Capstone Project

Medicure Health Care Project

By:- Jonna Padmarao

Source-code URL :- https://github.com/pj013525/star-agile-project-2.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/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
$
```

Name	Date modified	Type	Size
📁 .mvn	5/19/2025 12:34 PM	File folder	
📁 src	5/19/2025 12:34 PM	File folder	
📄 .DS_Store	5/19/2025 12:34 PM	DS_STORE File	7 KB
⚙️ .gitignore	5/19/2025 12:34 PM	Git Ignore Source File	1 KB
📄 ansible-playbook	5/19/2025 12:34 PM	Yaml Source File	1 KB
📄 Dockerfile	5/19/2025 12:34 PM	File	1 KB
📄 mvnw	5/19/2025 12:34 PM	File	11 KB
📄 mvnw	5/19/2025 12:34 PM	Windows Comman...	7 KB
📄 pom	5/19/2025 12:34 PM	XML Document	2 KB

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 --> To get origin list

git remote remove origin to remove the origin

```
MINGW64:/c/Users/Pj/Desktop/Star-agile-Finance-Pro/star-agile-banking-finance

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

Pj@lenovoIP300 MINGW64 ~/Desktop/Star-agile-Finance-Pro/star-agile-banking-finance (master)
$ git remote remove origin

Pj@lenovoIP300 MINGW64 ~/Desktop/Star-agile-Finance-Pro/star-agile-banking-finance (master)
$ git remote -v

Pj@lenovoIP300 MINGW64 ~/Desktop/Star-agile-Finance-Pro/star-agile-banking-finance (master)
$
```
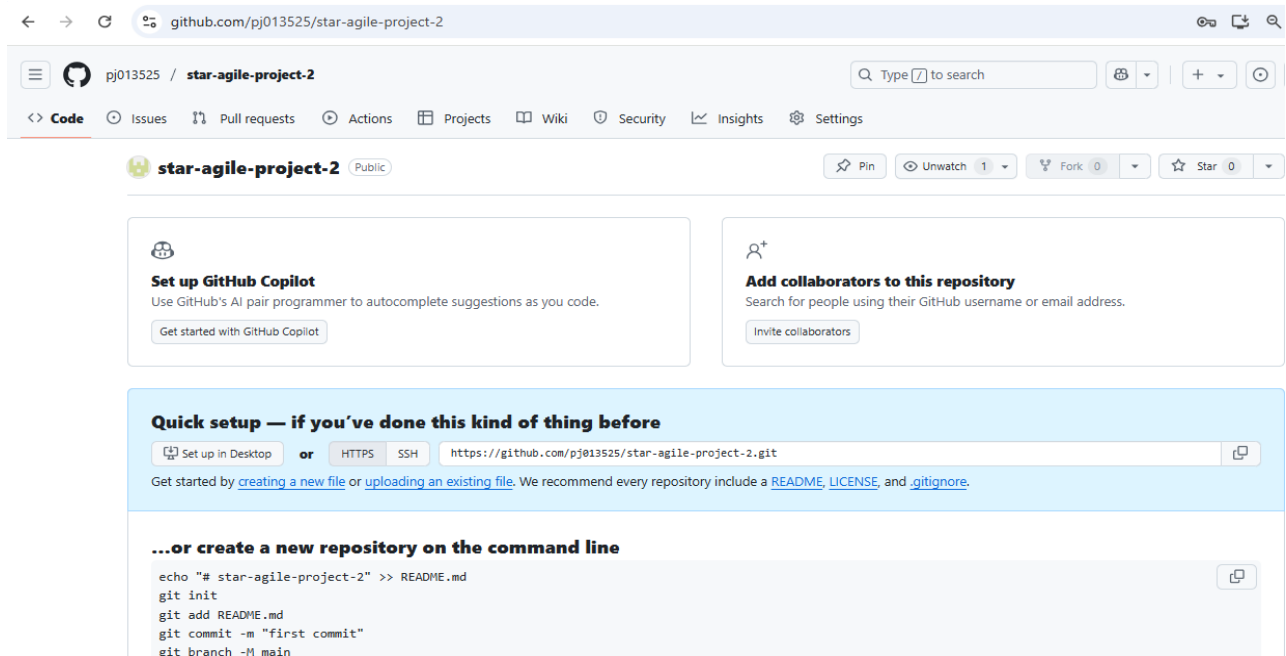
Step4:- Now go to github and create a new repo and copy the url in the gitbash

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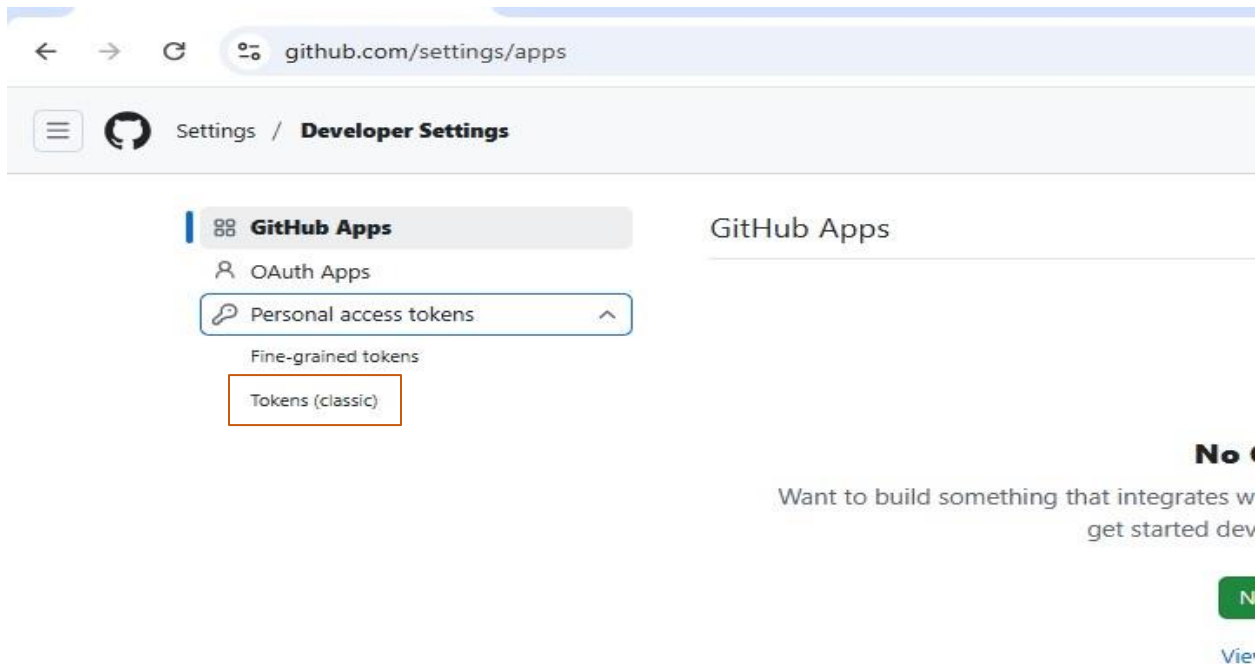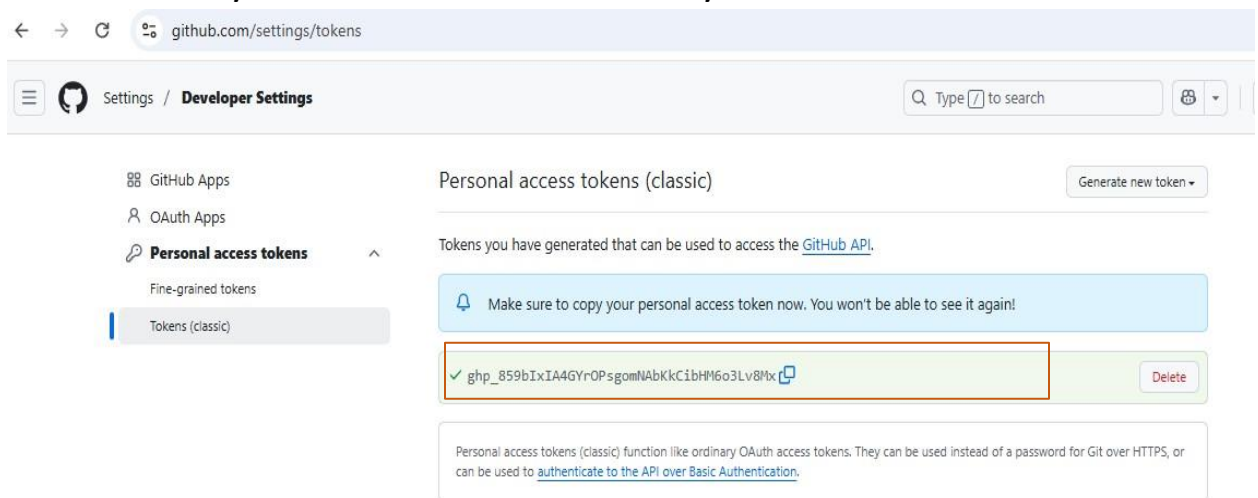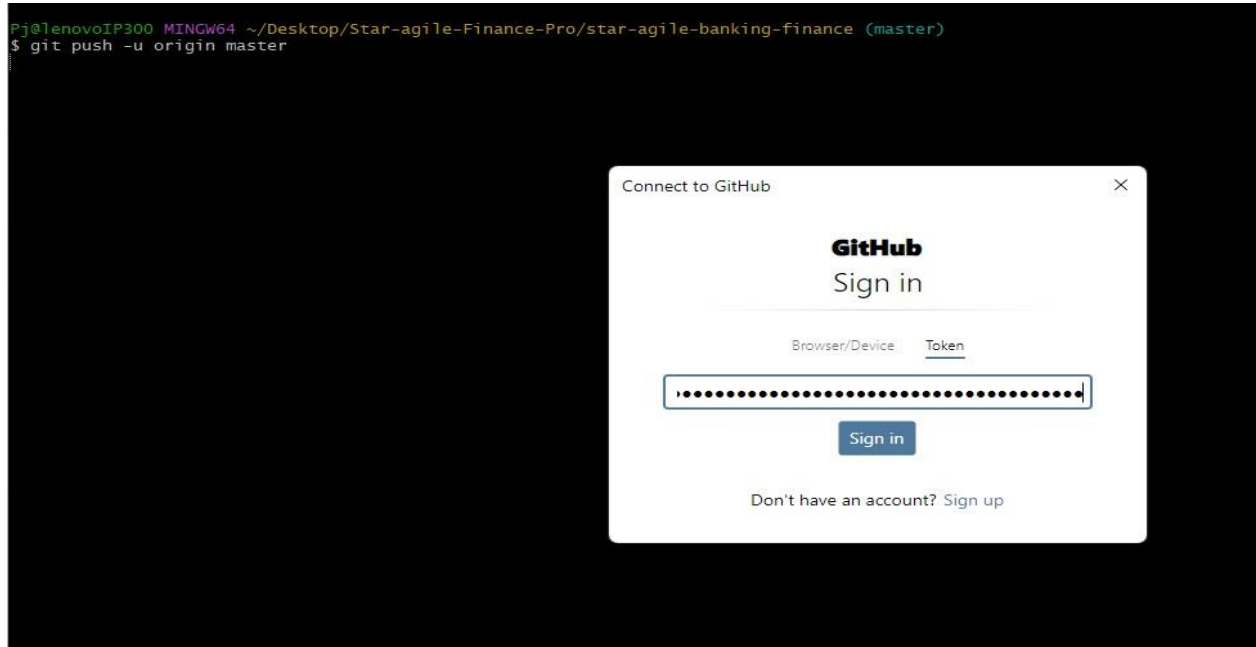
Step6:- Now again go to github ❼ Profile setting ❼Developer settings ❼ Personal access token Tokens(classic) ❼Generate new token

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

```
Pj@lenovoIP300 MINGW64 ~/Desktop/Star-agile-Finance-Pro/star-agile-banking-finance (master)
$ git push -u origin master
```

Connect to GitHub   ✕

GitHub

Sign in

Browser/Device   Token

●●●

Sign in

Don't have an account? Sign up

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

```
Pj@lenovoIP300 MINGW64 ~/Desktop/Star-agile-health-care-Pro/star-agile-health-care (master)
$ git push -u origin master
Enumerating objects: 174, done.
Counting objects: 100% (174/174), done.
Delta compression using up to 4 threads
Compressing objects: 100% (84/84), done.
Writing objects: 100% (174/174), 1.81 MiB | 1.30 MiB/s, done.
Total 174 (delta 52), reused 174 (delta 52), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (52/52), done.
To https://github.com/pj013525/star-agile-project-2.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

Pj@lenovoIP300 MINGW64 ~/Desktop/Star-agile-health-care-Pro/star-agile-health-care (master)
$ |
```

**Step10:-** Now go to the github repo and you will see the source code in that repo
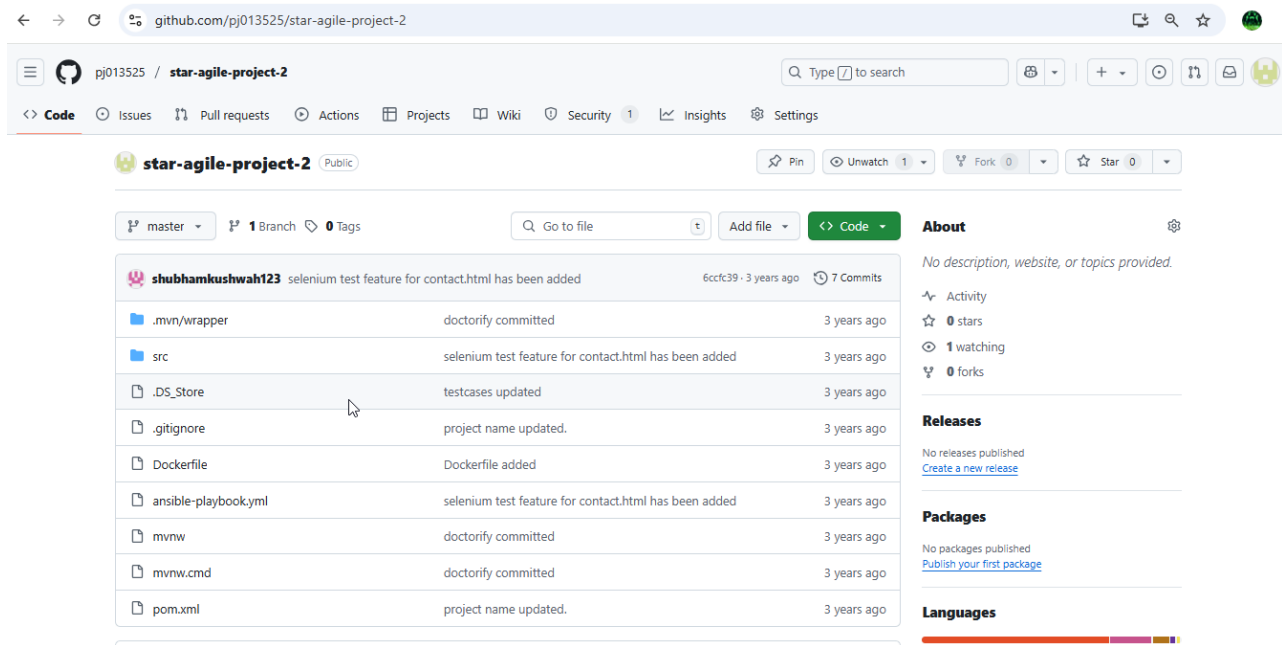
Step11:- Now create an instance using terraform as Iaac , and for that create a folder on desktop and go to browser download terraform for windows then a terraform application will be generated , now copy this application in to that folder and save
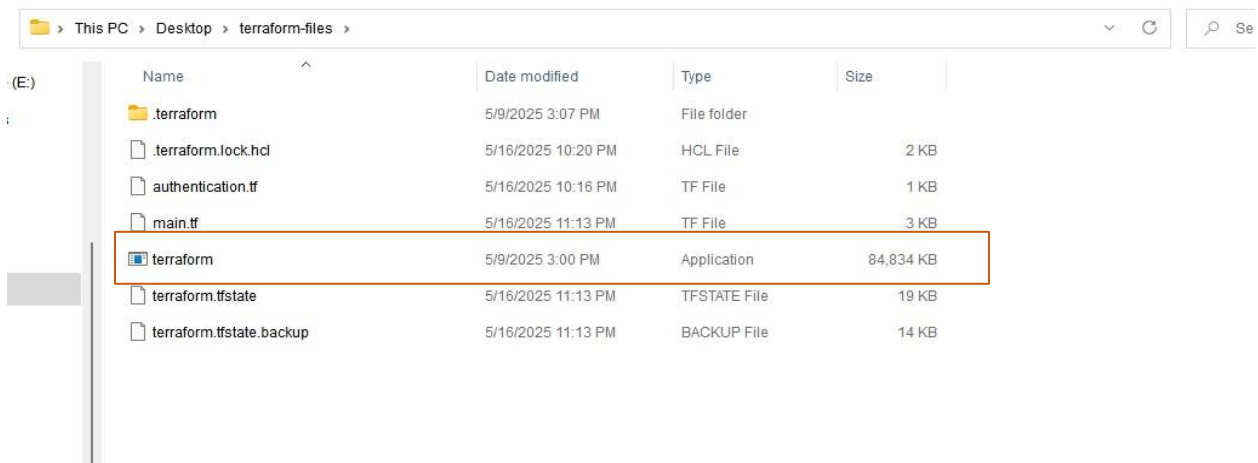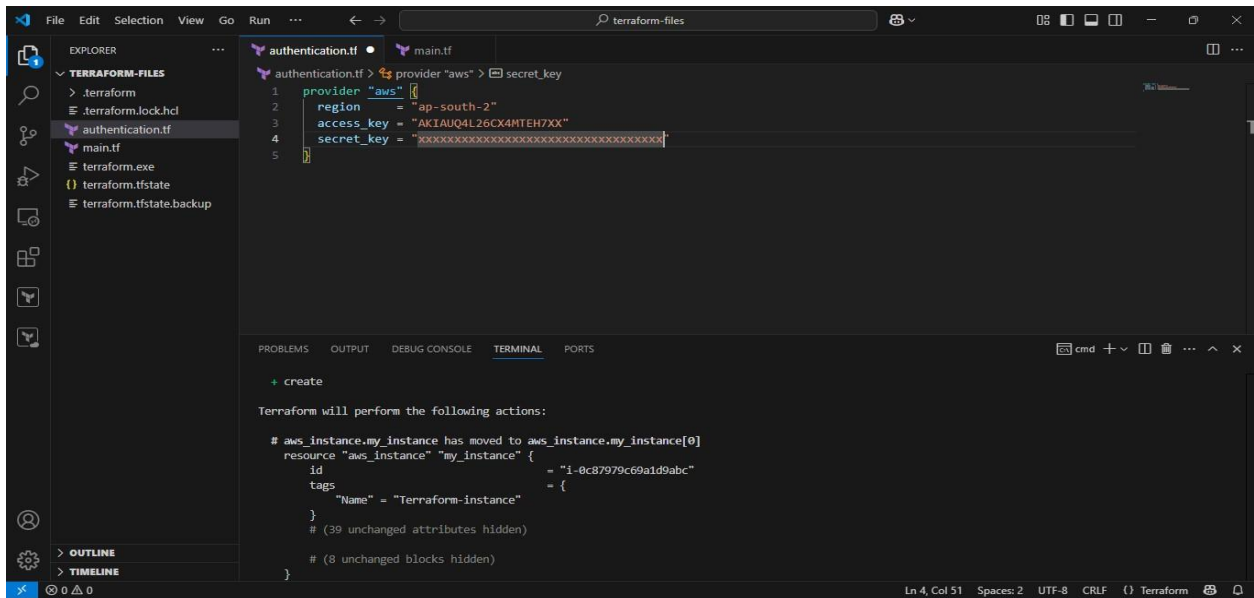
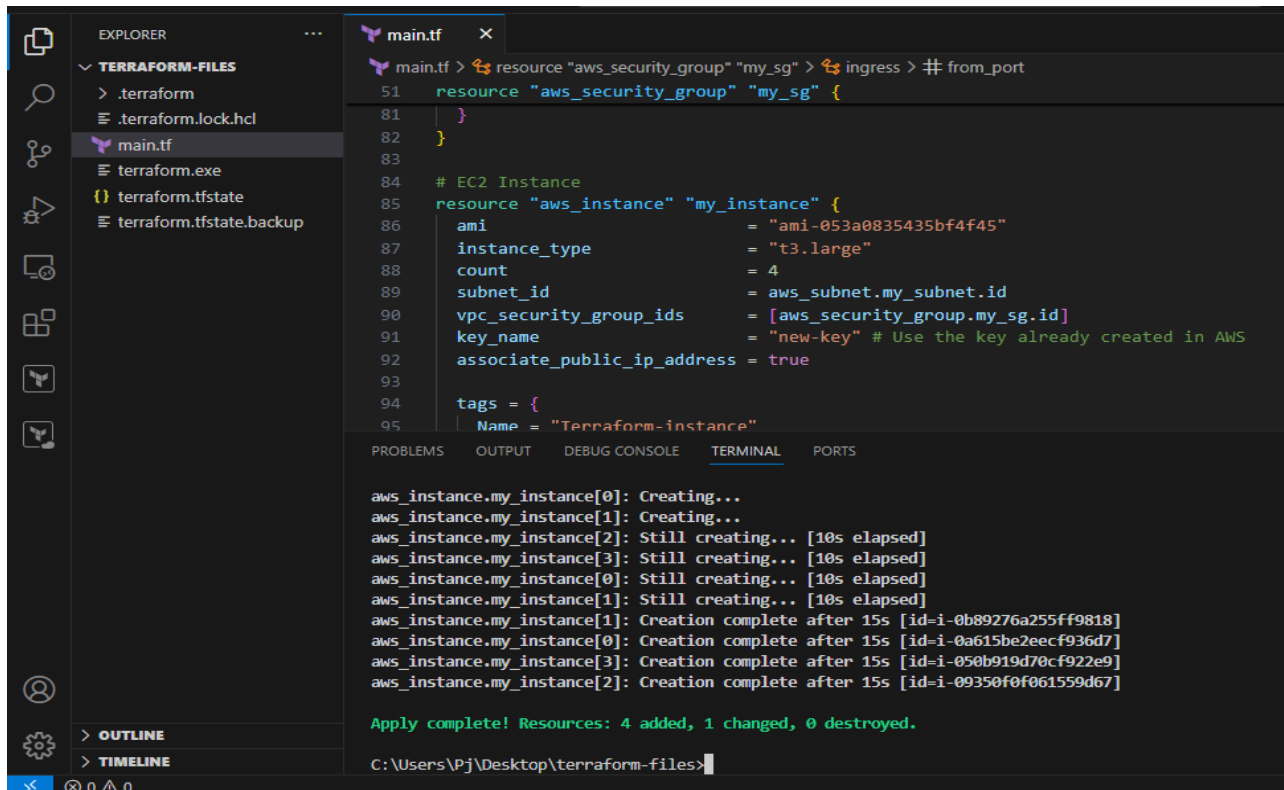**Step12:-** open visual studio code and go to terraform folder
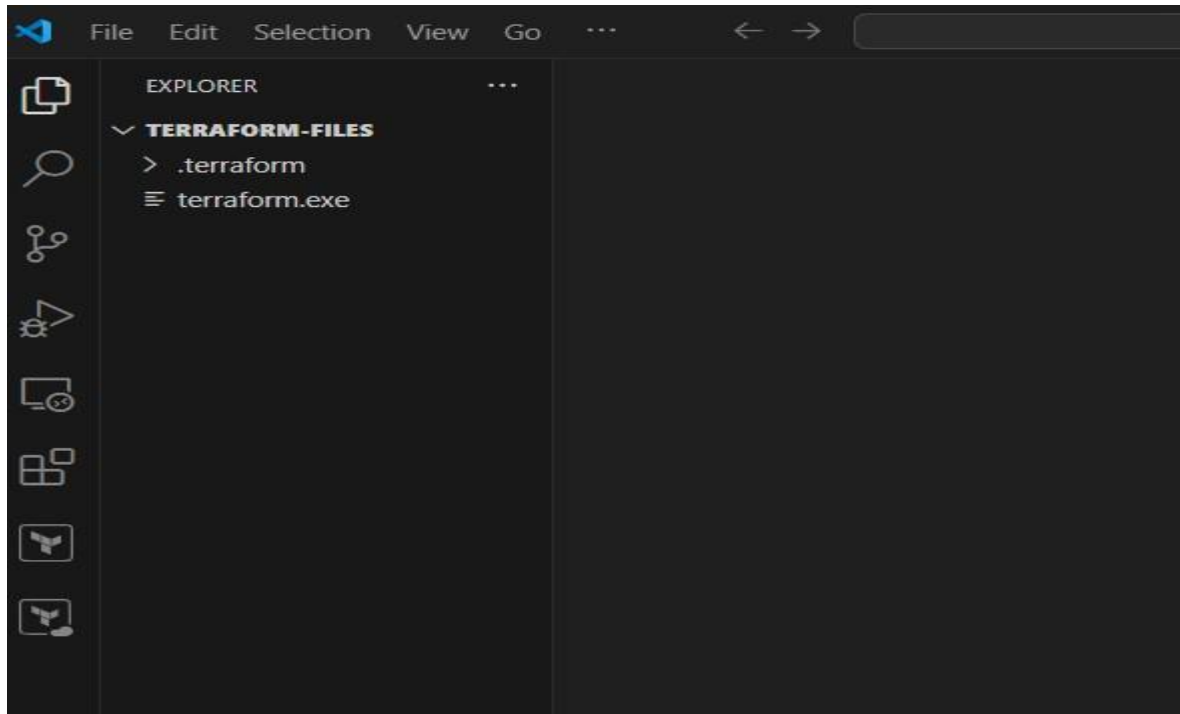
Step13:- Now create a file authentication.tf and give the provider and for that select the region in which you want to launch the server and go to aws account and go to profile ❼ credentials and go to access keys
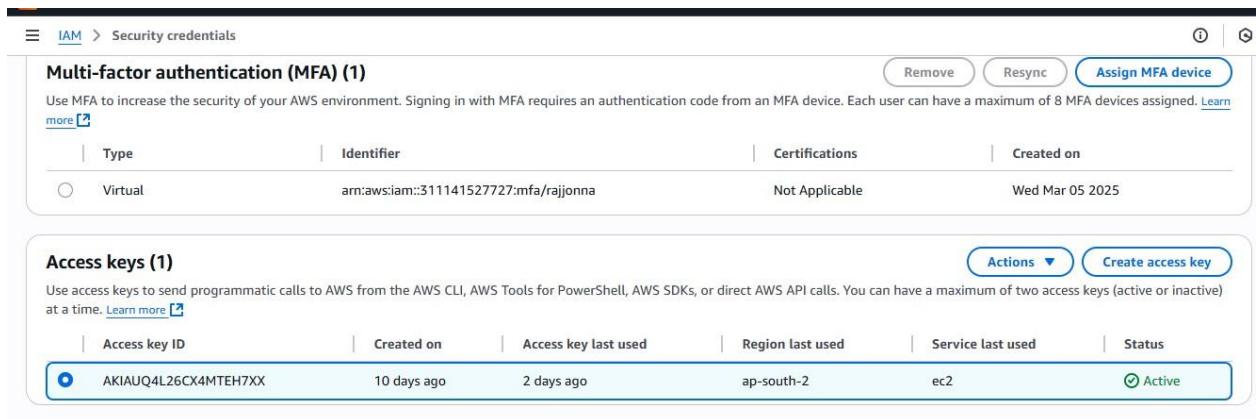
Step14:- now copy this access key details and paste it in this authentication.tf file and initialize it



```
provider "aws" {
  region     = "ap-south-2"
  access_key = "AKIAUQ4L26CX4MTEH7XX"
  secret_key = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
}
```

```
+ create

Terraform will perform the following actions:

  # aws_instance.my_instance has moved to aws_instance.my_instance[0]
    resource "aws_instance" "my_instance" {
        id                    = "i-0c87979c69a1d9abc"
        tags                  = {
            "Name" = "Terraform-instance"
        }
        # (39 unchanged attributes hidden)

        # (8 unchanged blocks hidden)
    }
```

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


```
51    resource "aws_security_group" "my_sg" {
81    }
82    }
83
84    # EC2 Instance
85    resource "aws_instance" "my_instance" {
86      ami                         = "ami-053a0835435bf4f45"
87      instance_type               = "t3.large"
88      count                       = 4
89      subnet_id                   = aws_subnet.my_subnet.id
90      vpc_security_group_ids      = [aws_security_group.my_sg.id]
91      key_name                    = "new-key" # Use the key already created in AWS
92      associate_public_ip_address = true
93
94      tags = {
95        Name = "Terraform-instance"
```

```
aws_instance.my_instance[0]: Creating...
aws_instance.my_instance[1]: Creating...
aws_instance.my_instance[2]: Still creating... [10s elapsed]
aws_instance.my_instance[3]: Still creating... [10s elapsed]
aws_instance.my_instance[0]: Still creating... [10s elapsed]
aws_instance.my_instance[1]: Still creating... [10s elapsed]
aws_instance.my_instance[1]: Creation complete after 15s [id=i-0b89276a255ff9818]
aws_instance.my_instance[0]: Creation complete after 15s [id=i-0a615be2eecf936d7]
aws_instance.my_instance[3]: Creation complete after 15s [id=i-050b919d70cf922e9]
aws_instance.my_instance[2]: Creation complete after 15s [id=i-09350f0f061559d67]

Apply complete! Resources: 4 added, 1 changed, 0 destroyed.

C:\Users\Pj\Desktop\terraform-files>
```

Step16:- After it is successful go and check the aws console and rename them as Ansible , Target and Master, Worker and Grafana instances
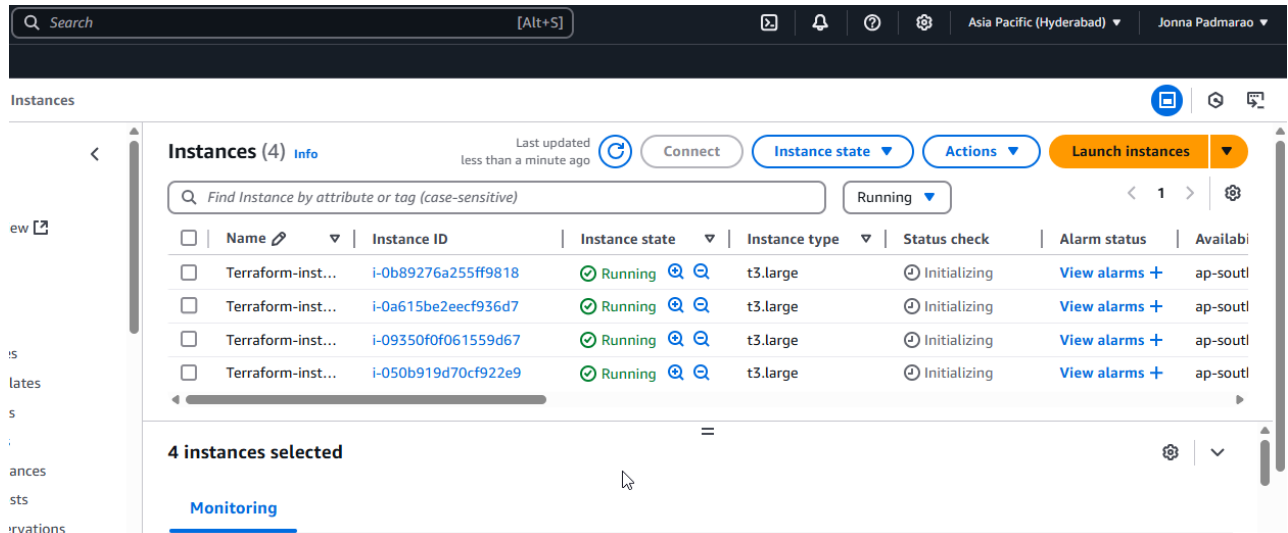
Step17:- Now connect to Ansible and target and master and worker servers using Mobaxterm agent and launch an instance
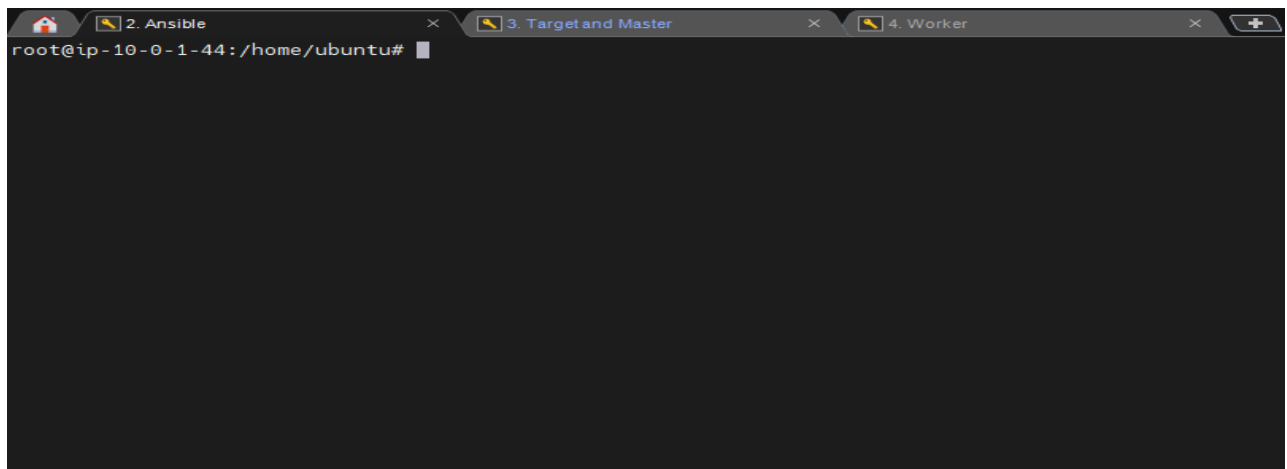
Step18:- Now install Ansible in Ansible server and connect this server with the Target and master sever and enable All traffic in the security group of this server

```
root@ip-10-0-1-44:/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-10-0-1-44:/home/ubuntu#
```

```
root@ip-10-0-1-44:/home/ubuntu# ansible all -m ping
[WARNING]: Platform linux on host 10.0.1.242 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.
10.0.1.242 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3.12"
    },
    "changed": false,
    "ping": "pong"
}
root@ip-10-0-1-44:/home/ubuntu#
```

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


```
$ cat installation.yml
---
- name: Install Java 17, Maven, Git, Jenkins, and Docker
  hosts: target
  become: yes

  tasks:
    - name: Install dependencies
      apt:
        name:
          - apt-transport-https
          - ca-certificates
          - curl
          - software-properties-common
          - gnupg
        update_cache: yes
        state: present

    - name: Install OpenJDK 17
      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 Jenkins GPG key
      apt_key:
        url: https://pkg.jenkins.io/debian/jenkins.io.key
        state: present
```

```
          state: present

    - name: Add Jenkins repository
      apt_repository:
          repo: 'deb https://pkg.jenkins.io/debian binary/'
          state: present

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

    - name: Enable and start Jenkins
      systemd:
          name: jenkins
          enabled: yes
          state: started

    - 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
          update_cache: yes
          state: present
```

Step20:- Now go to target and master node and verify the packages

```
TASK [Gathering Facts] **********************************************************
[WARNING]: Platform linux on host 10.0.1.242 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: [10.0.1.242]

TASK [Update apt cache] **********************************************************
changed: [10.0.1.242]

TASK [Install dependencies] **********************************************************
ok: [10.0.1.242]

TASK [Install Java 17] **********************************************************
ok: [10.0.1.242]

TASK [Install Maven] **********************************************************
ok: [10.0.1.242]

TASK [Install Git] **********************************************************
ok: [10.0.1.242]

TASK [Add Docker GPG key] **********************************************************
ok: [10.0.1.242]

TASK [Add Docker repository] **********************************************************
ok: [10.0.1.242]

TASK [Install Docker] **********************************************************
ok: [10.0.1.242]

TASK [Enable and start Docker service] **********************************************************
ok: [10.0.1.242]

PLAY RECAP **********************************************************
10.0.1.242                 : ok=10   changed=1   unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

root@ip-10-0-1-44:/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:- Now I am installing sonarqube in the target for code Quality analysis for checking any type of bugs , vulnerabilities in the code and logging into the sonarqube home page using browser with ip:9000
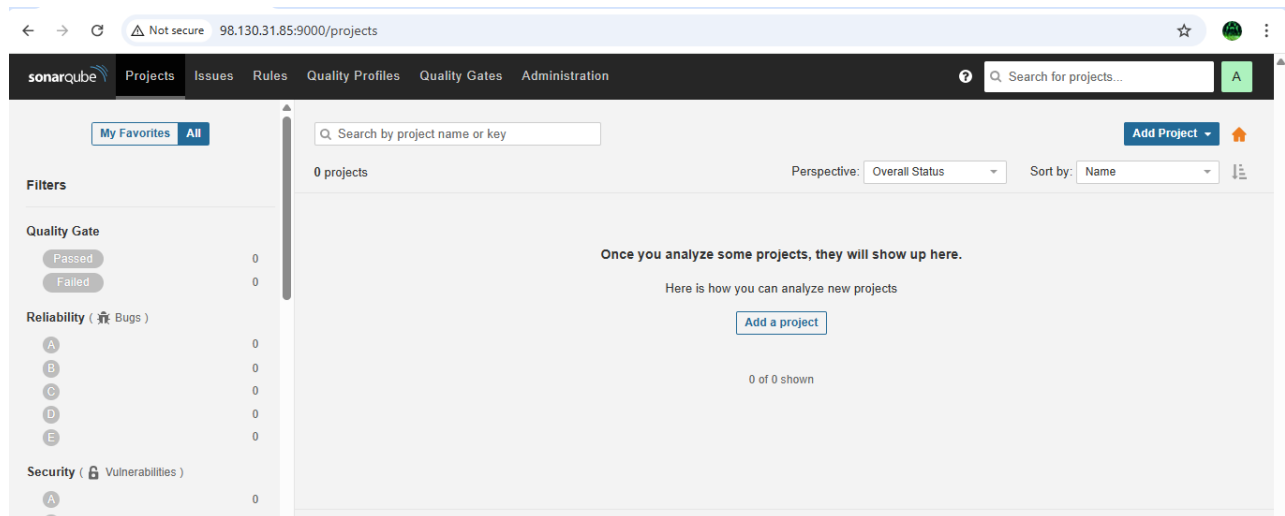
Step23:- Now i am installing nexus tool in the target node to store the artifacts that generated from the maven build
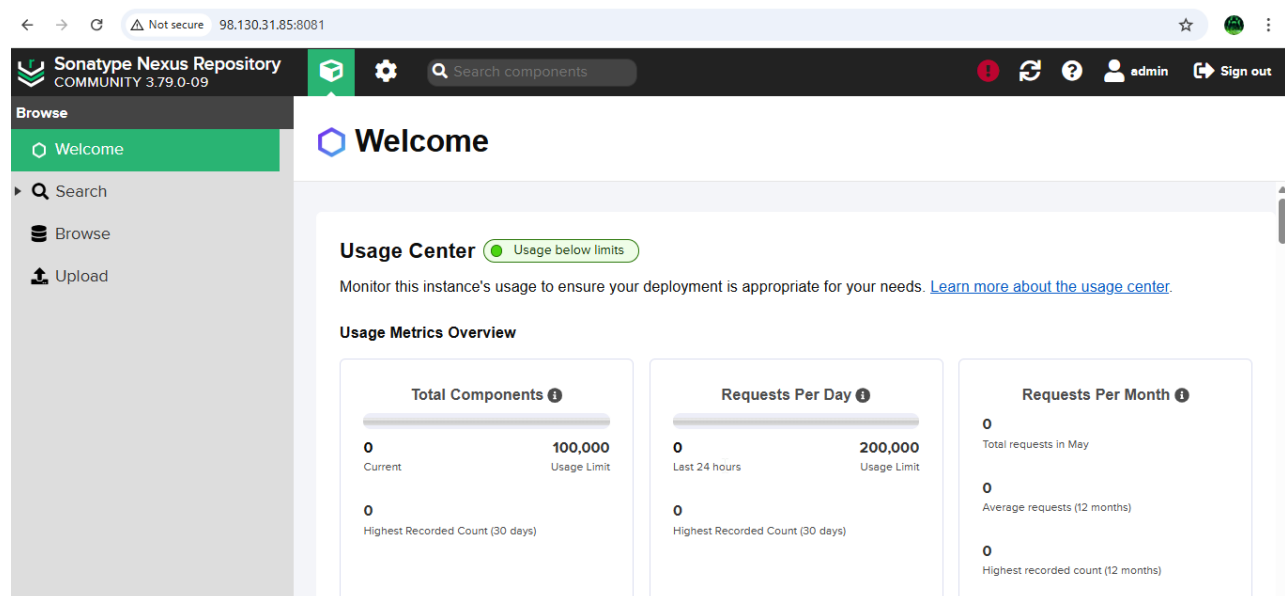
Step24:- Go to the any browser and give the details and click on recommended plugins and login to the Jenkins
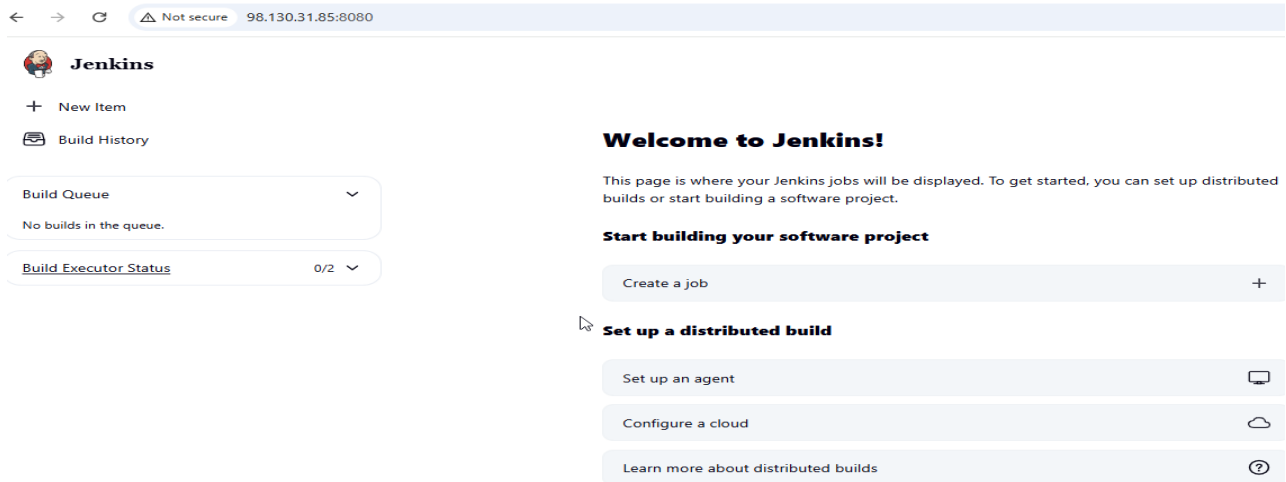
Step25:- Now in the Jenkins dashboard click on new item an give any name and select pipeline project as type and click on ok
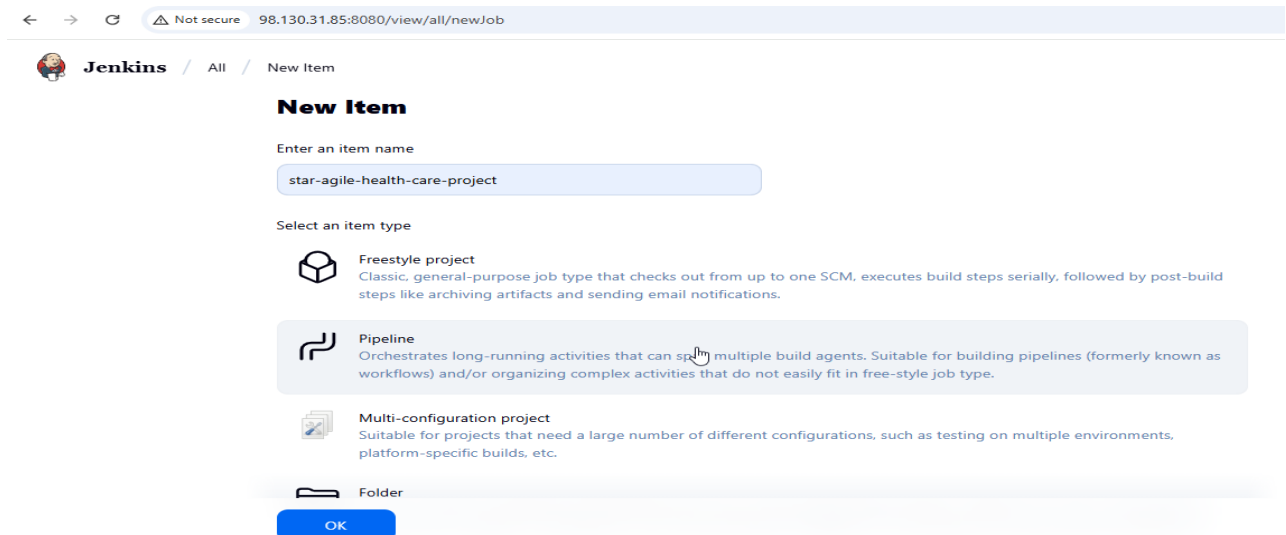
Step26:- Now install docker and other required plugins in the Jenkins

Pipeline stage view

Git Plugin   nexus artifact uploader

                                                              SonarQube scanner

Docker Pipeline Plugin

Credentials Binding Plugin   Sonar quality gates

Docker Commons Plugin

 Pipeline: GitHub

Maven Integration Plugin

Jenkins / Manage Jenkins / Plugins

Plugins

- Updates
- Available plugins
- Installed plugins
- Advanced settings
- Download progress

Email Extension	✓ Success
Mailer	✓ Success
Theme Manager	✓ Success
Dark Theme	✓ Success
Loading plugin extensions	✓ Success
Pipeline: REST API	✓ Success
Pipeline: Stage View	✓ Success
Javadoc	✓ Success
Dev Tools Symbols API	✓ Success
jsoup API	✓ Success
JSch dependency	✓ Success
Maven Integration	✓ Success
Authentication Tokens API	✓ Success
Docker Commons	✓ Success
Docker Pipeline	✓ Success
JavaMail API	✓ Success
Commons HttpClient 3.x API	✓ Success
Nexus Artifact Uploader	✓ Success
SonarQube Scanner	✓ Success
Apache HttpComponents Client 5.x API	✓ Success
Sonar Quality Gates	✓ Success
Loading plugin extensions	✓ Success

Step27:- 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

Jenkins / Manage Jenkins / Credentials / System / Global credentials (unrestricted) / pj013525/****** (dockerhub)

- Update
- Delete
- Move

Update credentials

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Username ?
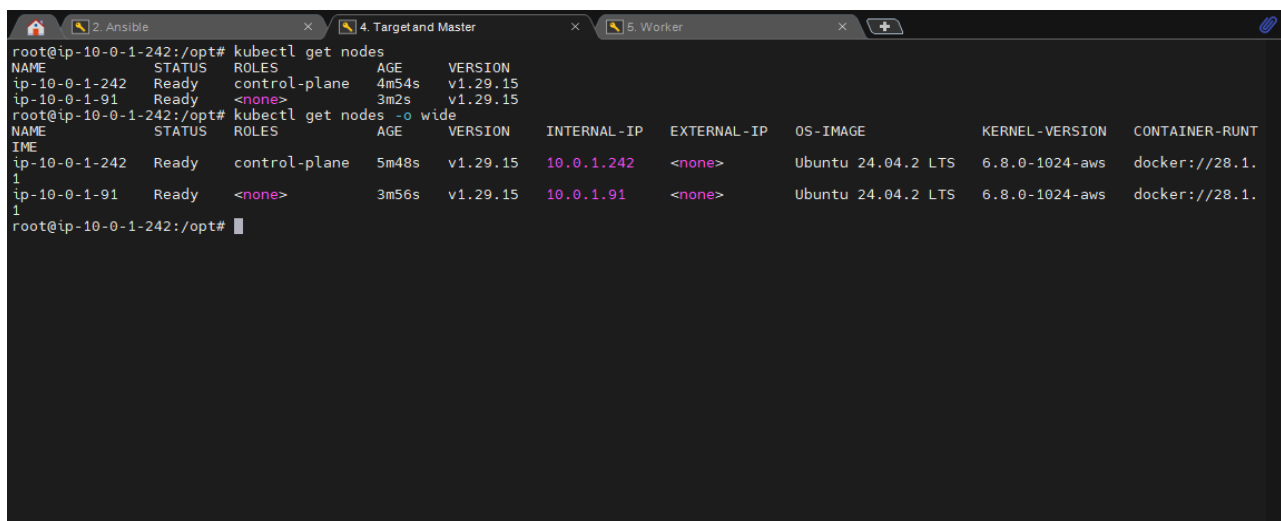
pj013525

☐ Treat username as secret ?

Password ?

🔒 Concealed   [Change Password]

ID ?

dockerhub-creds

Description ?

dockerhub

[Save]

Step28:- Now in jenkins configure the sonarqube and nexus tools

Go to SonarQube ----> Projects ----> Add project ----> Manually ---->
Project key: Health-care, Display name: Health-care ----> Setup ---->
Generate a token: Health-care ----> Generate ----> Copy the token
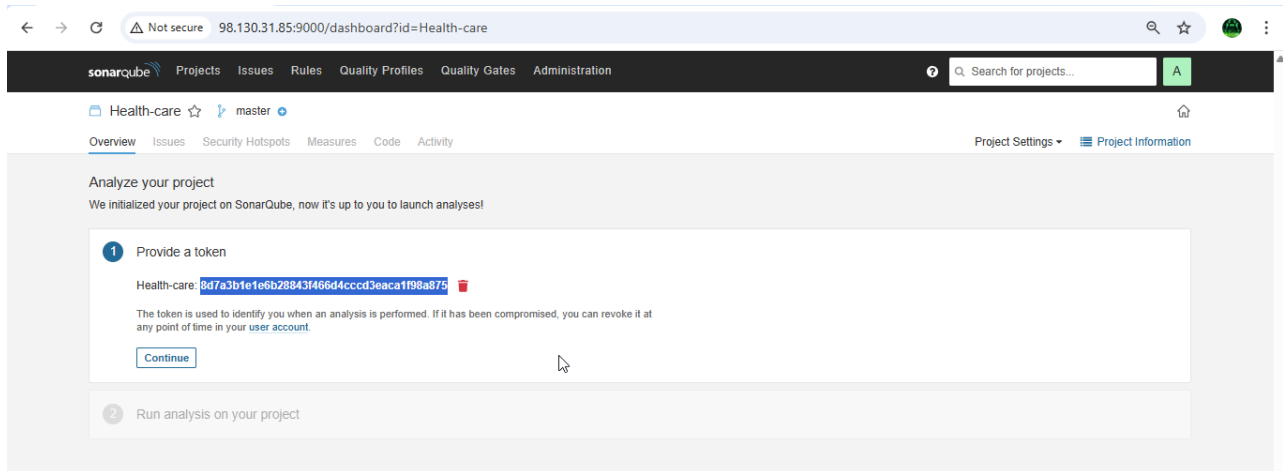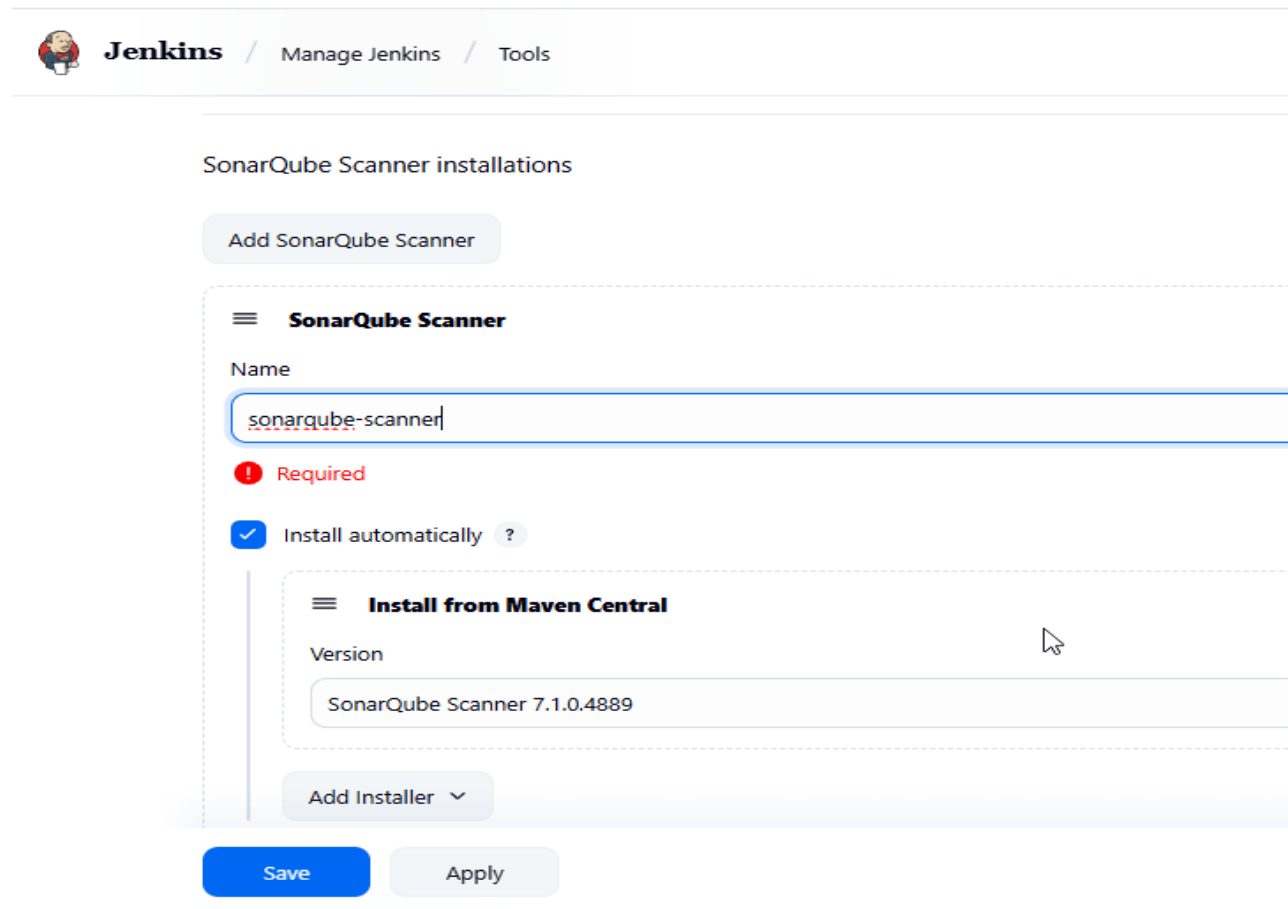
Step29:- Lets configure the SonarQube credentials in Jenkins and for
that manage-jenkins ==> Credentials ==> global ==> Add credentials==>
select secret text and give the text that copied form the sonarqube
home page
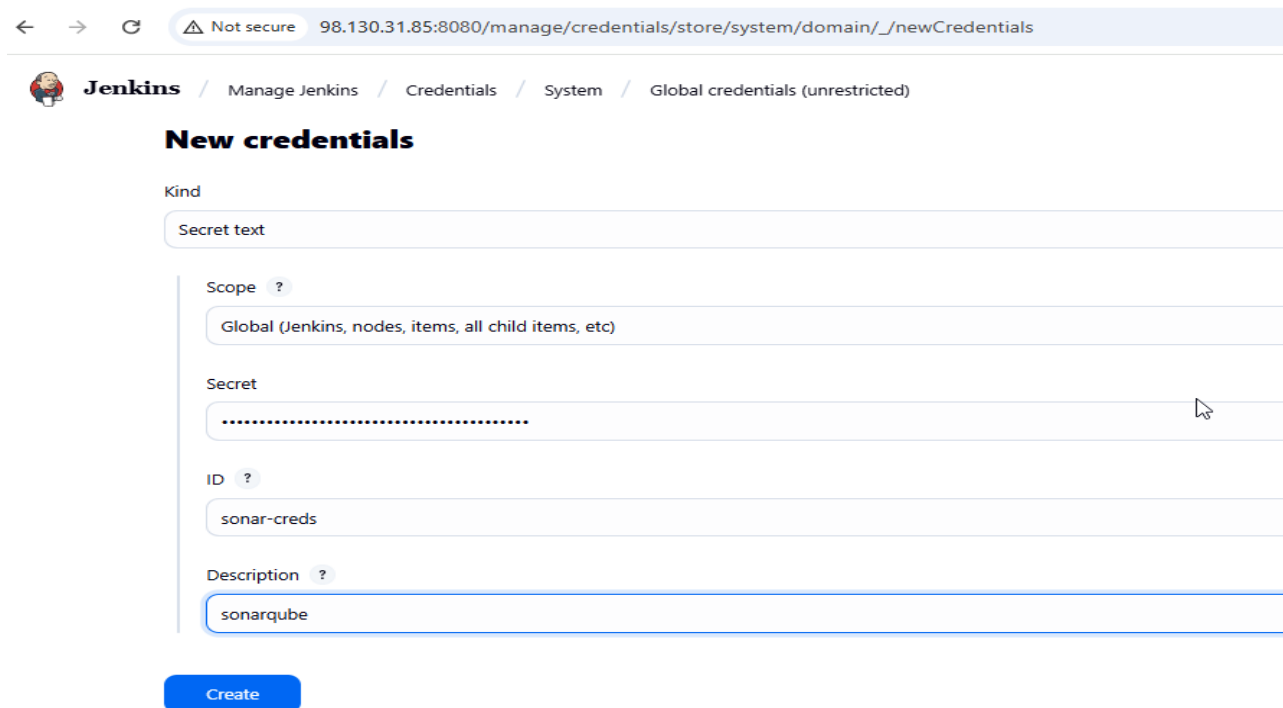
Step30:- Similarly add sonarqube tool also as

jenkins ----> Manage Jenkins ----> System Configuration ----> Tools ---->
Scroll down to 'sonarqube Scanner Installations' ----> Add SonarQube
scanner ----> Name: sonarscanner, 'Check' Install automatically, Version:
7.1.0.4889 ----> Apply ----> Save.

Step31:- Now configure the sonarqube in the system of the jenkins and
for that jenkins ----> Manage Jenkins ----> System Configuration ---->
System ----> Scroll down to 'SonarQube servers' ----> 'Check'
environment variables, Add SonarQube ----> Name: sonarqube, Server
URL: <SonarQube URL> [only upto 9000] ----> Server authentication
token: Select 'sonarqube' from dropdown ----> Apply ----> Save

SonarQube installations
List of SonarQube installations

Name

sonarqube-server

Server URL
Default is http://localhost:9000

http://98.130.31.85:9000

Server authentication token
SonarQube authentication token. Mandatory when anonymous access is disabled.

sonarqube

+ Add

Advanced ∨

Save   Apply

Step32:- Now configure the nexus in the jenkins for that first

Go to Nexus tab in browser --> Click on settings icon ----> Click on 'repositories' ----> Create repo ----> Scroll down and Click on 'maven2 hosted' ----> Name: Health-care, Version policy: Snapshot, Layout policy: Strict, Content disposition: Inline, Blob store:default, Deployment policy: Allow redeploy ----> Create repository ----> You will see 'Health-care' repo

Step33:- Now configure the nexus credentials in the jenkins dash board through Jenkins ----> Manage Jenkins ----> Security ----> Credentials ----> Click on 'global' ----> Add creds ----> Kind: Username with Password, Scope: Global, Username: admin, Password: <Enter The Nexus> , ID: nexus-creds, Description: nexus-creds ----> Create

Step34:- Now go to Target and Master instance and install Kubernetes and configure with Worker node and verify

Step35:- Now go to Project repo in the github and there add
deployment.yml and jenkinsfile to perfomr the pipeline in the Jenkins.

Code   Blame   40 lines (36 loc) - 650 Bytes   Code 55% faster with GitHub Copilot

```yaml
1    ---
2    apiVersion: apps/v1
3    kind: Deployment
4    metadata:
5      name: health-care-deployment
6      labels:
7        app: health-care
8    spec:
9      replicas: 2
10     selector:
11       matchLabels:
12         app: health-care
13     template:
14       metadata:
15         labels:
16           app: health-care
17       spec:
18         containers:
19           - name: health-care-container
20             image: pj013525/health-care:v1
21             ports:
22               - containerPort: 8082
23             imagePullPolicy: Always
24    ---
25    apiVersion: v1
26    kind: Service
27    metadata:
28      name: health-care-service
29    spec:
30      selector:
31        app: health-care
32      ports:
33        - protocol: TCP
34          port: 80
35          targetPort: 8082
36      type: NodePort
37
```

Code   Blame   68 lines (59 loc) · 1.94 KB   Code 55% faster with GitHub Copilot

```groovy
1     pipeline {
2         agent any
3
4         stages {
5             stage('Git checkout') {
6                 steps {
7                     git branch: 'master', url: 'https://github.com/pj013525/star-agile-project-2.git'
8                 }
9             }
10
11            stage('Compilation') {
12                steps {
13                    sh 'mvn compile'|
14                }
15            }
16
17            stage('Testing') {
18                steps {
19                    sh 'mvn test'
20                }
21            }
22
23            stage('SonarQube Scanner') {
24                steps {
25                    withSonarQubeEnv('sonarqube-server') {
26                        sh 'mvn org.sonarsource.scanner.maven:sonar-maven-plugin:5.0.0.4389:sonar'
27                    }
28                }
29            }
30
31            stage('Packing') {
32                steps {
33                    sh 'mvn clean package'
```

```
35              }
36
37          stage('Pushing Artifact to Nexus') {
38              steps {
39                  nexusArtifactUploader artifacts: [[artifactId: 'medicure', classifier: '', file: 'target/medicure-0.0.1-SNAPSHOT.jar', type: 'jar']], credentialsId: 'nexus-creds', groupId:
40              }
41          }
42
43          stage('Build the Docker Image') {
44              steps {
45                  script {
46                      sh 'docker build -t pj013525/health-care:v1 .'
47                      sh 'docker images'
48                  }
49              }
50          }
51
52          stage('Push Image to DockerHub') {
53              steps {
54                  withCredentials([string(credentialsId: 'dockerhub-details', variable: 'dockerhubPassword')]) {
55                      sh "docker login -u pj013525 -p ${dockerhubPassword}"
56                      sh 'docker push pj013525/health-care:v1'
57                  }
58              }
59          }
60          stage('Deploy using k8s') {
61              steps {
62                  sh 'sudo kubectl apply -f deployment.yml'
63                  sh 'sudo kubectl get all'
64              }
65          }
66      }
67  }
```

Step36:- 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 2pods are also created in the ec2 Worker node.

Step37:- Now go to sonarqube page and check the the project details
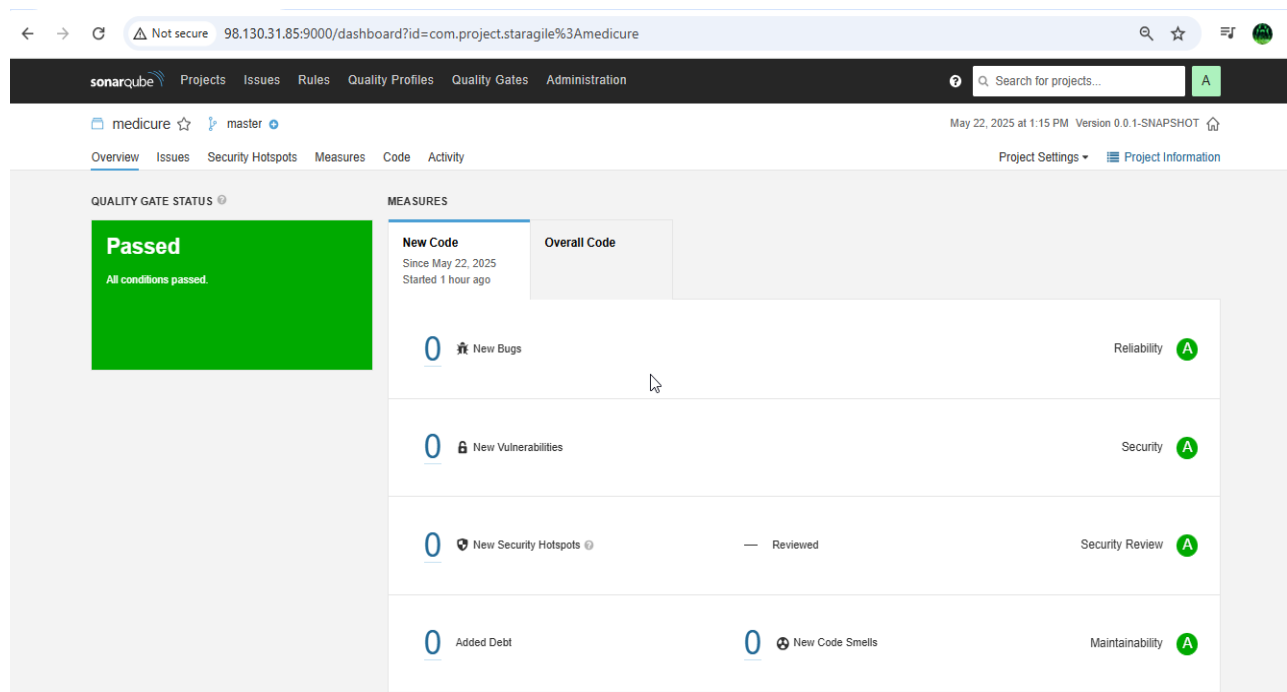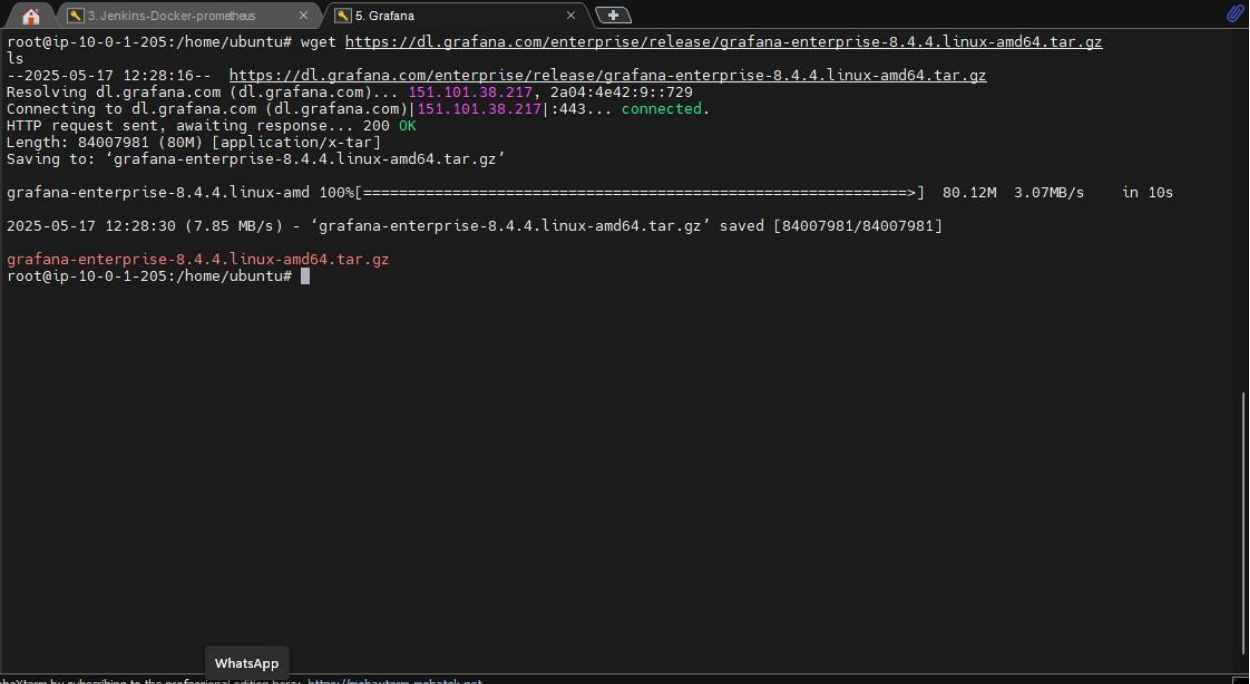
Step38:- Now check the test results if it is passed then the code is correct and now check the Artifact that created in the build in Nexus
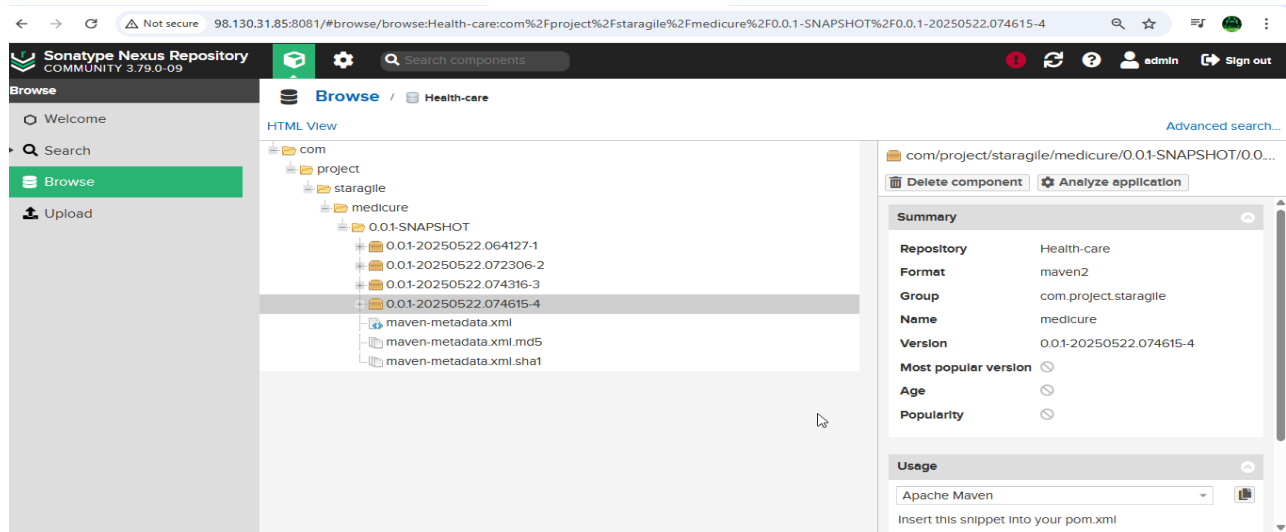
Step39:- Now go to any browser and give the <Workernode-IPaddress:31870> (since port is auto allocated) and click enter the you will see the home page of the project and thus the project deployment is successful using kubernetes.
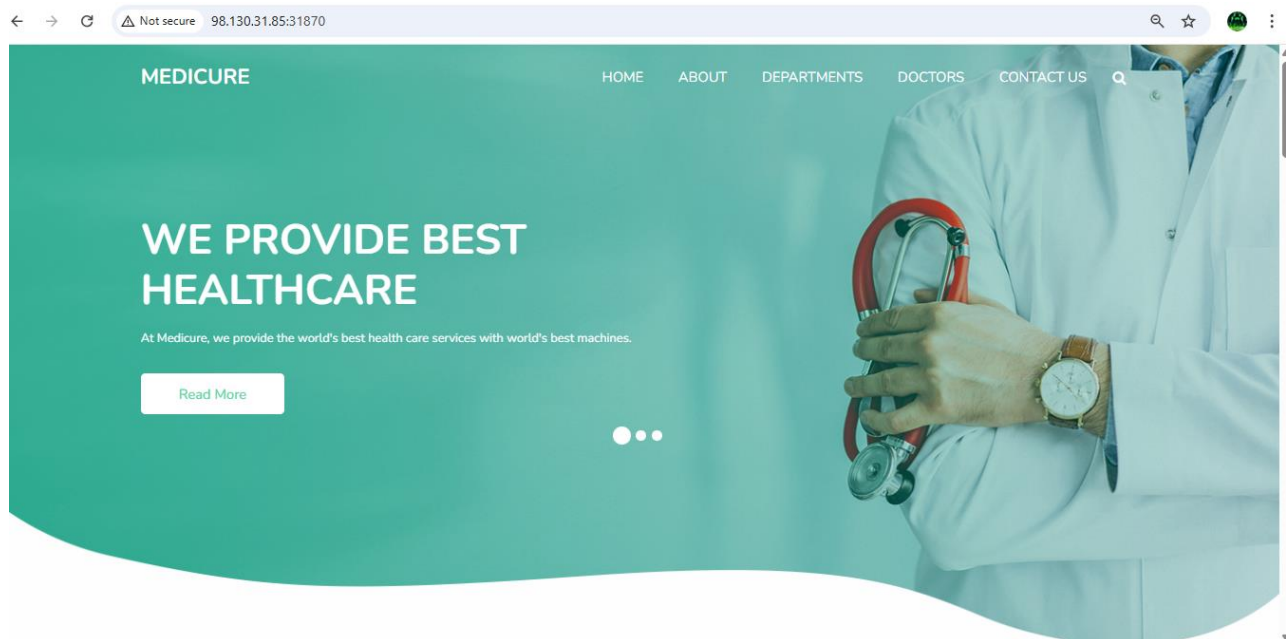
Step40:- Now monitor the pods using Prometheus and Grafana , for that install Prometheus in Jenkins-Docker server and Grafana in another server

Step41:- 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.
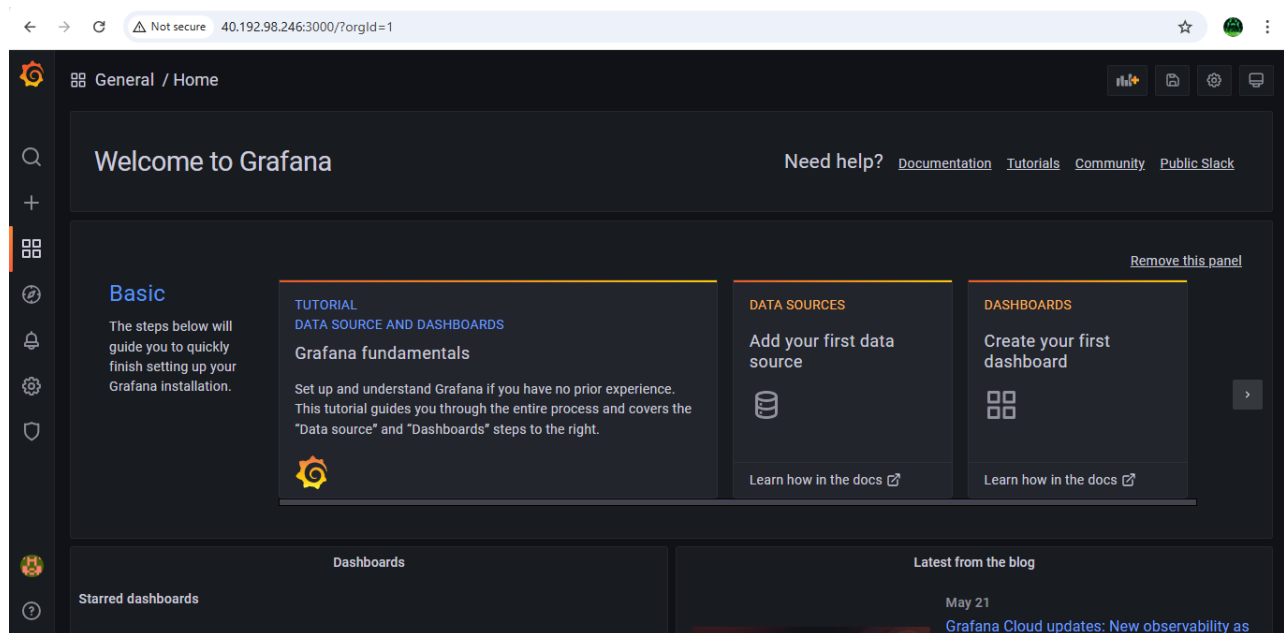
Step42:- Now setup the docker and Prometheus in another using by telling docker that Prometheus would track docker on port 9323

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

```
root@ip-10-0-1-242:/home/ubuntu/prometheus-2.34.0.linux-amd64# cat /etc/docker/daemon.json
{
        "metrics-addr" : "0.0.0.0:9323",
        "experimental" : true
}

root@ip-10-0-1-242:/home/ubuntu/prometheus-2.34.0.linux-amd64#
```

Step43:- 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

Step44:- 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


```
# The job name is added as a label `job=<job_name>` to any times
- job_name: "prometheus"

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

    static_configs:
        - targets: ["localhost:9090"]

- job_name: "docker"

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

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

"prometheus.yml" 38L, 1098B
```
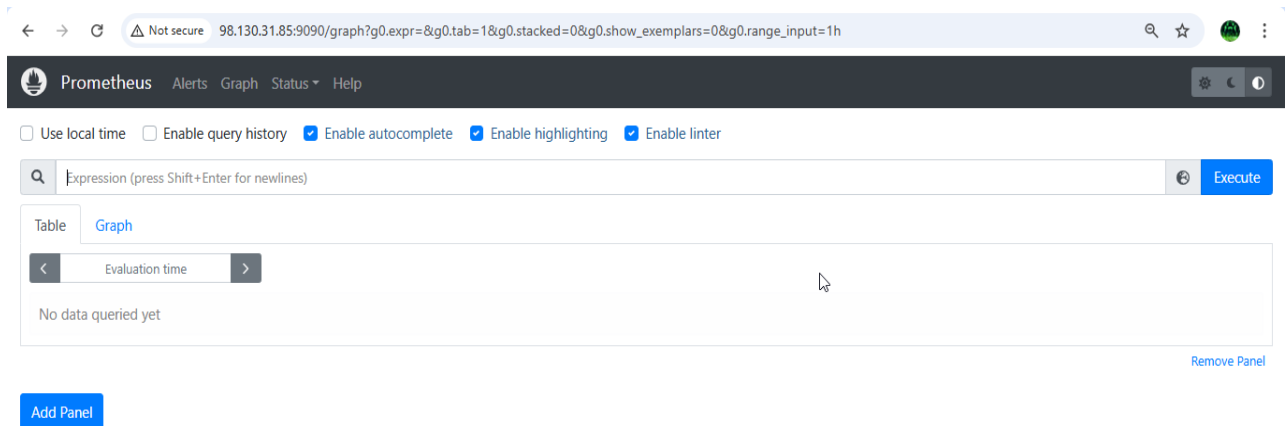
As you can see that the Prometheus have been started form the above image

Step45:- Now go browser and give docker ip:9090 and enter , then you will be successfully enter into the Prometheus homepage

Step46:- Now click on status ❼ targets then you will see the status of the of the docker and prometheus.

Step47:- Now go to grafana homepage ❼ configurations ❼ Data sources

Step48:- Now click on add Data sources ❼ Prometheus and give ipaddress:9090 and click on save and test
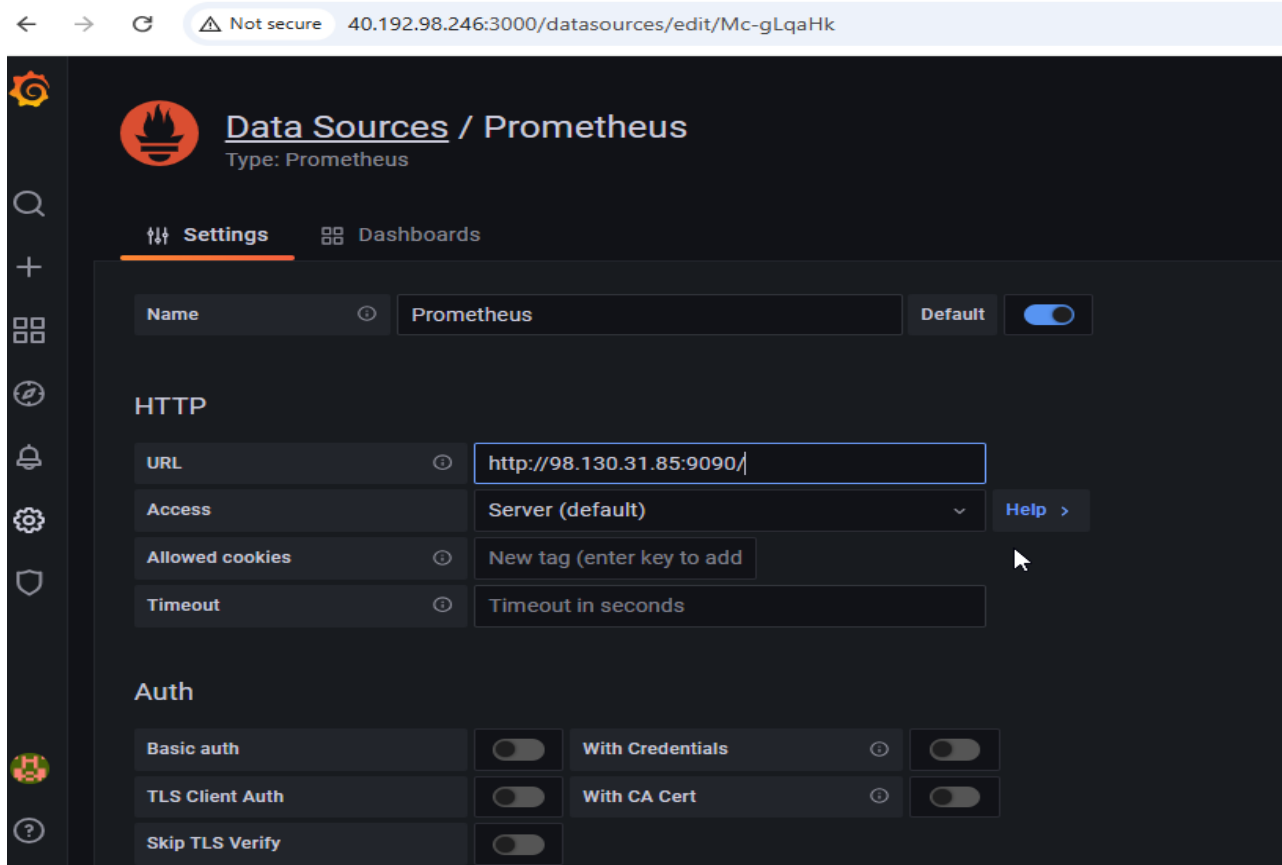
Step49:- Now click on Dash board ❼ add new panel
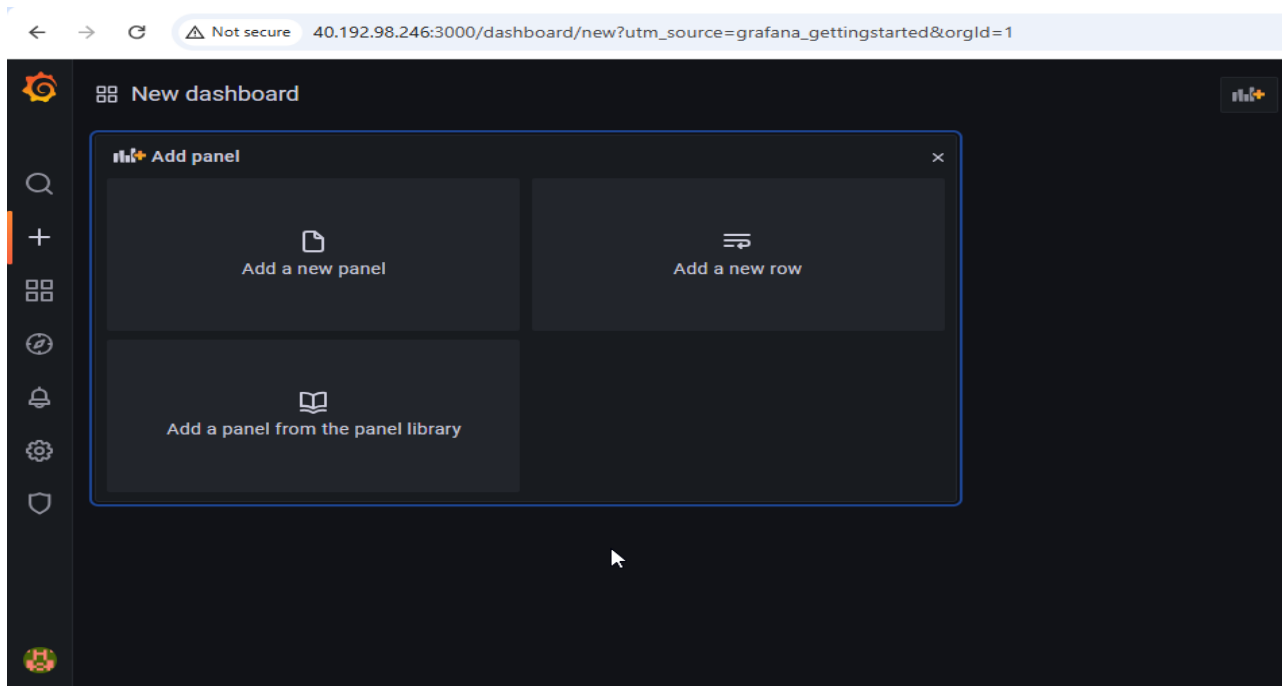
Step50:- Now in the metrics browser give engine daemon container states containers{state="running"} and you will see the result that same as in the metrics from the browser
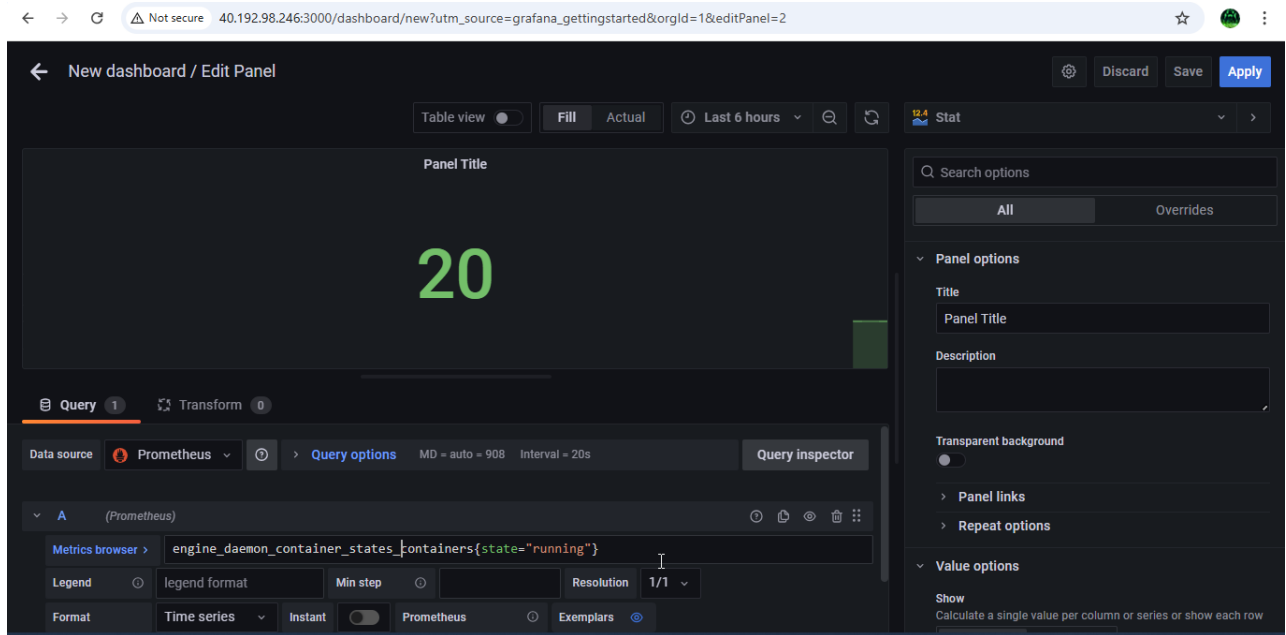
engine_daemon_container_states_containers{state="stopped"}
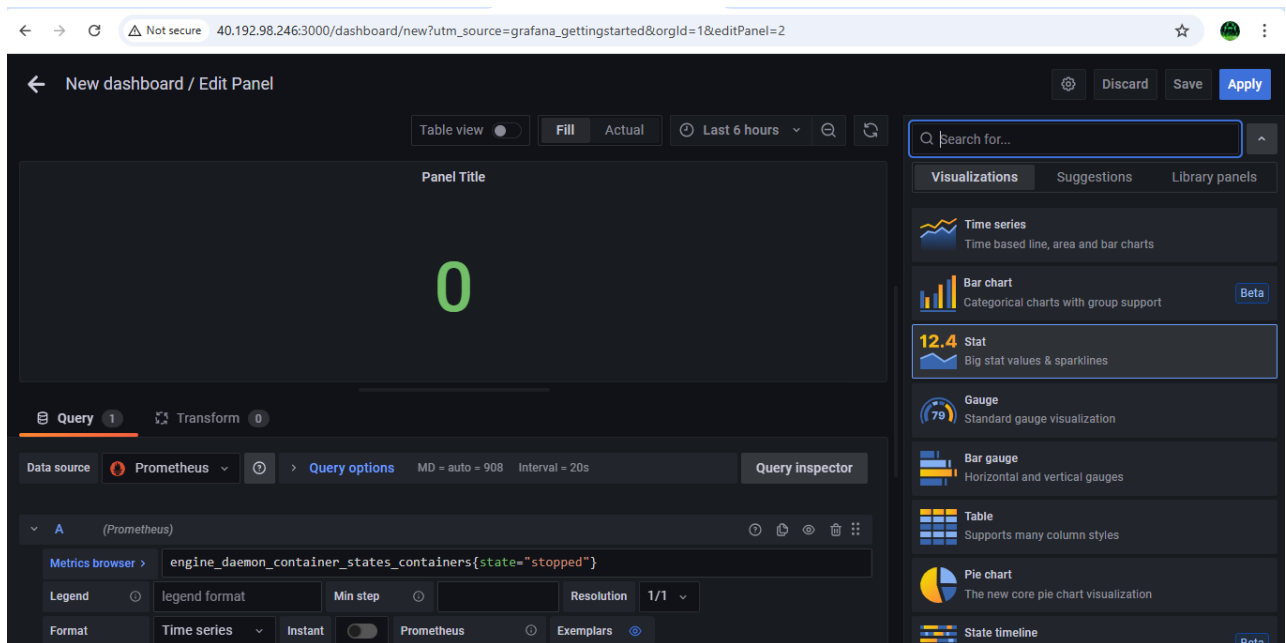
Step51:- The values shown in the panel must be equal to the that of shown in the docker stats, here the container which we created is in exited state so it is showing as stopped state in stats

```
# TYPE engine_daemon_container_states_containers gauge
engine_daemon_container_states_containers{state="paused"} 0
engine_daemon_container_states_containers{state="running"} 20
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
```

Step52:- Now go and check the containers running and in stopped state again and check the details again in the stats

```
root@ip-10-0-1-242:/home/ubuntu/prometheus-2.34.0.linux-amd64#
root@ip-10-0-1-242:/home/ubuntu/prometheus-2.34.0.linux-amd64# docker ps -a
CONTAINER ID   IMAGE          COMMAND                CREATED         STATUS          PORTS     NAMES
91f602e0f4e0   75392e3500e3   "start_runit"          20 minutes ago  Up 20 minutes             k8s_calico-node_calico-node-c
qnbf_kube-system_42036f07-3a4e-40ec-a1f7-66ef84ccee96_2
ede9e73e8764   cbb01a7bd410   "/coredns -conf /etc…" 20 minutes ago  Up 20 minutes             k8s_coredns_coredns-76f75df57
4-4j5rw_kube-system_97b8dd3f-d081-421f-b8d9-befe66d0287a_2
c38609c7b1b2   cbb01a7bd410   "/coredns -conf /etc…" 20 minutes ago  Up 20 minutes             k8s_coredns_coredns-76f75df57
4-m7lj_kube-system_51f5b119-3bdd-4934-a757-2523fceedd47_2
7c24457a16c5   registry.k8s.io/pause:3.6   "/pause"  20 minutes ago  Up 20 minutes             k8s_POD_coredns-76f75df574-ms
7lj_kube-system_51f5b119-3bdd-4934-a757-2523fceedd47_12
0ead39a58da8   registry.k8s.io/pause:3.6   "/pause"  20 minutes ago  Up 20 minutes             k8s_POD_coredns-76f75df574-4j
5rw_kube-system_97b8dd3f-d081-421f-b8d9-befe66d0287a_10
43c64d9c04e8   f9c3c1813269   "/usr/bin/kube-contr…" 20 minutes ago  Up 20 minutes             k8s_calico-kube-controllers_c
alico-kube-controllers-9d57d8f49-2bdnj_kube-system_693f5004-d190-4242-97b0-8562f49b7787_2
24abb70777e5   registry.k8s.io/pause:3.6   "/pause"  20 minutes ago  Up 20 minutes             k8s_POD_calico-kube-controlle
rs-9d57d8f49-2bdnj_kube-system_693f5004-d190-4242-97b0-8562f49b7787_9
aa4590fbd454   9ea0bd82ed4f   "kube-scheduler --au…" 20 minutes ago  Up 20 minutes             k8s_kube-scheduler_kube-sched
uler-ip-10-0-1-242_kube-system_d04a4ee1c17a254de8e4627f63052e36_2
af4deaf0a653   registry.k8s.io/pause:3.6   "/pause"  20 minutes ago  Up 20 minutes             k8s_POD_kube-scheduler-ip-10-
0-1-242_kube-system_d04a4ee1c17a254de8e4627f63052e36_2
5f2b8c2c419f   f44c6888a2d2   "kube-apiserver --ad…" 20 minutes ago  Up 20 minutes             k8s_kube-apiserver_kube-apise
rver-ip-10-0-1-242_kube-system_56d40a089d2264f5e98d93627e0b4d60_2
79d41470649d   255ec253085f   "/bin/node_exporter …" 20 minutes ago  Up 20 minutes             k8s_node-exporter_prometheus-
prometheus-node-exporter-stf79_default_9b40ea04-45bd-48fd-9189-88363a2ec373_2
8ff932a49856   f71614796eb7   "/usr/local/bin/kube…" 20 minutes ago  Up 20 minutes             k8s_kube-proxy_kube-proxy-rzk
wd_kube-system_9a75c0a7-73d8-4733-a724-5296bb5dae86_2
74a2a8abe308   b0cdcf76ac8e   "kube-controller-man…" 20 minutes ago  Up 20 minutes             k8s_kube-controller-manager_k
ube-controller-manager-ip-10-0-1-242_kube-system_e746a0c1e0e6fc16ddf7524c7c76f522_2
f388a697c493   a9e7e6b294ba   "etcd --advertise-cl…" 20 minutes ago  Up 20 minutes             k8s_etcd_etcd-ip-10-0-1-242_k
ube-system_63c7572858fc6bb1e69e5855a771c2b8_2
ab3d7d5df1ec   registry.k8s.io/pause:3.6   "/pause"  20 minutes ago  Up 20 minutes             k8s_POD_calico-node-cqnbf_kub
e-system_42036f07-3a4e-40ec-a1f7-66ef84ccee96_2
1a59792f79a1   registry.k8s.io/pause:3.6   "/pause"  20 minutes ago  Up 20 minutes             k8s_POD_kube-proxy-rzkwd_kube
-system_9a75c0a7-73d8-4733-a724-5296bb5dae86_2
3fe530e885d0   registry.k8s.io/pause:3.6   "/pause"  20 minutes ago  Up 20 minutes             k8s_POD_kube-apiserver-ip-10-
0-1-242_kube-system_56d40a089d2264f5e98d93627e0b4d60_2
55f165720145   registry.k8s.io/pause:3.6   "/pause"  20 minutes ago  Up 20 minutes             k8s_POD_kube-controller-manag
```
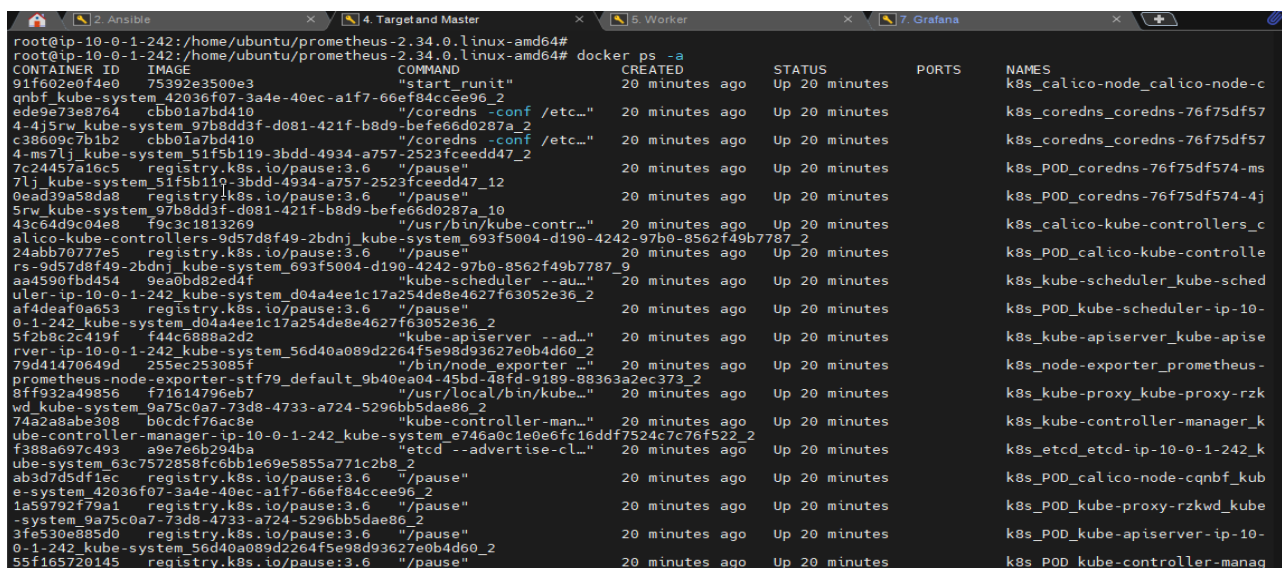
Step53:- This is how we monitor the health of a container automatically and visualizing the report using Prometheus and Grafana.