**Capstone Project**

Insurance Project

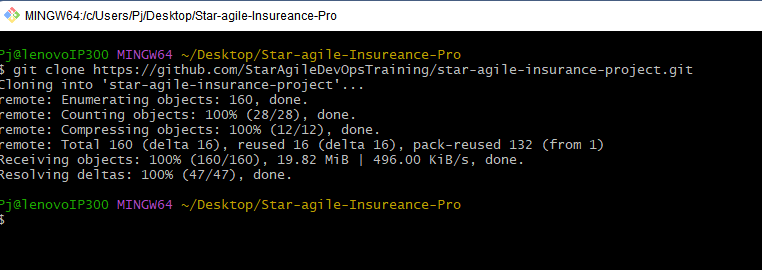
By:- Jonna Padmarao

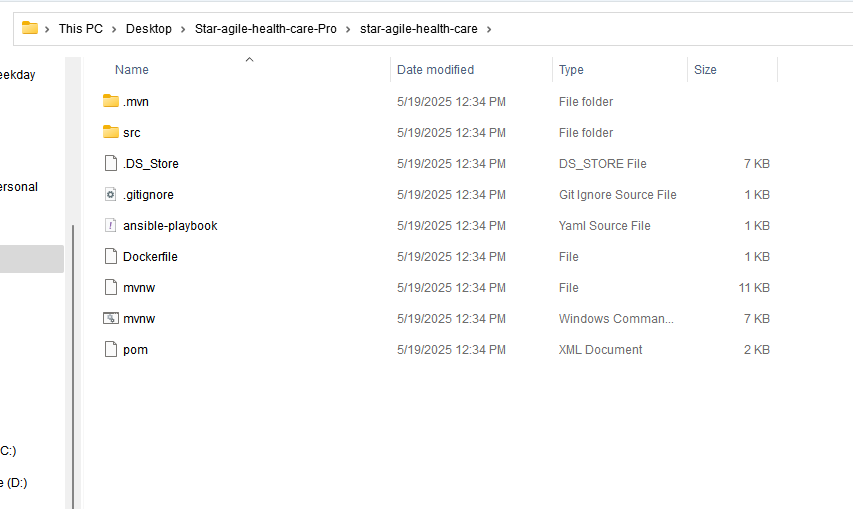
Source-code URL :- <https://github.com/pj013525/star-agile-project-2.git>

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](https://github.com/StarAgileDevOpsTraining/star-agile-banking-finance)-insurance-project.gitto get the project code in to that folder

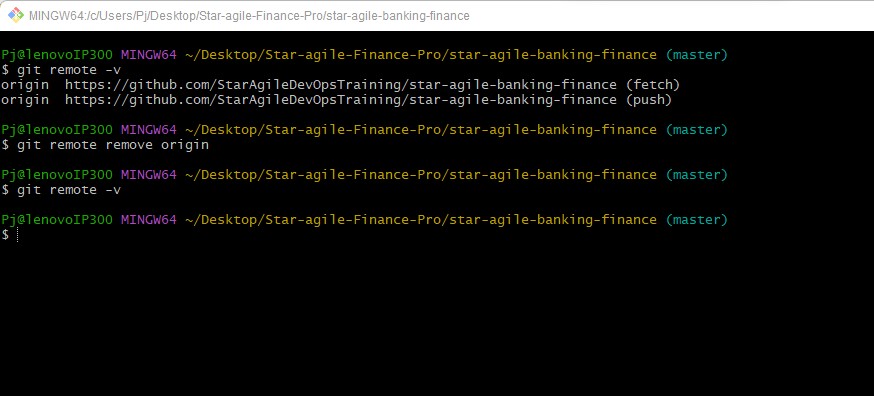




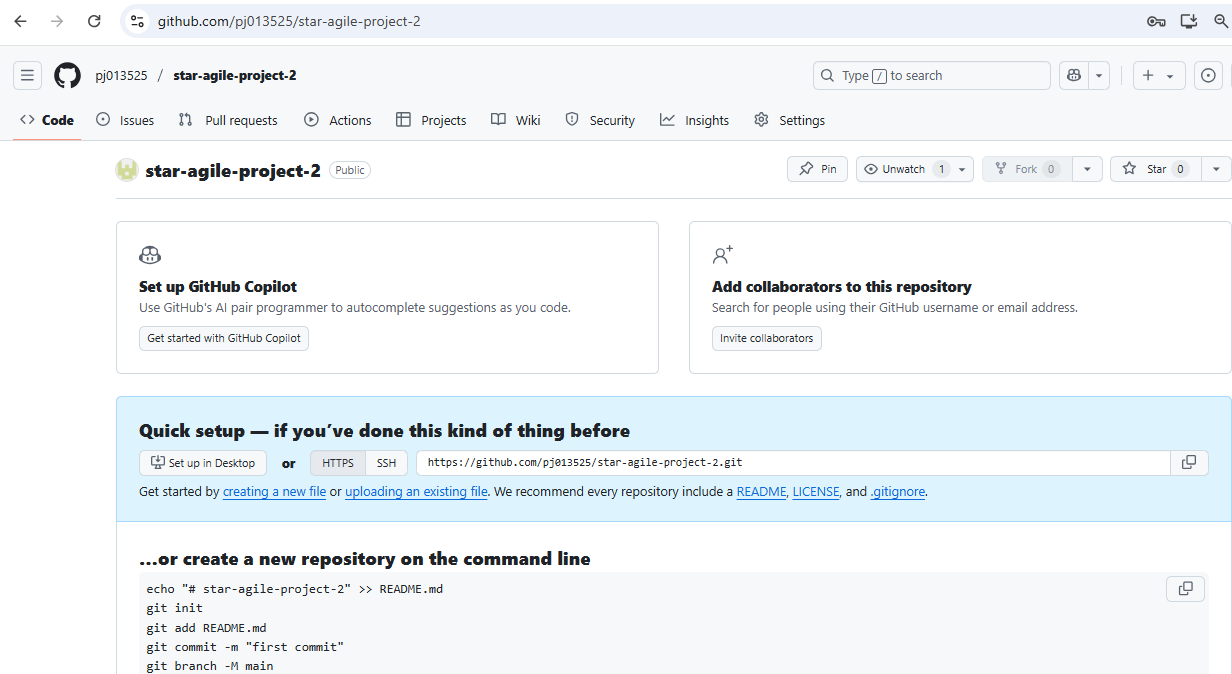
Step3:- Now go to the folder that we get from git clone and again open git bash there and check the origin and remove that origin

git remote -v --> To get origin list

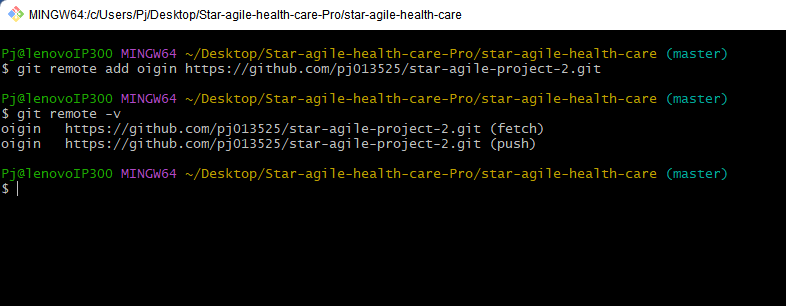
git remote remove origin to remove the origin



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

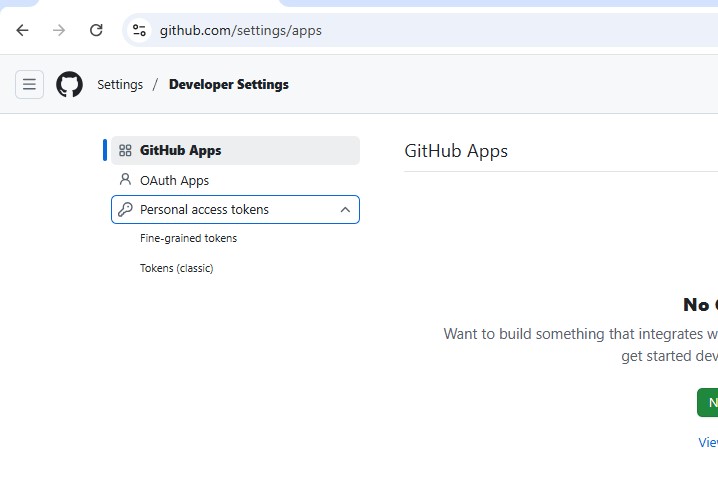


Step5:- Now again go to the gitbash and add this git repo url in the project by using git remote add origin <git-repo-url> and verify

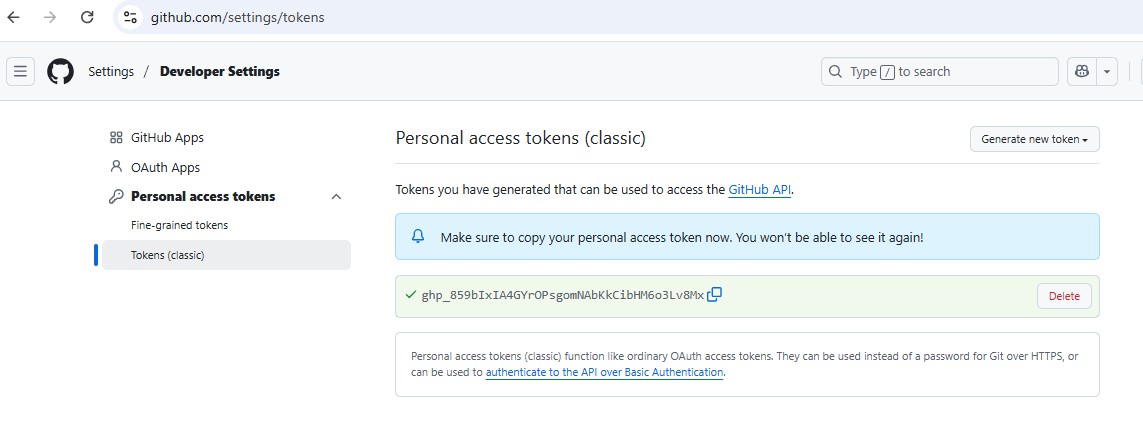


Step6:- Now again go to github → Profile setting →Developer settings

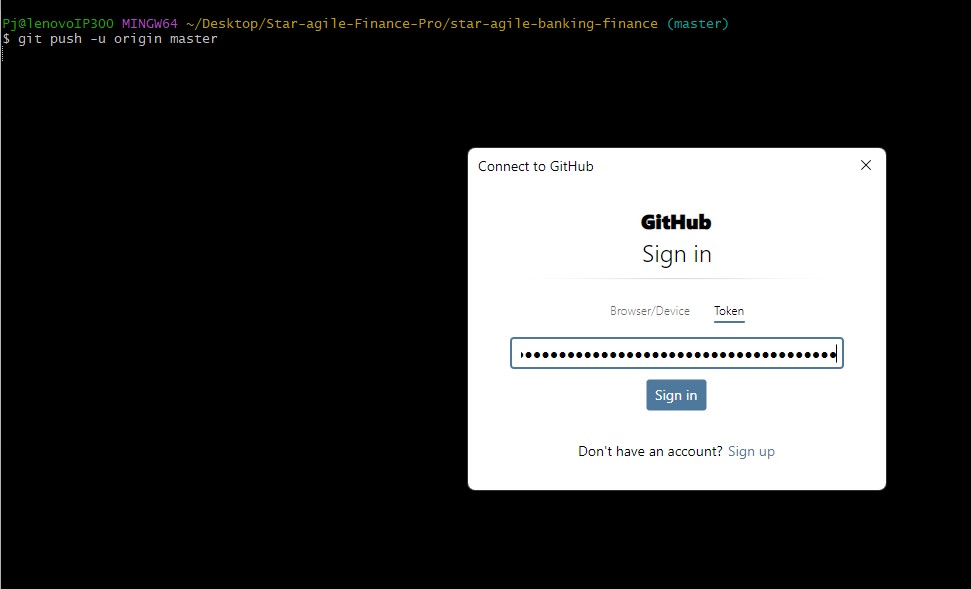
→ Personal access token Tokens(classic) →Generate new token



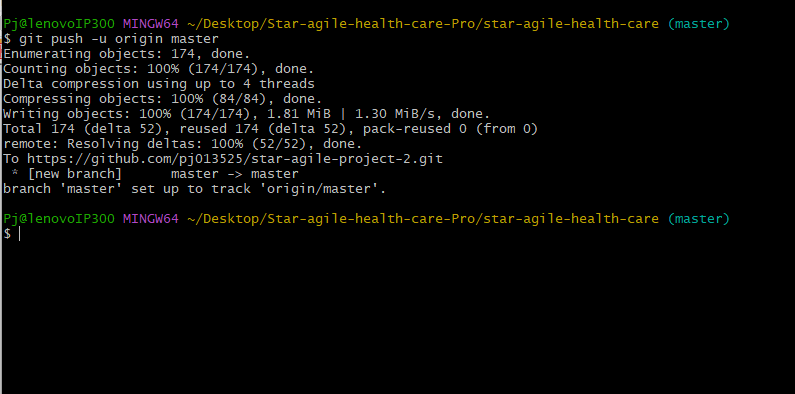
Step7:- Now a token will be generated , copy this token that generated since it is only available for one time only



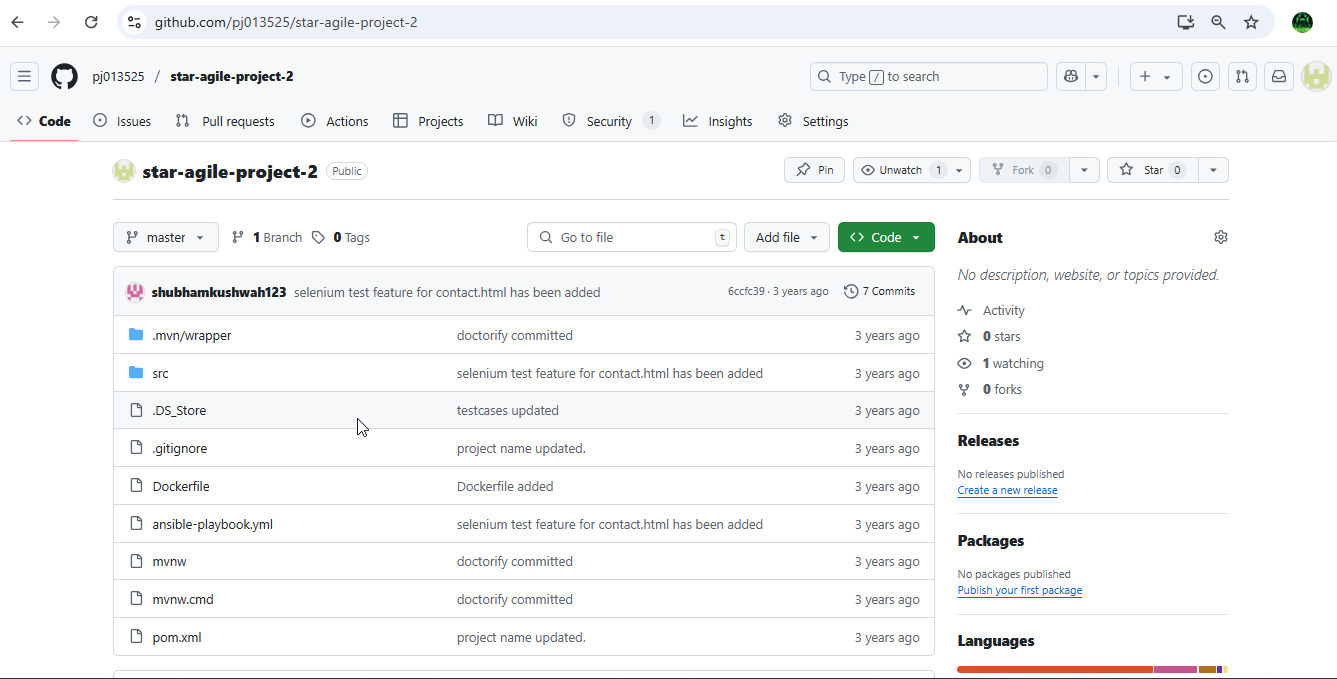
Step8:- Now give link this the remote repo with gitbash using this token git push -u origin master and paste the token the copied from the github and press sign in



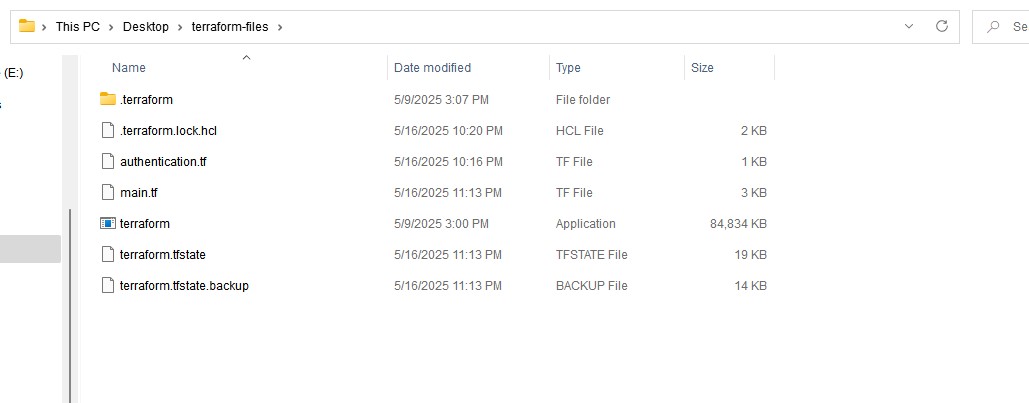
Step9:- Now the master branch will be set to our repo by default



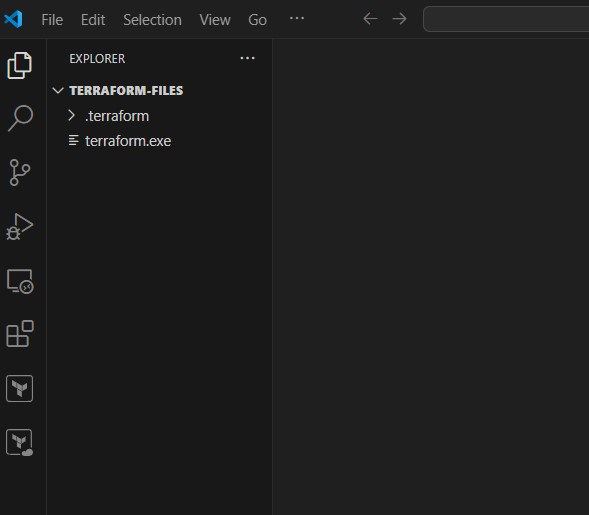
Step10:- Now go to the github repo and you will see the source code in that repo



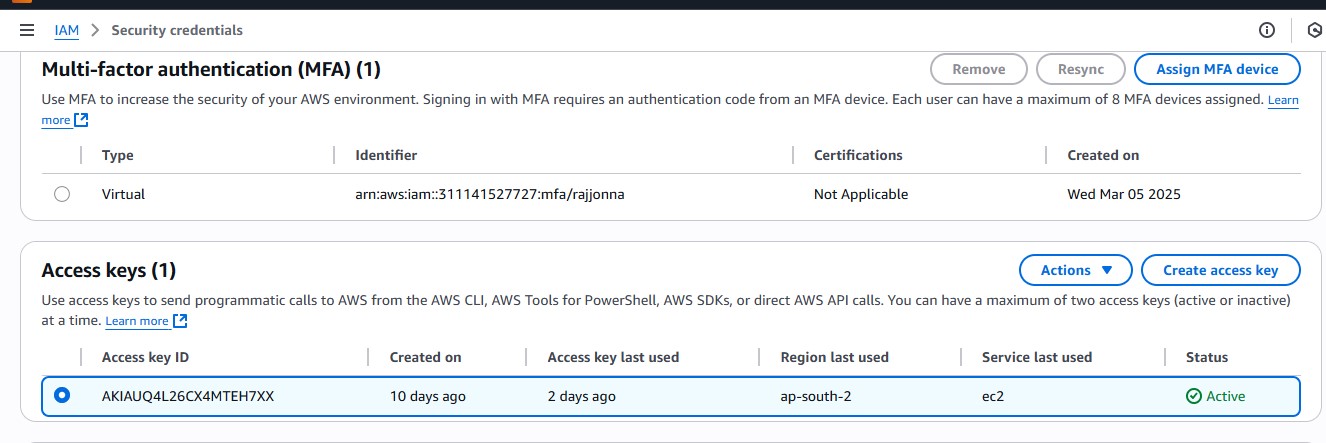
Step11:- Now create an instance using terraform as Iaac , and for that create a folder on desktop and go to browser download terraform for windows then a terraform application will be generated , now copy this application in to that folder and save



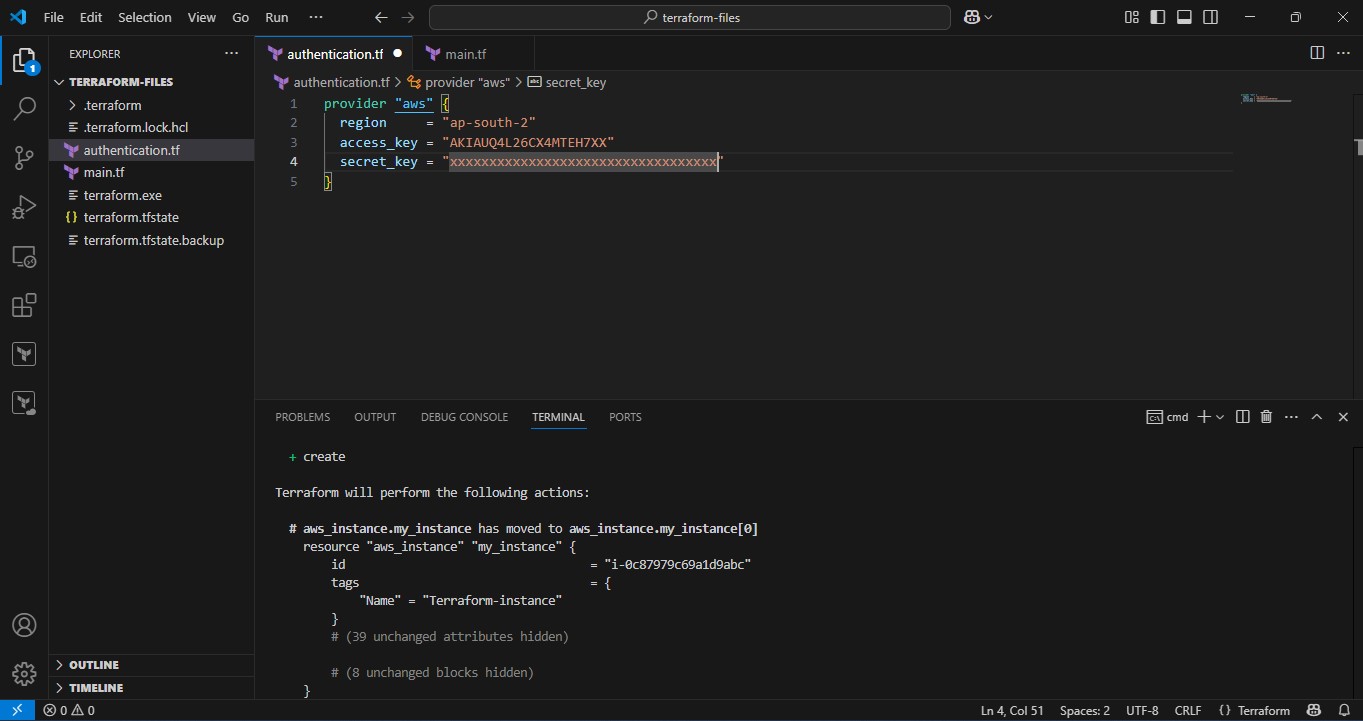
Step12:- open visual studio code and go to terraform folder



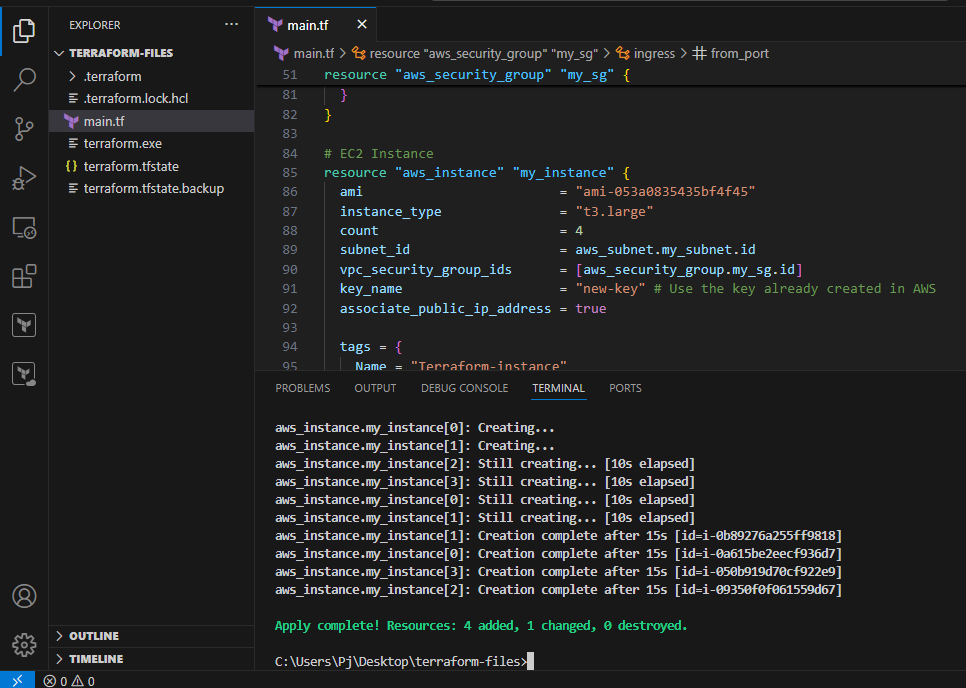
Step13:- Now create a file authentication.tf and give the provider and for that select the region in which you want to launch the server and go to aws account and go to profile → credentials and go to access keys



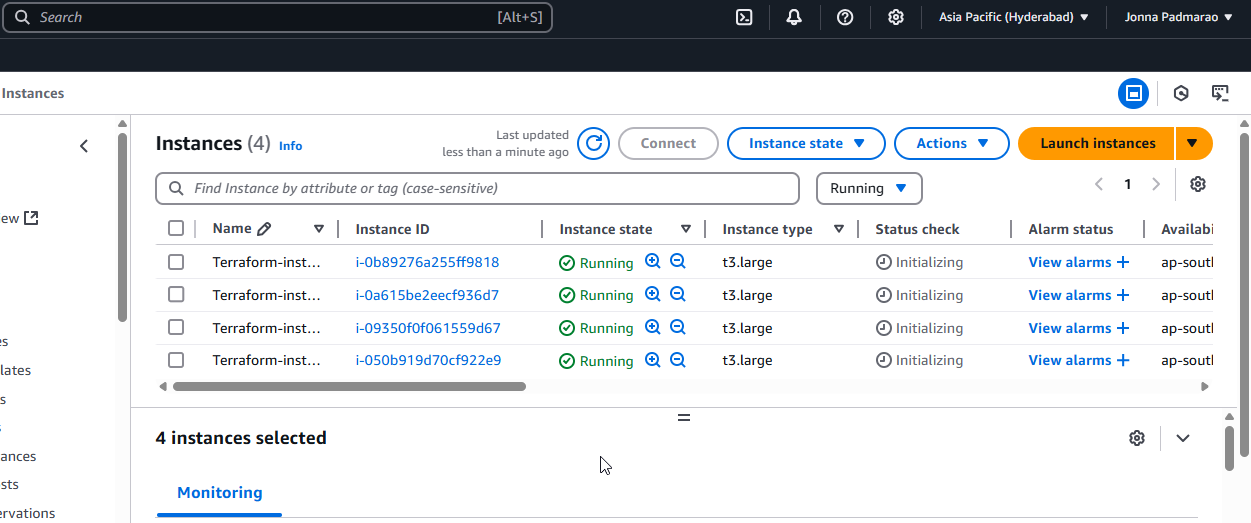
Step14:- now copy this access key details and paste it in this authentication.tf file and initialize it



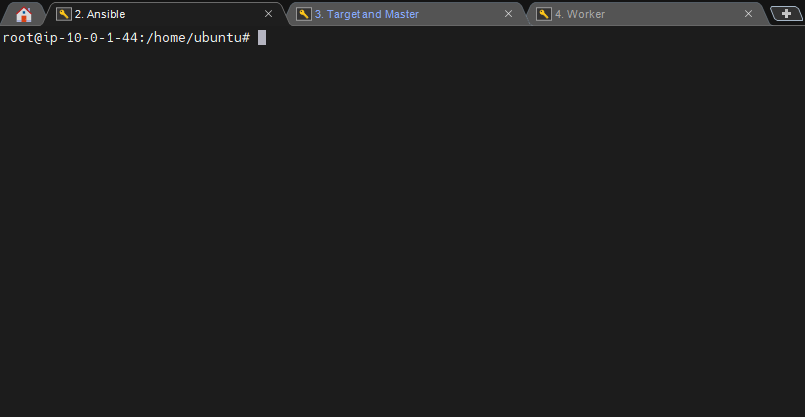
Step15:- After it is successful now create a new file main.tf and give resources details to create instance



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

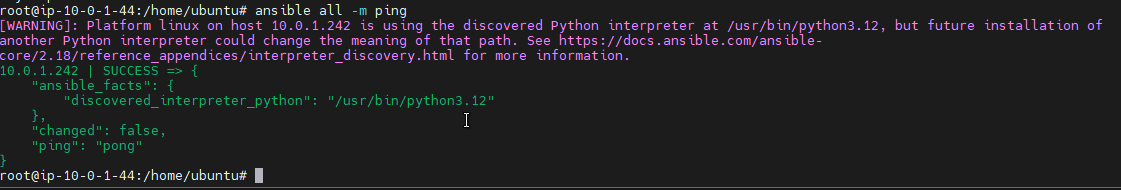


Step17:- Now connect to Ansible and target and master and worker servers using Mobaxterm agent and launch an instance

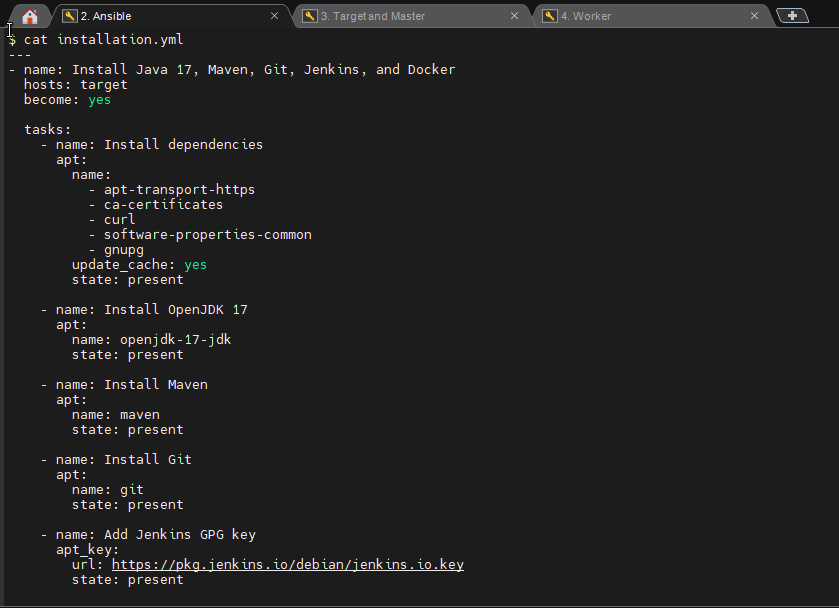


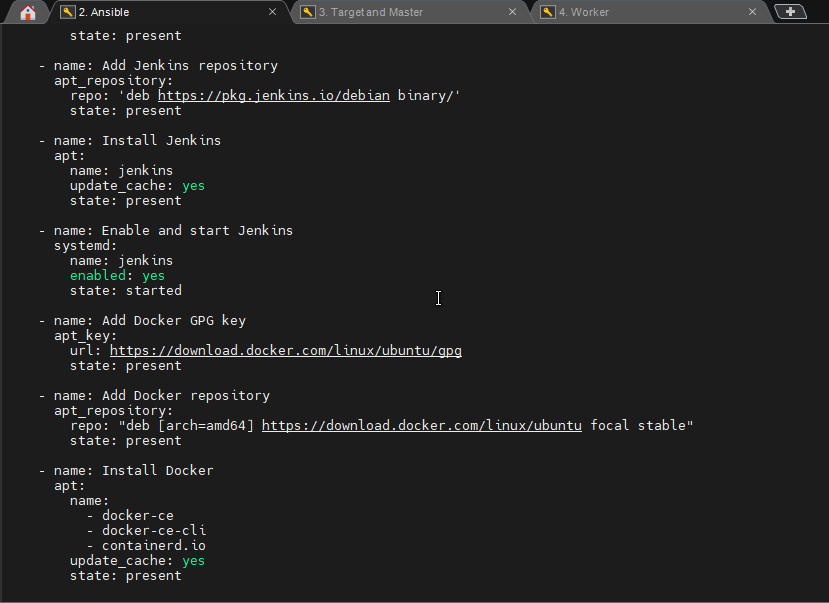
Step18:- Now install Ansible in Ansible server and connect this server with the Target and master sever and enable All traffic in the security group of this server



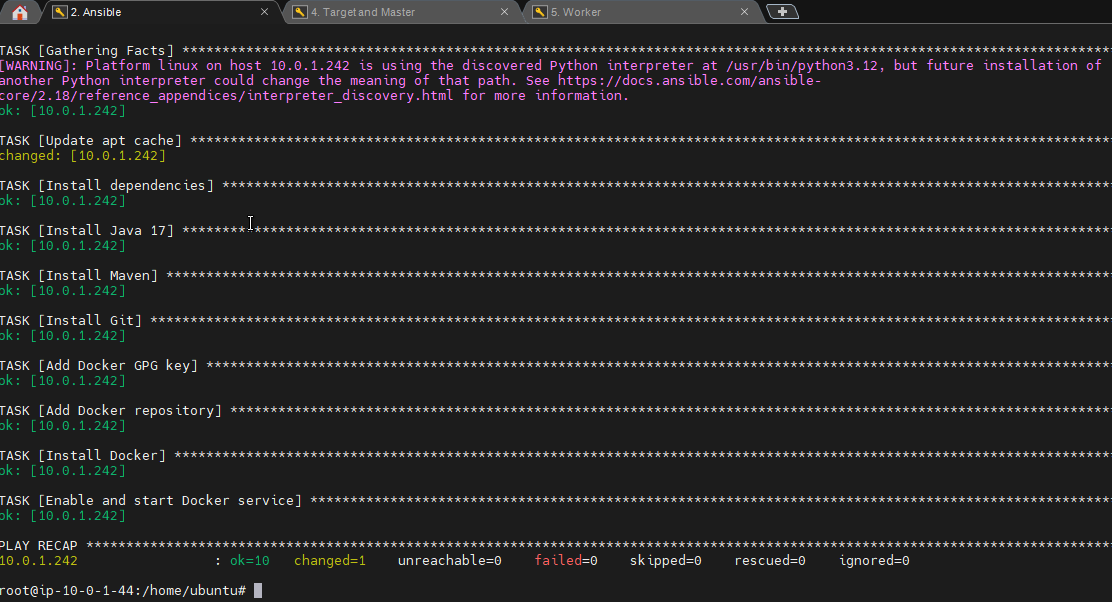


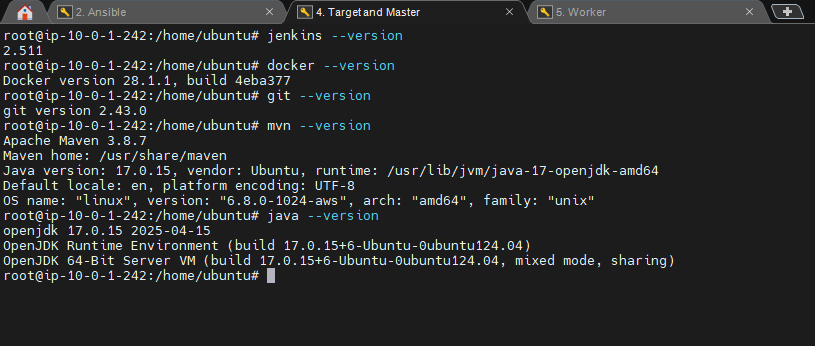
Step19:- Now install java, maven, docker, jenkins in the target and master server using Ansible sever





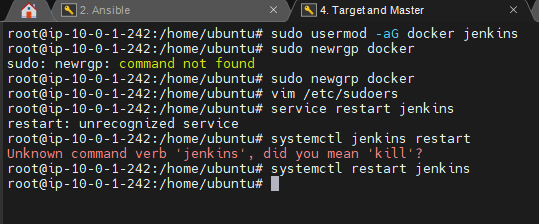
Step20:- Now go to target and master node and verify the packages



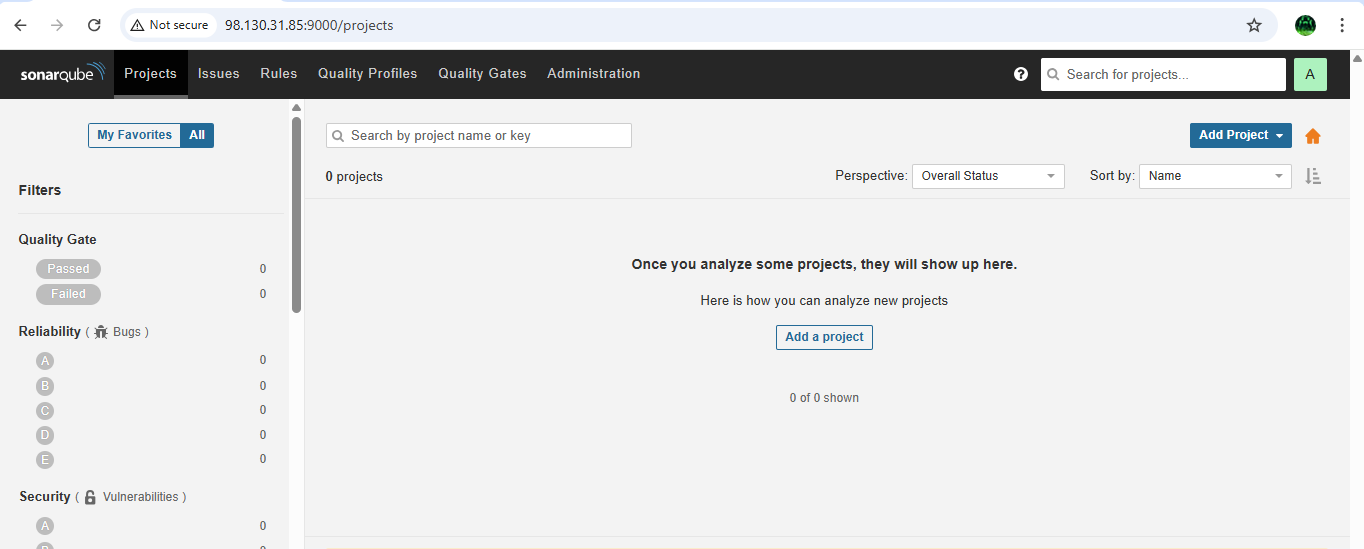


Step21:- Now add Jenkins group to docker and give root permissions to the Jenkins user in the sudoers file as under root give

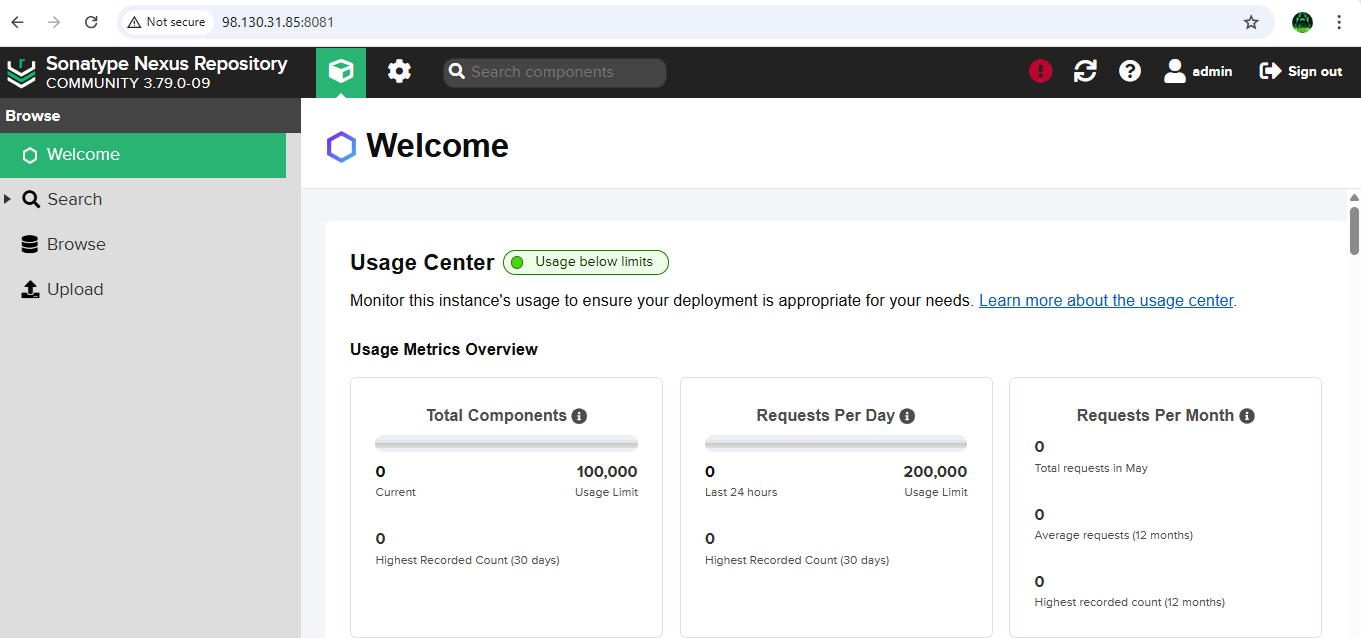
jenkins ALL=(ALL:ALL) NOAPSSWD: ALL restart the jenkins



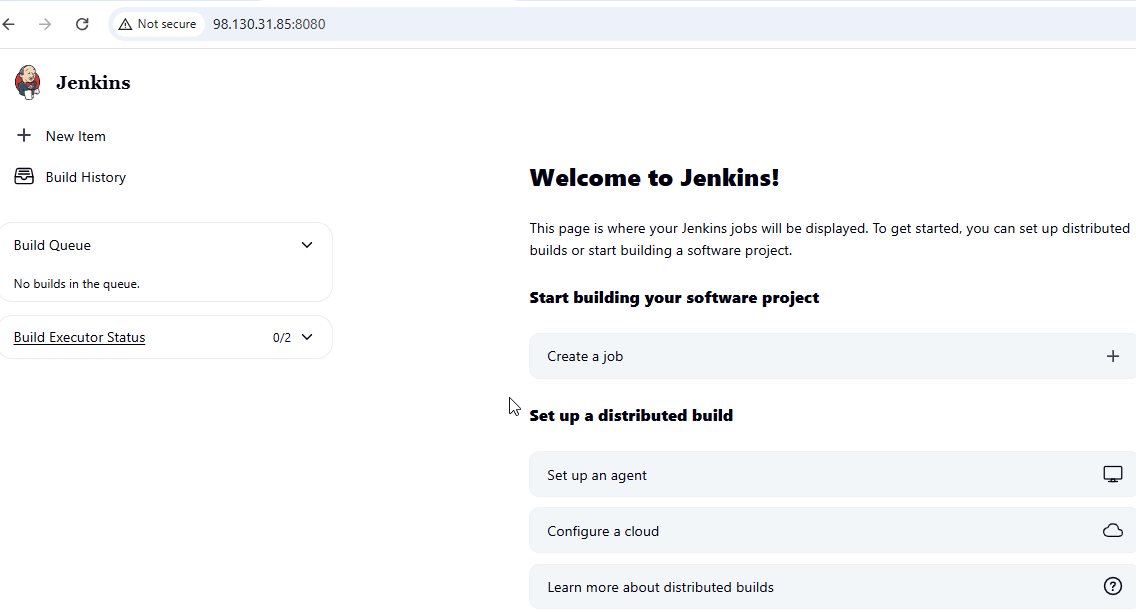
Step22:- Now I am installing sonarqube in the target for code Quality analysis for checking any type of bugs , vulnerabilities in the code and logging into the sonarqube home page using browser with ip:9000



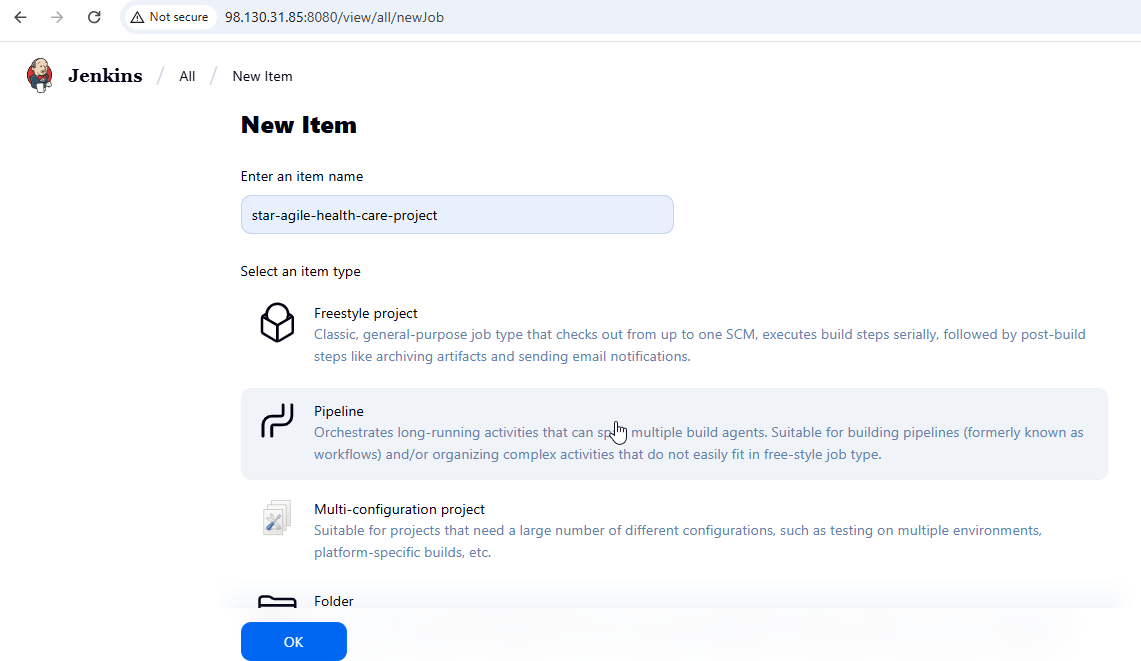
Step23:- Now i am installing nexus tool in the target node to store the artifacts that generated from the maven build



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

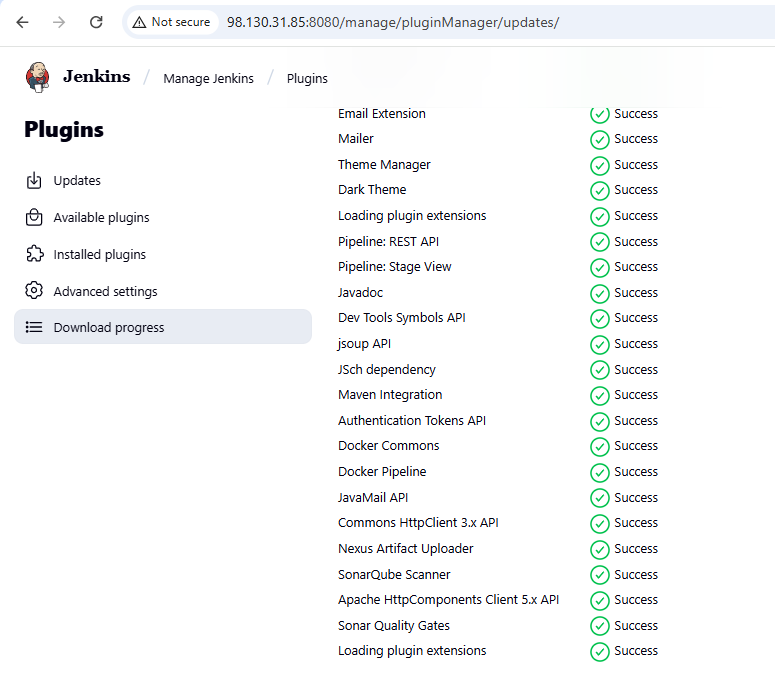
Docker Pipeline Plugin  SonarQube scanner

Credentials Binding Plugin Sonar quality gates

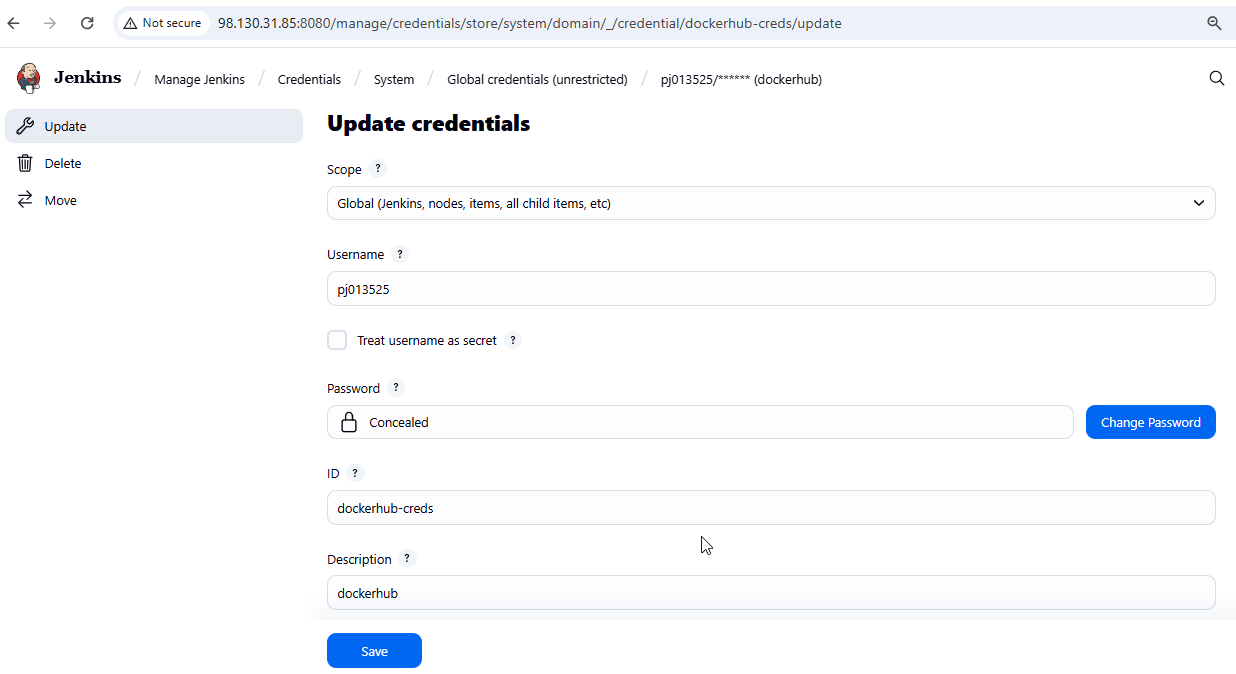
Docker Commons Plugin

Pipeline: GitHub

Maven Integration Plugin

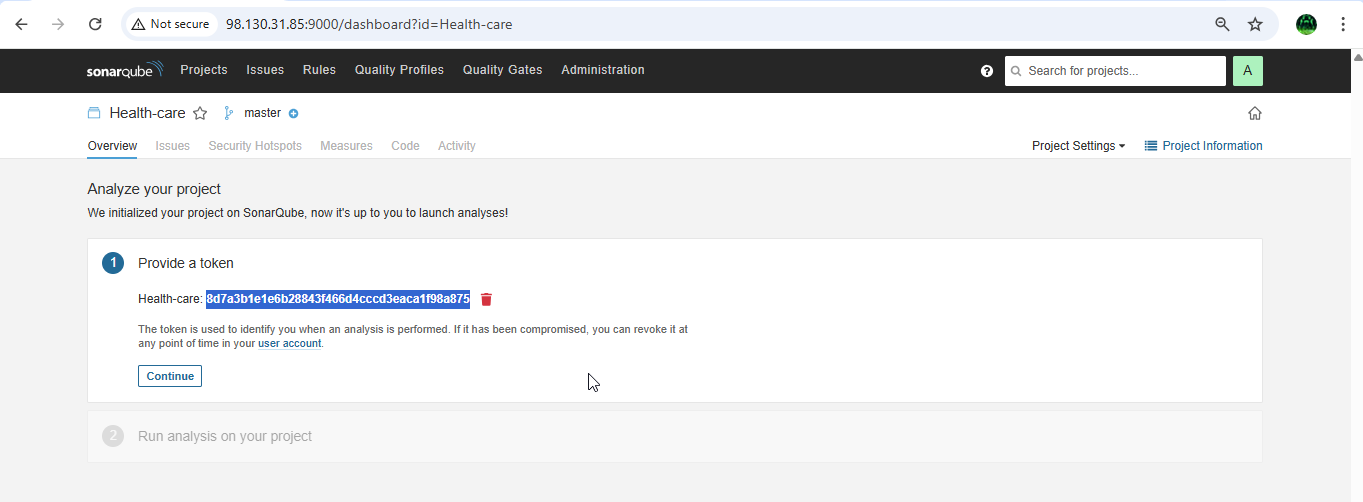


Step27:- Now in dash board → manage Jenkins → credentials → global → add credentials and give dockerhub user name and password and click on create

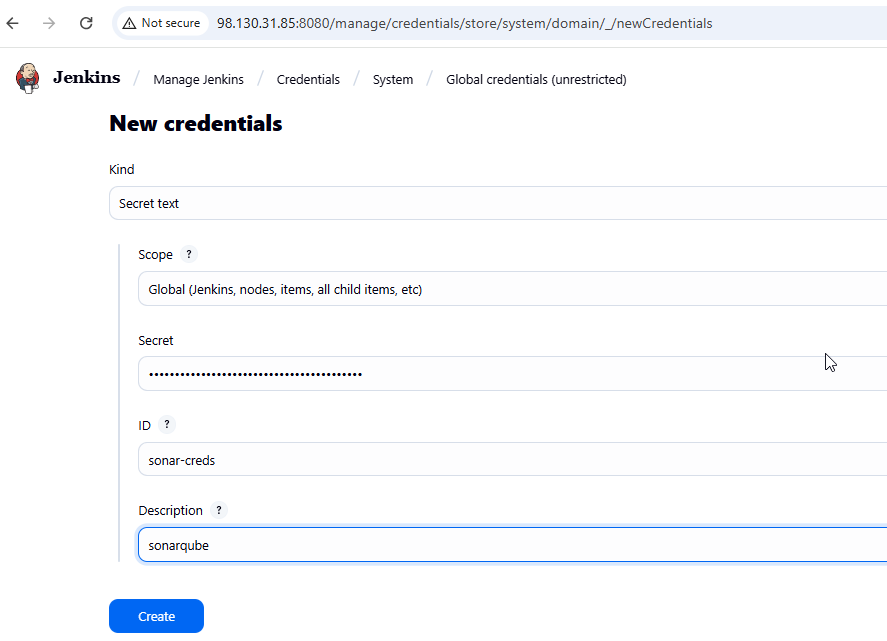


Step28:- Now in jenkins configure the sonarqube and nexus tools

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

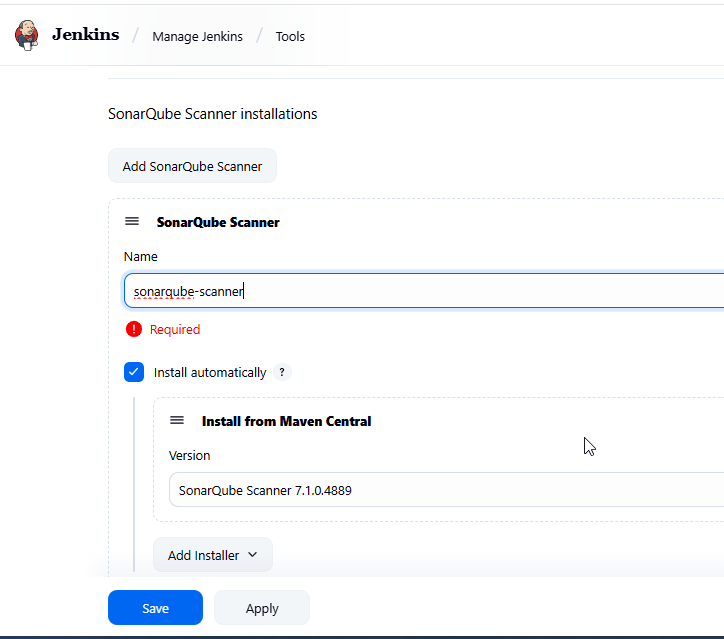


Step29:- Lets configure the SonarQube credentials in Jenkins and for that manage-jenkins ==> Credentials ==> global ==> Add credentials==> select secret text and give the text that copied form the sonarqube home page

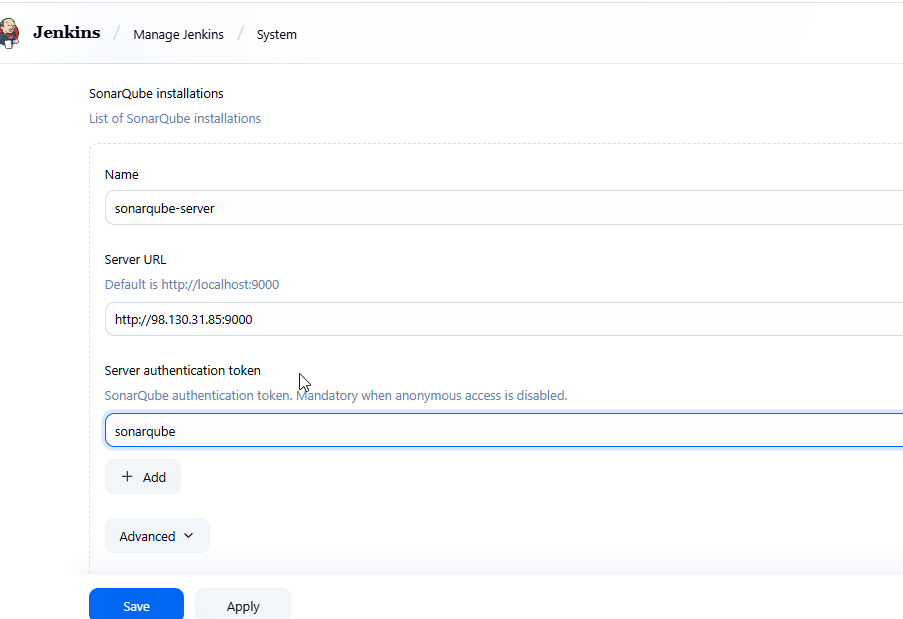


Step30:- Similarly add sonarqube tool also as

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.

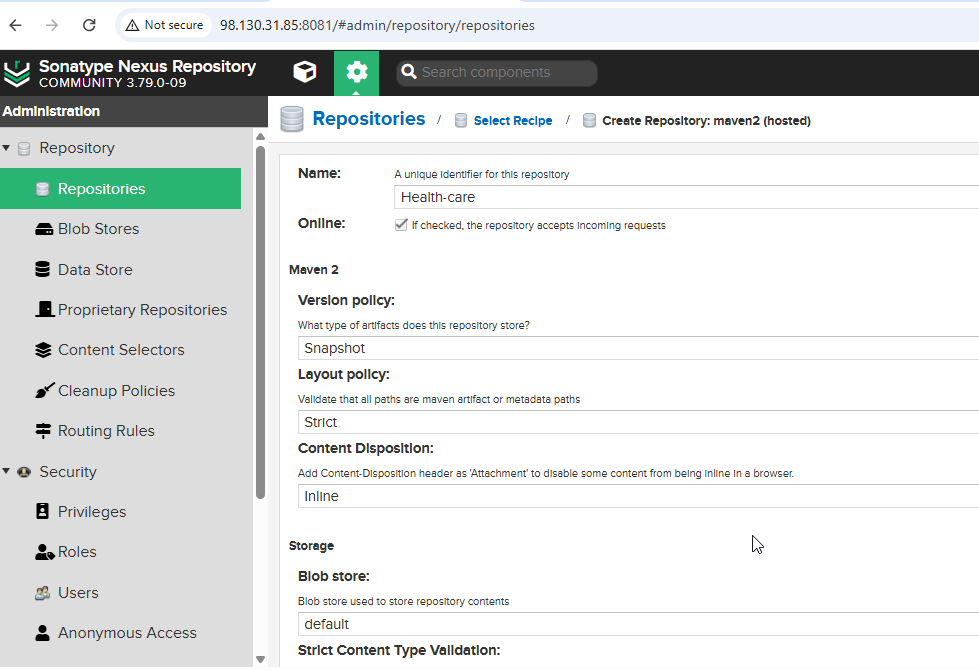


Step31:- Now configure the sonarqube in the system of the jenkins and for that jenkins ----> Manage Jenkins ----> System Configuration ----> System ----> Scroll down to 'SonarQube servers' ----> 'Check' environment variables, Add SonarQube ----> Name: sonarqube, Server URL: <SonarQube URL> [only upto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

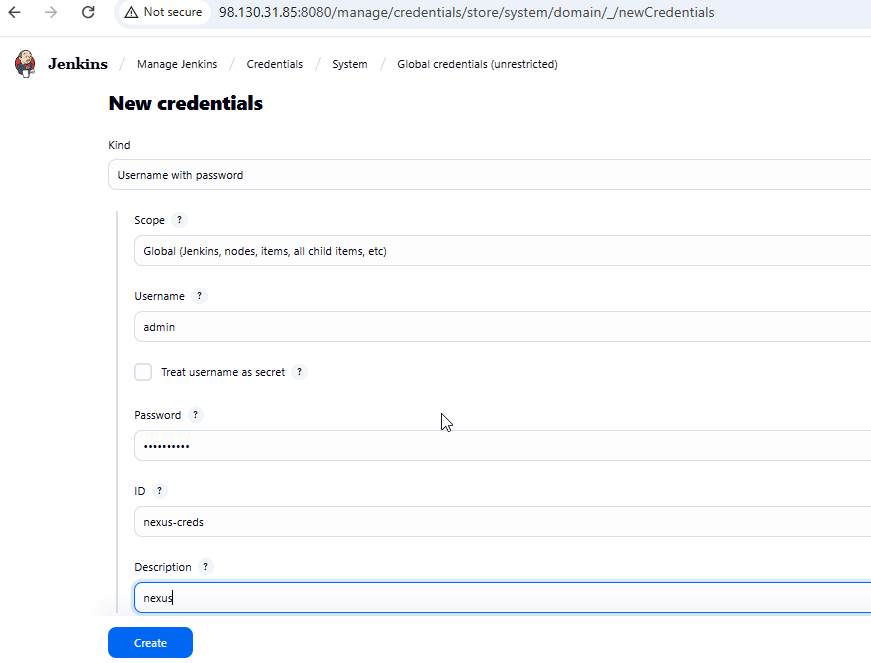


Step32:- Now configure the nexus in the jenkins for that first

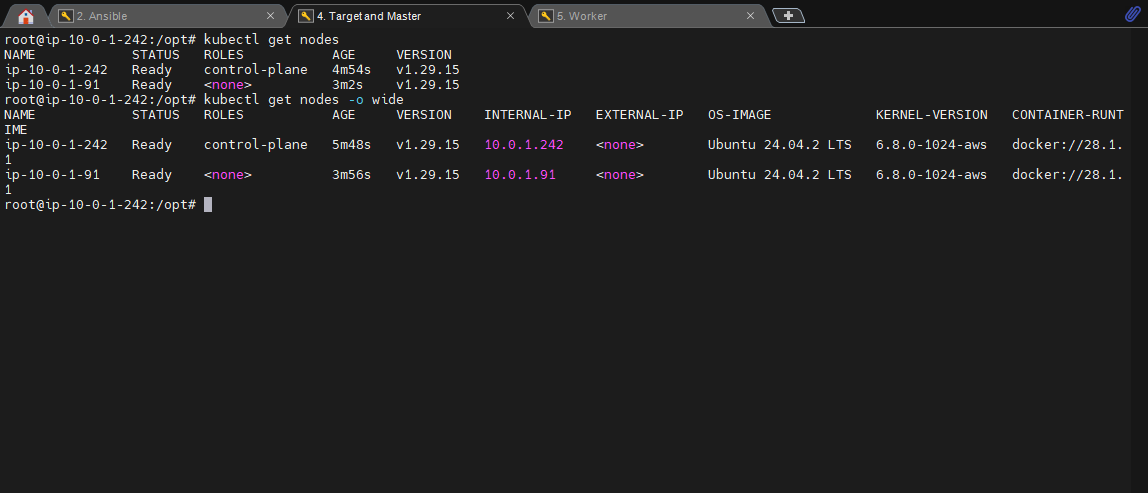
Go to Nexus tab in browser --> Click on settings icon ----> Click on 'repositories' ----> Create repo ----> Scroll down and Click on 'maven2 hosted' ----> Name: Health-care, Version policy: Snapshot, Layout policy: Strict, Content disposition: Inline, Blob store:default, Deployment policy: Allow redeploy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



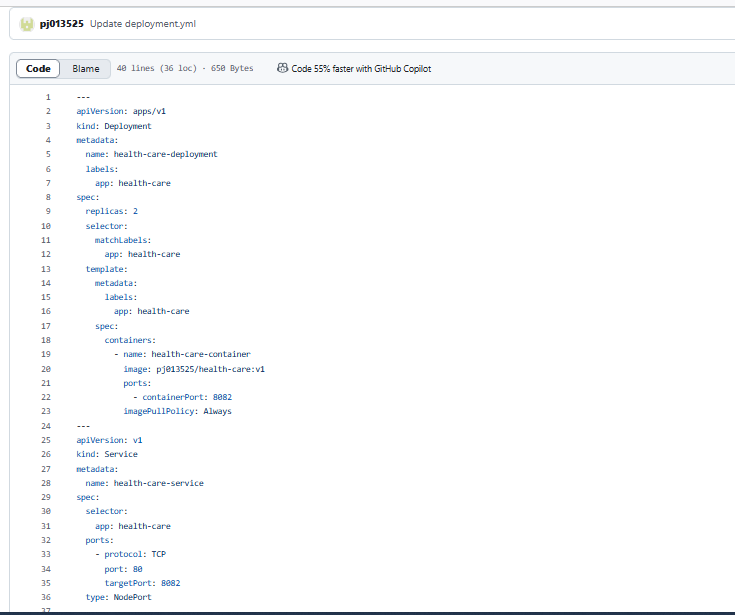
Step33:- Now configure the nexus credentials in the jenkins dash board through Jenkins ----> Manage Jenkins ----> Security ----> Credentials ----> Click on 'global' ----> Add creds ----> Kind: Username with Password, Scope: Global, Username: admin, Password: <Enter The Nexus> , ID: nexus-creds, Description: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

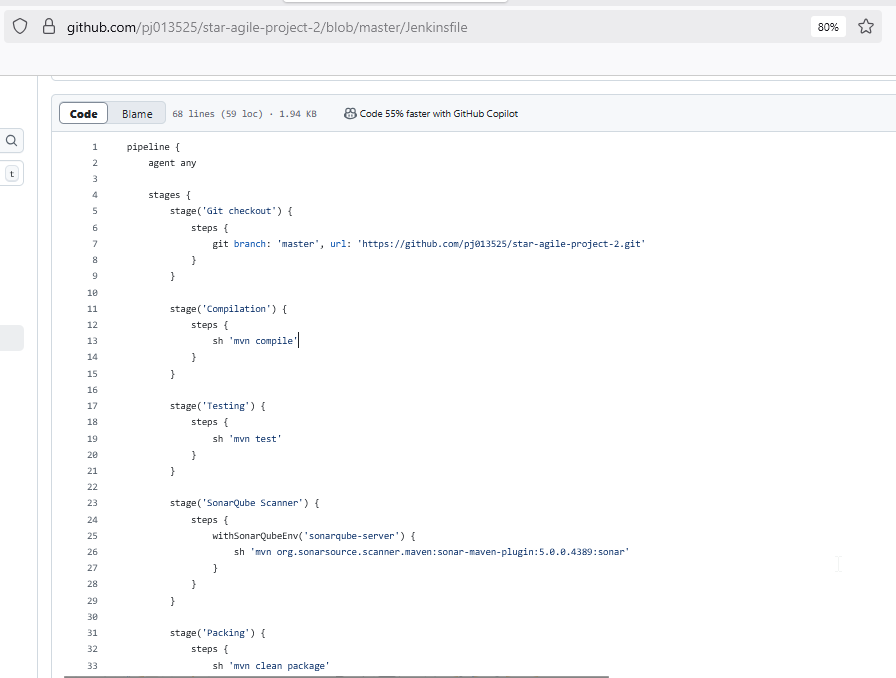


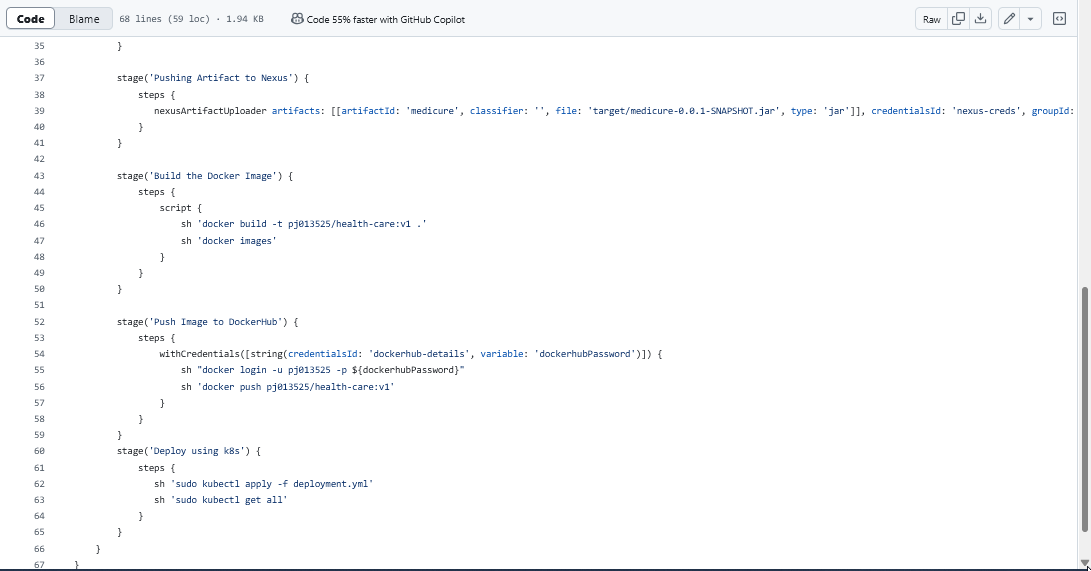
Step34:- Now go to Target and Master instance and install Kubernetes and configure with Worker node and verify



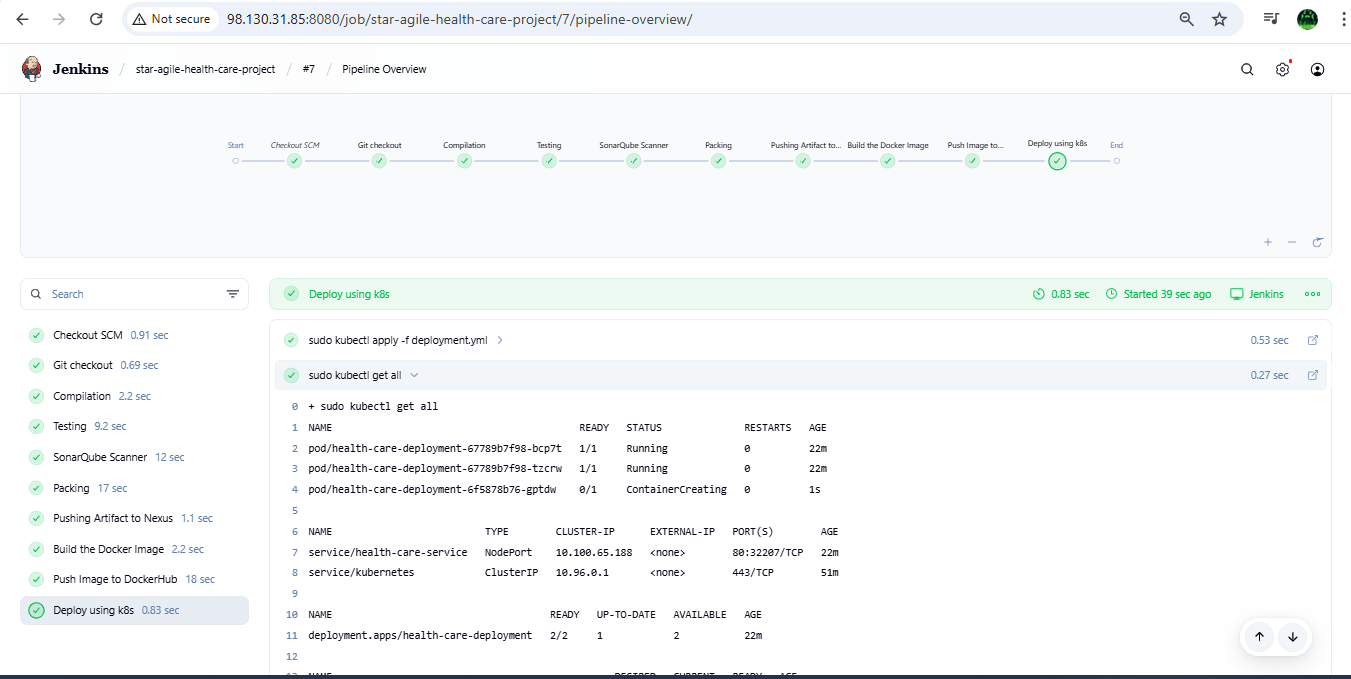
Step35:- Now go to Project repo in the github and there add deployment.yml and jenkinsfile to perfomr the pipeline in the Jenkins.

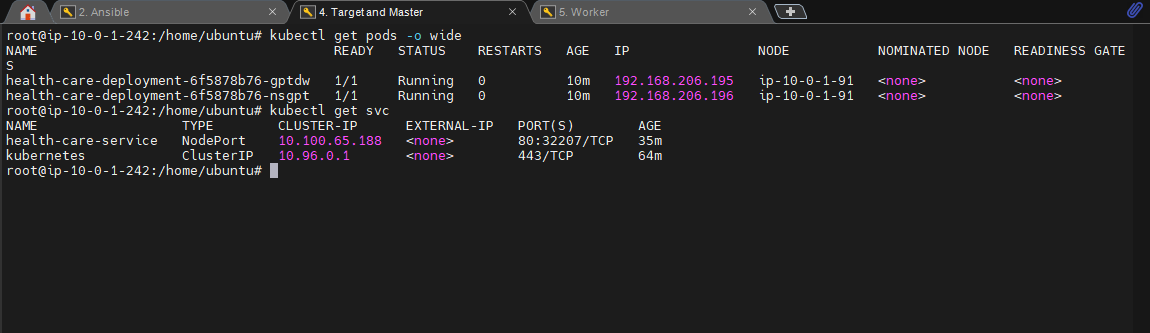




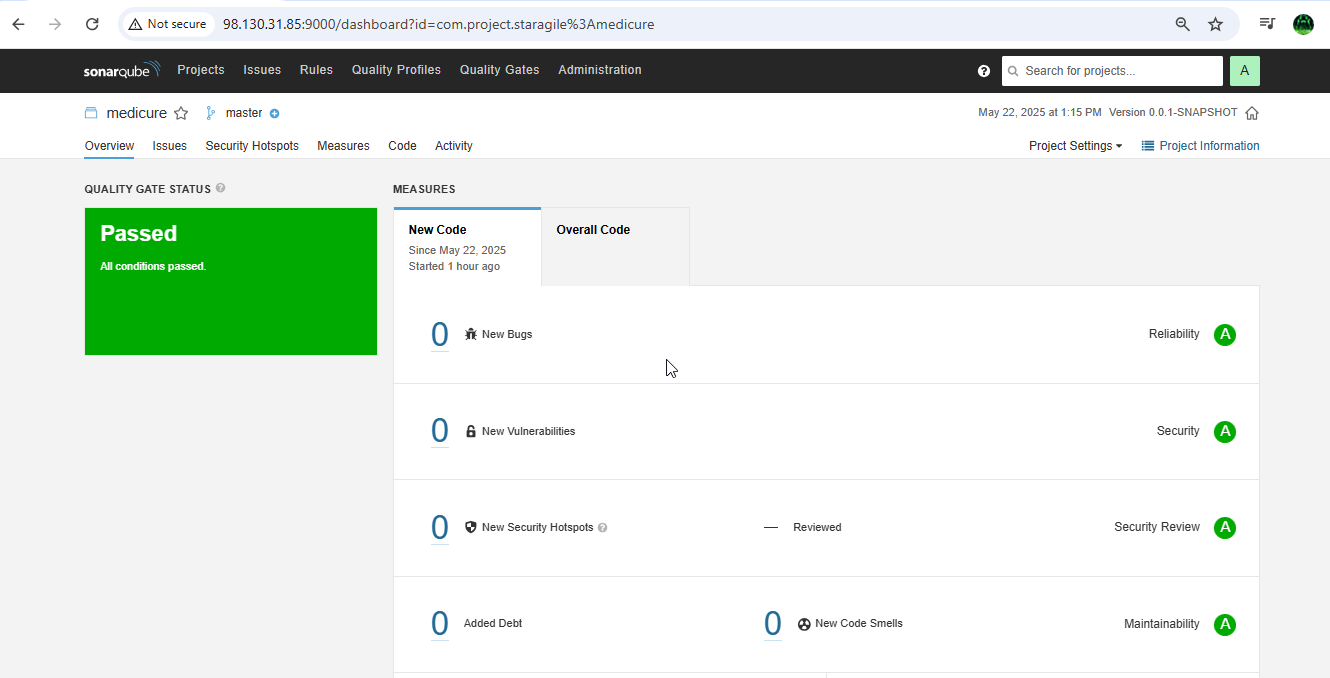


Step36:- Now again go back to Jenkins project and click on Build now to check the status of the build and as you can see that the build is successful and a 2pods are also created in the ec2 Worker node.

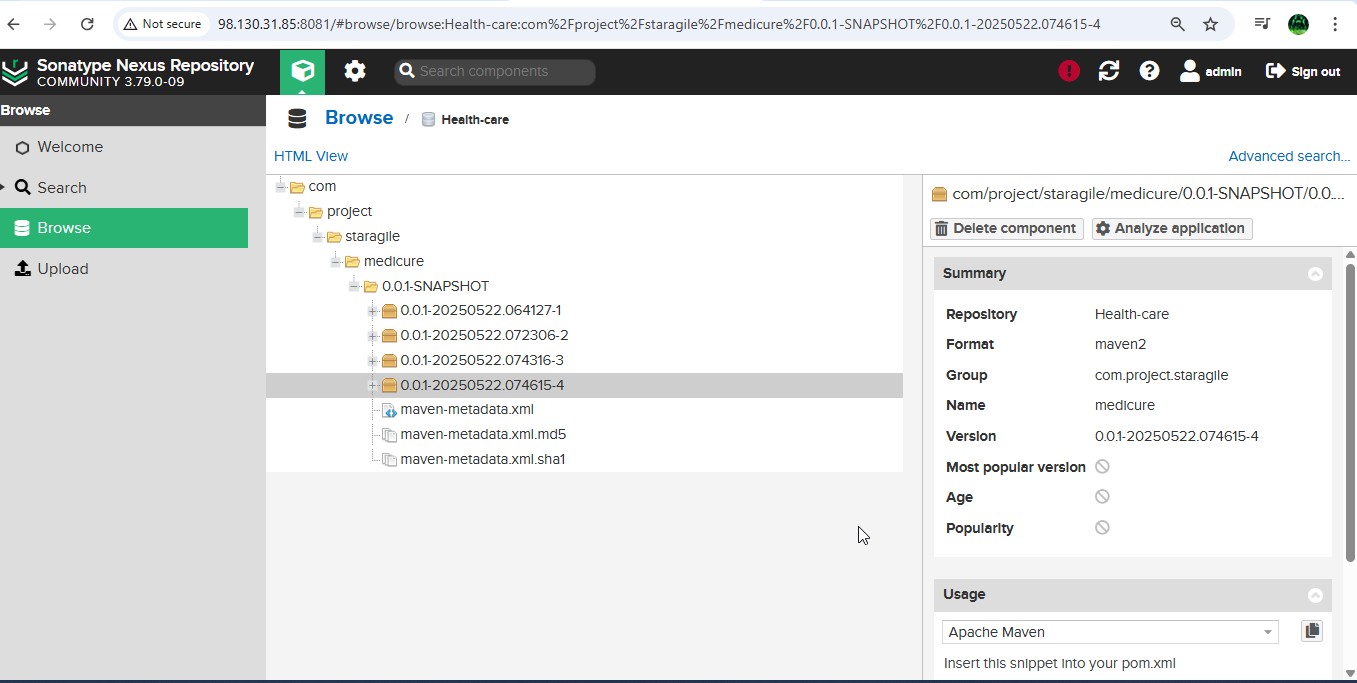




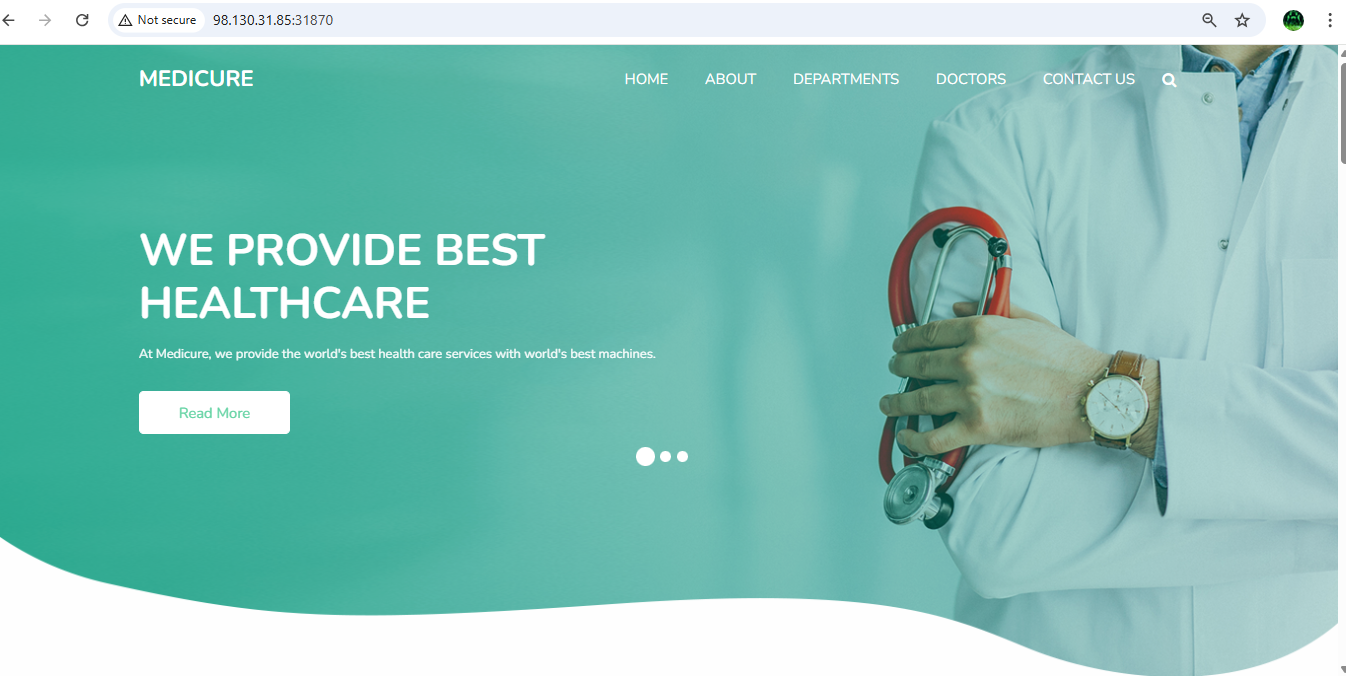
Step37:- Now go to sonarqube page and check the the project details



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

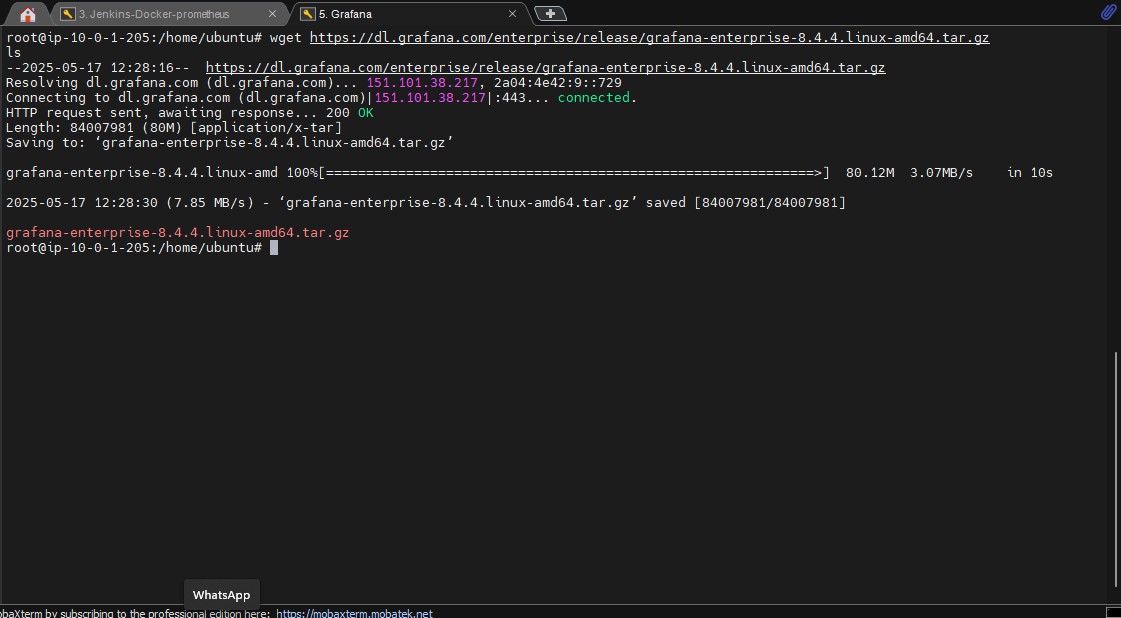


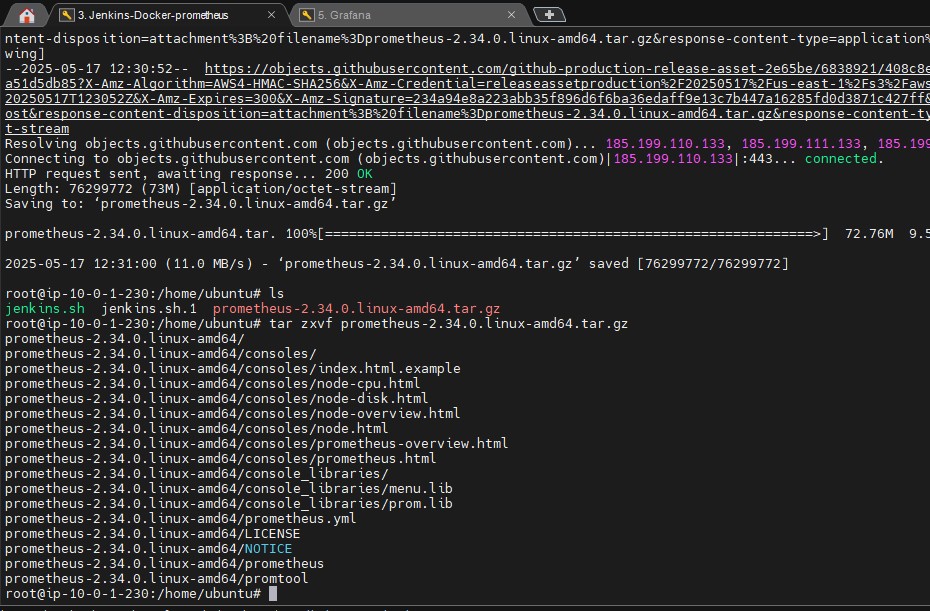
Step39:- Now go to any browser and give the <Workernode-IPaddress:31870> (since port is auto allocated) and click enter the you will see the home page of the project and thus the project deployment is successful using kubernetes.



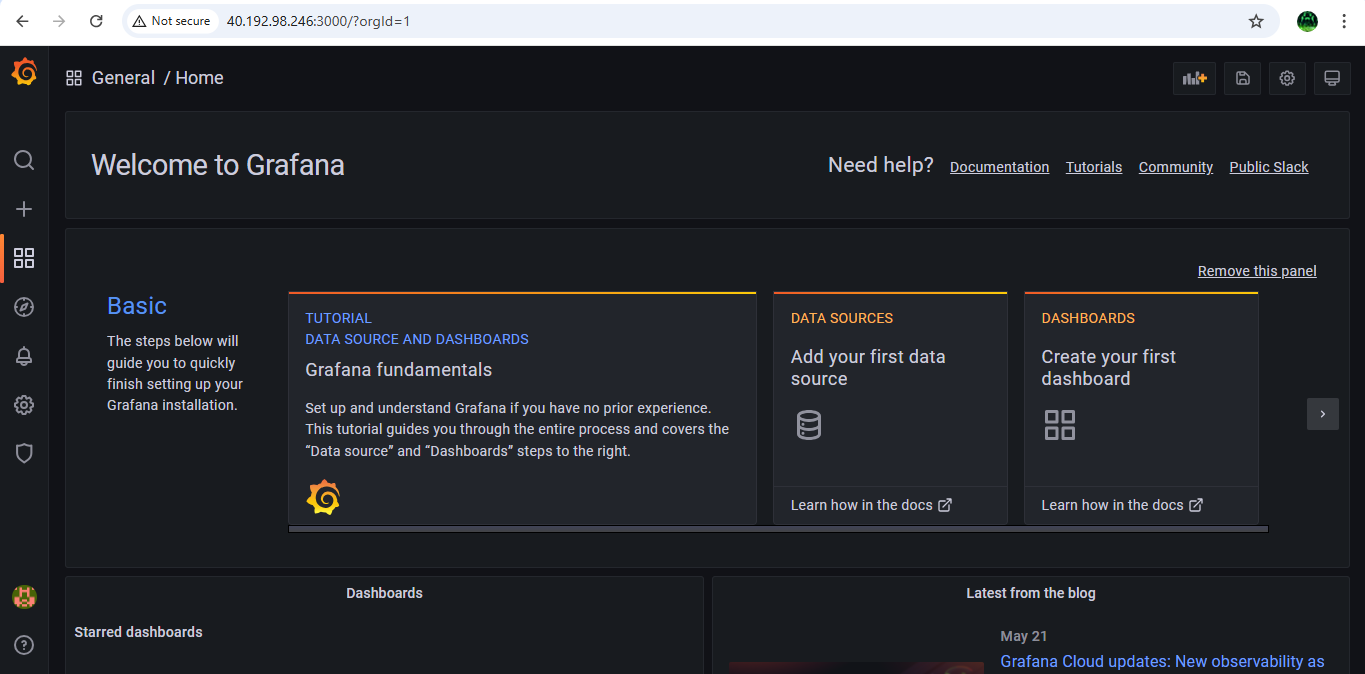
Step40:- Now monitor the pods using Prometheus and

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

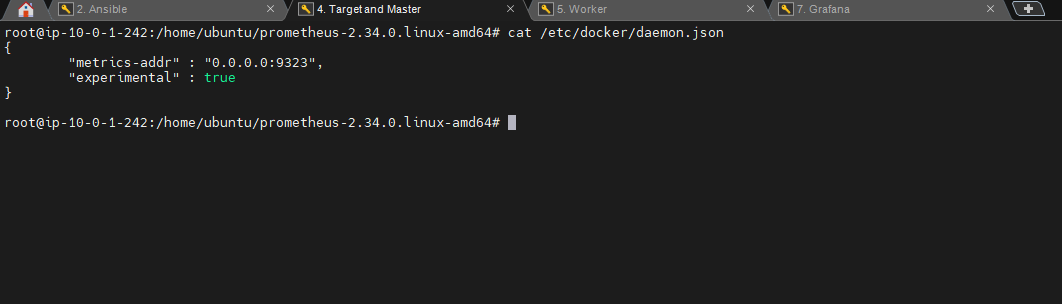
i.e., vi /etc/docker/daemon.json press I to insert

{

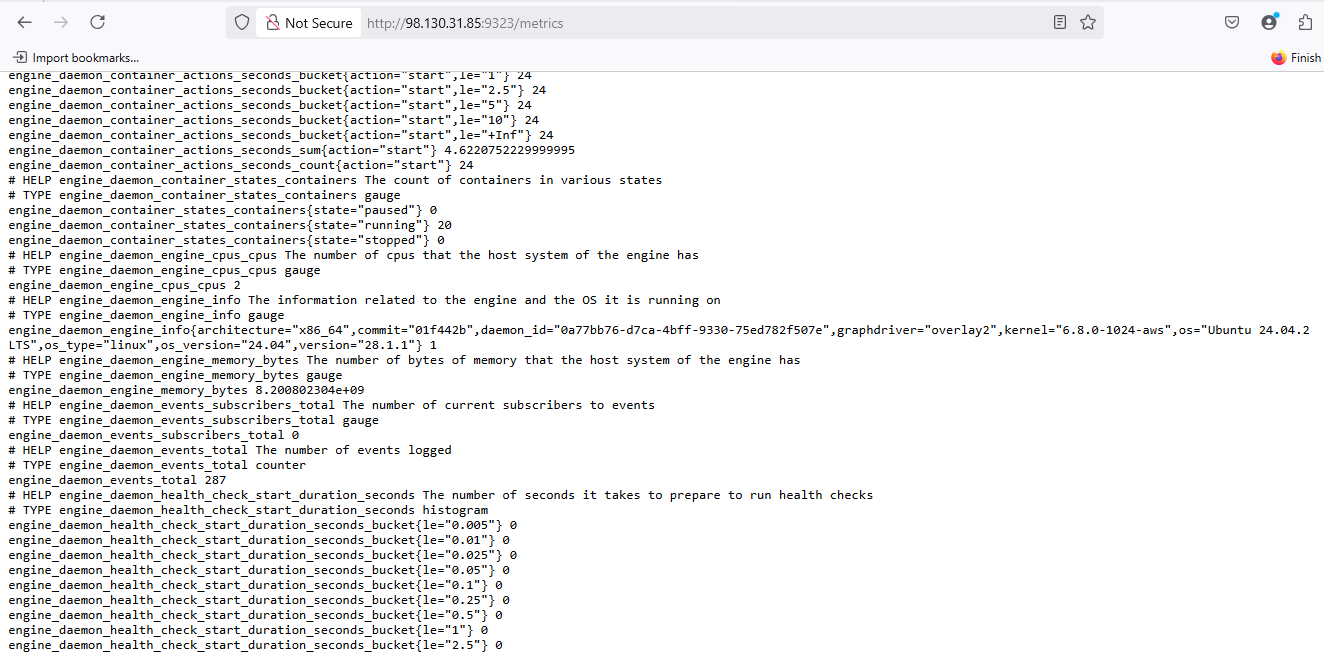
"metrics-addr" : "0.0.0.0:9323",

"experimental" : true

} then save and exit and restart the docker



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:

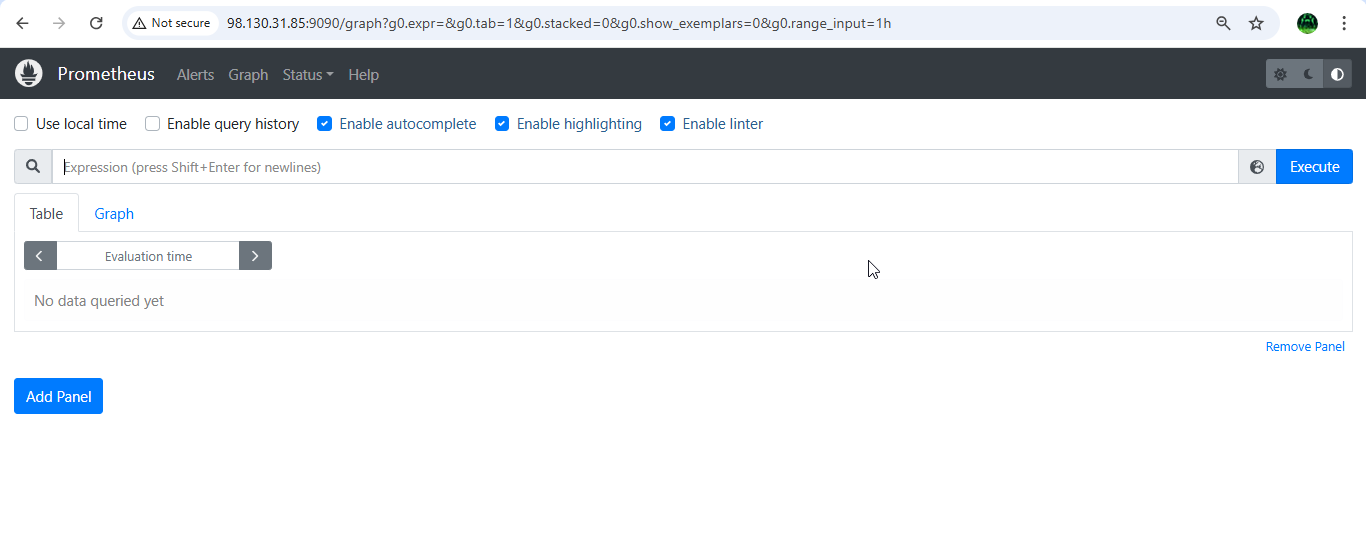
* targets: ["localhost:9323"]

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

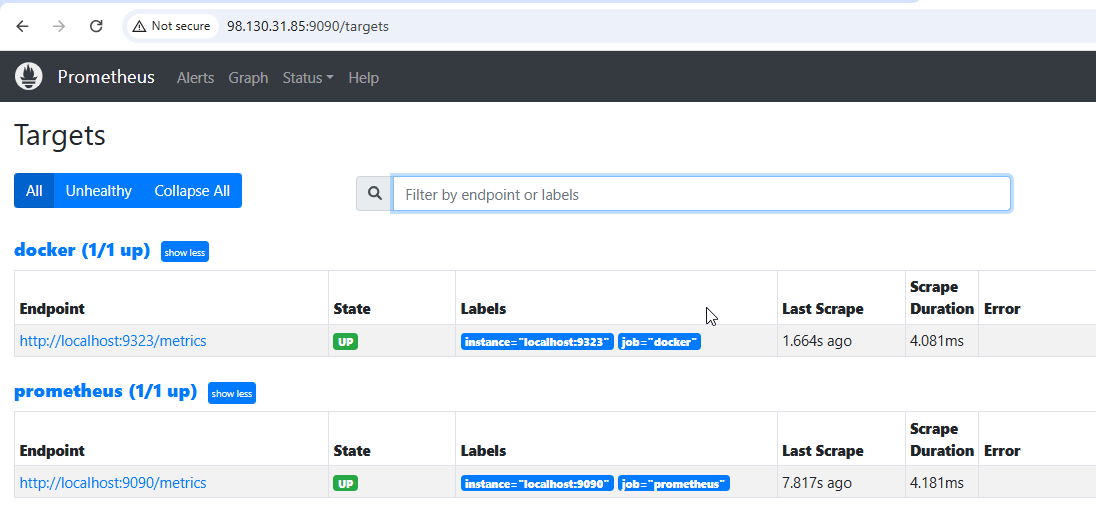


As you can see that the Prometheus have been started form the above image

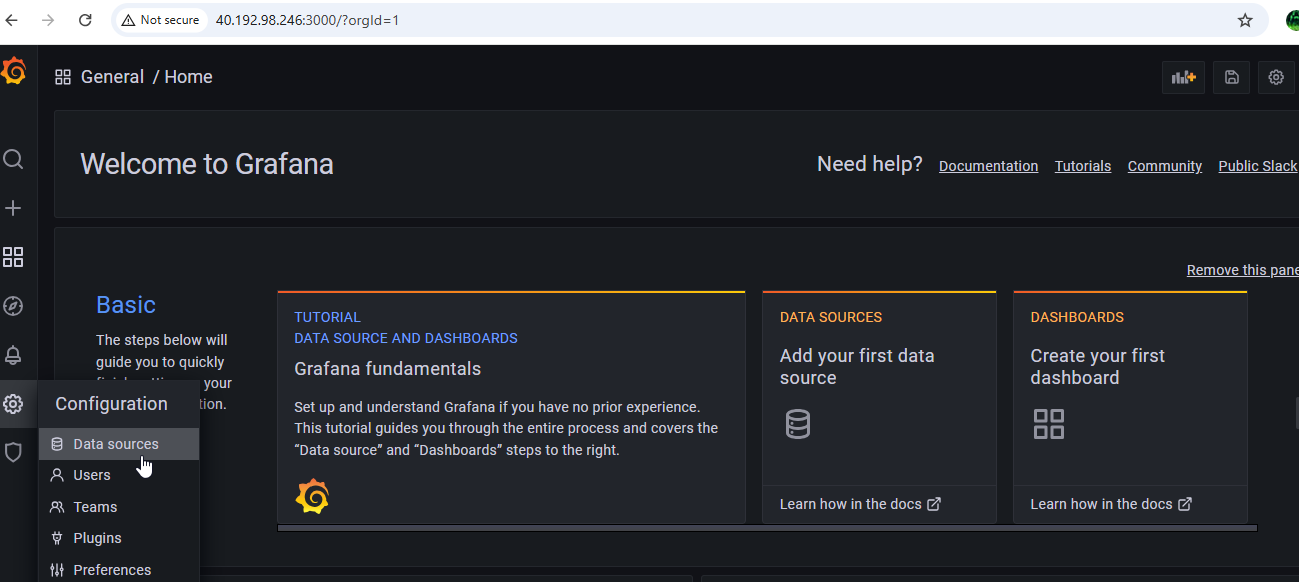
Step45:- Now go browser and give docker ip:9090 and enter , then you will be successfully enter into the Prometheus homepage



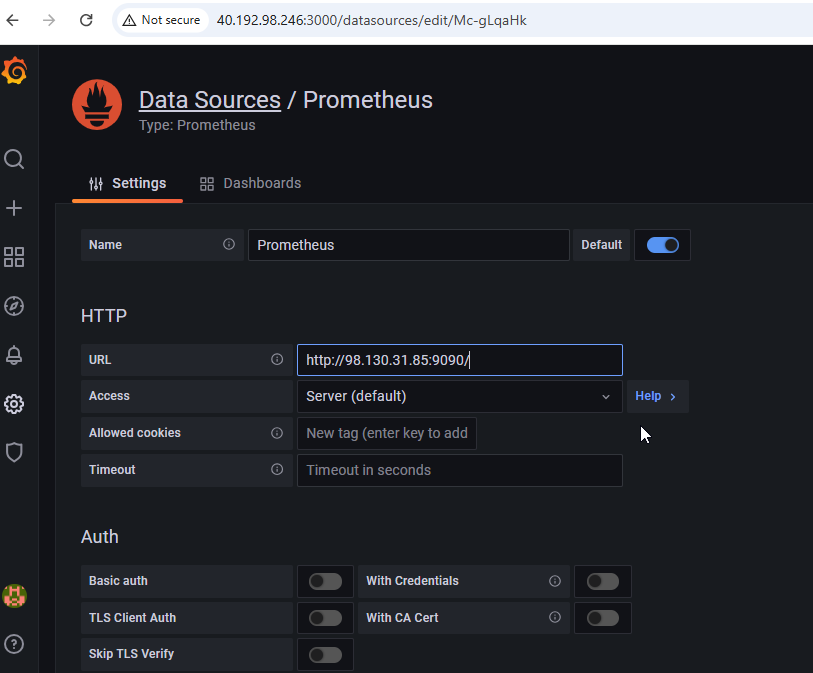
Step46:- Now click on status → targets then you will see the status of the of the docker and prometheus.



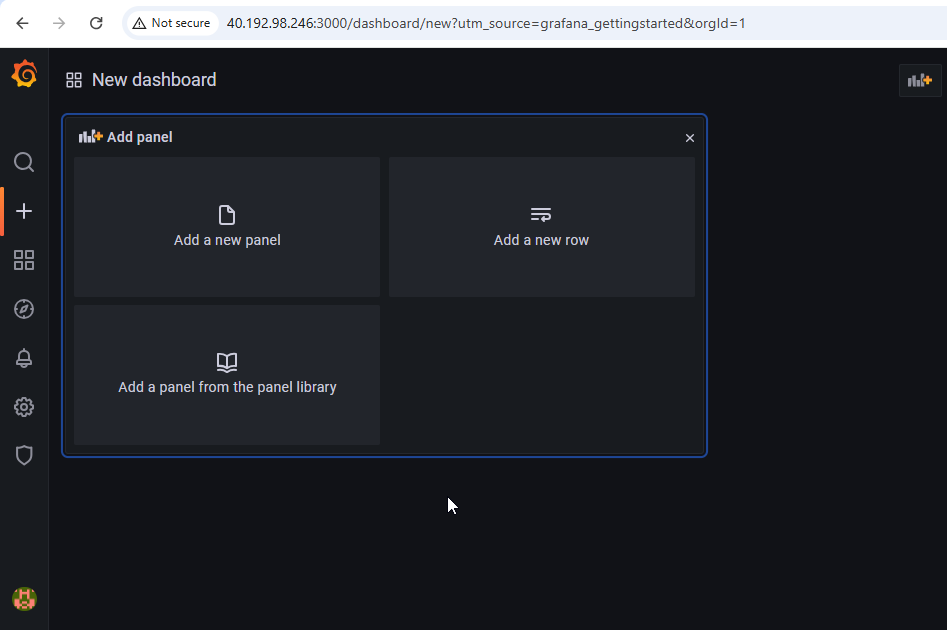
Step47:- Now go to grafana homepage → configurations → Data sources



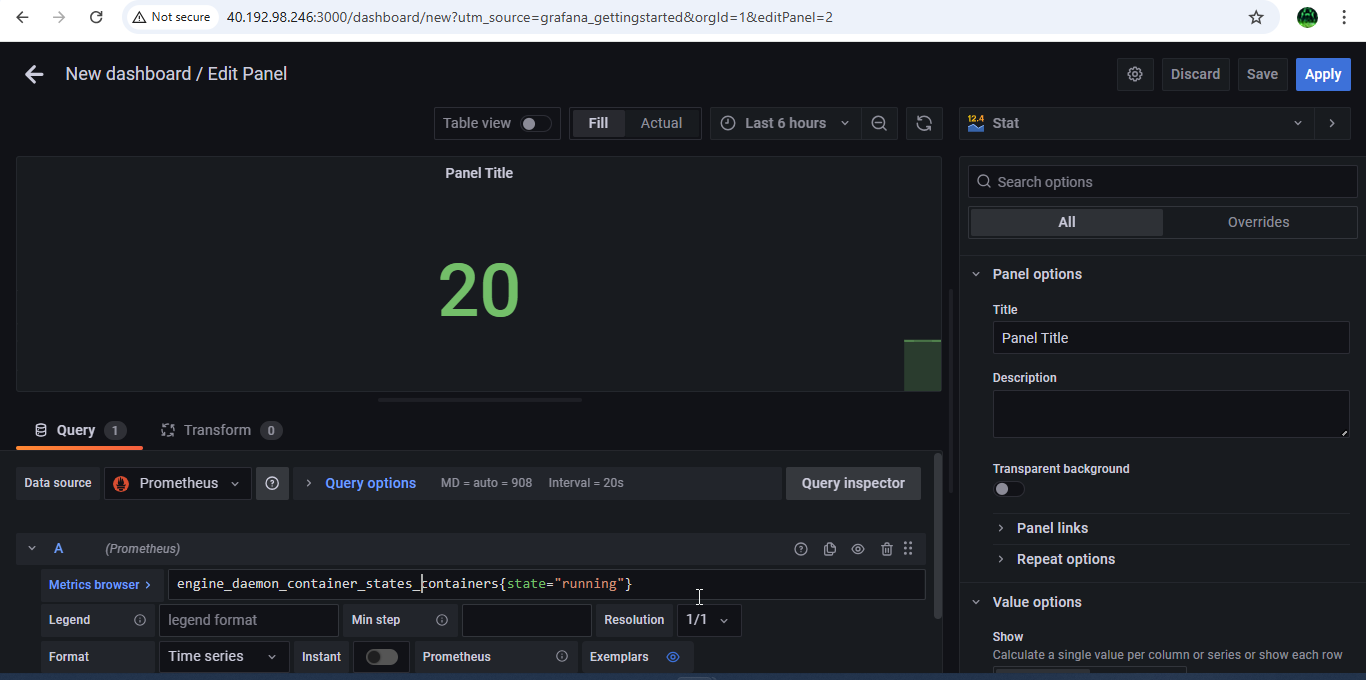
Step48:- Now click on add Data sources → Prometheus and give ipaddress:9090 and click on save and test



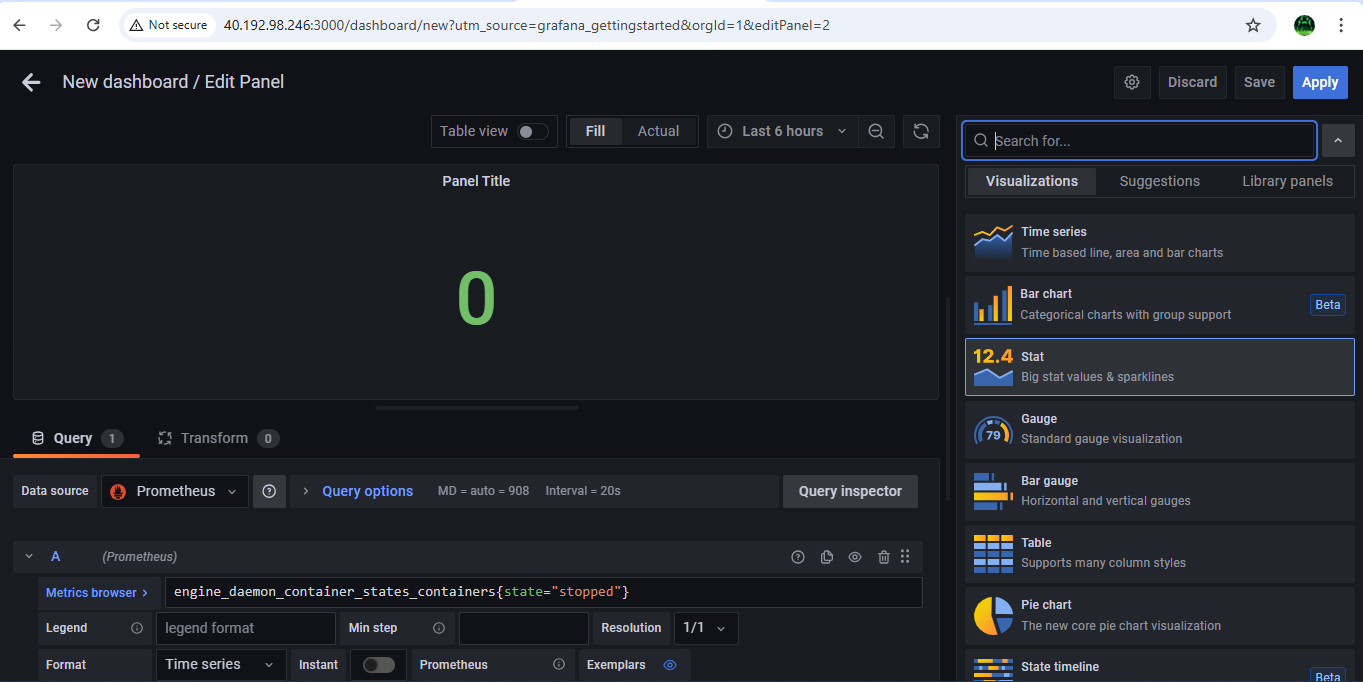
Step49:- Now click on Dash board → add new panel



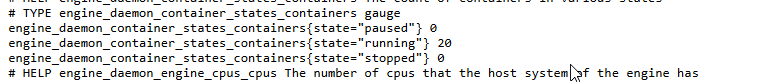
Step50:- Now in the metrics browser give engine daemon container states containers{state="running"} and you will see the result that same as in the metrics from the browser



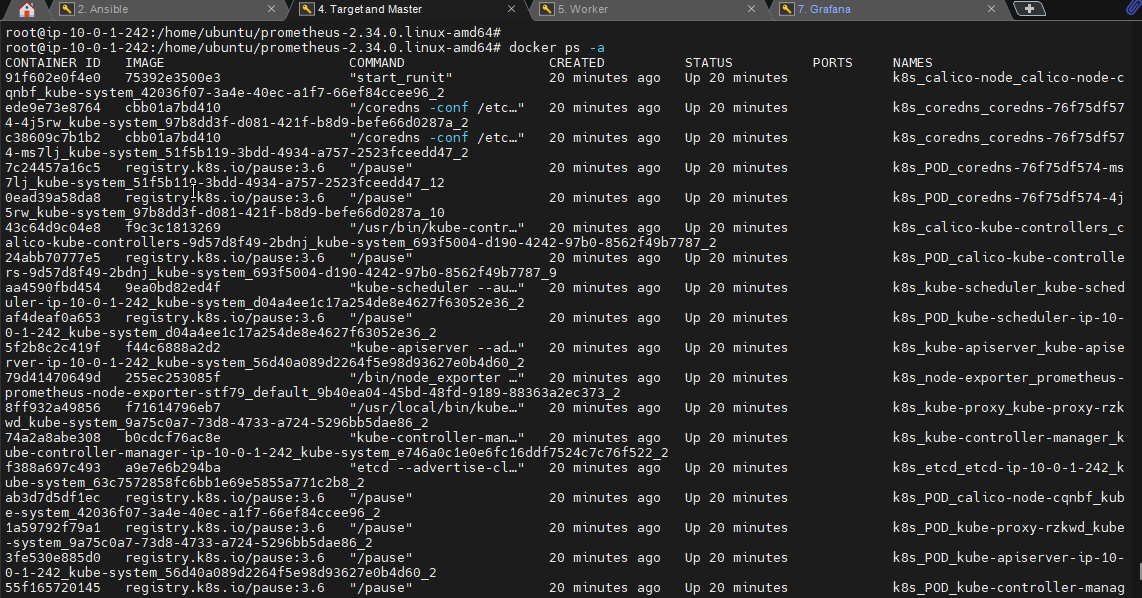
engine\_daemon\_container\_states\_containers{state="stopped"}



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



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



Step53:- This is how we monitor the health of a container automatically and visualizing the report using Prometheus and Grafana.