**Capstone Project**

Insurance Project

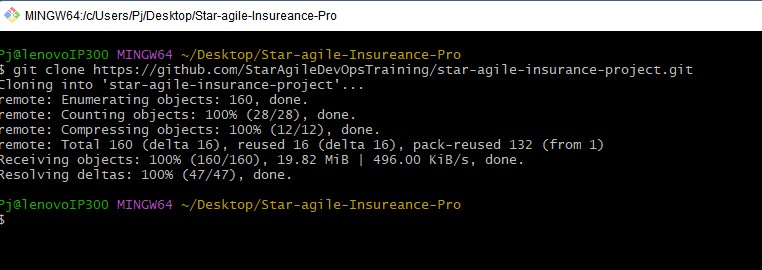
By:- Jonna Padmarao

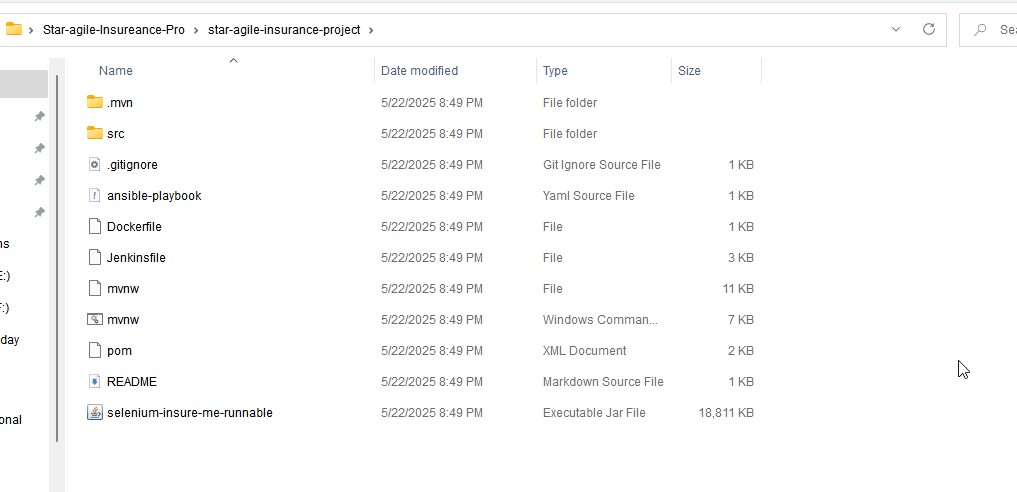
Source URL :- <https://github.com/pj013525/star-agile-project-3.git>

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

Step2:- Now give git clone

[https://github.com/StarAgileDevOpsTraining/sta](https://github.com/StarAgileDevOpsTraining/star)[r-agile-](https://github.com/StarAgileDevOpsTraining/star-agile-banking-finance)insuranceproject.git [t](https://github.com/StarAgileDevOpsTraining/star-agile-banking-finance)o get the project code in to that folder

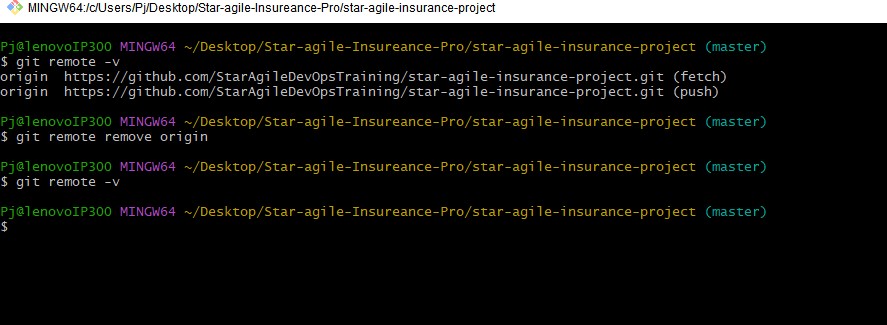




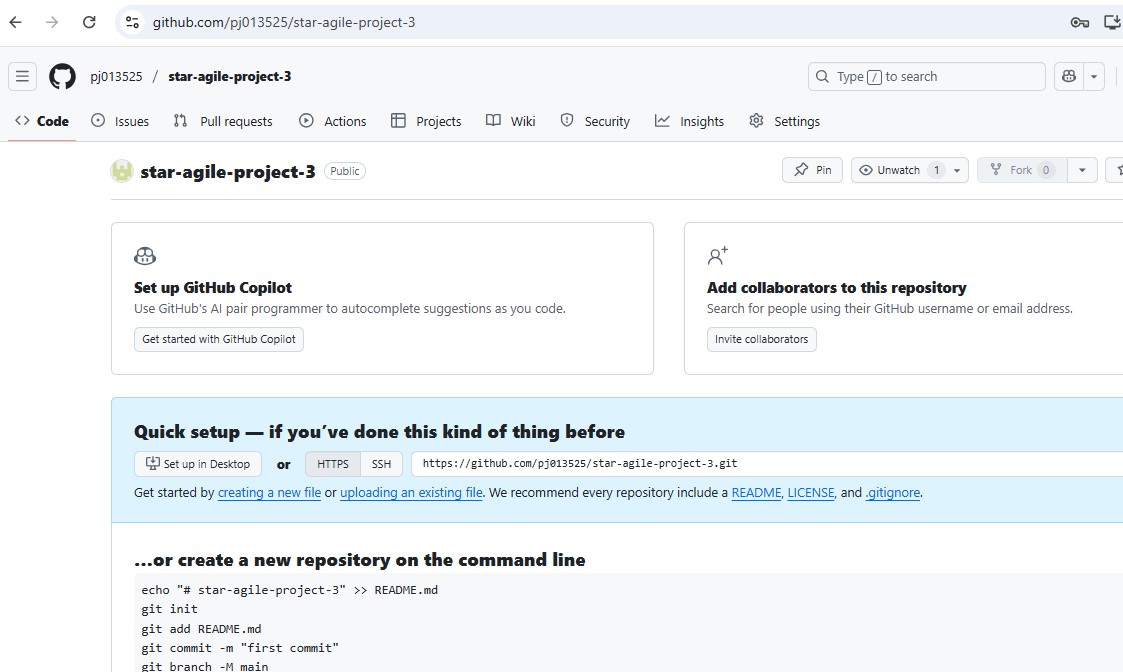
Step3:- Now go to the folder that we get from git clone and again opengit bash there and check the origin and remove that origin

git remote -v --> To get origin list

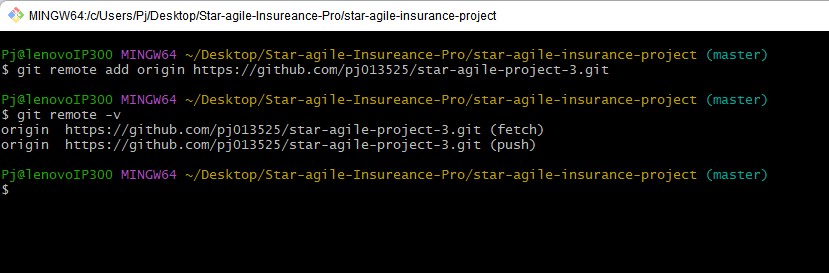
git remote remove origin ==> to remove the origin



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

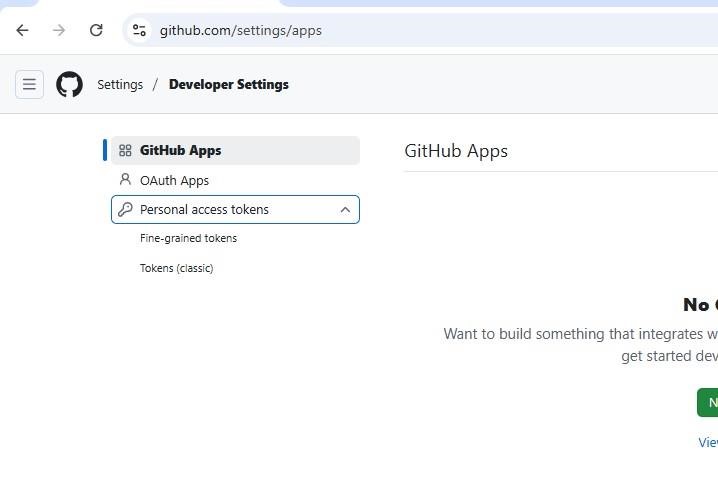


Step5:- Now again go to the gitbash and add this git repo url in the project by using git remote add origin <git-repo-url> and verify



Step6:- Now again go to github  Profile setting Developer settings

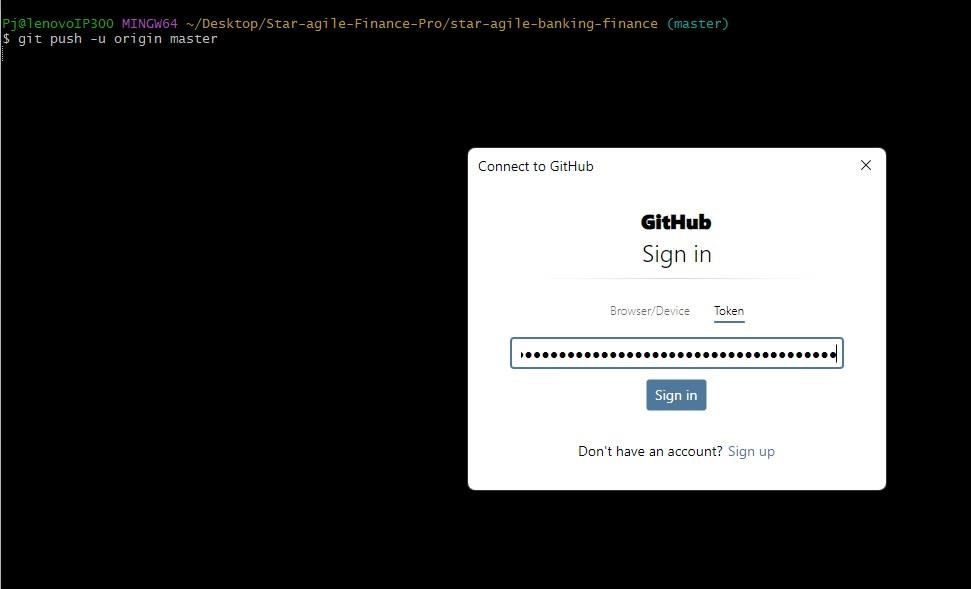
 Personal access token Tokens(classic) Generate new token



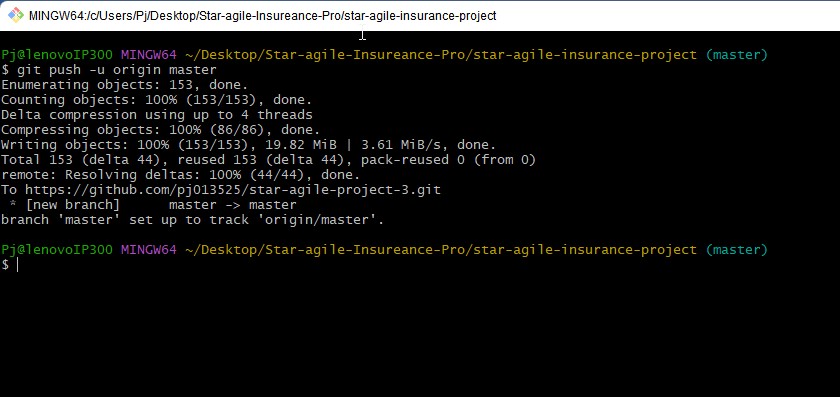
Step7:- Now a token will be generated , copy this token that generated since it is only available for one time only



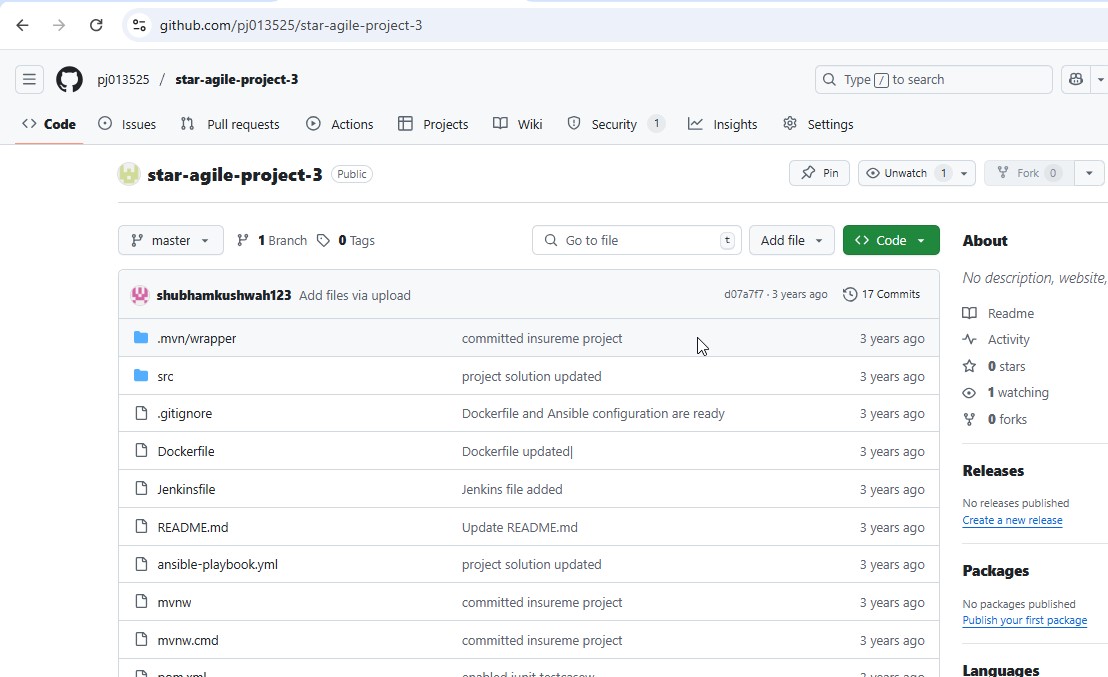
Step8:- Now give link this the remote repo with gitbash using this token git push -u origin master and paste the token the copied from the github and press sign in



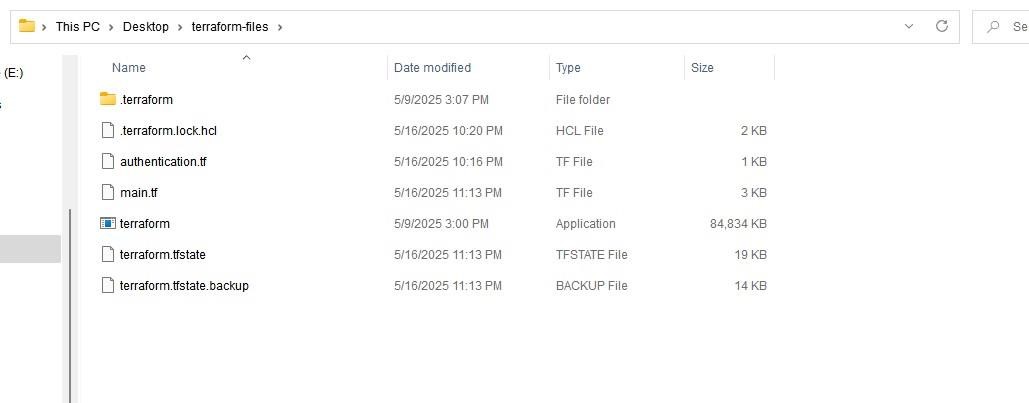
Step9:- Now the master branch will be set to our repo by default



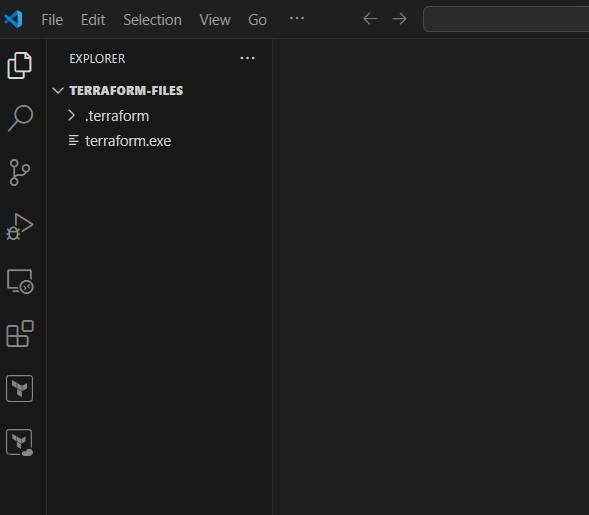
Step10:- Now go to the github repo and you will see the source code in that repo



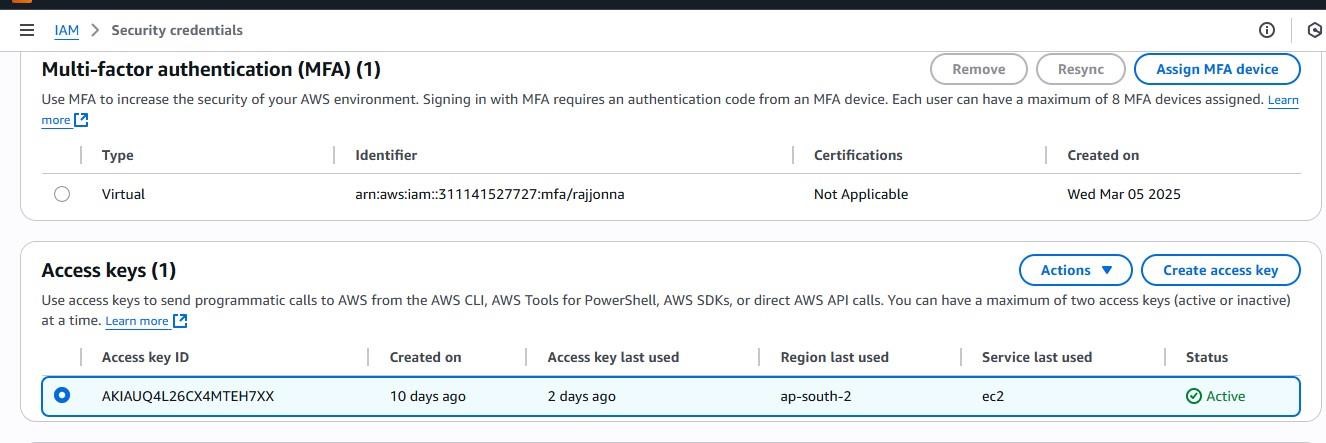
Step11:- Now create an instance using terraform as Iaac , and for that create a folder on desktop and go to browser download terraform for windows then a terraform application will be generated , now copy this application in to that folder and save



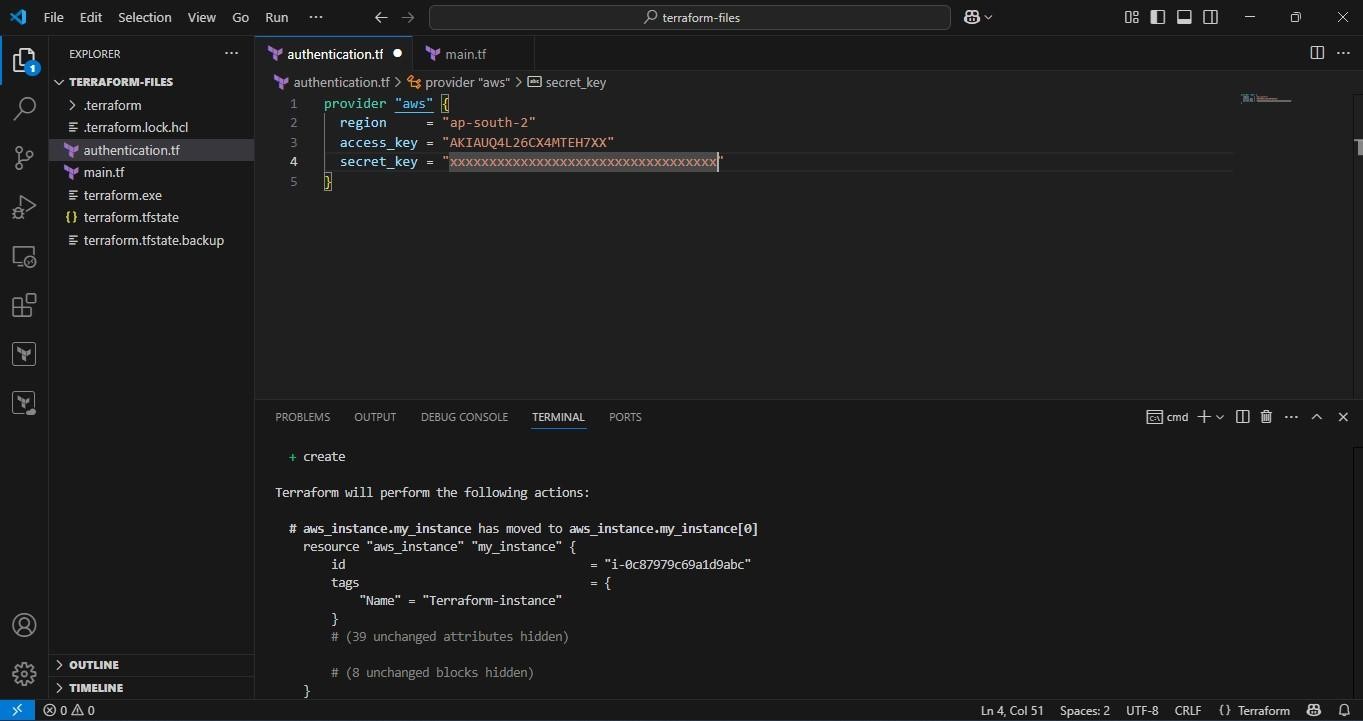
Step12:- open visual studio code and go to terraform folder



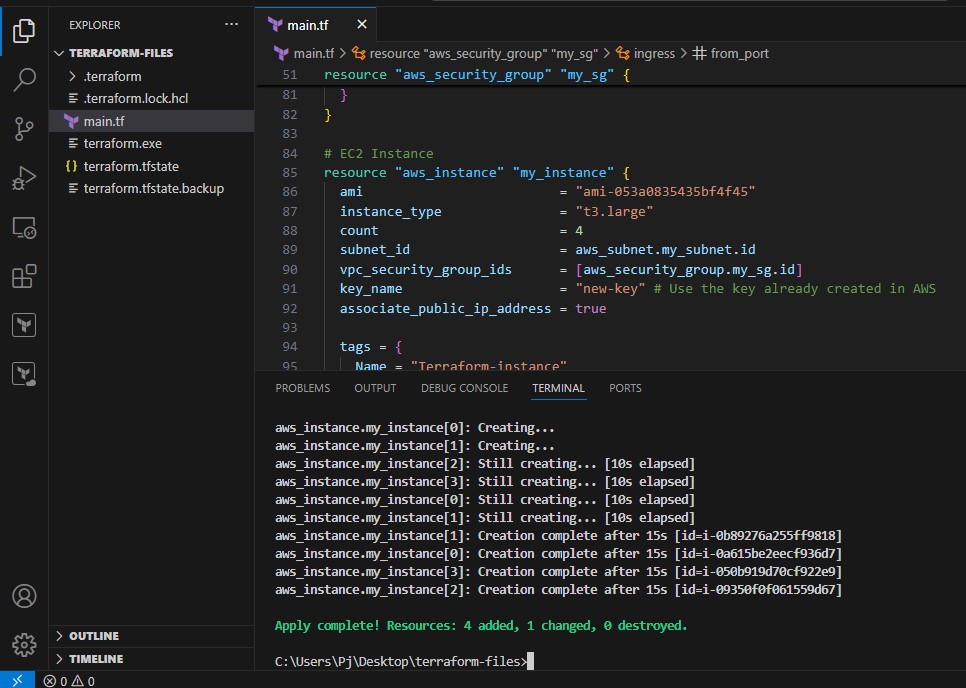
Step13:- Now create a file authentication.tf and give the provider and for that select the region in which you want to launch the server and go to aws account and go to profile  credentials and go to access keys



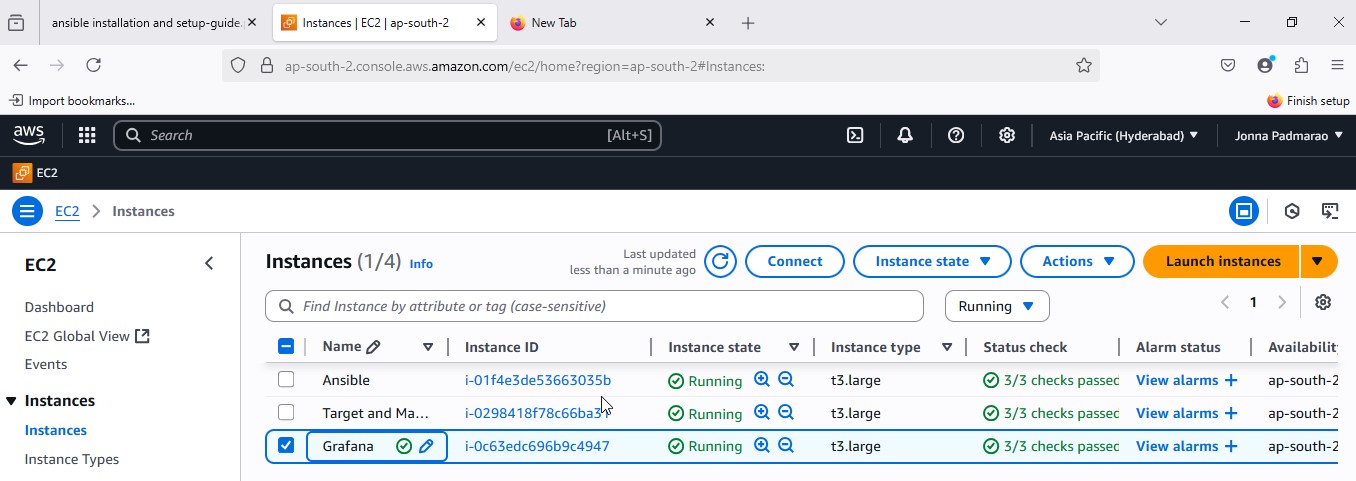
Step14:- now copy this access key details and paste it in this authentication.tf file and initialize it



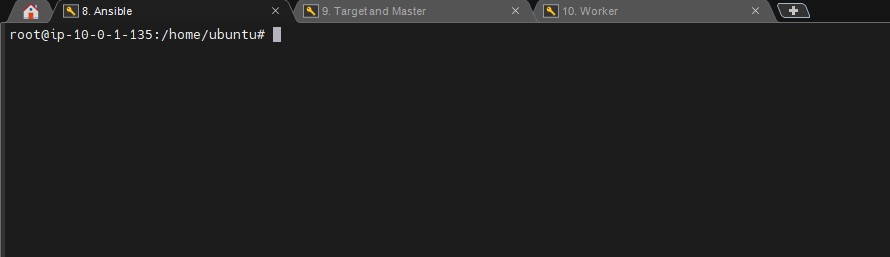
Step15:- After it is successful now create a new file main.tf and give resources details to create instance



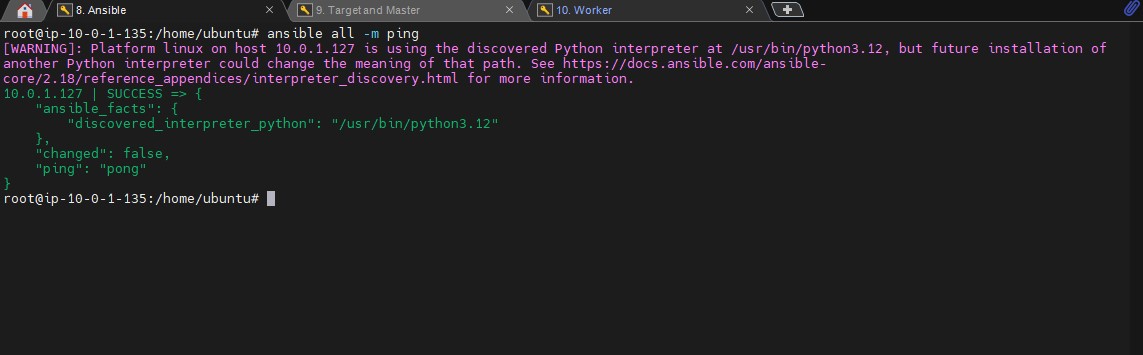
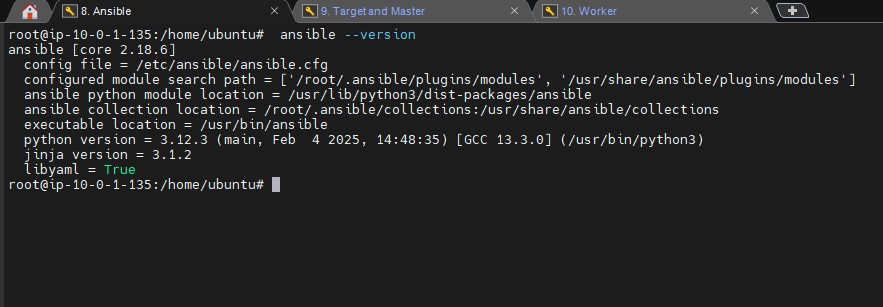
Step16:- After it is successful go and check the aws console and rename them as Ansible , Target and Master, Worker and Grafana instances



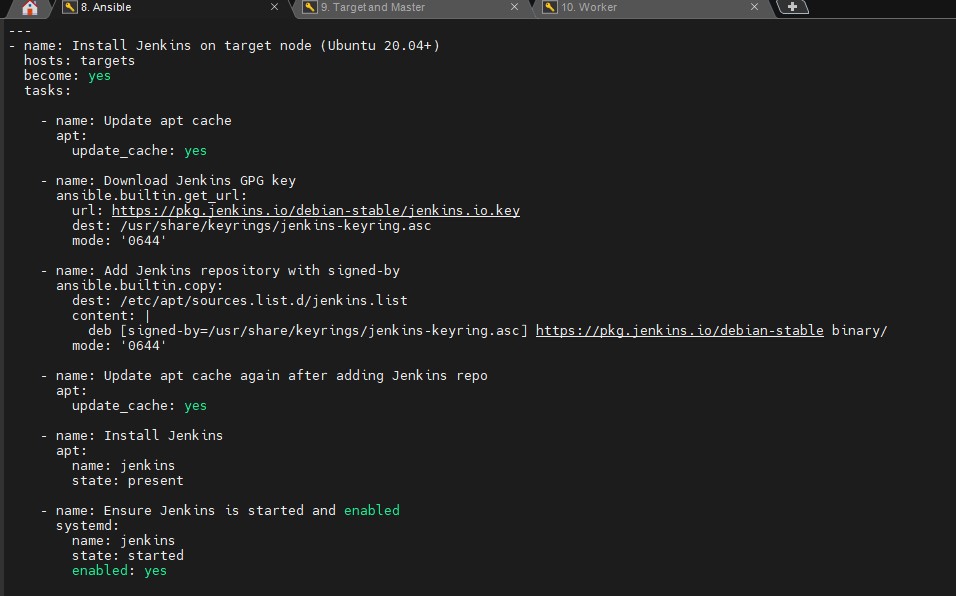
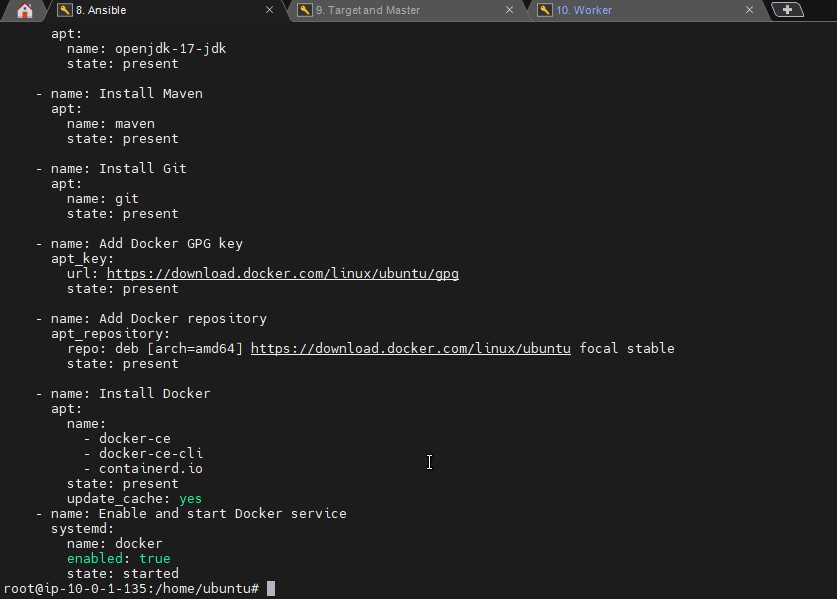
Step17:- Now connect to Ansible and target and master and worker servers using Mobaxterm agent and launch an instance



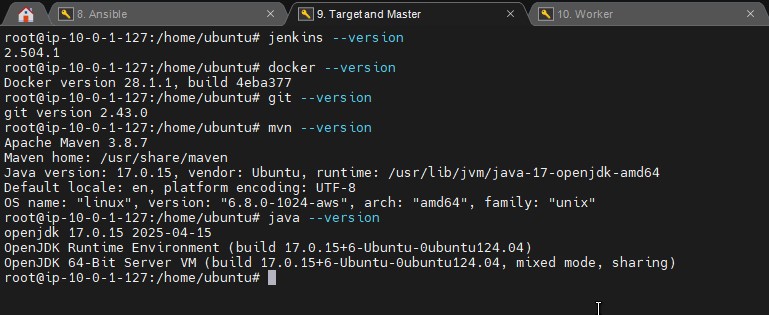
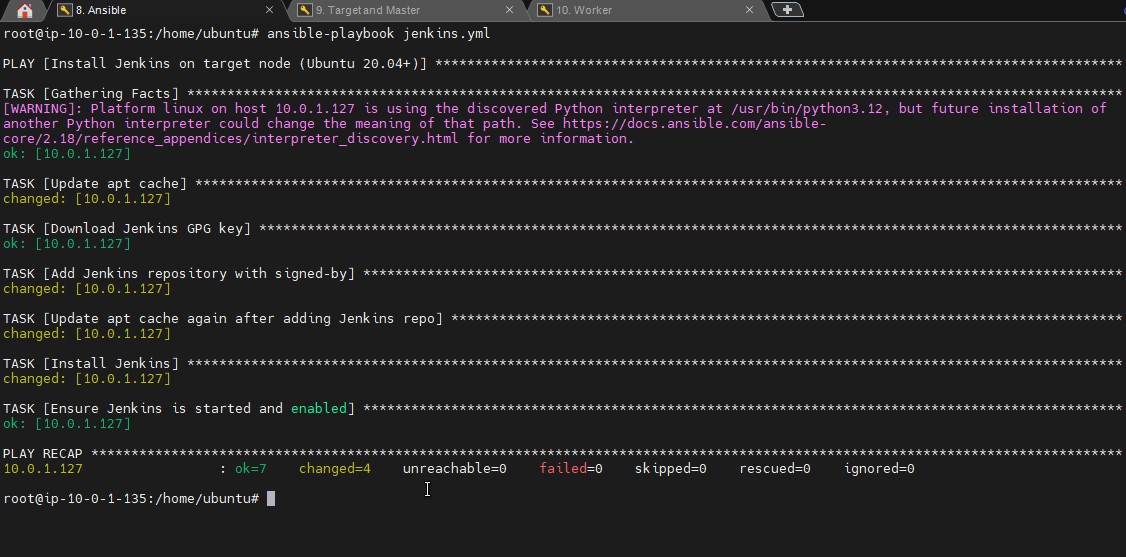
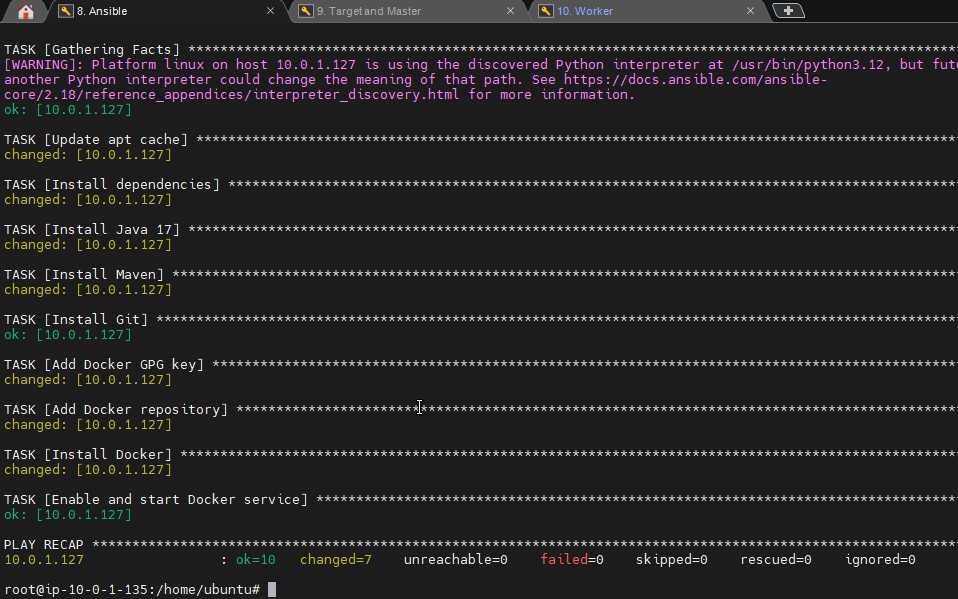
Step18:- Now install Ansible in Ansible server and connect this server with the Target and master sever and enable All traffic in the security group of this server



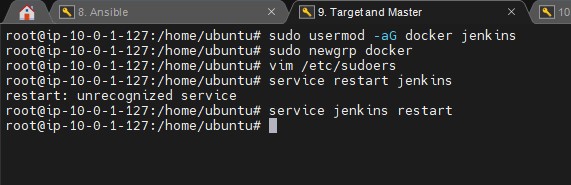
Step19:- Now install java, maven, docker, jenkins in the target and master server using Ansible sever



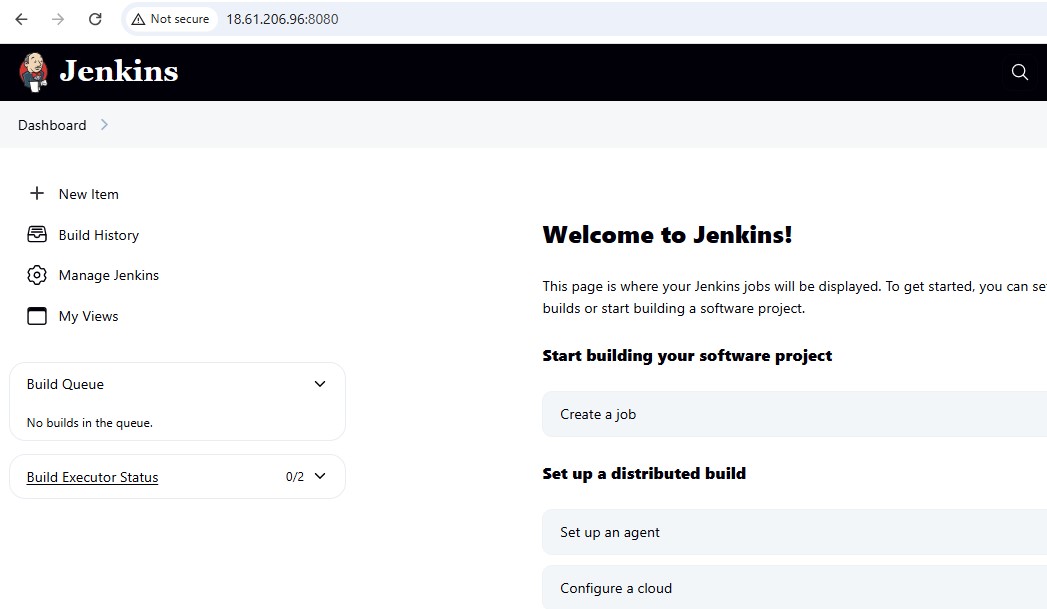
Step20:- Now run the yaml files to install packages in the target and master node and go to target and master node and verify the packages



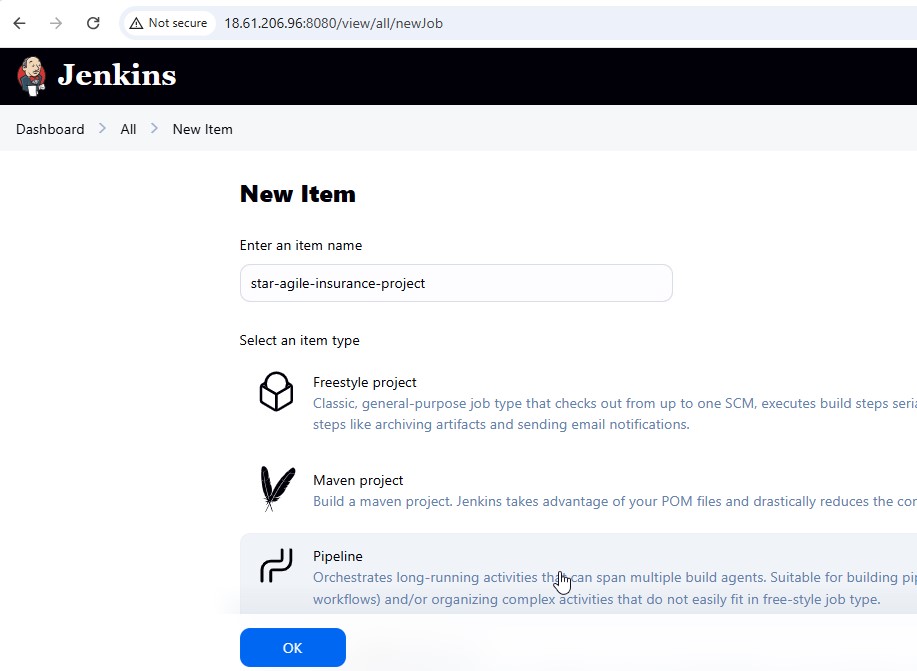
Step21:- Now add Jenkins group to docker and give root permissions to the Jenkins user in the sudoers file as under root give jenkins ALL=(ALL:ALL) NOAPSSWD: ALL restart the jenkins



Step22:- Go to the any browser and give the details and click on recommended plugins and login to the Jenkins



Step23:- Now in the Jenkins dashboard click on new item an give any name and select pipeline project as type and click on ok



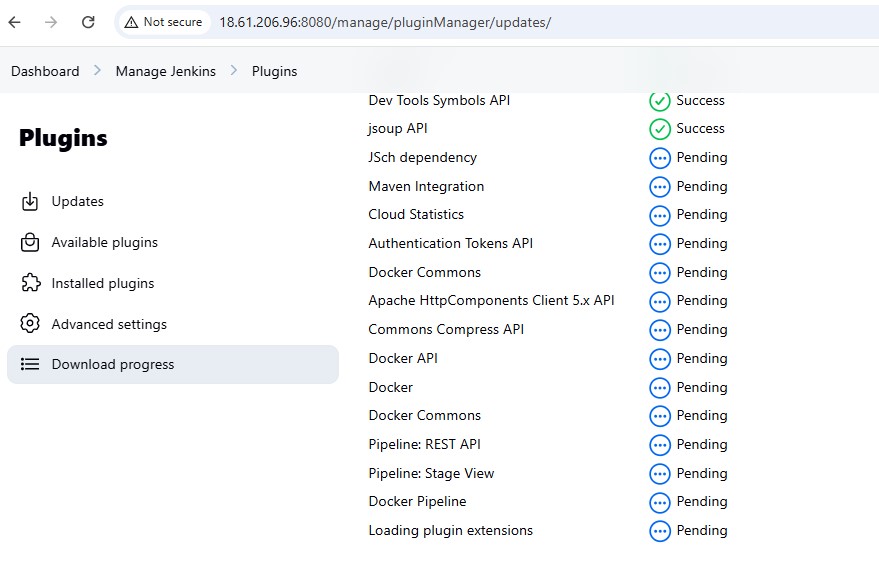
Step24:- Now install docker and other required plugins in the Jenkins

Pipeline stage view Maven Integration Plugin

Git Plugin Docker Commons Plugin

Docker Pipeline Plugin Pipeline: GitHub

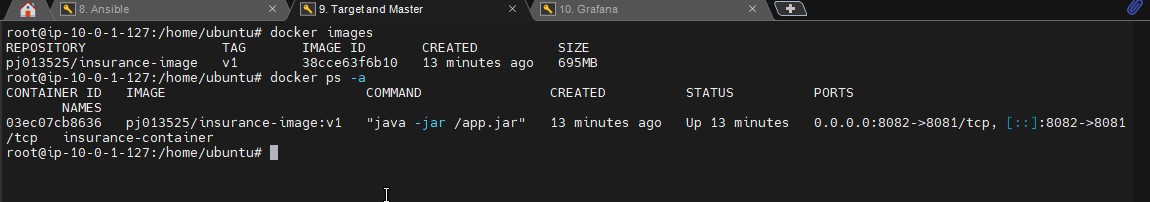
Credentials Binding Plugin

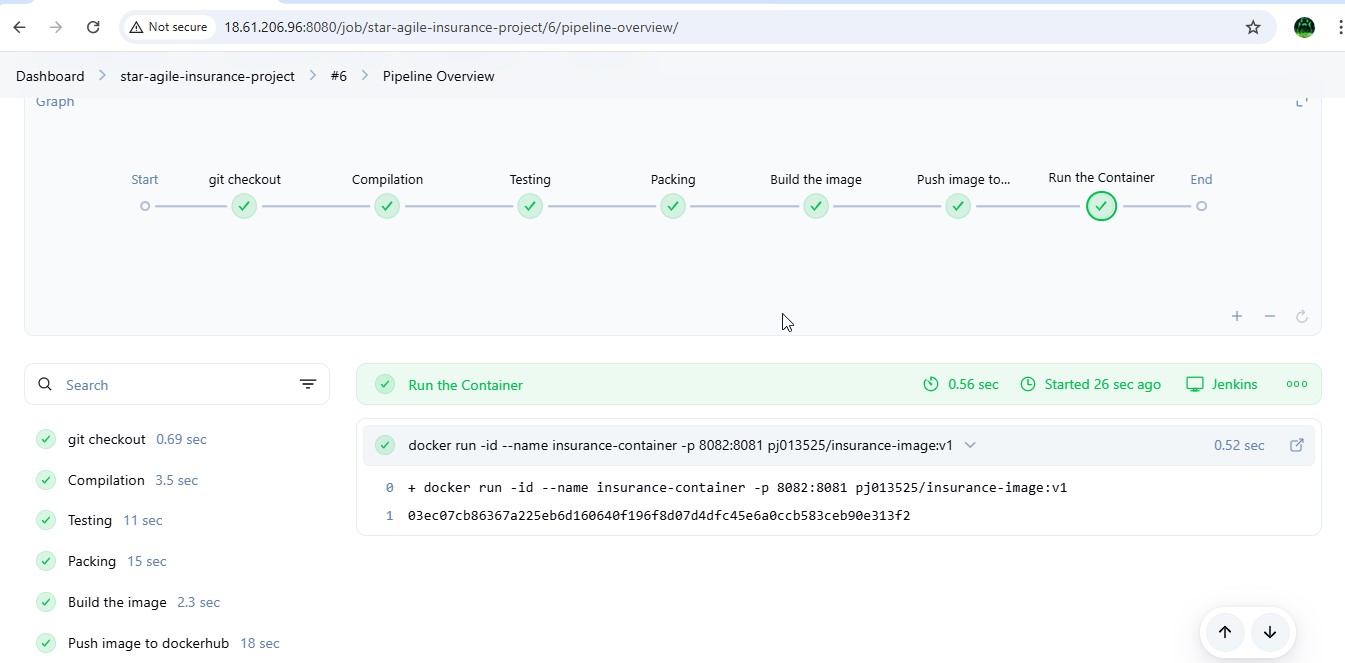


Step25:- Now to perform the pipeline in the Jenkins go to Project==> Pipeline ==> pipeline script and write pipeline using groovy script

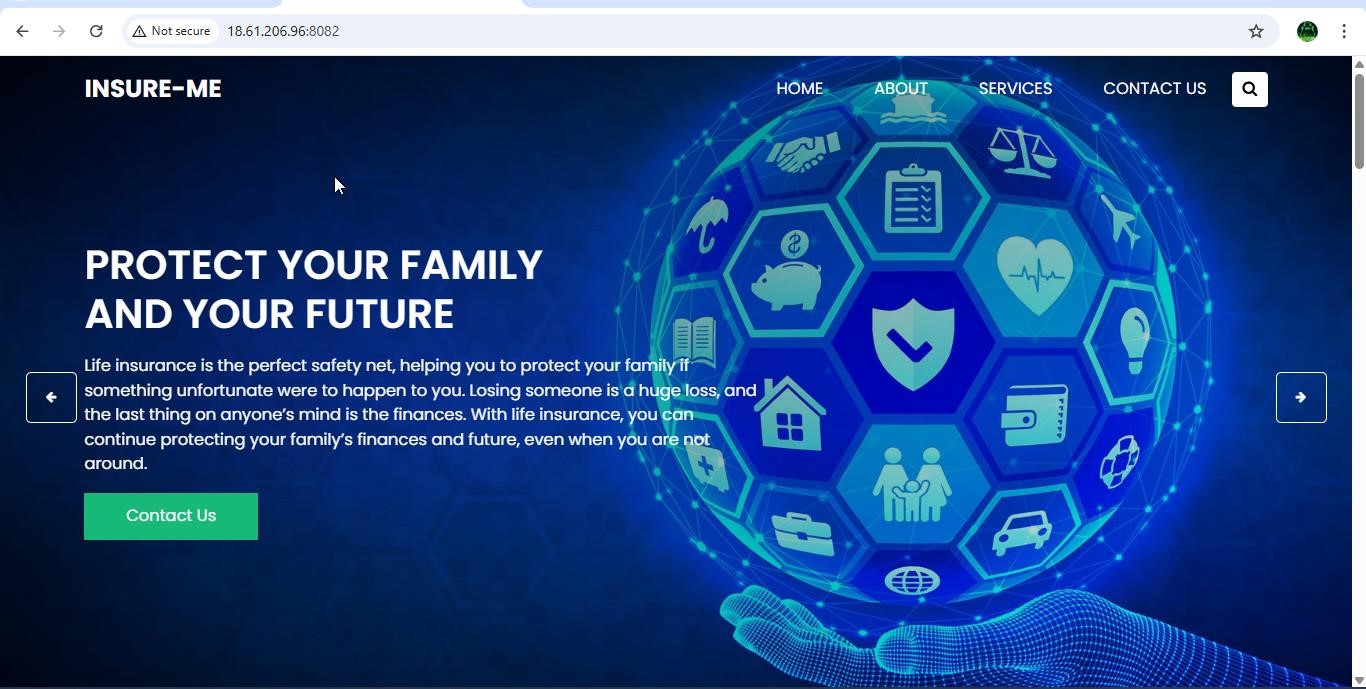


Step26:- Now again go back to Jenkins project and click on Build now to check the status of the build and as you can see that the build is successful and a container also created in the ec2 Target and Master node.





Step27:- Now go to any browser and give the <Target and MasternodeIP:cont-port> and click enter the you will see the home page of the project and thus the project deploy is successful using docker container.

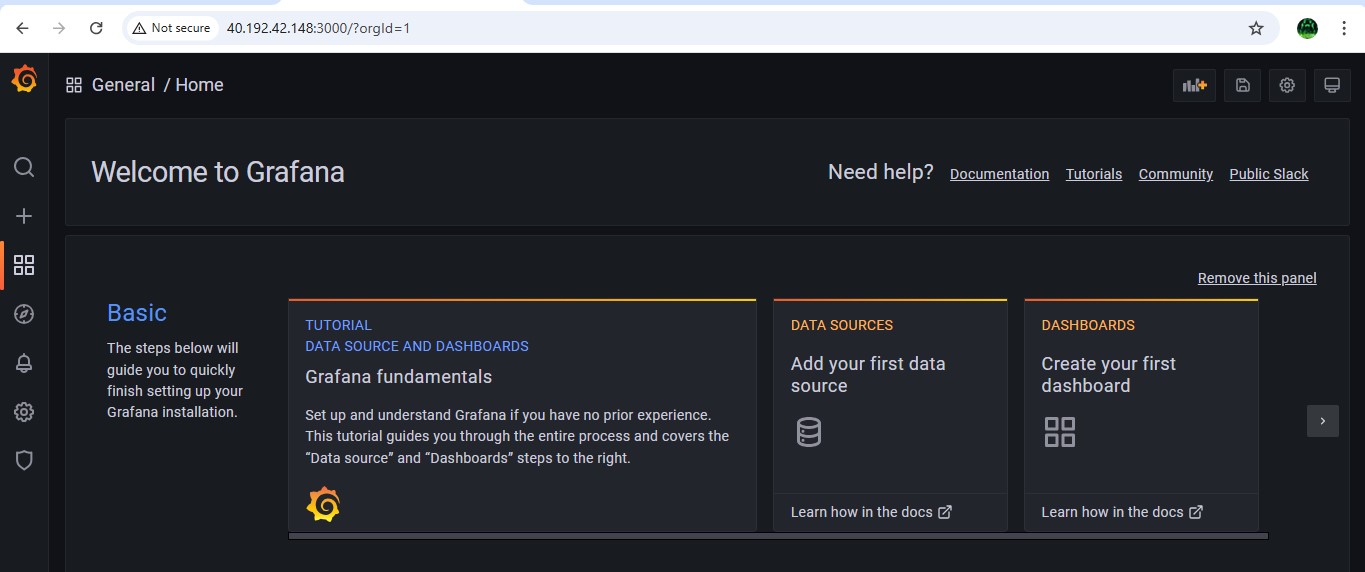
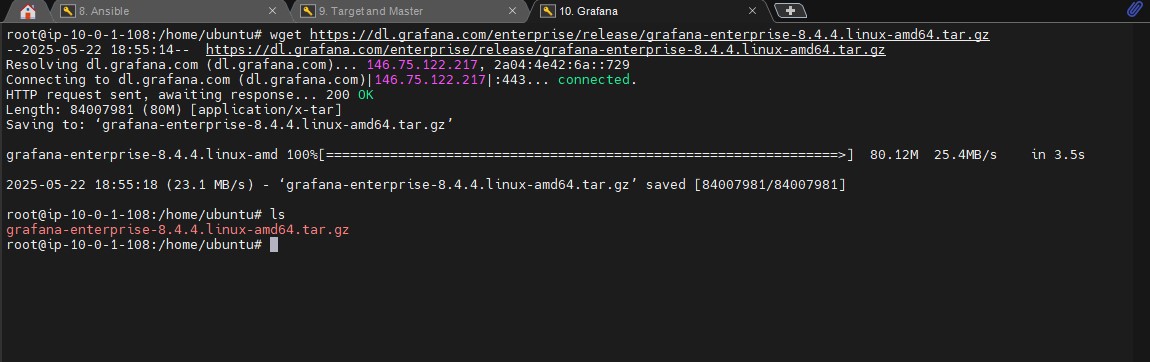


Step28:- Now monitor the containers using Prometheus and

Grafana , for that install Prometheus in Jenkins-Docker server and

Grafana in another server

Step29:- Now install grfana in the Grafana server and after successful installation of Grafana, now go to browser and give grafana server ipaddress: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.



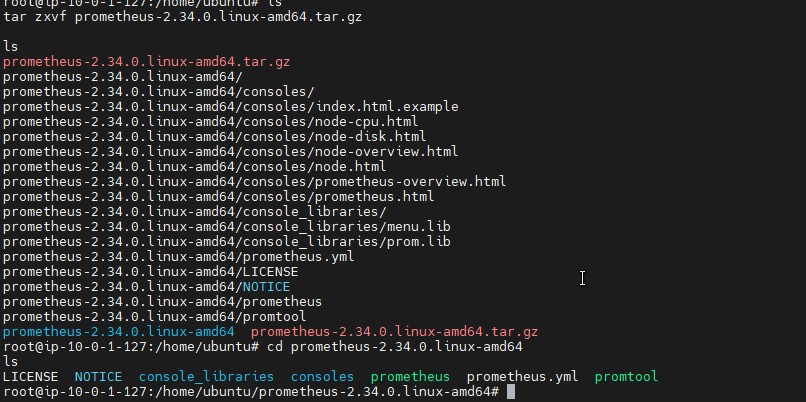
Step30:- Now install prometheus in Target and master node and to login to the prometheus homepage first give metric address in the docker daemon.json vi /etc/docker/daemon.json

{

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

"experimental" : true

}



Step31:- 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

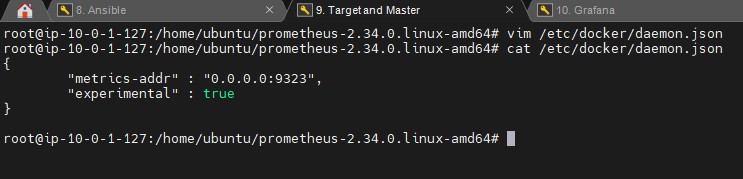
I to insert

{

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

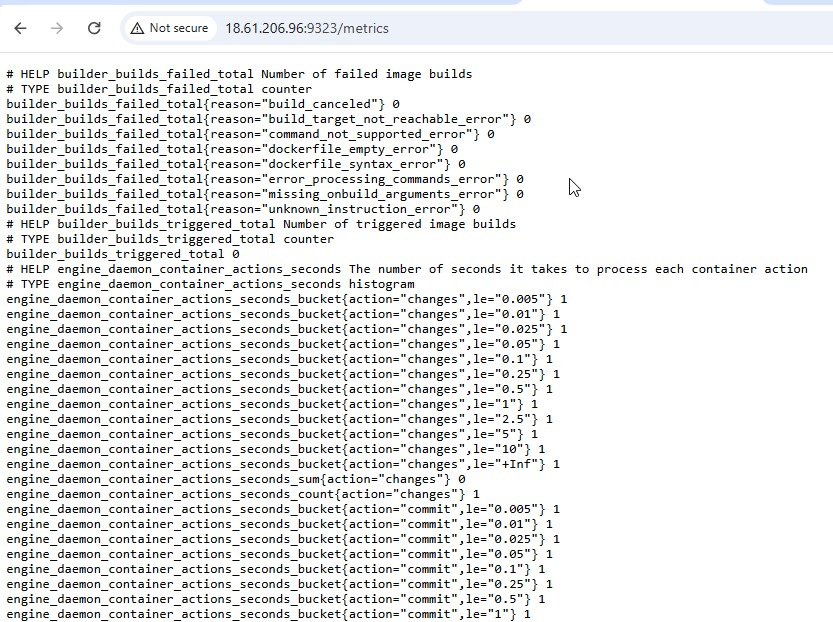
"experimental" : true

} then save and exit and restart the docker



Step32:- 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



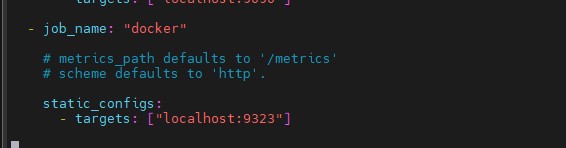
Step33:- Now add docker job in the Prometheus.yml file to give this stats to Prometheus vi prometheus.yml

* job\_name: "docker"

# metrics\_path defaults to '/metrics' # scheme defaults to 'http'. static\_configs:

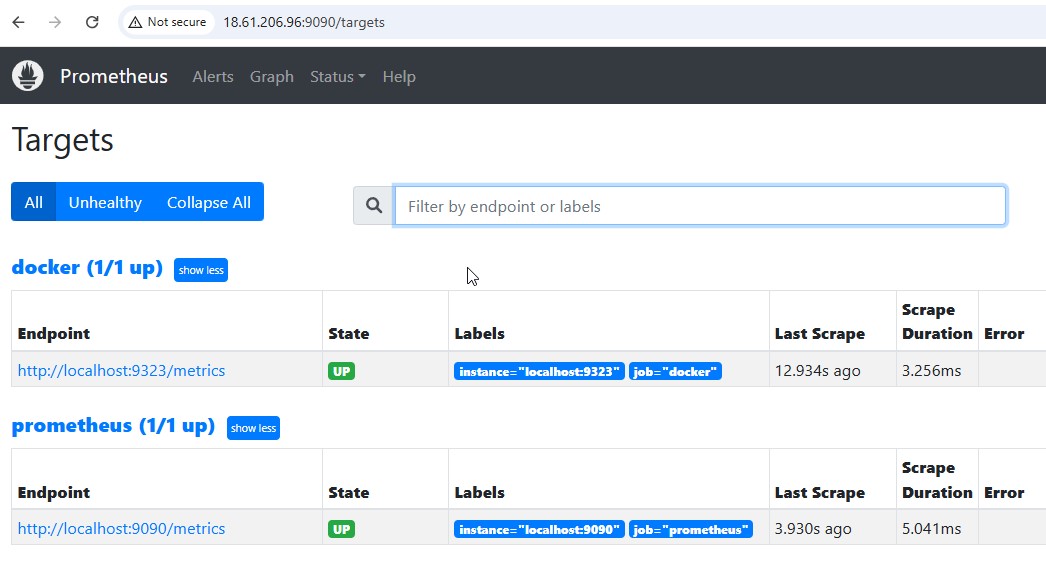
* targets: ["localhost:9323"]

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

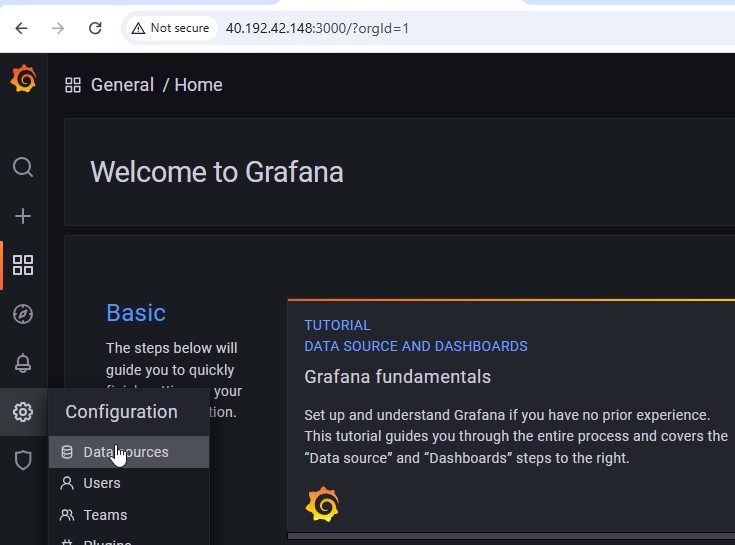


As you can see that the Prometheus have been started from the above image

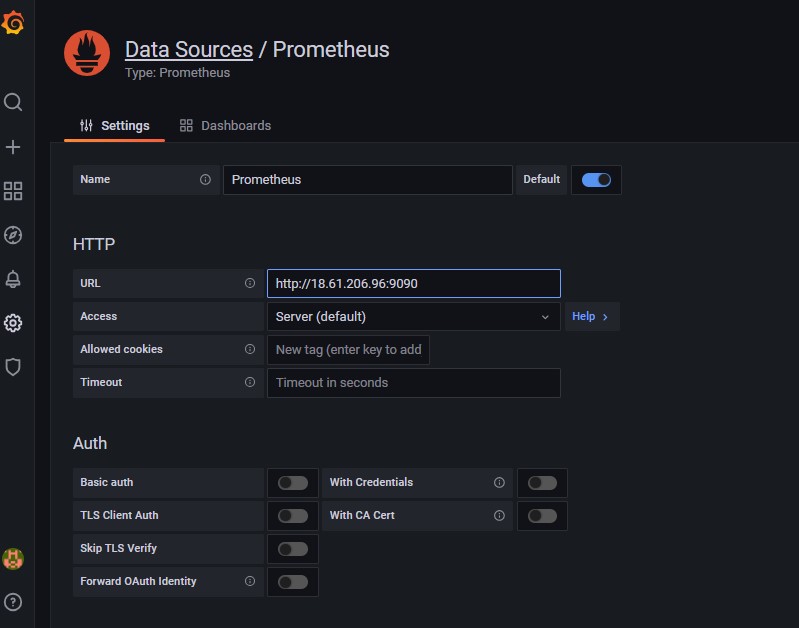
Step34:- Now go browser and give docker ip:9090 and enter , then you will be successfully enter into the Prometheus homepage and click on status  targets then you will see the status of the of the docker and prometheus.



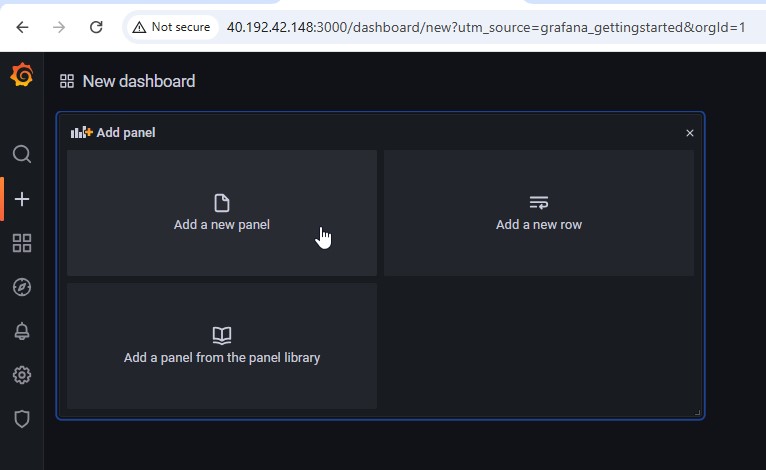
Step35:- Now go to grafana homepage  configurations  Data sources



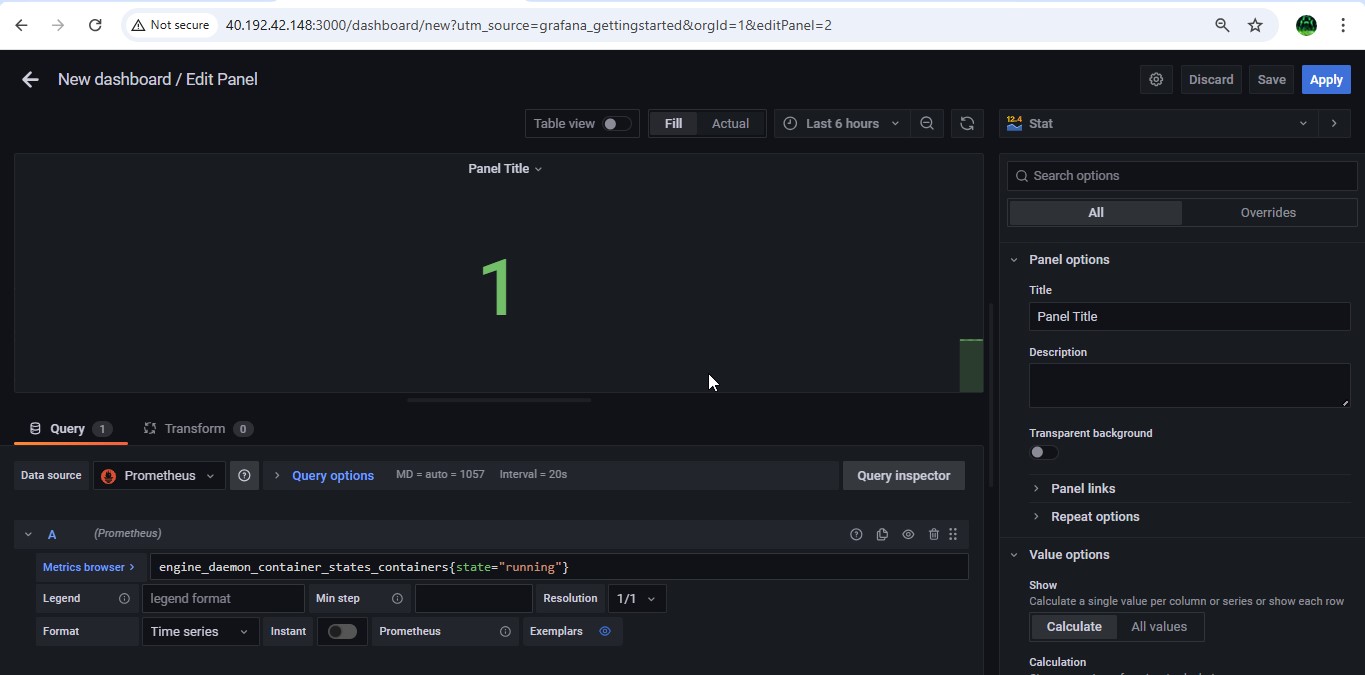
Step36:- Now click on add Data sources  Prometheus and give ipaddress:9090 and click on save and test



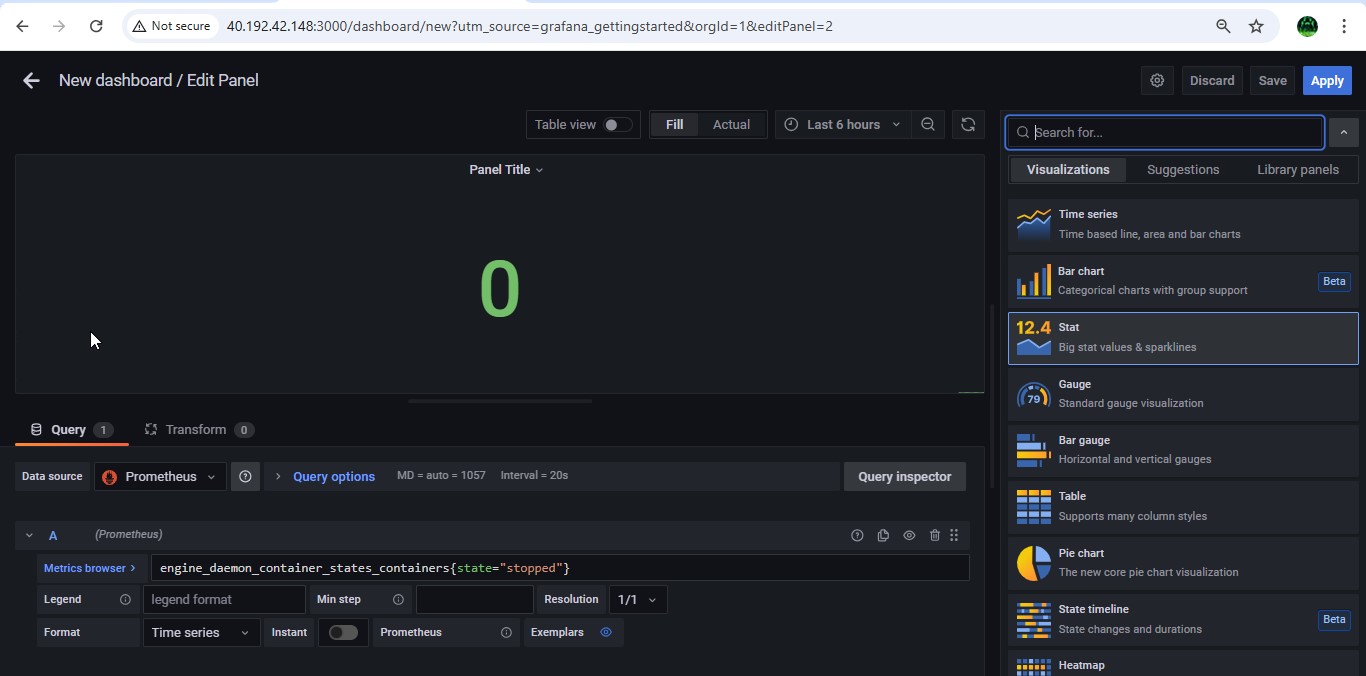
Step37:- Now click on Dash board  add new panel



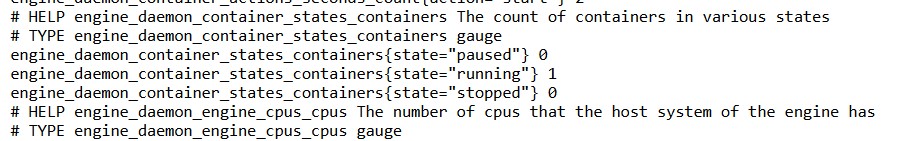
Step38:- Now in the metrics browser give engine daemon container states containers{state="running"} and you will see the result that same as in the metrics from the browser



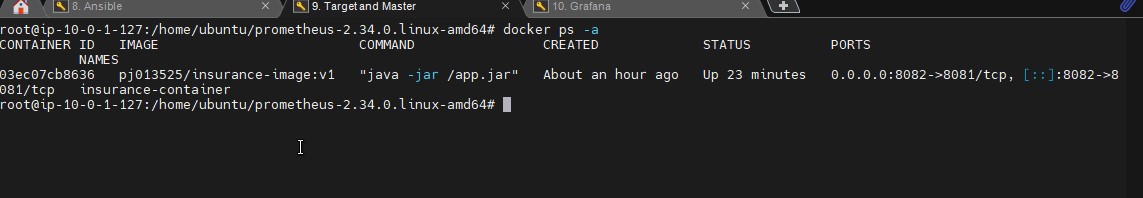
engine\_daemon\_container\_states\_containers{state="stopped"}



Step39:- The values shown in the panel must be equal to the that of shown in the docker stats, here the container which we created is in exited state so it is showing as stopped state in stats



Step40:- Now go and check the containers running or in stopped state again and check the details again in the stats



Step41:- As you can see that the container is in running state and the stats is also shown the same. This is how we monitor the health of a container automatically and visualizing the report using Prometheus and Grafana.