# Capstone Project

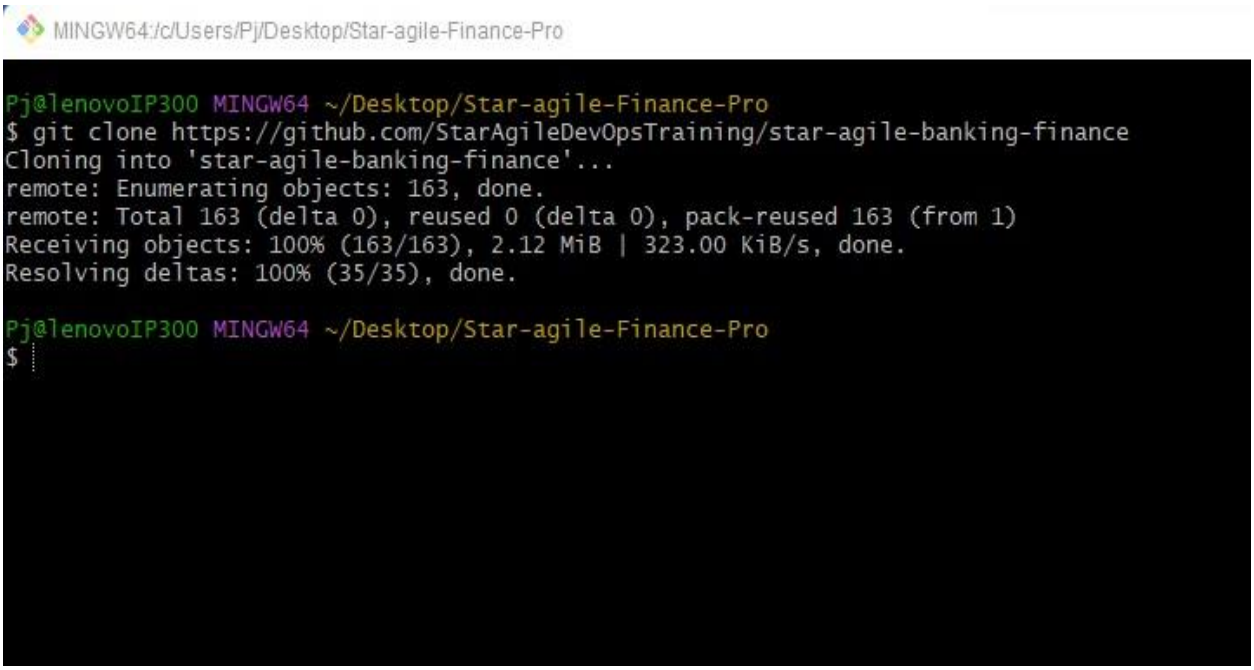
## Banking and Finance Domain Project

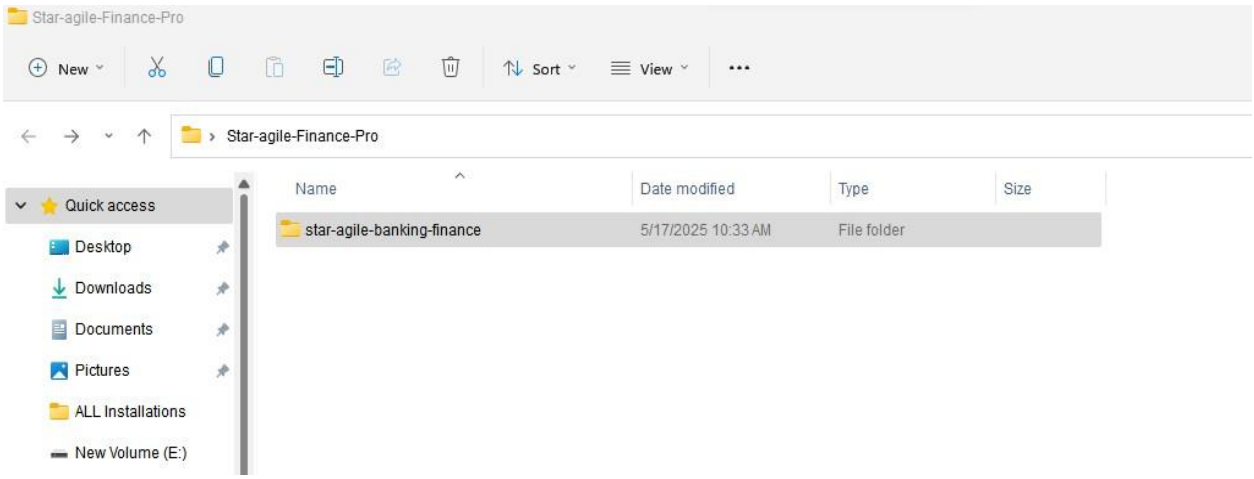
By:- Jonna Padmarao

Source code URL :- https://github.com/pj013525/star-agile-project-1.git

Step1:- On the desktop create a new folder (star-agile-finance-pro) and enter into that folder and open the git bash in that folder

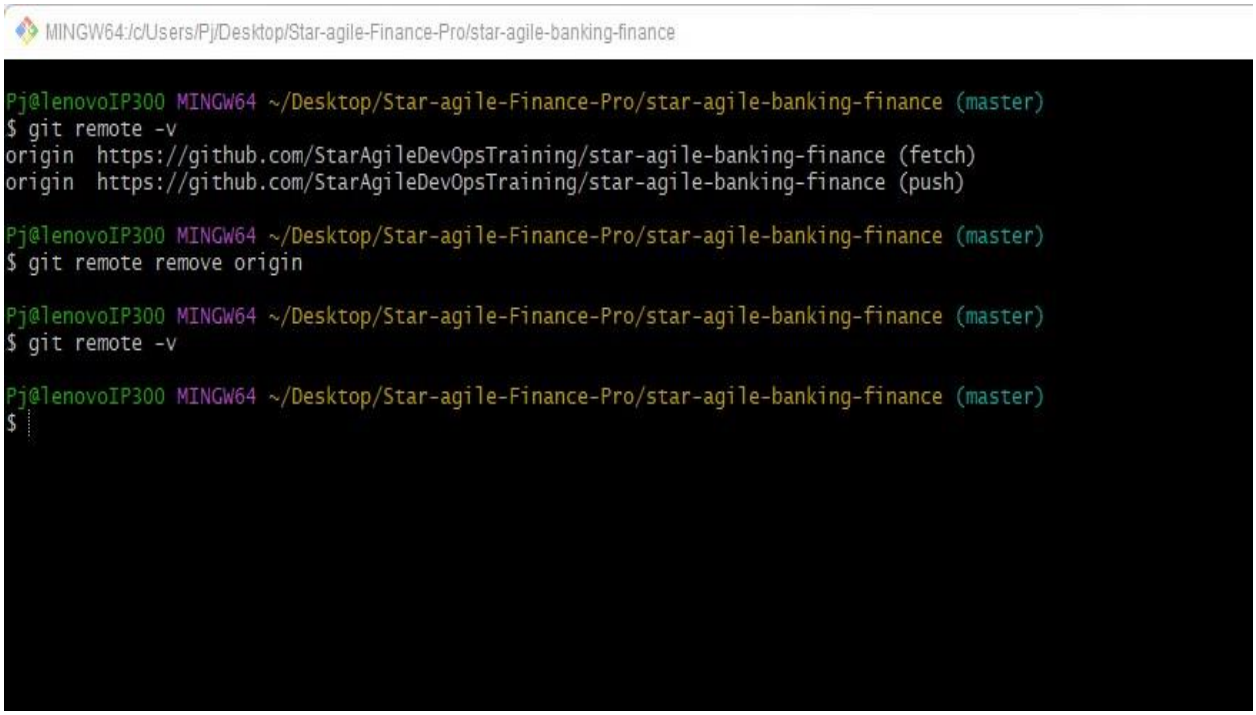
Step2:- Now give git clone https://github.com/StarAgileDevOpsTraining/star-agile-banking-finance to get the project code in to that folder



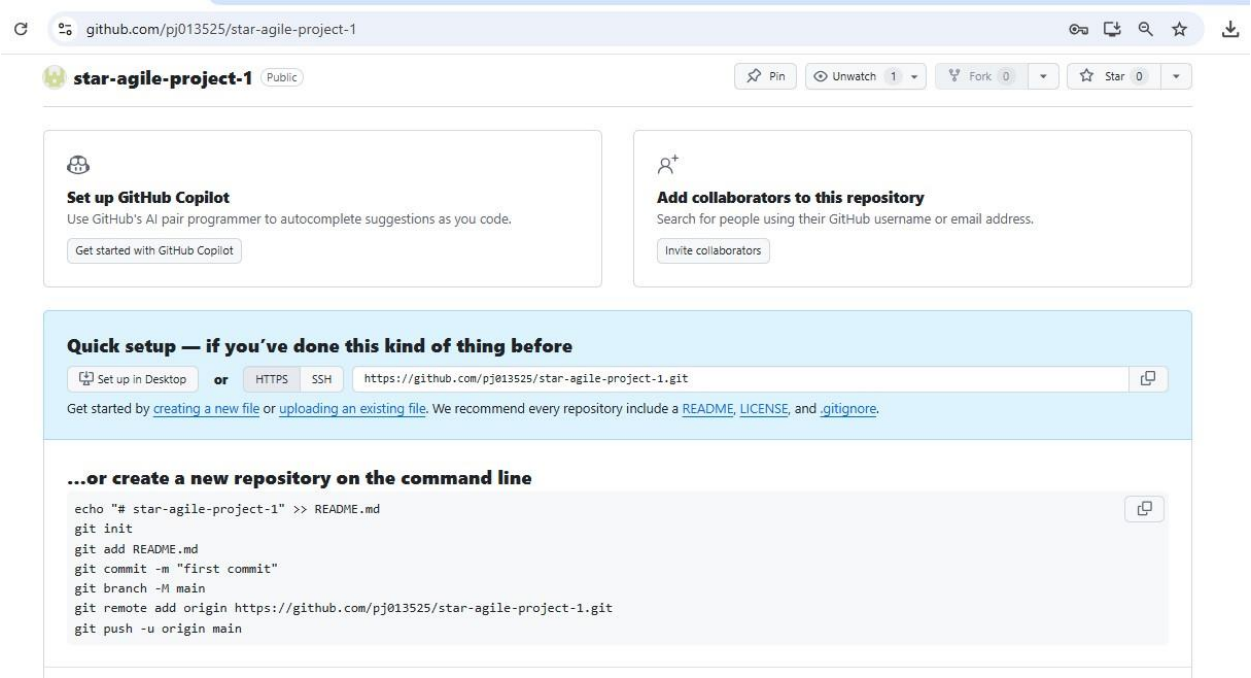



Step3:- Now go to the folder that we get from git clone and again open git bash there and check the origin and remove that origin  git remote -v ❼ To get origin list

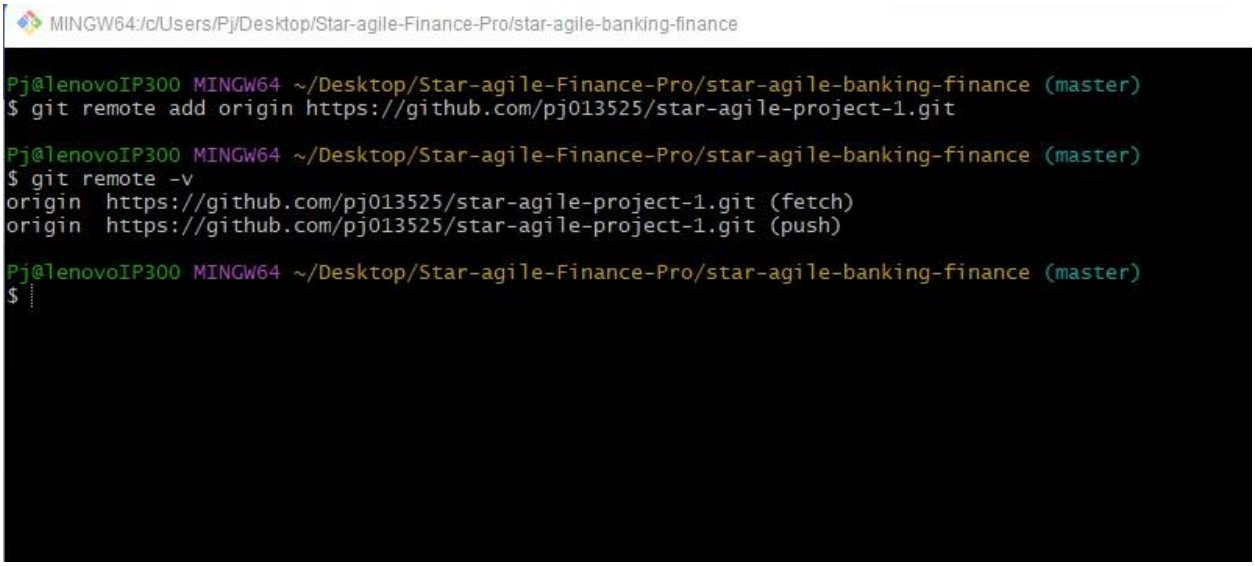
git remote remove origin ❼ to remove the origin



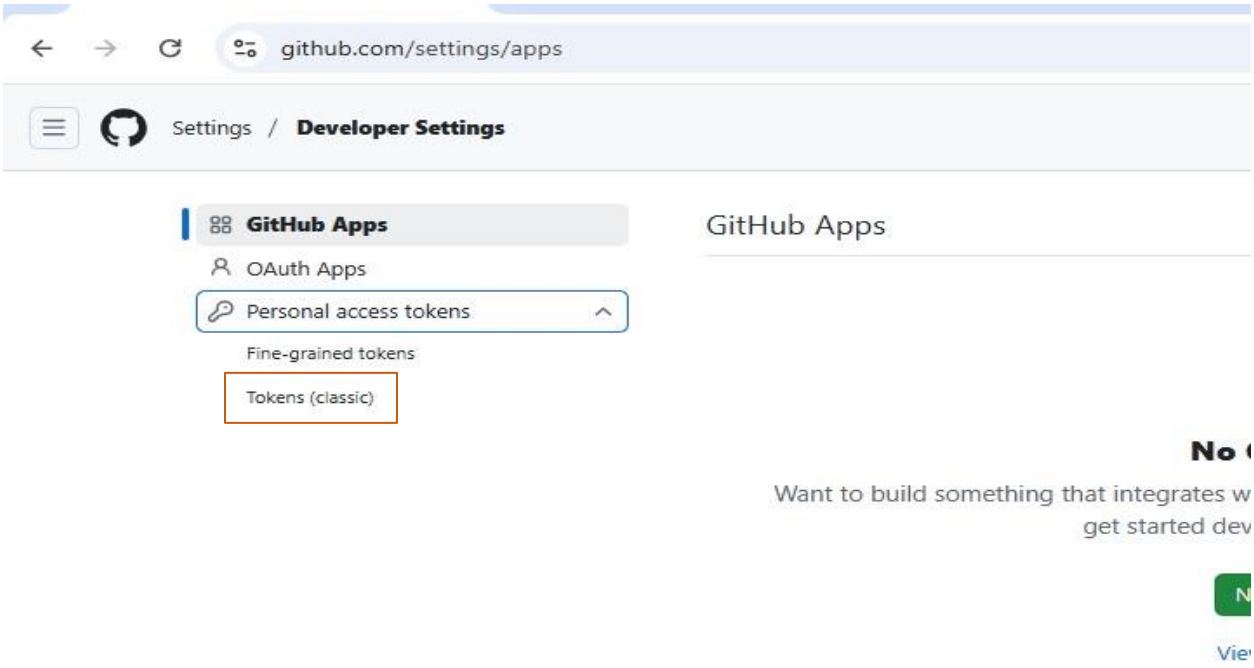
Step4:- Now go to github and create a new repo and copy the url in the gitbash



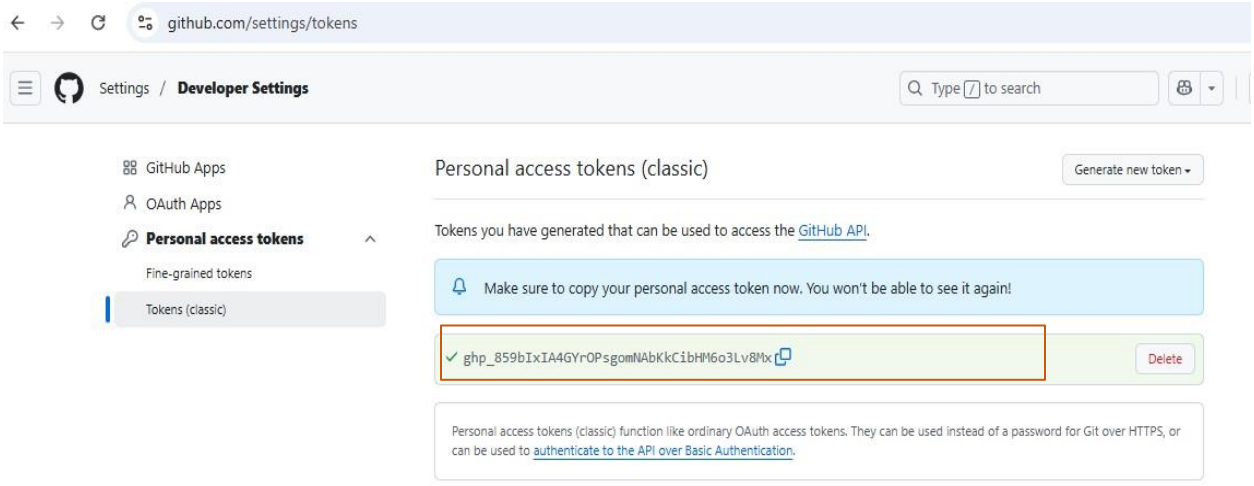

Step5:- Now again go to the gitbash and add this git repo url in the project by using git remote add <git-repo-url> and verify




Step6:- Now again go to github ❼ Profile setting ❼Developer settings ❼ Personal access token ❼ Tokens(classic) ❼Generate new token

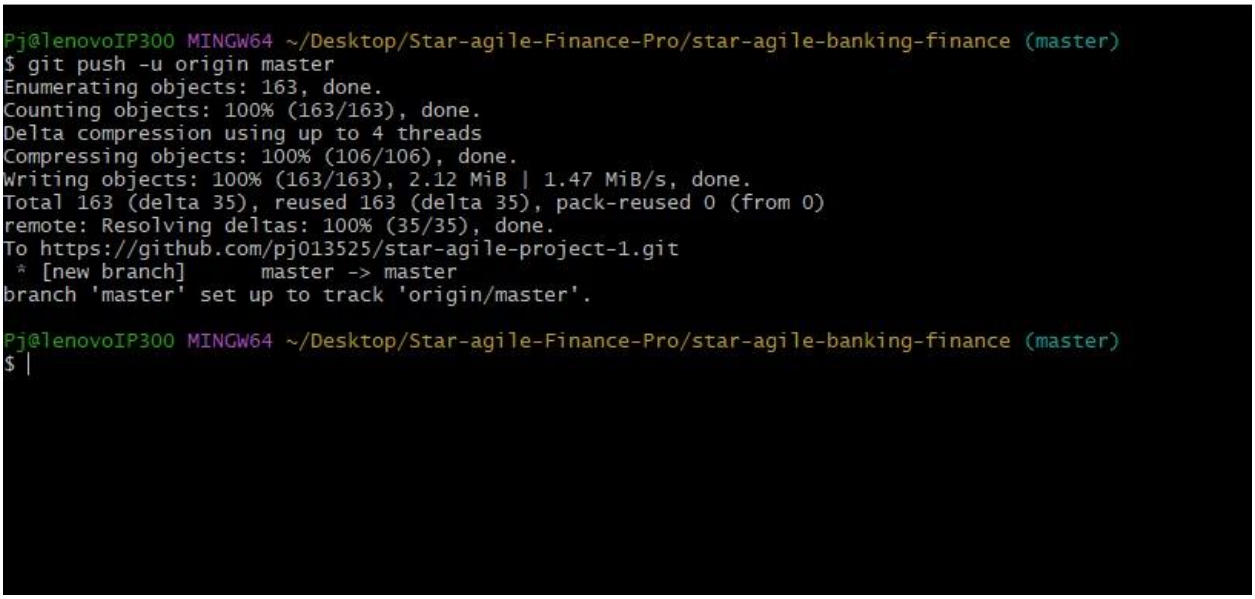



Step7:- Now a token will be generated , copy this token that generated since it is only available for one time only

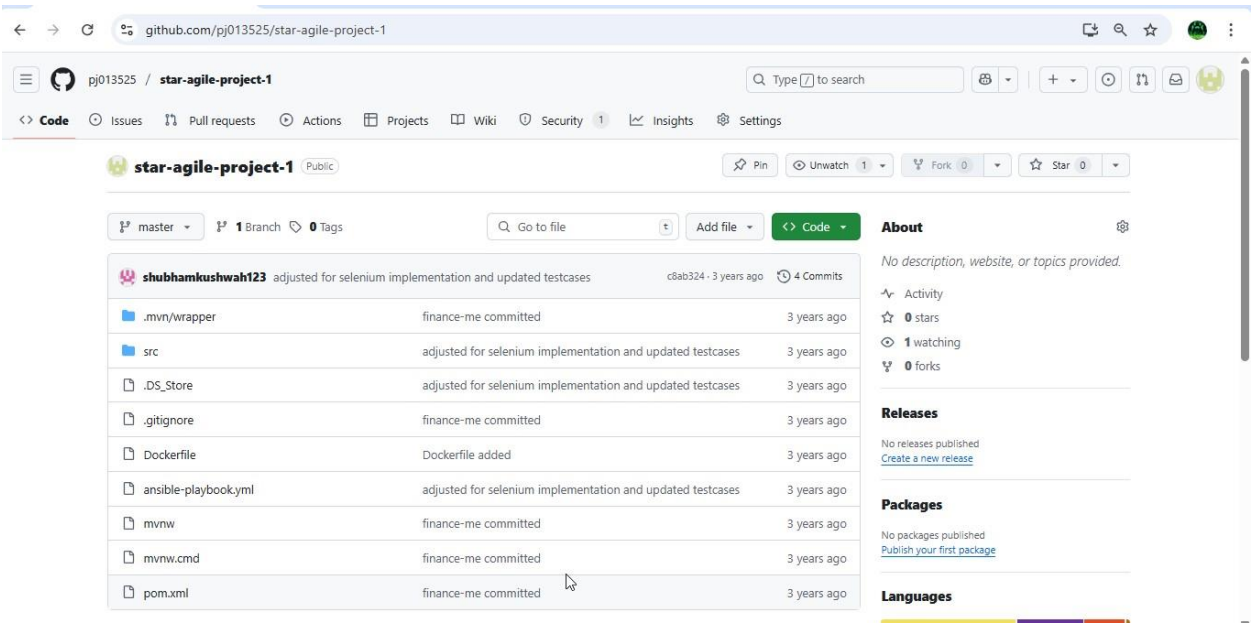



Step8:- Now give link this the remote repo with gitbash using this token git push -u origin master and paste the token the copied from the github and press sign in

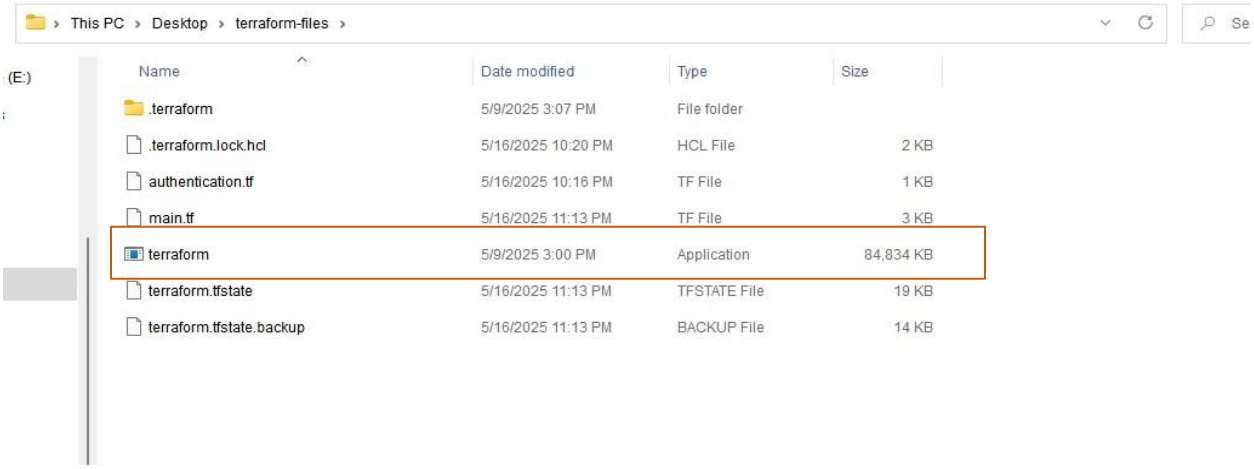

Step9:- Now the master branch will be set to our repo by default



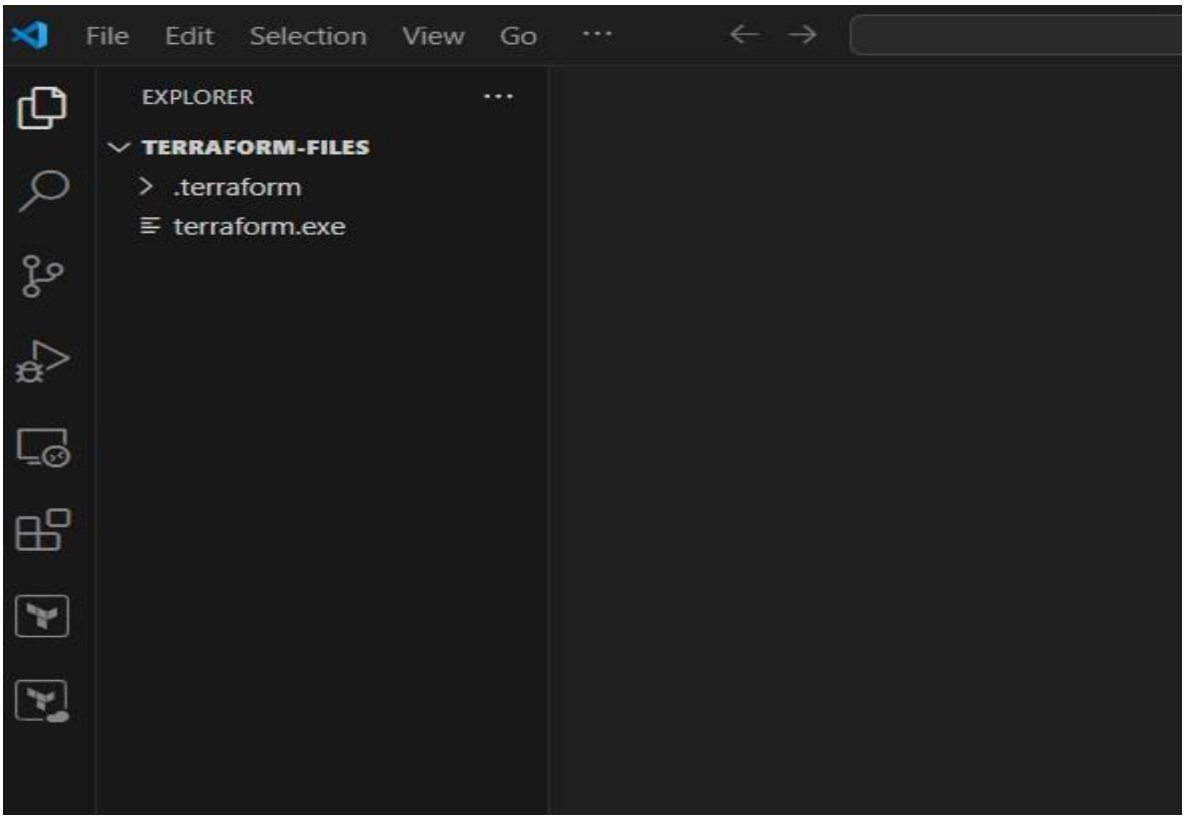

Step10:- Now go to the github repo and you will see the source code in that repo.



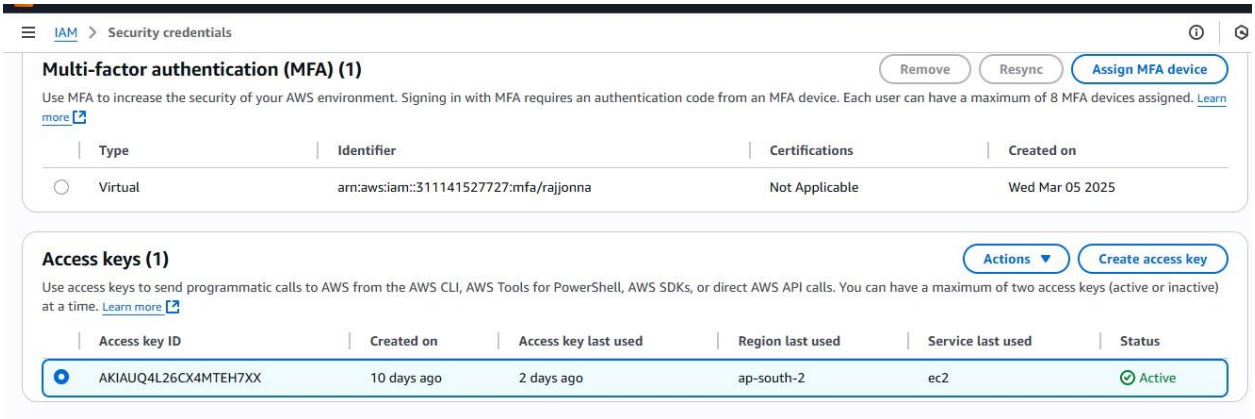

Step11:- Now create an instance using terraform as Iac , and for that create a folder on desktop and go to browser download terraform for windows then a terraform application will be generated , now copy this application in to that folder and save



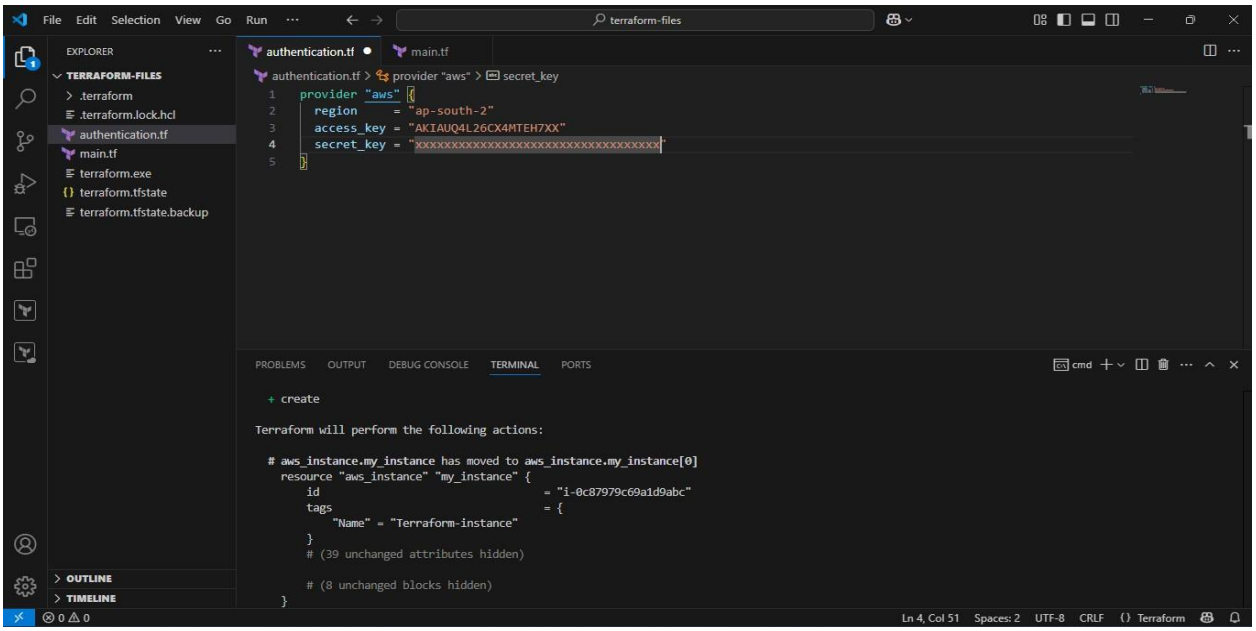

Step12:- open visual studio code and go to terraform folder.



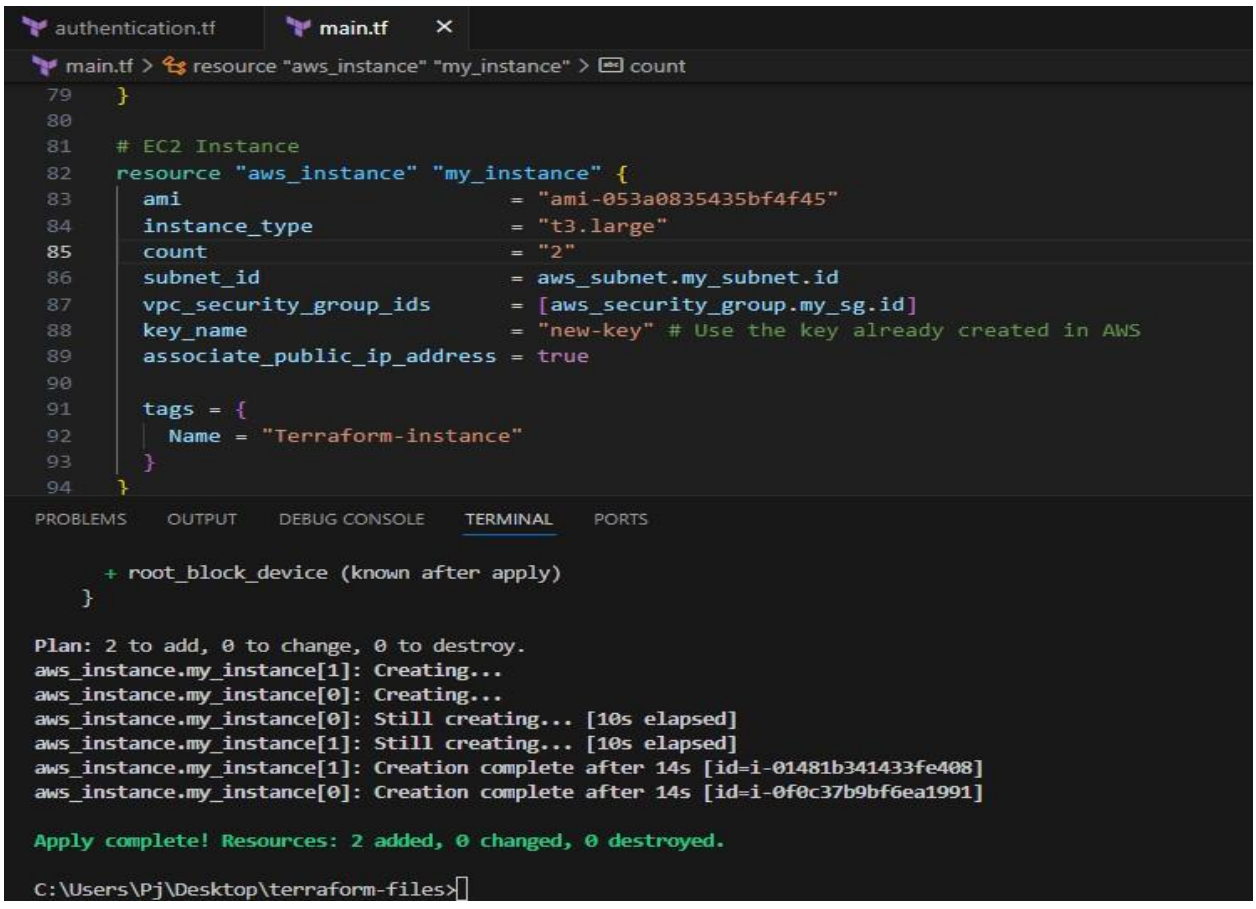

Step13:- Now create a file authentication.tf and give the provider and for that select the region in which you want to launch the server and go to aws account and go to profile ❼ credentials and go to access keys



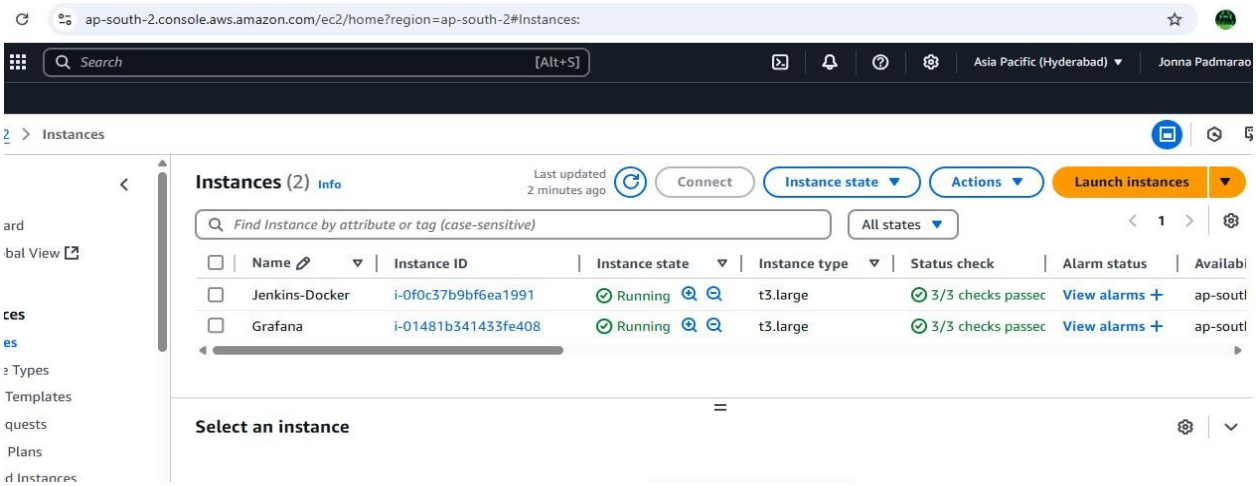

Step14:- now copy this access key details and paste it in this authentication.tf file and initialize it



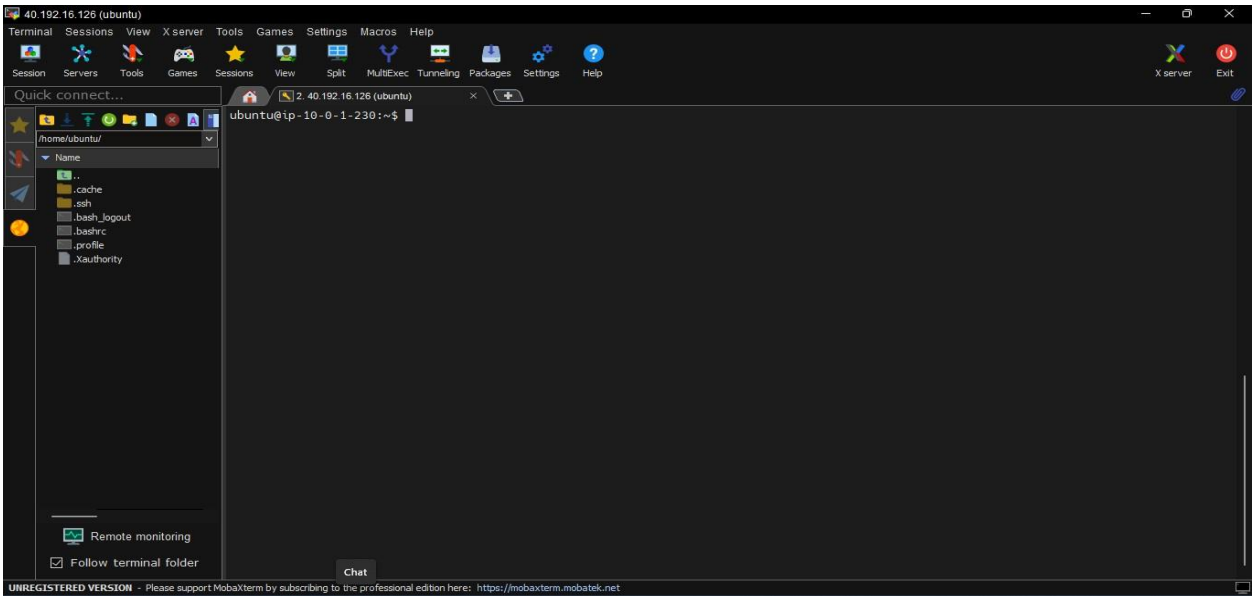

Step15:- After it is successful now create a new file main.tf and give resources details to create instance



Step16:- After it is successful go and check the aws console and rename them as Jenkins-docker and Grafana instances

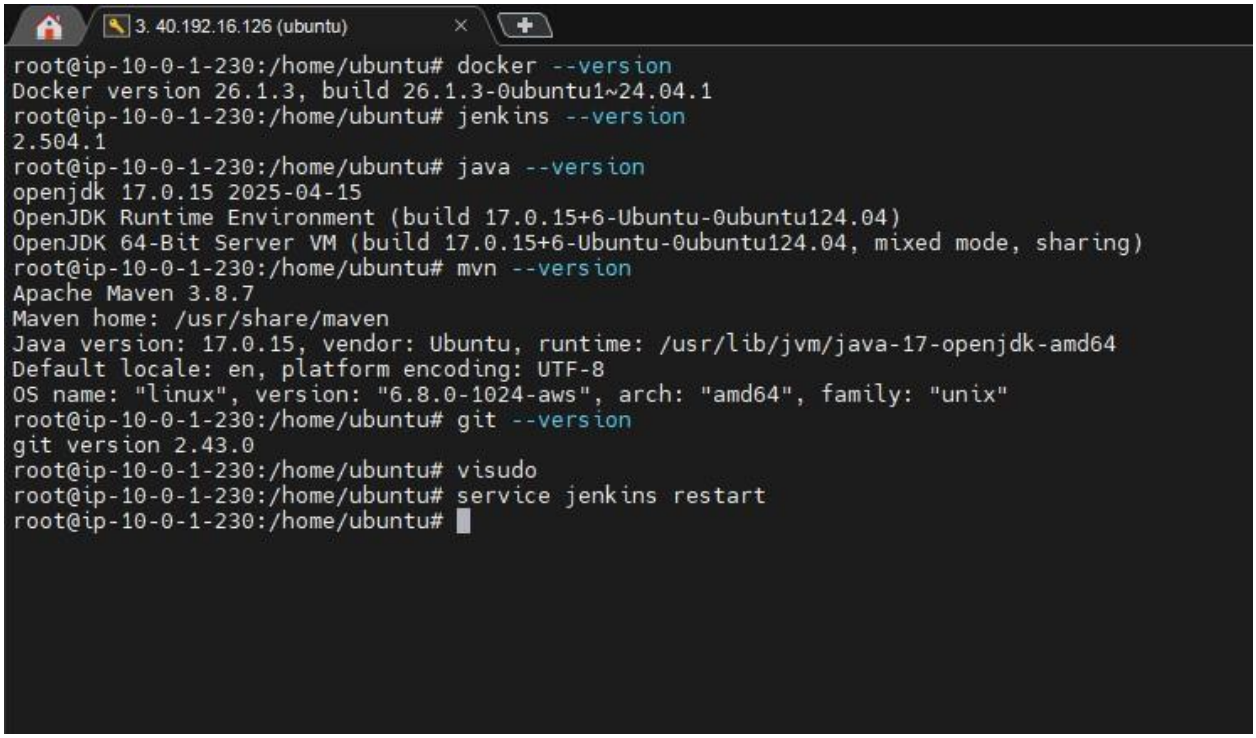



Step17:- Now connect to Jenkins-Docker server using Mobaxterm agent and launch an instance

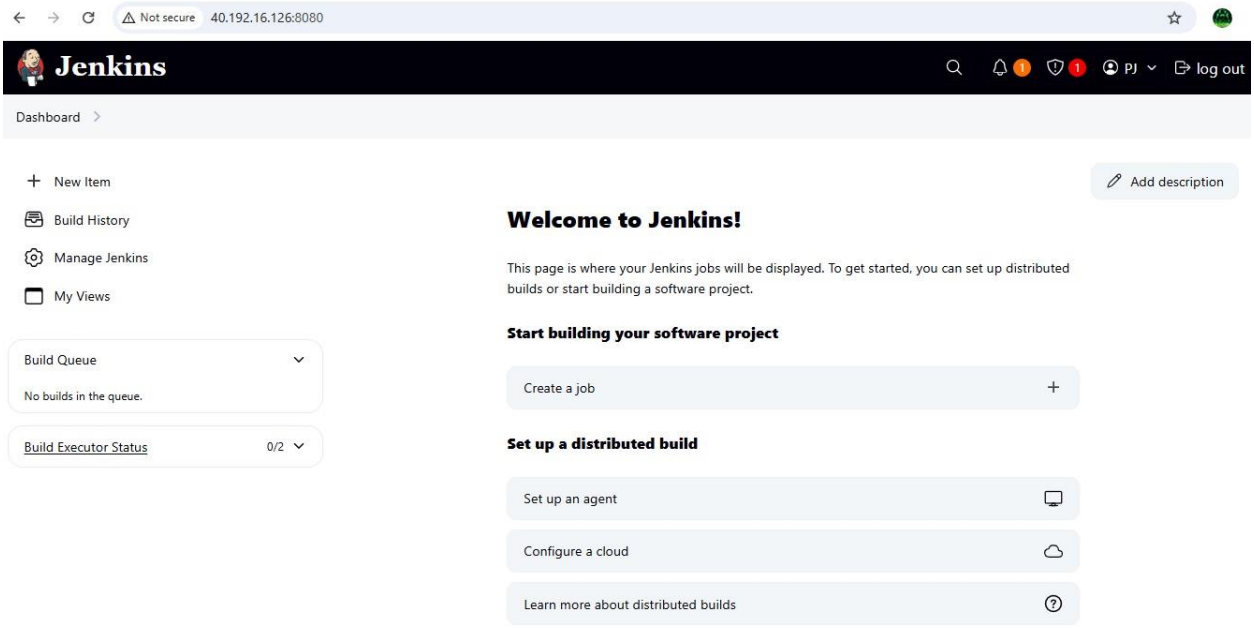



Step18:- Now install java and Jenkins and docker in this is sever and enable All traffic in the security group of this server and add Jenkins group to docker and give root permissions to the Jenkins user in the sudoers file as under root give

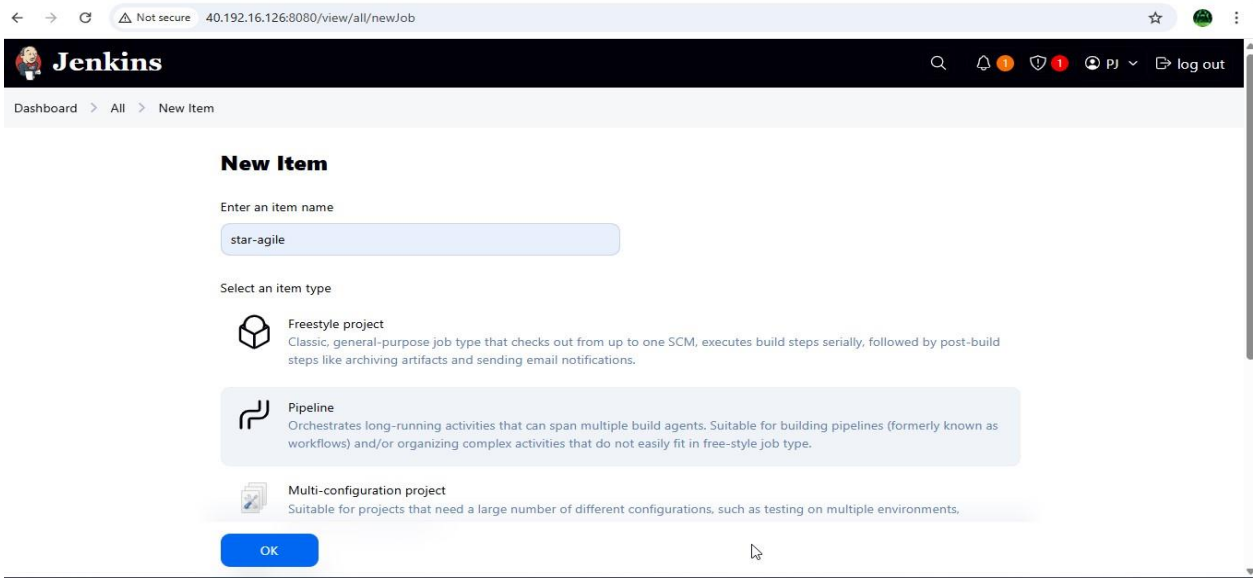
jenkins ALL=(ALL:ALL) NOAPSSWD: ALL restart the jenkins



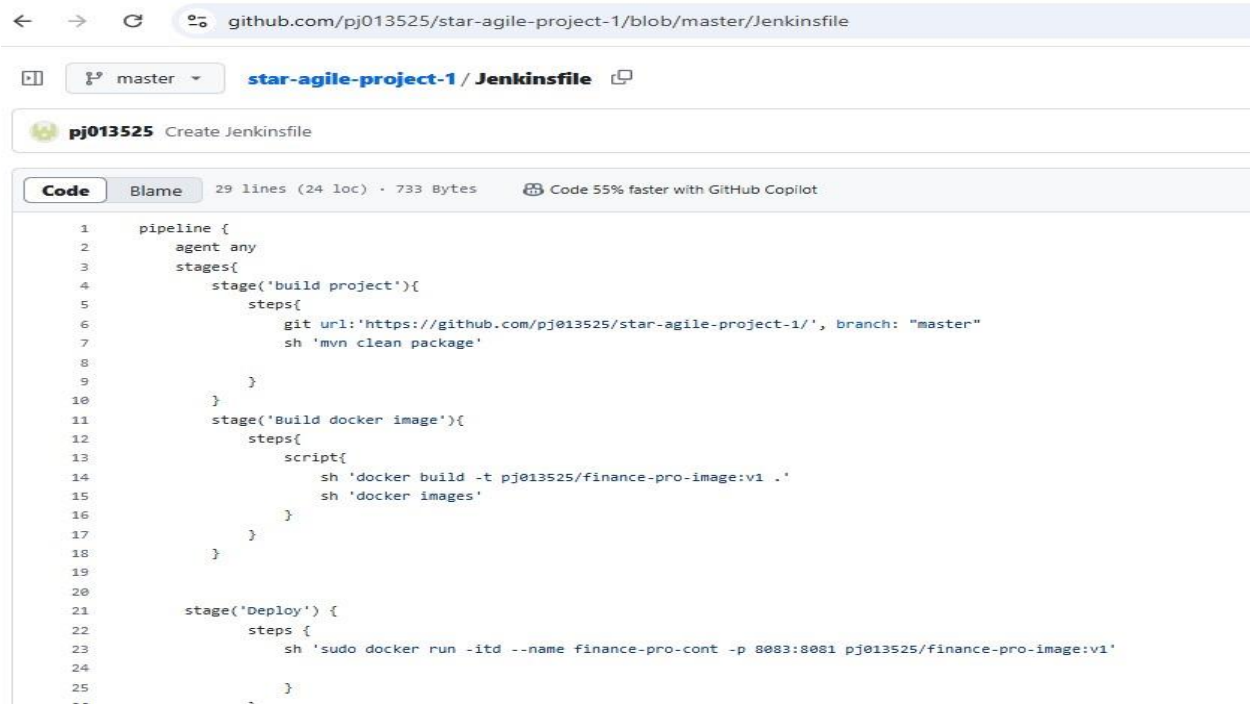

Step19:- Go to the any browser and give the details and click on recommended plugins and login to the Jenkins




Step20:- Now in the Jenkins dashboard click on new item an give any name and select pipeline project as type and click on ok

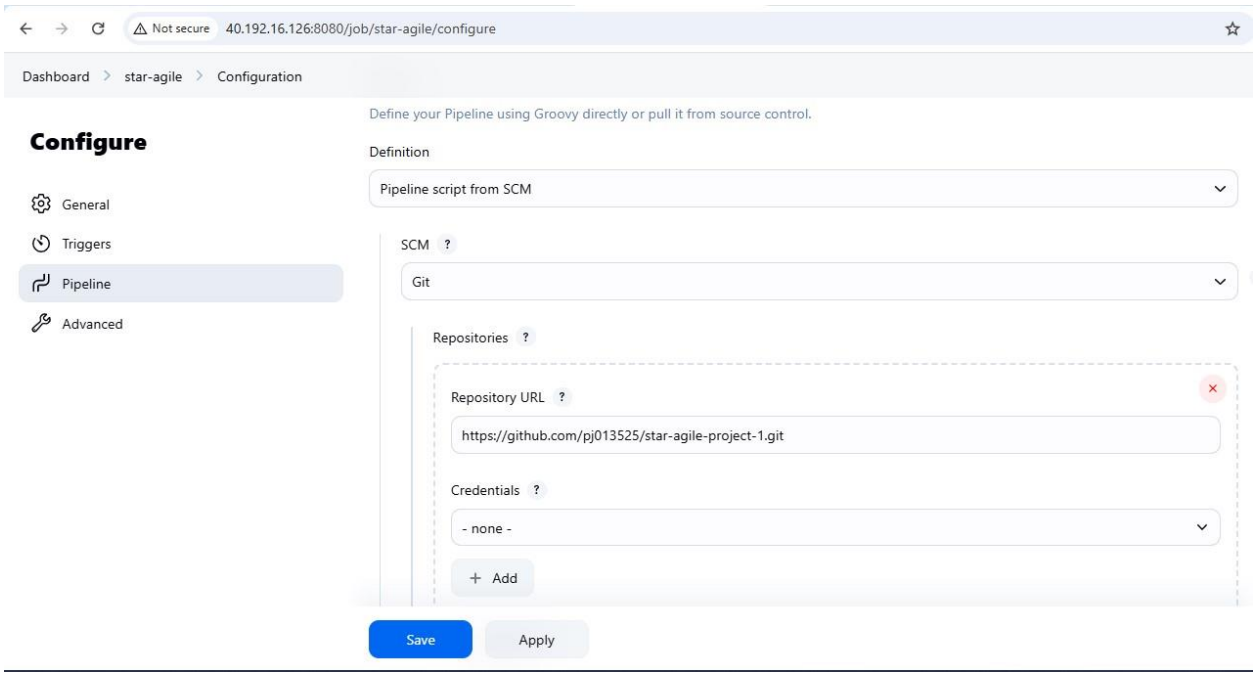



Step21:- Now go to github repo and create a new file with name as Jenkinsfile and press commit changes

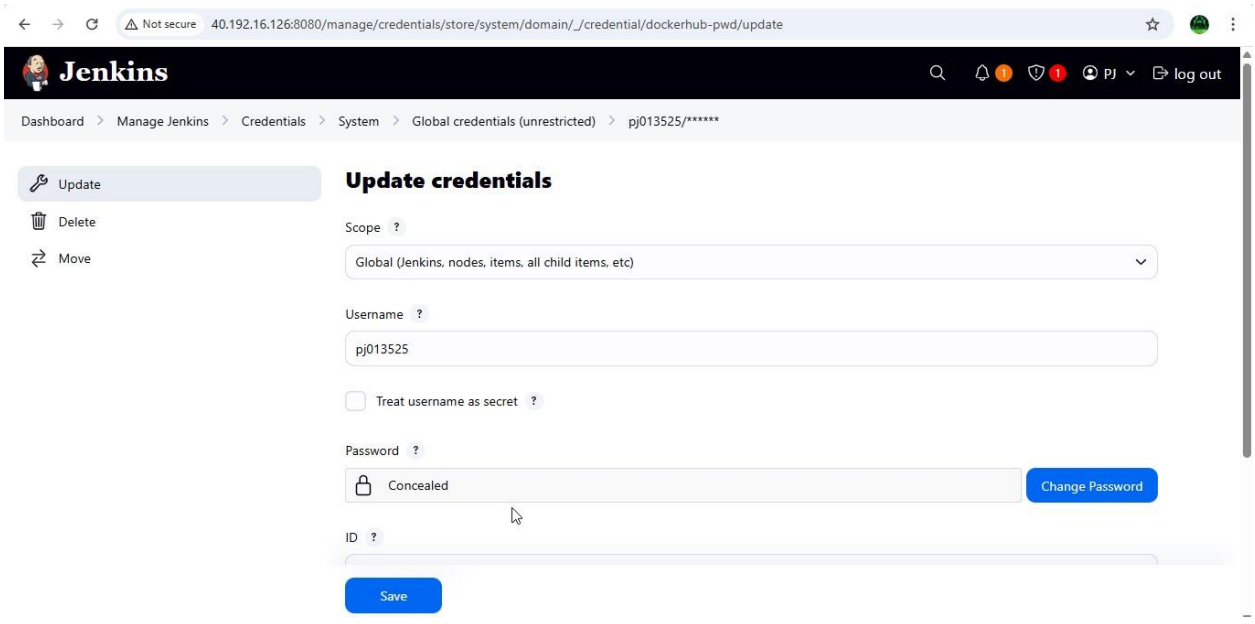



```
pipeline {
    agent any
    stages{
        stage('build project'){
            steps{
                git url:'https://github.com/pj013525/star-agile-project-1/', branch: "master"
                sh 'mvn clean package'

            }
        }
        stage('Build docker image'){
            steps{
                script{
                    sh 'docker build -t pj013525/finance-pro-image:v1 .'
                    sh 'docker images'
                }
            }
        }


     stage('Deploy') {
            steps {
                sh 'sudo docker run -itd --name finance-pro-cont -p 8083:8081 pj013525/finance-pro-image:v1'

                }
```

Step22:- Now go to project in the Jenkins and in the pipeline and do as follows and apply and save

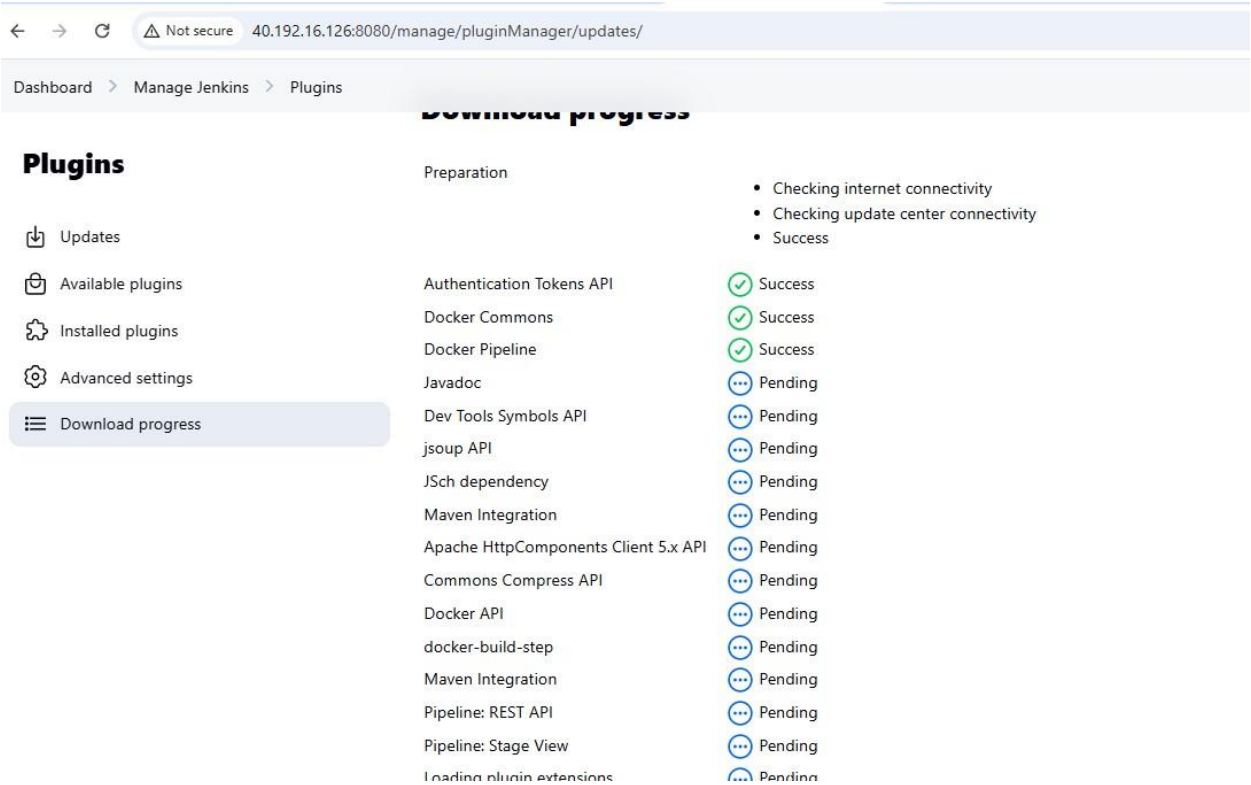



Step23:- Now in dash board ❼ manage Jenkins ❼ credentials ❼ global ❼ add credentials and give dockerhub user name and password and click on create

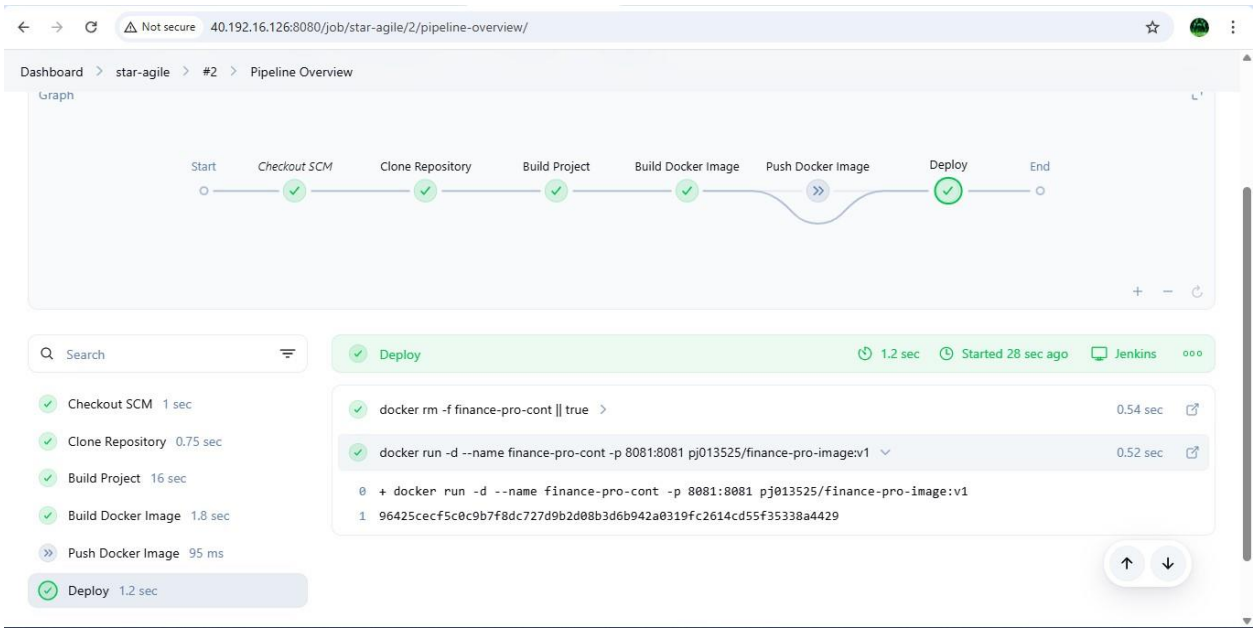


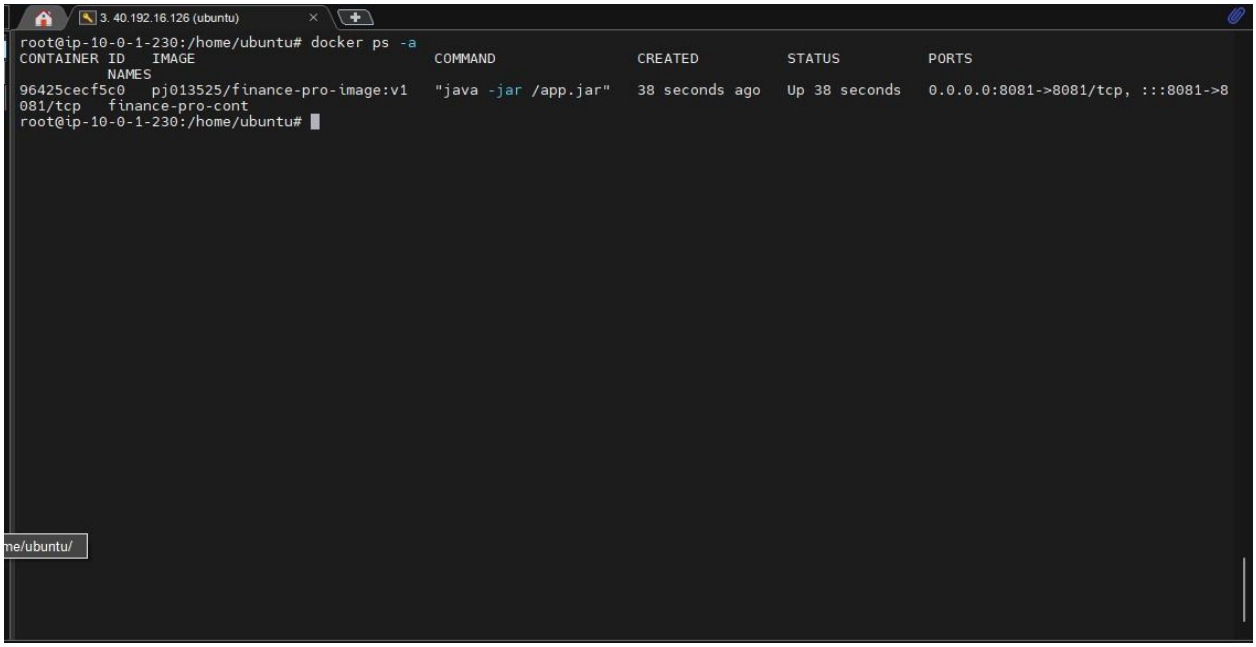

Step24:- Now install docker and other required plugins in the Jenkins

Pipeline
Git Plugin
Docker Pipeline Plugin
Credentials Binding Plugin
Docker Commons Plugin
Pipeline: GitHub
Maven Integration Plugin

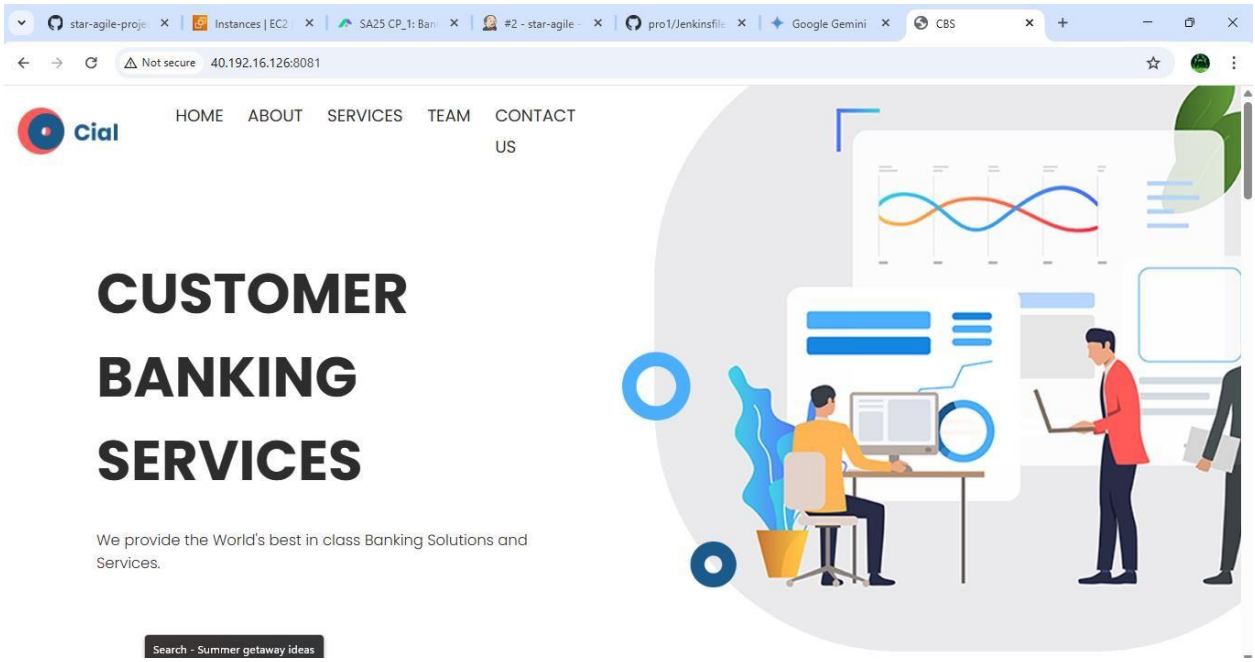



Step25:- Now again go back to Jenkins project and click on Build now to check the status of the build and as you can see that the build is successful and a docker container is also created in the ec2

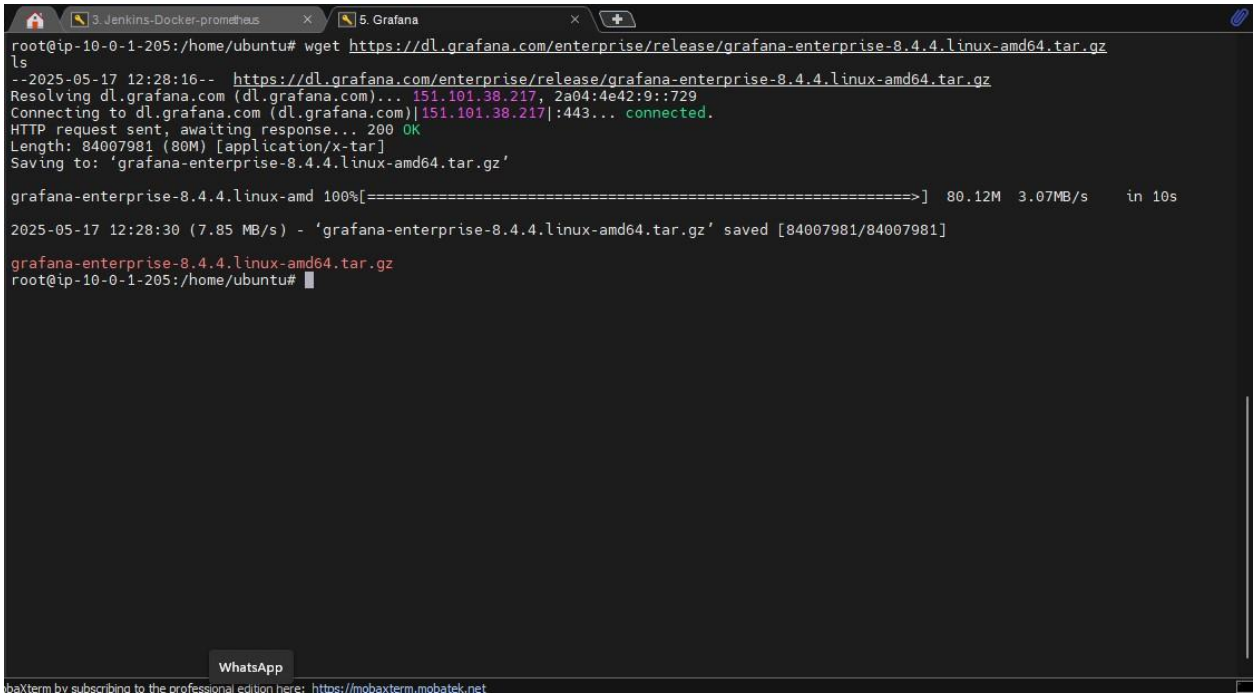


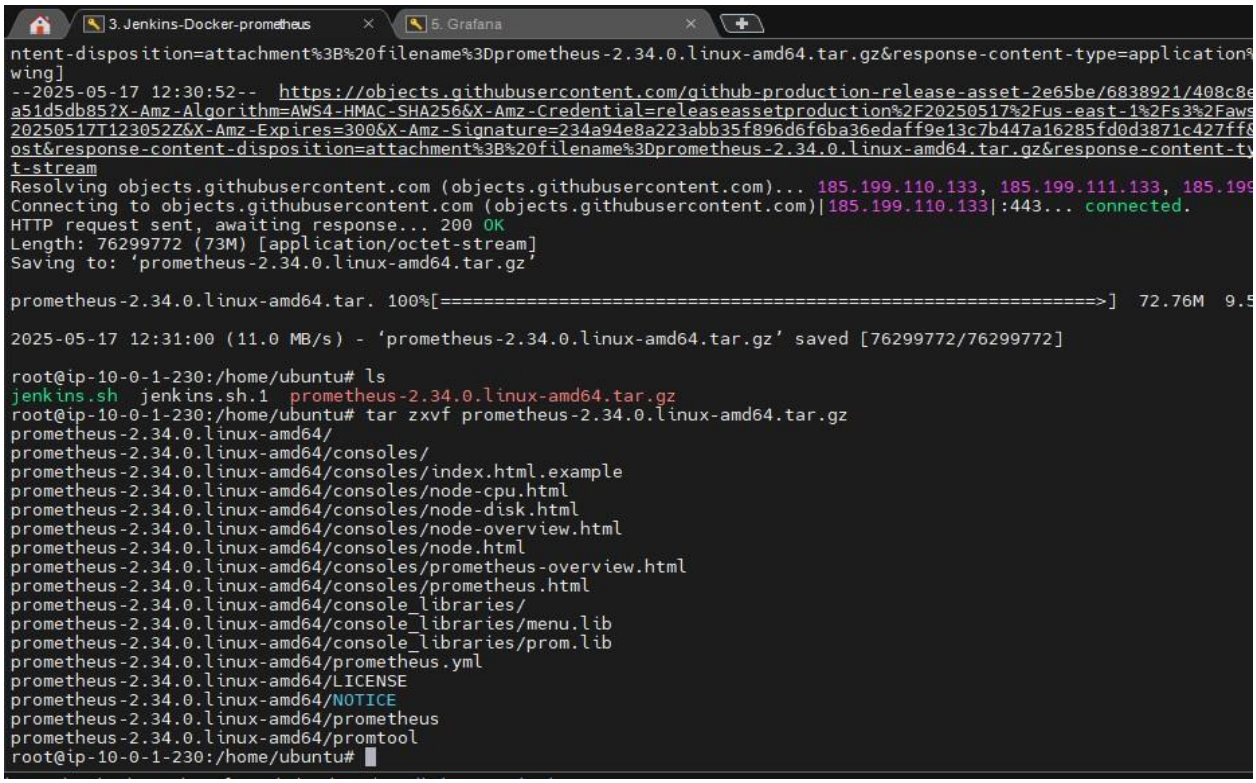




Step26:- Now go to any browser and give the IPaddress:8081 and click enter the you will see the home page of the project and thus the project deployment is successful.

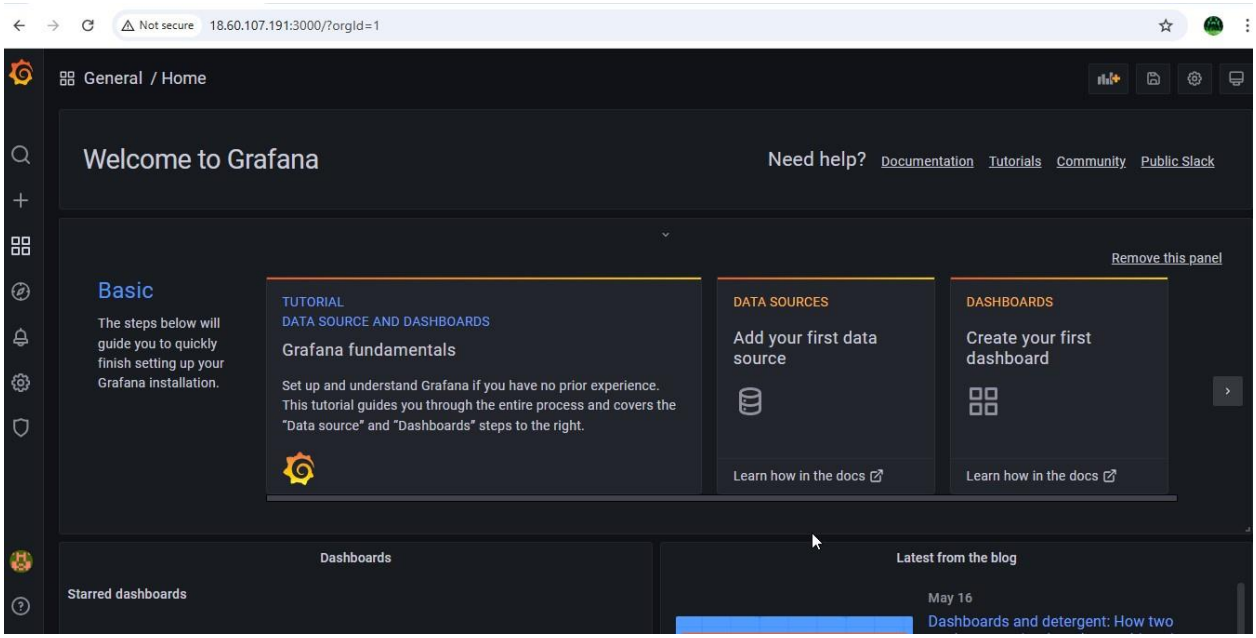



Step27:- Now monitor the docker container using Prometheus and Grafana , for that install Prometheus in Jenkins-Docker server and Grafana in another server






Step28:- After successful installation of Grafana now go to browser and give grafana server ip-address:3000 ( 3000 is default port number for grafana ) and use admin and admin as username and password as they are default and login to the grafana home page



Step29:- Now setup the docker and Prometheus in another using by telling docker that Prometheus would track docker on port 9323

i.e., vi /etc/docker/daemon.json press

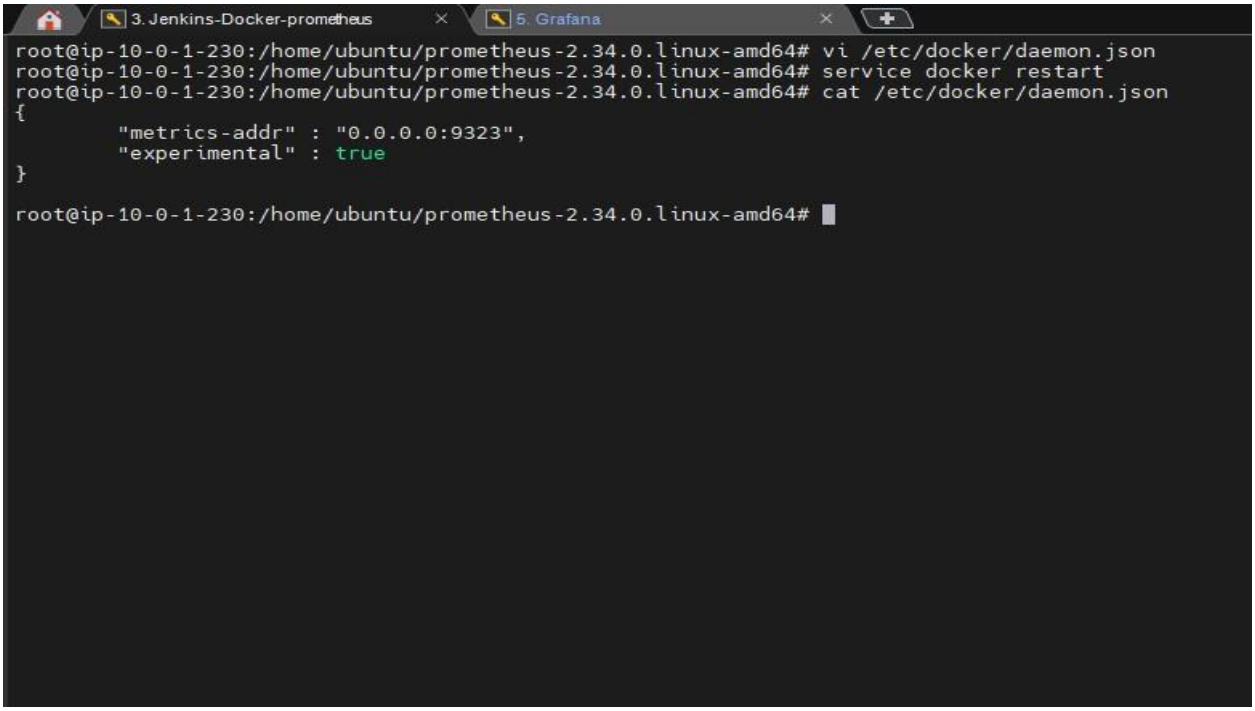
I to insert

{

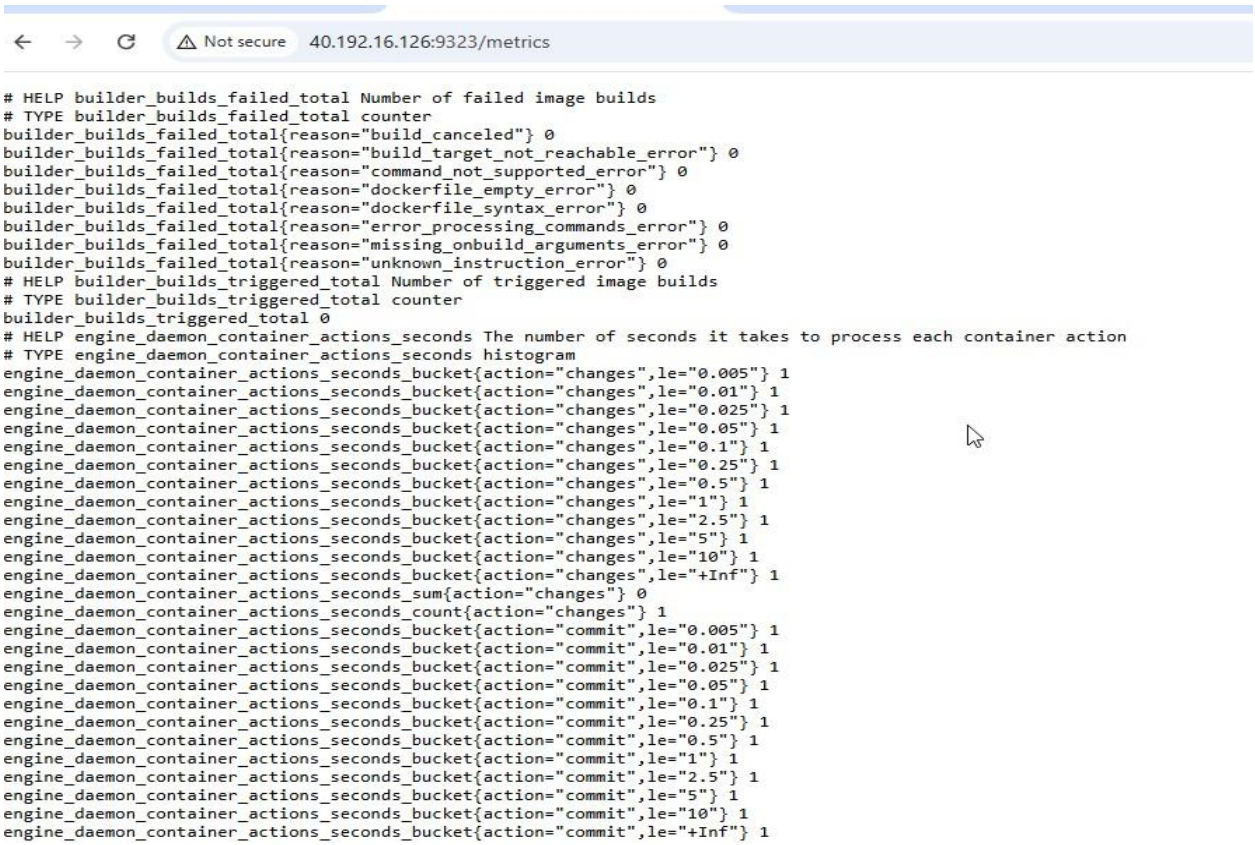
"metrics-addr" : "0.0.0.0:9323",

"experimental" : true

} then save and exit and restart the docker




Step30:- Now go to any browser and give docker ip-address:9323/metrics and in the below image you will see that the docker stats have been started successfully




Step31:- Now add docker job in the Prometheus.yml file to give this stats to Prometheus vi prometheus.yml

- job_name: "docker"

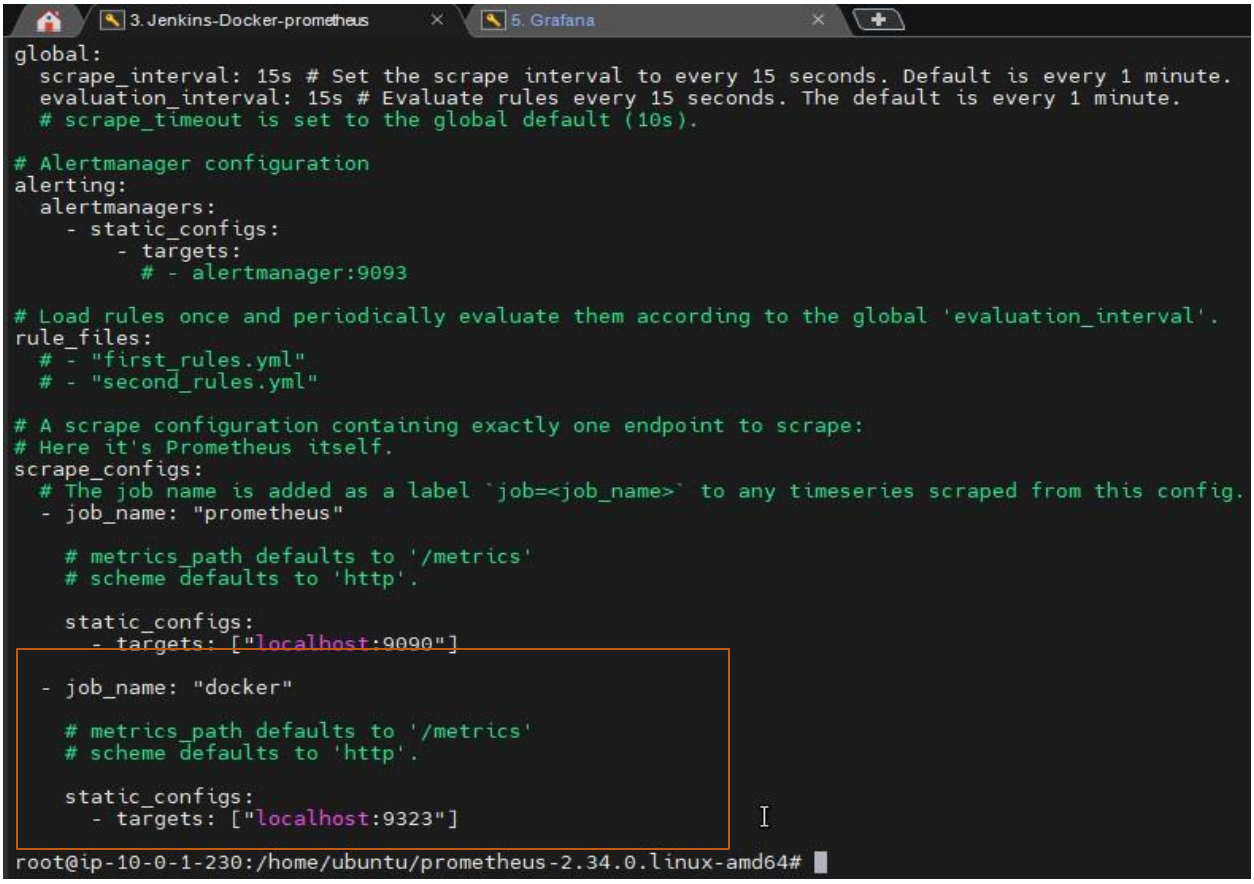
  # metrics_path defaults to '/metrics'

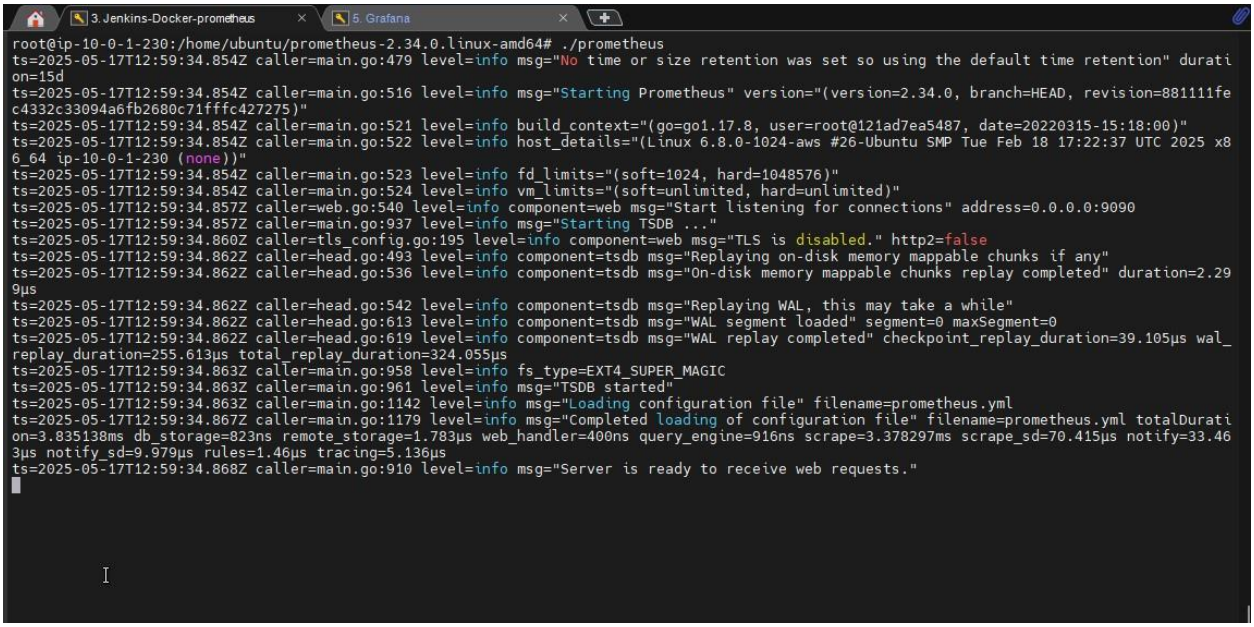
# scheme defaults to 'http'.

static_configs:

- targets: ["localhost:9323"]

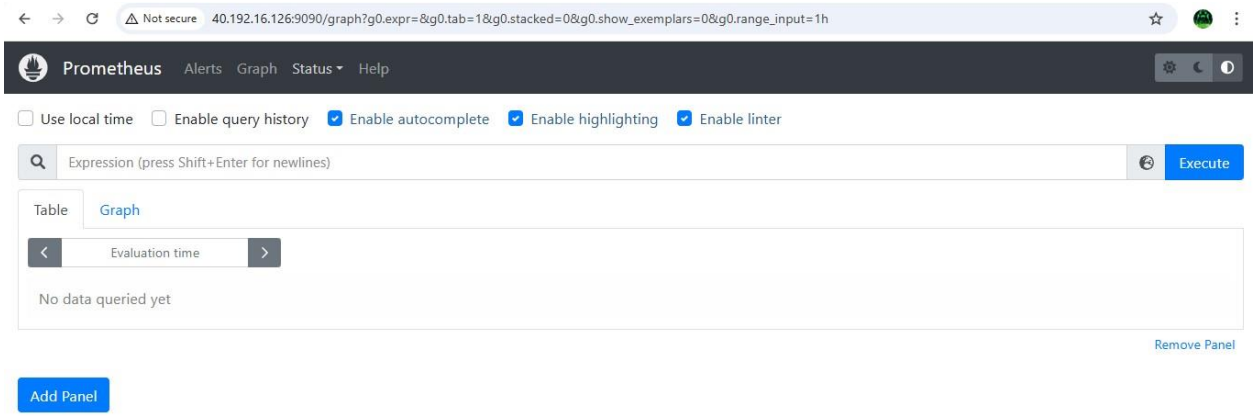
Save the file and exit and start the Prometheus using ./prometheus



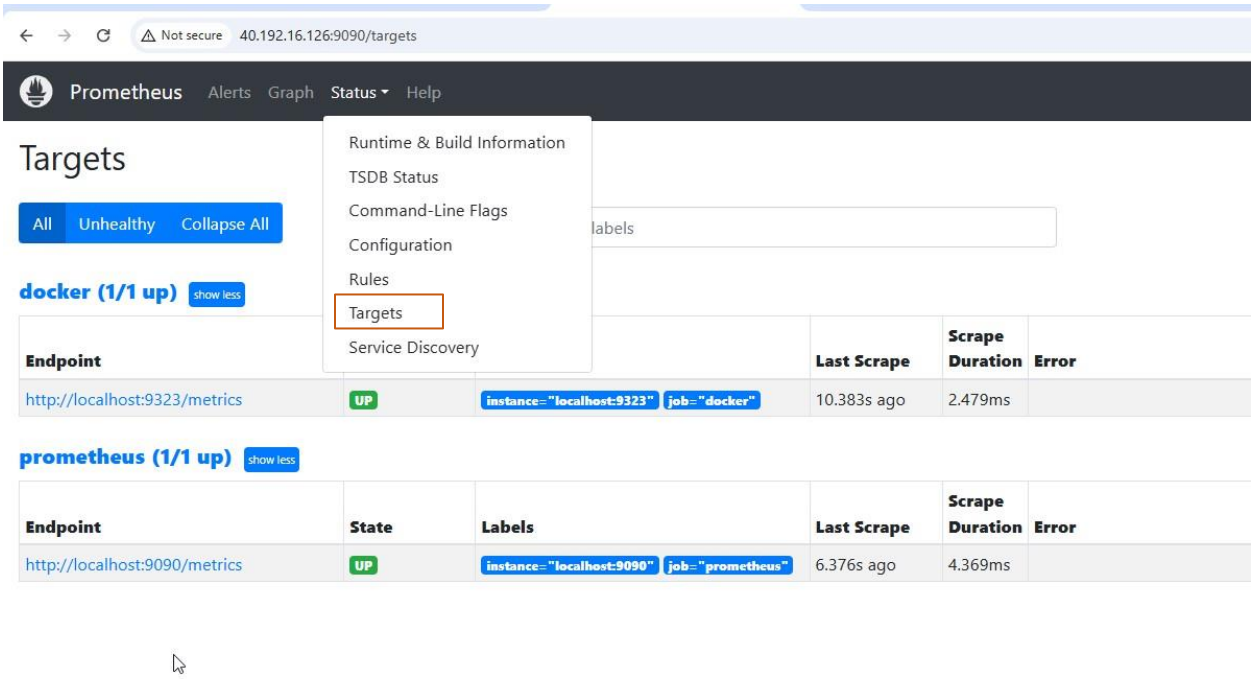

As you can see that the Prometheus have been started form the above image

Step32:- Now go browser and give docker ip:9090 and enter , then you will be successfully enter into the Prometheus homepage



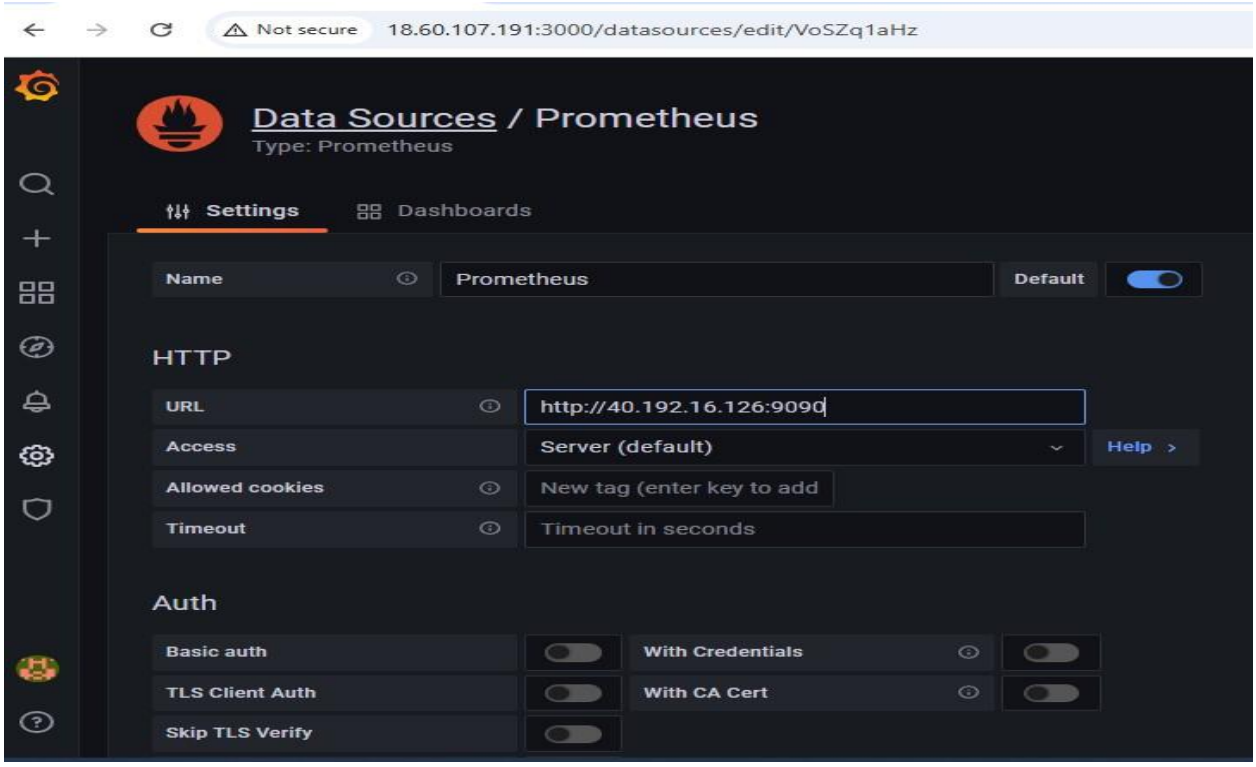


Step33:- Now click on status ❼ targets then you will see the status of the docker and Prometheus

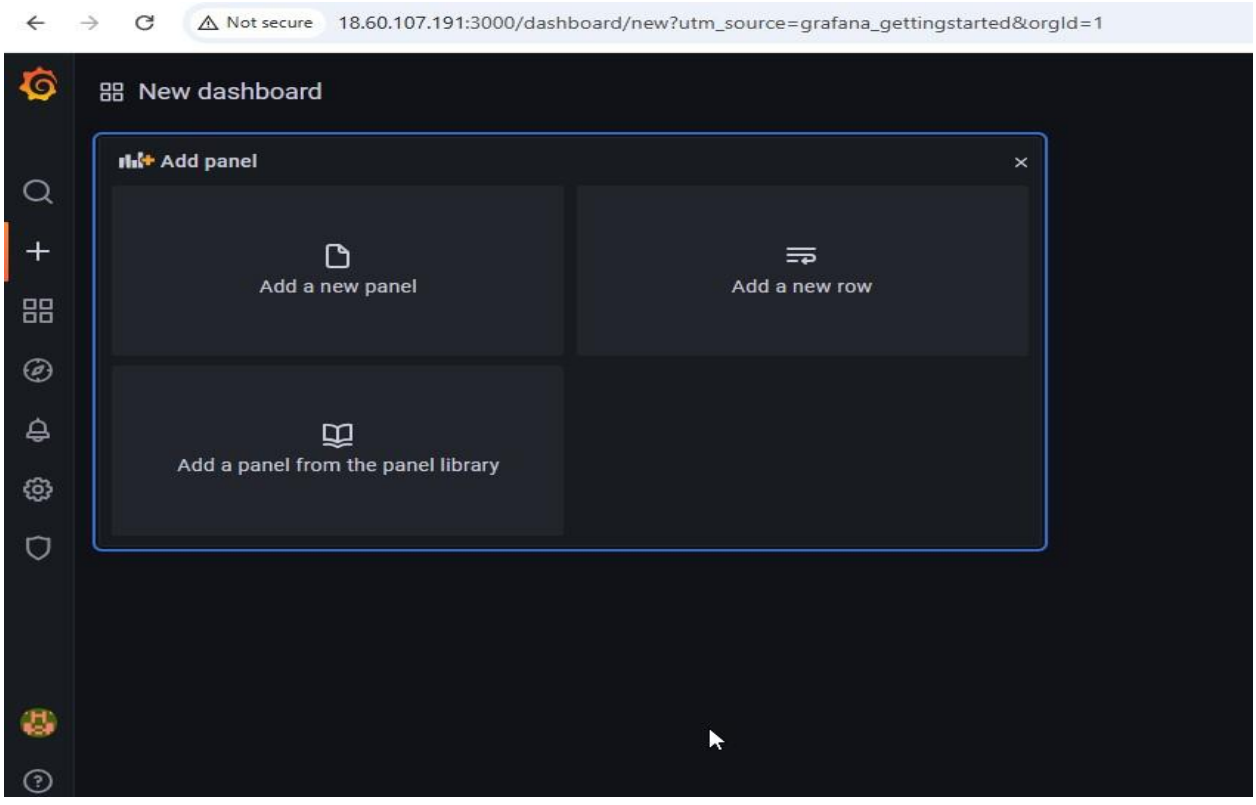


Step34:- Now go to grafana homepage ❼ configurations ❼ Data sources

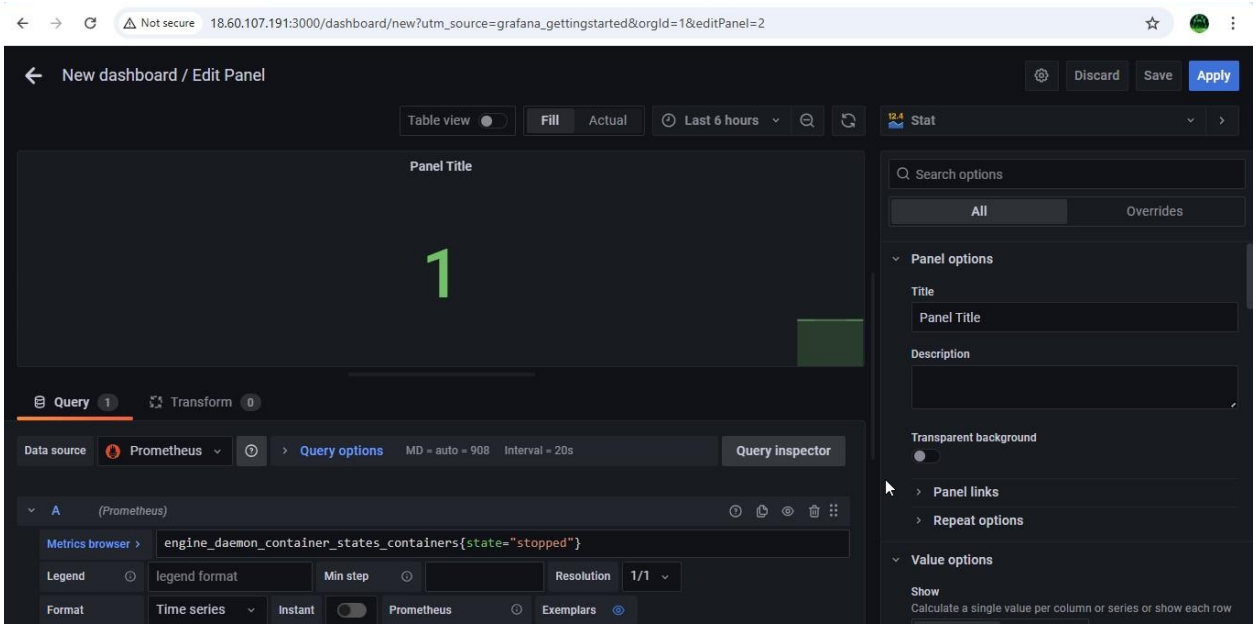

Step35:- Now click on add Data sources ❼ Prometheus and give ipaddress:9090 and click on save and test



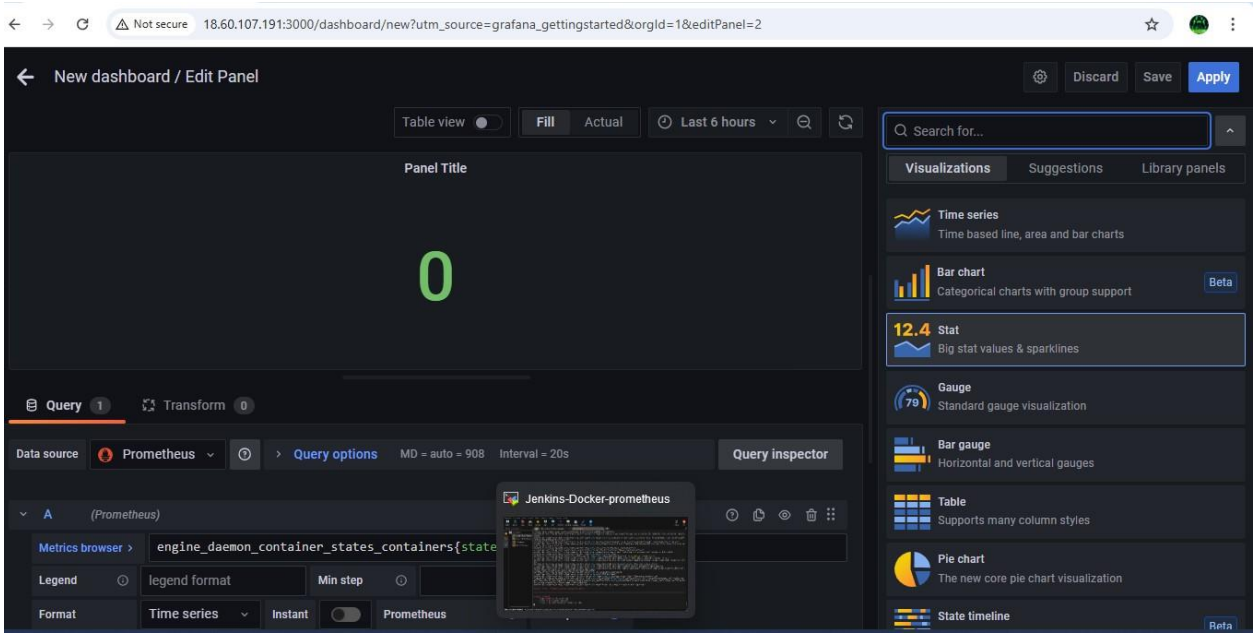

Step36:- Now click on Dash board ❼ add new panel



Step37:- Now in the metrics browser give  engine daemon container states containers{state="stopped"} and you will see the result that same as in the metrics from the browser



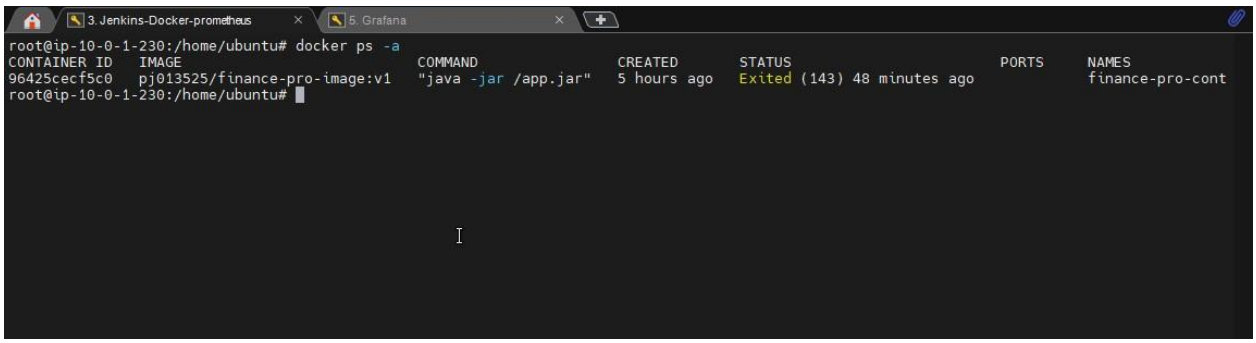

engine daemon container states containers{state="running"}



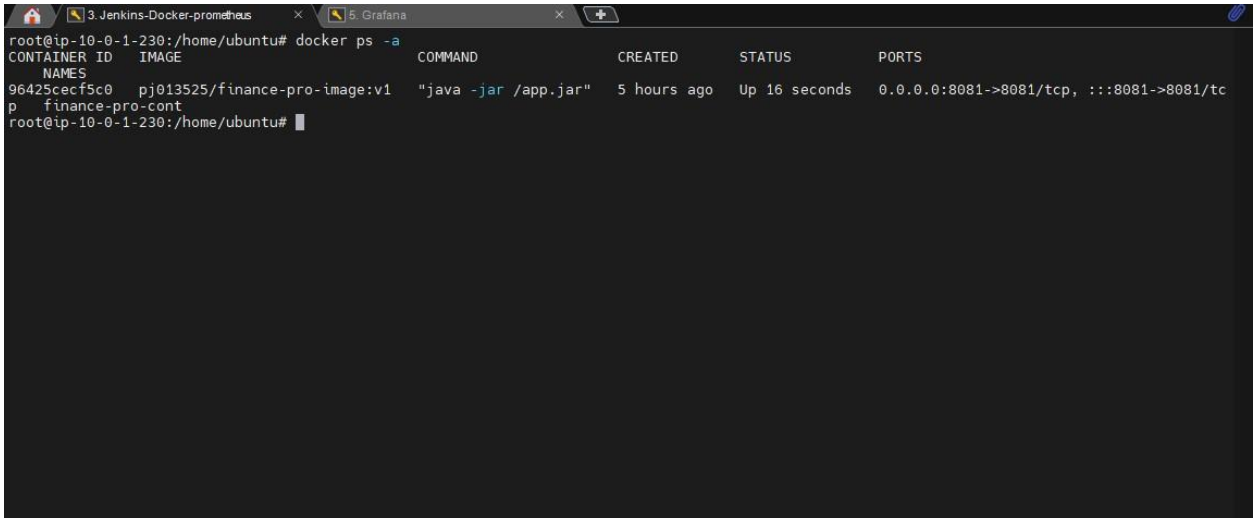

Step38:- The values shown in the panel must be equal to the that of shown in the docker stats

```
# TYPE engine_daemon_container_states_containers gauge
engine_daemon_container_states_containers{state="paused"} 0
engine_daemon_container_states_containers{state="running"} 0
engine_daemon_container_states_containers{state="stopped"} 1
# HELP engine_daemon_engine_cpus_cpus The number of cpus that the host system of the engine has
```

Here the container which we created is in exited state so it is showing as stopped state in stats



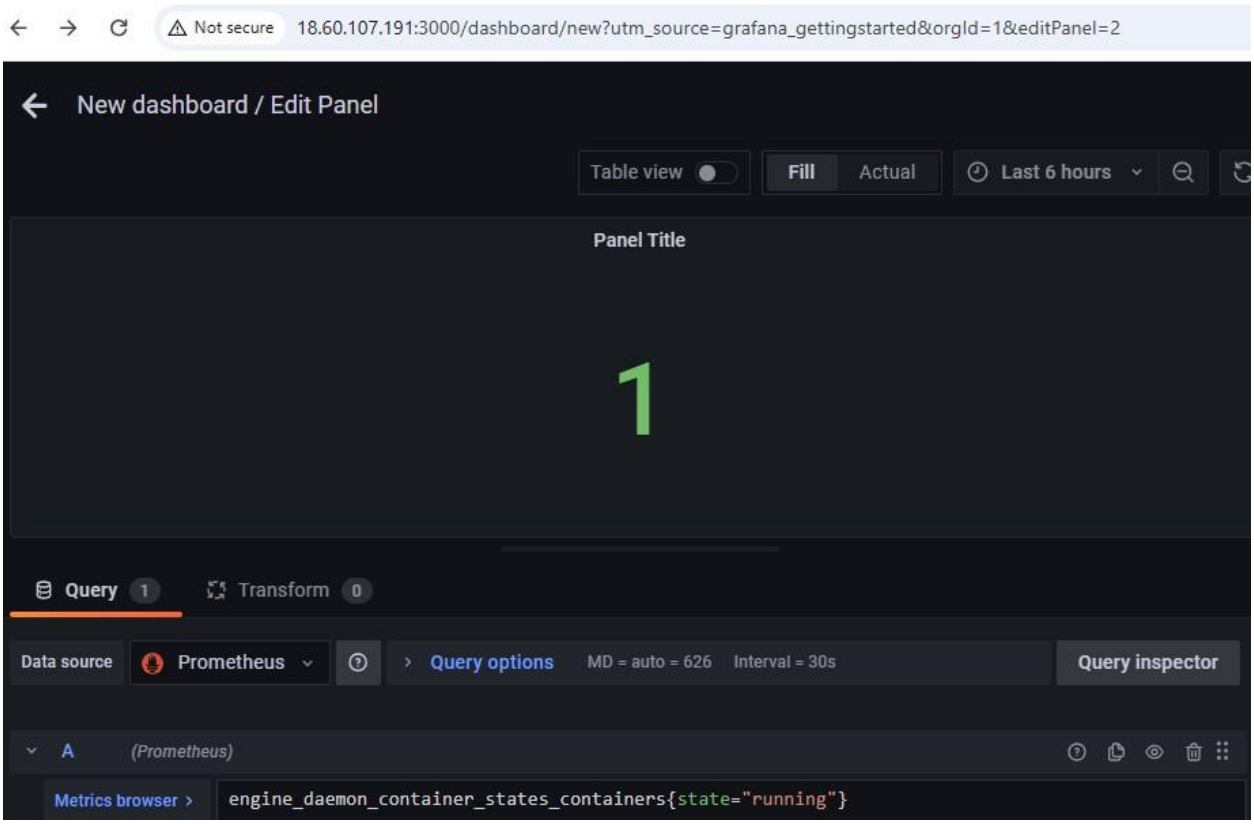

Step39:- Now start the container again and check the details again in the stats




Now check tin the docker stats

```
# TYPE engine_daemon_container_states_containers gauge
engine_daemon_container_states_containers{state="paused"} 0
engine_daemon_container_states_containers{state="running"} 1
engine_daemon_container_states_containers{state="stopped"} 0
# HELP engine_daemon_engine_cpus_cpus The number of cpus that the host system of the engine has
# TYPE engine_daemon_engine_cpus_cpus gauge
```

As you see that the state=running is 1 then give this in the metrics browser in the Grafana dash board panel and see the value




As you can the value is changed from 0 to 1 in the runng state as we started container from exited to running state

Step40:- This is how we monitor the health of a container automatically and visualising the report using Prometheus and Grafana.