Module Name : Capstone Project

Project name : HealthCare domain

Submitted by : Sachin V Bacha

On : 25-05-2025

Business challenge/requirement

As soon as the developer pushes the updated code on the GIT master branch, the Jenkins pipeline should be triggered and code should be checkout, compiled, tested, packaged and containerized. A new test-cluster should be provisioned and configured automatically with all the required software's and as soon as the cluster is healthy and available, the application must be deployed to the test-server automatically using Kubernetes.

The deployment should then be tested using a test automation tool, and if the build is successful, it should be deployed to the prod server/cluster using Kubernetes.

All this should happen automatically and should be triggered from a push to the GitHub master branch. Kubernetes cluster must contain at least 2 servers and must be monitored continuously using Prometheus and dashboard must be visualized using Grafana.

Tech stack

✓ Git - For version control for tracking changes in the code files

✓ Jenkins - For continuous integration and continuous deployment

✓ Docker - For containerizing applications

✓ Ansible - Configuration management tools

✓ Selenium - For automating tests on the deployed web application

✓ Terraform - For creation of infrastructure.

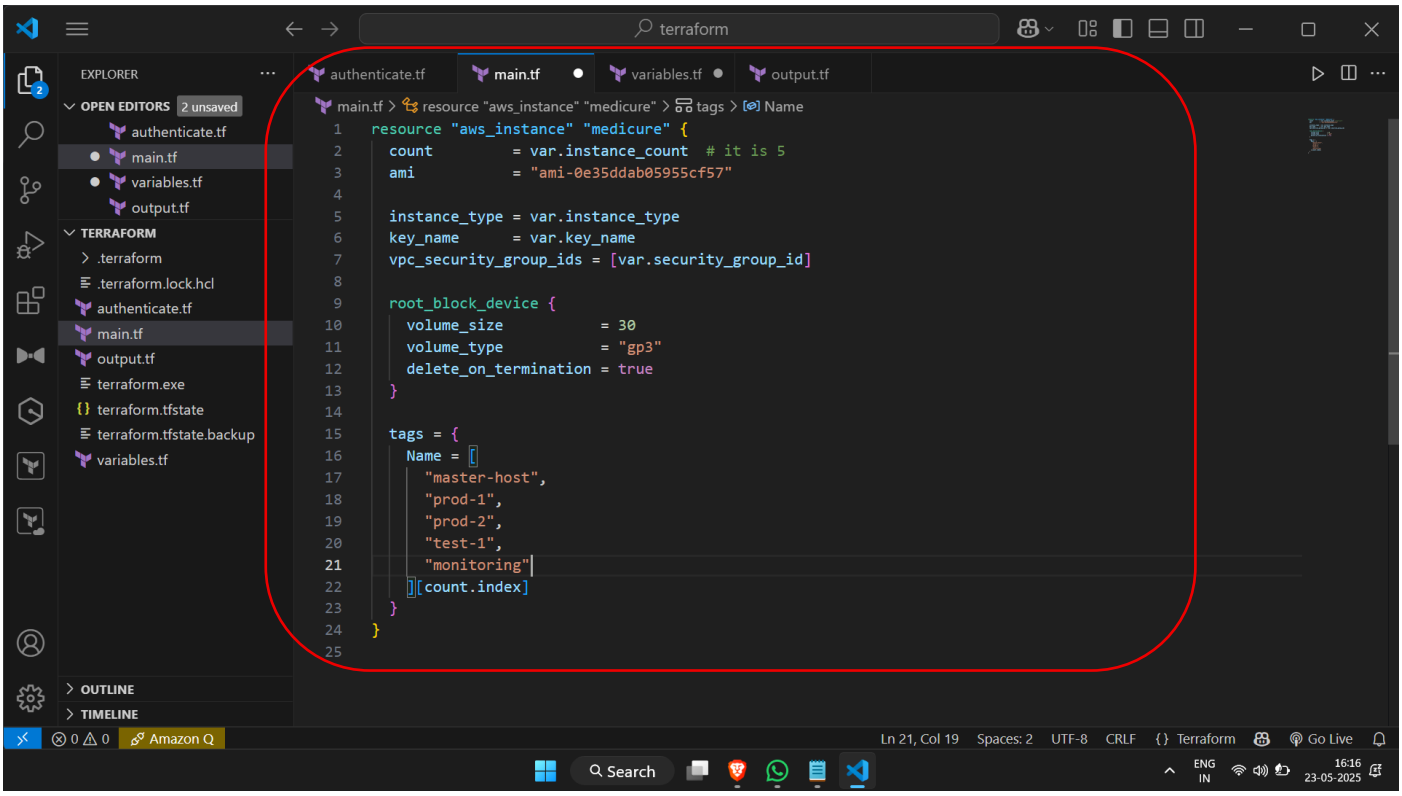
✓ Kubernetes – for running containerized application in managed cluster

Launch 5 Instance with terraform

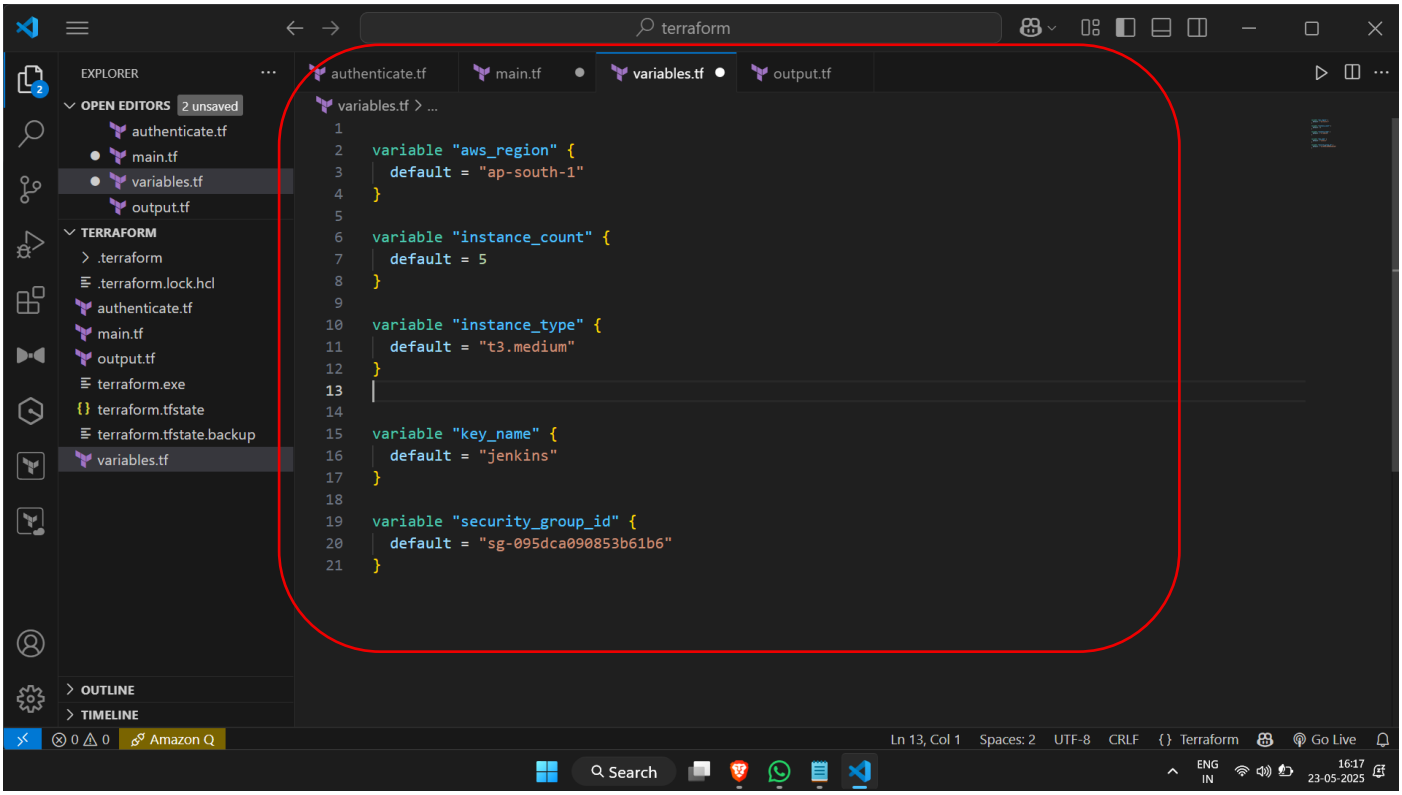
Name	Instance type	OS
Host-master	- t3.medium	ubuntu
Production-1	- t3.medium	ubuntu
Production-2	- t3.medium	ubuntu
Test-1	- t2.medium	ubuntu
Monitoring	-t3.medium	ubuntu

1. Create a terraform directory consist of a terraform.exe file
2. Create Four files authenticate.tf , main.tf , output.tf , variables.tf

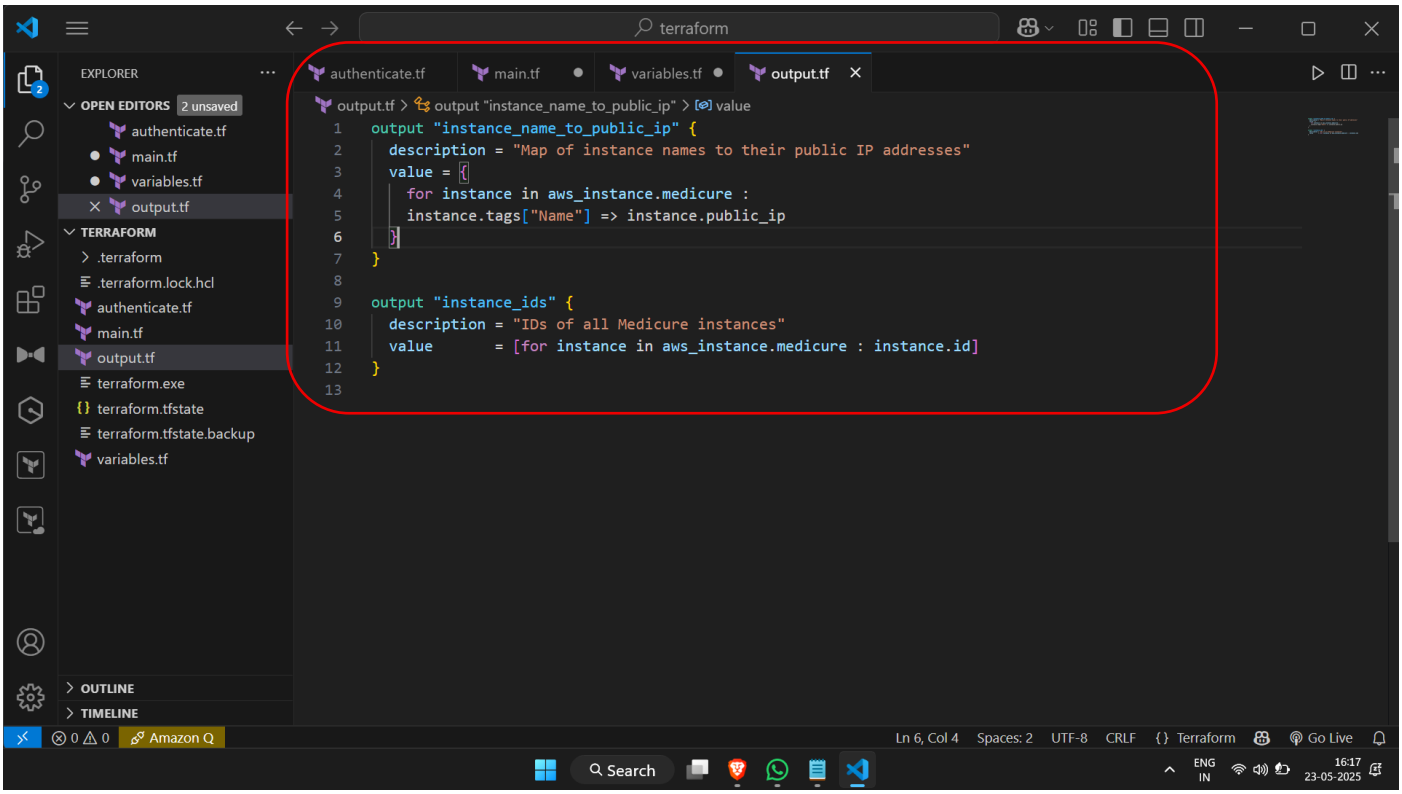
Main.tf



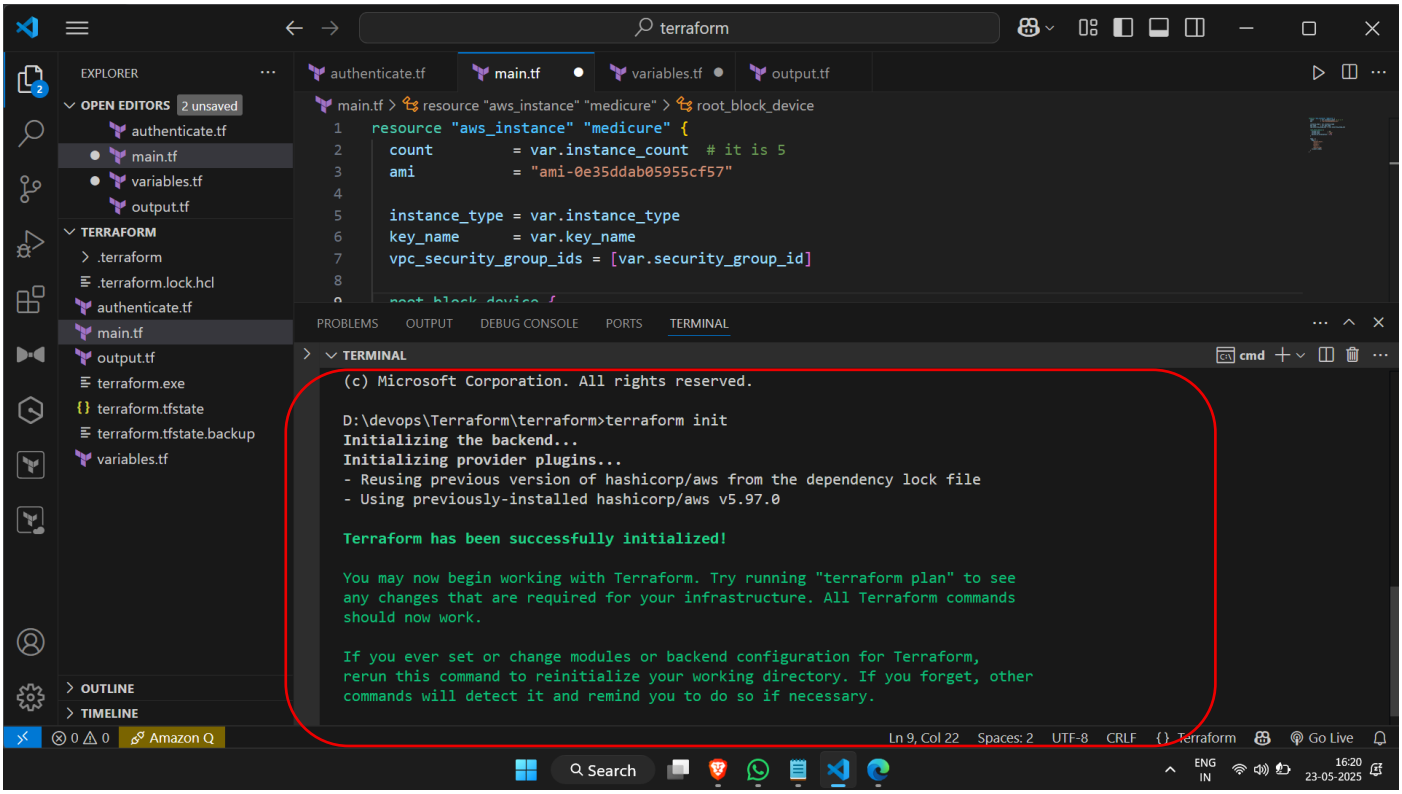
Variables.tf



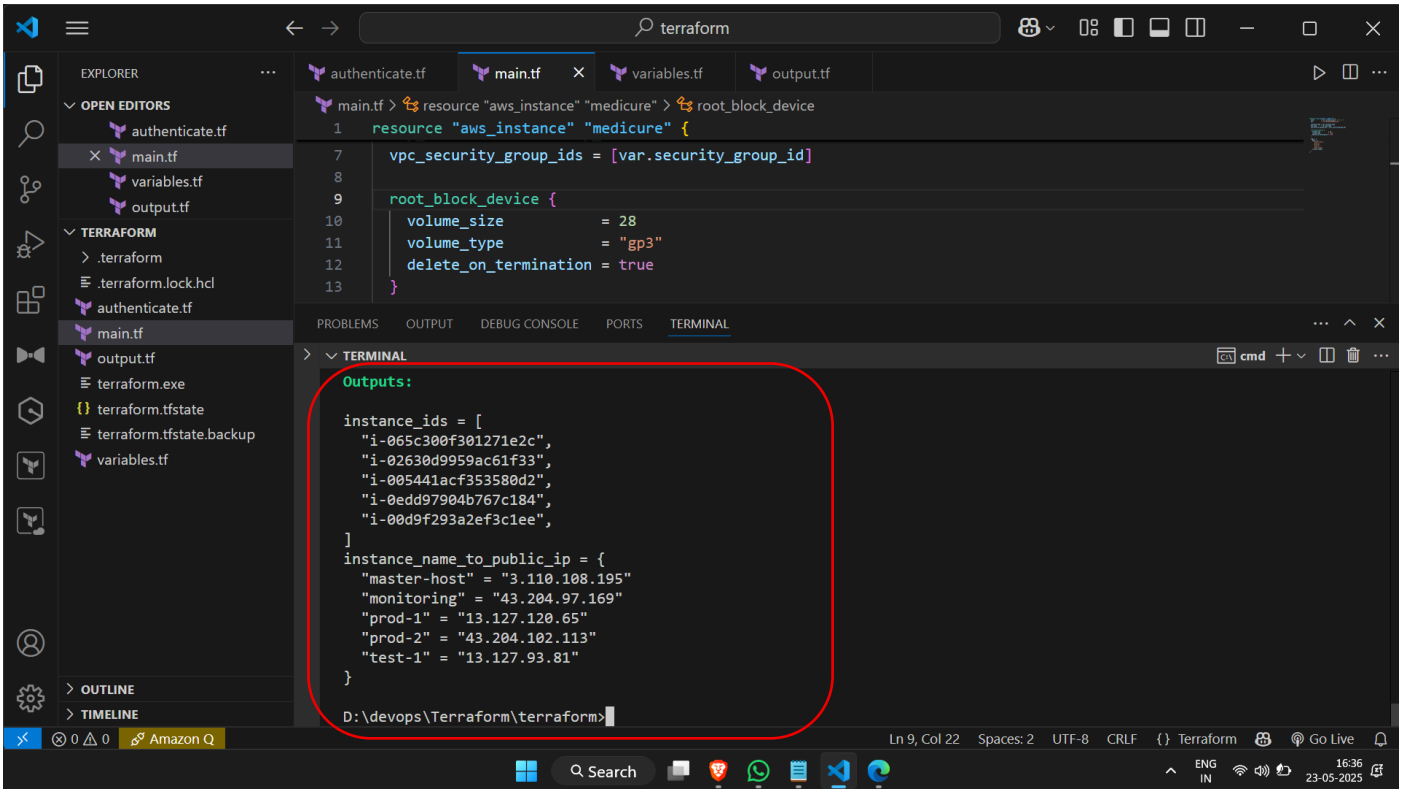
Output.tf



Open a new CMD terminal and initialize the terraform using “terraform init” command



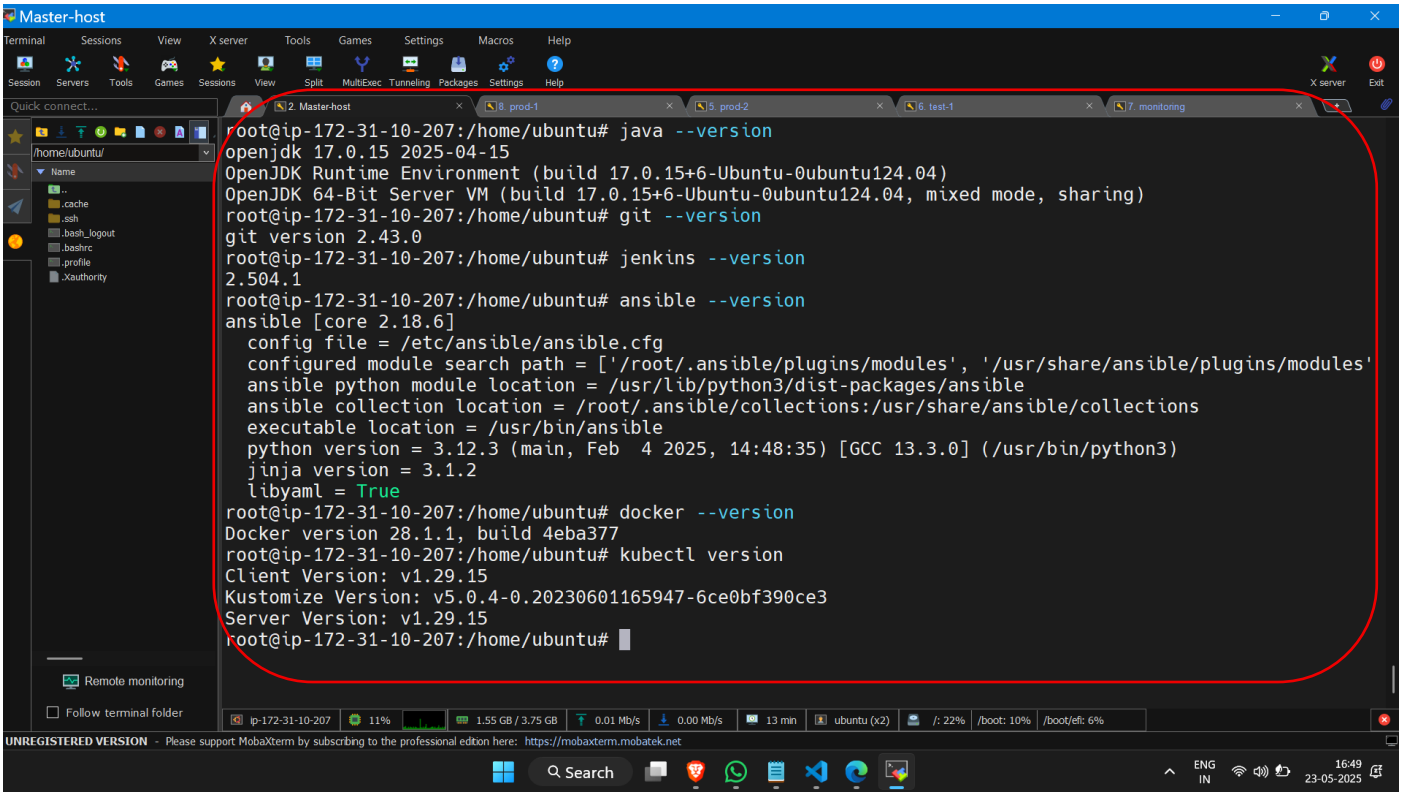
Use “terraform plan” command to check and verify it the resources are matching as required then use command “terraform apply” to create the resources



Connect the all the instances through MobaXterm using the key pair attached in terraform script

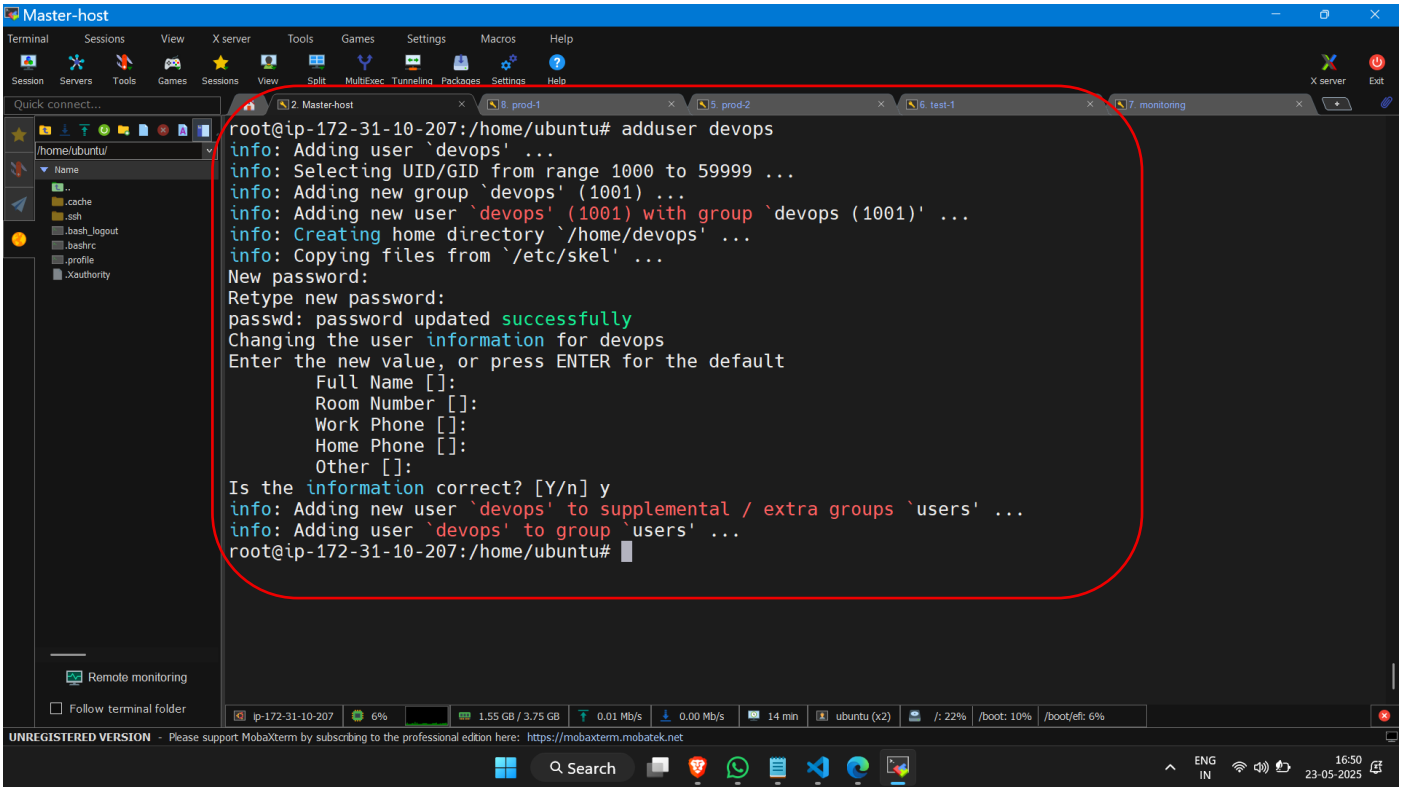
Install all the required packages in the master machine (Jenkins, Java, maven,git, ansible, docker, kubenetes)

which is the Jenkins host and ansible controller and the Kubernetes master node .

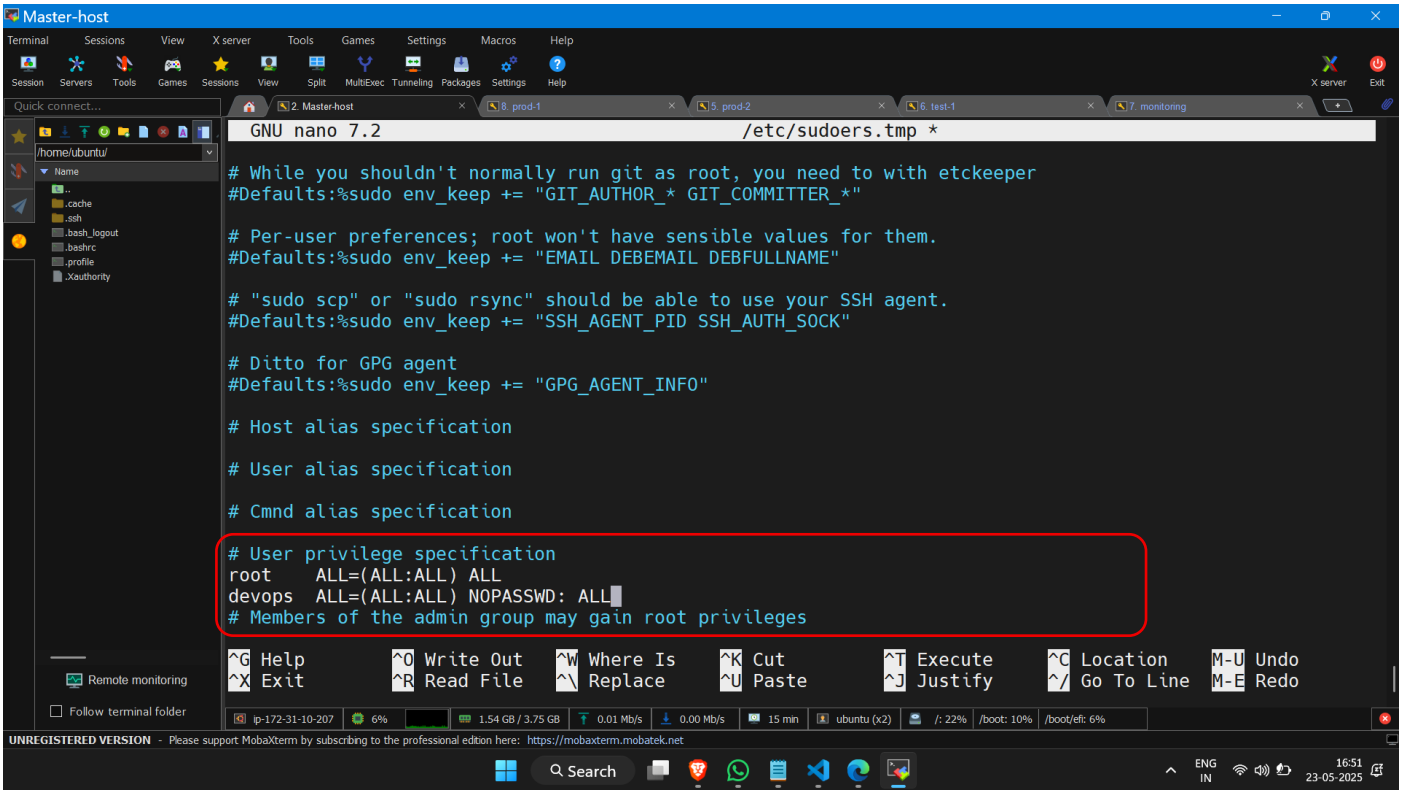


Now do Ansible configuration from master node to connect all other nodes as user “devops” with password

In master and all other nodes

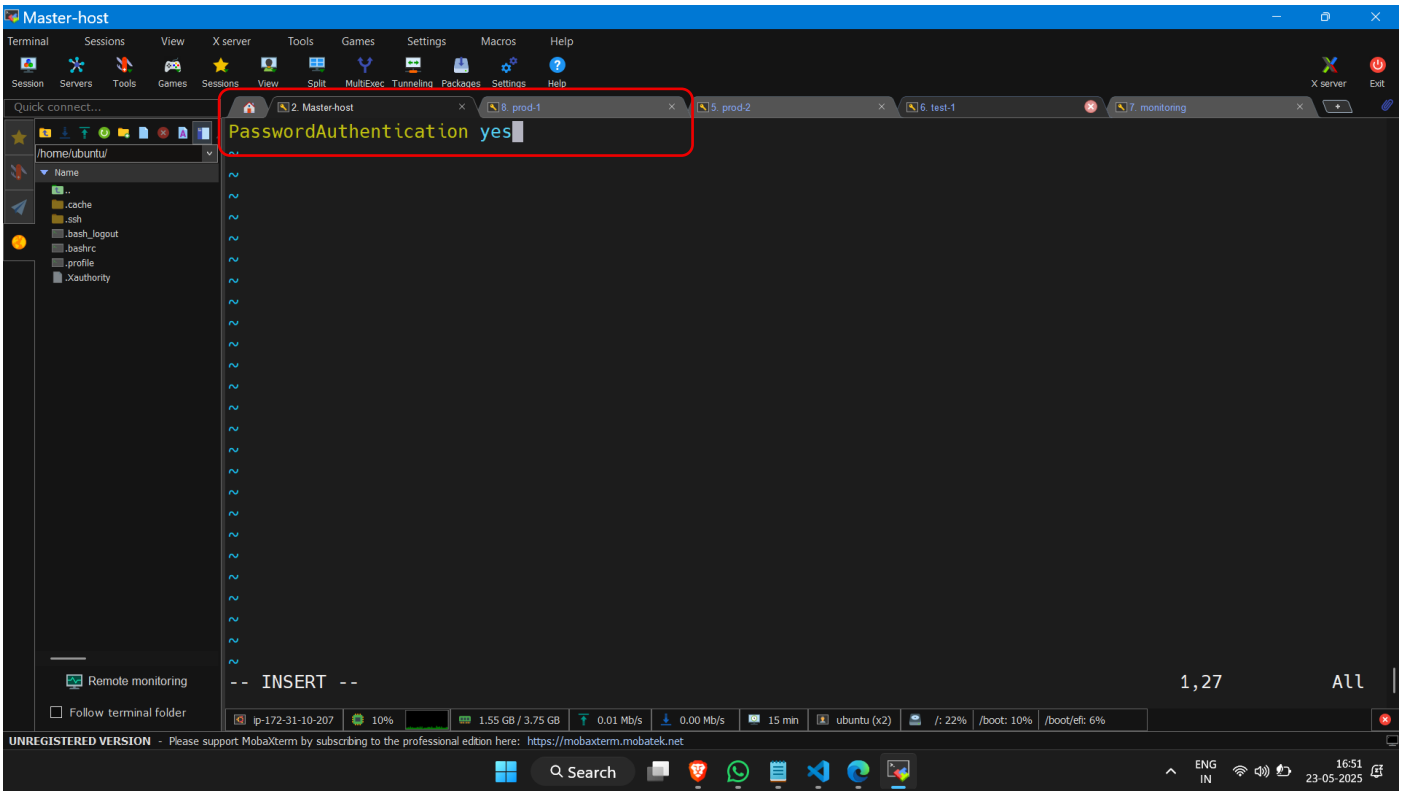


Now add the user to the sudoers file with the permissions . perform the same in all the other nodes



Now configure sshd configuration in both master and other nodes

Get into the " vi /etc/ssh/sshd_config.d/60-cloudimg-settings.conf "and change passwordauthentication to yes

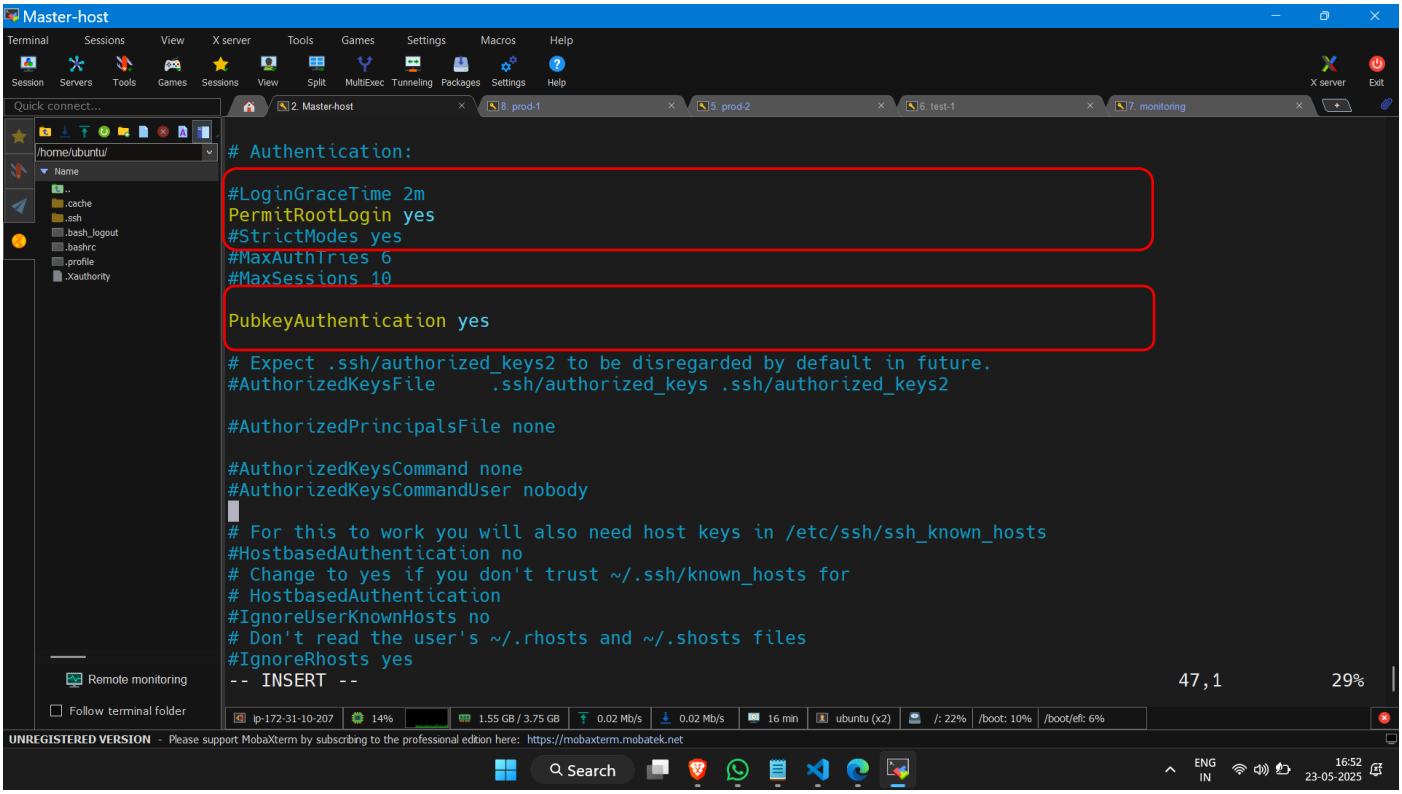


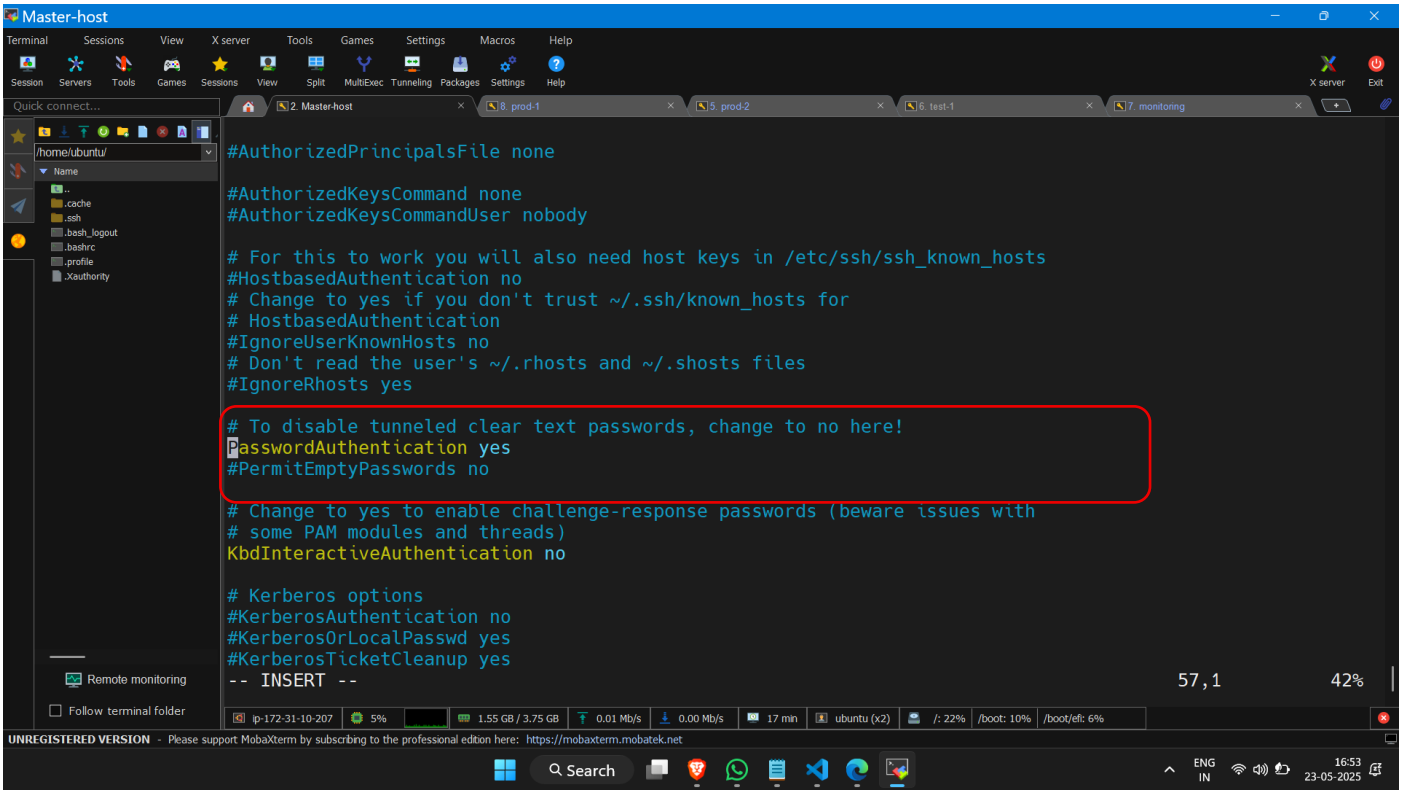
Also change the configuration file in using command “vi /etc/ssh/sshd_config” change

Permitlogin to yes

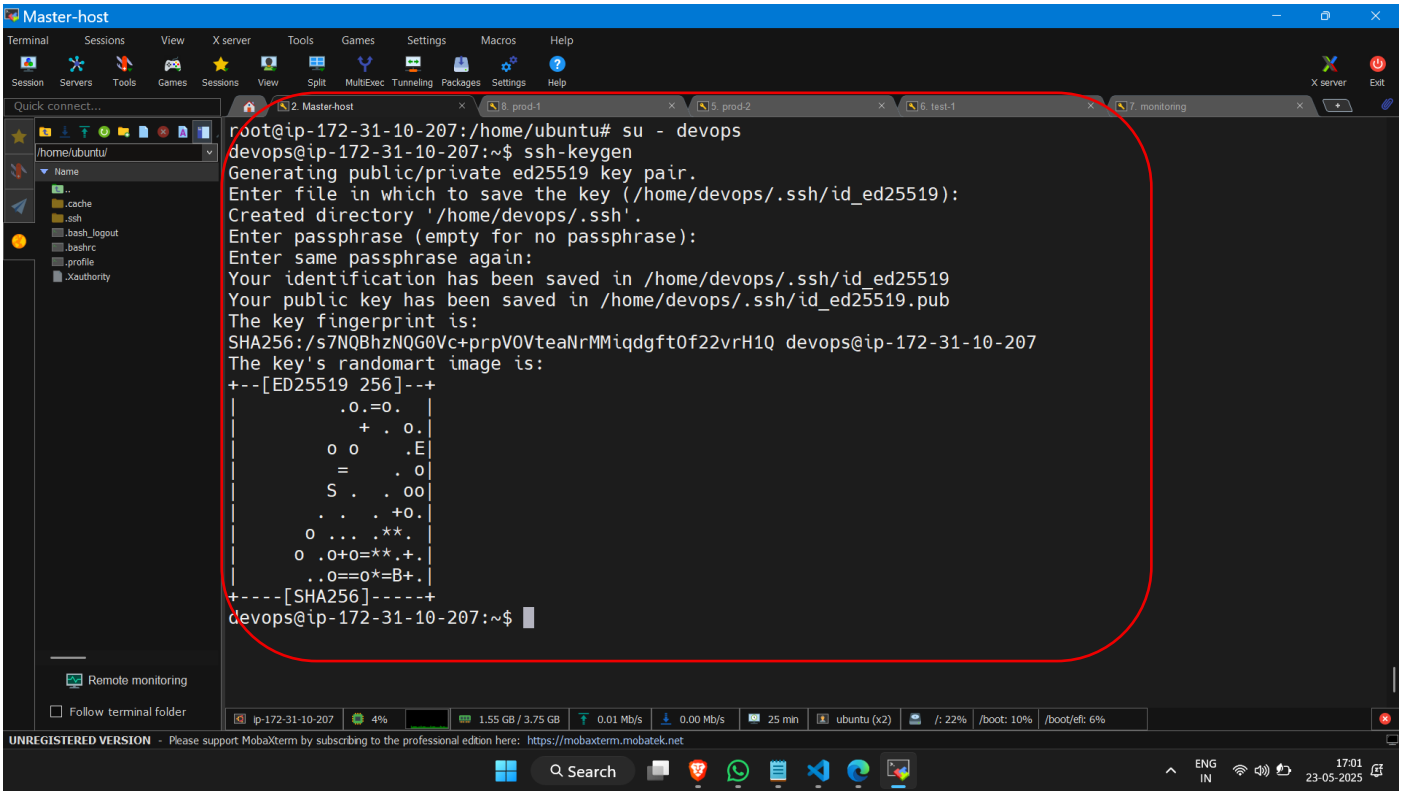
Pubkeyauthentication to yes

And passwordauthentication to yes -perform the same activity in nodes as well

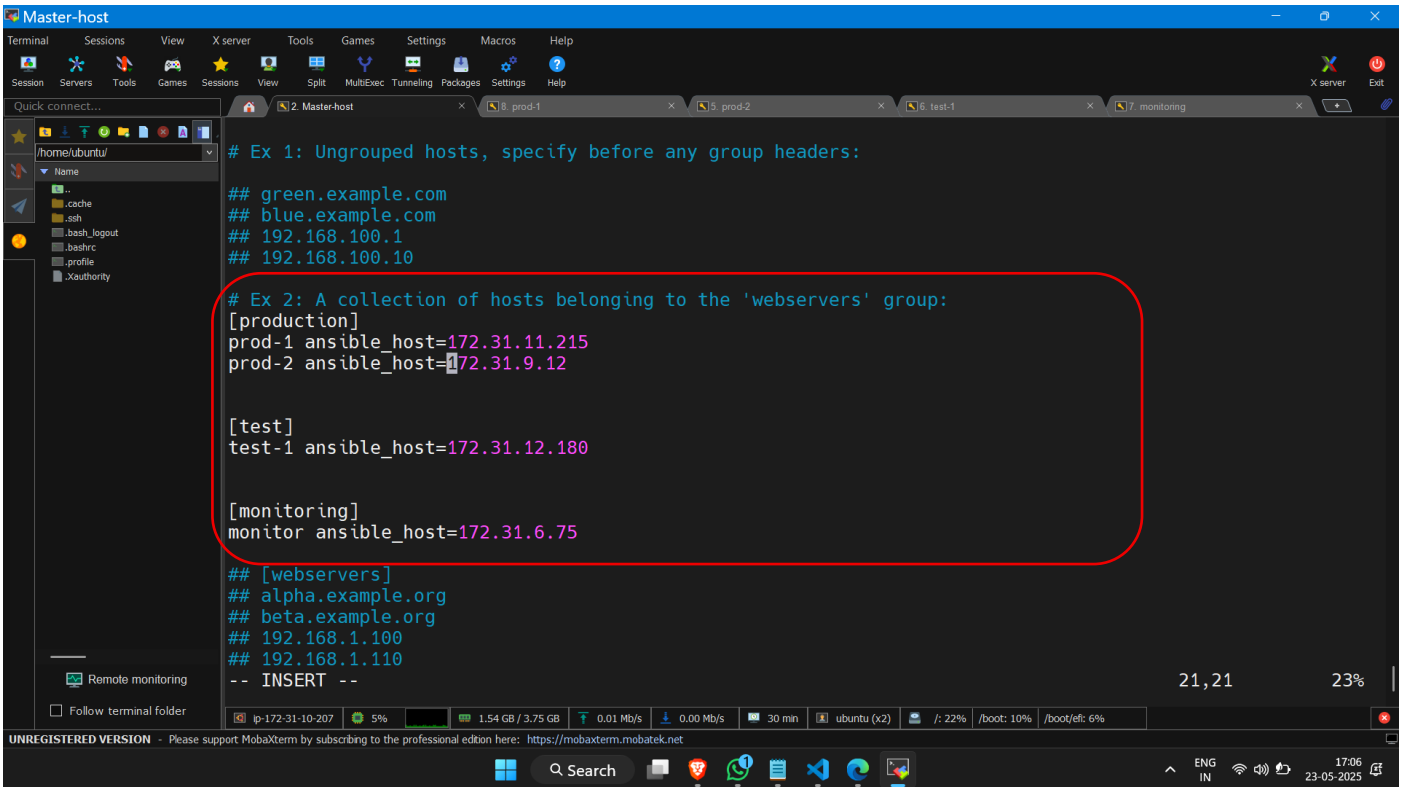




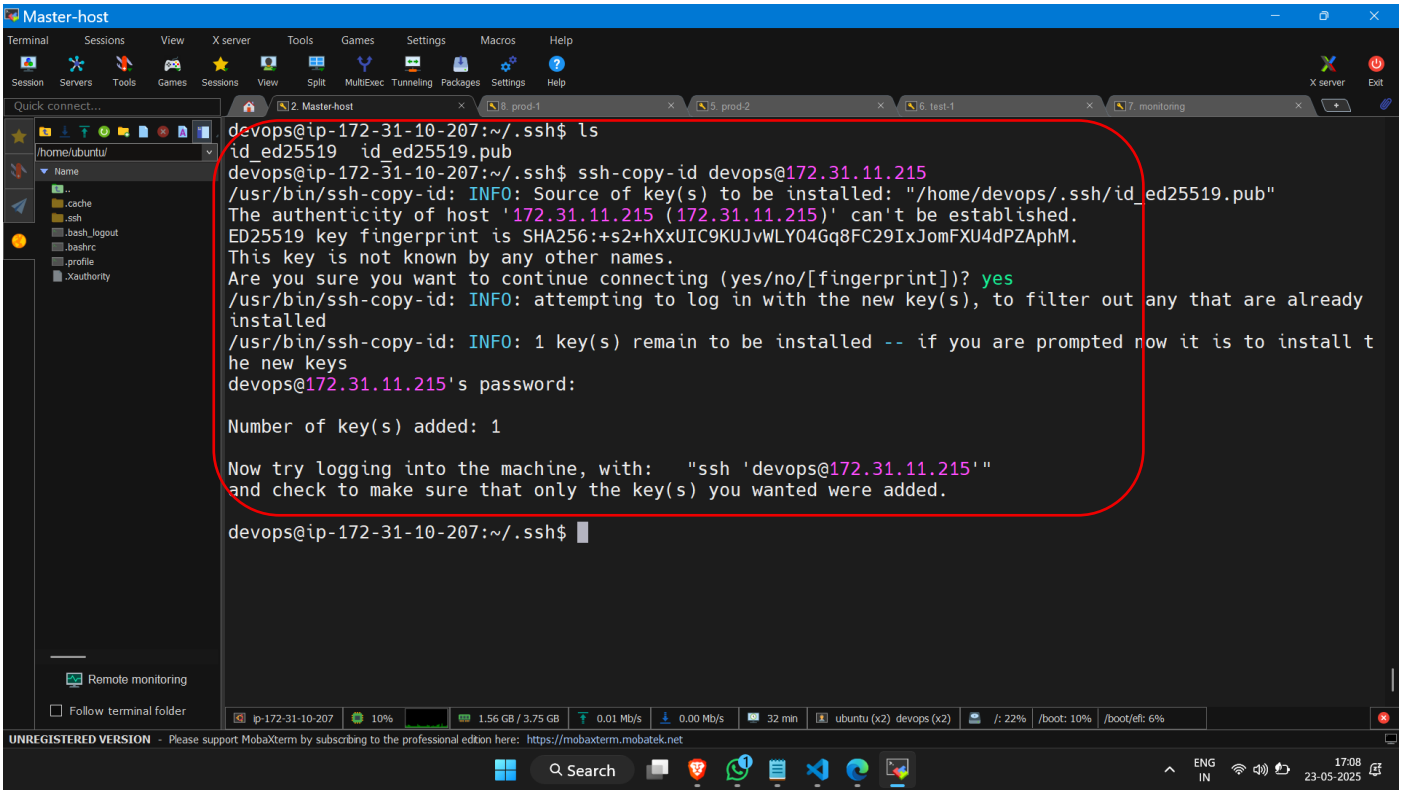
Now get into the user devops using command “su – devops” and generate a ssh key using “ssh-keygen” command

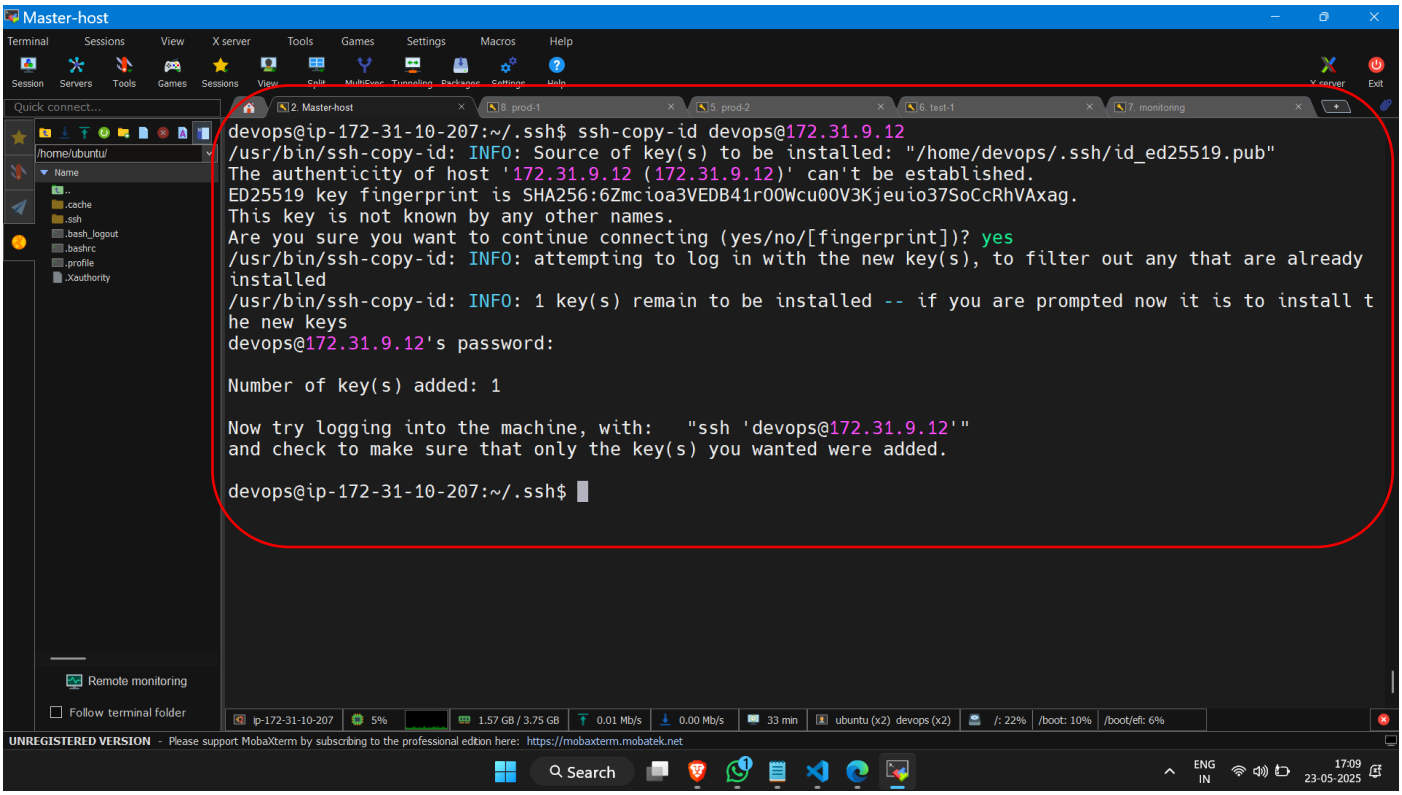


Add the other nodes into the hosts file /etc/ansible/hosts file with the private ip address

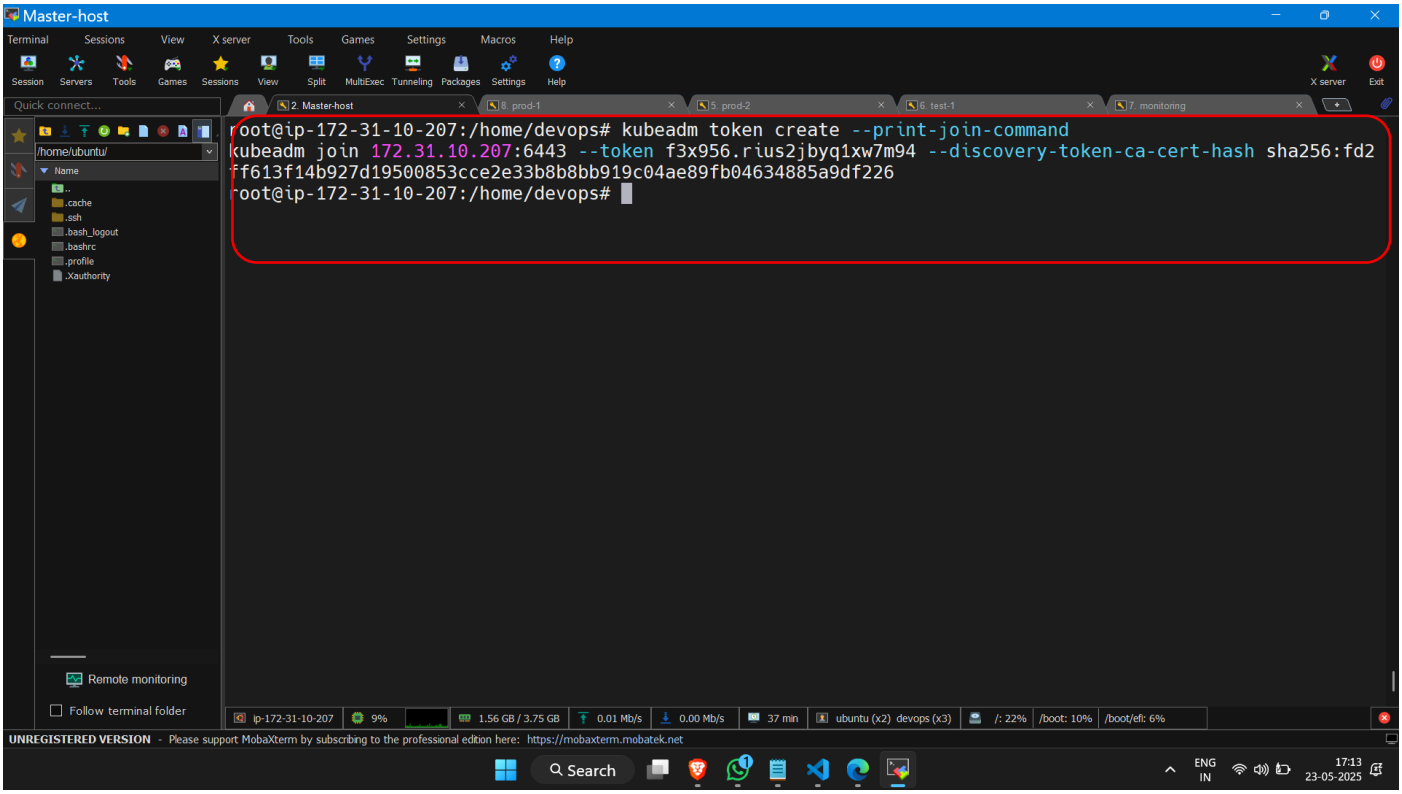


And now we have to copy the ssh into other nodes using ssh-copy-id devops@<private_ip> perform this to every node



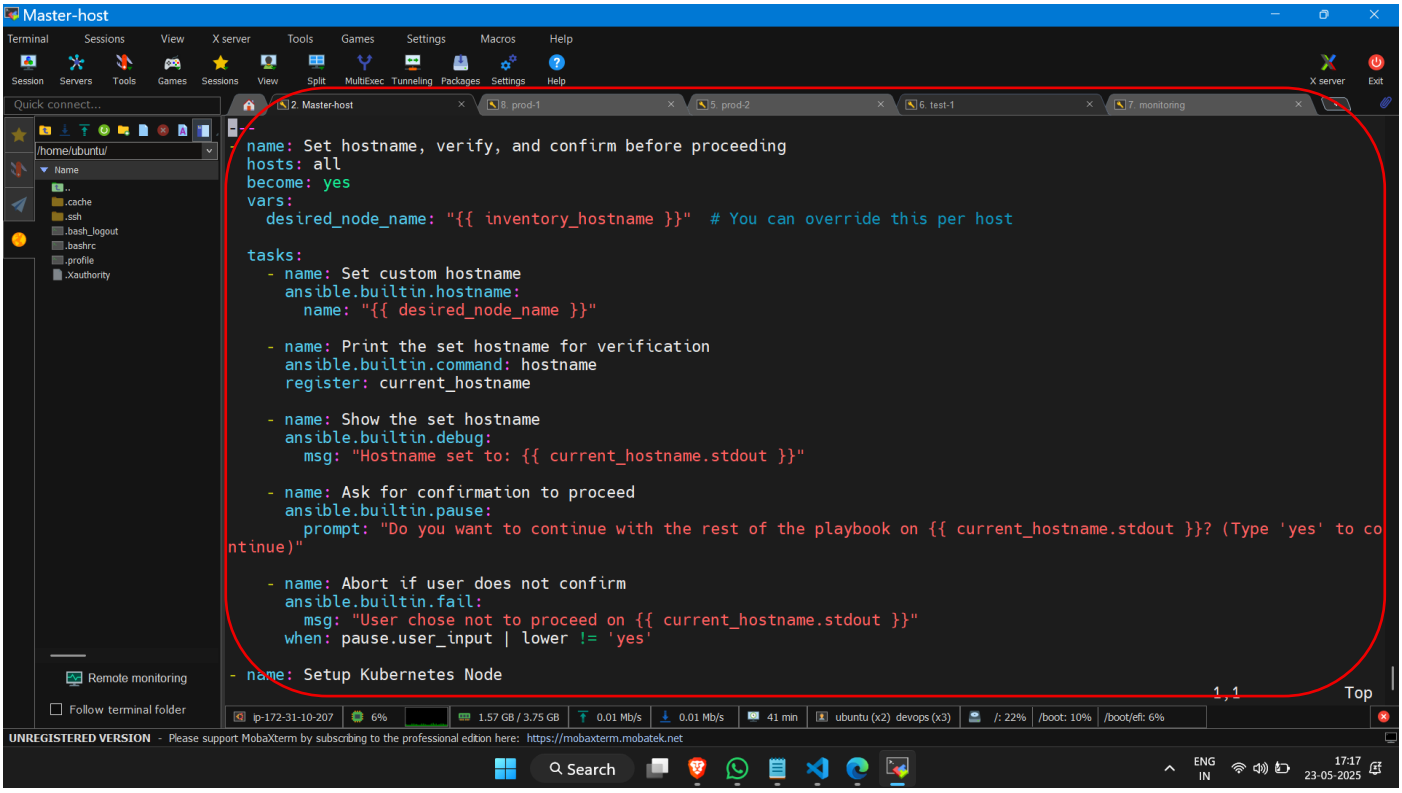


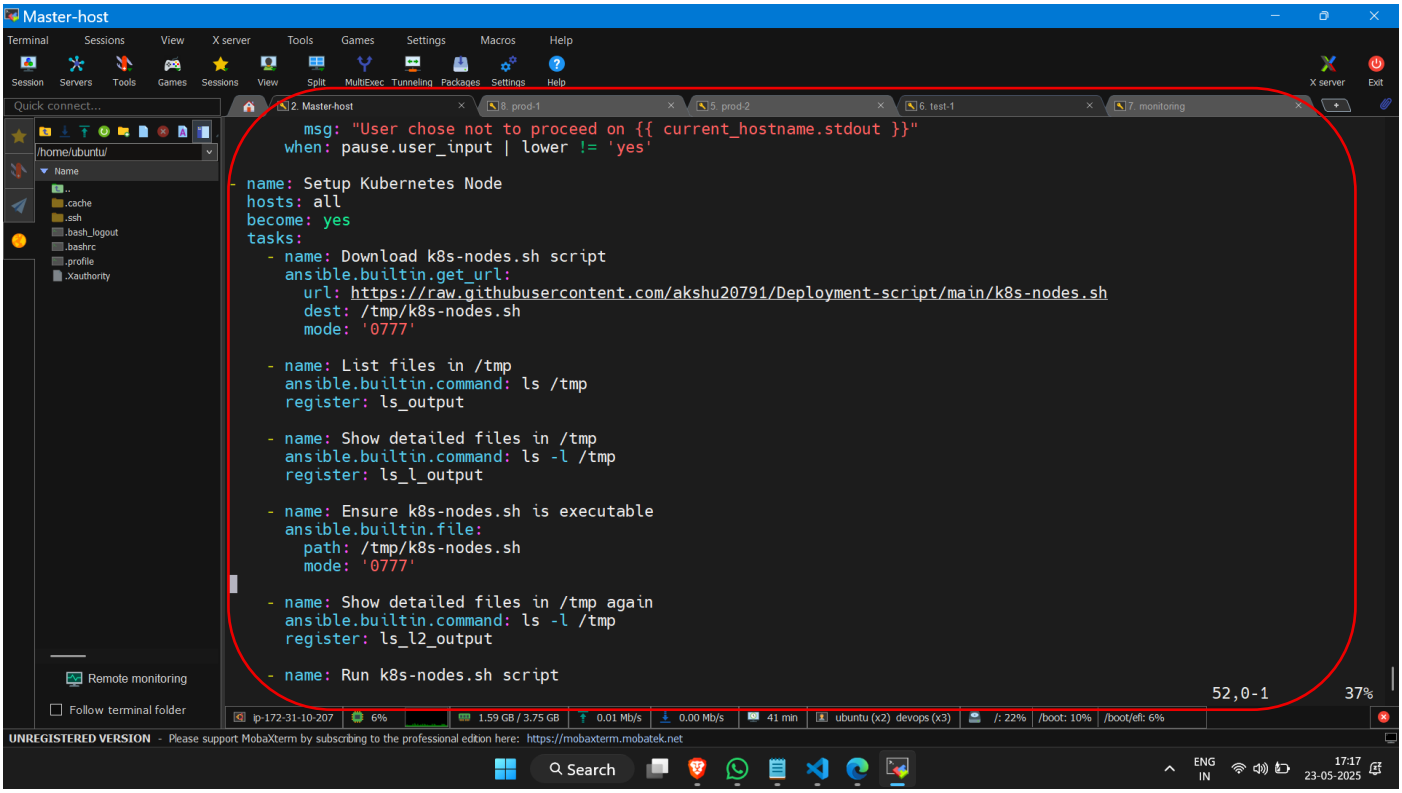
Now generate a Kubernetes join key in the master node using command " kubeadm create –print-join-command" and copy the token generated for further use

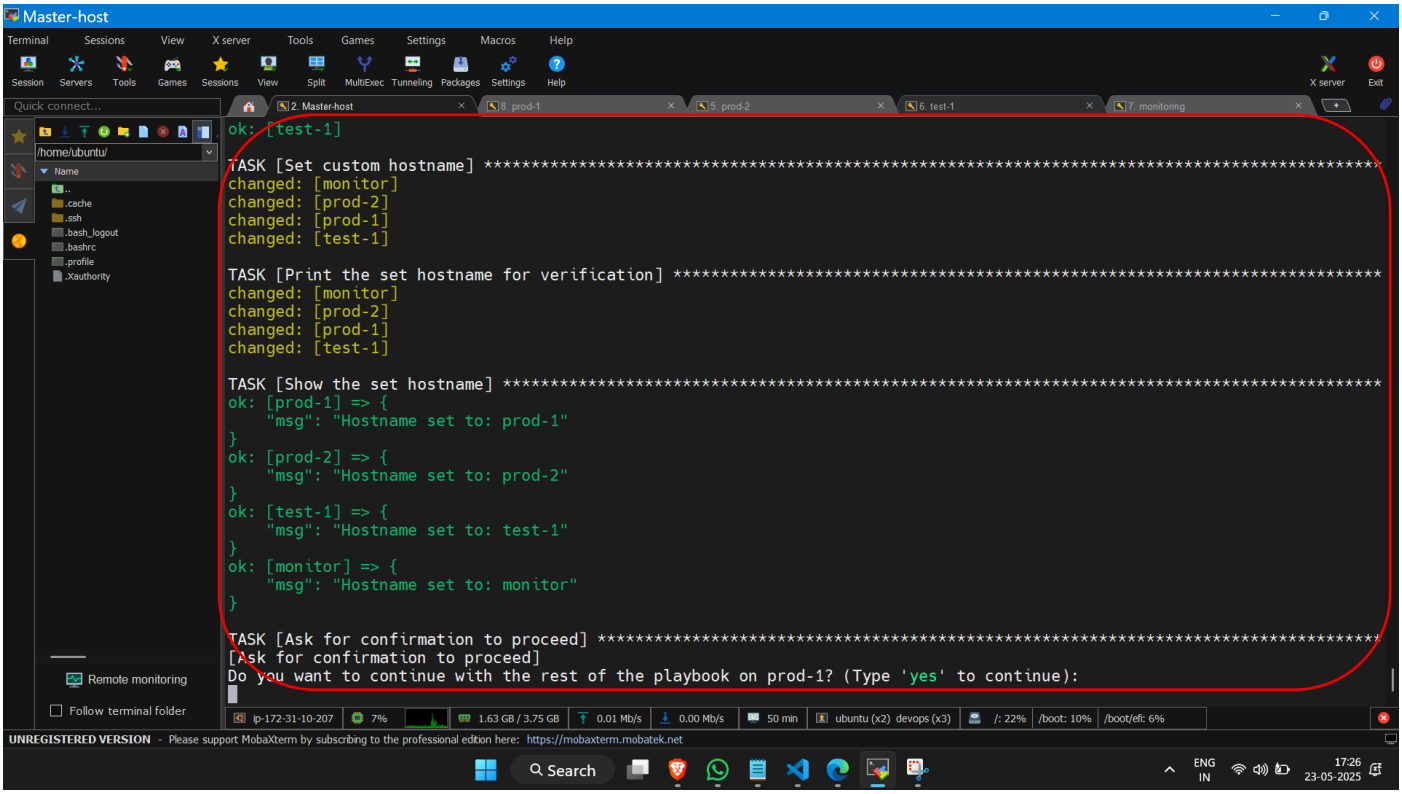


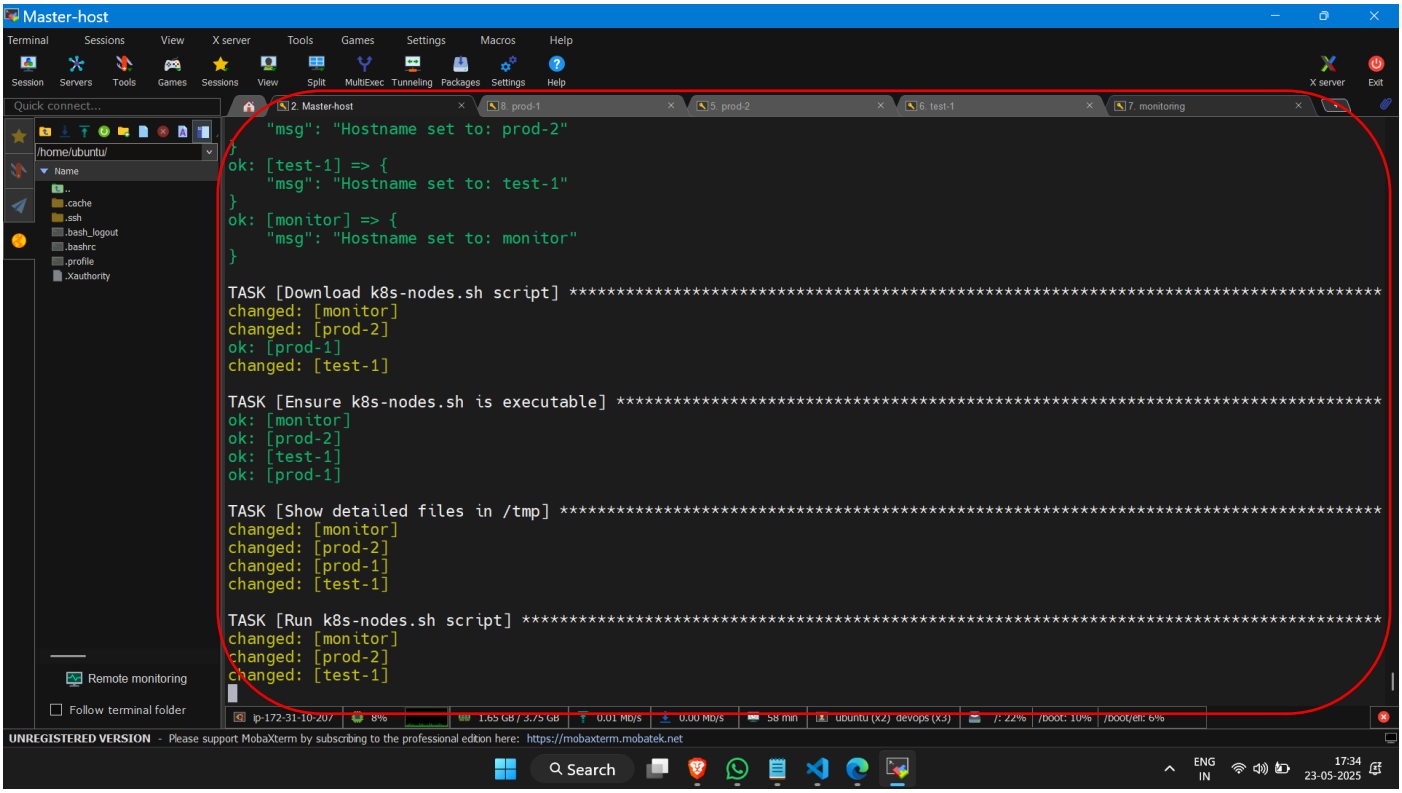
Now write a ansible playbook to connect the nodes with the kubernetes cluster master ans also set the hostname of each instance accordingly to the hosts file which has been written previously

This book will install the kubernetes slave configuration , network bridging and Kubernetes join token with cri socket

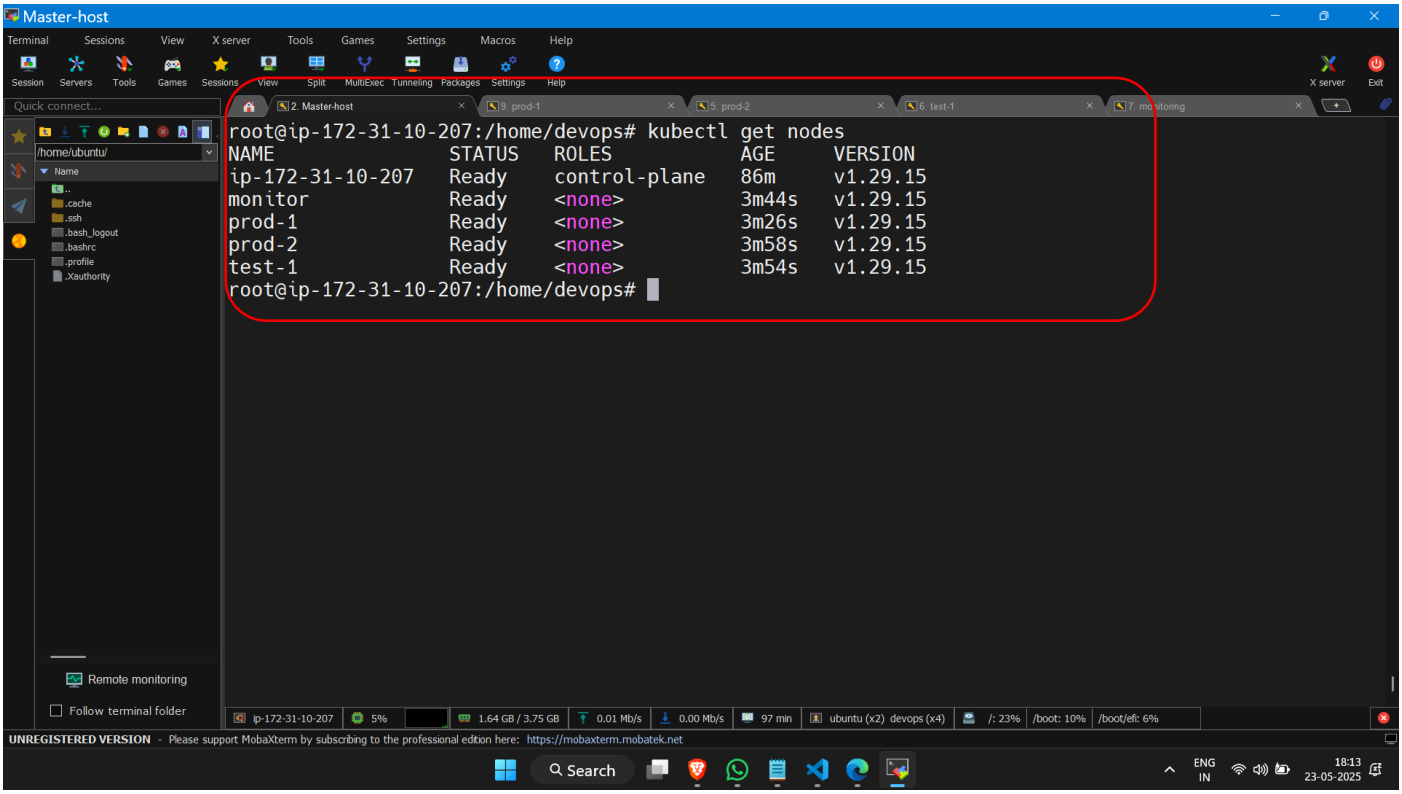



Master-host
msg: "User chose not to proceed on {{ current_hostname.stdout }}"
when: pause.user_input | lower != 'yes'
- name: Setup Kubernetes Node
hosts: all
become: yes
tasks:
- name: Download k8s-nodes.sh script
ansible.builtin.get_url:
url: https://raw.githubusercontent.com/akshu20791/Deployment-script/main/k8s-nodes.sh
dest: /tmp/k8s-nodes.sh
mode: '0777'
- name: List files in /tmp
ansible.builtin.command: ls /tmp
register: ls_output
- name: Show detailed files in /tmp
ansible.builtin.command: ls -l /tmp
register: ls_l_output
- name: Ensure k8s-nodes.sh is executable
ansible.builtin.file:
path: /tmp/k8s-nodes.sh
mode: '0777'
- name: Show detailed files in /tmp again
ansible.builtin.command: ls -l /tmp
register: ls_l2_output
- name: Run k8s-nodes.sh script
52,0-1
37%
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net


Master-host
ok: [test-1]
TASK [Set custom hostname]
changed: [monitor]
changed: [prod-2]
changed: [prod-1]
changed: [test-1]
TASK [Print the set hostname for verification]
changed: [monitor]
changed: [prod-2]
changed: [prod-1]
changed: [test-1]
TASK [Show the set hostname]
ok: [prod-1] => {
"msg": "Hostname set to: prod-1"
}
ok: [prod-2] => {
"msg": "Hostname set to: prod-2"
}
ok: [test-1] => {
"msg": "Hostname set to: test-1"
}
ok: [monitor] => {
"msg": "Hostname set to: monitor"
}
TASK [Ask for confirmation to proceed]
[Ask for confirmation to proceed]
Do you want to continue with the rest of the playbook on prod-1? (Type 'yes' to continue):
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

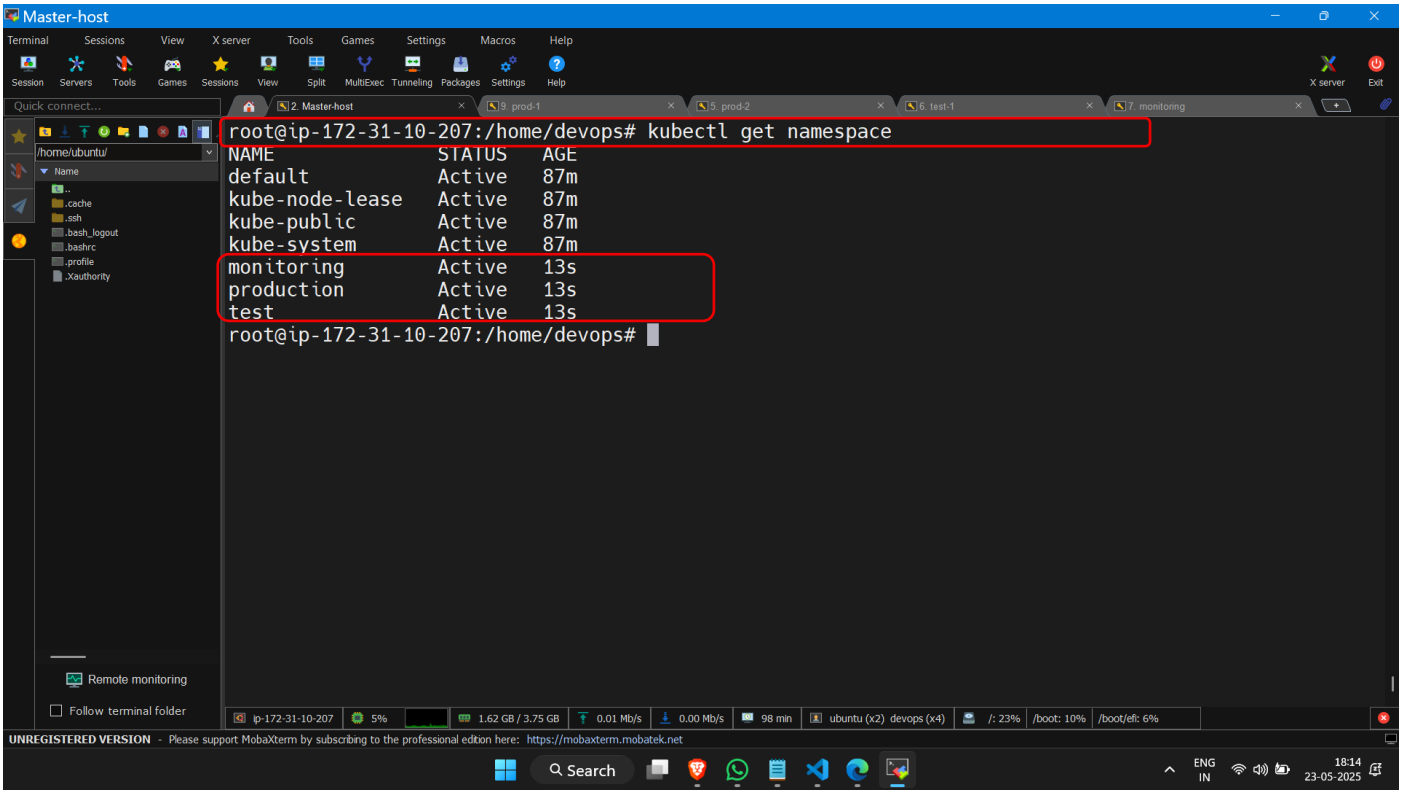


After successful run of ansible playbook check if the nodes got connected successfully by running the command “kubectl get nodes”



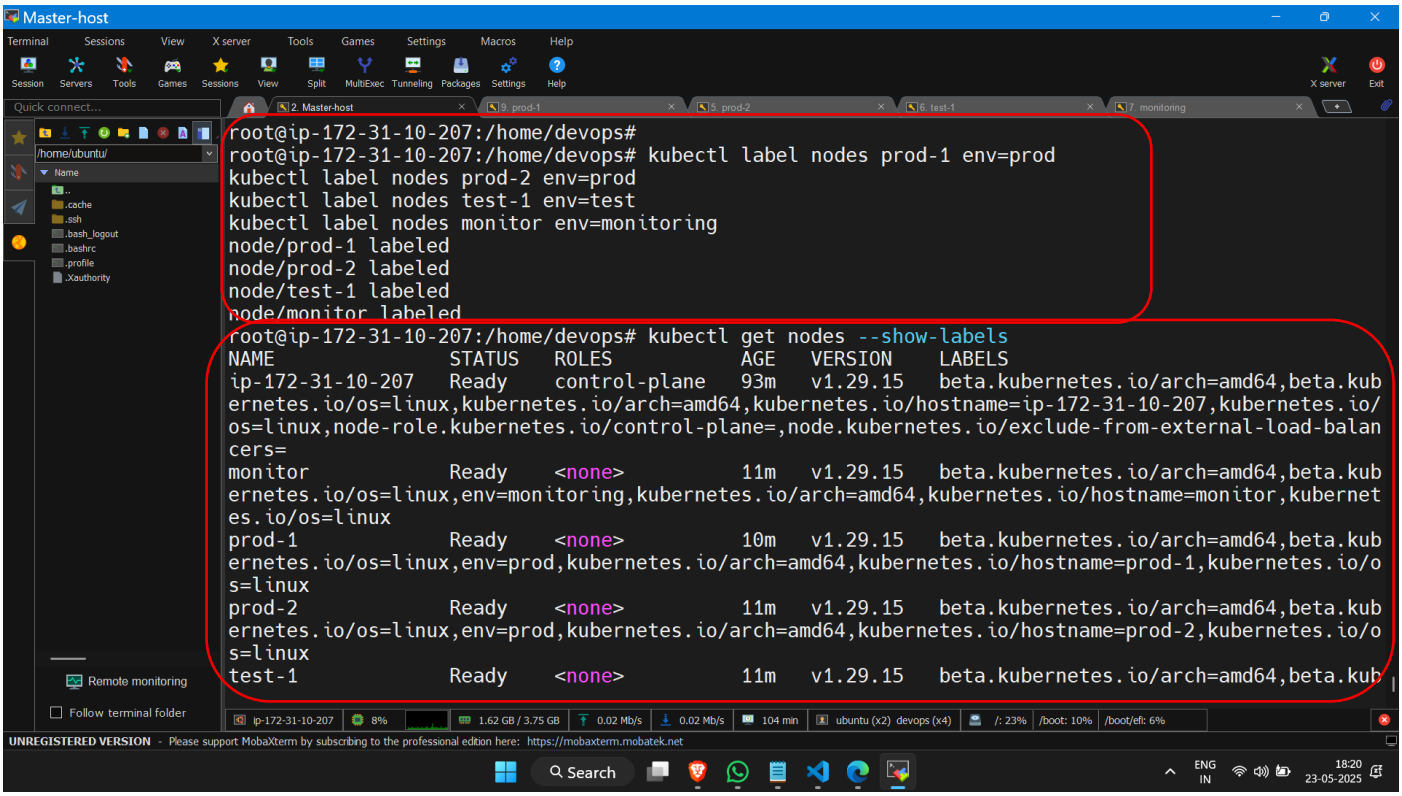
Create kubernetes namespace in order to separate the nodes using the namespace using command

“kubectl create namespace <namespace-name>” here I have created 3 namespace test, monitoring, production and verify them by command “kubectl get namespace”

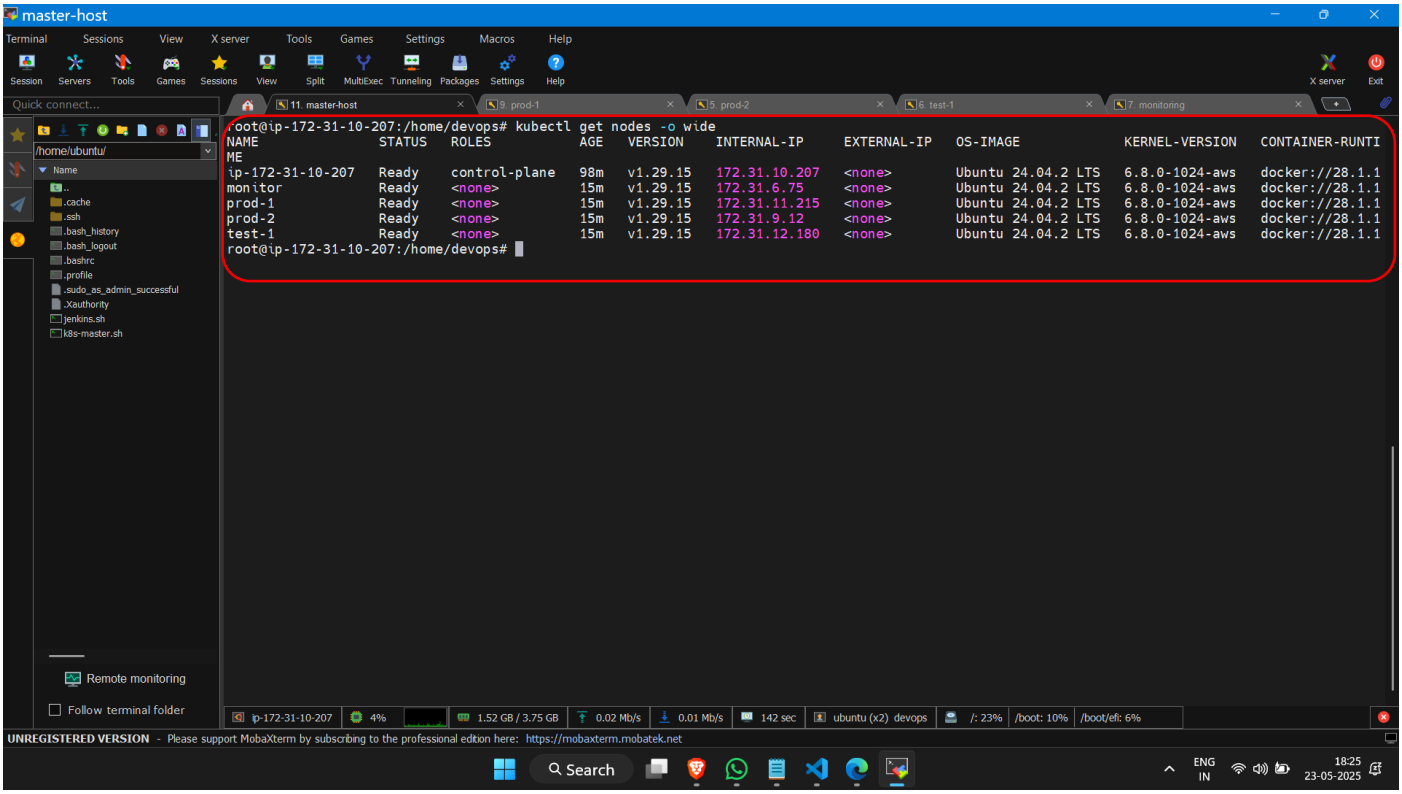


Now set the labels each node using env tag using command “kubectl label nodes <node-name> env=<name>” here I have set 3 env i.e prod, test, & monitoring

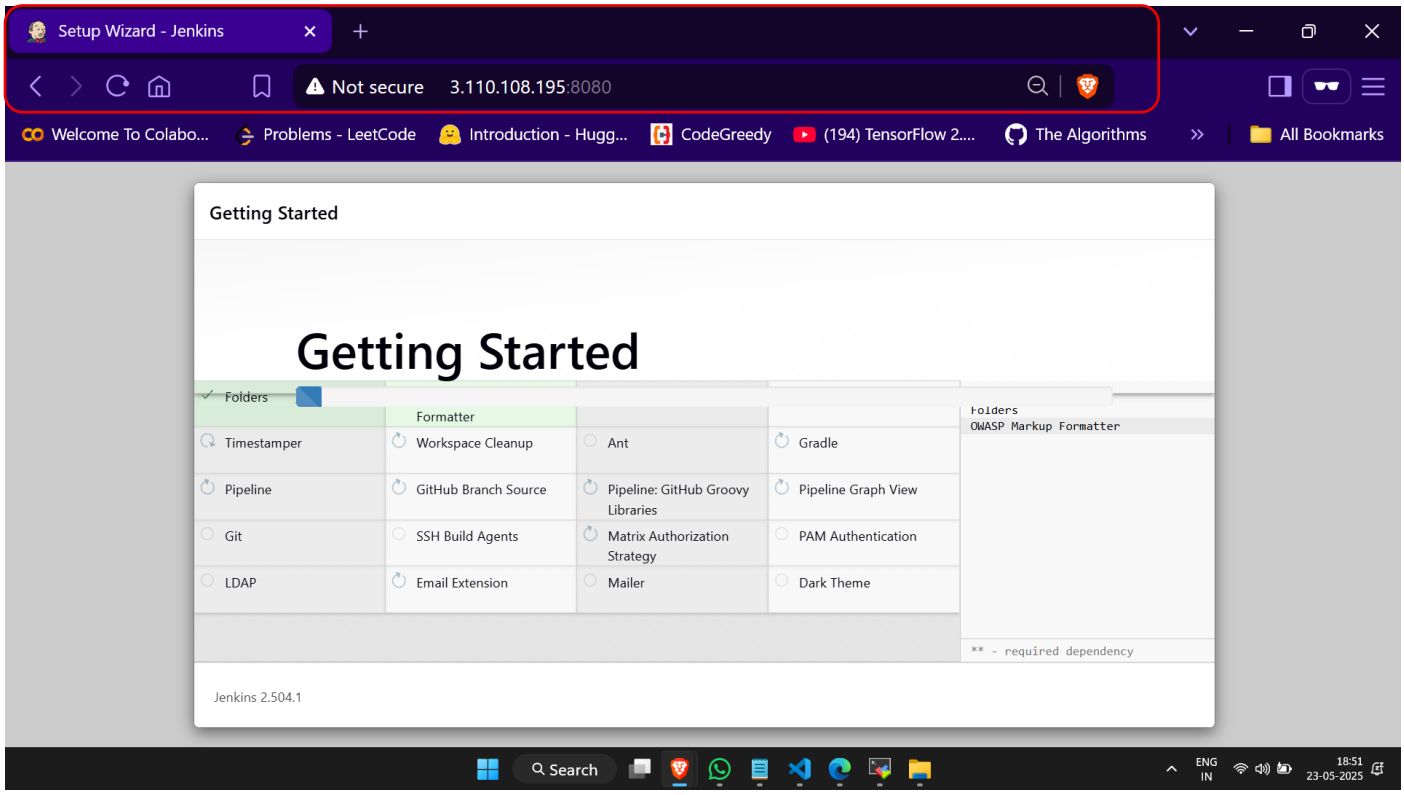
Check the labels using command kubectl get nodes –show-labels”



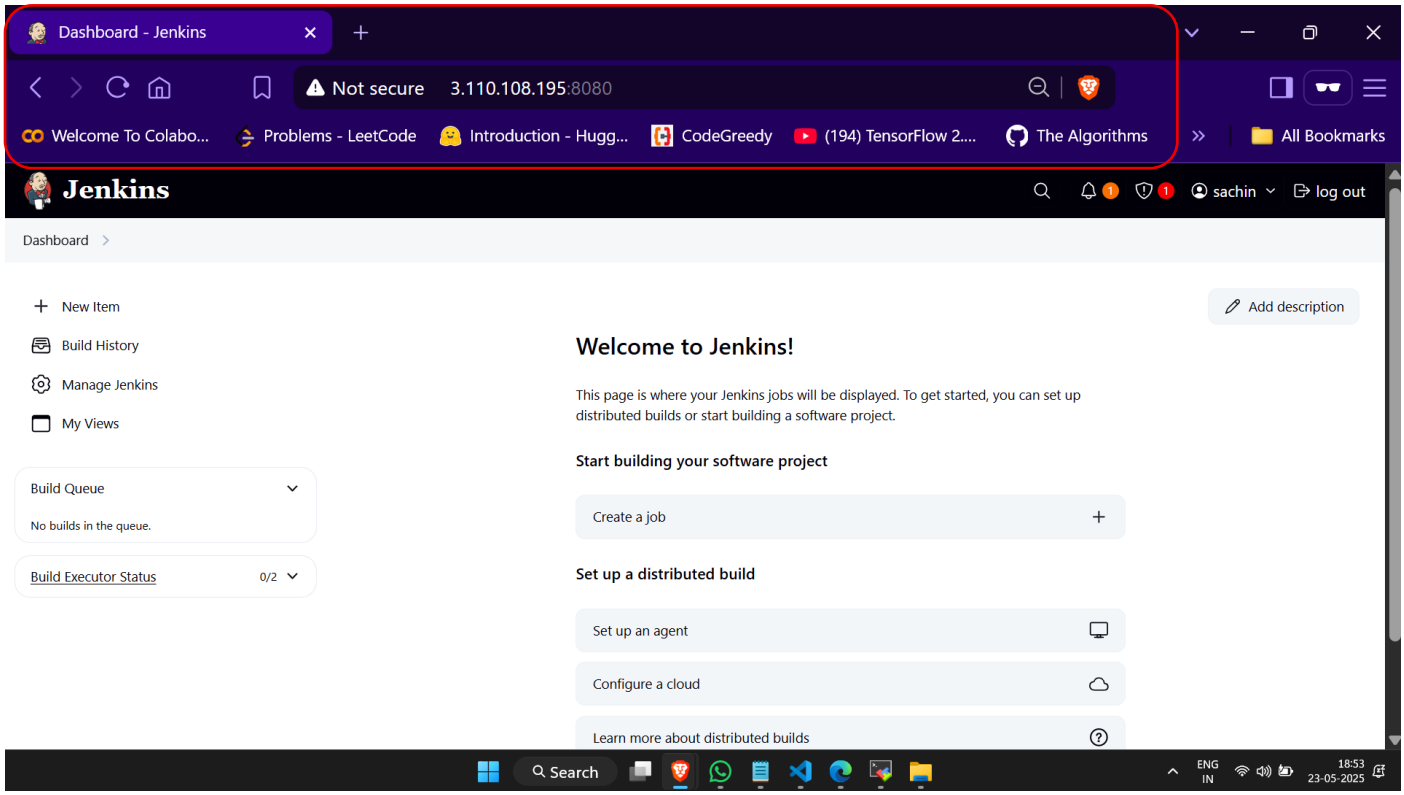
Now verify the nodes status using command “kubectl get nodes -o wide”



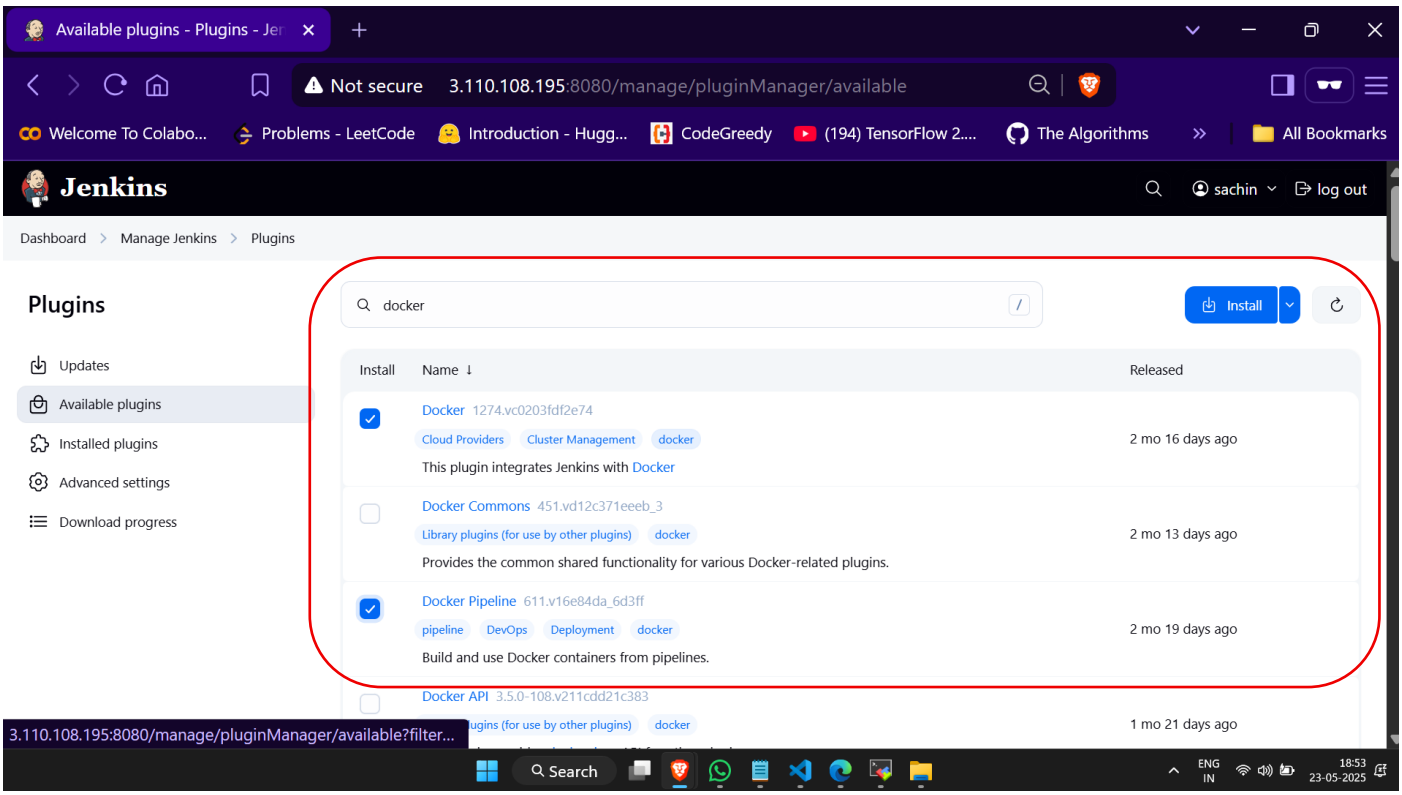
Now access the Jenkins service using the public ip on port 8080



Login into the Jenkins using initial password which will get at path /var/jenkins_home/secrets/initialAdminPassword

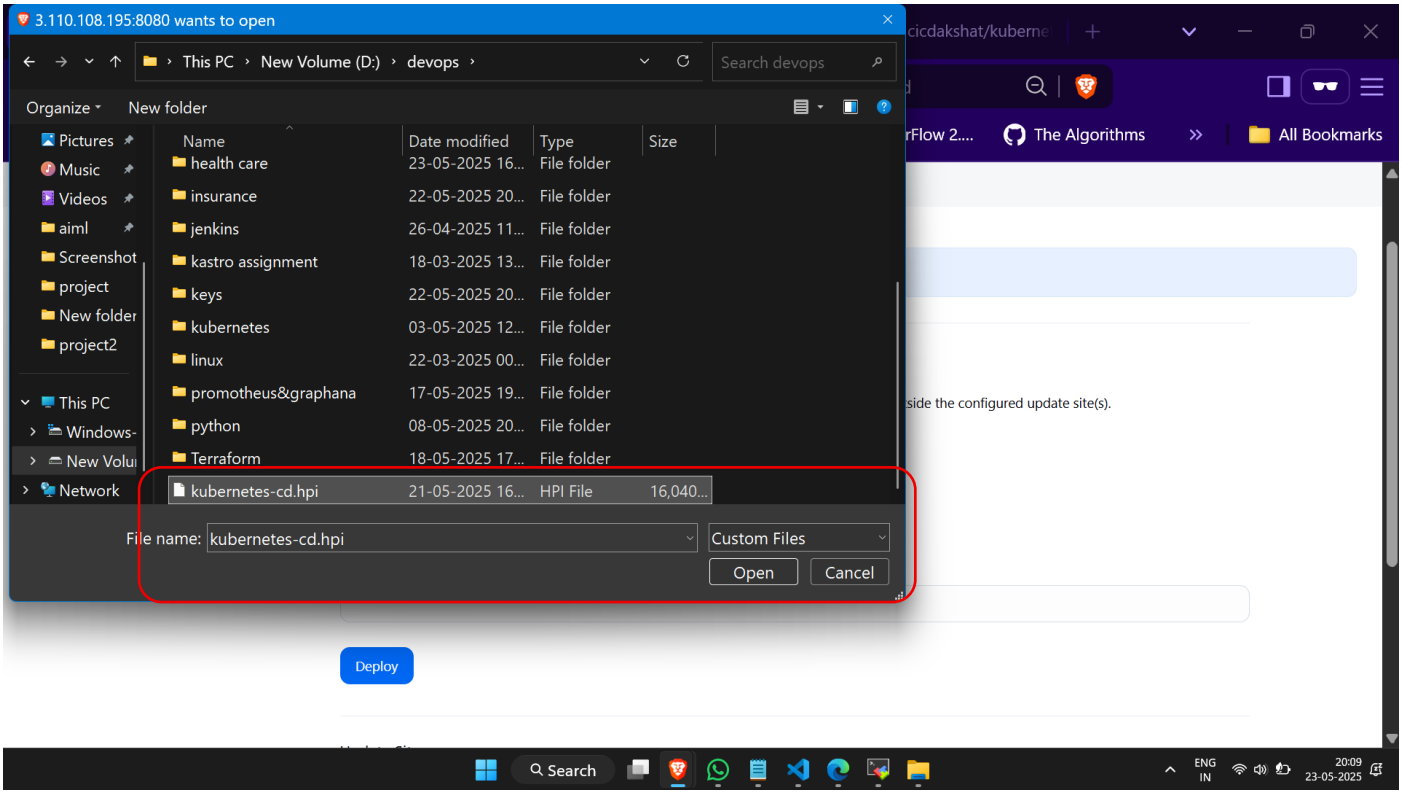


Navigate to manage Jenkins -> plugins ->available plugins -> docker install docker plugins



Download Kubernetes plugin the HPI file manually from the repository "https://github.com/akshu20791/cicdakshat " and load it manually into the jenkins

Navigate to manage Jenkins -> plugins ->advanced setting -> then upload the hpi file and install it



Run this commands in the master machine and get the configid

cd ~

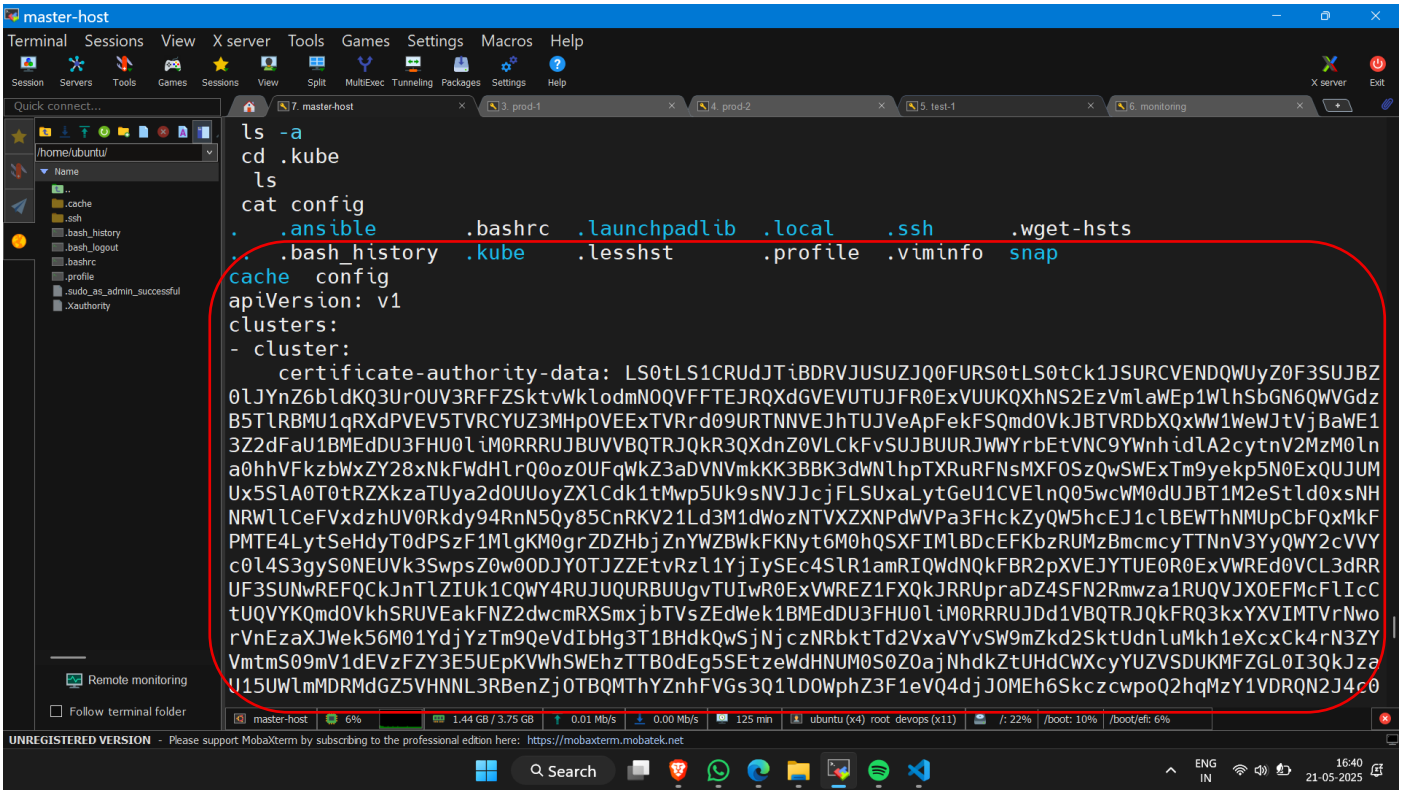
ls -a

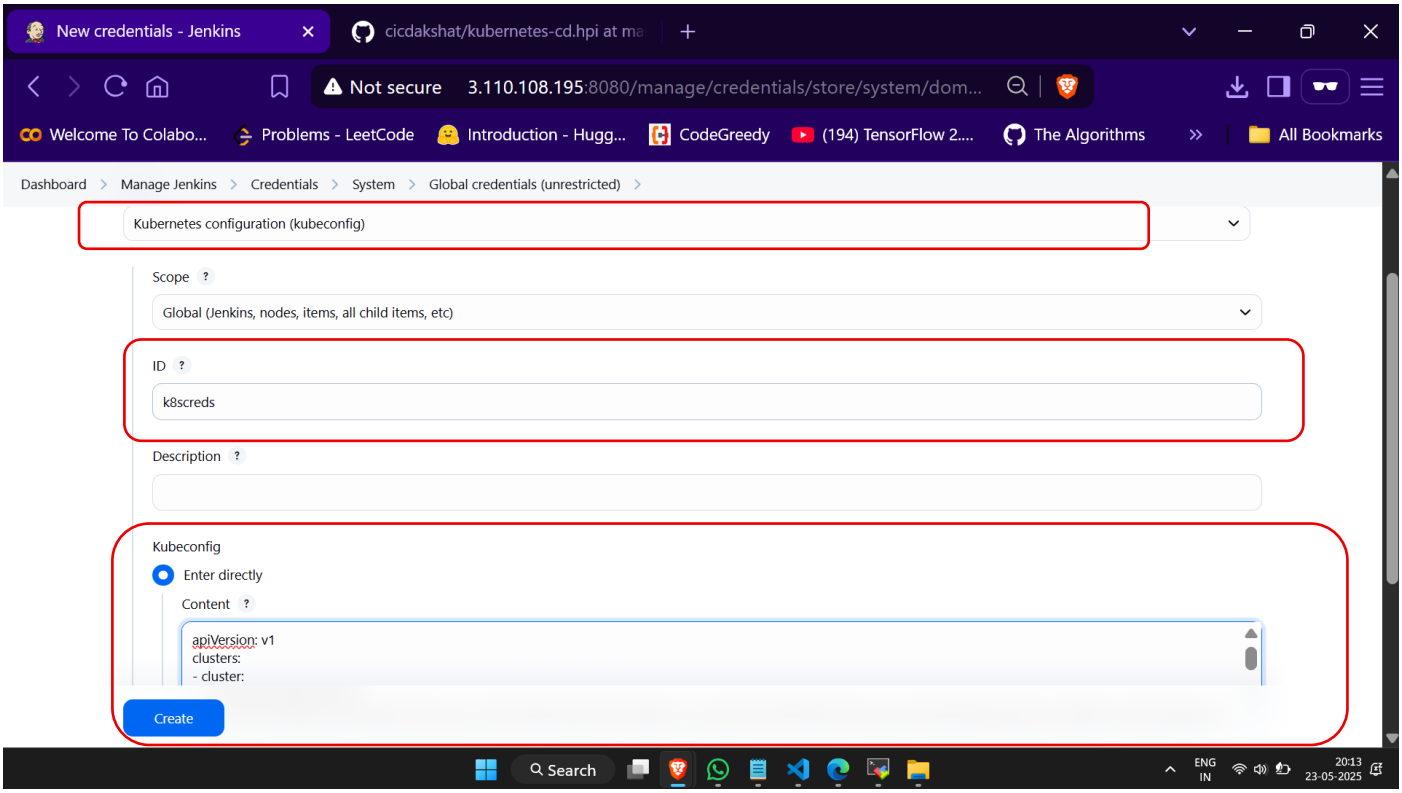
cd .kube

ls

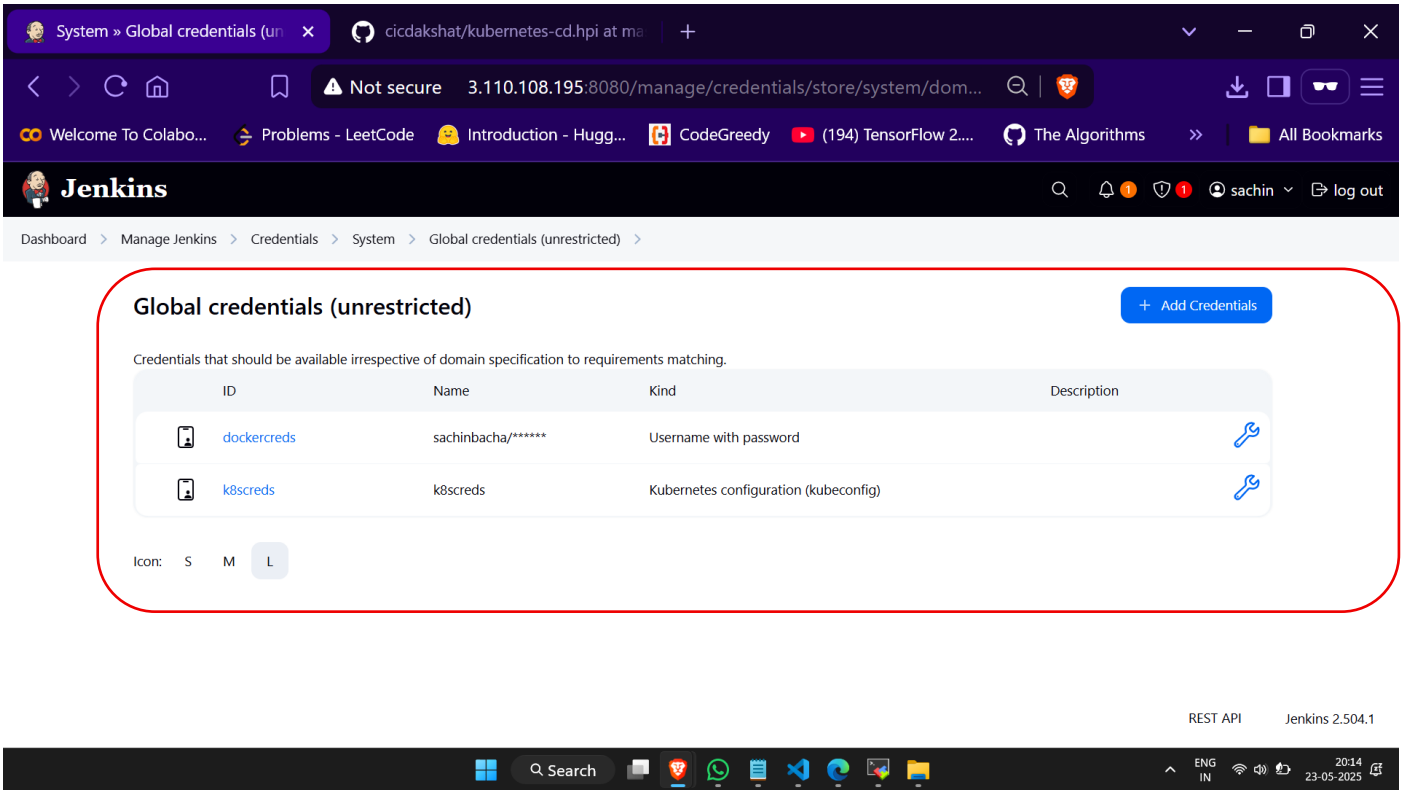
cat config

copy the whole config id which has came up for furhter adding it to jenkins credentails

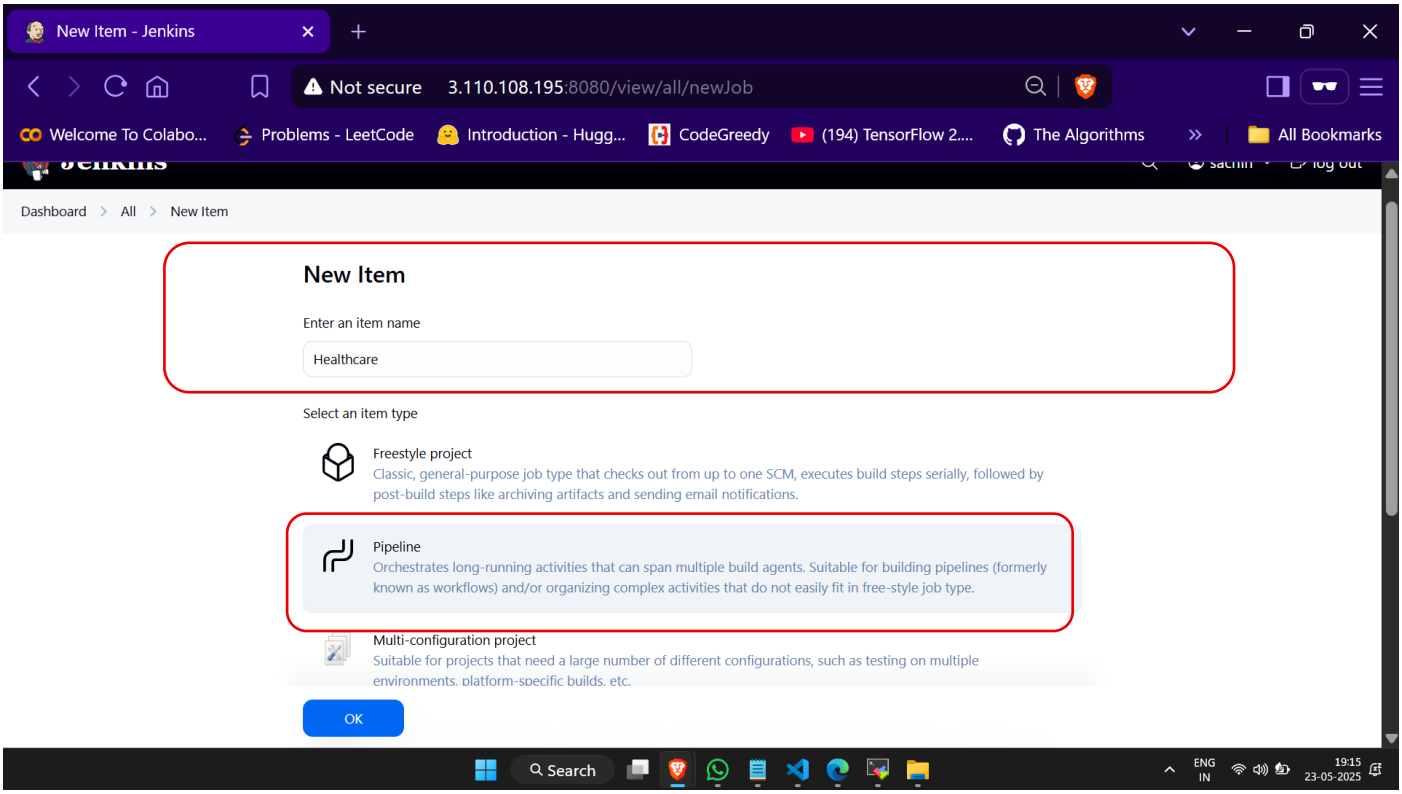

master-host
Terminal Sessions View X server Tools Games Settings Macros Help
ls -a
cd .kube
ls
cat config
. .ansible .bashrc .launchpadlib .local .ssh .wget-hsts
.. .bash_history .kube .lesshst .profile .viminfo snap
cache config
apiVersion: v1
clusters:
- cluster:
certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURCVENDQWUyZ0F3SUJBZ
0lJYnZ6bldKQ3UrOUV3RFFZSktvWklodmNOQVFFTEJRQXdGVEVUTUJFR0ExVUUKQXhNS2EzVmlaWEp1WlhSbGN6QWVGdz
B5TlRBMU1qRXdPVEV5TVRCYUZ3MHpOVEExTVRrd09URTNNVEJhTUJVeApFekFSQmdOVkJBTVRDbXQxWW1WeWJtVjBaWE1
3Z2dFaU1BMEdDU3FHU0liM0RRRUJBUVVBQTRJQkR3QXdnZ0VLCkFvSUJBUURJWWYrbEtVNC9YWnhidlA2cytnV2MzM0ln
a0hhVFkzbWxZY28xNkFWdHlrQ0ozOUFqWkZ3aDVNVmkKK3BBK3dWNlhpTXRuRFNsMXFOSzQwSWExTm9yekp5N0ExQUJUM
Ux5SlA0T0tRZXkzaTUya2dOUUoyZXlCdk1tMwp5Uk9sNVJJcjFLSUxaLytGeU1CVElnQ05wcWM0dUJBT1M2eStld0xsNH
NRWllCeFVxdzhUV0Rkdy94RnN5Qy85CnRKV21Ld3M1dWozNTVXZXNPdWVPa3FHckZyQW5hcEJ1clBEWThNMUpCbFQxMkF
PMTE4LytSeHdyT0dPSzF1MlgKM0grZDZHbjZnYWZBWkFKNyt6M0hQSXFIMlBDcEFKbzRUMzBmcmcyTTNnV3YyQWY2cVVY
c0l4S3gyS0NEUVk3SwpsZ0w0ODJYOTJZZEtvRzl1YjIySEc4SlR1amRIQWdNQkFBR2pXVEJYTUE0R0ExVWREd0VCL3dRR
UF3SUNwREFQCkJnTlZIUk1CQWY4RUJUQURBUUgvTUIwR0ExVWREZ1FXQkJRRUpraDZ4SFN2Rmwza1RUQVJXOEFMcFlIcC
tUQVYKQmdOVkhSRUVEakFNZ2dwcmRXSmxjbTVsZEdWek1BMEdDU3FHU0liM0RRRUJDd1VBQTRJQkFRQ3kxYXVIMTVrNwo
rVnEzaXJWek56M01YdjYzTm9QeVdIbHg3T1BHdkQwSjNjczNRbktTd2VxaVYvSW9mZkd2SktUdnluMkh1eXcxCk4rN3ZY
VmtmS09mV1dEVzFZY3E5UEpKVWhSWEhzTTBOdEg5SEtzeWdHNUM0S0ZOajNhdkZtUHdCWXcyYUZVSDUKMFZGL0I3QkJza
U15UWlmMDRMdGZ5VHNNL3RBenZjOTBQMThYZnhFVGs3Q1lDOWphZ3F1eVQ4djJOMEh6SkczcwpoQ2hqMzY1VDRQN2J4c0
Remote monitoring
Follow terminal folder
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net
Search
ENG IN
16:40 21-05-2025


New credentials - Jenkins
cicdakshat/kubernetes-cd.hpi at ma
Not secure 3.110.108.195:8080/manage/credentials/store/system/dom...
Welcome To Colabo... Problems - LeetCode Introduction - Hugg... CodeGreedy (194) TensorFlow 2.... The Algorithms All Bookmarks
Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >
Kubernetes configuration (kubeconfig)
Scope ?
Global (Jenkins, nodes, items, all child items, etc)
ID ?
k8screds
Description ?
Kubeconfig
Enter directly
Content ?
apiVersion: v1
clusters:
- cluster:
Create
Search
ENG IN
20:13 23-05-2025

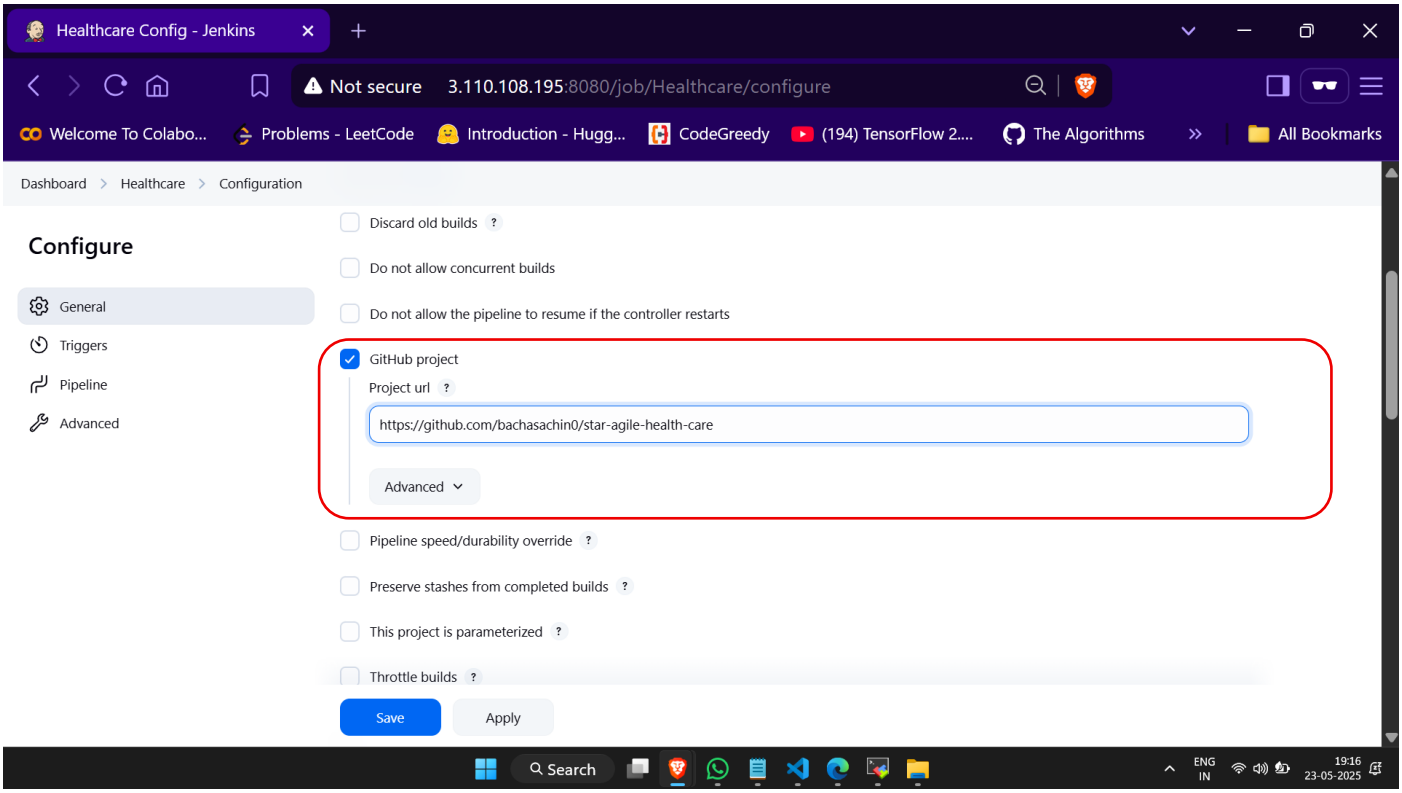
In the credentials add the docker credentials (dockercreds) and kuberntes credentials (k8screds)



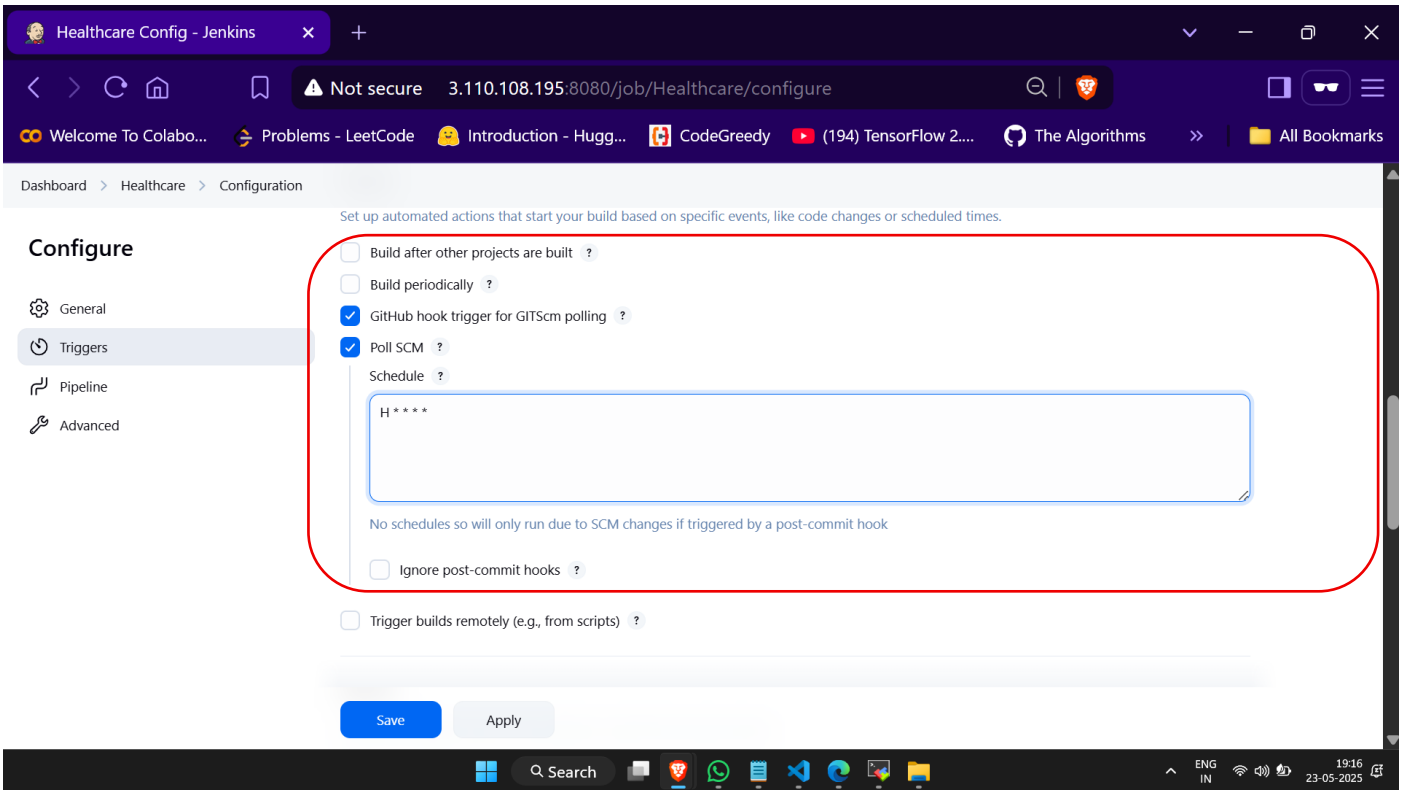
Now create a new item with name Healthcare and select the item type as pipeline



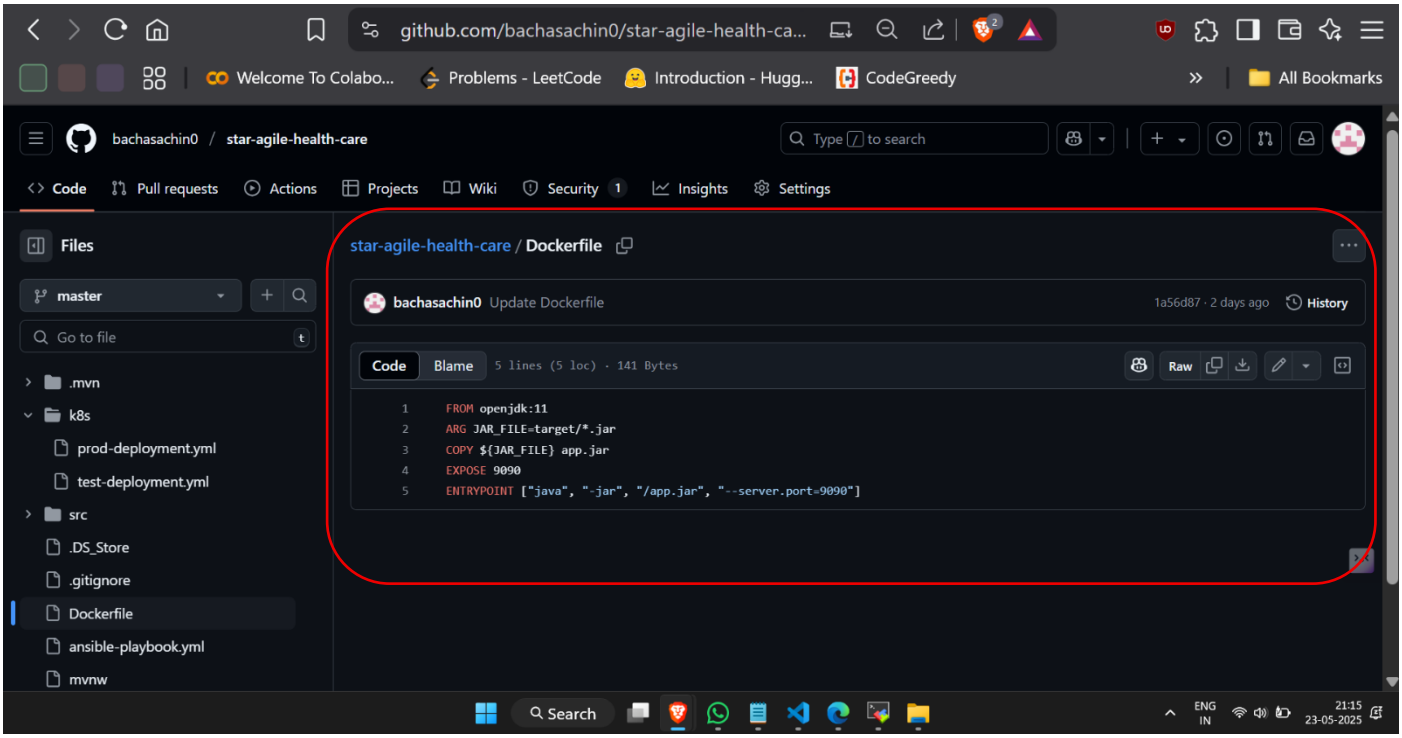
Add the github repository in the general configurations add the forked repository



Add git triggers

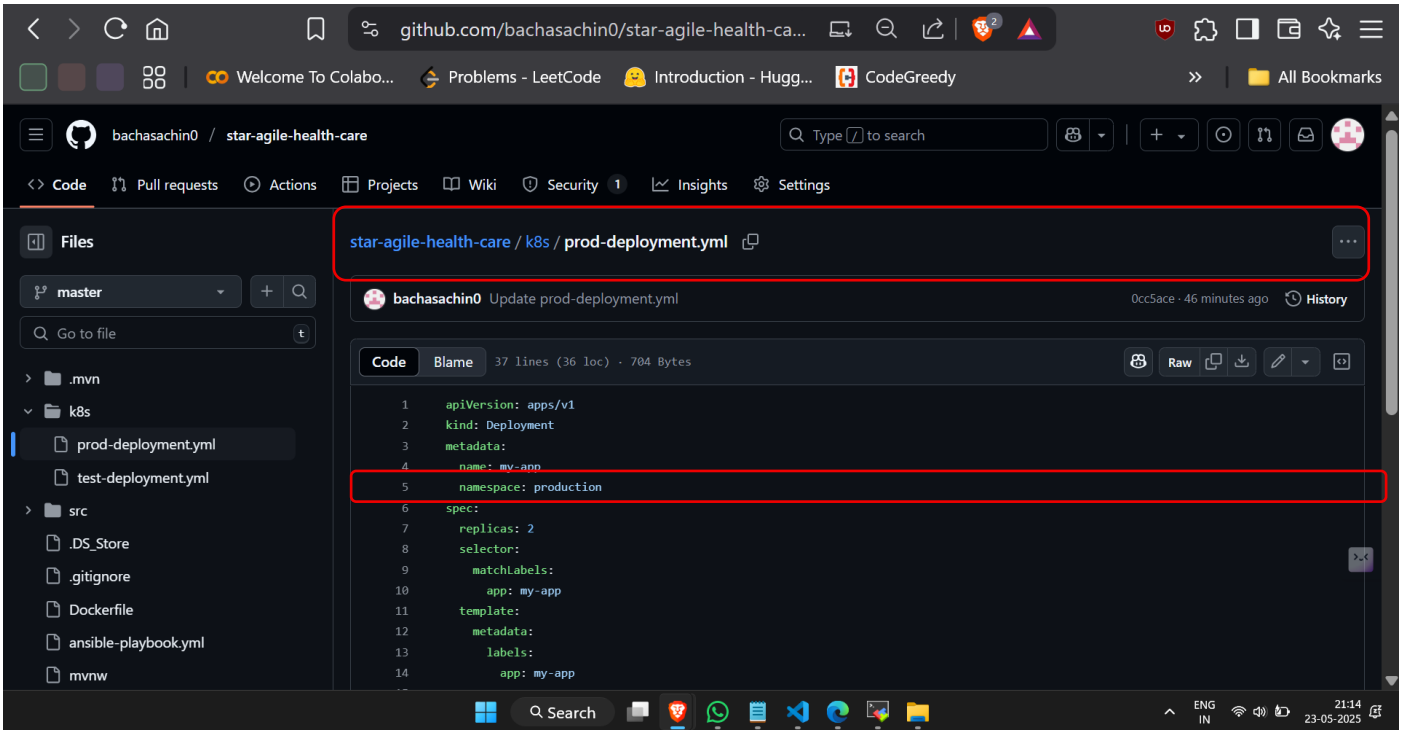


Write the Dockerfile

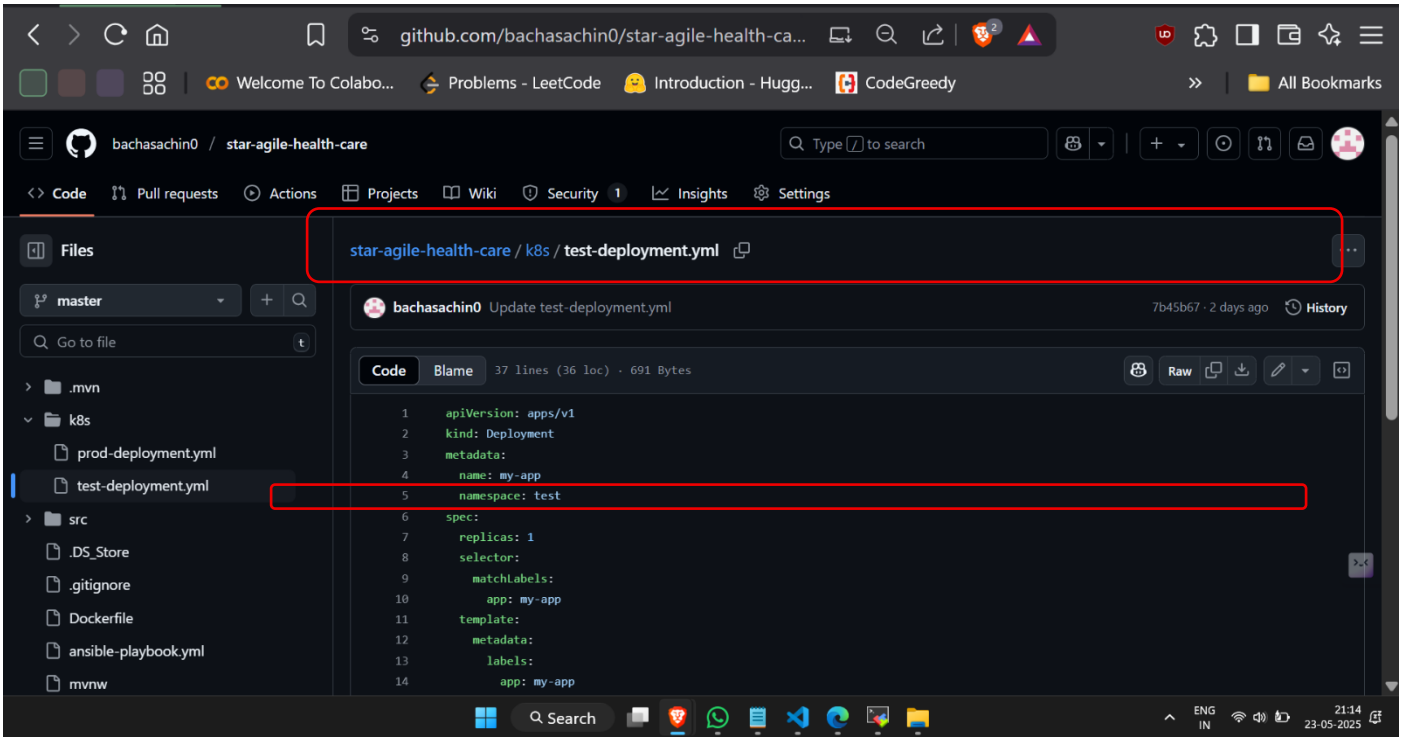


Create a k8s folder and write the deployment scripts of both production and test deployment with namespace production and tag prod

This will ensure that the applications are deployed on the assigned tags and namespace



Also write the test deployment script with the namespace test and tag env of test this will ensure that the application is deployed on the specified nodes only



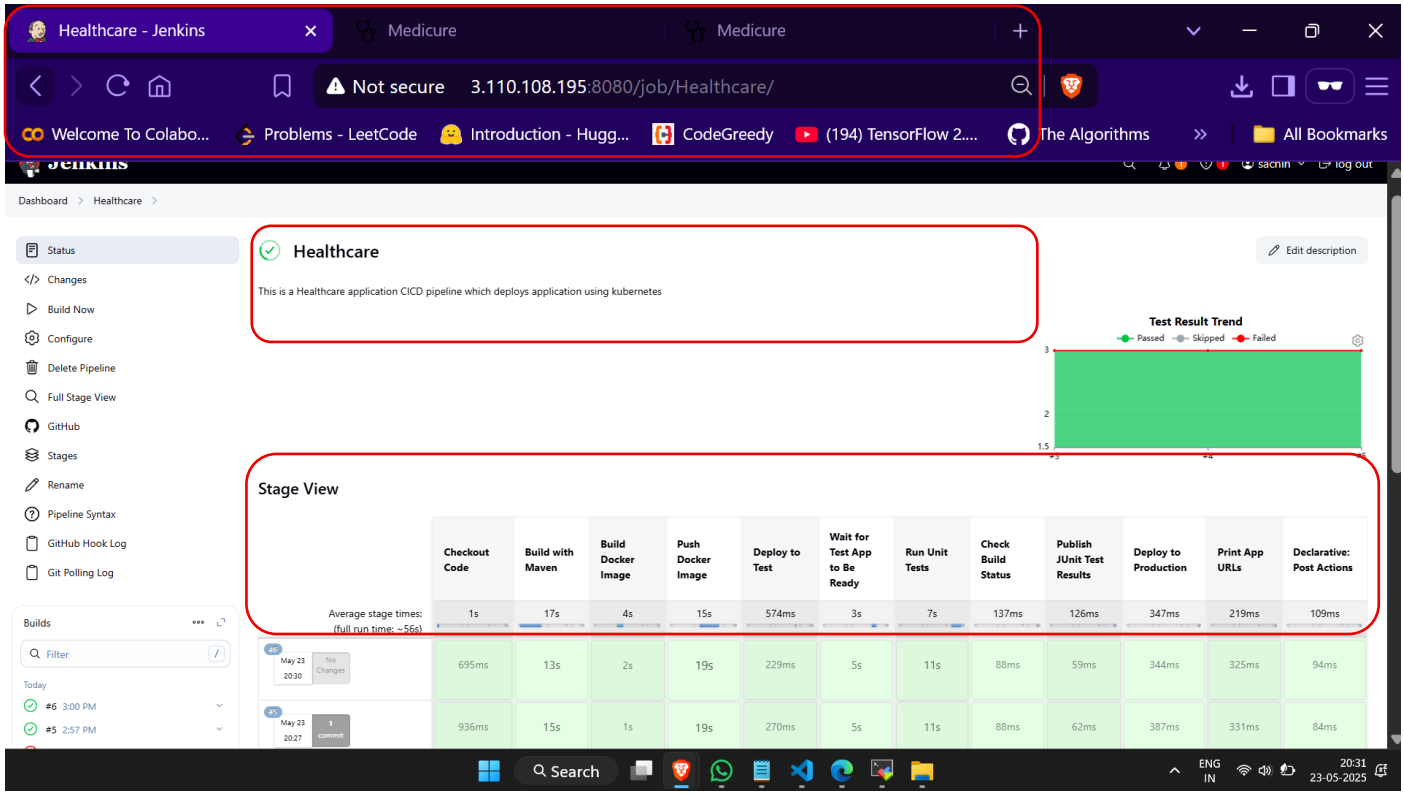
Write the Jenkins script to deploy the application

```
pipeline {
  agent any

  stages {
    stage('Checkout Code') {
      steps {
        git url: 'https://github.com/bachasachin0/star-agile-health-care.git', branch: 'master'
      }
    }

    stage('Build with Maven') {
      steps {
        sh './mvnw clean package'
      }
    }

    stage('Build Docker Image') {
      steps {
        sh 'docker build -t sachinbacha/medicure:latest .'
      }
    }
```

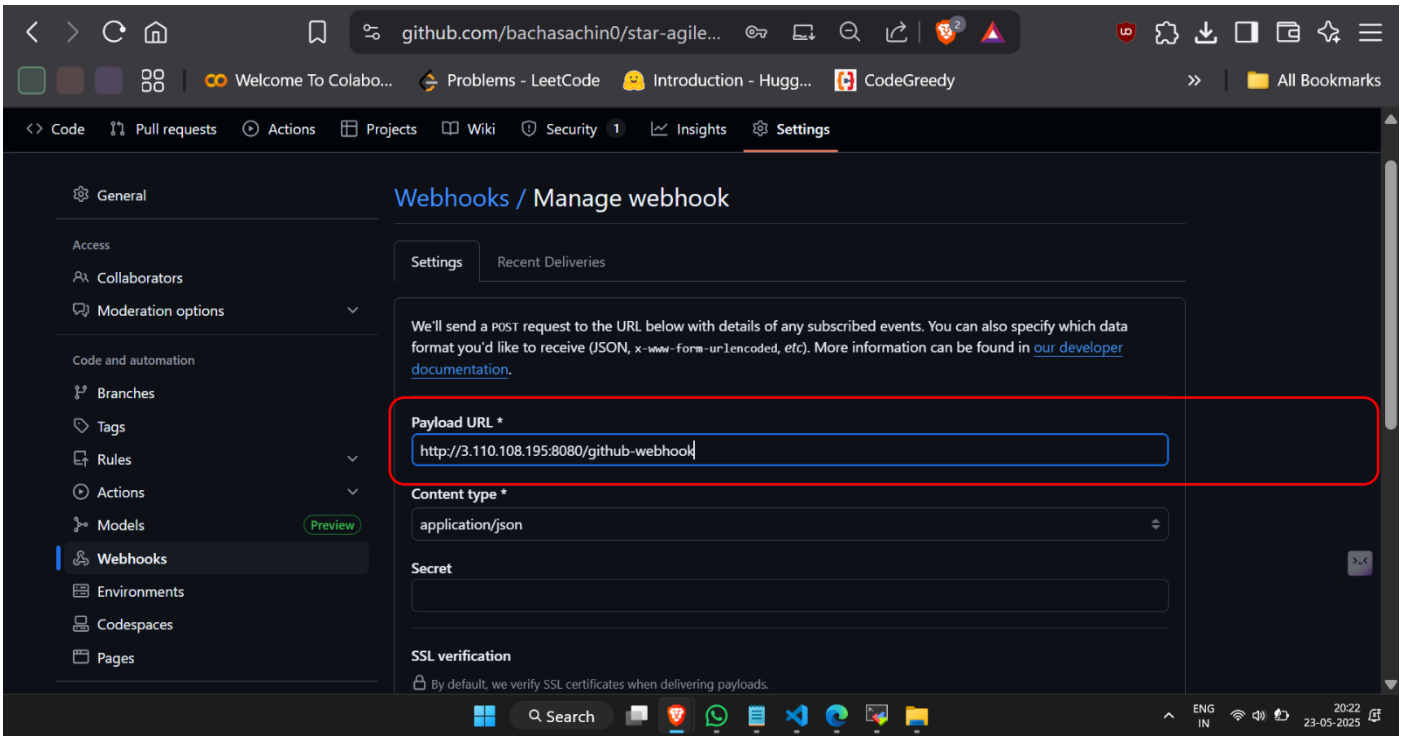
```
        stage('Push Docker Image') {
            steps {
                withCredentials([usernamePassword(credentialsId: 'dockercreds', usernameVariable: 'USERNAME',
passwordVariable: 'PASSWORD')]) {
                    sh 'echo $PASSWORD | docker login -u $USERNAME --password-stdin'
                    sh 'docker push sachinbacha/medicure:latest'
                }
            }
        }

        stage('Deploy to Test') {
            steps {
                kubernetesDeploy(
                    kubeconfigId: 'k8spasswd',
                    configs: 'k8s/test-deployment.yml'
                )
            }
        }

        stage('Wait for Test App to Be Ready') {
          steps {
            sh 'sleep 10'
        }
       }

        stage('Run Unit Tests') {
            steps {
                sh './mvnw test'
            }
        }
        stage('Check Build Status') {
            steps {
                    script {
                        echo "Current build result: ${currentBuild.result}"
    }
  }
}
        stage('Publish JUnit Test Results') {
          steps {
            junit 'target/surefire-reports/TEST-*.xml'
    }
}
        stage('Deploy to Production') {
```

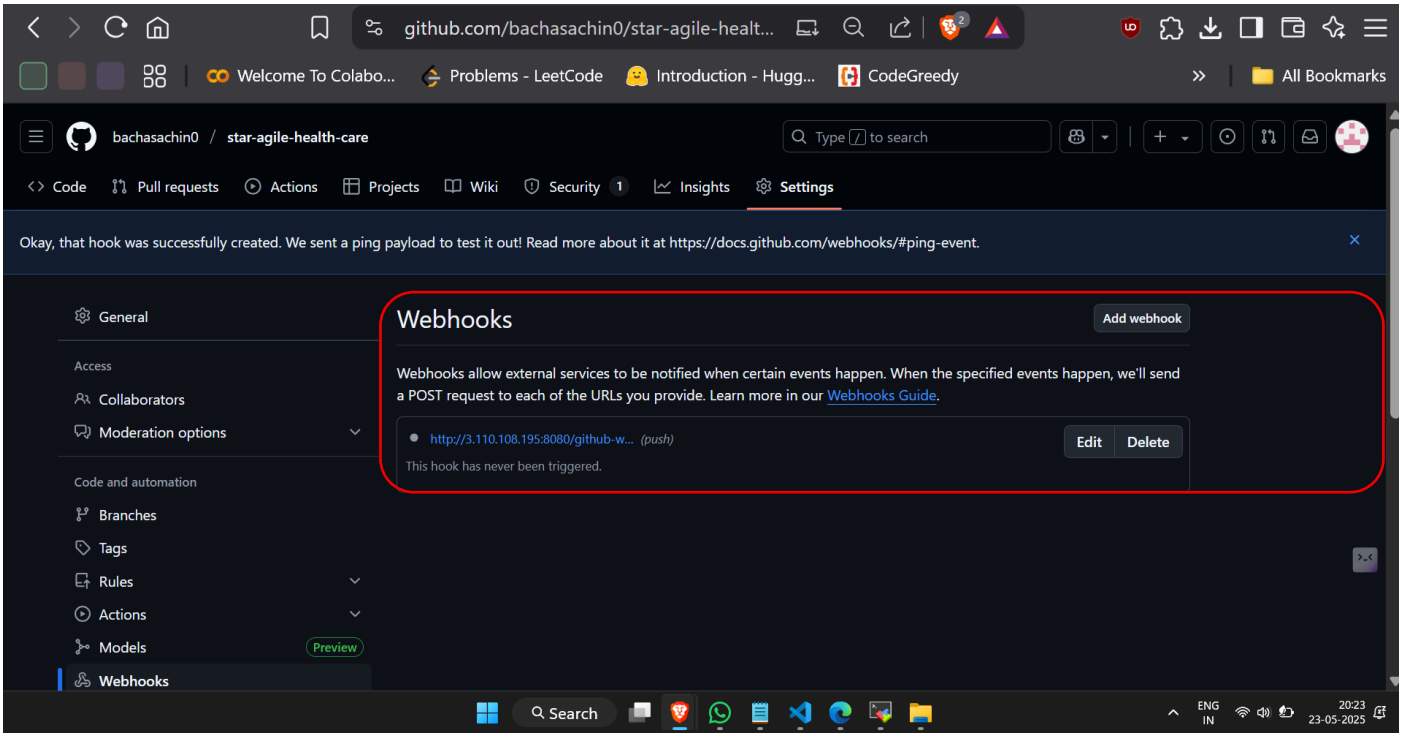
```
        when {
            expression { currentBuild.result == null || currentBuild.result == 'SUCCESS' }
        }
        steps {
            kubernetesDeploy(
                kubeconfigId: 'k8spasswd',
                configs: 'k8s/prod-deployment.yml'
            )
        }
    }

    stage('Print App URLs') {
        steps {
            sh '''
            echo "👉 Test App:     http://3.110.43.250:30090"
            echo "👉 Production App: http://3.110.43.250:30091"
            '''
        }
    }
  }

  post {
    success {
        echo "✅ Pipeline completed successfully!"
    }
    failure {
        echo "❌ Pipeline failed. Check the logs above."
    }
  }
}
```

Run the Jenkins pipeline

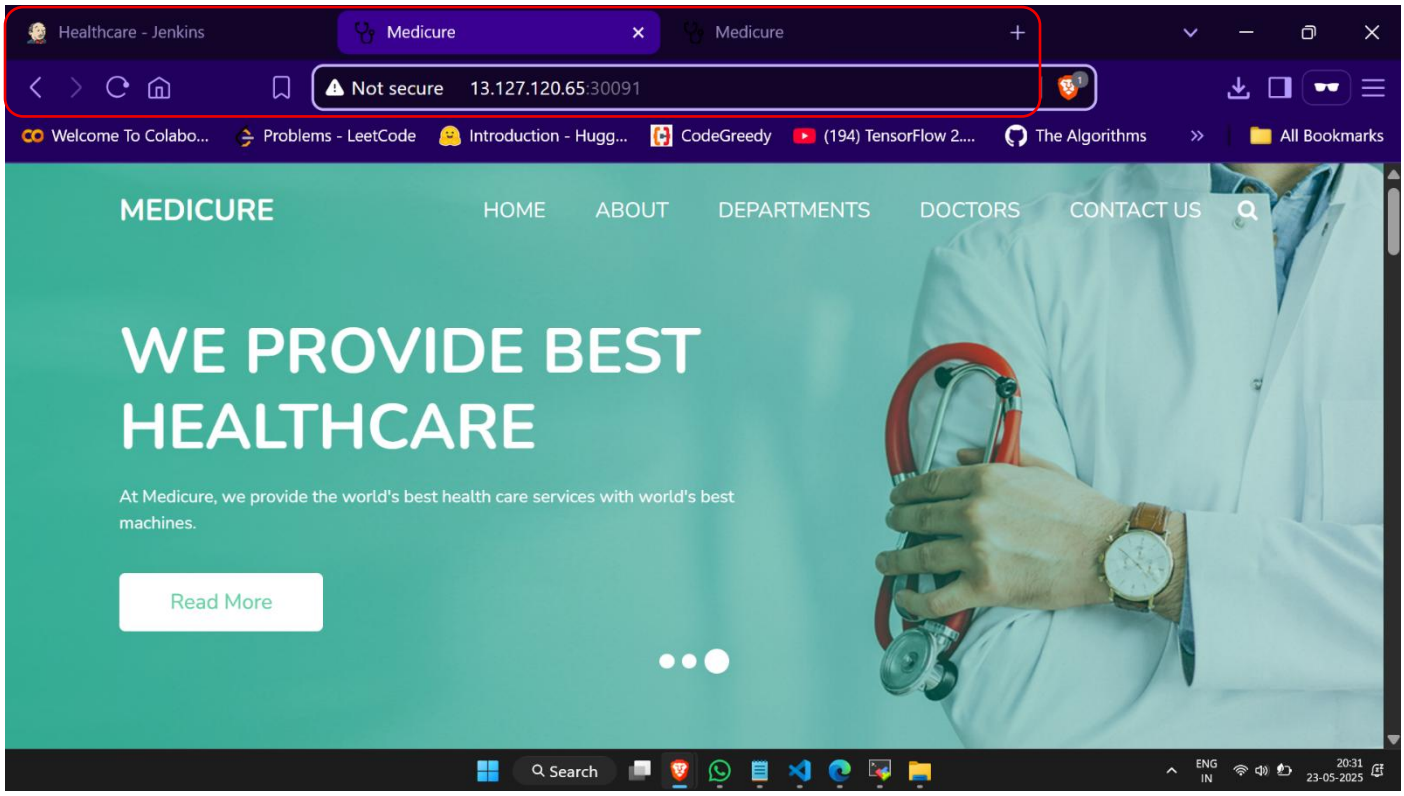


Add github webhook to the repository from settings and add the Jenkins ip address link

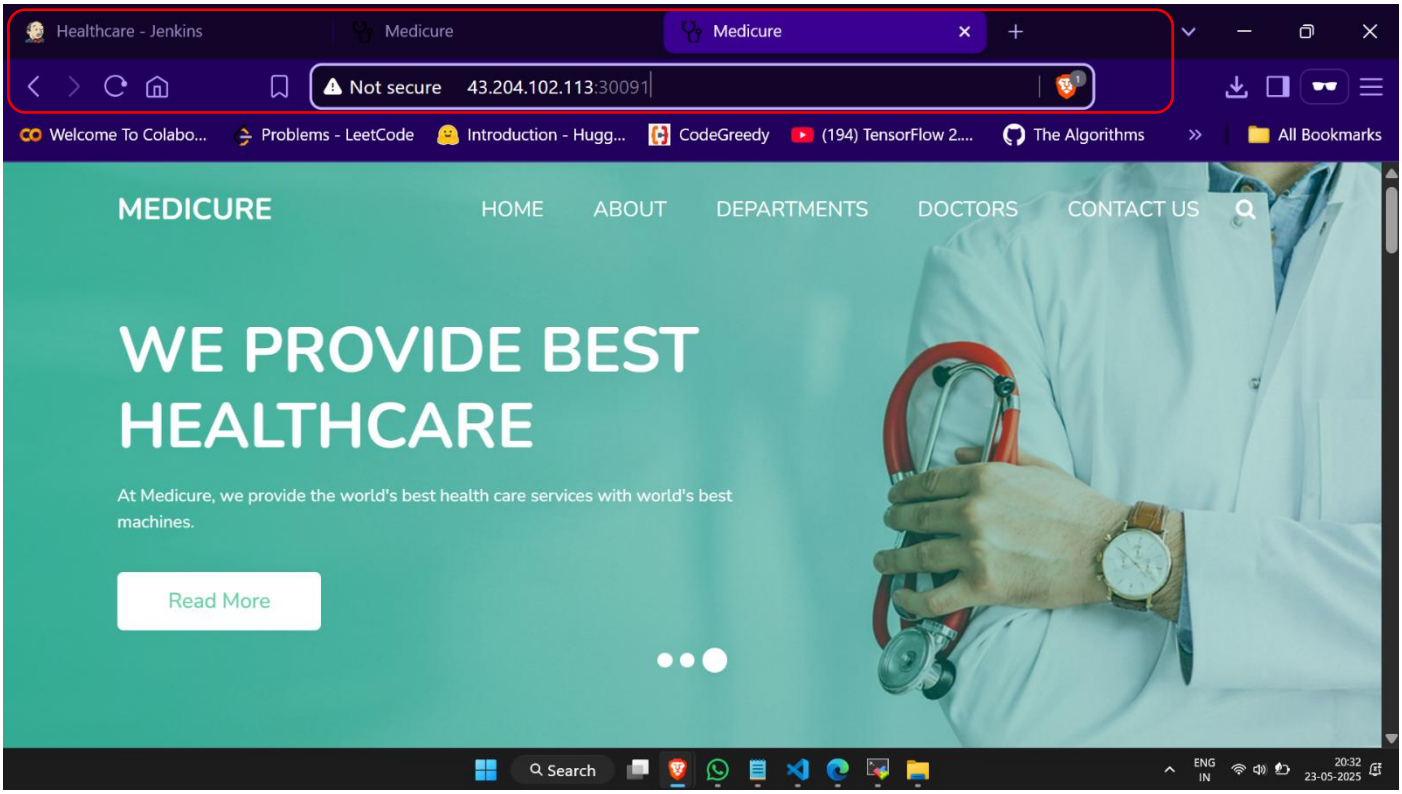




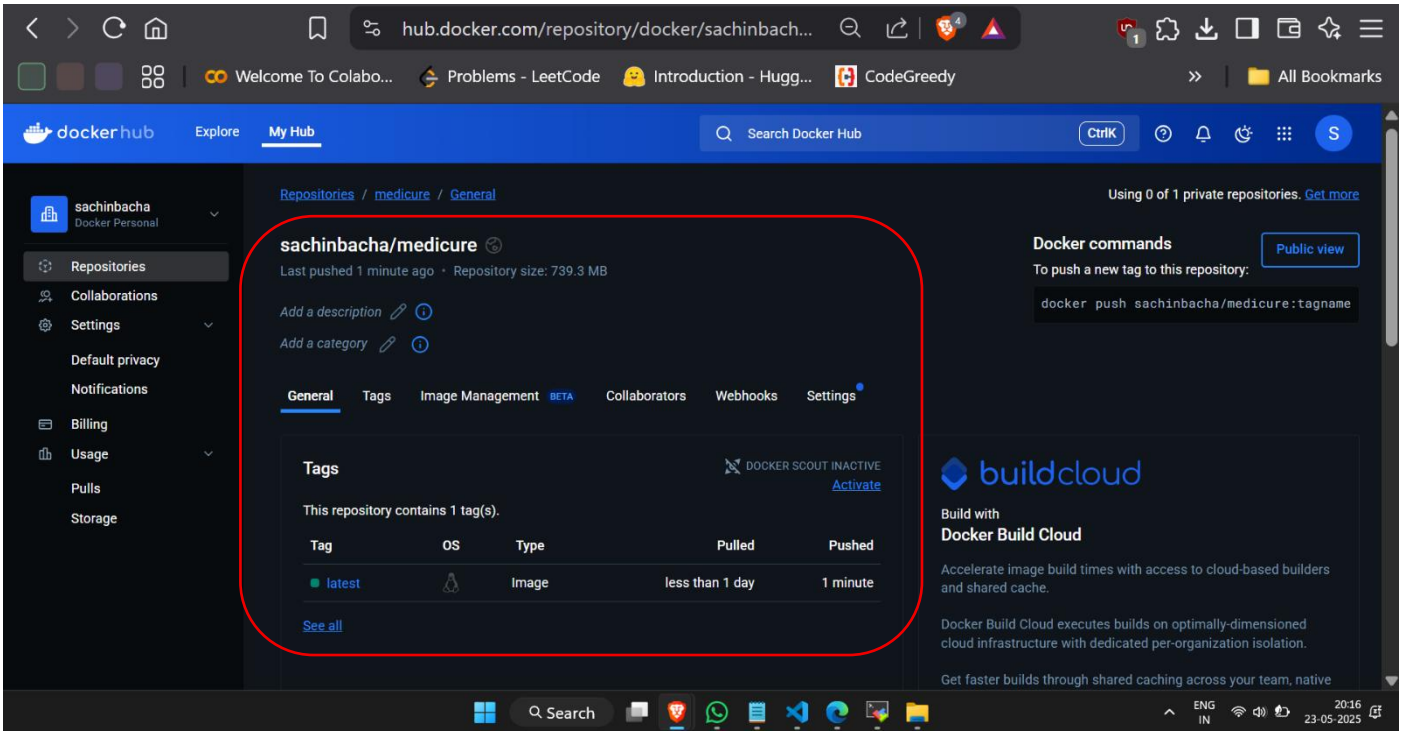
After that access the application on both the production servers using the public ip address on port number 30091



Access the application on the other server also



Check the docker hub if the image got created correctly

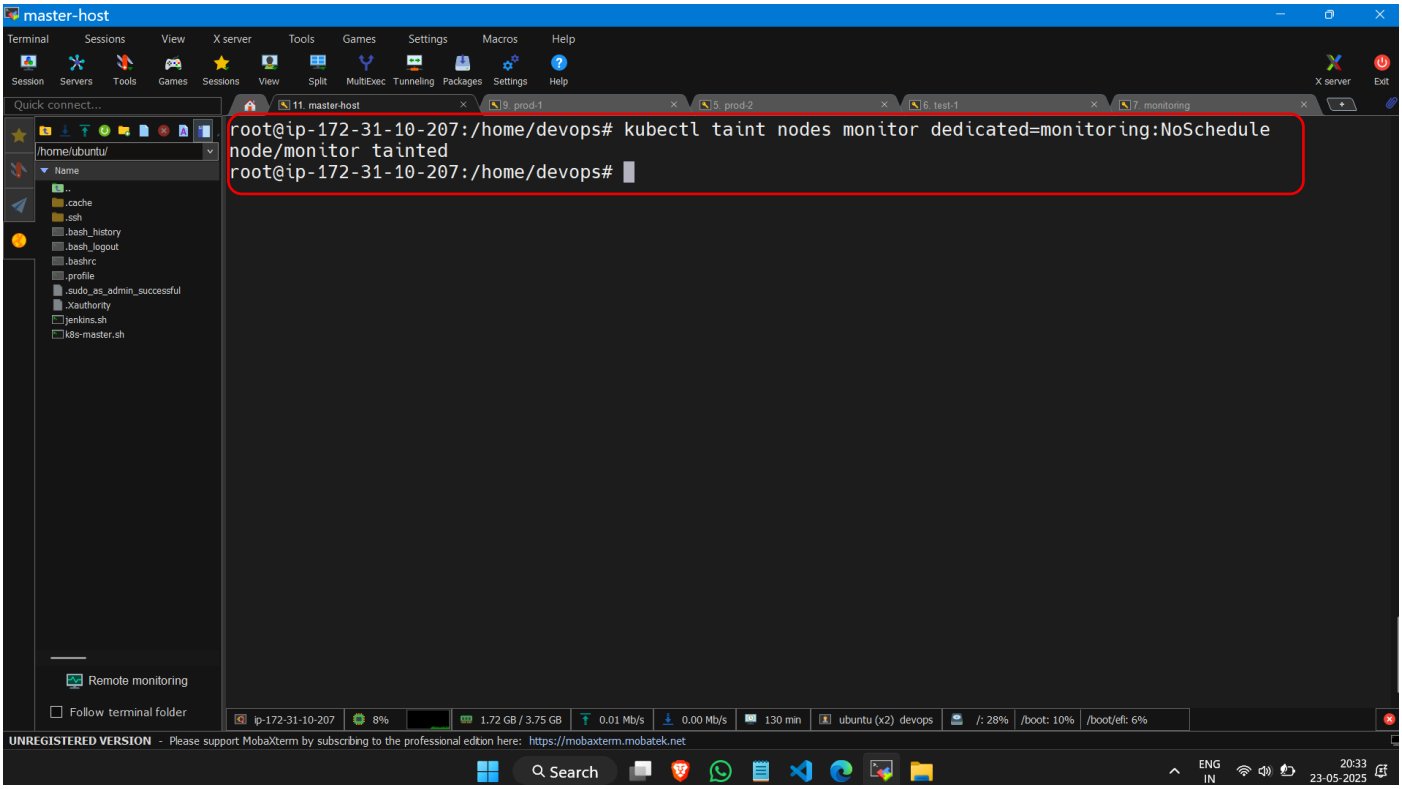


Prometheus and Grafana setup

Now in master machine

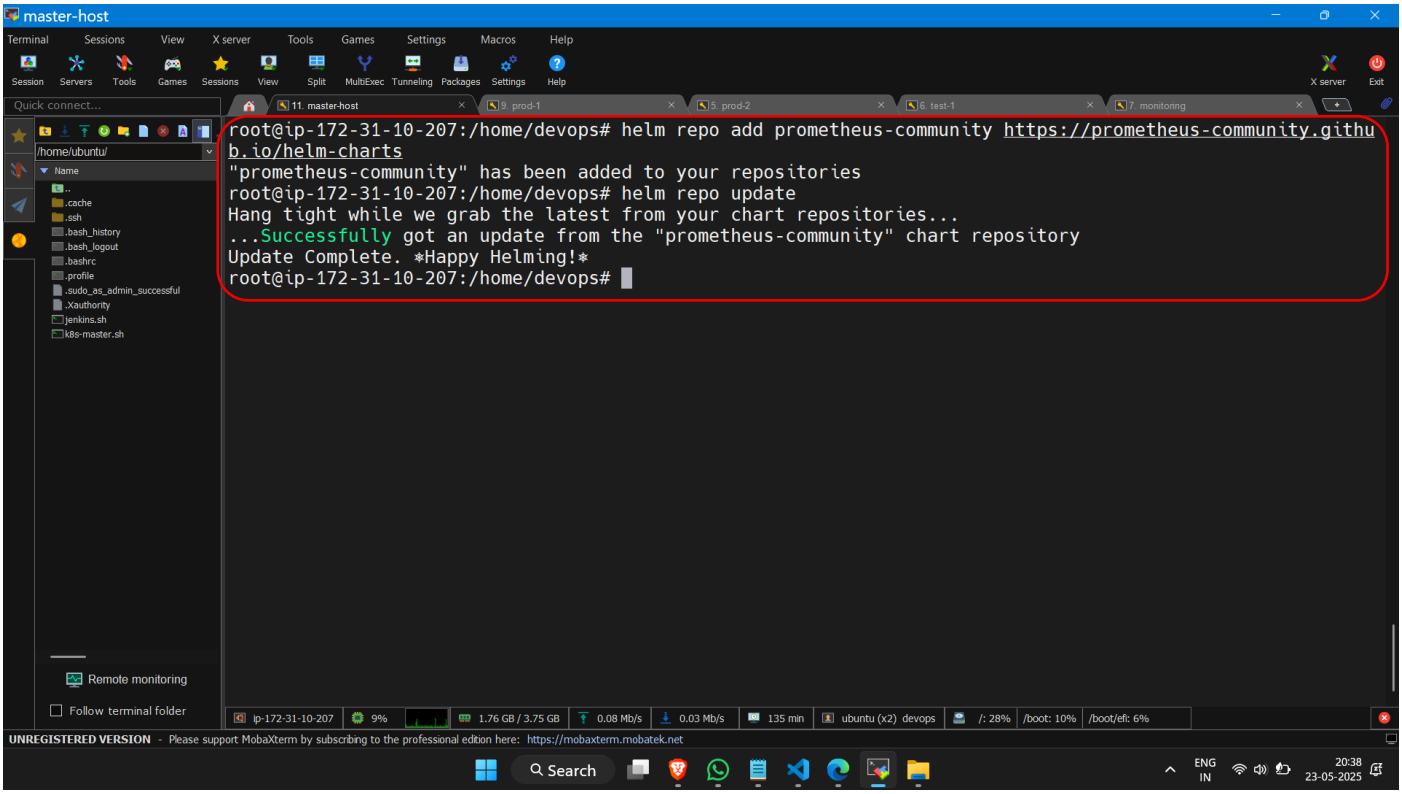
Taint the node monitoring which will separate the node from other nodes will be logically separated and no scheduling is done on that node

Using command “kubectl taint nodes <node_name> dedicated=<namespace>:Noschedule

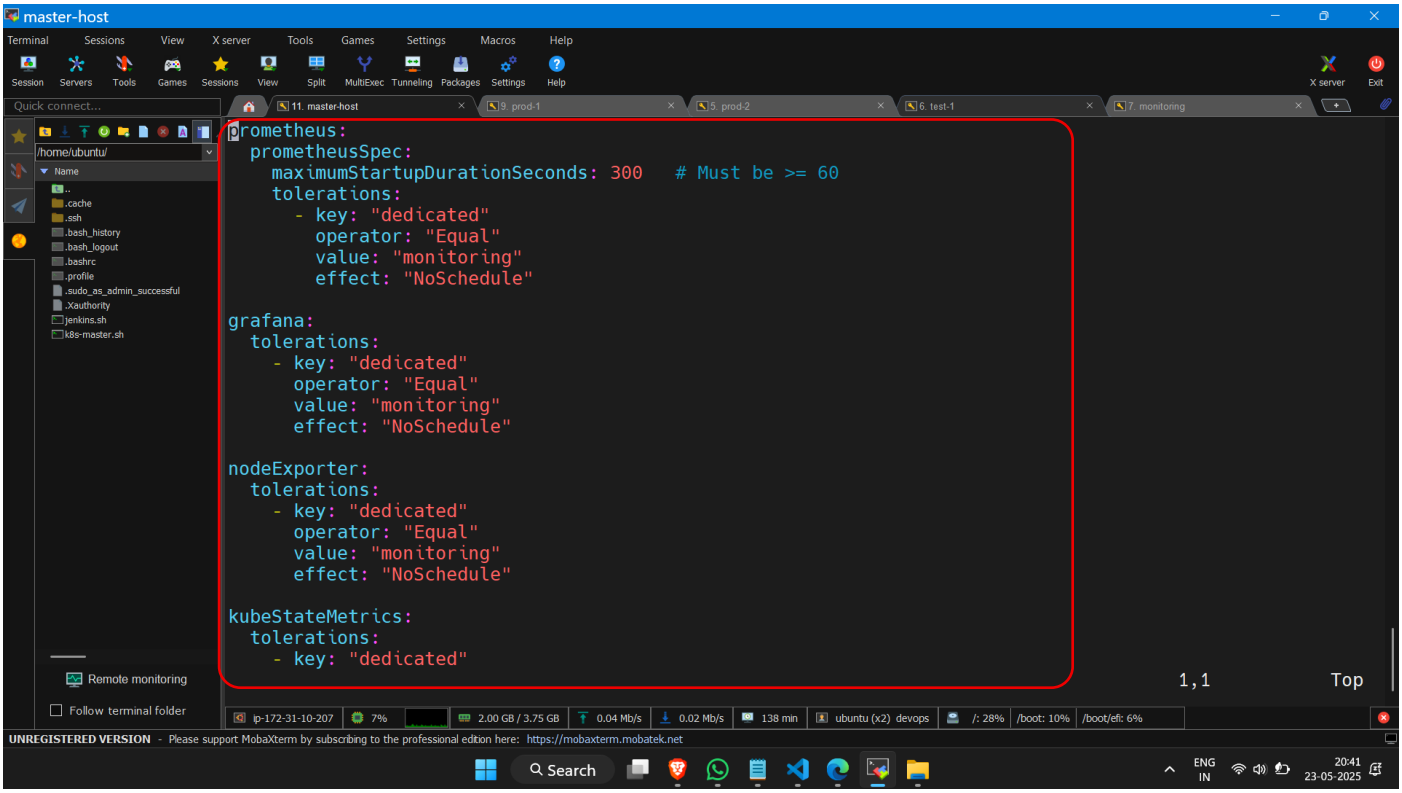


Install helm using command “curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash “

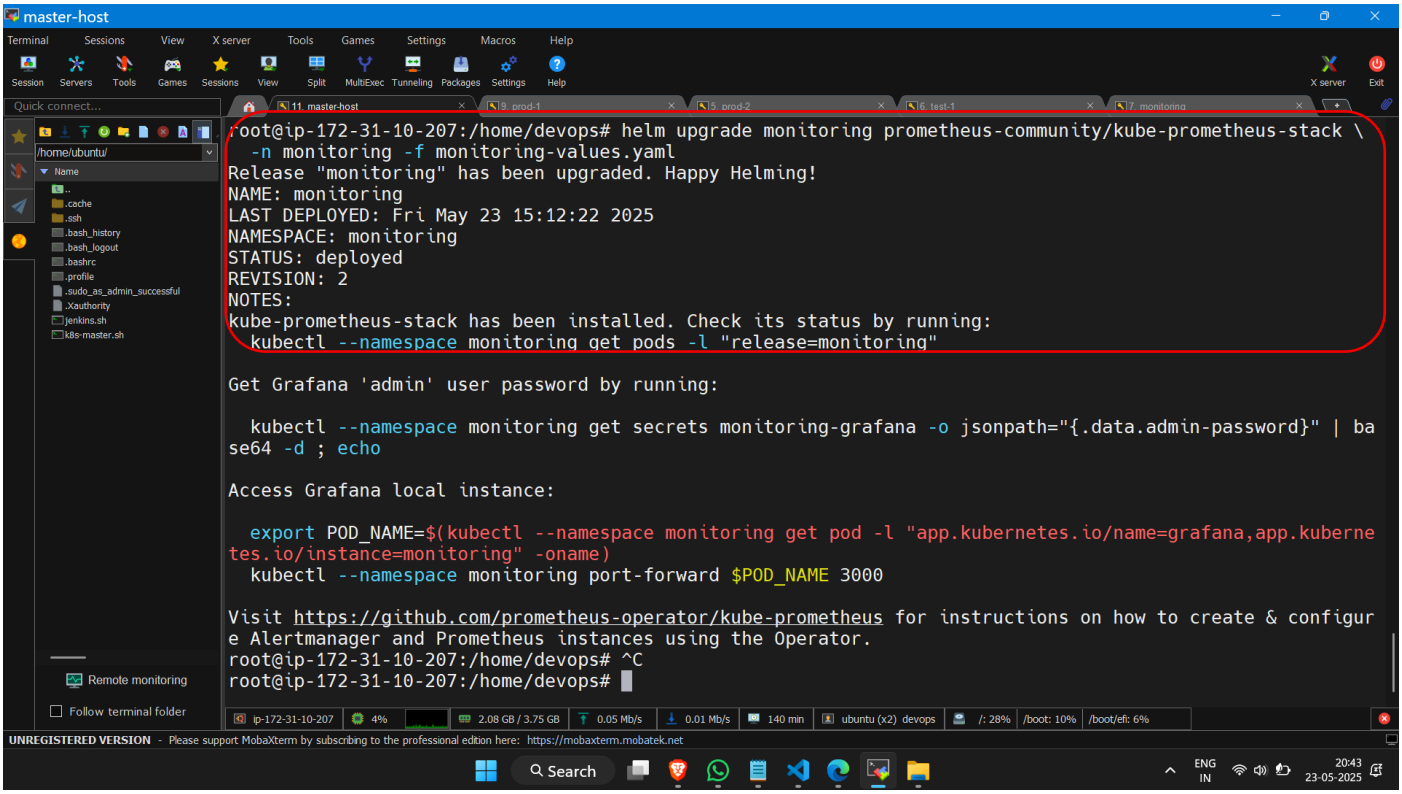
Install helm community repository and update it



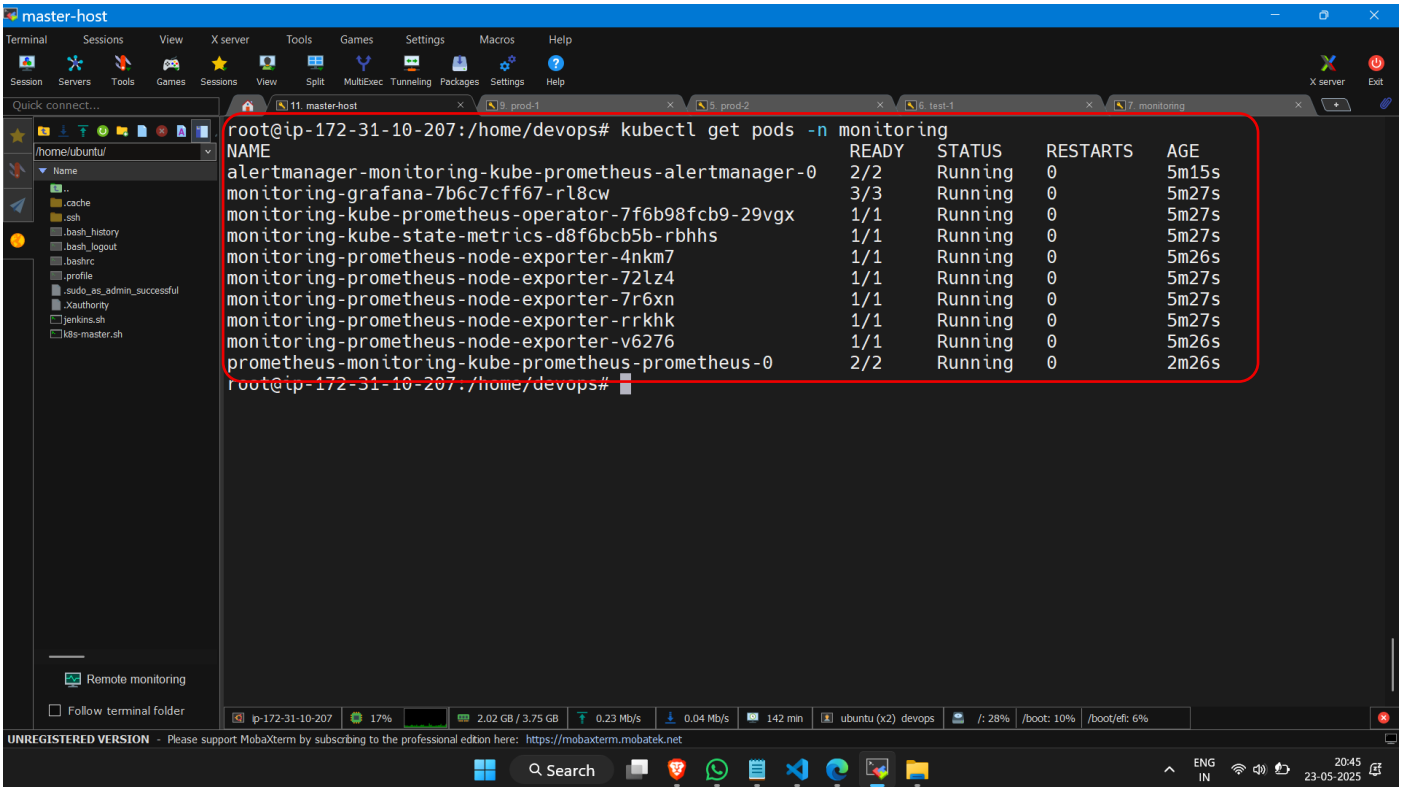
Create a monitoring-values.yml to add tolerations so pods get scheduled only on just **monitoring node**



Install Prometheus kube stack using helm



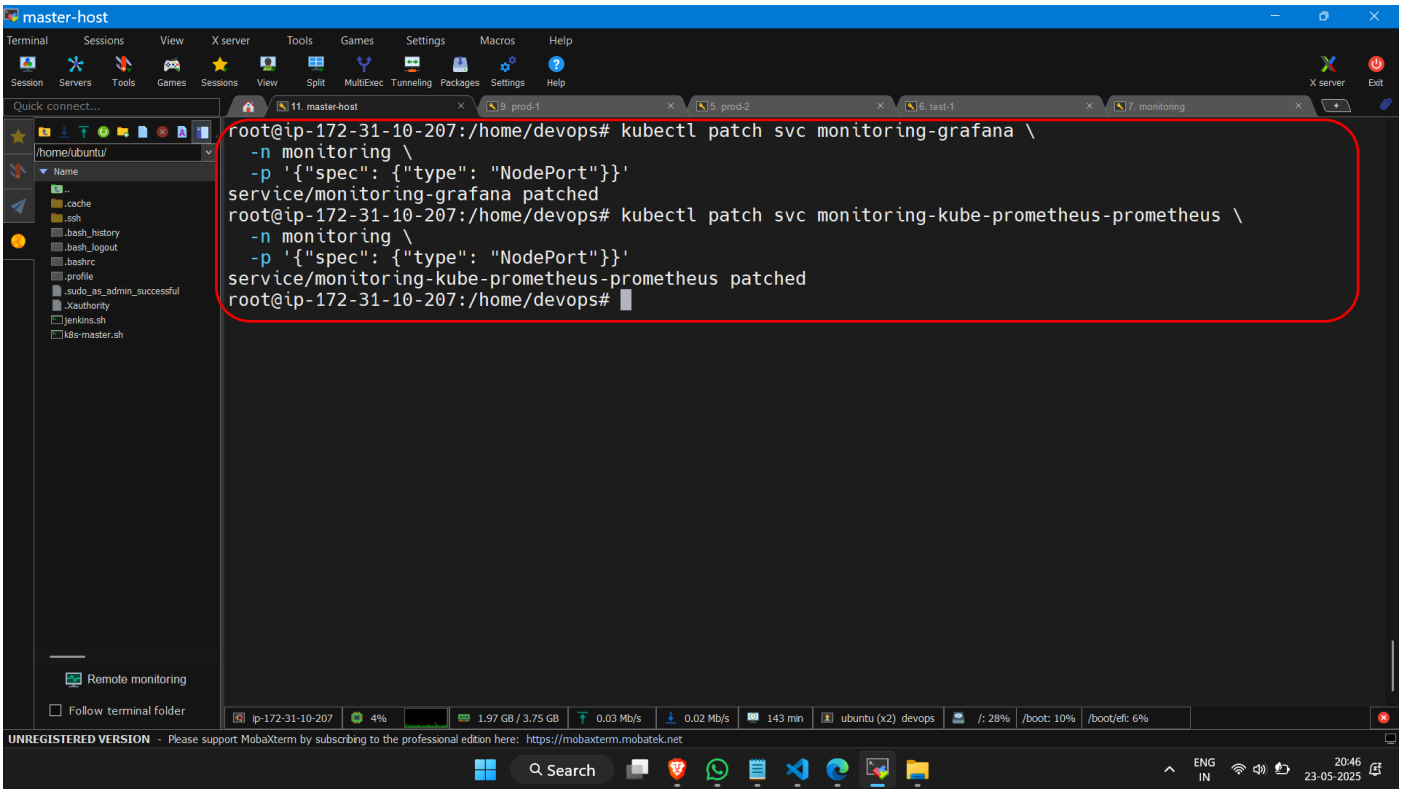
Now get the pods running in the monitoring namespace i.e the pods will now run on just monitoring node and verify if the nodes are running



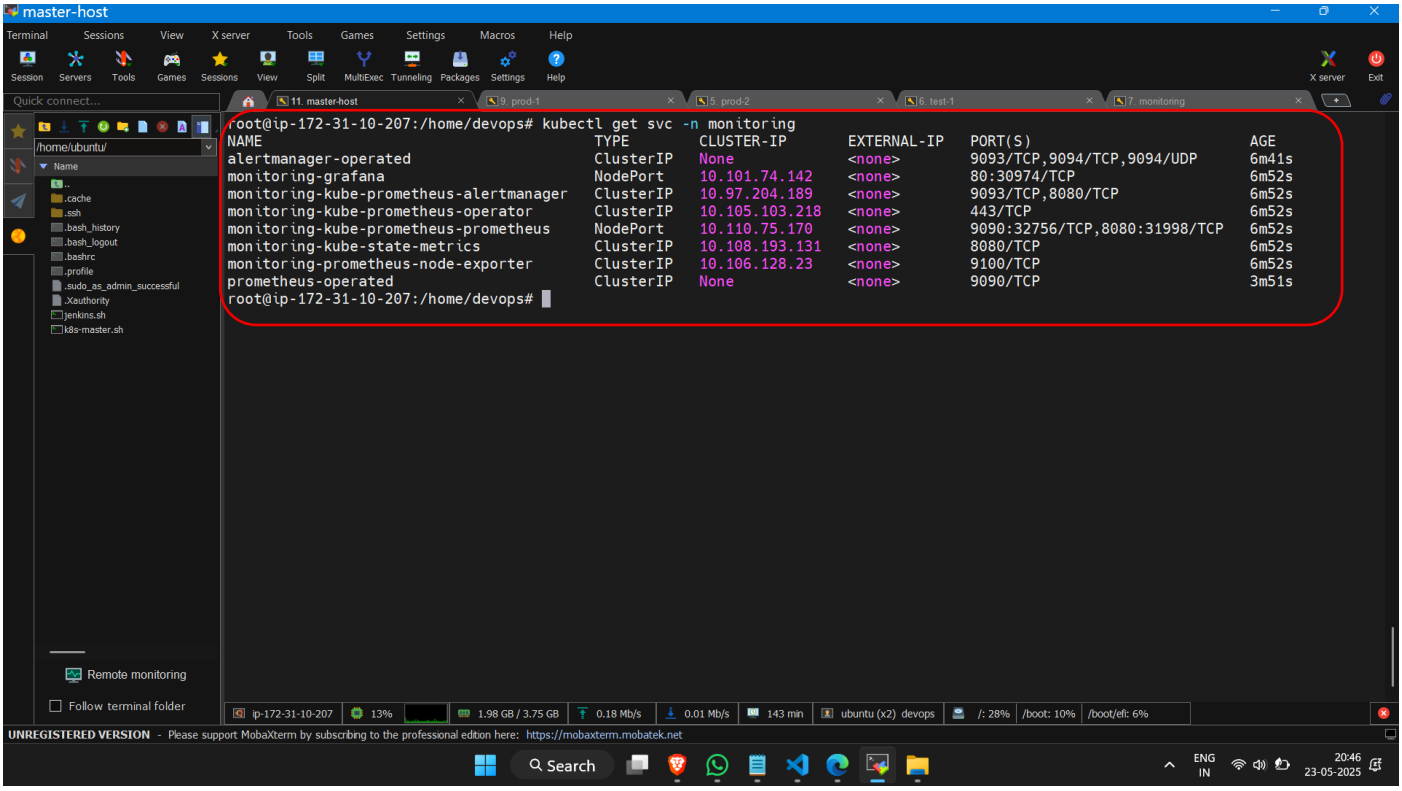
Expose Grafana to nodeport to access it through the internet (remote access)

run [“kubectl path svc monitoring-grafana\ -n monitoring \ -p ‘{“spec”:{“type”:”Nodeport”}}’ service /monitoring-grafana patched “]

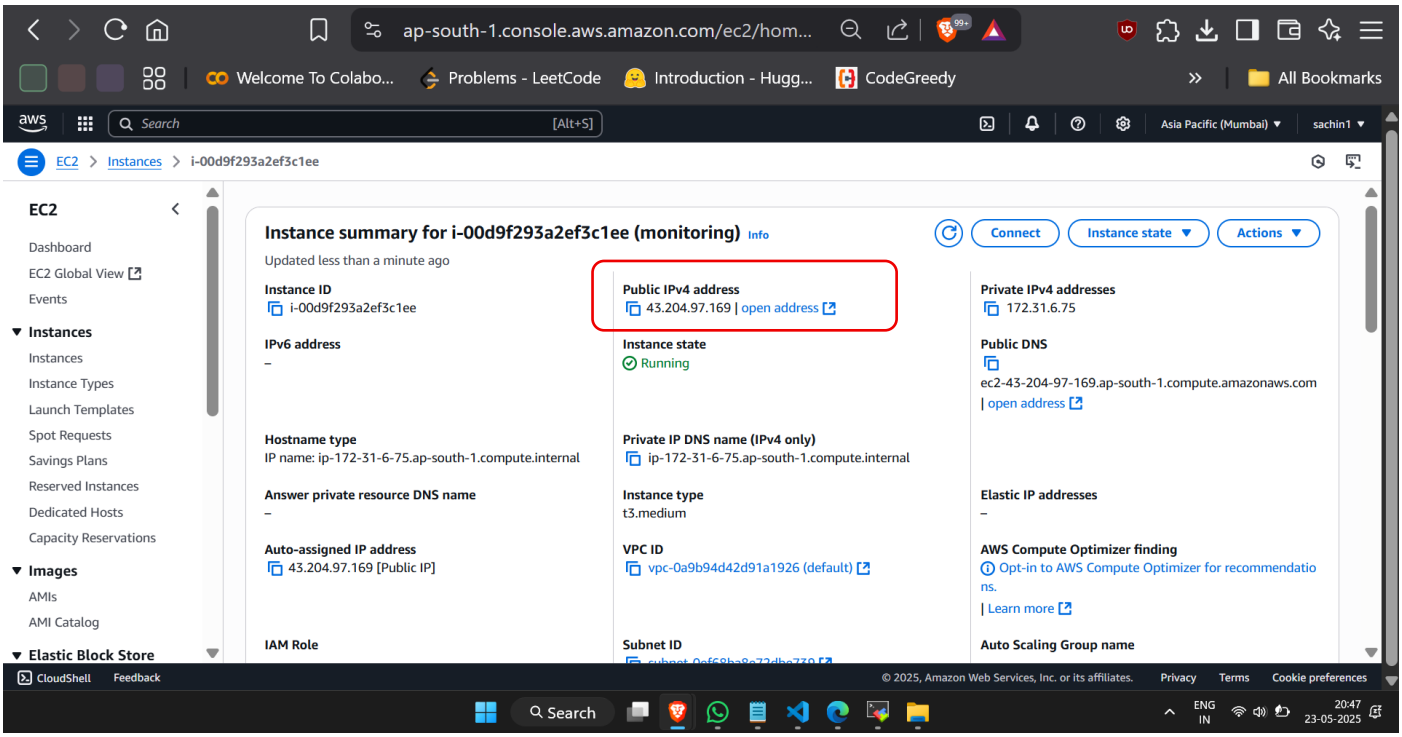
&



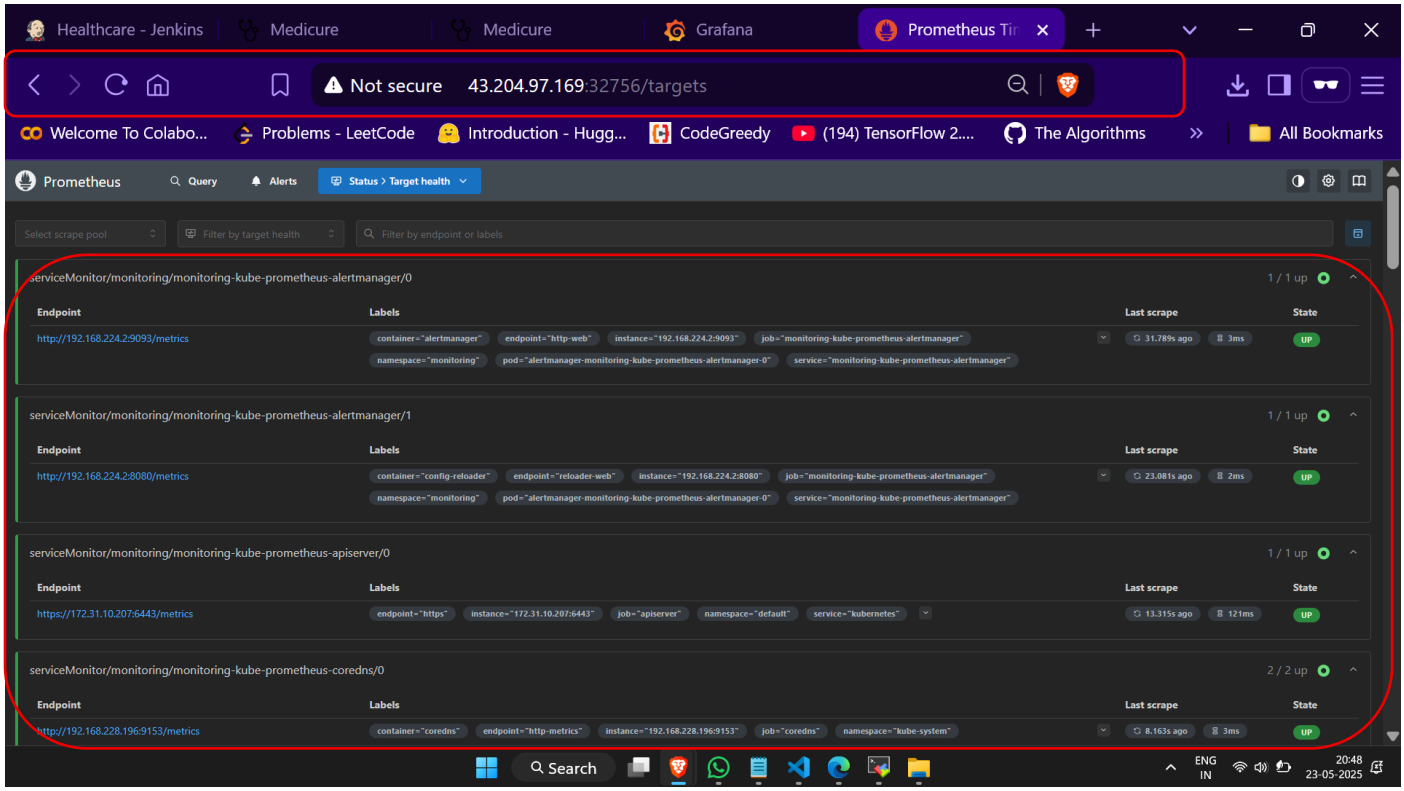
Now get the service list using “kubectl get svc -n monitoring” to get the cluster ip and the ports on which they are running.



Get the monitoring public ip address

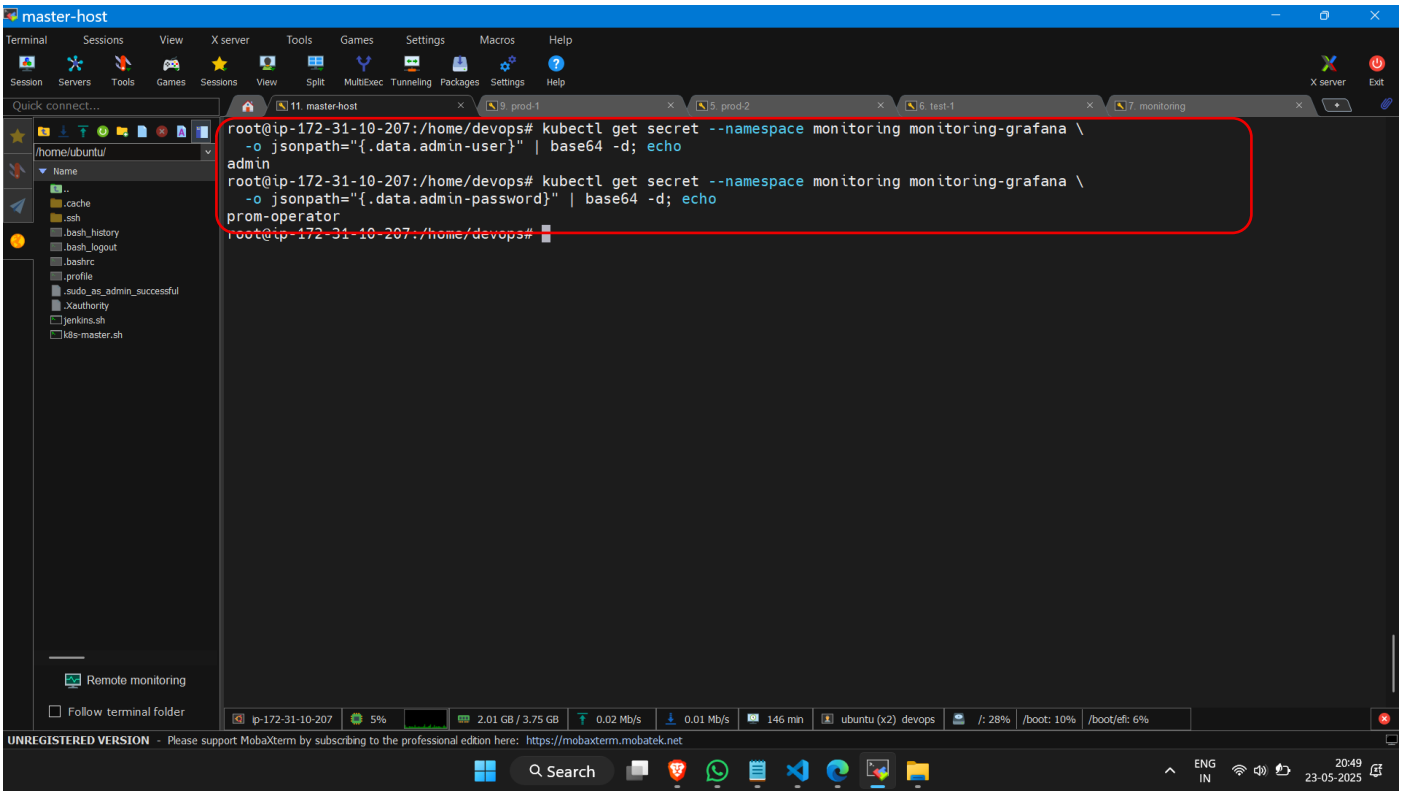


Now access the Prometheus using the monitoring ip address on the exposed port number i.e 32756

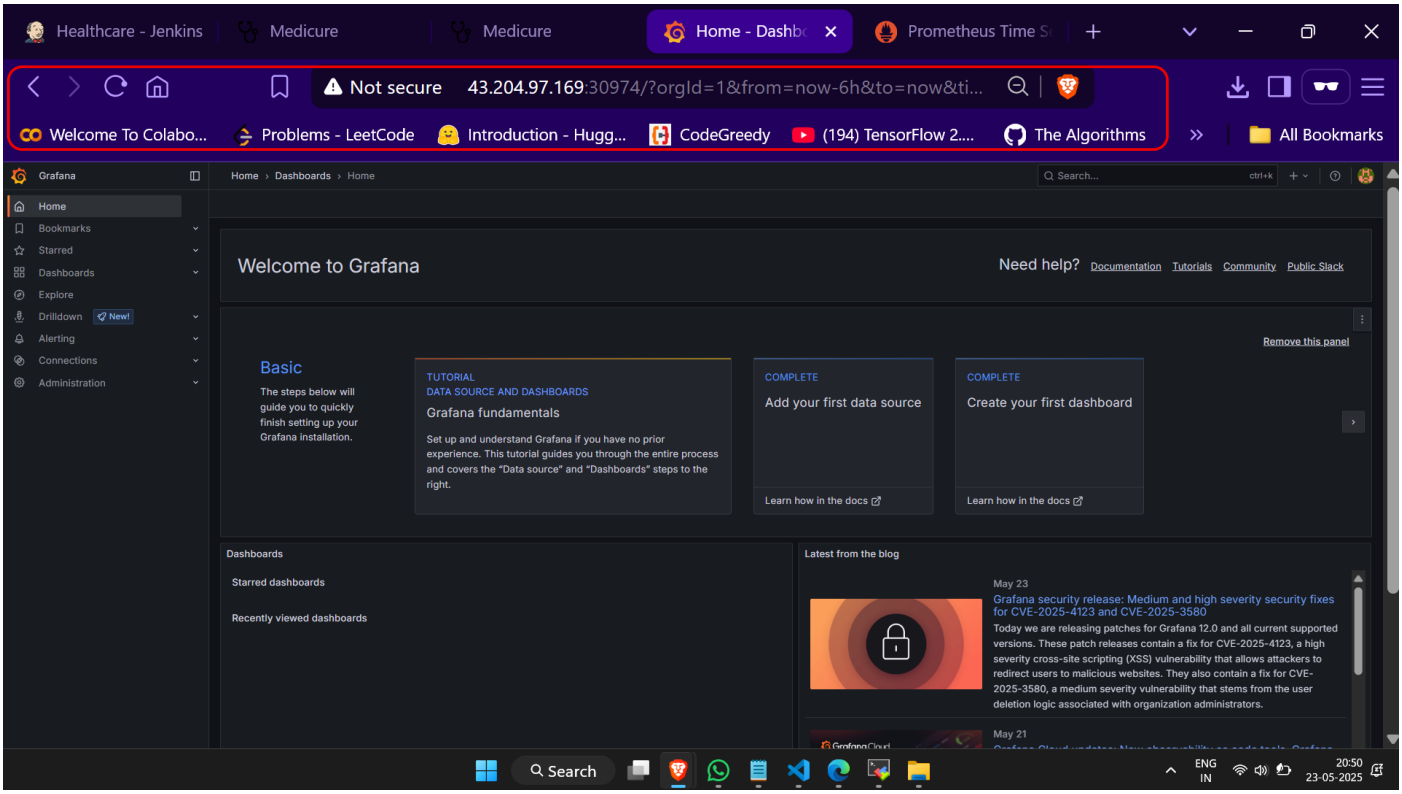


Get the password of the Grafana using the command “kubectl get secret -n monitoring monitoring-grafana \

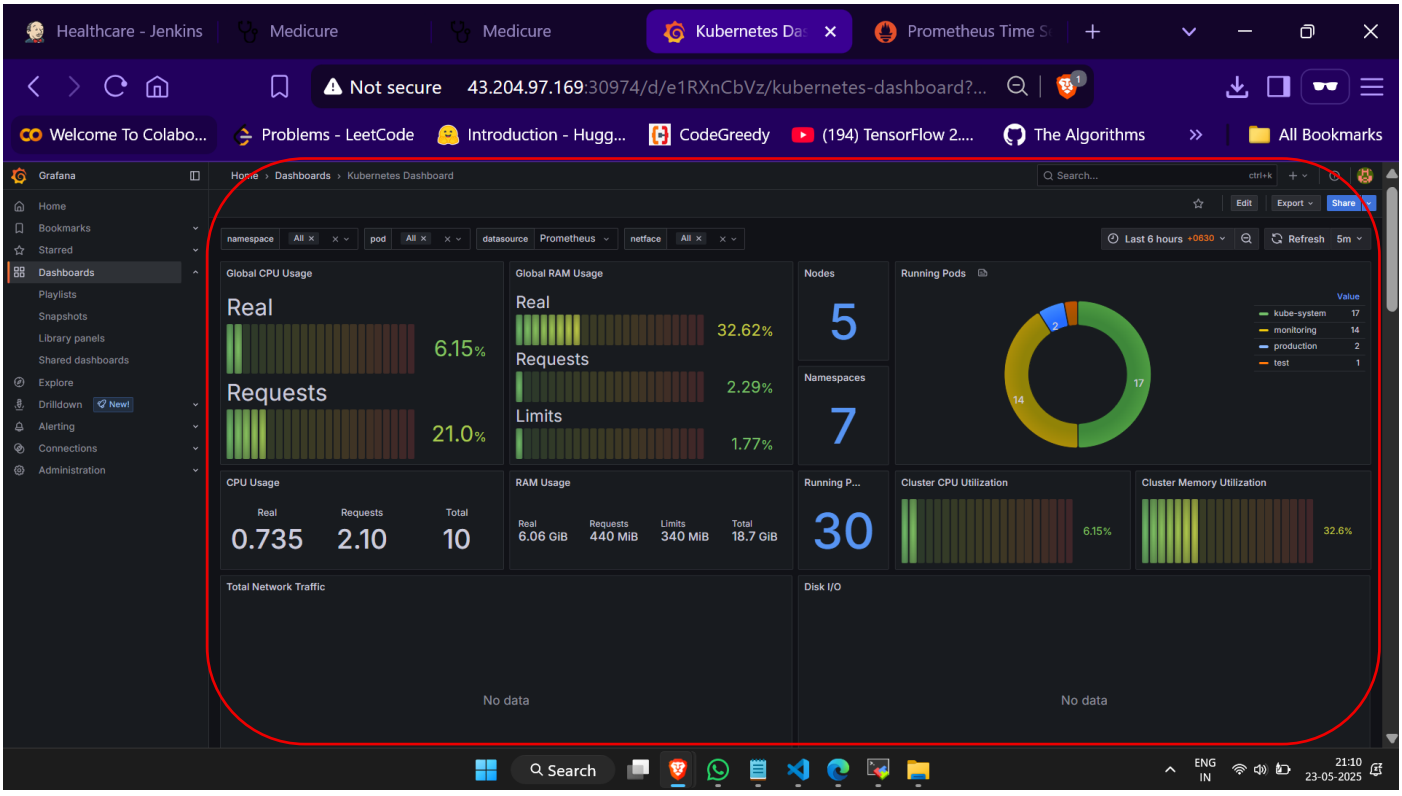
-o jsonpath="{.data.admin-password}" | base64 -d && echo”



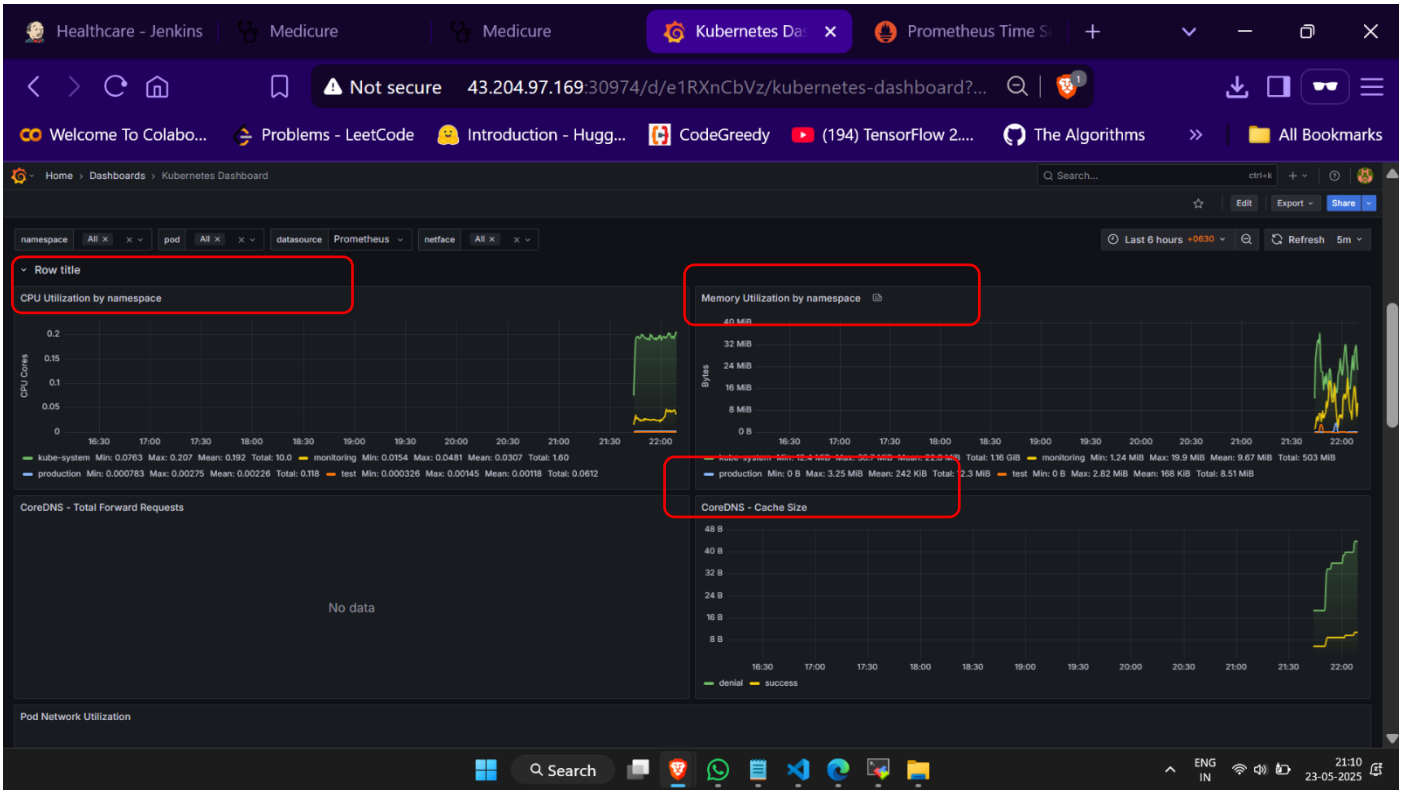
Now using the credentials login into the Grafana on port 30974



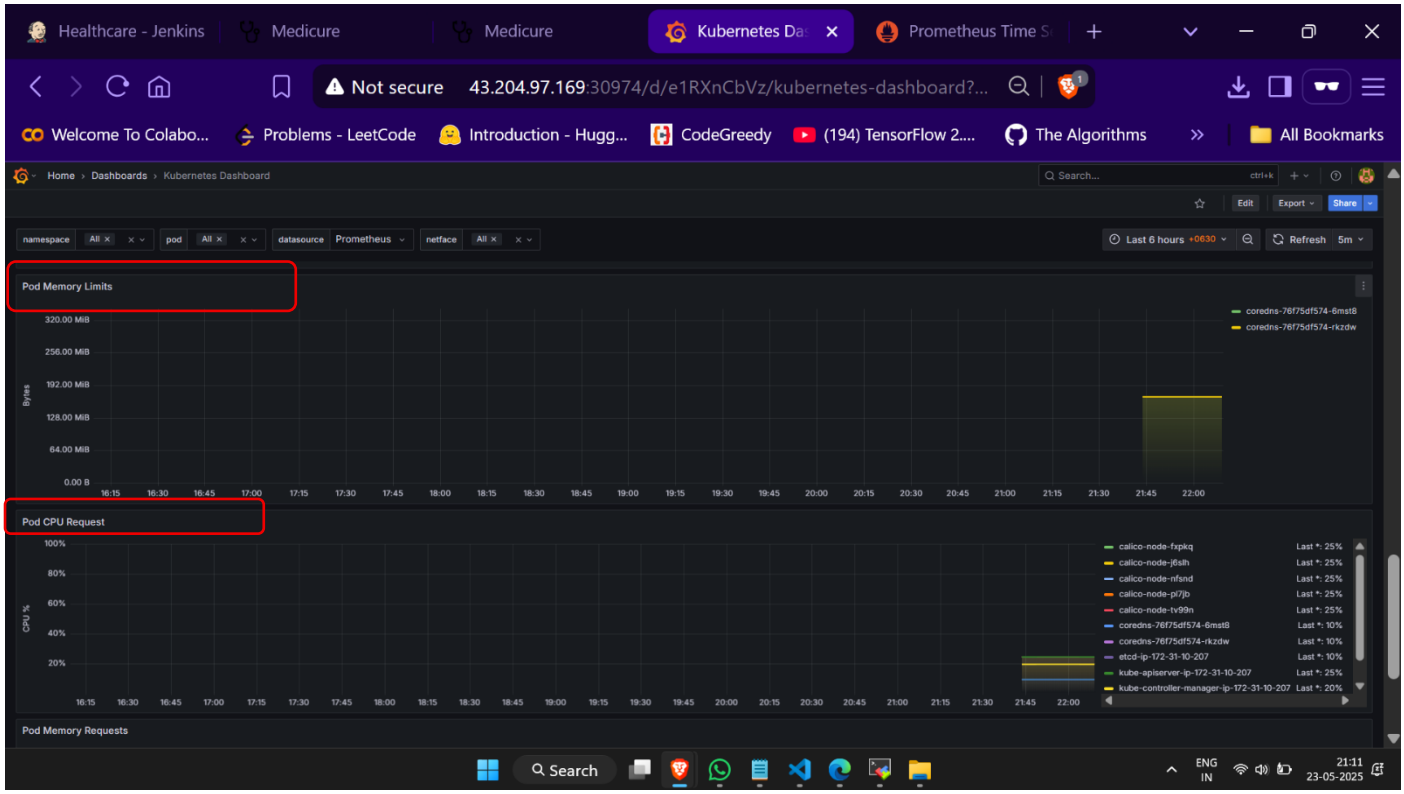
Now create the dashboard using the community supported charts which are pre built but works on our data to moitor all the pods and the production machine nodes using the above filter. we can monitor different metrics by namespace , pods, and networking



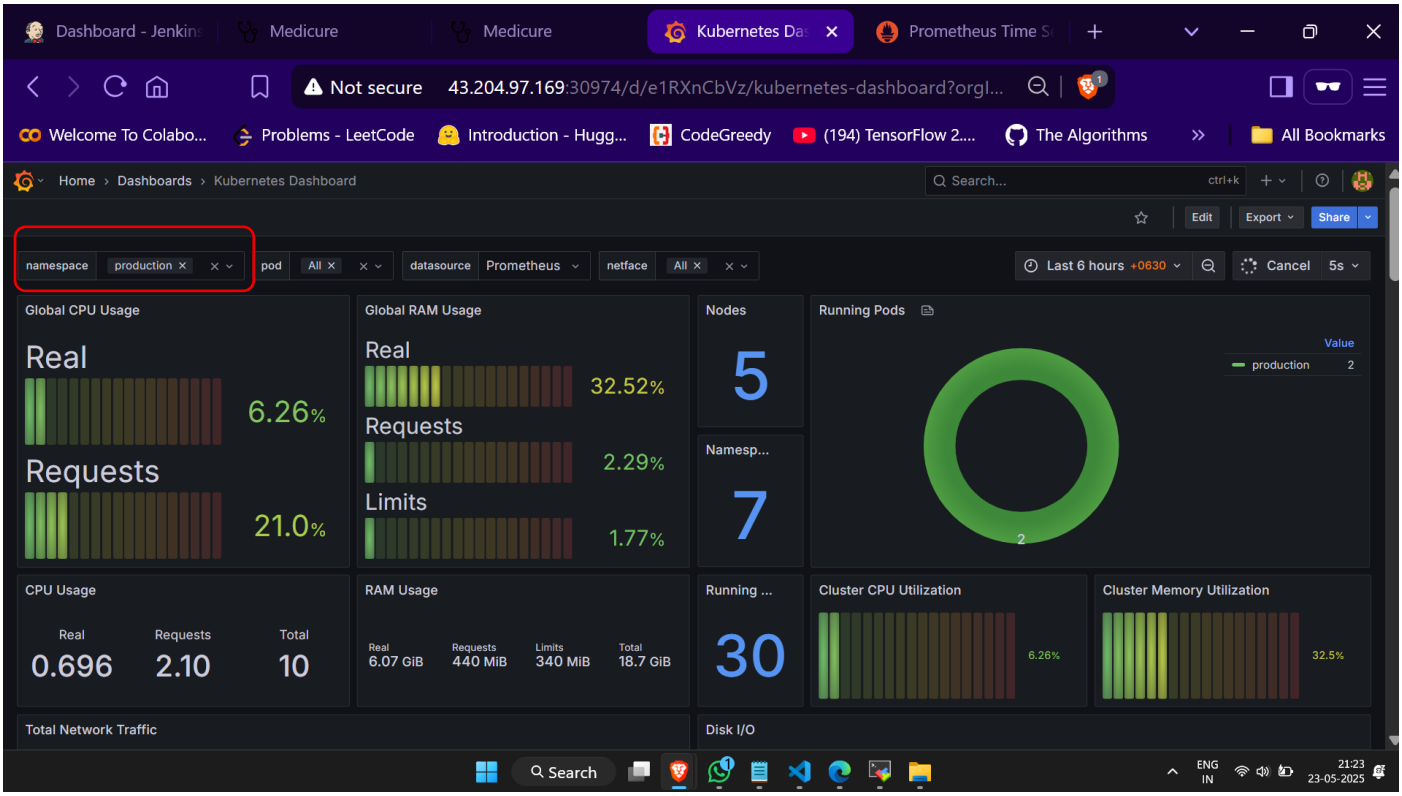
CPU utilization of the nodes i,e by namespace



Memory utilization of nodes



Metrics with namespace filter of production



Now verify the pods running by namespace