Module - Capstone Project

Insurance Project

Submitted by:- Jonna Padmarao

Submission Date:- 23/05/2025

Resubmission Date:-

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/star-agile-insurance-project.git to get the project code in to that folder

MINGW64:/c/Users/Pj/Desktop/Star-agile-Insureance-Pro

```
Pj@lenovoIP300 MINGW64 ~/Desktop/Star-agile-Insureance-Pro
$ git clone https://github.com/StarAgileDevOpsTraining/star-agile-insurance-project.git
Cloning into 'star-agile-insurance-project'...
remote: Enumerating objects: 160, done.
remote: Counting objects: 100% (28/28), done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 160 (delta 16), reused 16 (delta 16), pack-reused 132 (from 1)
Receiving objects: 100% (160/160), 19.82 MiB | 496.00 KiB/s, done.
Resolving deltas: 100% (47/47), done.

Pj@lenovoIP300 MINGW64 ~/Desktop/Star-agile-Insureance-Pro
$
```

Star-agile-Insureance-Pro › star-agile-insurance-project ›

Name	Date modified	Type	Size
.mvn	5/22/2025 8:49 PM	File folder	
src	5/22/2025 8:49 PM	File folder	
.gitignore	5/22/2025 8:49 PM	Git Ignore Source File	1 KB
ansible-playbook	5/22/2025 8:49 PM	Yaml Source File	1 KB
Dockerfile	5/22/2025 8:49 PM	File	1 KB
Jenkinsfile	5/22/2025 8:49 PM	File	3 KB
mvnw	5/22/2025 8:49 PM	File	11 KB
mvnw	5/22/2025 8:49 PM	Windows Comman...	7 KB
pom	5/22/2025 8:49 PM	XML Document	2 KB
README	5/22/2025 8:49 PM	Markdown Source File	1 KB
selenium-insure-me-runnable	5/22/2025 8:49 PM	Executable Jar File	18,811 KB

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

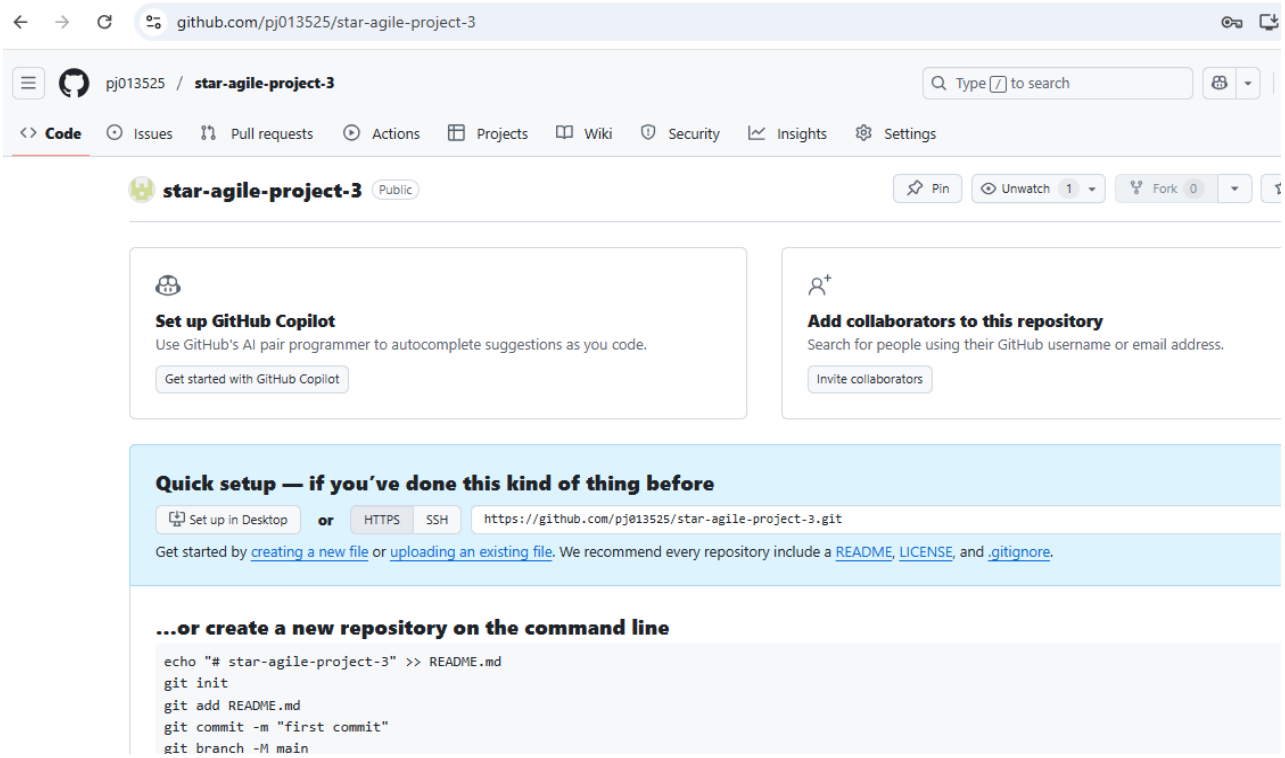
git remote -v --> To get origin list

git remote remove origin ==> to remove the origin

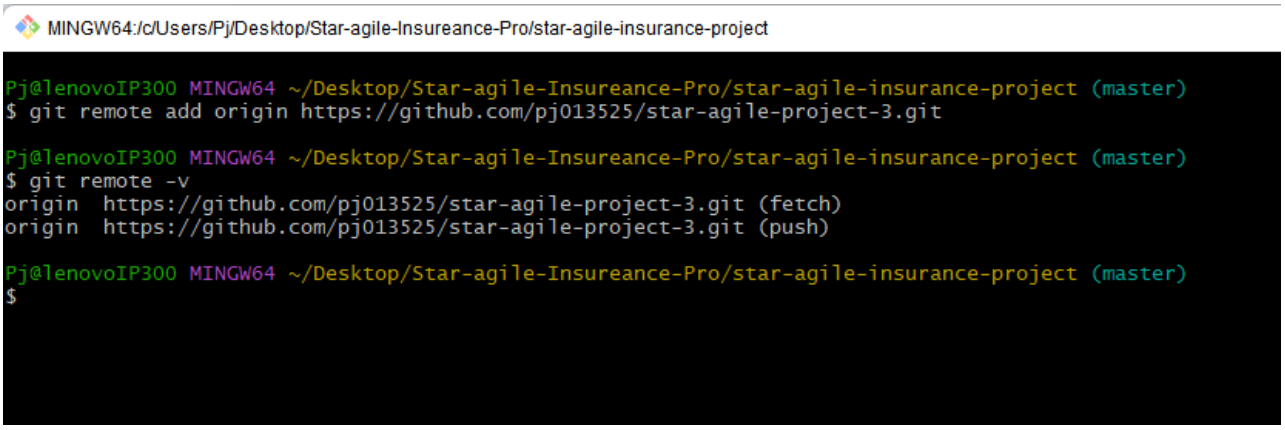
MINGW64:/c/Users/Pj/Desktop/Star-agile-Insureance-Pro/star-agile-insurance-project

```
Pj@lenovoIP300 MINGW64 ~/Desktop/Star-agile-Insureance-Pro/star-agile-insurance-project (master)
$ git remote -v
origin  https://github.com/StarAgileDevOpsTraining/star-agile-insurance-project.git (fetch)
origin  https://github.com/StarAgileDevOpsTraining/star-agile-insurance-project.git (push)

Pj@lenovoIP300 MINGW64 ~/Desktop/Star-agile-Insureance-Pro/star-agile-insurance-project (master)
$ git remote remove origin

Pj@lenovoIP300 MINGW64 ~/Desktop/Star-agile-Insureance-Pro/star-agile-insurance-project (master)
$ git remote -v

Pj@lenovoIP300 MINGW64 ~/Desktop/Star-agile-Insureance-Pro/star-agile-insurance-project (master)
$
```

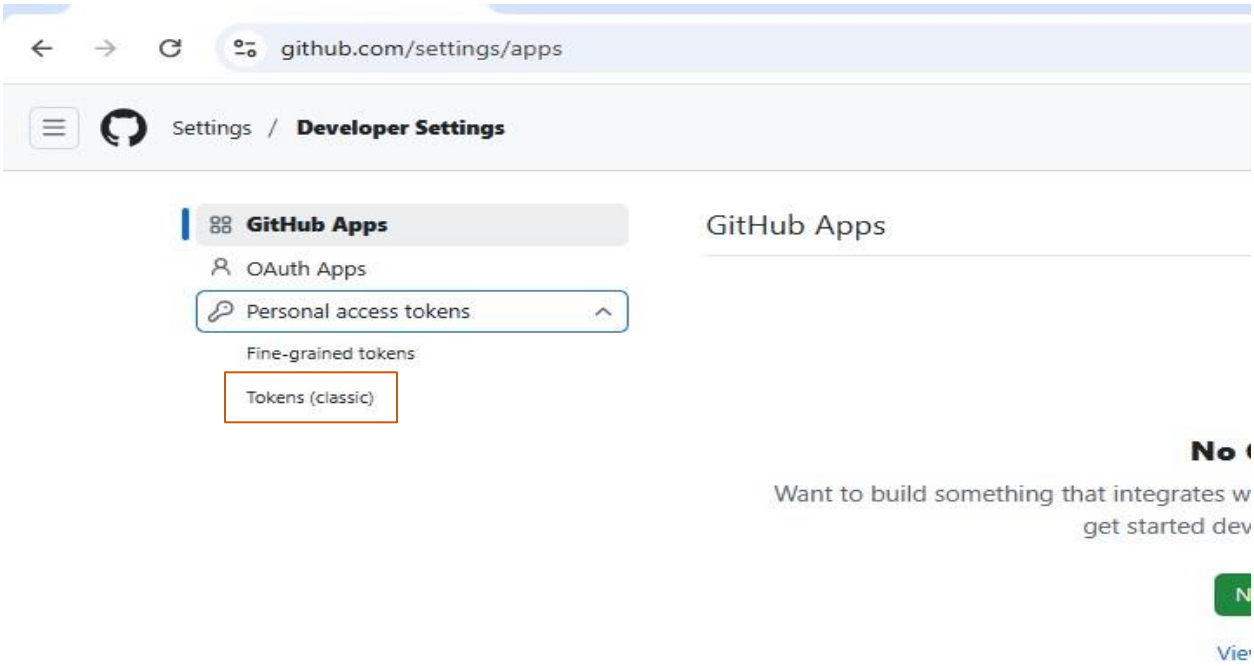
Step4:- Now go to github and create a new repo and copy the url in the gitbash



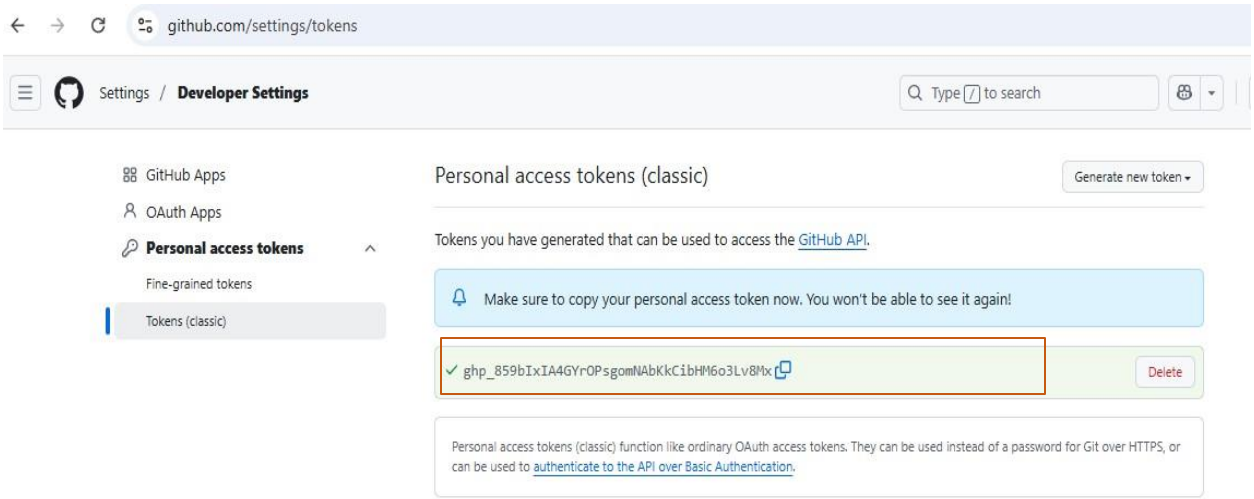
Step5:- Now again go to the gitbash and add this git repo url in the project by using git remote add origin <git-repo-url> and verify



Step6:- Now again go to github ❼ Profile setting ❼Developer settings ❼ Personal access token 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

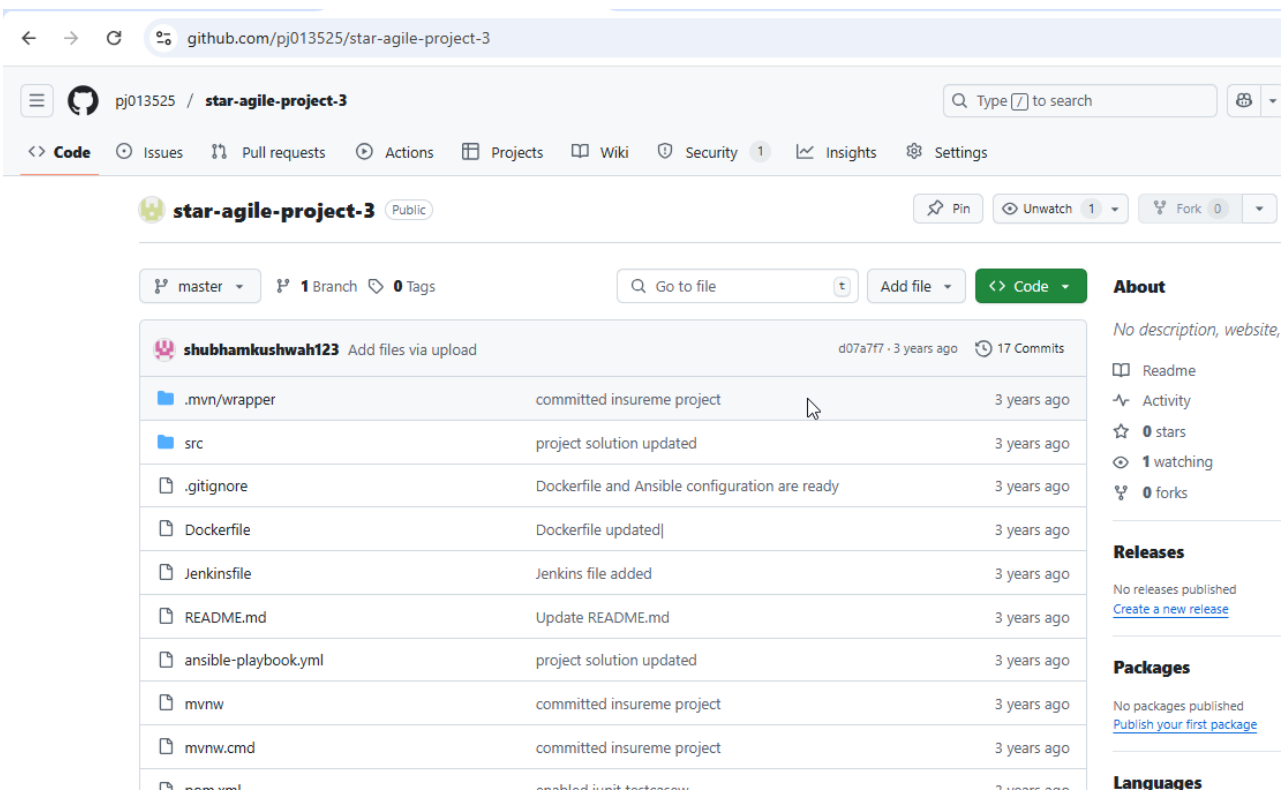


Step8:- Now give link this the remote repo with gitbash using this token git push -u origin master and paste the token the copied from the github and press sign in

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

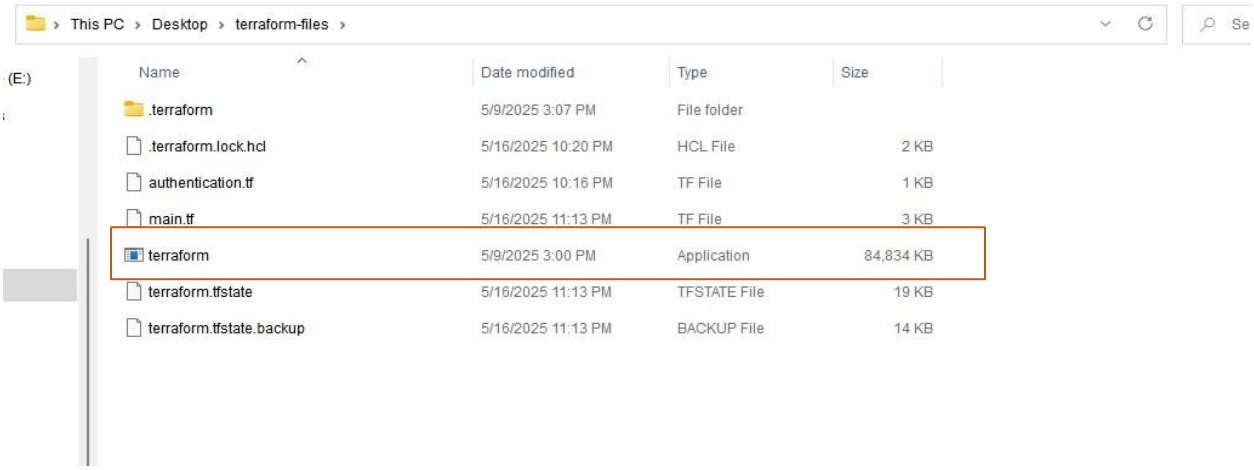
```
MINGW64:/c/Users/Pj/Desktop/Star-agile-Insureance-Pro/star-agile-insurance-project

Pj@lenovoIP300 MINGW64 ~/Desktop/Star-agile-Insureance-Pro/star-agile-insurance-project (master)
$ git push -u origin master
Enumerating objects: 153, done.
Counting objects: 100% (153/153), done.
Delta compression using up to 4 threads
Compressing objects: 100% (86/86), done.
Writing objects: 100% (153/153), 19.82 MiB | 3.61 MiB/s, done.
Total 153 (delta 44), reused 153 (delta 44), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (44/44), done.
To https://github.com/pj013525/star-agile-project-3.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

Pj@lenovoIP300 MINGW64 ~/Desktop/Star-agile-Insureance-Pro/star-agile-insurance-project (master)
$ |
```

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

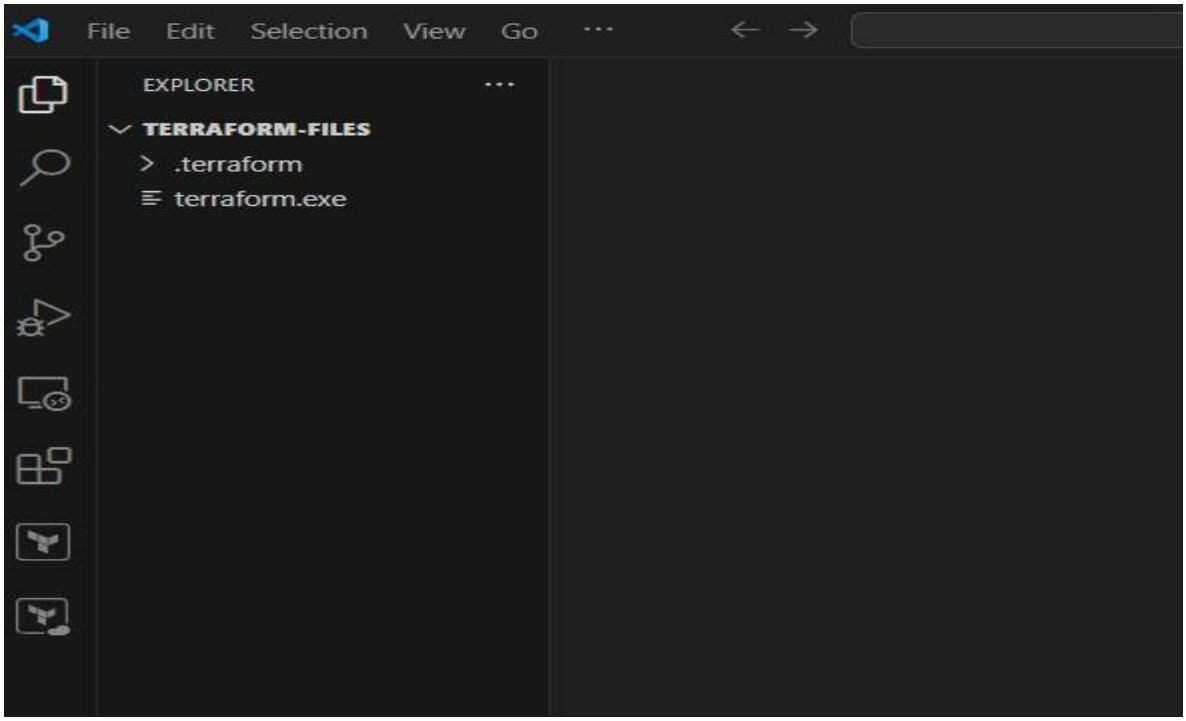


Step11:- Now create an instance using terraform as Iaac , and for that create a folder on desktop and go to browser download terraform for

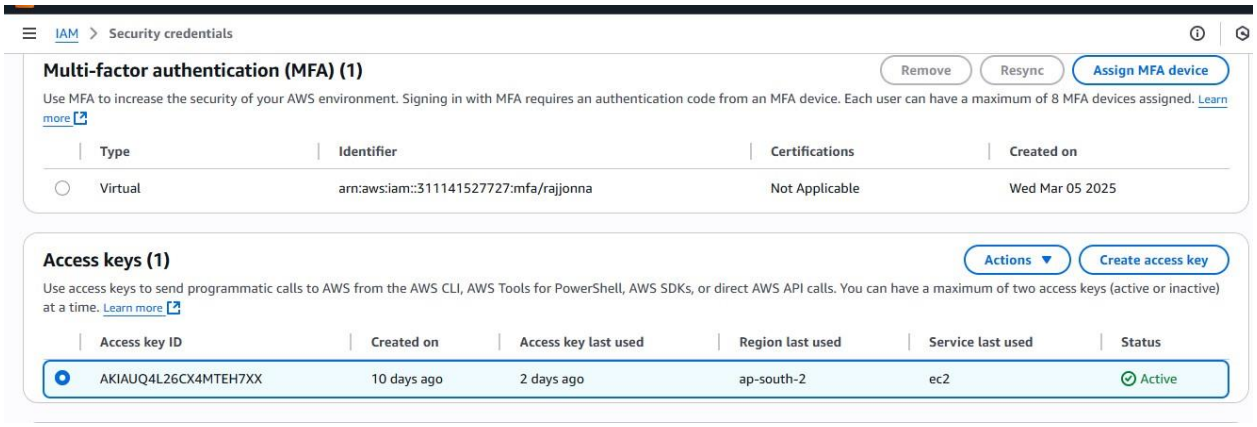
windows then a terraform application will be generated , now copy this application in to that folder and save



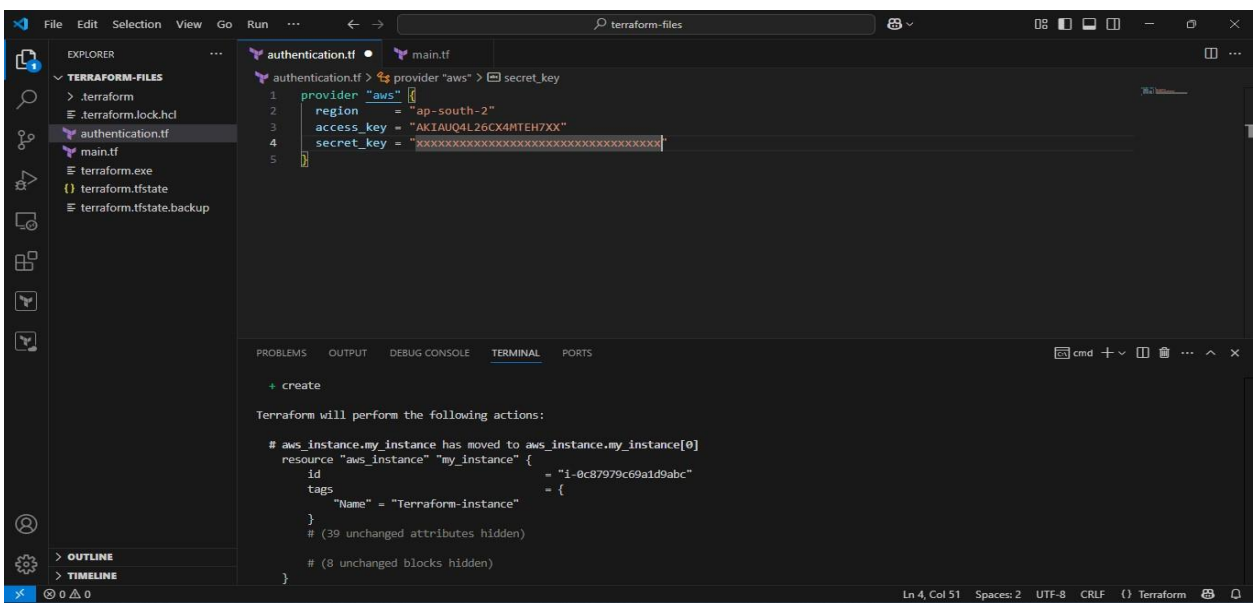
Step12:- open visual studio code and go to terraform folder



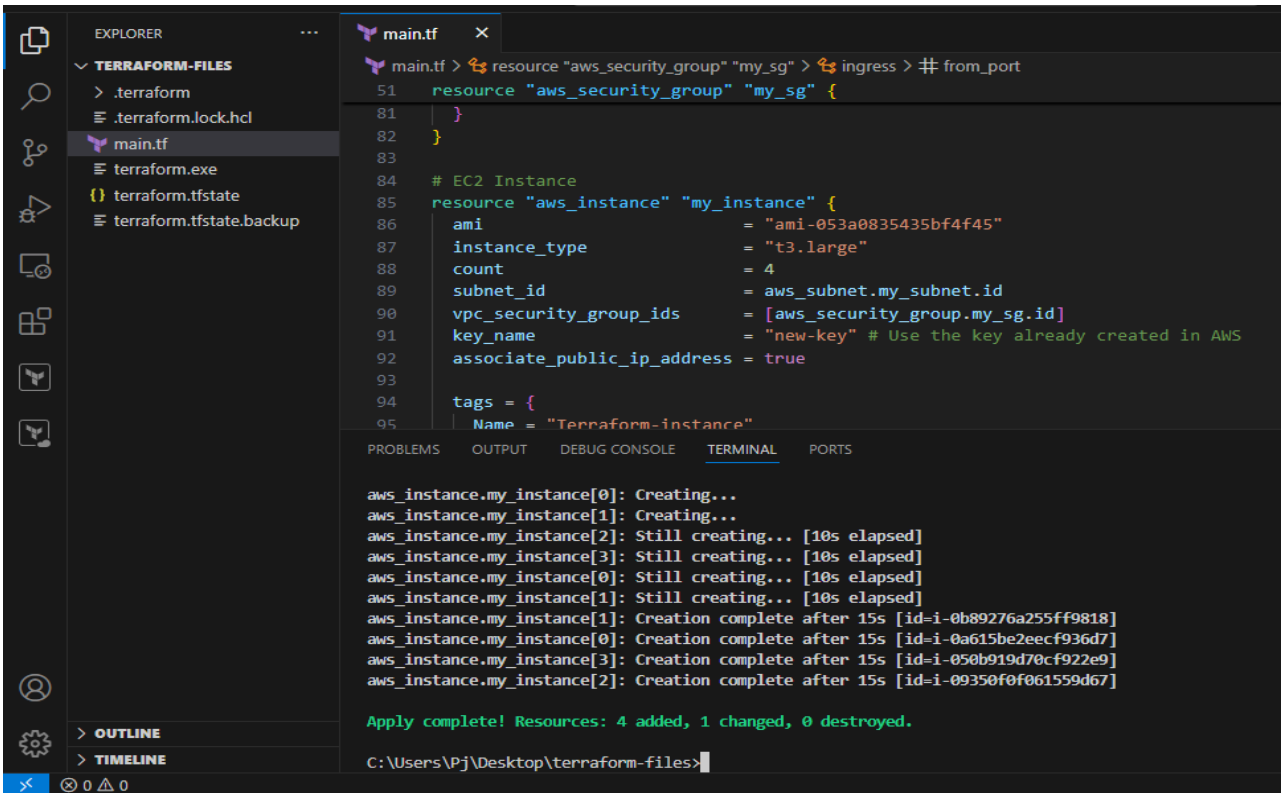
Step13:- Now create a file authentication.tf and give the provider and for that select the region in which you want to launch the server and go to aws account and go to profile ❼ credentials and go to access keys



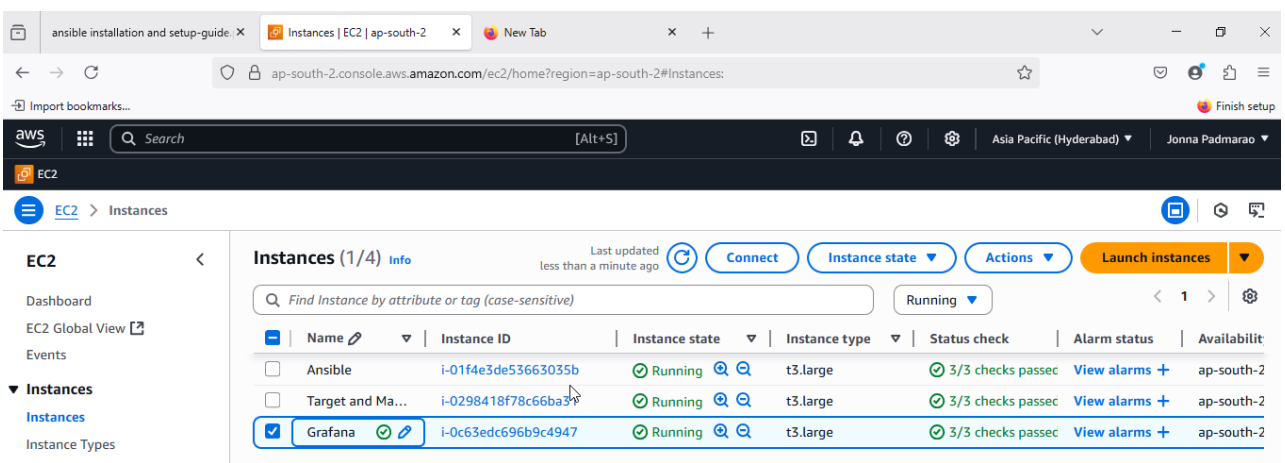
Step14:- now copy this access key details and paste it in this authentication.tf file and initialize it



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 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

```
8. Ansible    9. Target and Master    10. Worker
root@ip-10-0-1-135:/home/ubuntu#
```

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

```
8. Ansible    9. Target and Master    10. Worker
root@ip-10-0-1-135:/home/ubuntu#  ansible --version
ansible [core 2.18.6]
  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 = 3.12.3 (main, Feb  4 2025, 14:48:35) [GCC 13.3.0] (/usr/bin/python3)
  jinja version = 3.1.2
  libyaml = True
root@ip-10-0-1-135:/home/ubuntu#
```

```
8. Ansible    9. Target and Master    10. Worker
root@ip-10-0-1-135:/home/ubuntu# ansible all -m ping
[WARNING]: Platform linux on host 10.0.1.127 is using the discovered Python interpreter at /usr/bin/python3.12, but future installation of
another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-
core/2.18/reference_appendices/interpreter_discovery.html for more information.
10.0.1.127 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3.12"
    },
    "changed": false,
    "ping": "pong"
}
root@ip-10-0-1-135:/home/ubuntu#
```

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

```
8. Ansible    9. Target and Master    10. Worker
      apt:
        name: openjdk-17-jdk
        state: present

    - name: Install Maven
      apt:
        name: maven
        state: present

    - name: Install Git
      apt:
        name: git
        state: present

    - name: Add Docker GPG key
      apt_key:
        url: https://download.docker.com/linux/ubuntu/gpg
        state: present

    - name: Add Docker repository
      apt_repository:
        repo: deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable
        state: present

    - name: Install Docker
      apt:
        name:
          - docker-ce
          - docker-ce-cli
          - containerd.io
        state: present
        update_cache: yes
    - name: Enable and start Docker service
      systemd:
        name: docker
        enabled: true
        state: started
root@ip-10-0-1-135:/home/ubuntu#
```

```
8. Ansible    9. Target and Master    10. Worker
---
- name: Install Jenkins on target node (Ubuntu 20.04+)
  hosts: targets
  become: yes
  tasks:

    - name: Update apt cache
      apt:
        update_cache: yes

    - name: Download Jenkins GPG key
      ansible.builtin.get_url:
        url: https://pkg.jenkins.io/debian-stable/jenkins.io.key
        dest: /usr/share/keyrings/jenkins-keyring.asc
        mode: '0644'

    - name: Add Jenkins repository with signed-by
      ansible.builtin.copy:
        dest: /etc/apt/sources.list.d/jenkins.list
        content: |
          deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/
        mode: '0644'

    - name: Update apt cache again after adding Jenkins repo
      apt:
        update_cache: yes

    - name: Install Jenkins
      apt:
        name: jenkins
        state: present

    - name: Ensure Jenkins is started and enabled
      systemd:
        name: jenkins
        state: started
        enabled: yes
```

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

8. Ansible | 9. Target and Master | 10. Worker

```
TASK [Gathering Facts] ***********************************************************************************
[WARNING]: Platform linux on host 10.0.1.127 is using the discovered Python interpreter at /usr/bin/python3.12, but fut
another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-
core/2.18/reference_appendices/interpreter_discovery.html for more information.
ok: [10.0.1.127]

TASK [Update apt cache] **********************************************************************************
changed: [10.0.1.127]

TASK [Install dependencies] ******************************************************************************
changed: [10.0.1.127]

TASK [Install Java 17] ***********************************************************************************
changed: [10.0.1.127]

TASK [Install Maven] *************************************************************************************
changed: [10.0.1.127]

TASK [Install Git] ***************************************************************************************
ok: [10.0.1.127]

TASK [Add Docker GPG key] ********************************************************************************
changed: [10.0.1.127]

TASK [Add Docker repository] *****************************************************************************
changed: [10.0.1.127]

TASK [Install Docker] ************************************************************************************
changed: [10.0.1.127]

TASK [Enable and start Docker service] *******************************************************************
ok: [10.0.1.127]

PLAY RECAP ***********************************************************************************************
10.0.1.127                 : ok=10   changed=7    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

root@ip-10-0-1-135:/home/ubuntu#
```

8. Ansible | 9. Target and Master | 10. Worker

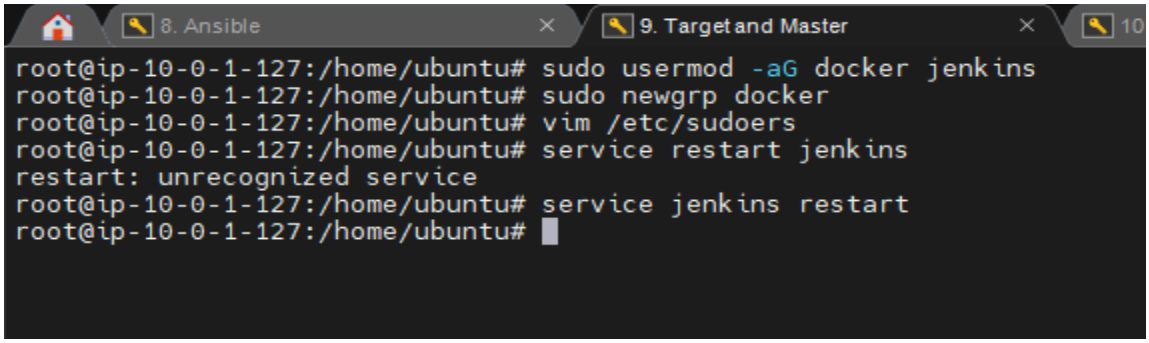
```
root@ip-10-0-1-135:/home/ubuntu# ansible-playbook jenkins.yml

PLAY [Install Jenkins on target node (Ubuntu 20.04+)] ****************************************************

TASK [Gathering Facts] ***********************************************************************************
[WARNING]: Platform linux on host 10.0.1.127 is using the discovered Python interpreter at /usr/bin/python3.12, but future installation of
another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-
core/2.18/reference_appendices/interpreter_discovery.html for more information.
ok: [10.0.1.127]

TASK [Update apt cache] **********************************************************************************
changed: [10.0.1.127]

TASK [Download Jenkins GPG key] **************************************************************************
ok: [10.0.1.127]

TASK [Add Jenkins repository with signed-by] *************************************************************
changed: [10.0.1.127]

TASK [Update apt cache again after adding Jenkins repo] **************************************************
changed: [10.0.1.127]

TASK [Install Jenkins] ***********************************************************************************
changed: [10.0.1.127]

TASK [Ensure Jenkins is started and enabled] *************************************************************
ok: [10.0.1.127]

PLAY RECAP ***********************************************************************************************
10.0.1.127                 : ok=7    changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

root@ip-10-0-1-135:/home/ubuntu#
```

8. Ansible | 9. Target and Master | 10. Worker

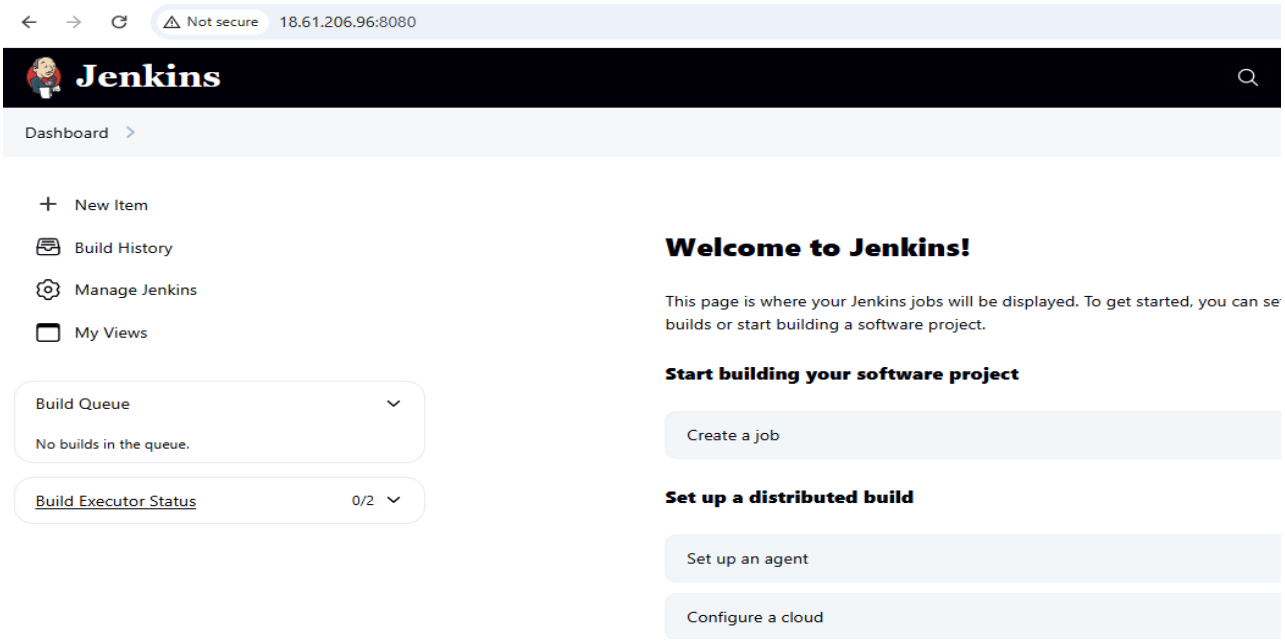
```
root@ip-10-0-1-127:/home/ubuntu# jenkins --version
2.504.1
root@ip-10-0-1-127:/home/ubuntu# docker --version
Docker version 28.1.1, build 4eba377
root@ip-10-0-1-127:/home/ubuntu# git --version
git version 2.43.0
root@ip-10-0-1-127:/home/ubuntu# mvn --version
Apache Maven 3.8.7
Maven home: /usr/share/maven
Java version: 17.0.15, vendor: Ubuntu, runtime: /usr/lib/jvm/java-17-openjdk-amd64
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "6.8.0-1024-aws", arch: "amd64", family: "unix"
root@ip-10-0-1-127:/home/ubuntu# java --version
openjdk 17.0.15 2025-04-15
OpenJDK Runtime Environment (build 17.0.15+6-Ubuntu-0ubuntu124.04)
OpenJDK 64-Bit Server VM (build 17.0.15+6-Ubuntu-0ubuntu124.04, mixed mode, sharing)
root@ip-10-0-1-127:/home/ubuntu#
```

Step21:- Now add Jenkins group to docker and give root permissions to the Jenkins user in the sudoers file as under root give

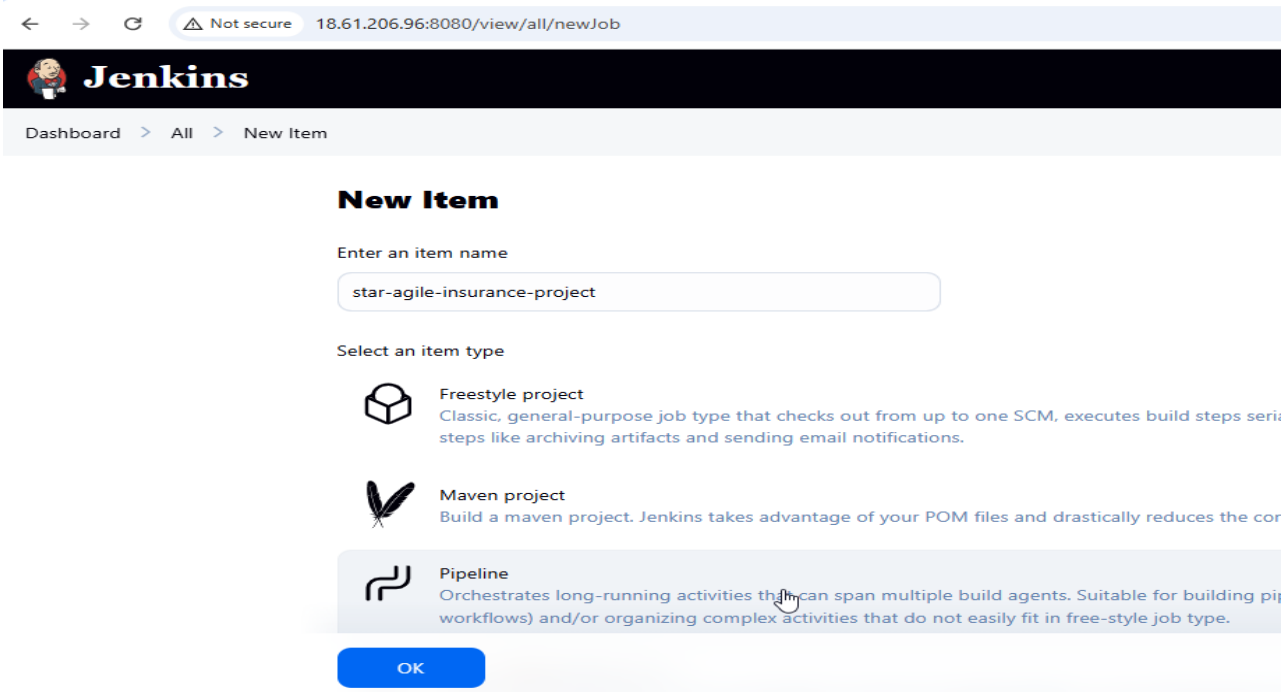
jenkins ALL=(ALL:ALL) NOAPSSWD: ALL restart the jenkins



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

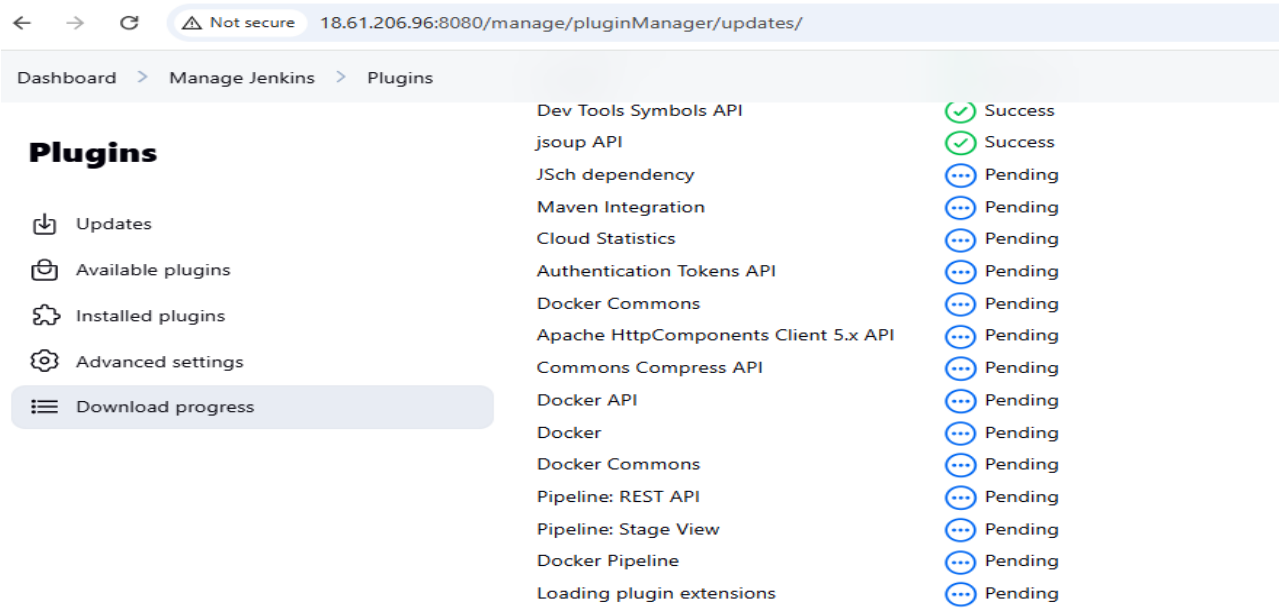
Git Plugin

Docker Commons Plugin

Docker Pipeline Plugin

Pipeline: GitHub

Credentials Binding Plugin

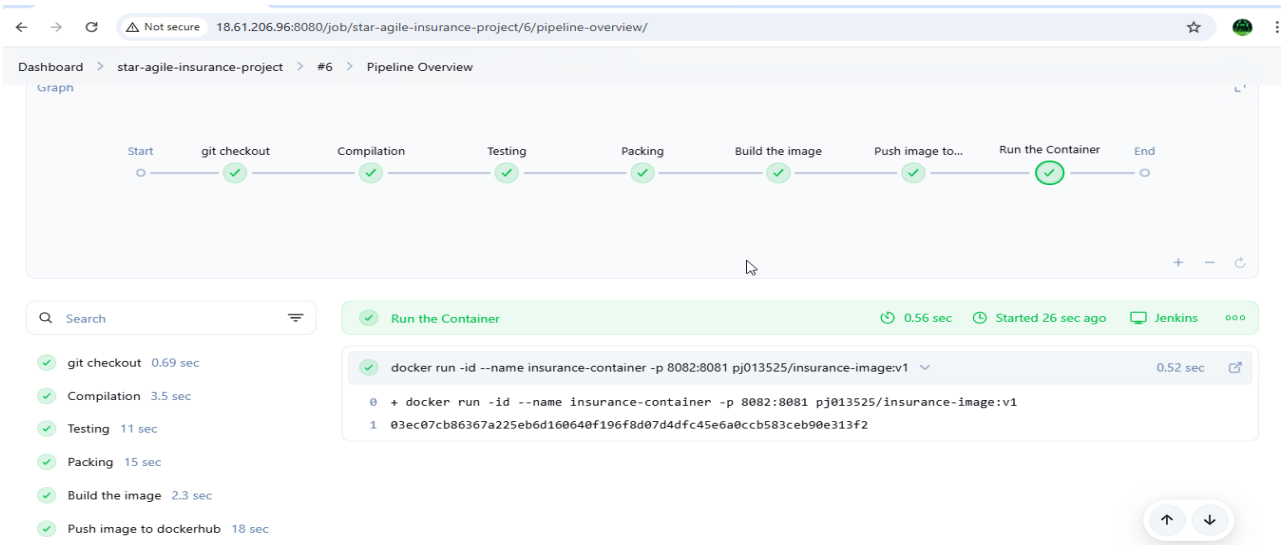


Step25:- Now to perform the pipeline in the Jenkins go to Project==> Pipeline ==> pipeline script and write pipeline using groovy script

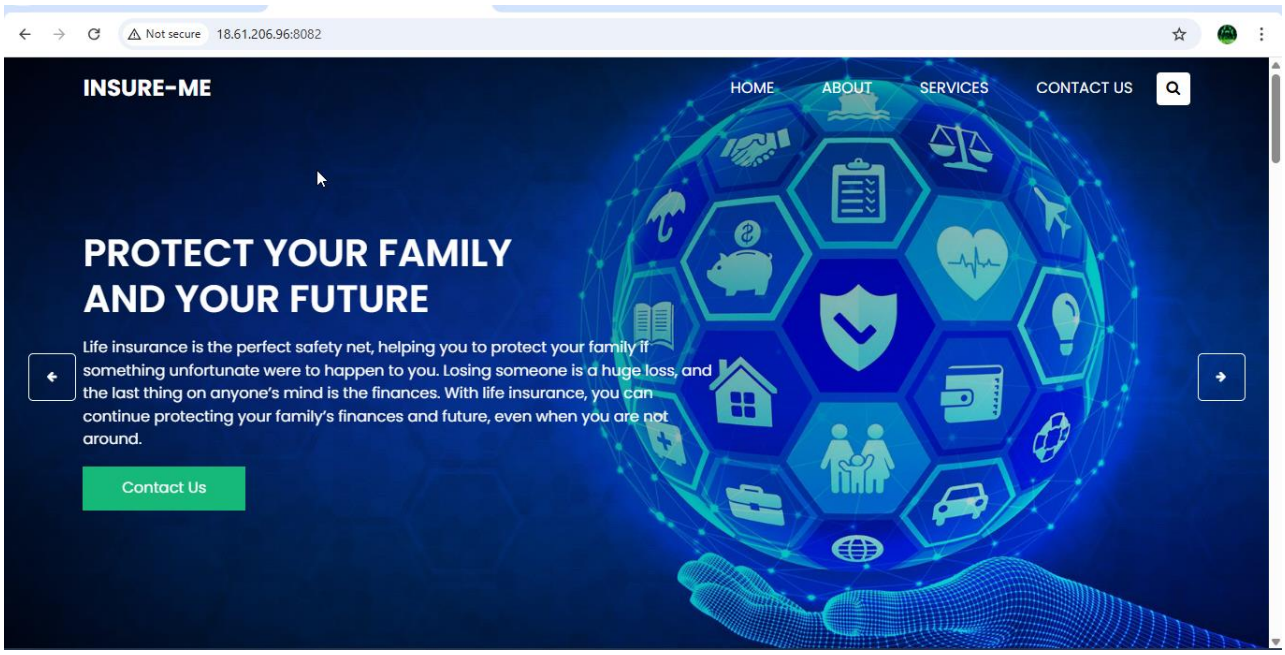
```
pipeline {
  agent any

  stages {
    stage('git checkout') {
      steps {
        git 'https://github.com/pj013525/star-agile-project-3.git'
      }
    }
    stage('Compilation') {
      steps {
        sh 'mvn compile'
      }
    }
    stage('Testing') {
      steps {
        sh 'mvn test'
      }
    }
    stage('Packing ') {
      steps {
        sh 'mvn clean package'
      }
    }
    stage('Build the image ') {
      steps {
        sh 'docker build -t pj013525/insurance-image:v1 .'
        sh 'docker images'
      }
    }
    stage('Push image to dockerhub') {
      steps {
        withCredentials([string(credentialsId: 'dockerhub-details', variable: 'dockerhub_pwd')]) {
        sh 'echo "${dockerhub_pwd}" | docker login -u pj013525 -p ${dockerhub_pwd}'
        sh 'docker push pj013525/insurance-image:v1'
        }
      }
    }
    stage('Run the Container') {
      steps {
        sh 'docker run -id --name insurance-container -p 8082:8081 pj013525/insurance-image:v1'
      }
    }
```

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.

```
root@ip-10-0-1-127:/home/ubuntu# docker images
REPOSITORY                TAG       IMAGE ID       CREATED          SIZE
pj013525/insurance-image   v1        38cce63f6b10   13 minutes ago   695MB
root@ip-10-0-1-127:/home/ubuntu# docker ps -a
CONTAINER ID   IMAGE                        COMMAND                  CREATED          STATUS          PORTS
       NAMES
03ec07cb8636   pj013525/insurance-image:v1   "java -jar /app.jar"   13 minutes ago   Up 13 minutes   0.0.0.0:8082->8081/tcp, [::]:8082->8081
/tcp   insurance-container
root@ip-10-0-1-127:/home/ubuntu#
```

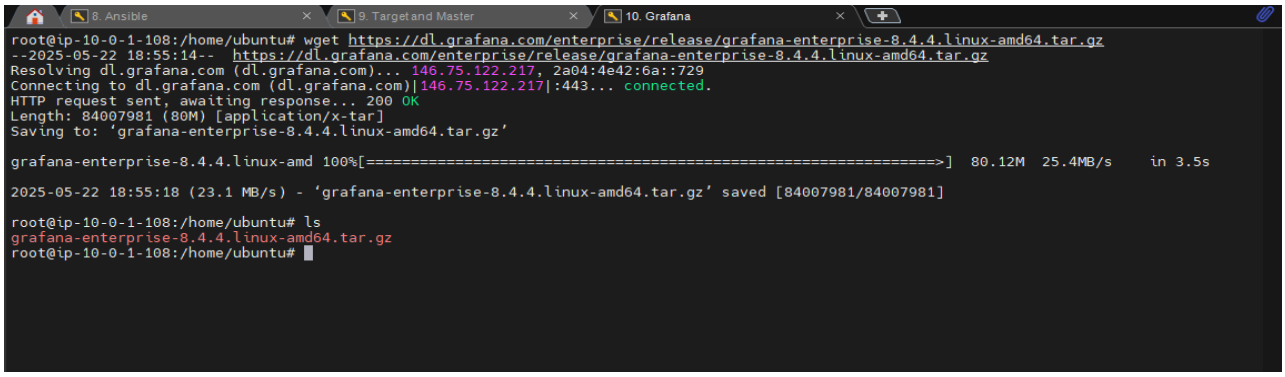


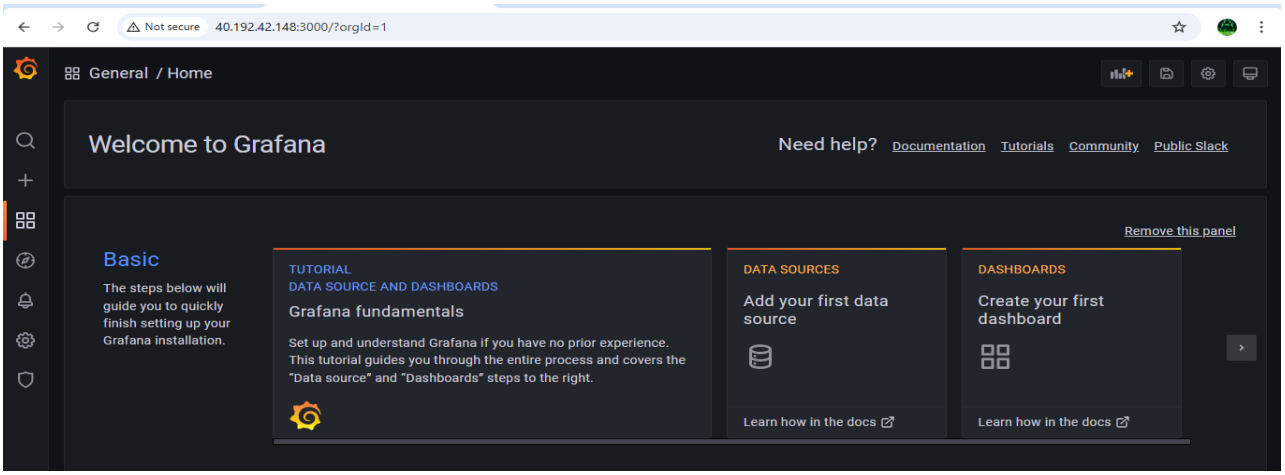
Step27:- Now go to any browser and give the <Target and Masternode-IP:cont-port> and click enter the you will see the home page of the project and thus the project deploy is successful using docker container.



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 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.





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
}
```

```
root@ip-10-0-1-127:/home/ubuntu# ls
tar zxvf prometheus-2.34.0.linux-amd64.tar.gz

ls
prometheus-2.34.0.linux-amd64.tar.gz
prometheus-2.34.0.linux-amd64/
prometheus-2.34.0.linux-amd64/consoles/
prometheus-2.34.0.linux-amd64/consoles/index.html.example
prometheus-2.34.0.linux-amd64/consoles/node-cpu.html
prometheus-2.34.0.linux-amd64/consoles/node-disk.html
prometheus-2.34.0.linux-amd64/consoles/node-overview.html
prometheus-2.34.0.linux-amd64/consoles/node.html
prometheus-2.34.0.linux-amd64/consoles/prometheus-overview.html
prometheus-2.34.0.linux-amd64/consoles/prometheus.html
prometheus-2.34.0.linux-amd64/console_libraries/
prometheus-2.34.0.linux-amd64/console_libraries/menu.lib
prometheus-2.34.0.linux-amd64/console_libraries/prom.lib
prometheus-2.34.0.linux-amd64/prometheus.yml
prometheus-2.34.0.linux-amd64/LICENSE
prometheus-2.34.0.linux-amd64/NOTICE
prometheus-2.34.0.linux-amd64/prometheus
prometheus-2.34.0.linux-amd64/promtool
prometheus-2.34.0.linux-amd64  prometheus-2.34.0.linux-amd64.tar.gz
root@ip-10-0-1-127:/home/ubuntu# cd prometheus-2.34.0.linux-amd64
ls
LICENSE  NOTICE  console_libraries  consoles  prometheus  prometheus.yml  promtool
root@ip-10-0-1-127:/home/ubuntu/prometheus-2.34.0.linux-amd64#
```

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

```
8. Ansible    9. Target and Master    10. Grafana
root@ip-10-0-1-127:/home/ubuntu/prometheus-2.34.0.linux-amd64# vim /etc/docker/daemon.json
root@ip-10-0-1-127:/home/ubuntu/prometheus-2.34.0.linux-amd64# cat /etc/docker/daemon.json
{
        "metrics-addr" : "0.0.0.0:9323",
        "experimental" : true
}

root@ip-10-0-1-127:/home/ubuntu/prometheus-2.34.0.linux-amd64#
```

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

Not secure 18.61.206.96:9323/metrics

```
# HELP builder_builds_failed_total Number of failed image builds
# TYPE builder_builds_failed_total counter
builder_builds_failed_total{reason="build_canceled"} 0
builder_builds_failed_total{reason="build_target_not_reachable_error"} 0
builder_builds_failed_total{reason="command_not_supported_error"} 0
builder_builds_failed_total{reason="dockerfile_empty_error"} 0
builder_builds_failed_total{reason="dockerfile_syntax_error"} 0
builder_builds_failed_total{reason="error_processing_commands_error"} 0
builder_builds_failed_total{reason="missing_onbuild_arguments_error"} 0
builder_builds_failed_total{reason="unknown_instruction_error"} 0
# HELP builder_builds_triggered_total Number of triggered image builds
# TYPE builder_builds_triggered_total counter
builder_builds_triggered_total 0
# HELP engine_daemon_container_actions_seconds The number of seconds it takes to process each container action
# TYPE engine_daemon_container_actions_seconds histogram
engine_daemon_container_actions_seconds_bucket{action="changes",le="0.005"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="0.01"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="0.025"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="0.05"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="0.1"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="0.25"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="0.5"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="1"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="2.5"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="5"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="10"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="+Inf"} 1
engine_daemon_container_actions_seconds_sum{action="changes"} 0
engine_daemon_container_actions_seconds_count{action="changes"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="0.005"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="0.01"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="0.025"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="0.05"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="0.1"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="0.25"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="0.5"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="1"} 1
```

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

- job_name: "docker"

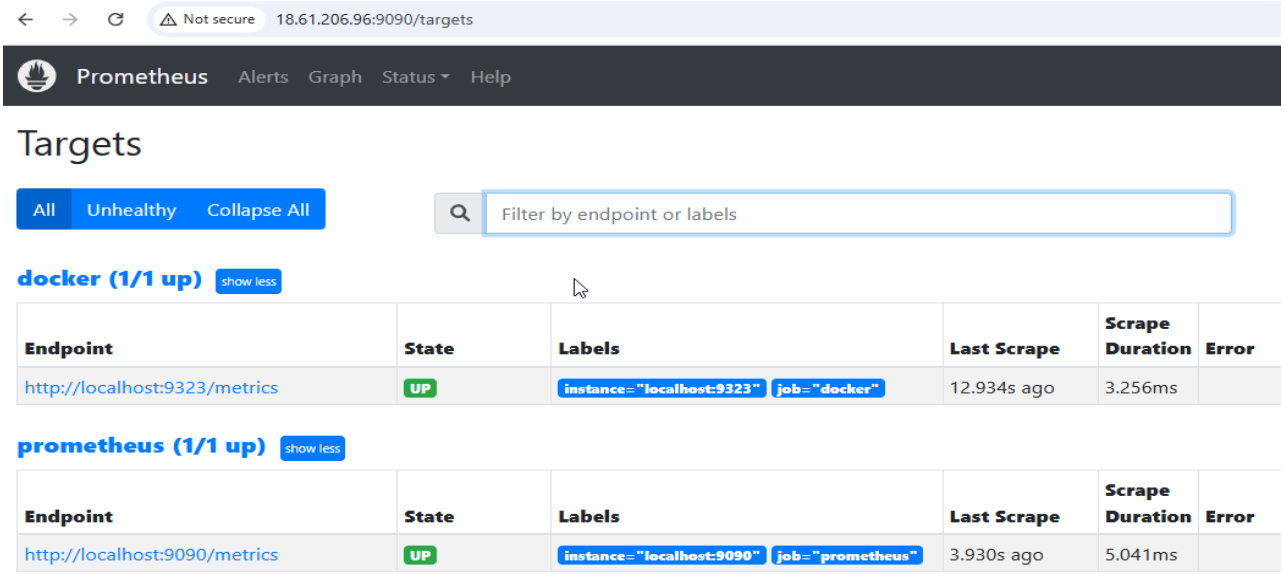
```
  # metrics_path defaults to '/metrics'
  # scheme defaults to 'http'.
  static_configs:
   - targets: ["localhost:9323"]
```

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

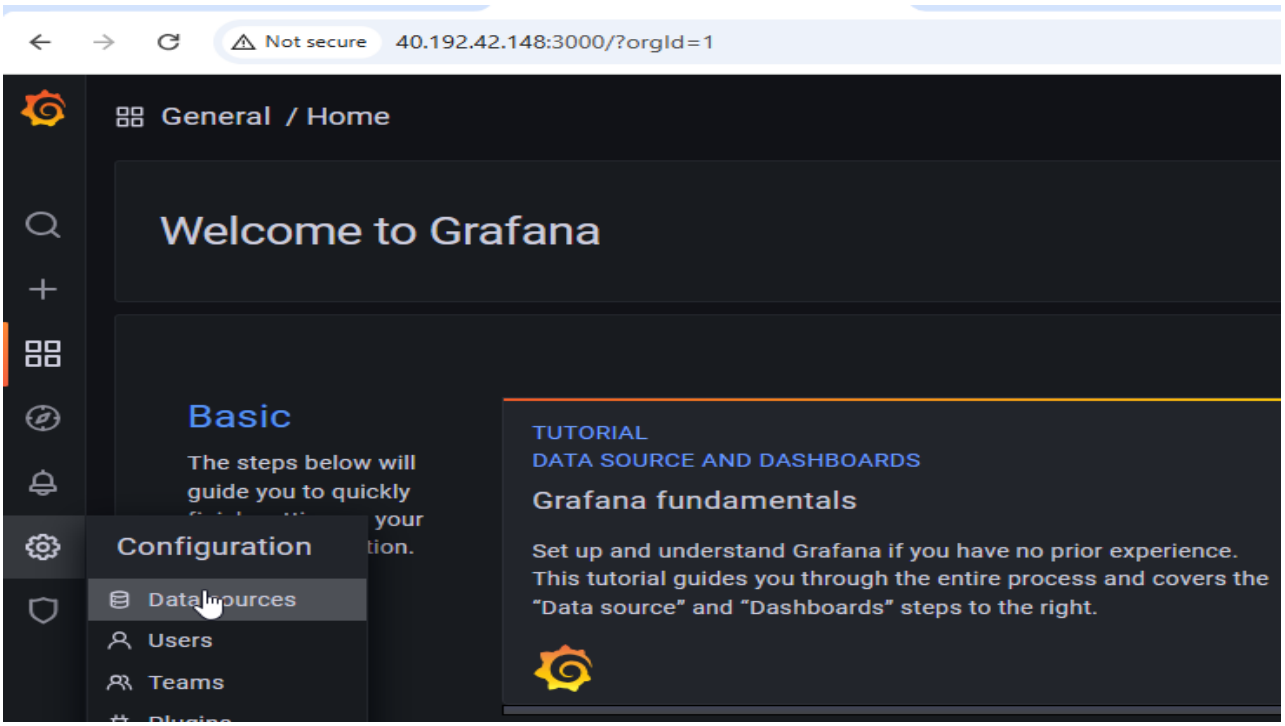
```
  - job_name: "docker"

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ["localhost:9323"]
```

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

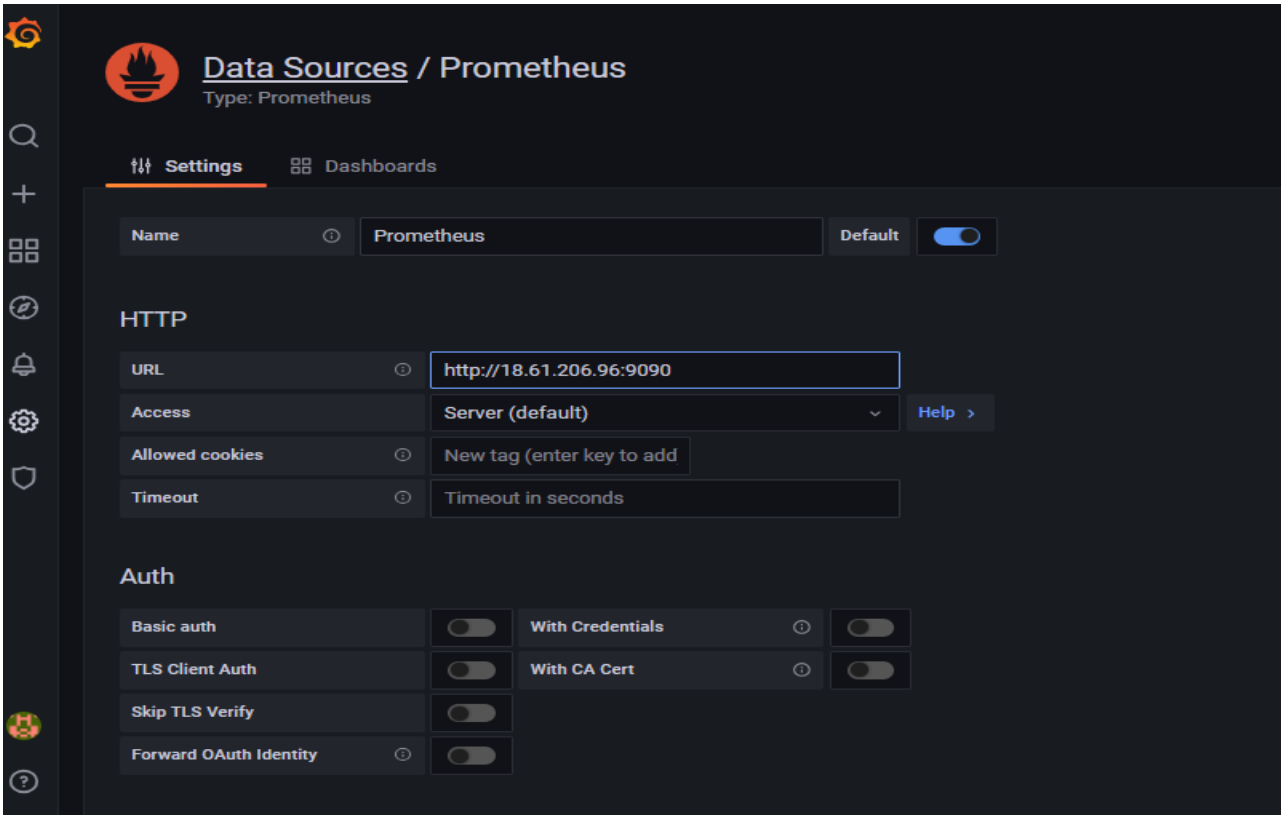
Step34:- Now go browser and give docker ip:9090 and enter , then you will be successfully enter into the Prometheus homepage and click on status ❼ targets then you will see the status of the of the docker and prometheus.



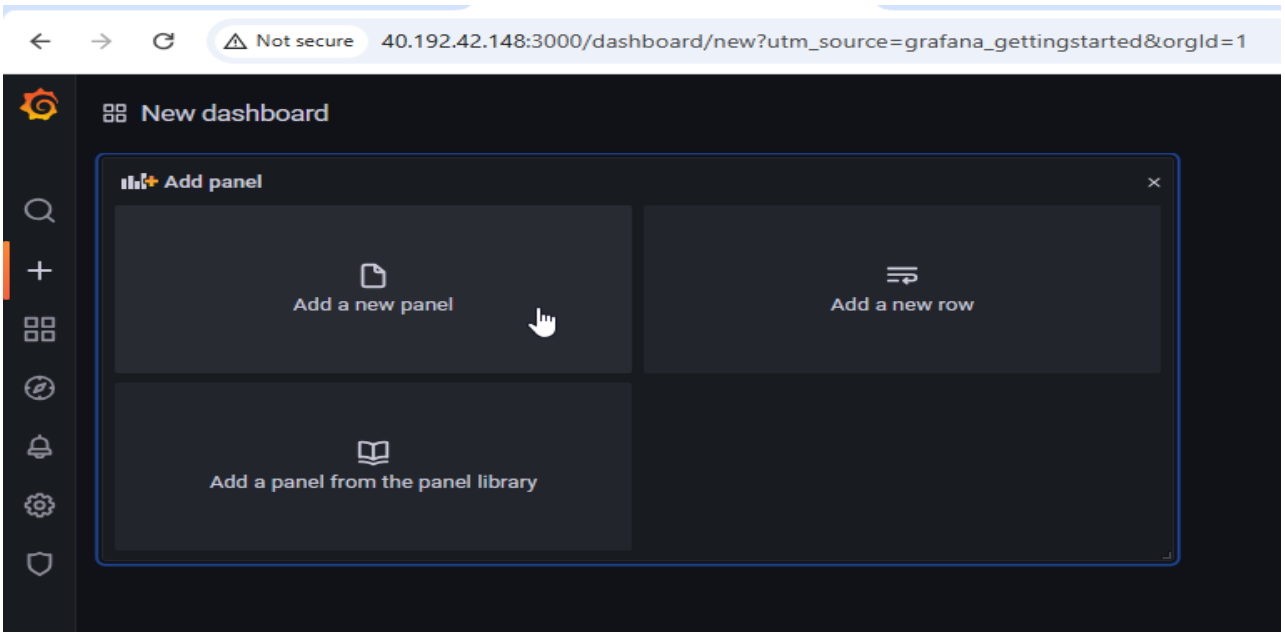
Step35:- Now go to grafana homepage ❼ configurations ❼ Data sources



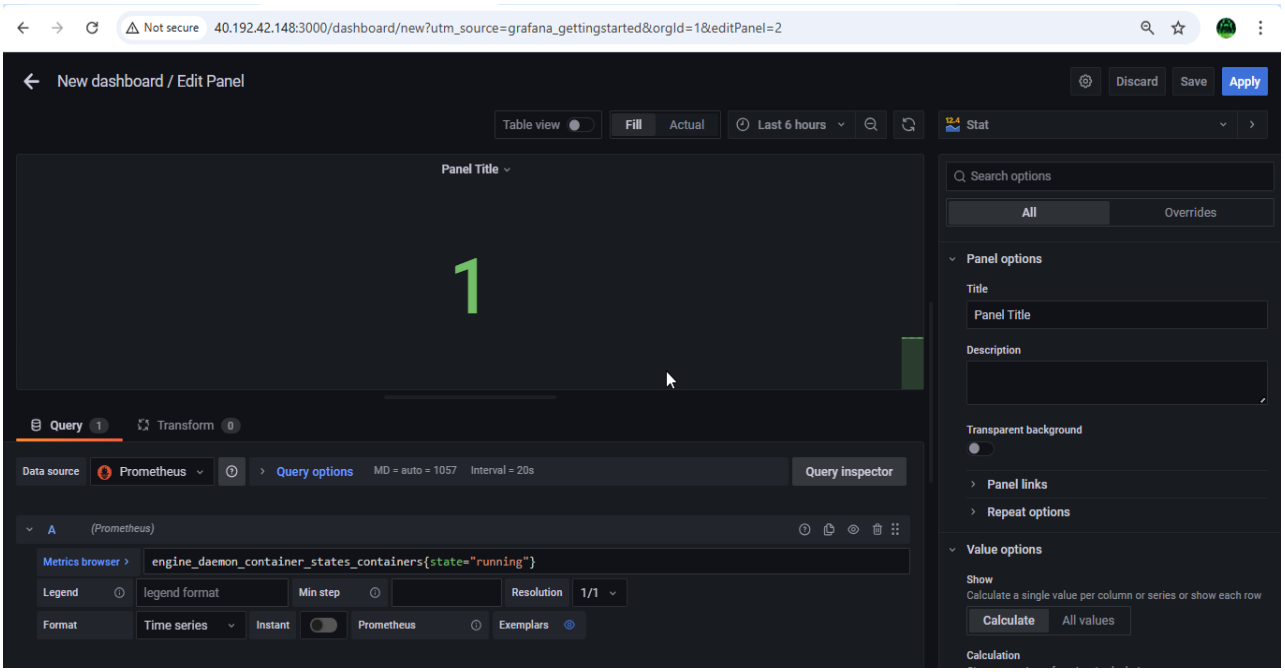
Step36:- Now click on add Data sources ❼ Prometheus and give ipaddress:9090 and click on save and test



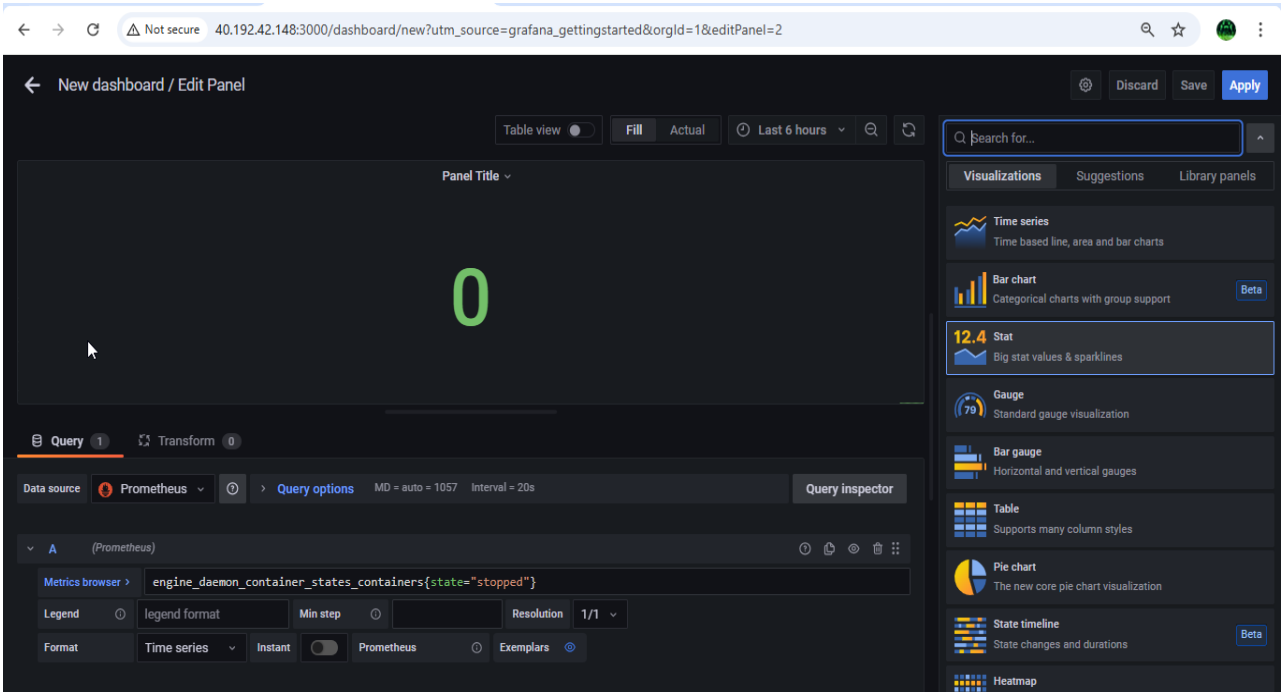
Step37:- Now click on Dash board ❼ add new panel



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

```
# HELP engine_daemon_container_states_containers The count of containers in various states
# 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
```

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

```
root@ip-10-0-1-127:/home/ubuntu/prometheus-2.34.0.linux-amd64# docker ps -a
CONTAINER ID   IMAGE                          COMMAND                  CREATED             STATUS          PORTS
          NAMES
03ec07cb8636   pj013525/insurance-image:v1   "java -jar /app.jar"   About an hour ago   Up 23 minutes   0.0.0.0:8082->8081/tcp, [::]:8082->8
081/tcp   insurance-container
root@ip-10-0-1-127:/home/ubuntu/prometheus-2.34.0.linux-amd64#
```

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.