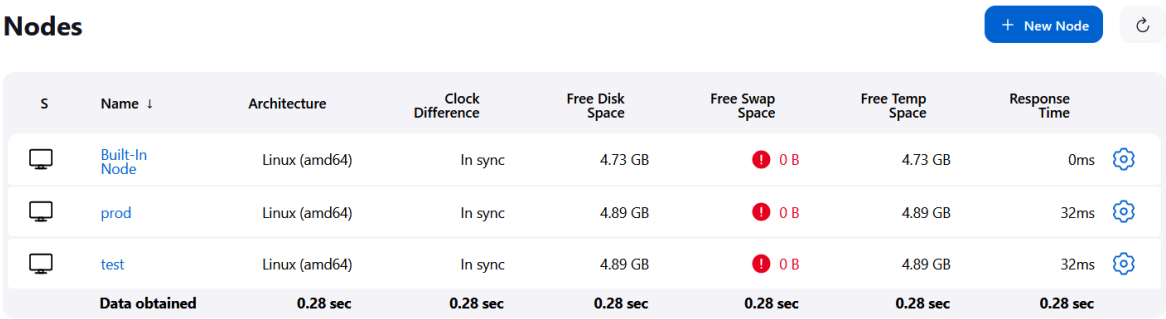
Then save it.

- Then under nodes we could able to see the new node has been created successfully. We could see that the **new node** should be in sync that means the node has been connected to Jenkins master.



S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	4.73 GB	0 B	4.73 GB	0ms
	test	Linux (amd64)	In sync	4.89 GB	0 B	4.89 GB	86ms
	Data obtained	**0.31 sec**	**0.28 sec**	**0.26 sec**	**0.25 sec**	**0.26 sec**	**0.25 sec**

Like this we need to add prod server to this Jenkins master:



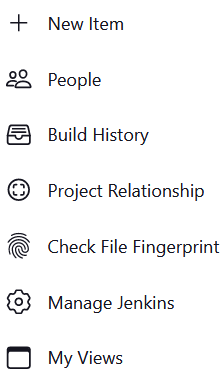
S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	4.73 GB	0 B	4.73 GB	0ms
	prod	Linux (amd64)	In sync	4.89 GB	0 B	4.89 GB	32ms
	test	Linux (amd64)	In sync	4.89 GB	0 B	4.89 GB	32ms
	Data obtained	**0.28 sec**	**0.28 sec**	**0.28 sec**	**0.28 sec**	**0.28 sec**	**0.28 sec**

Both the nodes have been created and added successfully.

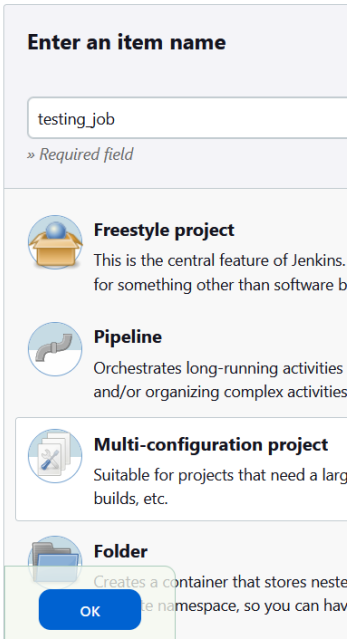
****

Creating Jenkins pipeline jobs:

1st pipeline job: [for testing docker image on testing server]



- click new item, for creating job:



- naming the job, and selecting the **pipeline type:**

General

- naming the description according to the project:

Description

for testing docker image

Pipeline

Definition

Pipeline script

Script

try sample Pipeline...

```
stages {
    stage('Git cloning') {
        steps {
            git branch: 'master',
            url: 'https://github.com/Ravivarman16/guvi_task.git'
        }
    }

    stage ('Building a dockerimage') {
        steps {
            sh 'docker build -t ravivarman46/guviimage .'
            sh 'docker run -d --name container1 -p 8080:80 ravivarman46/guviimage'
        }
    }
}
}
```

Use Groovy Sandbox

Pipeline Syntax

Save Apply

- Then under pipeline entering the script for execution, click save and apply:

The job has been created:

All +

S	W	Name ↓	Last Success	Last Failure	Last Duration
		testing_job	N/A	N/A	N/A

Icon: S M L Icon legend Atom feed for all Atom feed for failures Atom feed for just latest builds

Now we need to execute 1st pipeline job for that we need to create dockerfile and index.html file

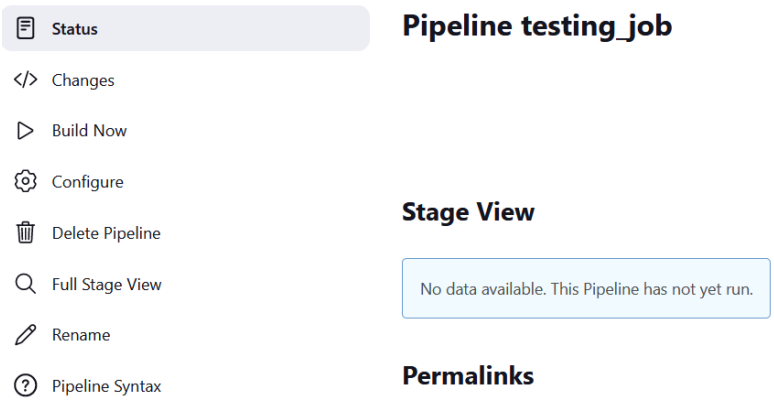
```
root@ip-172-31-17-250:/home/ubuntu/git# git init
Initialized empty Git repository in /home/ubuntu/git/.git/
root@ip-172-31-17-250:/home/ubuntu/git# ls
root@ip-172-31-17-250:/home/ubuntu/git# vi dockerfile
root@ip-172-31-17-250:/home/ubuntu/git# vi index.html
root@ip-172-31-17-250:/home/ubuntu/git# git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        dockerfile
        index.html

nothing added to commit but untracked files present (use "git add" to track)
root@ip-172-31-17-250:/home/ubuntu/git# git add dockerfile index.html
root@ip-172-31-17-250:/home/ubuntu/git# git commit -m "image files"
[master (root-commit) d1c029d] image files
 Committer: root <root@ip-172-31-17-250.us-west-2.compute.internal>
```

initializing the git by using **git init** command.

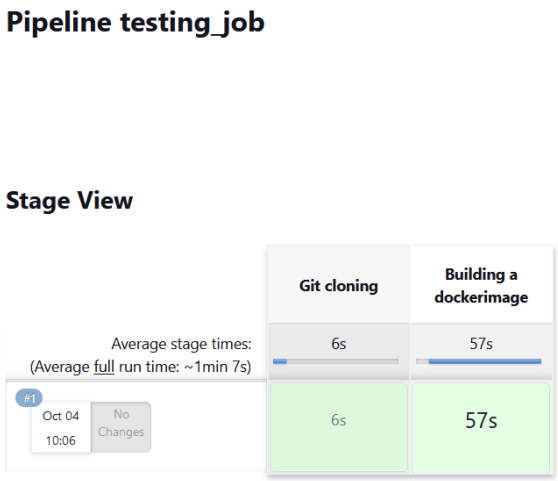
Adding and commiting the files.

```
root@ip-172-31-17-250:/home/ubuntu/git# git remote add origin https://github.com/Ravivarman16/guvi_task.git
root@ip-172-31-17-250:/home/ubuntu/git# git push origin master
Username for 'https://github.com': Ravivarman16
Password for 'https://Ravivarman16@github.com':
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 436 bytes | 436.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.com/Ravivarman16/guvi_task/pull/new/master
remote:
To https://github.com/Ravivarman16/guvi_task.git
 * [new branch]      master -> master
root@ip-172-31-17-250:/home/ubuntu/git#
```

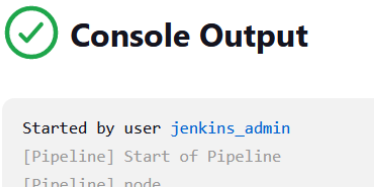
Pushing it to the github:



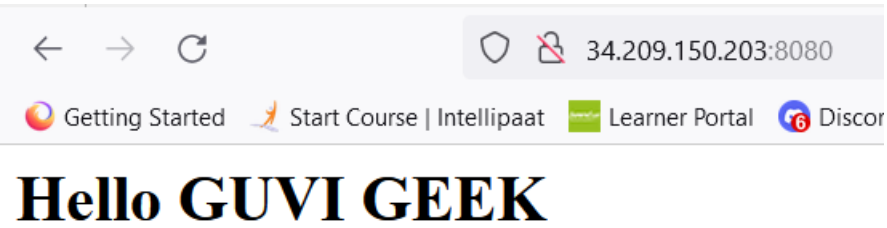
now click **build now** on the job which is available on left side:



The job has been executed successfully.



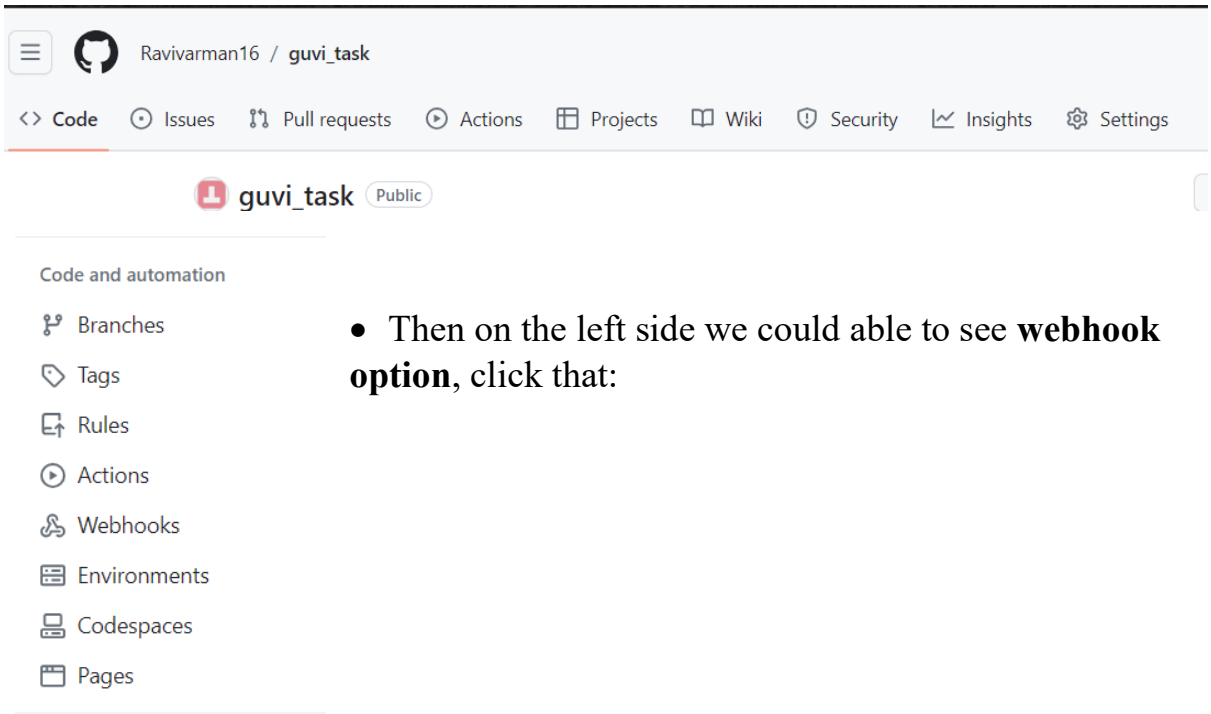
For the output, copy and paste the public ip address of test server on the browser along with the port number 8080 which were the container is running:



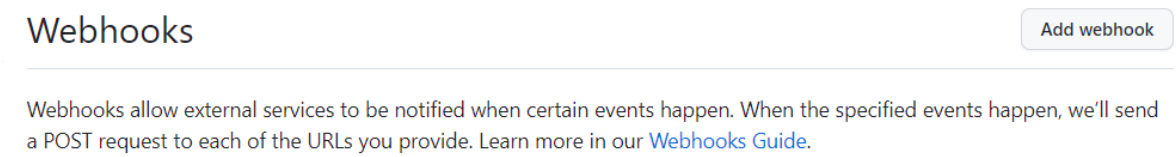
What we done is manual, but in the real time it should be automated once the files and images are pushed to the github, the job should be executed automatically.

For that we need to create **GitHub hook trigger:**

- Under GitHub repository select settings:

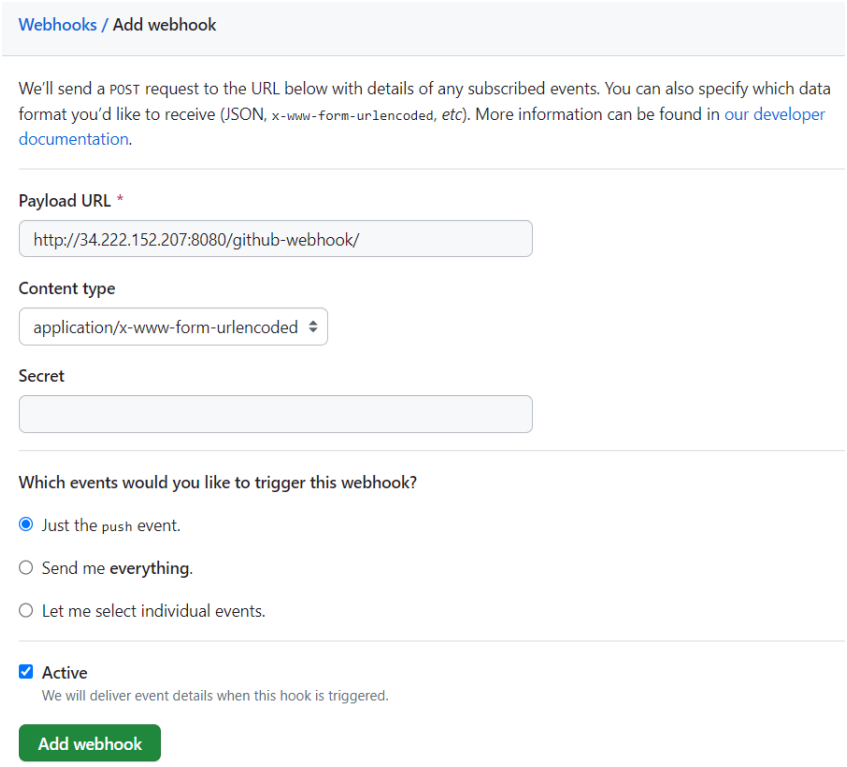


- Then on the left side we could able to see **webhook option**, click that:



Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.

- Then click add webhook:

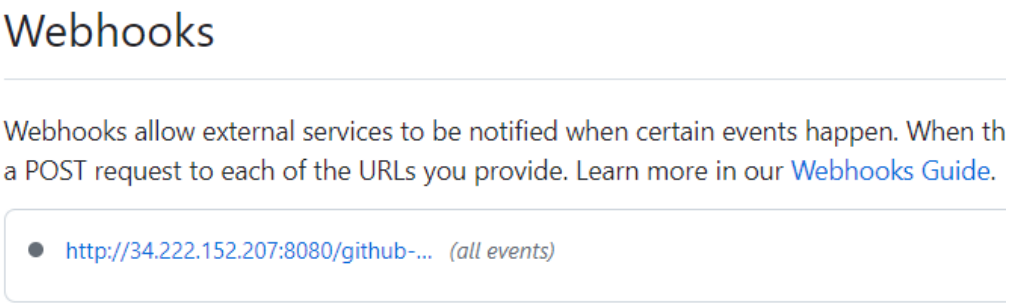


- **under payload URL** enter the **Jenkins URL** along with **github-webhook trigger option:**

- Under the secret enter the secret of your github id: for security purpose.

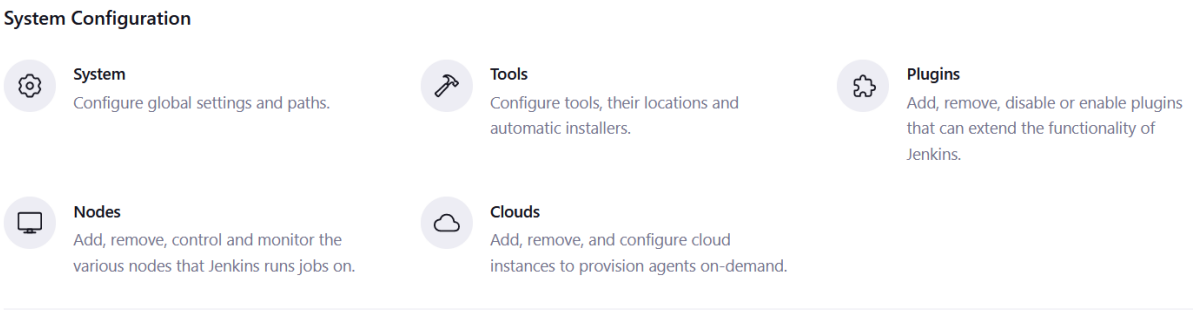
- Select the events according to your choice.

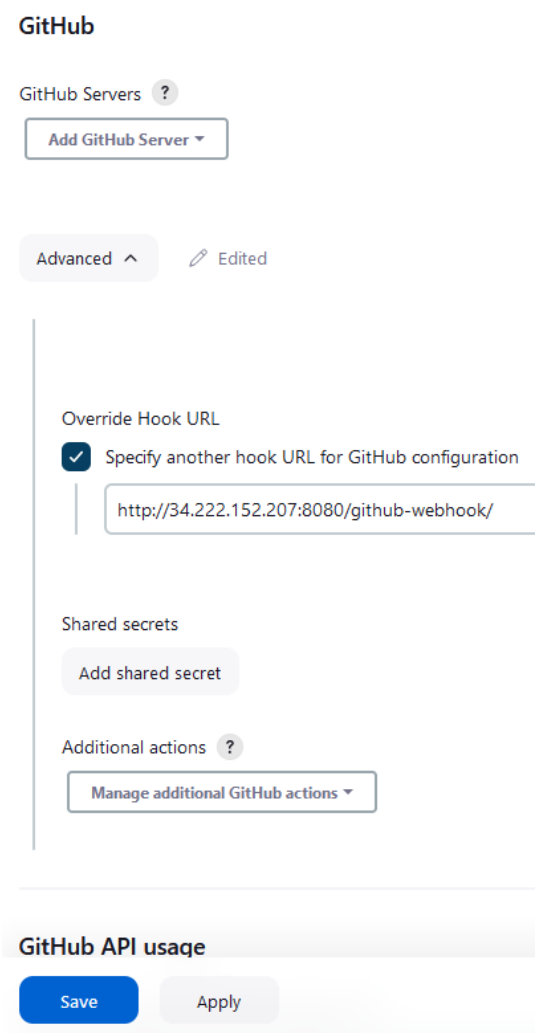
Click add webhook at last:



Webhook has been created successfully on **github side:**

Now we need to setup on Jenkins side also**: under manage Jenkins, system configuration click system:**





under github select the **advanced** option:

select the override hook URL option:

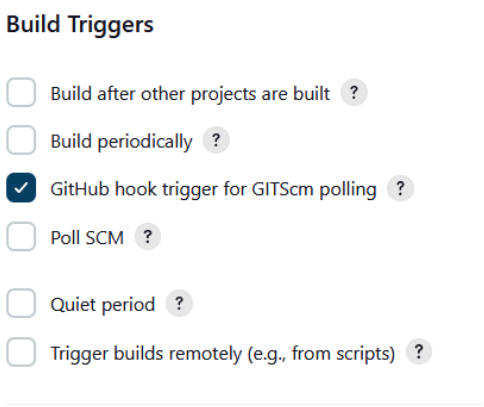
check the URL which we enter on **github webhook payload:**

click save and apply:

github webhook trigger has been setuped successfully.

Check the webhook trigger by **making changes in index.html,** then adding and commiting.

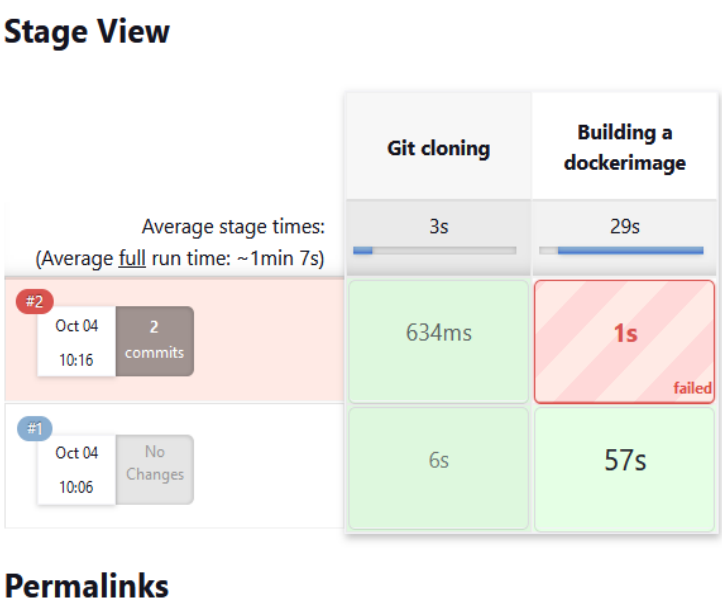
```
root@ip-172-31-17-250:/home/ubuntu/git# vi index.html
root@ip-172-31-17-250:/home/ubuntu/git# git add index.html
root@ip-172-31-17-250:/home/ubuntu/git# git commit -m "updated index.html"
[master 9f1d4ef] updated index.html
 Committer: root <root@ip-172-31-17-250.us-west-2.compute.internal>
```



under the pipeline job edit the configurations, under **build triggers select github hook trigger option:**

then **push the content into the GitHub**

```
root@ip-172-31-17-250:/home/ubuntu/git# git push origin master
Username for 'https://github.com': Ravivarman16
Password for 'https://Ravivarman16@github.com':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 330 bytes | 330.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/Ravivarman16/guvi_task.git
   d1c029d..9f1d4ef  master -> master
root@ip-172-31-17-250:/home/ubuntu/git#
```



Then we could able to see the pipeline triggered automatically. But the job is failed this is because already a container is running on that image name with the same port number for that we need to make a small change in the pipeline script:

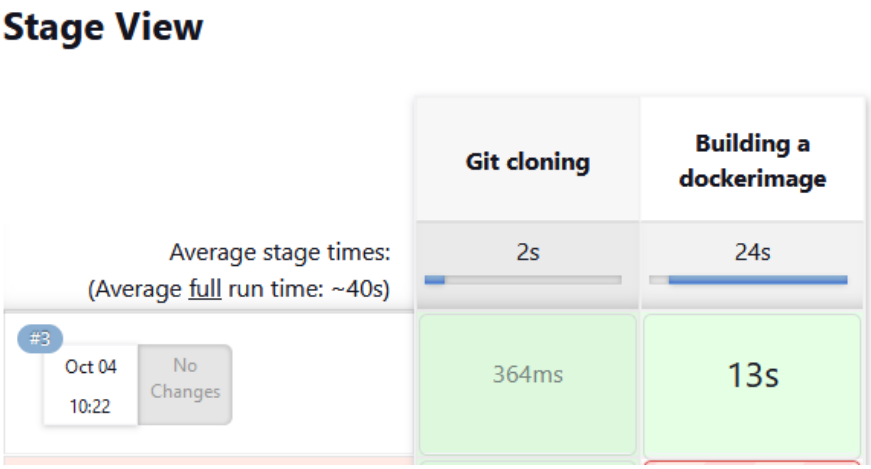
In the 1st Jenkins job pipeline script: under the stage building dockerimage

Adding the line: **docker rm -f $(docker ps -aq)** at starting itself, whenever the job executes this will delete all the running container whichever is running. So again, the image will be built and container will be created.

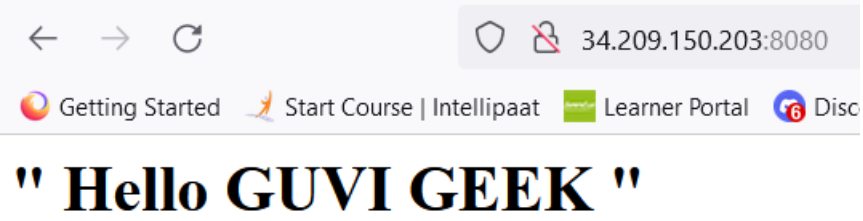
```
stage ('Building a dockerimage') {
            steps {
                sh 'docker rm -f $(docker ps -aq)'
                sh 'docker build -t ravivarman46/guviimage
.'
                sh 'docker run -d --name container1 -p
8080:80 ravivarman46/guviimage'
                sh 'docker push ravivarman46/guviimage'
            }
        }
```

- For pushing the docker image into Docker Hub, I am adding my credentials in testing server for pushing the image:

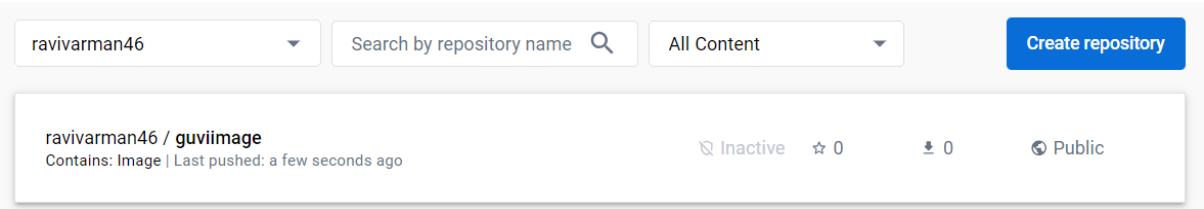
```
root@ip-172-31-18-41:/home/ubuntu# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't ha
to create one.
Username: ravivarman46
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
root@ip-172-31-18-41:/home/ubuntu#
```



making changes in the file and pushing it to the github. The job is executed automatically. But this time the job executed successfully.

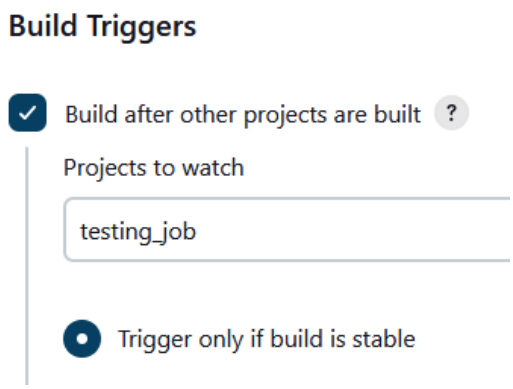


the output where we can see the difference the previous image and this image:



The image has been pushed to the Github:

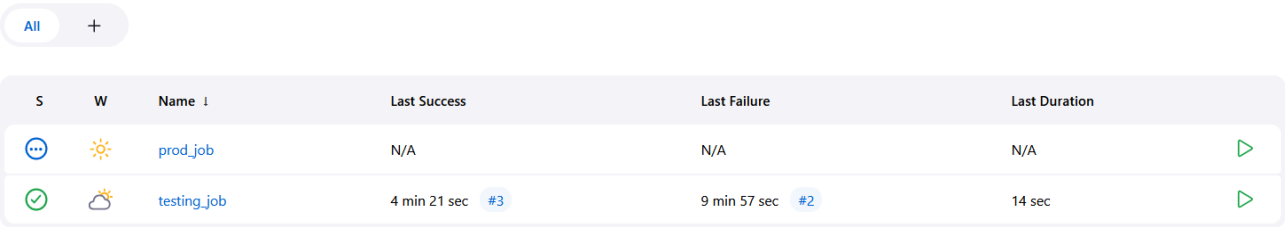
2nd Pipeline job: [the image has been built successfully in the previous job; on this job the container will run directly from the build image]



under build triggers, selecting the option, **build after other projects are built.**

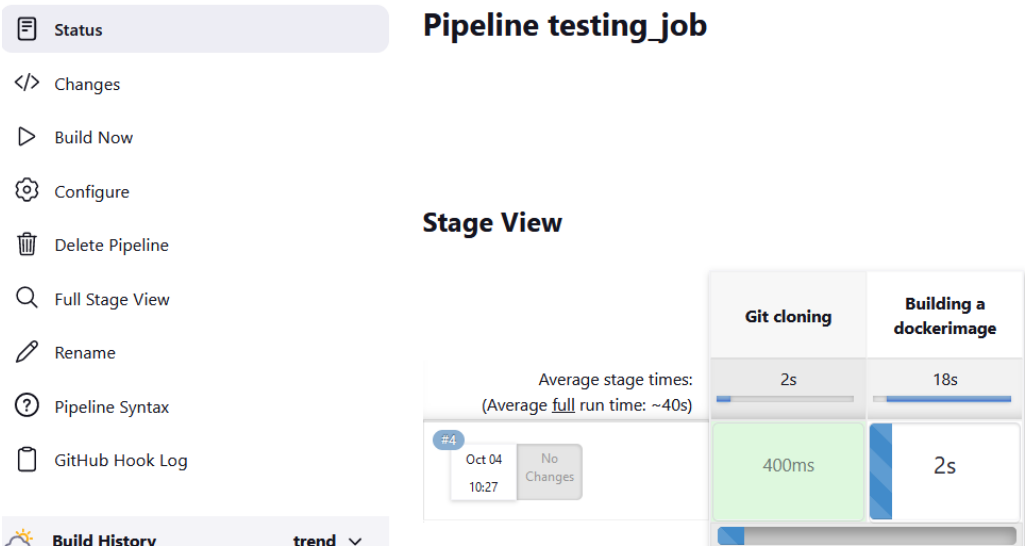
Selecting the first pipeline job, selecting the option trigger the 2nd job only if the first job is **build is stable.**

Entering the pipeline script and saving the job:

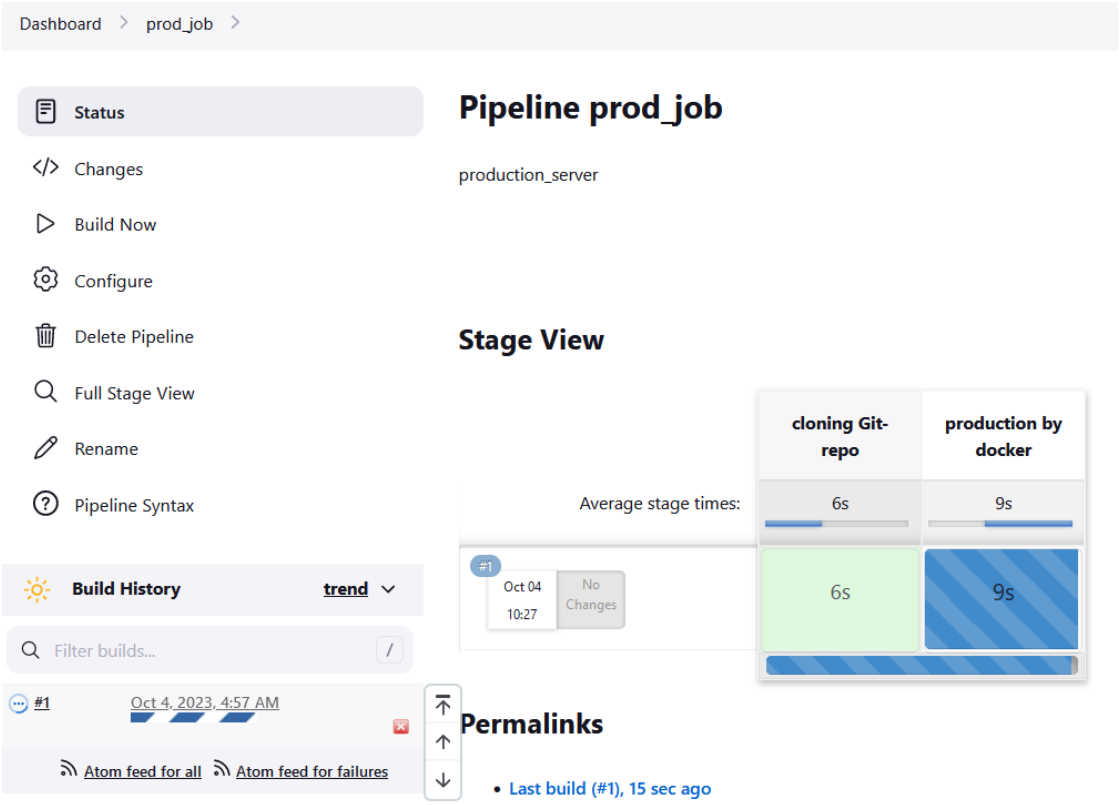


Testing the pipeline:

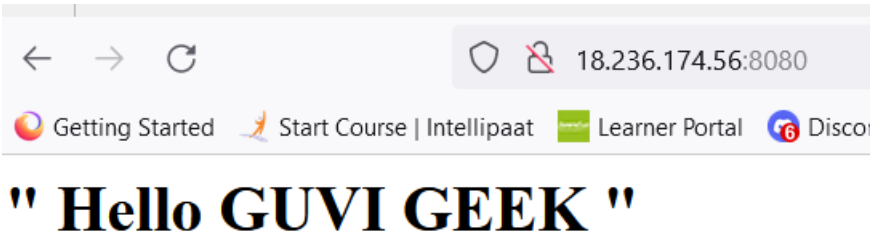
- I am manually running the first job:



The second job is triggered automatically once the first job is executed successfully.



The output of the 2nd job:

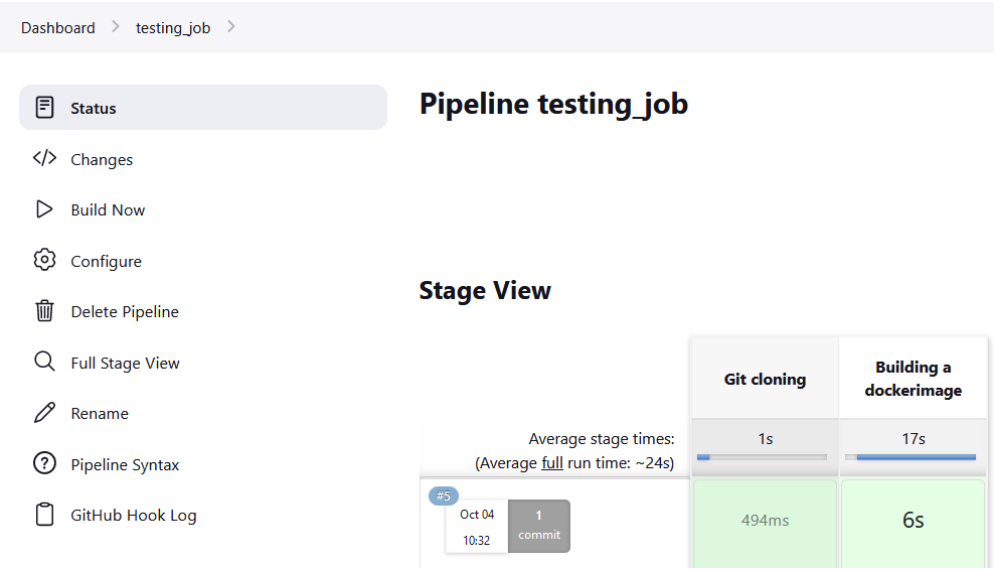


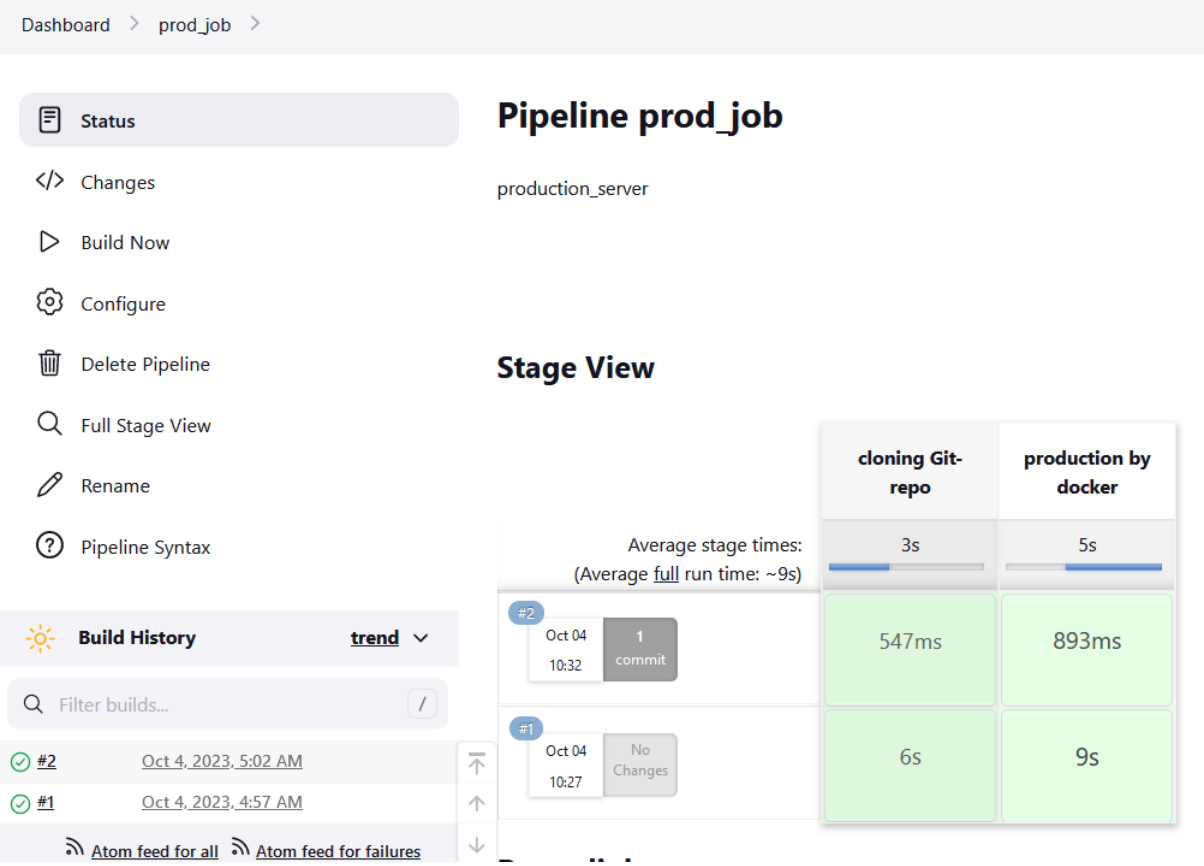
" Hello GUVI GEEK "

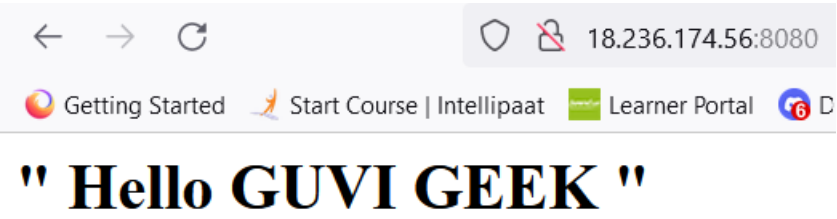
Testing entire pipeline:

Just creating empty file and pushing it to the github:

```
root@ip-172-31-17-250:/home/ubuntu/git# touch demofile
root@ip-172-31-17-250:/home/ubuntu/git# git add d
fatal: pathspec 'd' did not match any files
root@ip-172-31-17-250:/home/ubuntu/git# git add demofile
root@ip-172-31-17-250:/home/ubuntu/git# git commit -m "just for testing"
[master 9963baa] just for testing
 Committer: root <root@ip-172-31-17-250.us-west-2.compute.internal>
```







" Hello GUVI GEEK "

The entire pipeline is working fine as excepted.

We can also the job details on command line also where we have given the remote location of nodes:

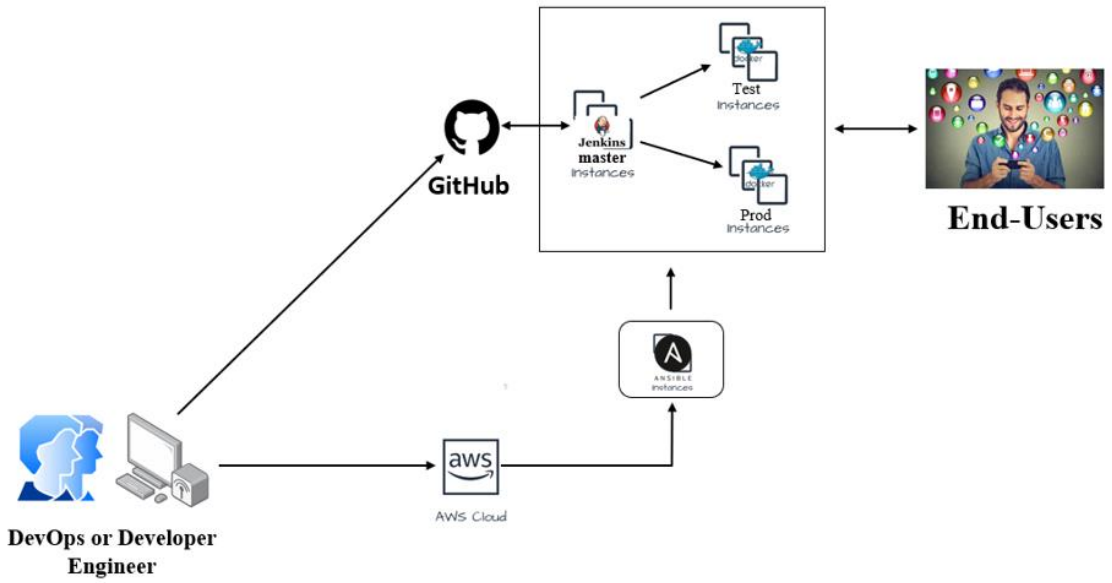
Test_server:

```
root@ip-172-31-18-41:/home/ubuntu# cd /home/ubuntu/jenkins/
root@ip-172-31-18-41:/home/ubuntu/jenkins# ls
caches  remoting  remoting.jar  workspace
root@ip-172-31-18-41:/home/ubuntu/jenkins# cd workspace/
root@ip-172-31-18-41:/home/ubuntu/jenkins/workspace# ls
testing_job  testing_job@tmp
root@ip-172-31-18-41:/home/ubuntu/jenkins/workspace# cd testing_job
root@ip-172-31-18-41:/home/ubuntu/jenkins/workspace/testing_job# ls
demofile  dockerfile  index.html
root@ip-172-31-18-41:/home/ubuntu/jenkins/workspace/testing_job#
```

Prod_server:

```
root@ip-172-31-23-18:/home/ubuntu# cd /home/ubuntu/jenkins/
root@ip-172-31-23-18:/home/ubuntu/jenkins# ls
caches  remoting  remoting.jar  workspace
root@ip-172-31-23-18:/home/ubuntu/jenkins# cd workspace/
root@ip-172-31-23-18:/home/ubuntu/jenkins/workspace# ls
prod_job  prod_job@tmp
```

THE ARCHITECTURE DIAGRAM OVERVIEW:



All the script files required to build the task has been upload on github repository

- **The repository link: https://github.com/Ravivarman16/guvi_task.git**
- **Docker Hub image: https://hub.docker.com/repository/docker/ravivarman46/guviimage/general**