Module Name : Capstone Project

Project name : Banking and Finance 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 code should be checked out, compiled, tested, packaged and containerized. A new test-server should be provisioned using terraform and should be automatically configured using Ansible with all the required software's and as soon as the server is available, the application must be deployed to the test-server automatically.

The deployment should then be tested using a test automation tool, and if the build is Successful, Prod server must be configured with all the software it should be pushed to the prod server.

All this should happen automatically and should be triggered from a push to the GitHub master branch.

Continuous monitoring server must be configured to monitor the test as well as prod server using Prometheus and Grafana should be configured to display a dashboard with following metrics.

1. CPU utilization
2. Disk Space Utilization
3. Total Available Memory

Tech stack Required to use:

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

✓ Maven – For Continuous Build

✓ Jenkins - For continuous integration and continuous deployment

✓ Docker - For deploying containerized applications

✓ Ansible - Configuration management tools

✓ Selenium - For automating tests on the deployed web application

✓ Terraform - For creation of infrastructure.

✓ Prometheus and Grafana – For Automated Monitoring and Report Visualization

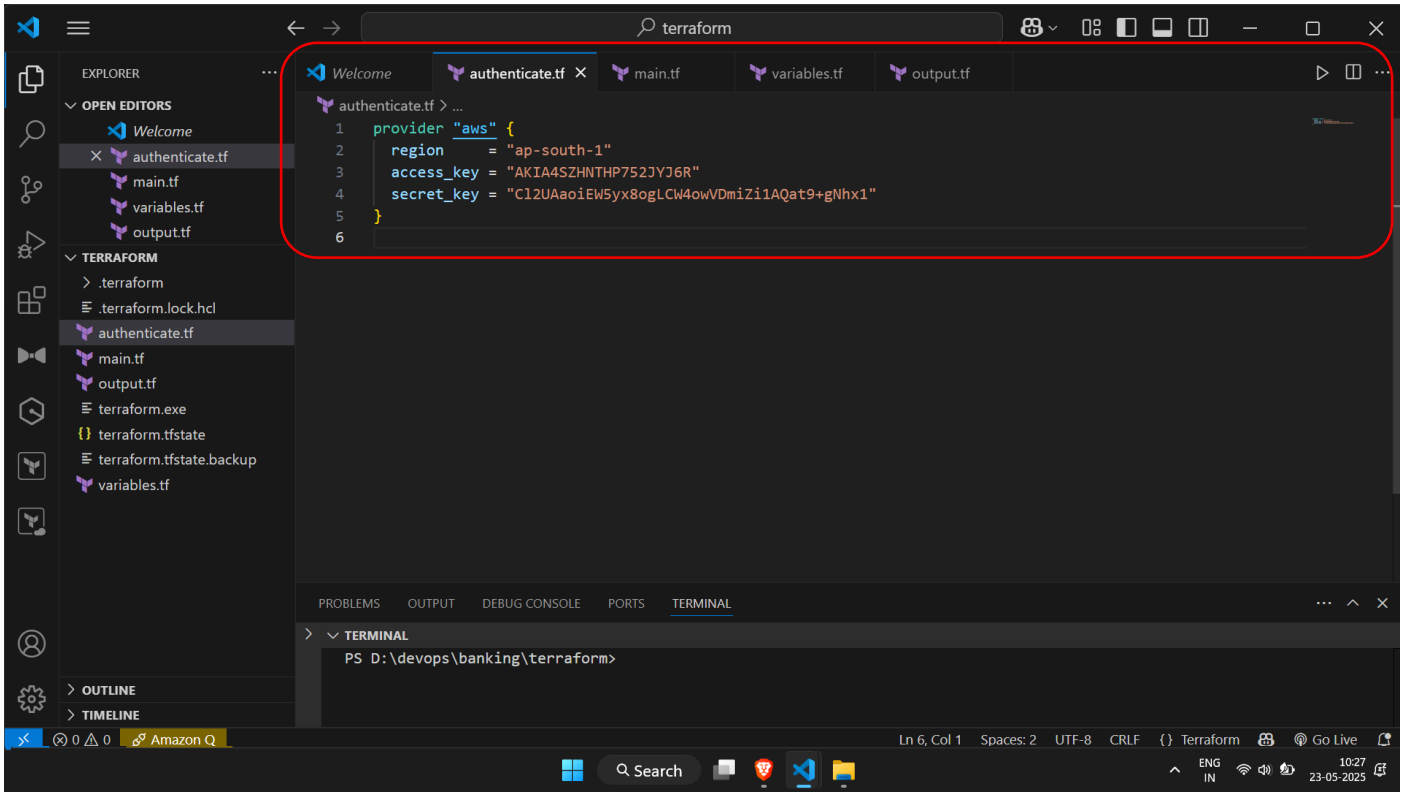
Steps for project :

1. Launch 4 Instance with terraform

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

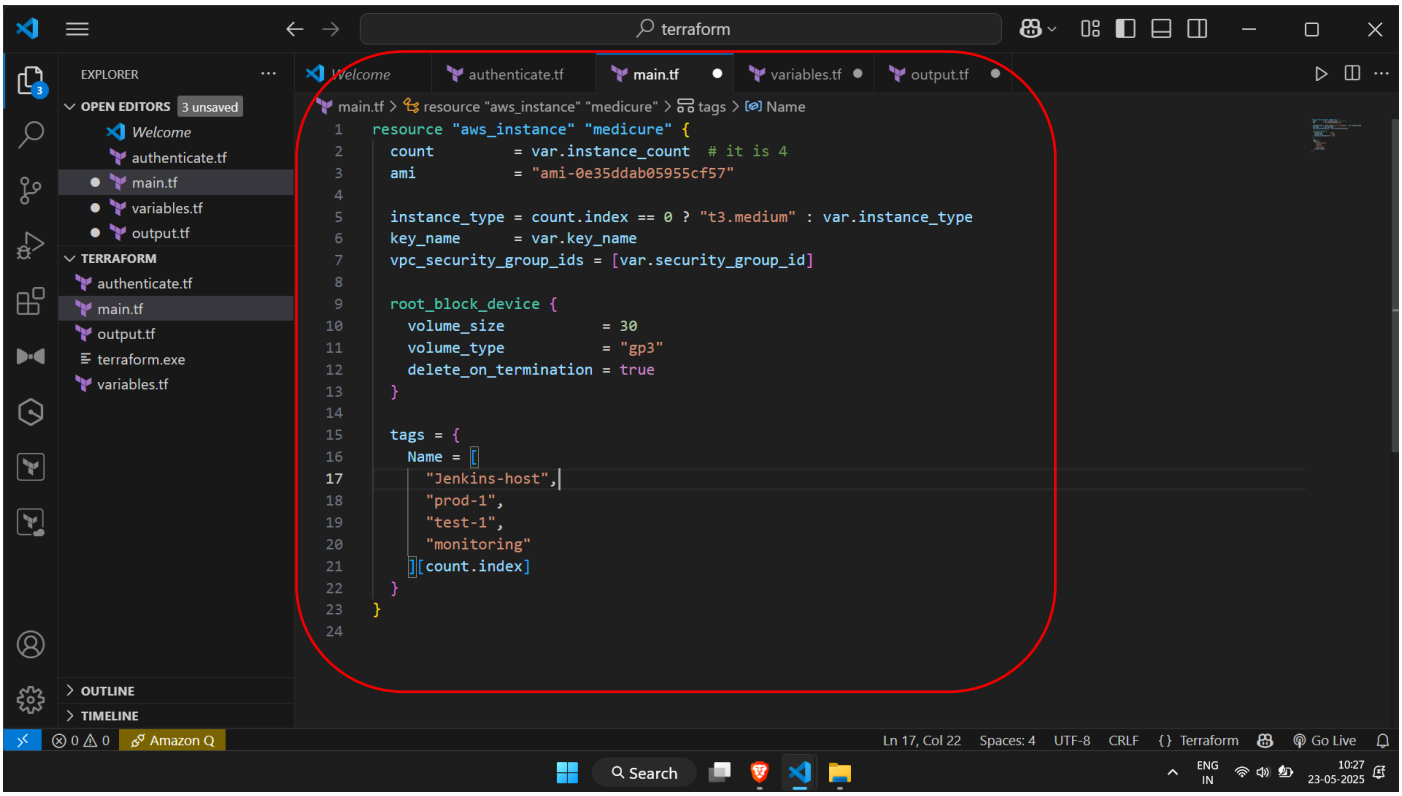
1. Create a terraform directory consist of a terraform .exe file .
2. Create Four files authenticate.tf , main.tf , output.tf , variables.tf

Authenticate..tf

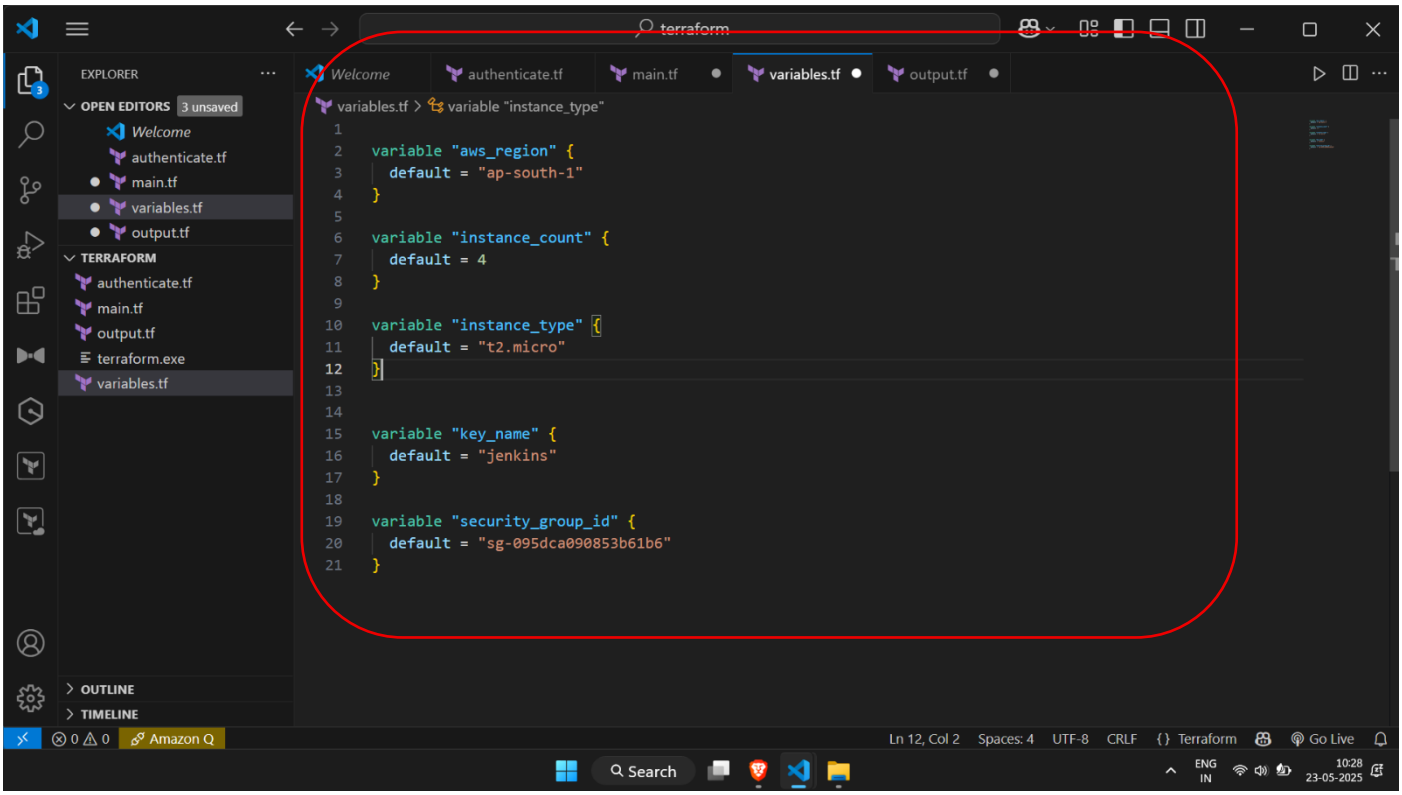


This should consist of access key and secret key which has been created in aws using IAM service by creating a new user with Full permission policies attached . Then it will provide a secret key and access key which will now be used as credentials to terraform to launch the instances.

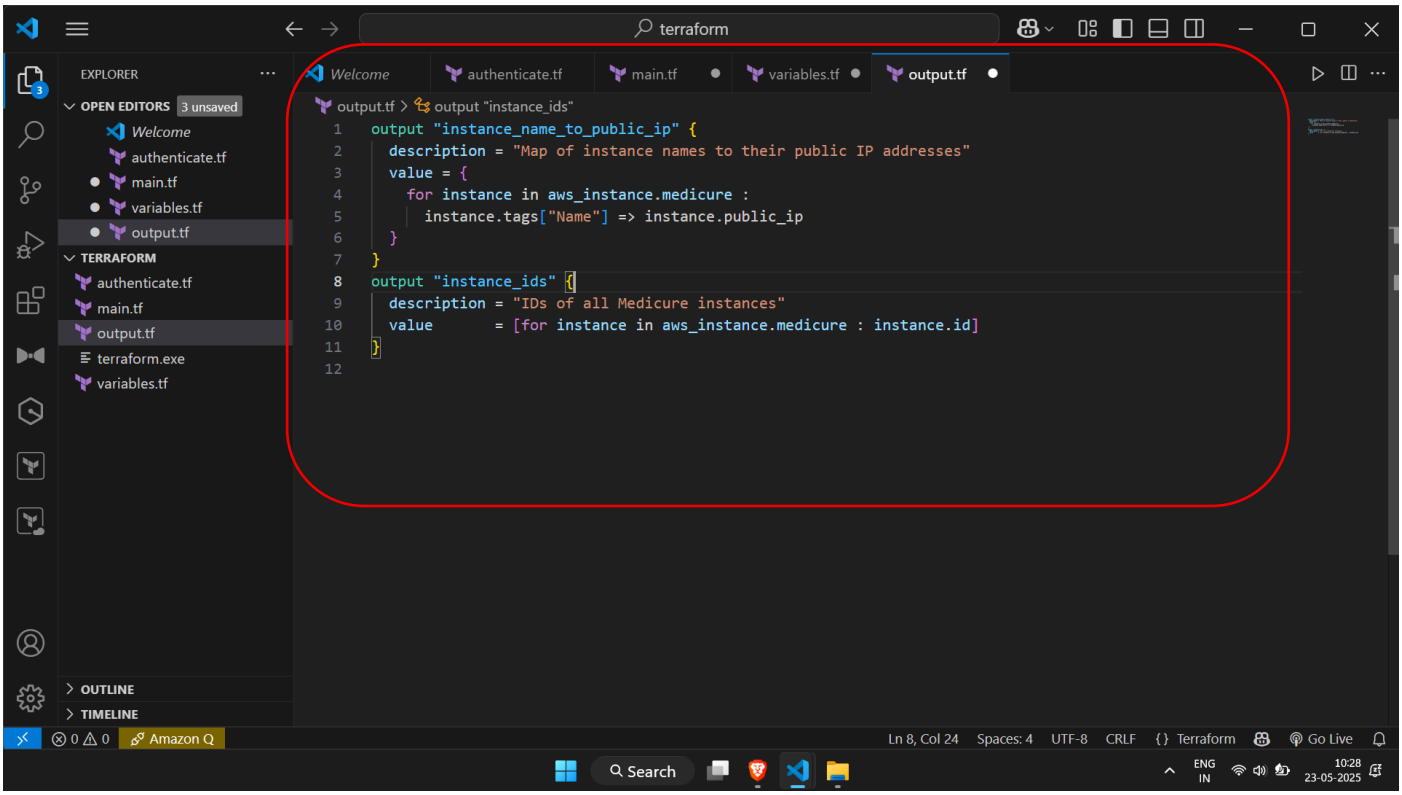
Main.tf



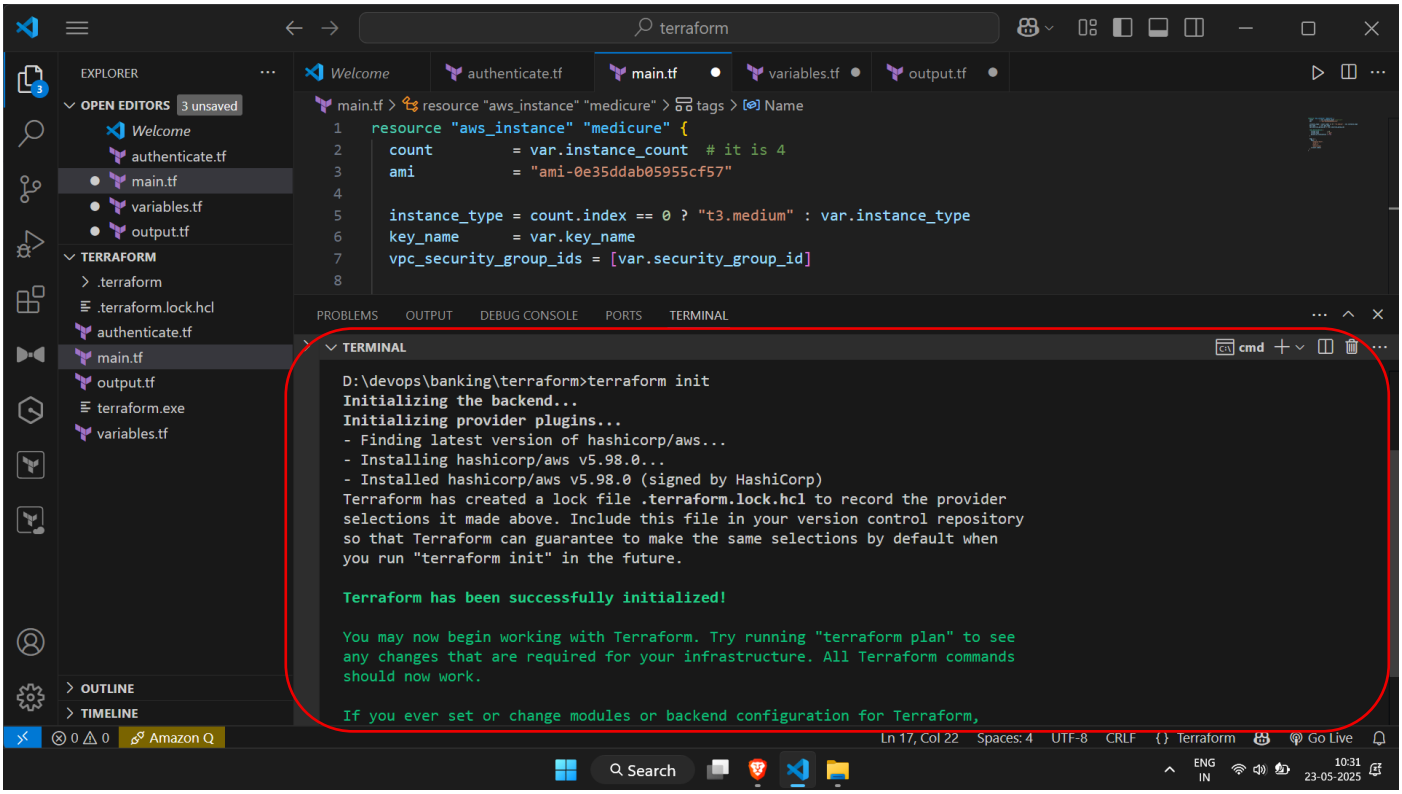
Variables.tf



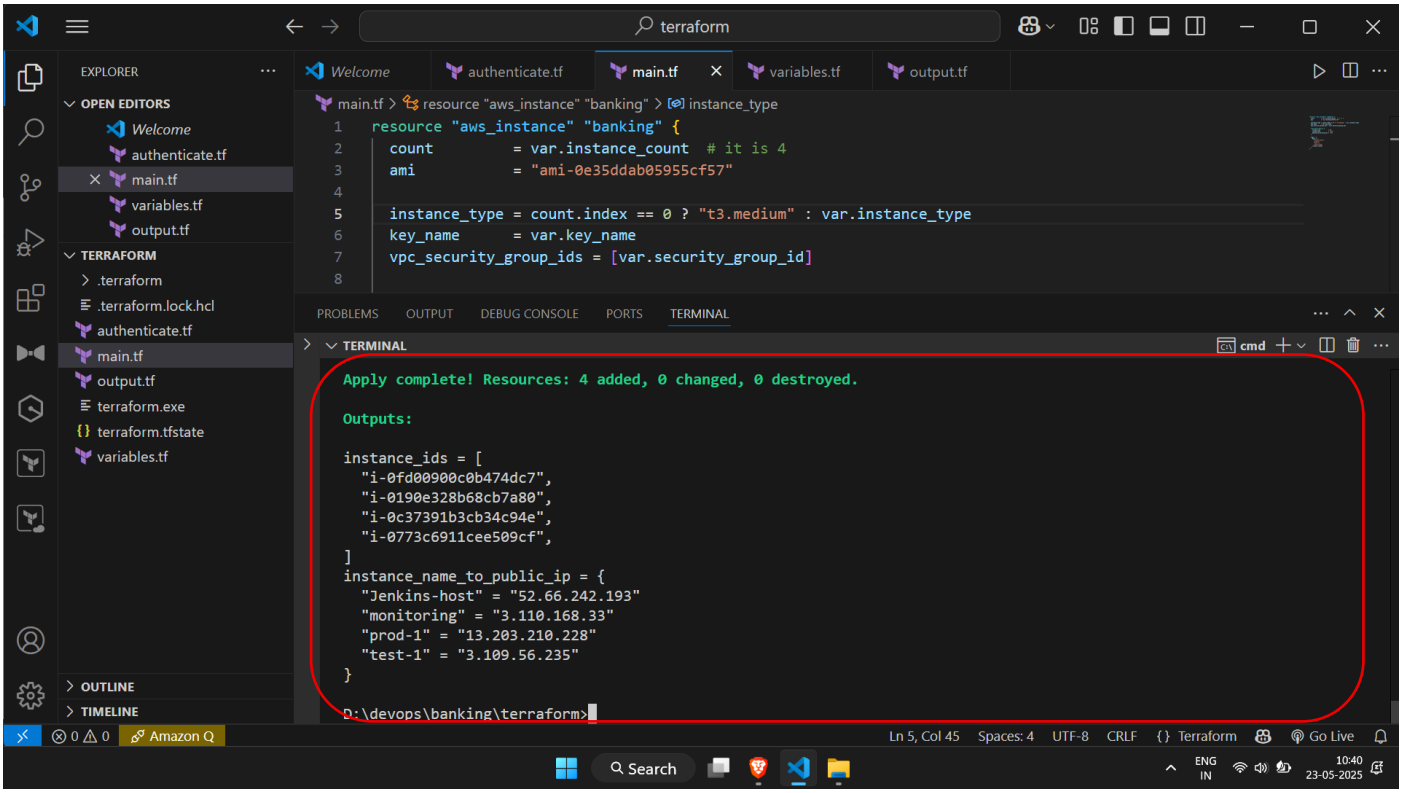
Output.tf



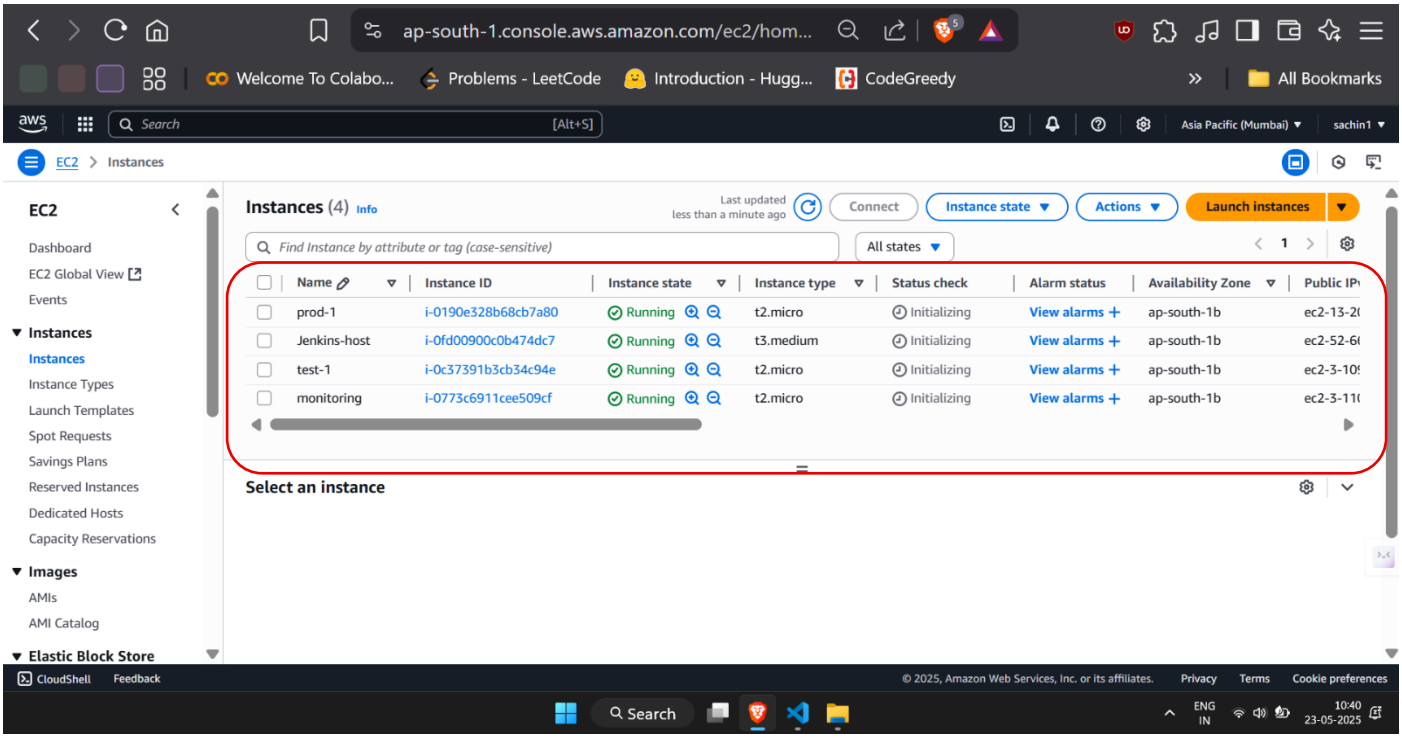
Now open a new cmd terminal and initialize the terraform in the directory using command "terraform init"



Afterward use command “terraform plan “ to get an idea of the resources to be created and if verified then use command “terraform apply “ to create the resources .

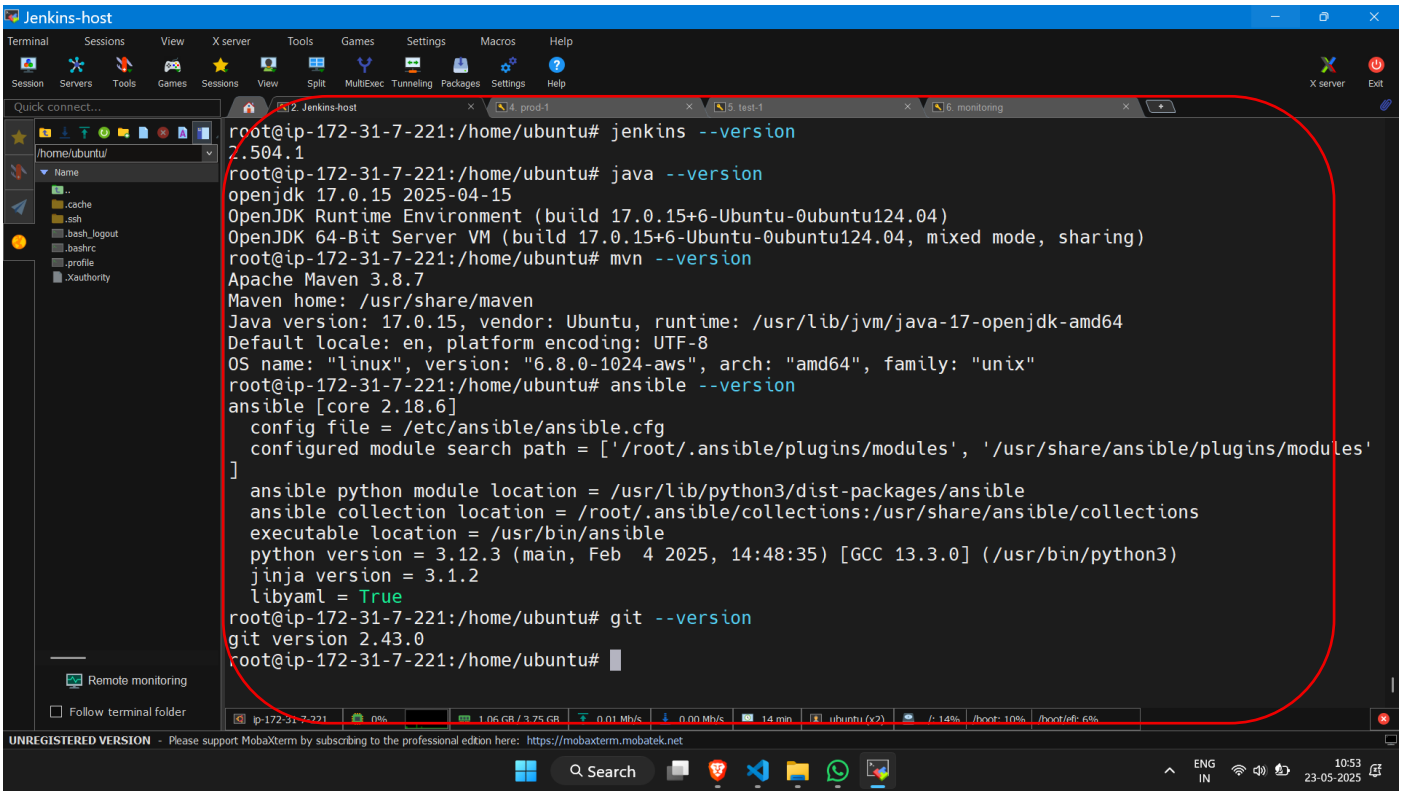


Now login to the aws account and check if the resources are created correctly



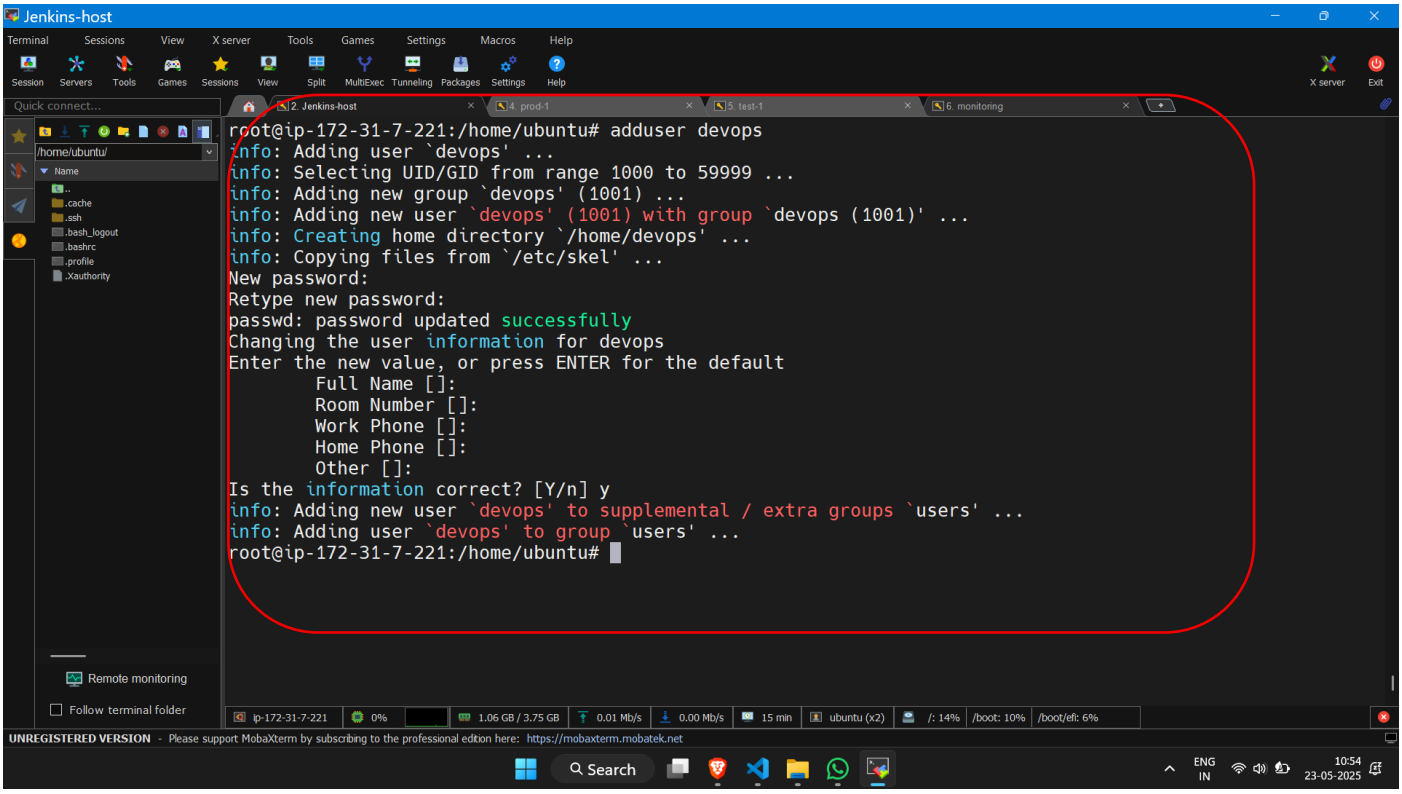
Now connect all the instances to MobaXterm using the attached keypair which is been used in terraform script also.

And install **Jenkins, Java, Maven, ansible, git and also Docker** this instances will be the Jenkins host and also the ansible controller node .

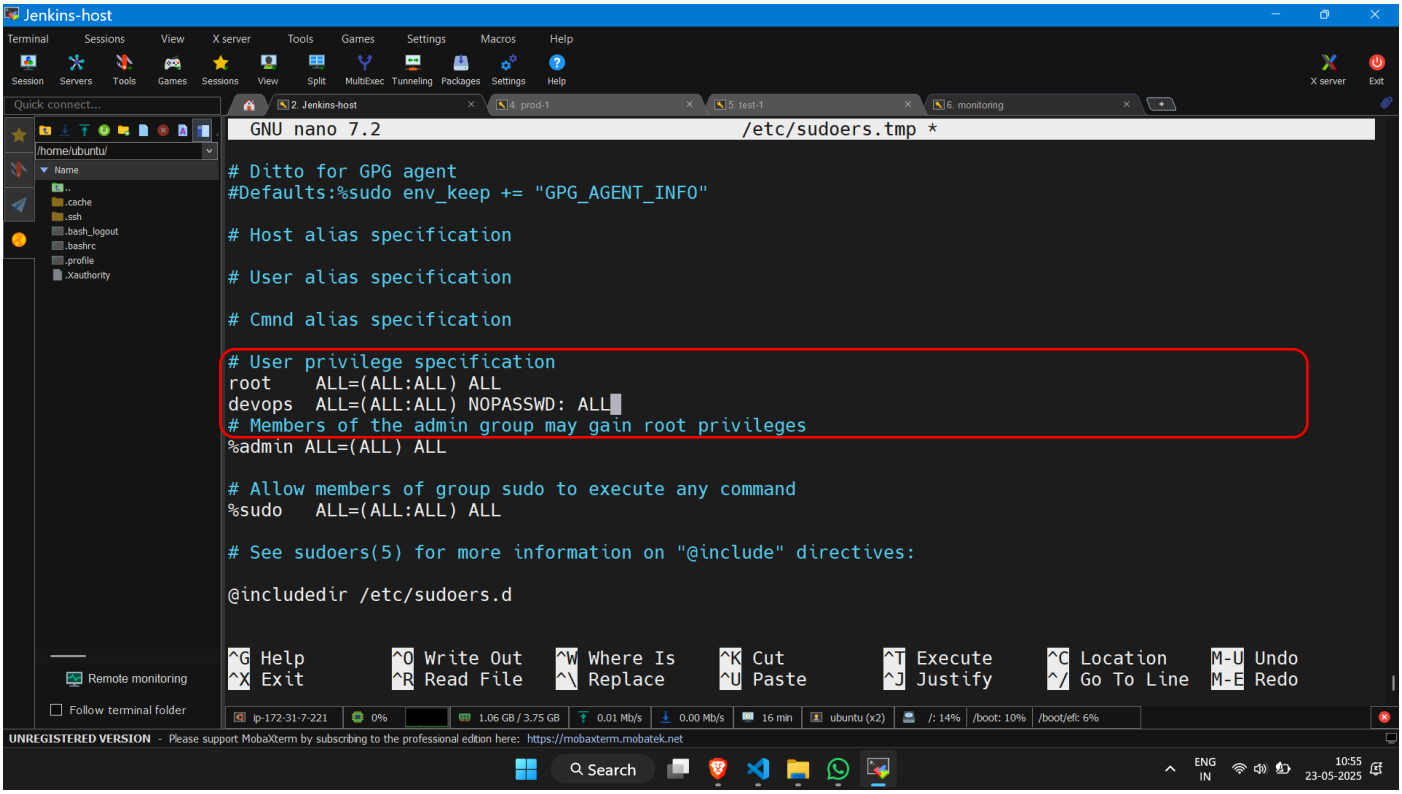


Ansible Setup and configuration with other instances

As a root user add a user devops with password in the master machine and also do the same in other instances (i.e production and test instances)

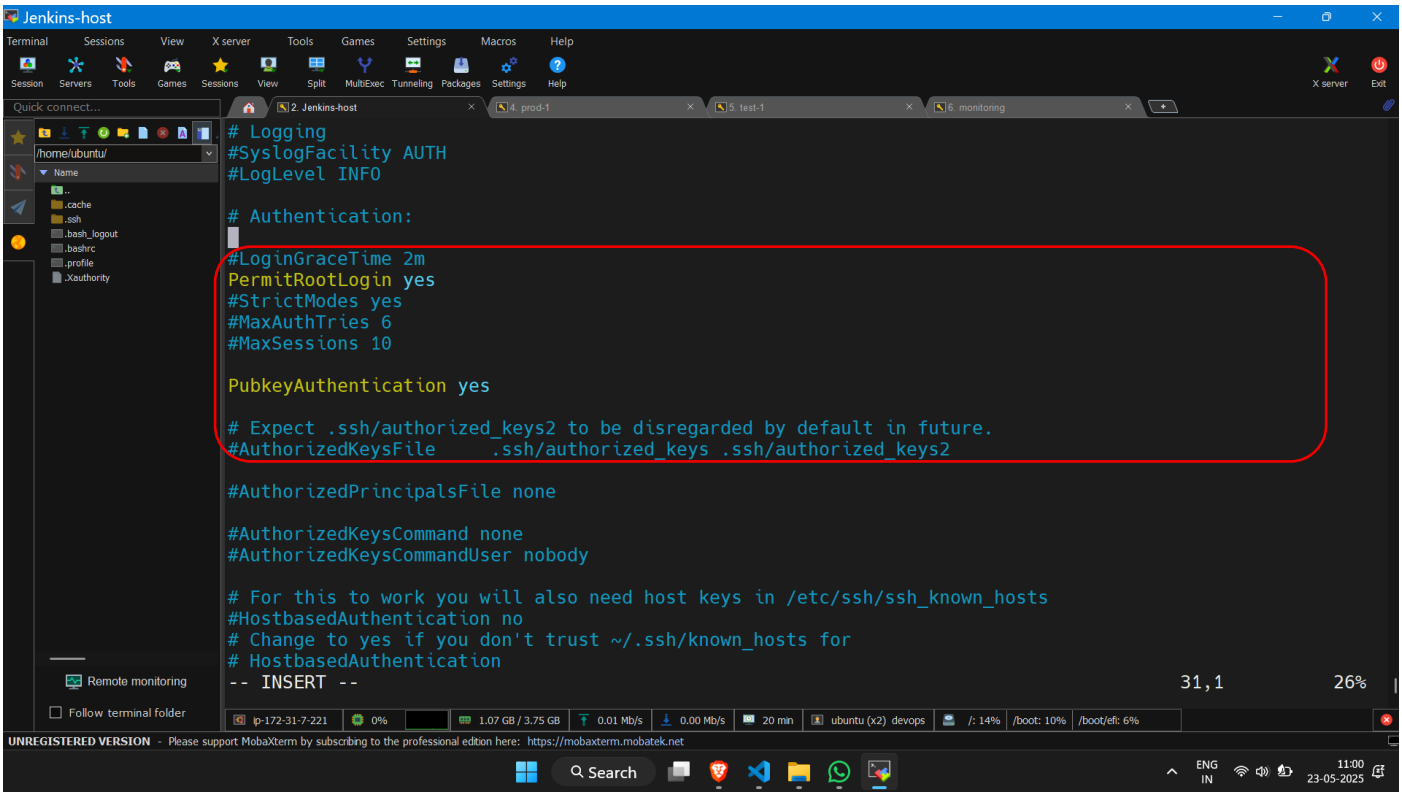


Now give the devops user sudo permission by adding it to the sudoers file do the same in all the other nodes also



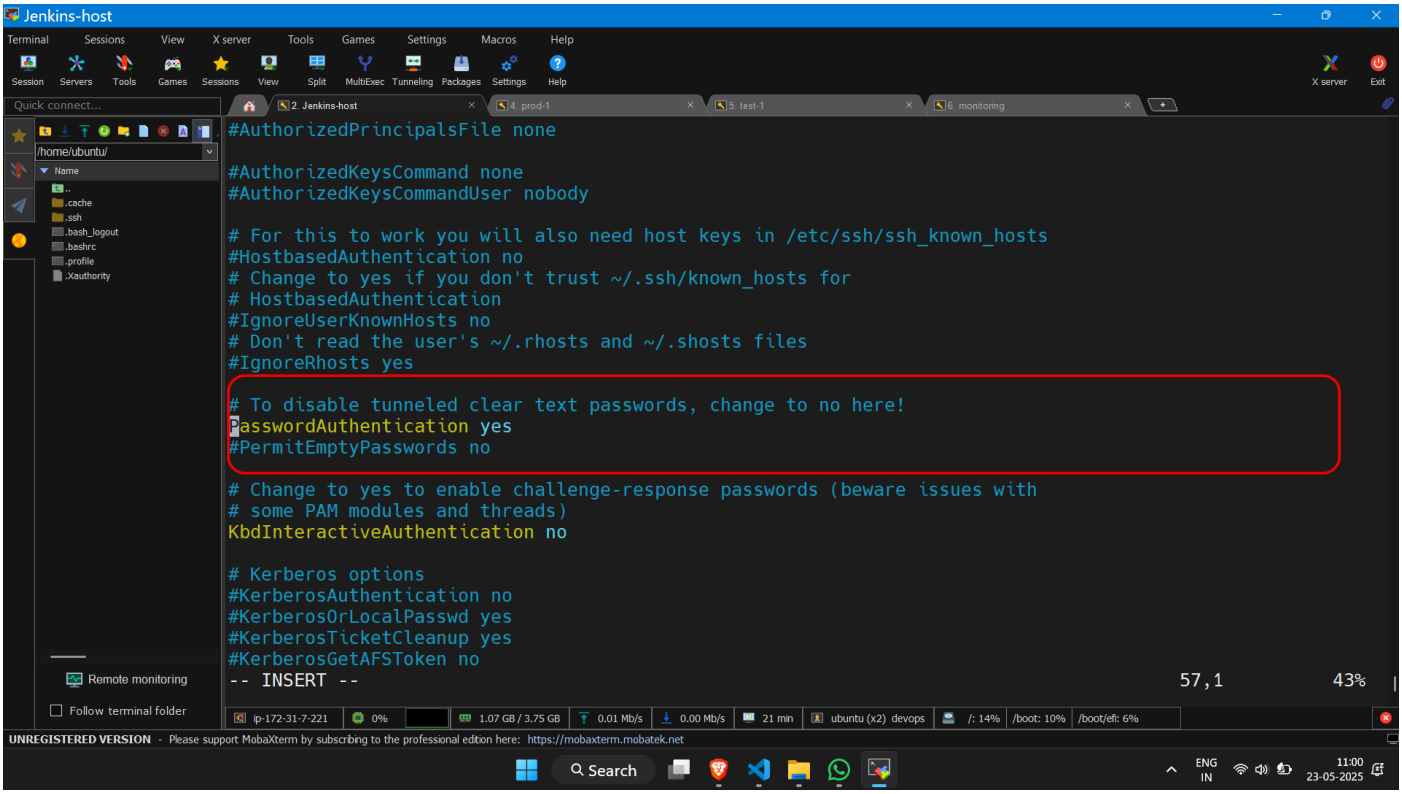
Now we have to do sshd configuration I the file ssh_config use command “vi /etc/ssh/sshd_config” to get into the file and then edit the “ PemitRootLogin permission to yes “ and and remove

“# from PubkeyAuthentication “



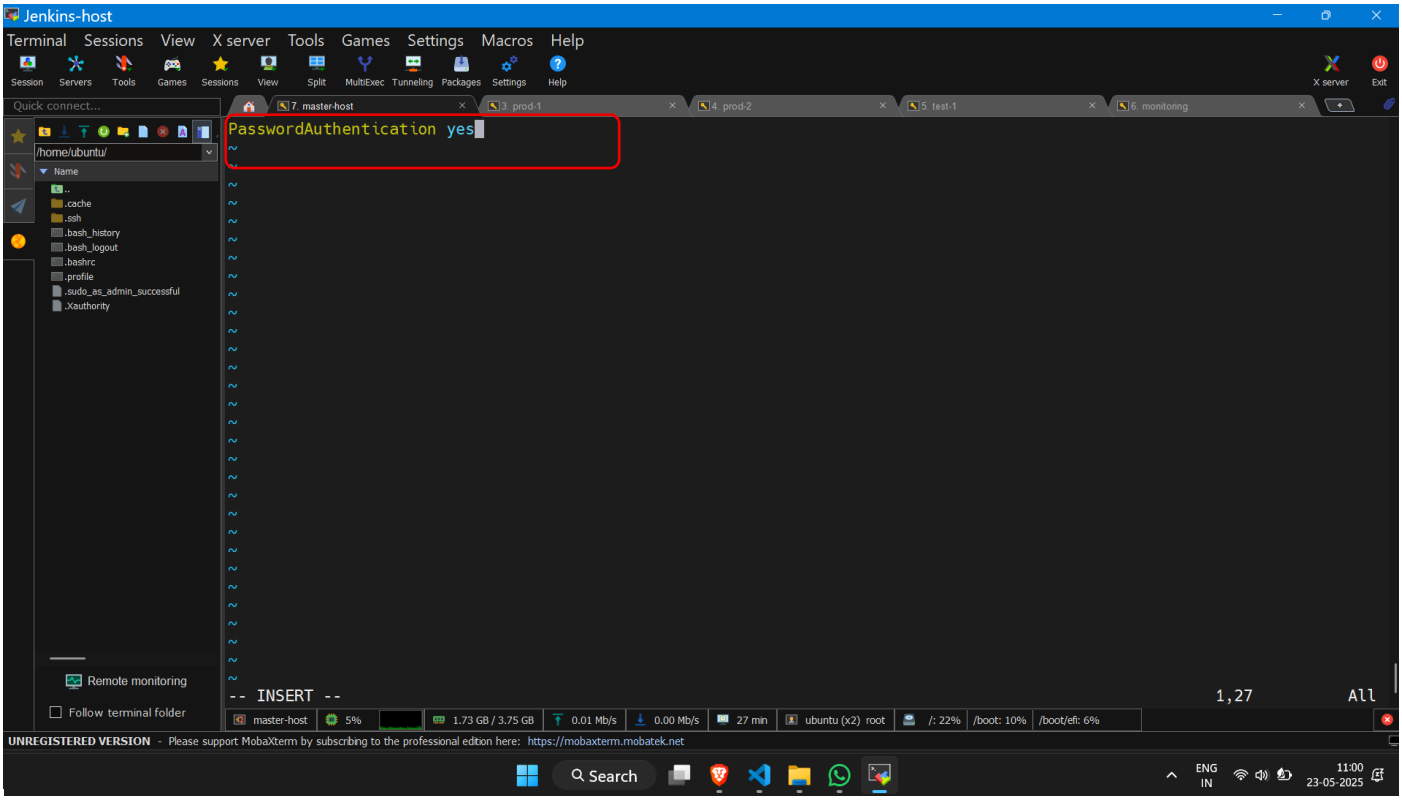
In the same file do down and remove “ # from PassswordAuthentication “

And the save and exit the file



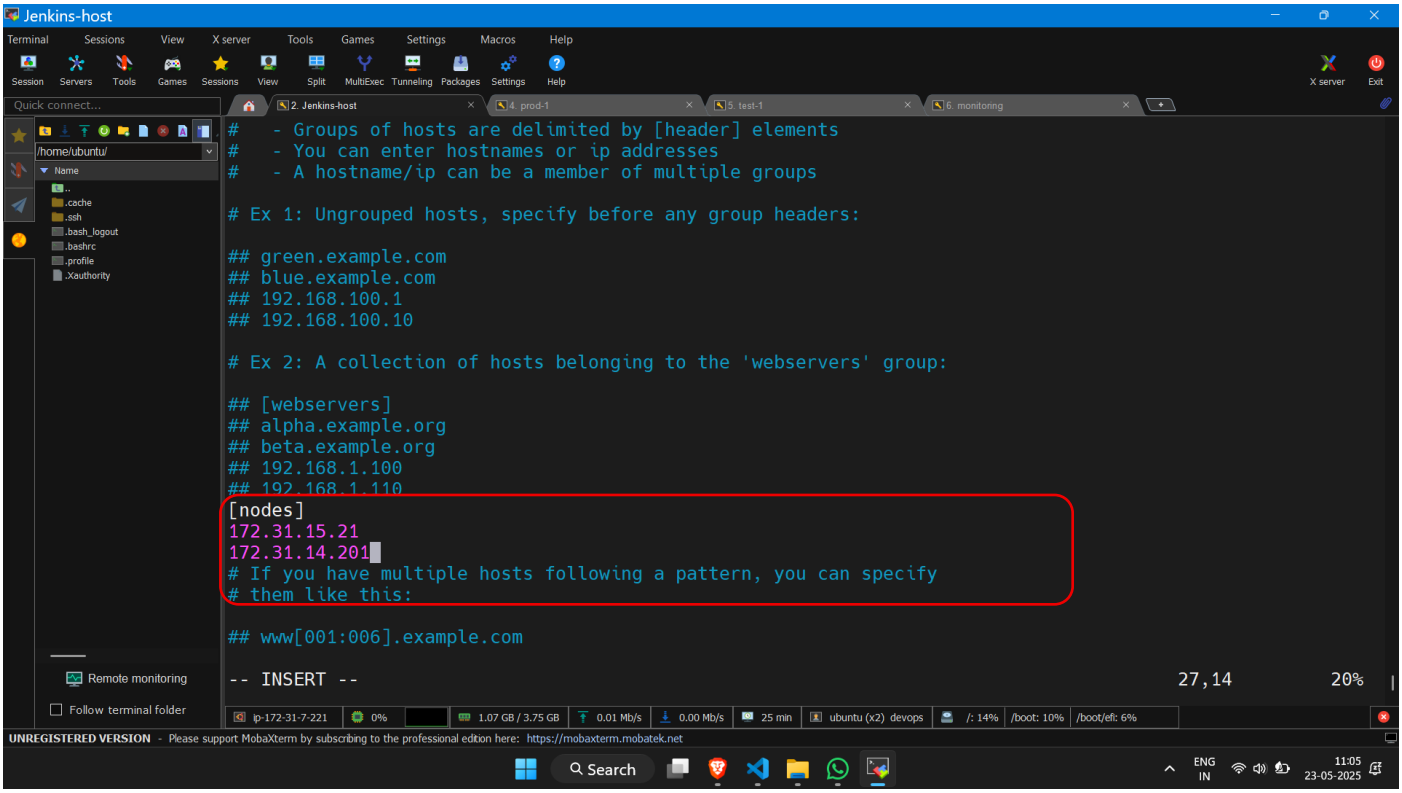
Also edit the setting.conf file which has to be done for new ubuntu version instances doing this by command “vi /etc/ssh/sshd_config.d/60-cloudimg-settings.conf “

Do all the file changes in all the other nodes



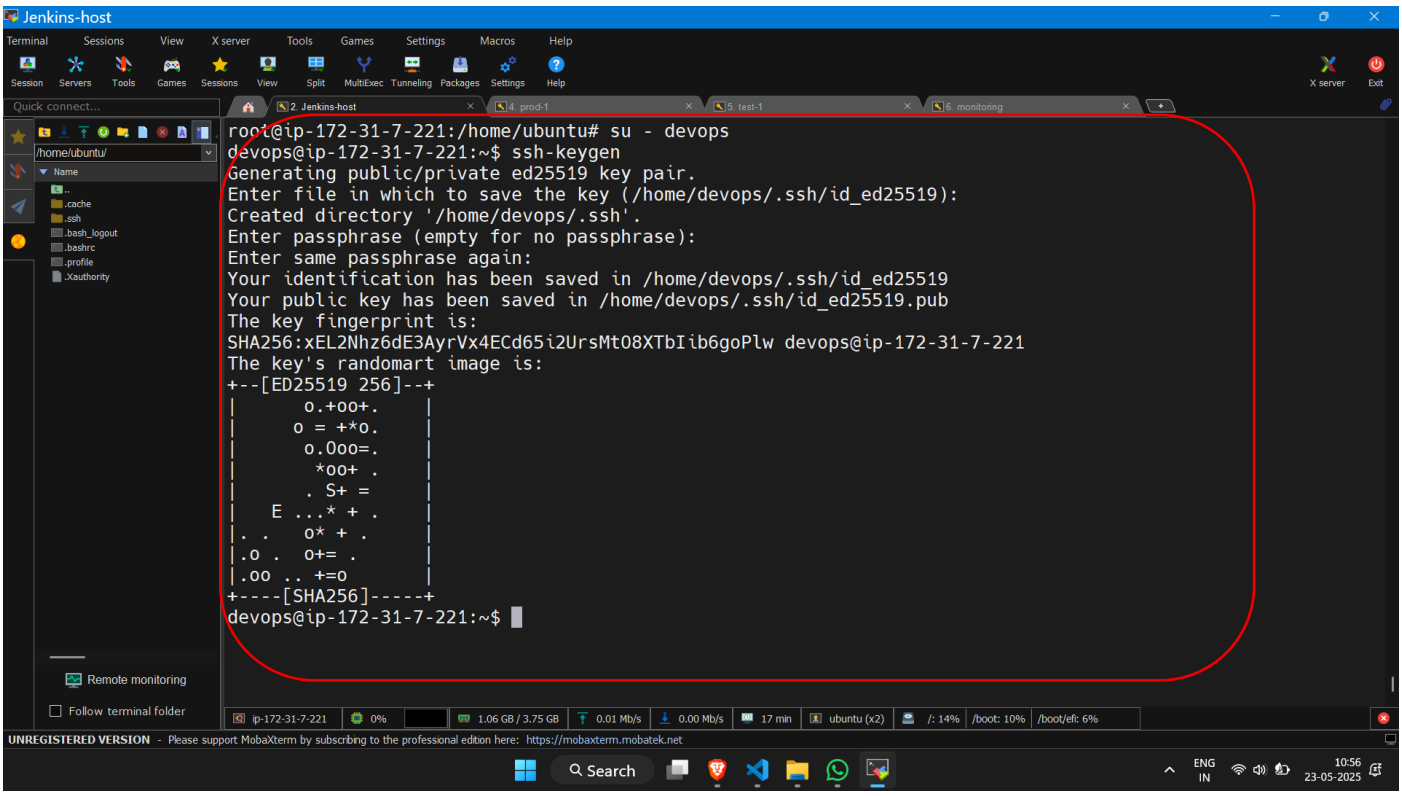
And add the inventory file the private ip of the nodes instances which has to be connected to the controller in the file

Hosts using the command “ vi /etc/ansible/hosts “



And then do “service ssh restart “ and then go to the devops user using command “ su – devops “

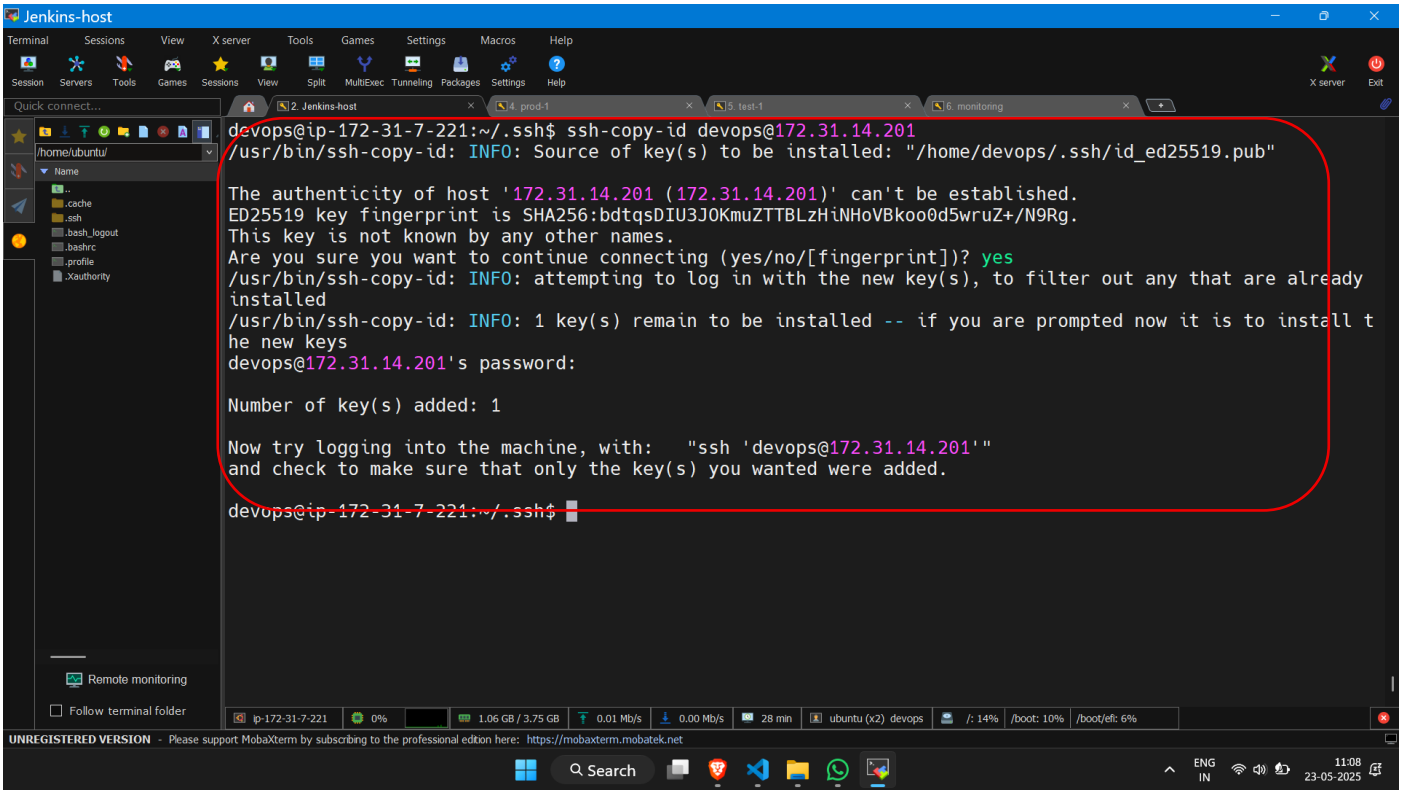
Then generate a ssh key using command “ssh-keygen “



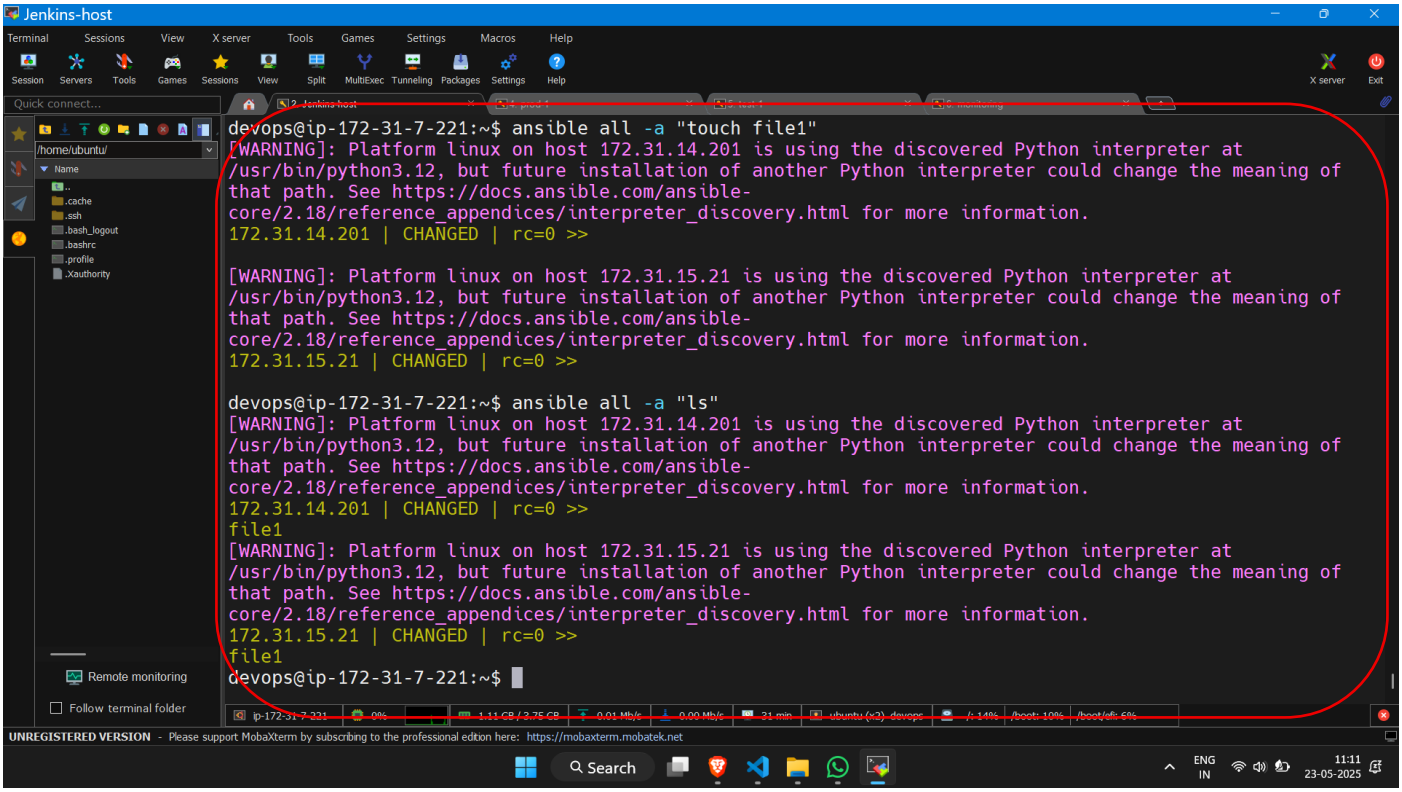
The go to the .ssh directory now we have to copy the ssh key other nodes on them using the command “ ssh-copy-id devops@<private_ip_of_instaces> “

Then enter the password in order to connect which has been set during the creation of the user in other nodes

Do this for other instance too using the private ip



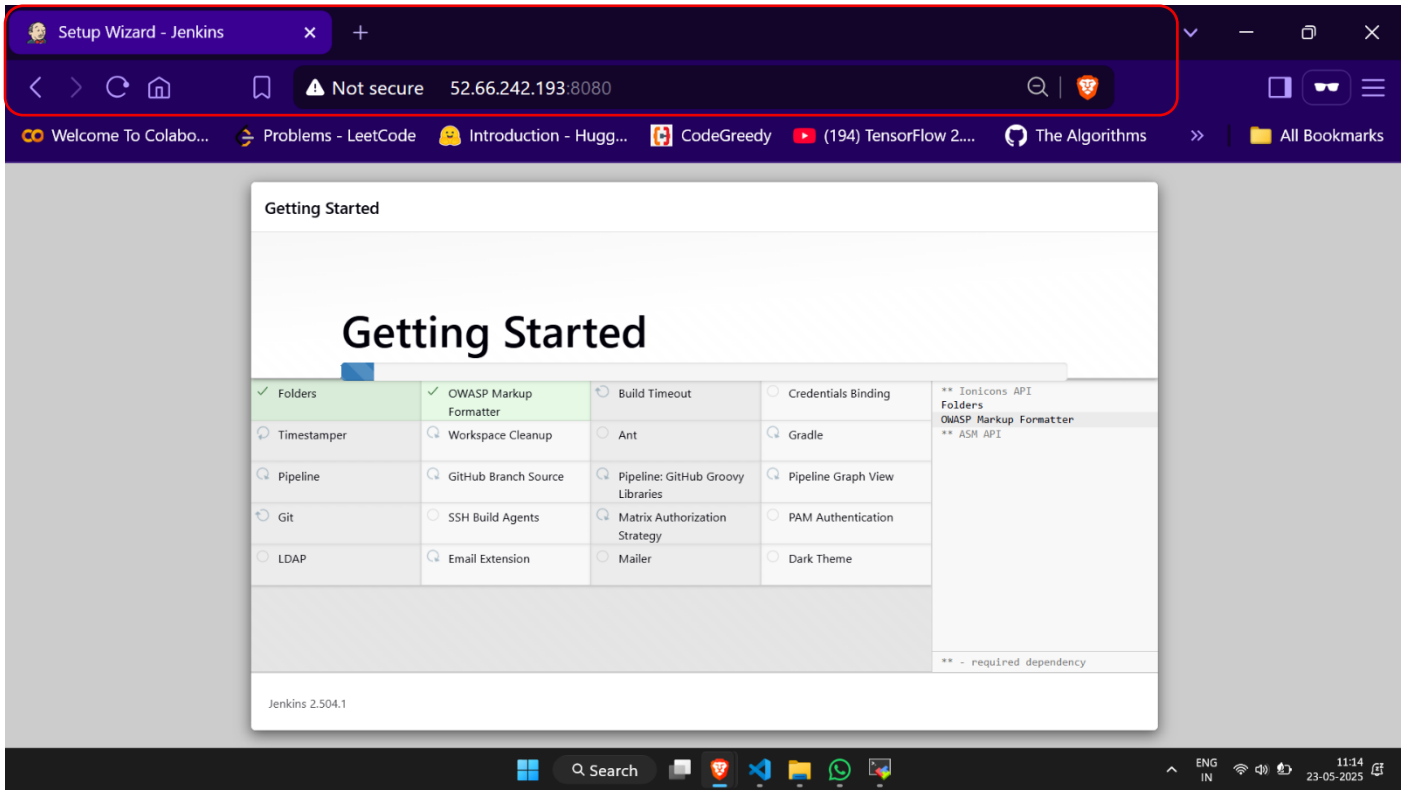
Now in order to check the connection run command [ansible all -a “touch file1”] and [ansible all -a “ls”] you will see that the files has been created successfully in the nodes



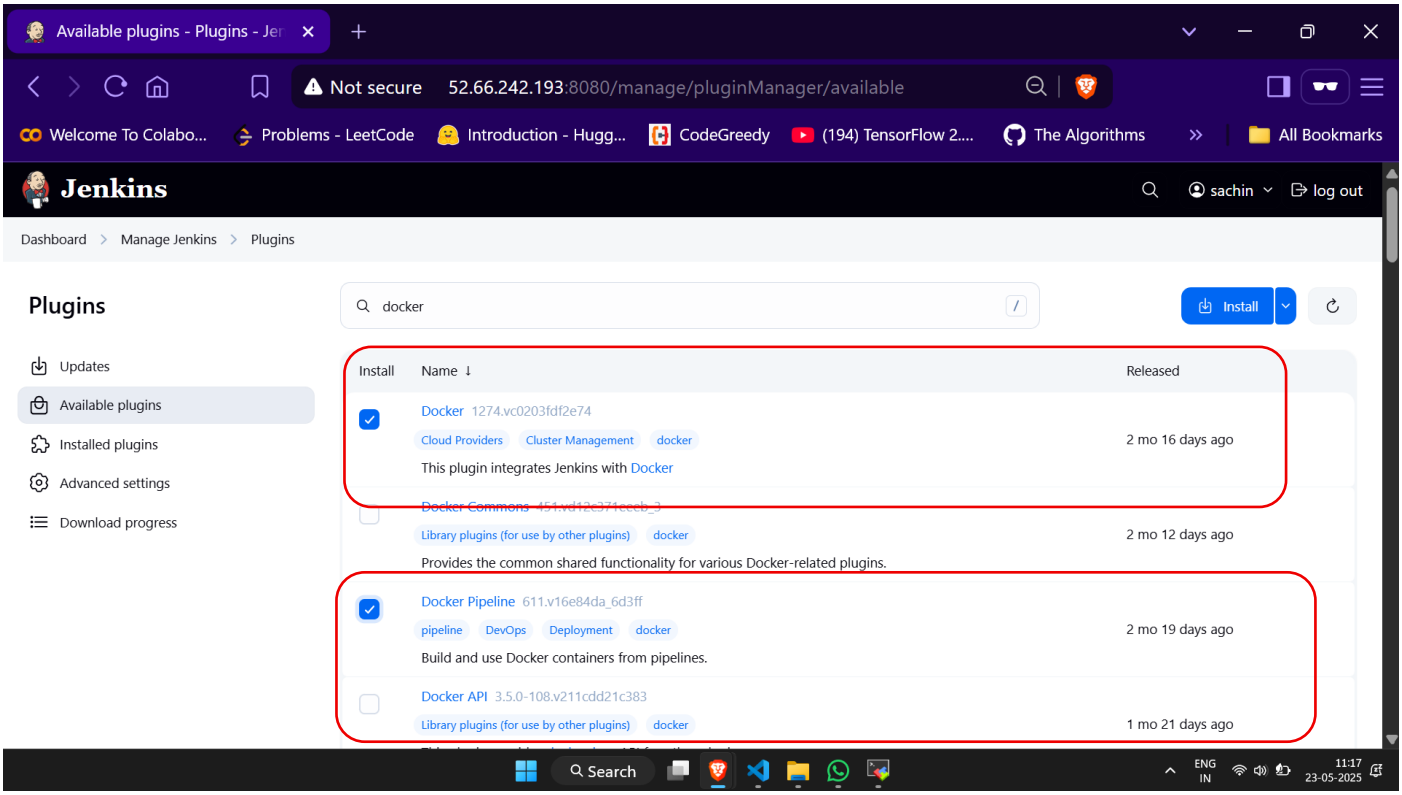
Now access the Jenkins application using the public ip of the host machine on the port 8080 ensuring that the security group is configured correctly which allows 8080 port in the inbound rules

Login using the Jenkins password which is stored at /var/lib/jenkins/secrets/initialAdminPassword

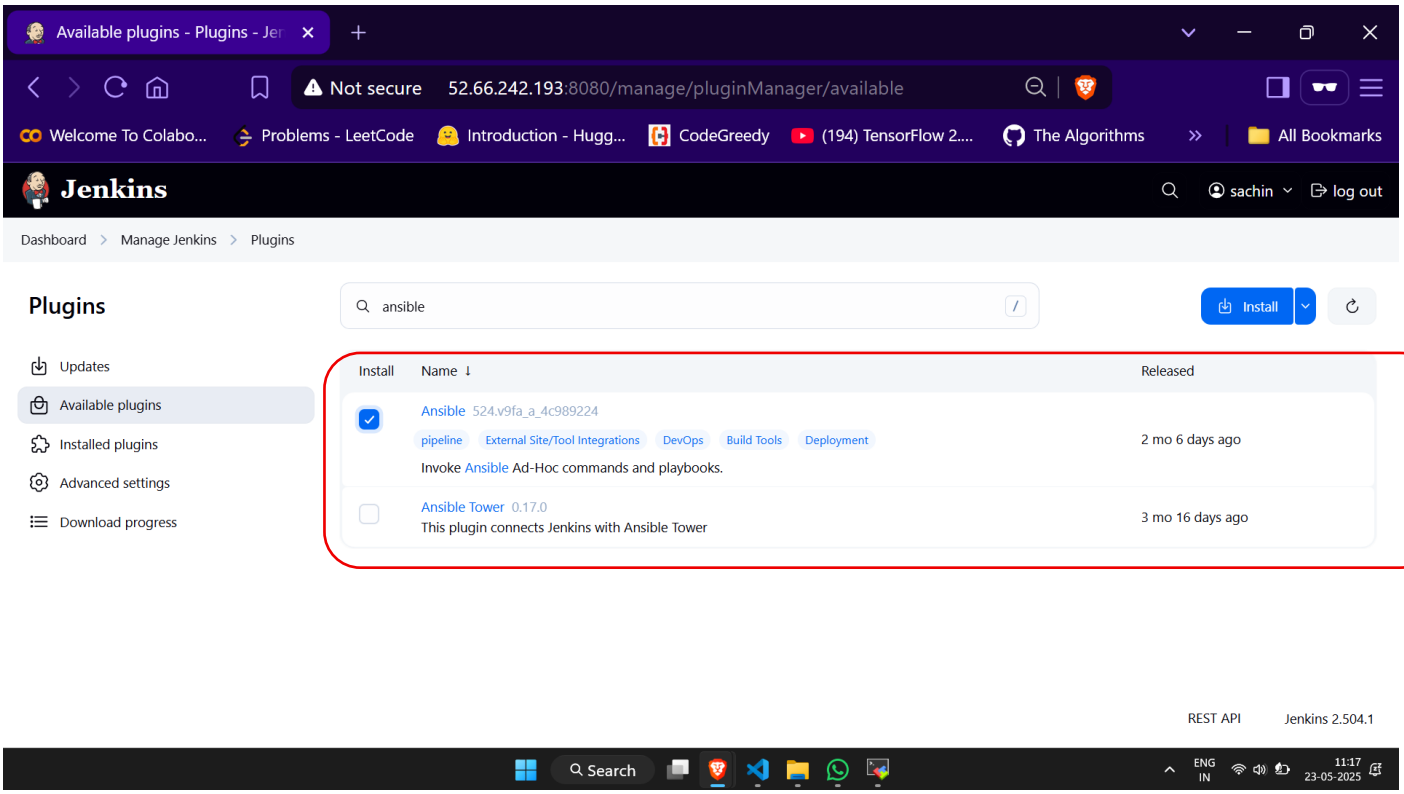
Then set user required name and password



After loging in navigate to the manage Jenkins in the nav bar and then -> plugins -> then install docker plugins

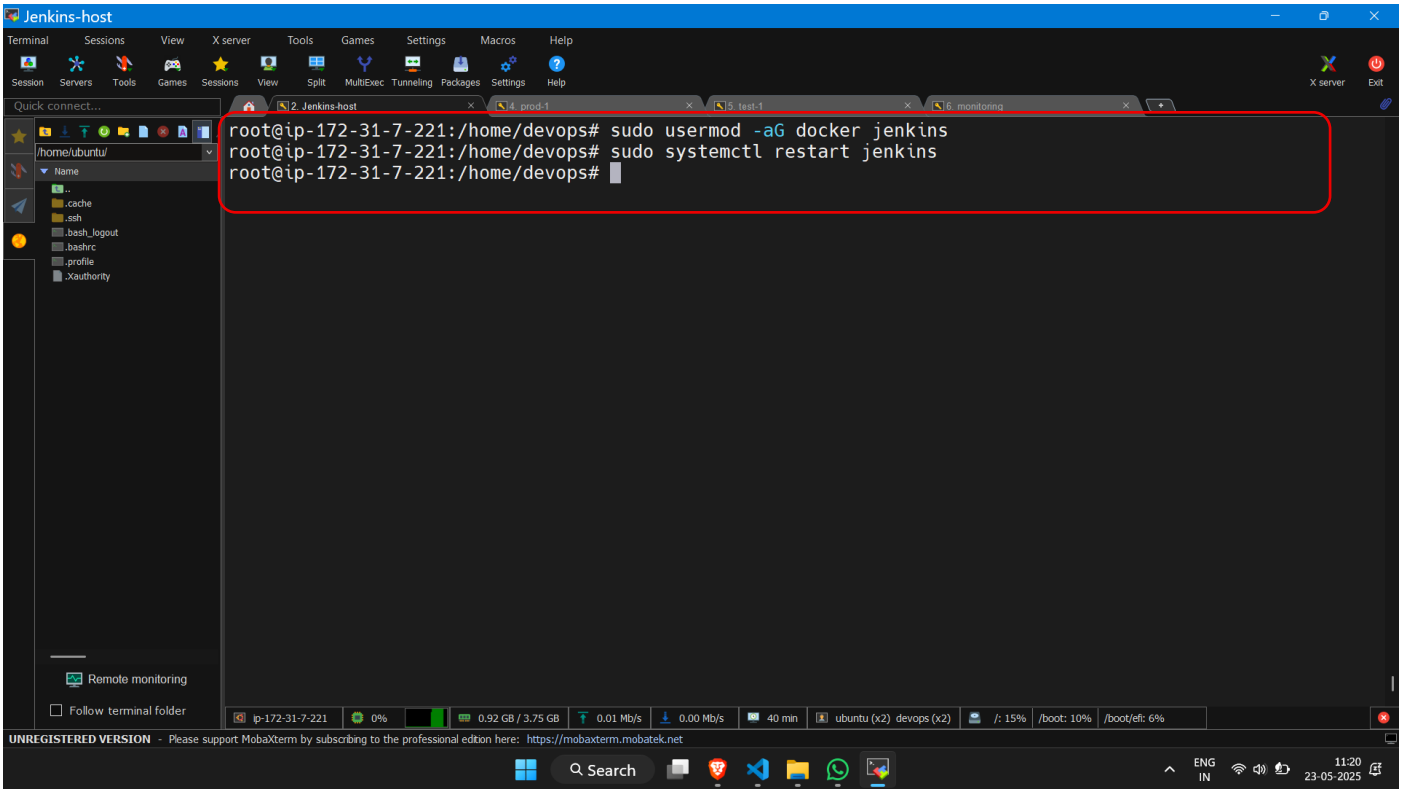


Also install ansible plugin

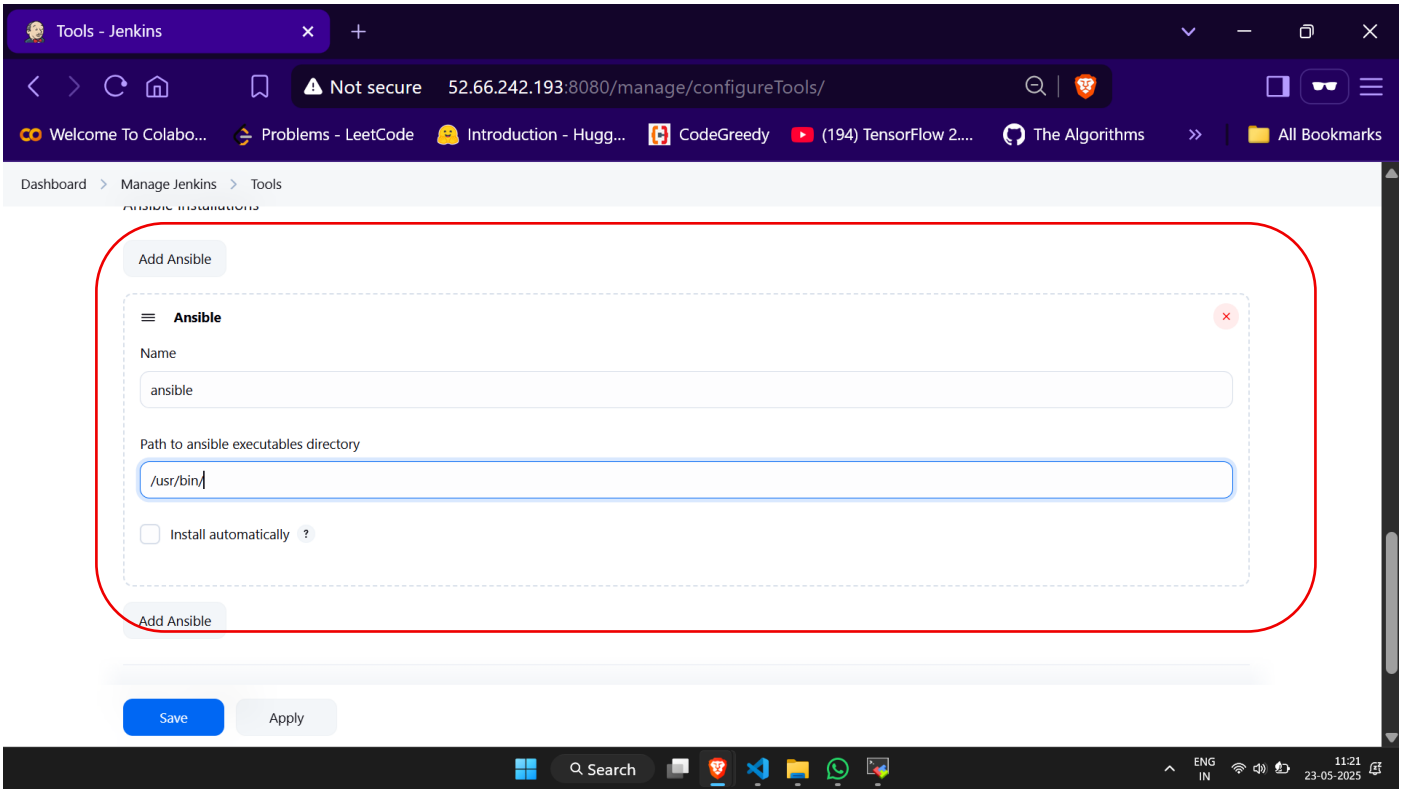


Also set the docker user permission to the Jenkins using the command “sudo usermod -aG docker Jenkins”

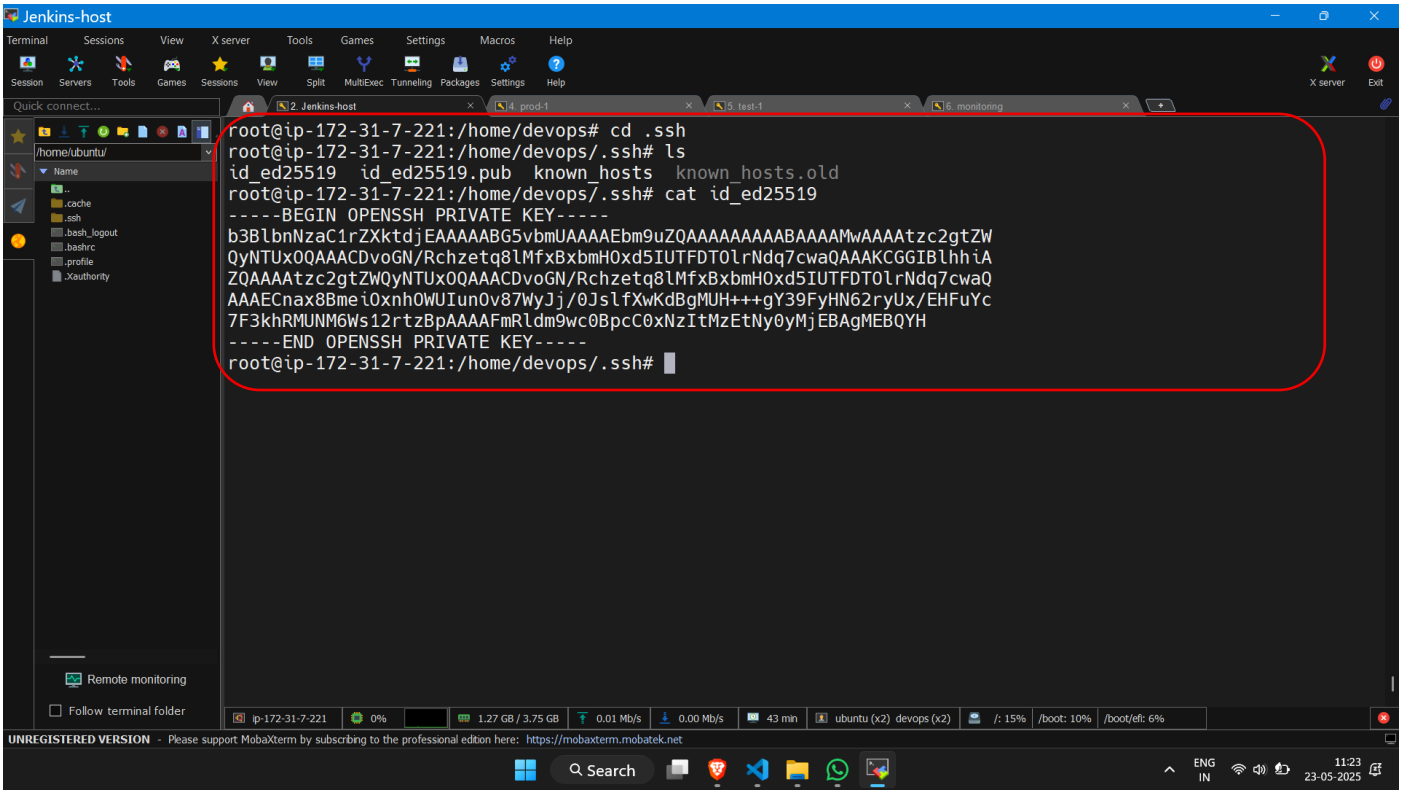
And then restart the Jenkins service



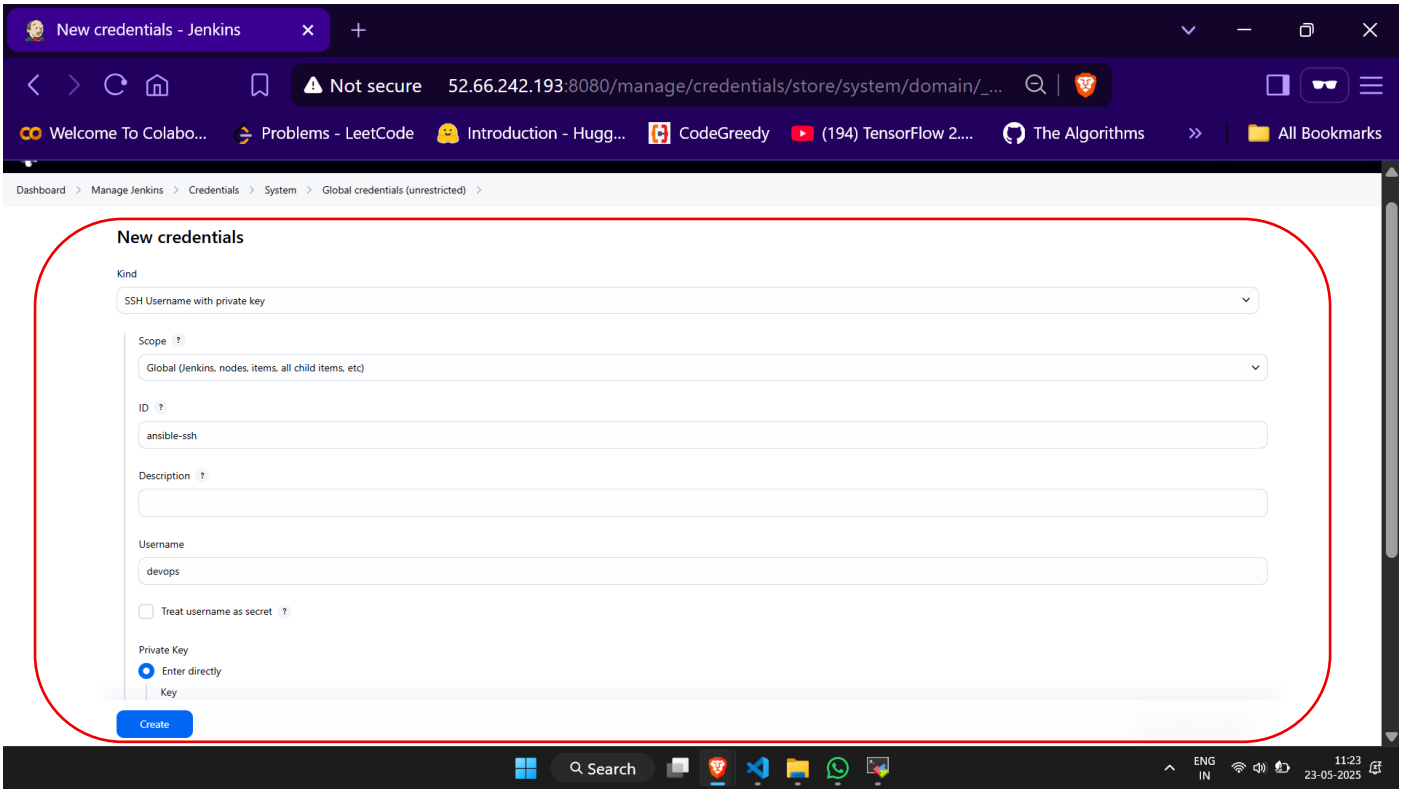
Configure the ansible path in the jenkins

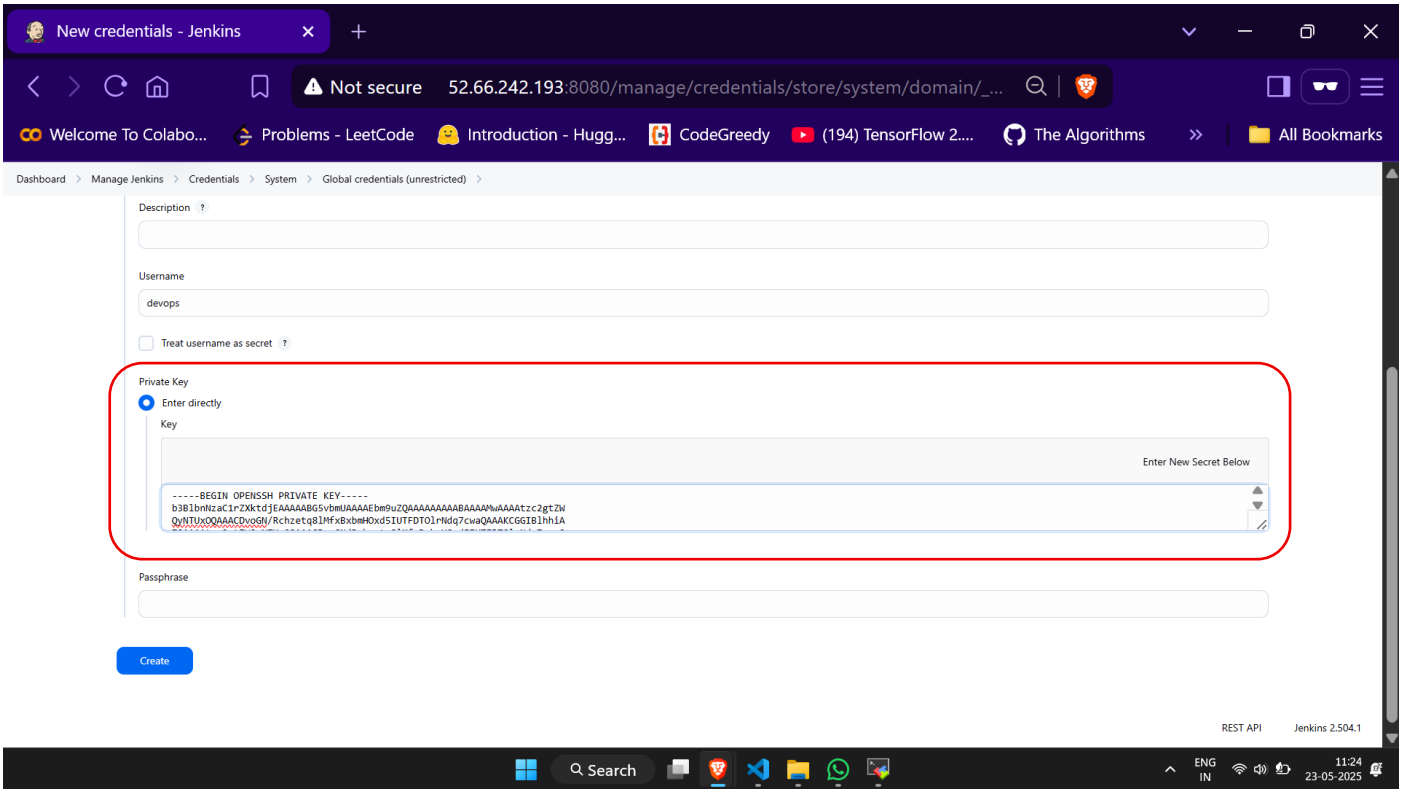


Now copy the ssh key which is generated during ansible setup to add that credentials into jenkins

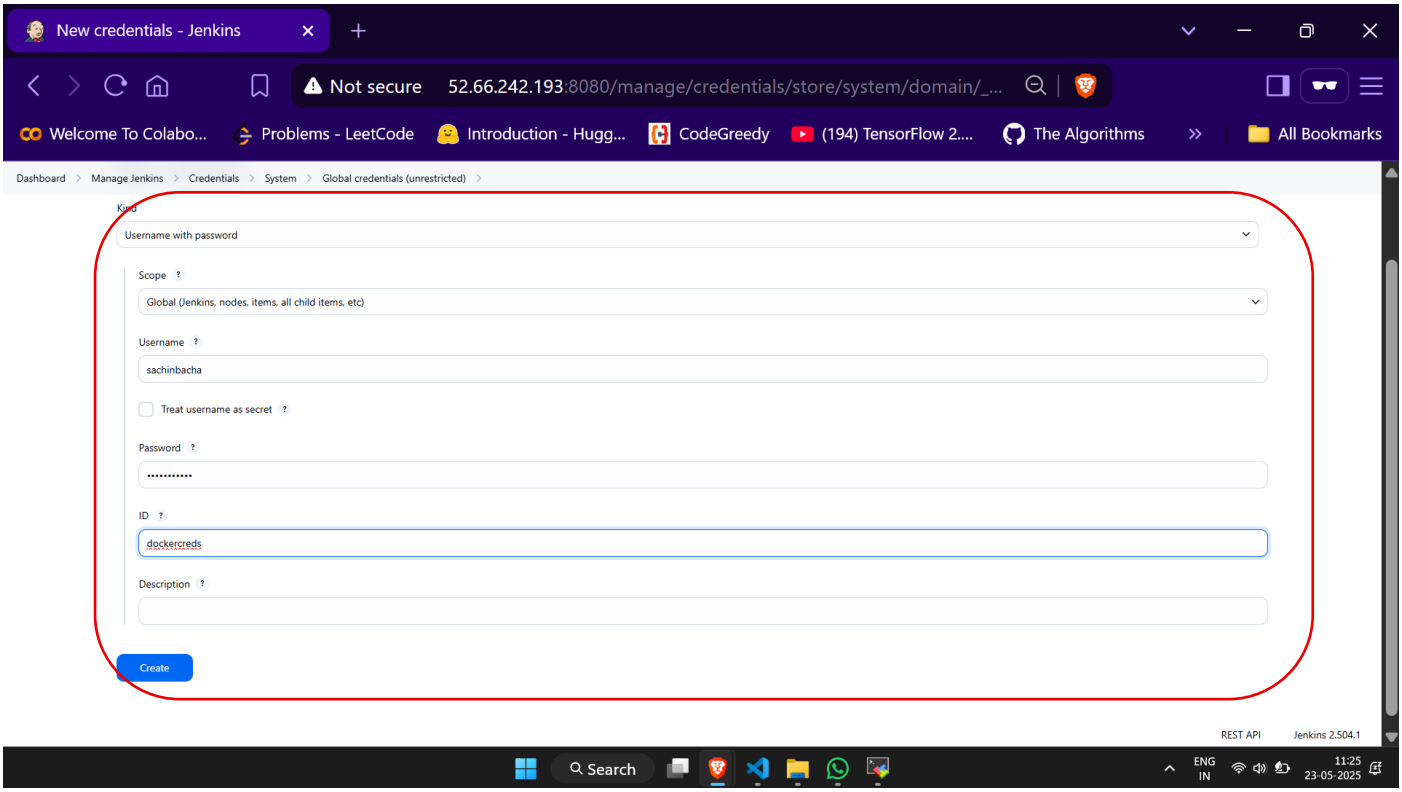


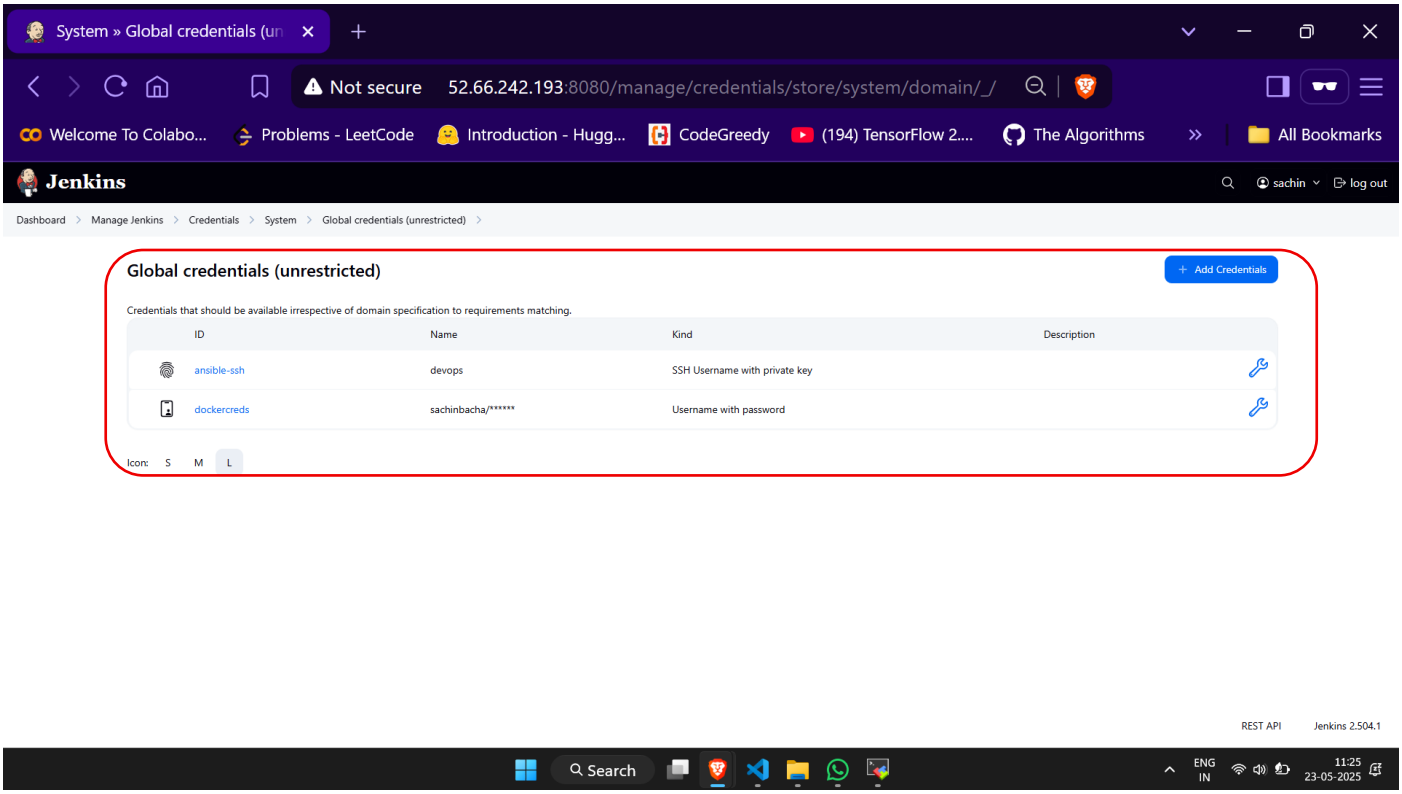
Under the credentials section add new credential -> then select ssh username and privatekey -> and provide id ,username, and paste the secret key which was copied previously in the private key section using enter directly button .





Now also add the docker credentials in the credentials section and select username and password and add dockerhub username and password

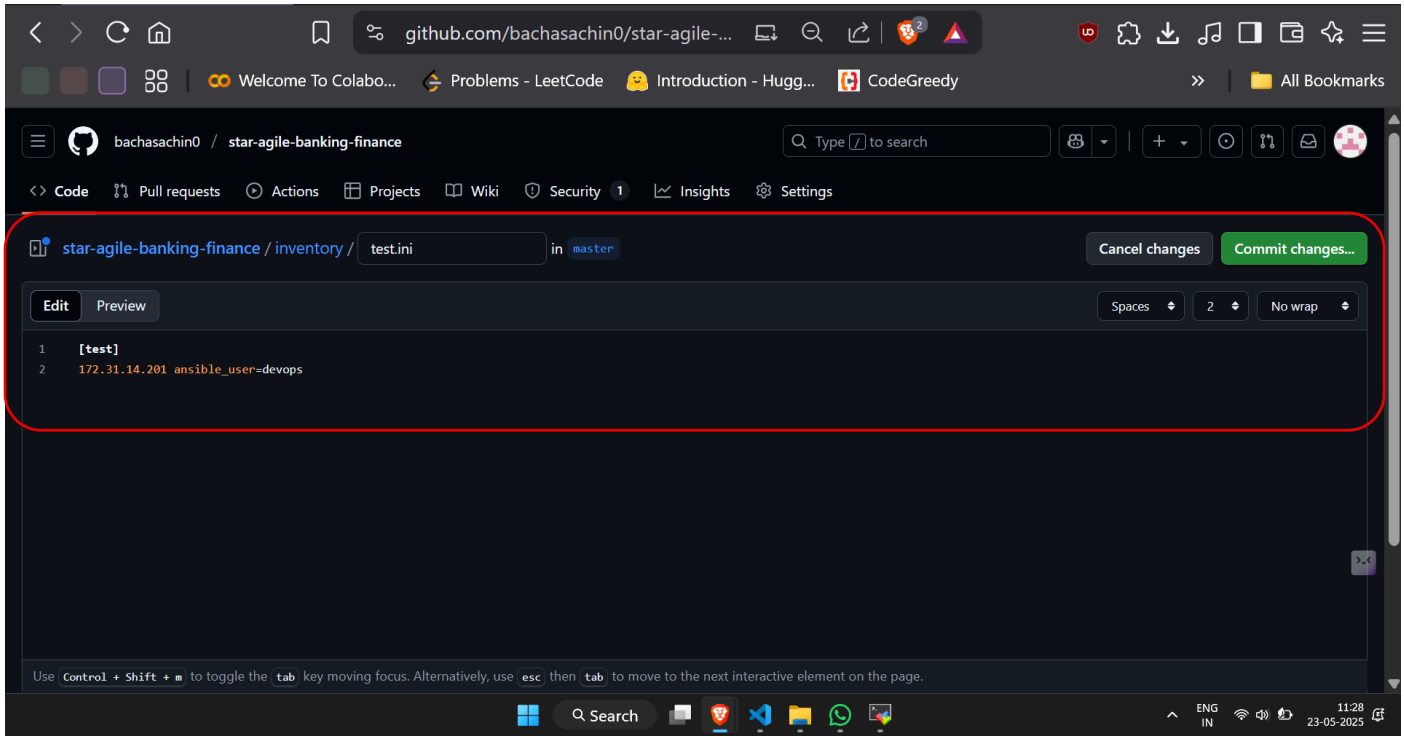




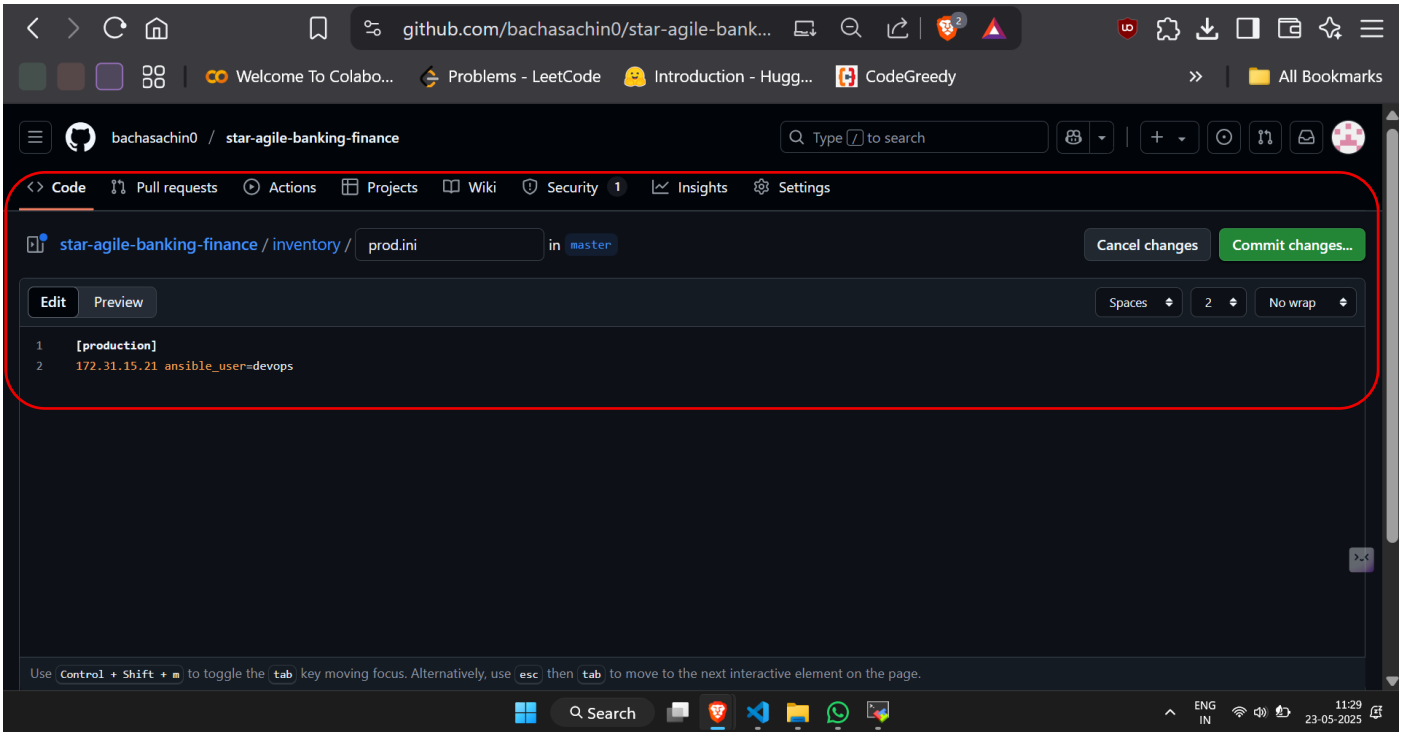
Now go to the github repository where you have your project source code .

Here I am adding the inventory file by myself although it is not recommended but for proof of concept I am creating a

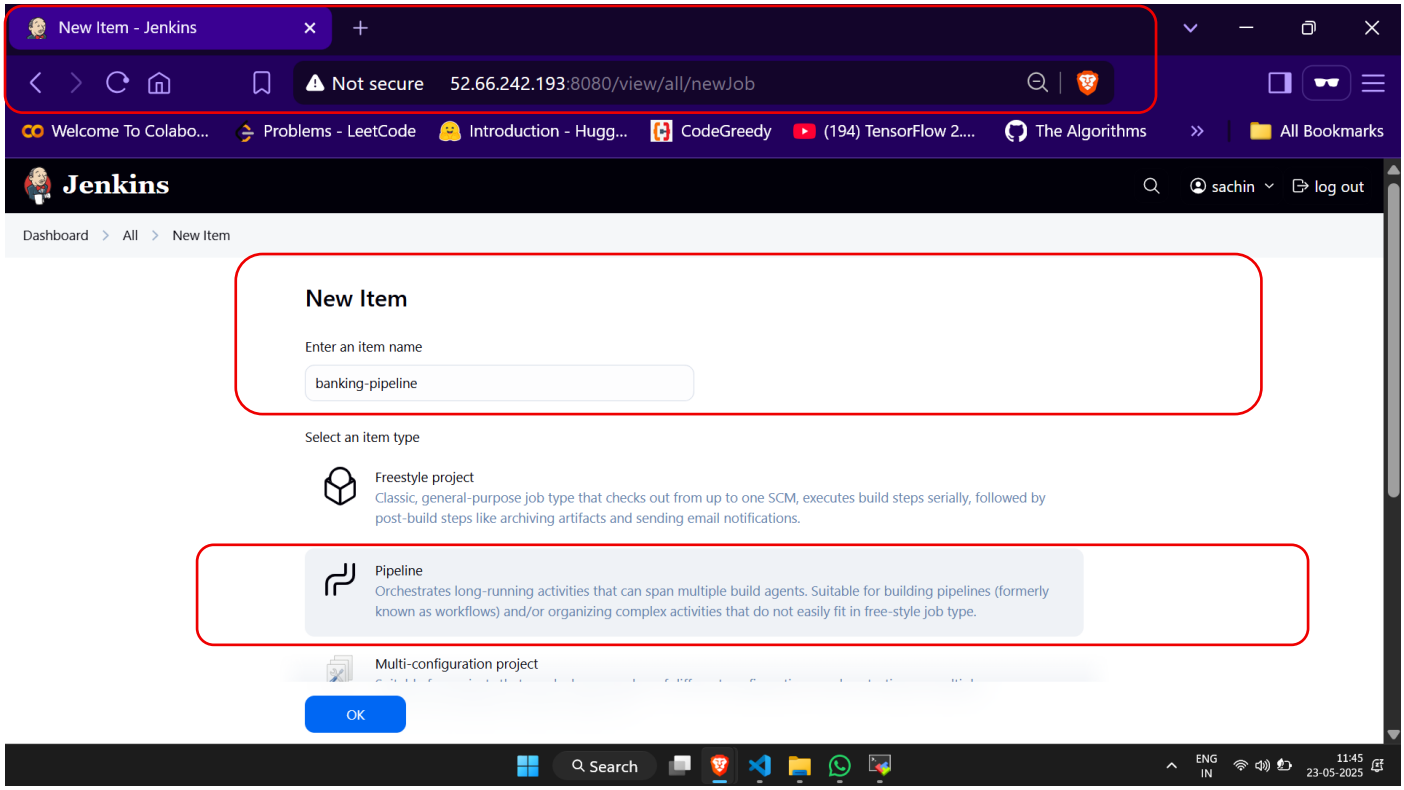
Inventory file which consist of two files test.ini and prod.ini which nothing but similar to the hosts setup we have done during the ansible setup consist of private ip of the instances to be connected.



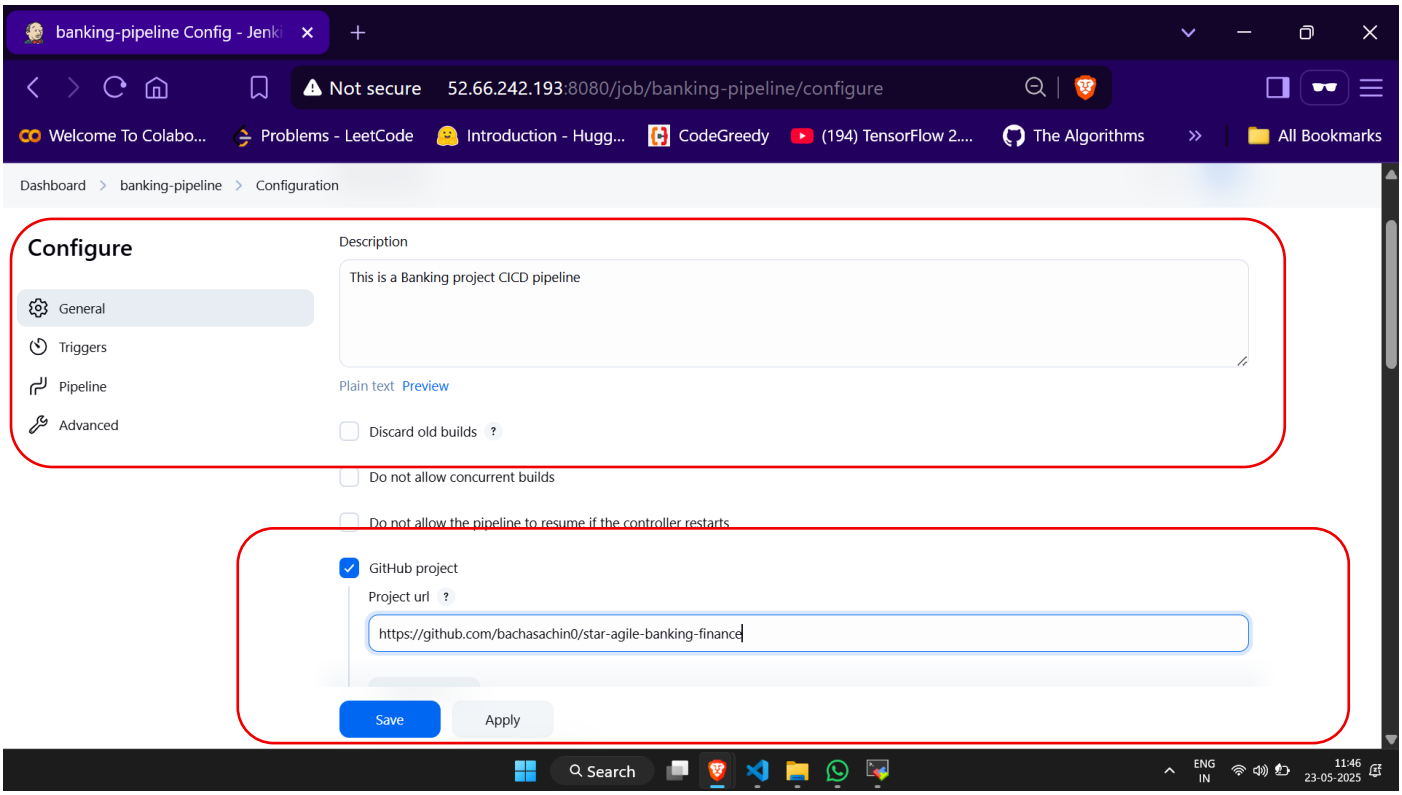
Add the private ip of the production server



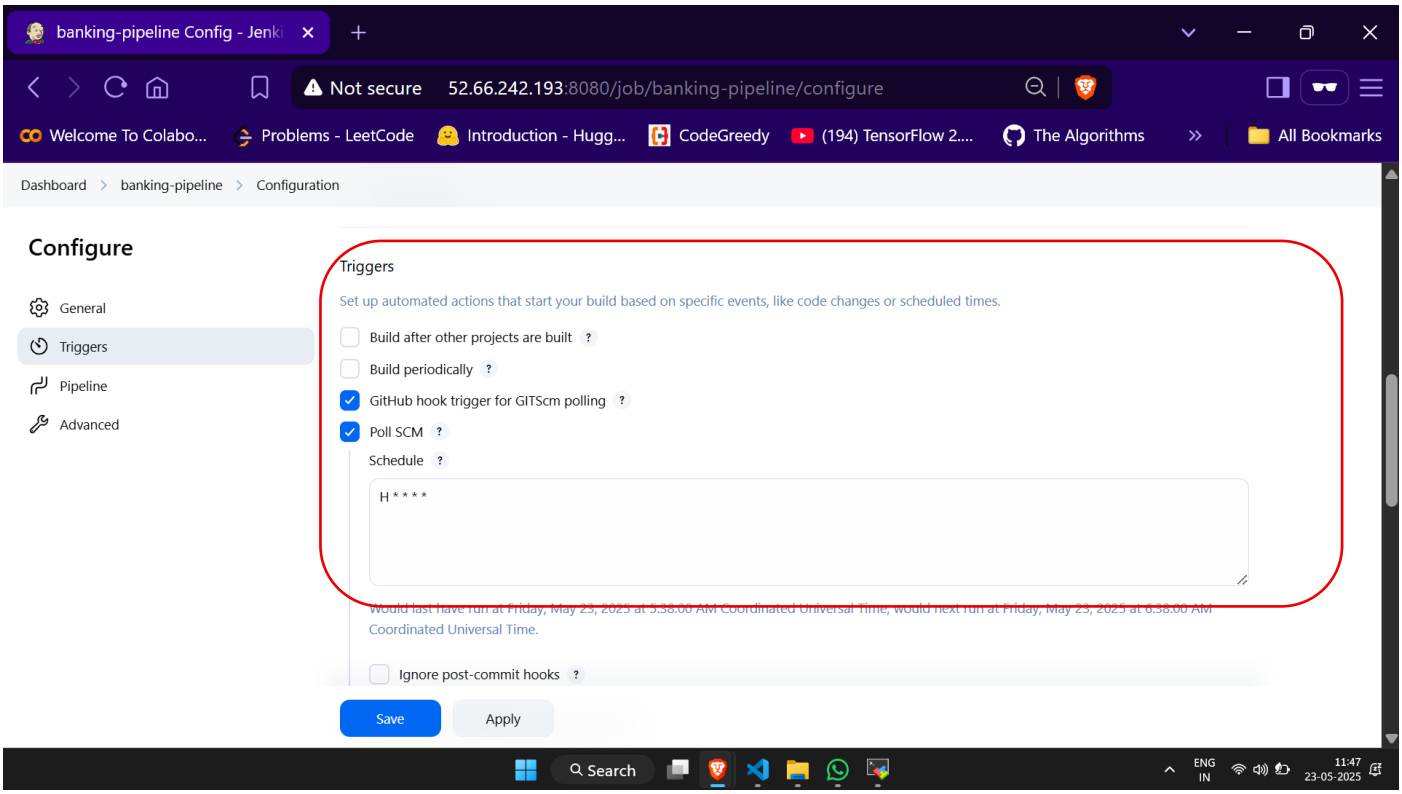
Now create a new item in Jenkins -> enter the name -> and select pipeline (type of your project)



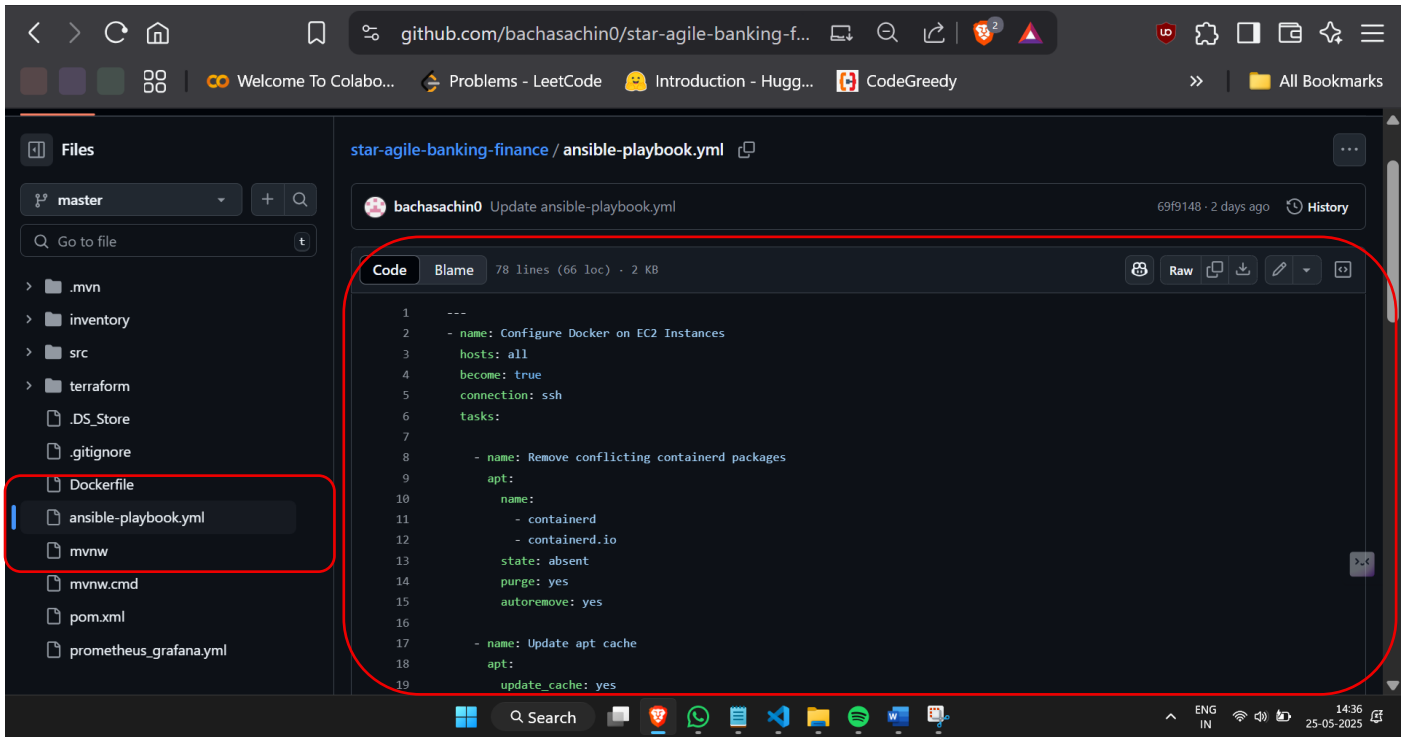
Now do general configuration of the project add description for details and add the github repo link

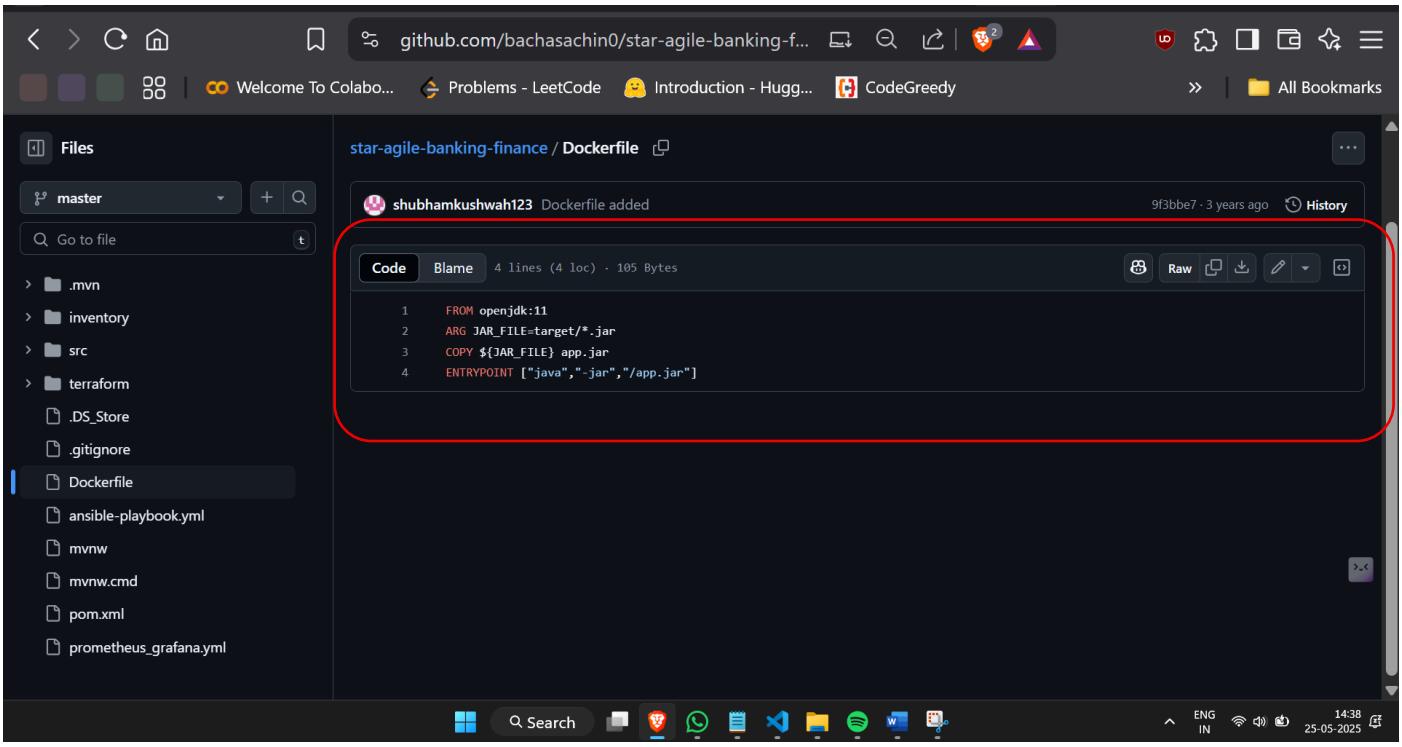


Add the triggers github poll scm and poll scm

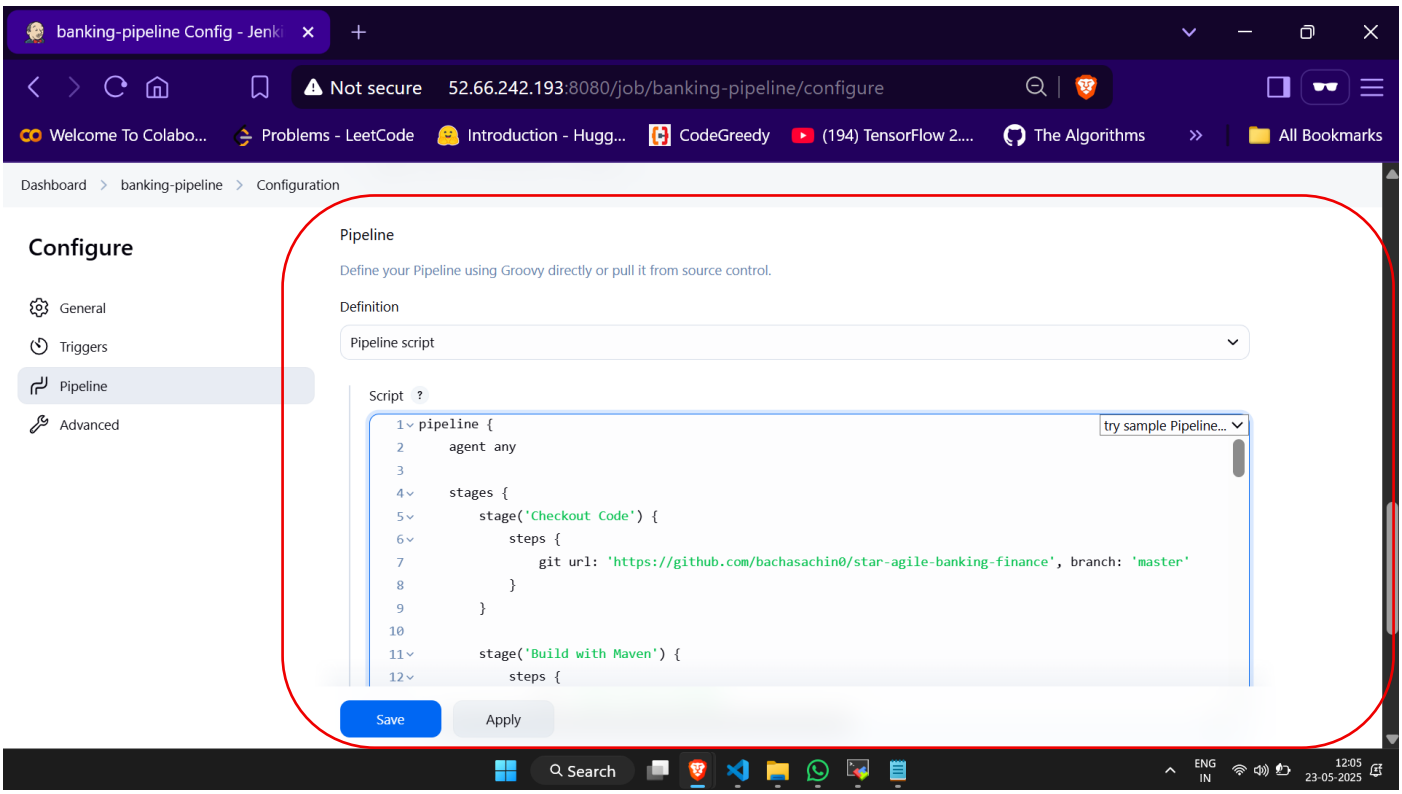


Before writing the pipeline script ensure you have the Dockerfile and ansible-playbook to deploy the application .





Now write the pipeline script to deploy this application . first it should git checkout -> build the application -> docker contenerization -> then push to the docker hub -> then deploy it to the test instance -> then perform test -> if the test passes then -> deploy it to the production machine .



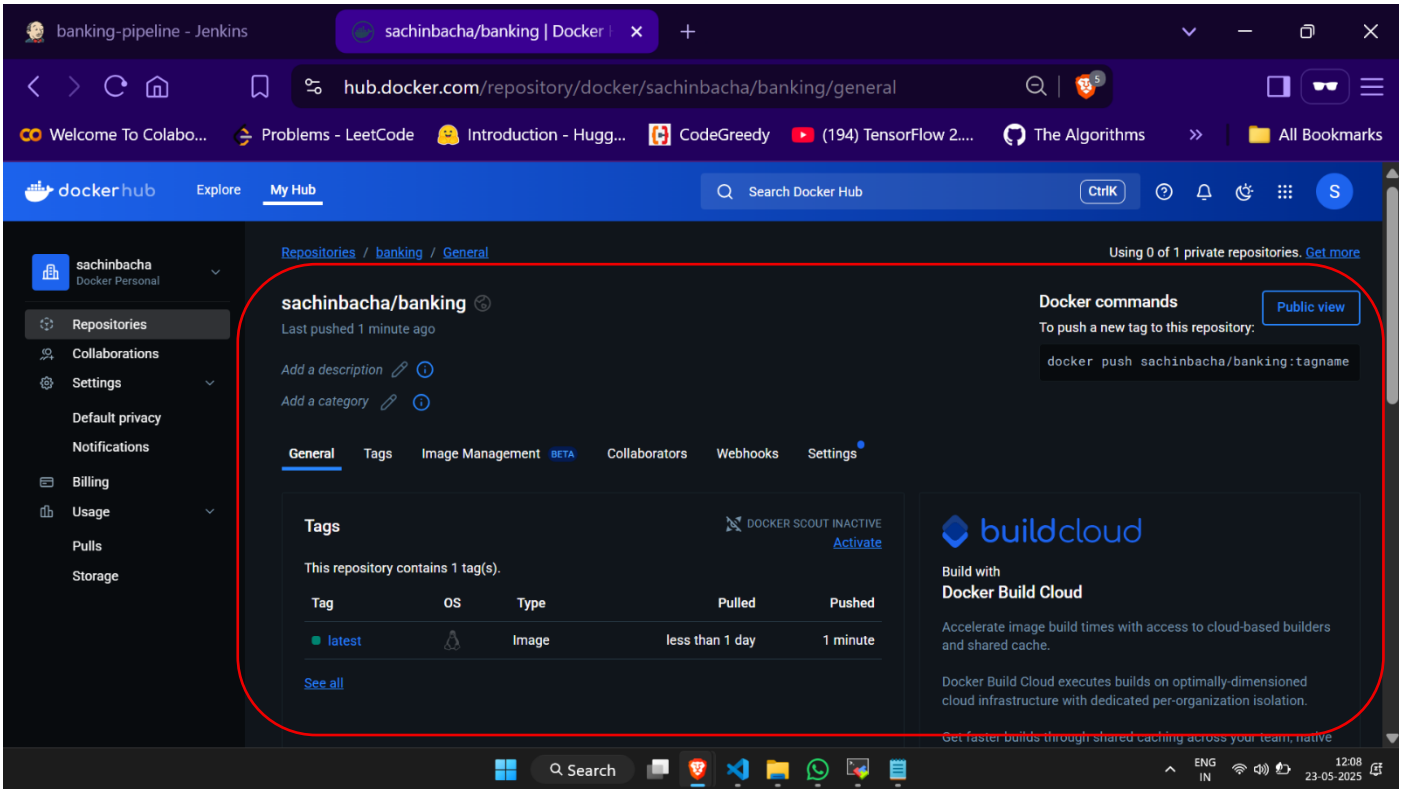
Jenkins pipeline script

```
pipeline {
    agent any
    stages {
        stage('Checkout Code') {
            steps {
                git url: 'https://github.com/bachasachin0/star-agile-banking-finance/tree/master', branch: 'master'
            }
        }

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

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

        stage('Deploy to Test Using Ansible') {
            steps {
                ansiblePlaybook credentialsId: 'ansible-ssh', disableHostKeyChecking: true, installation: 'ansible', inventory: 'inventory/test.ini', playbook: 'ansible-playbook.yml', vaultTmpPath: ''
            }
        }

        stage('Wait for Test App to Be Ready') {
          steps {
            sh 'sleep 5'
```

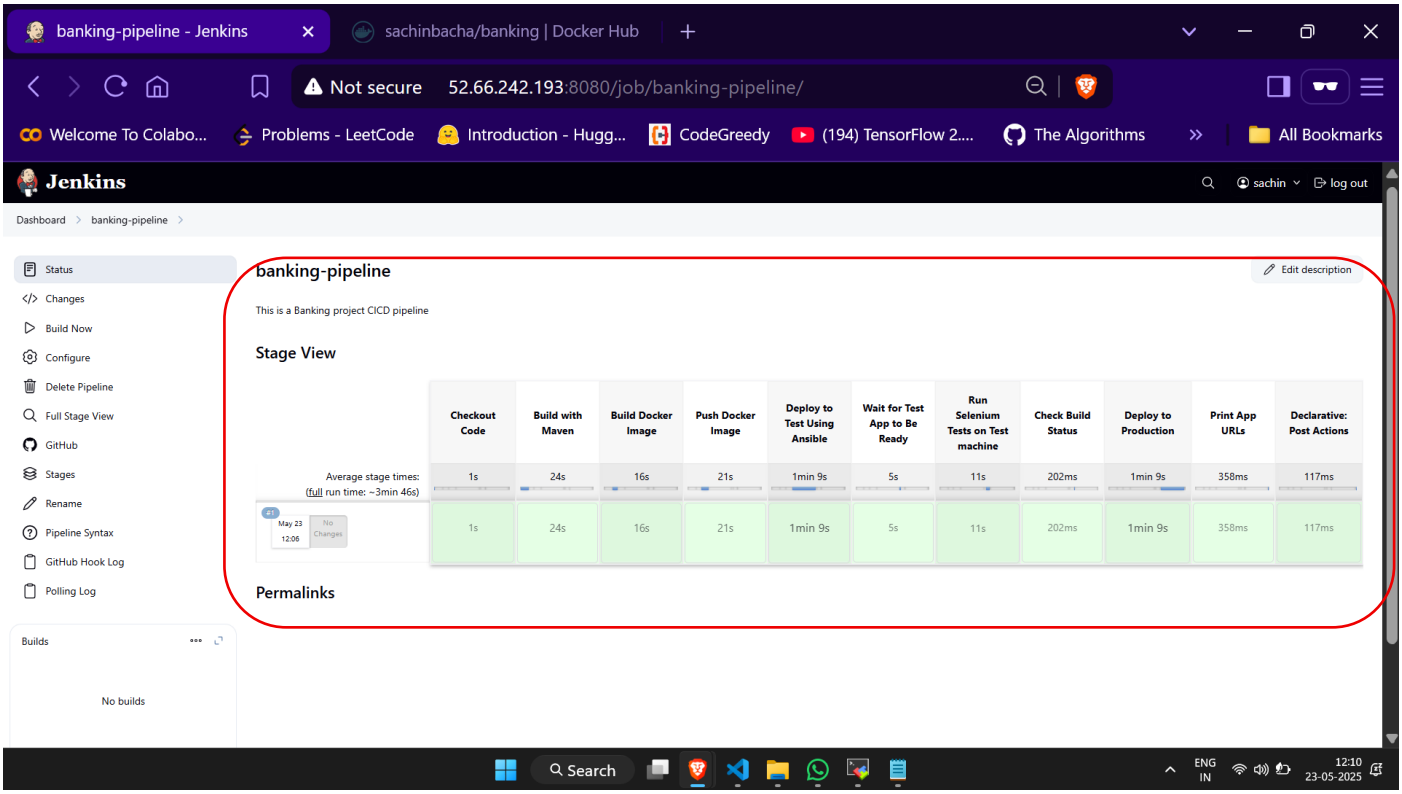
```
    }
   }

    stage('Run Selenium Tests on Test machine') {
      steps {
        sh './mvn test'
      }

  }
    stage('Check Build Status') {
      steps {
          script {
            echo "Current build result: ${currentBuild.result}"
  }
 }
}
    stage('Deploy to Production') {
      when {
        expression { currentBuild.result == null || currentBuild.result == 'SUCCESS' }
      }
      steps {
        ansiblePlaybook credentialsId: 'ansible-ssh', disableHostKeyChecking: true, installation: 'ansible', inventory:
'inventory/prod.ini', playbook: 'ansible-playbook.yml', vaultTmpPath: ''
      }
    }

    stage('Print App URLs') {
      steps {
        sh '''
        echo "👉 Test App:     http://3.109.56.235:8084"
        echo "👉Production App: http://13.203.210.228:8084"
        '''
      }
    }
  }

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

}

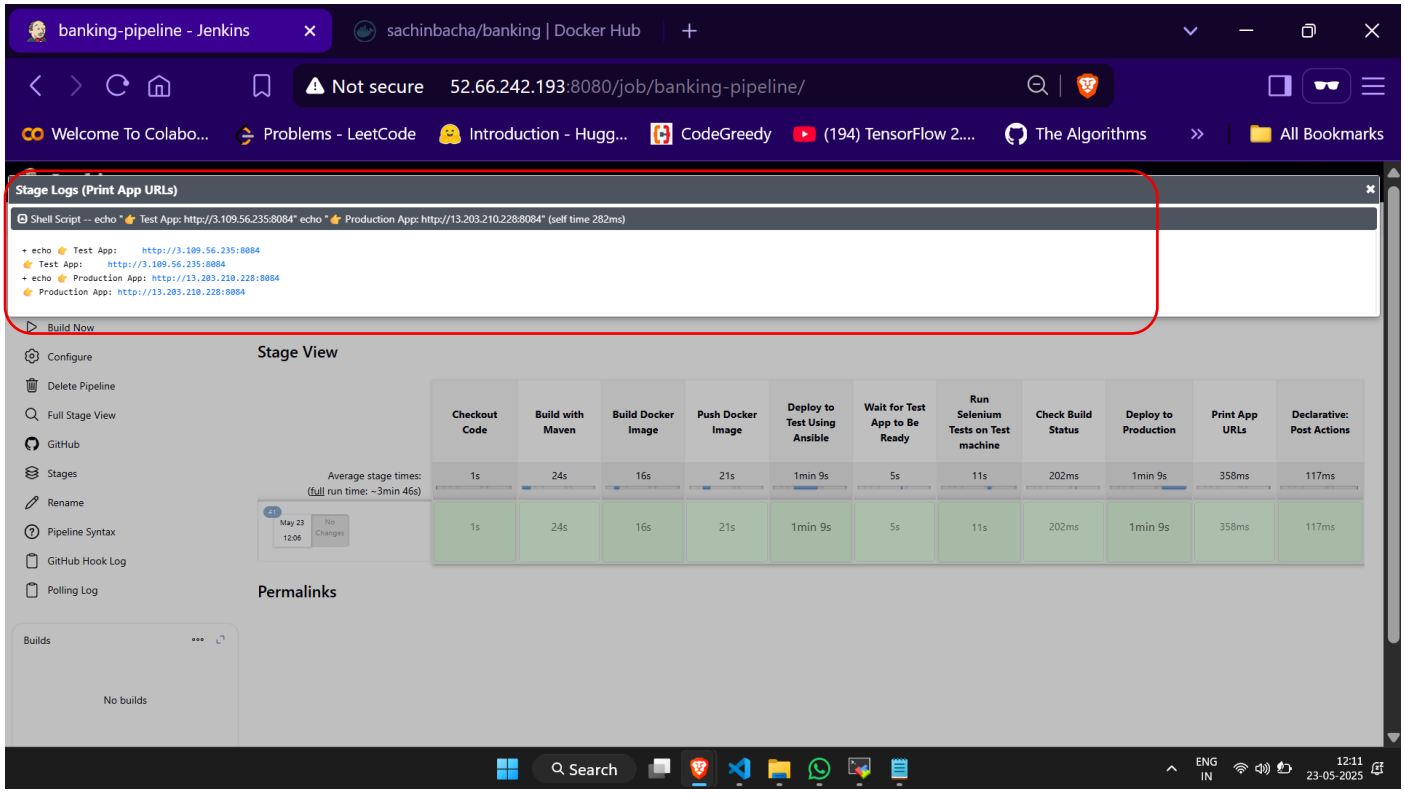
After running the build process . the build will get initiated . Access the docker hub to check if the docker image has been pushed successfully



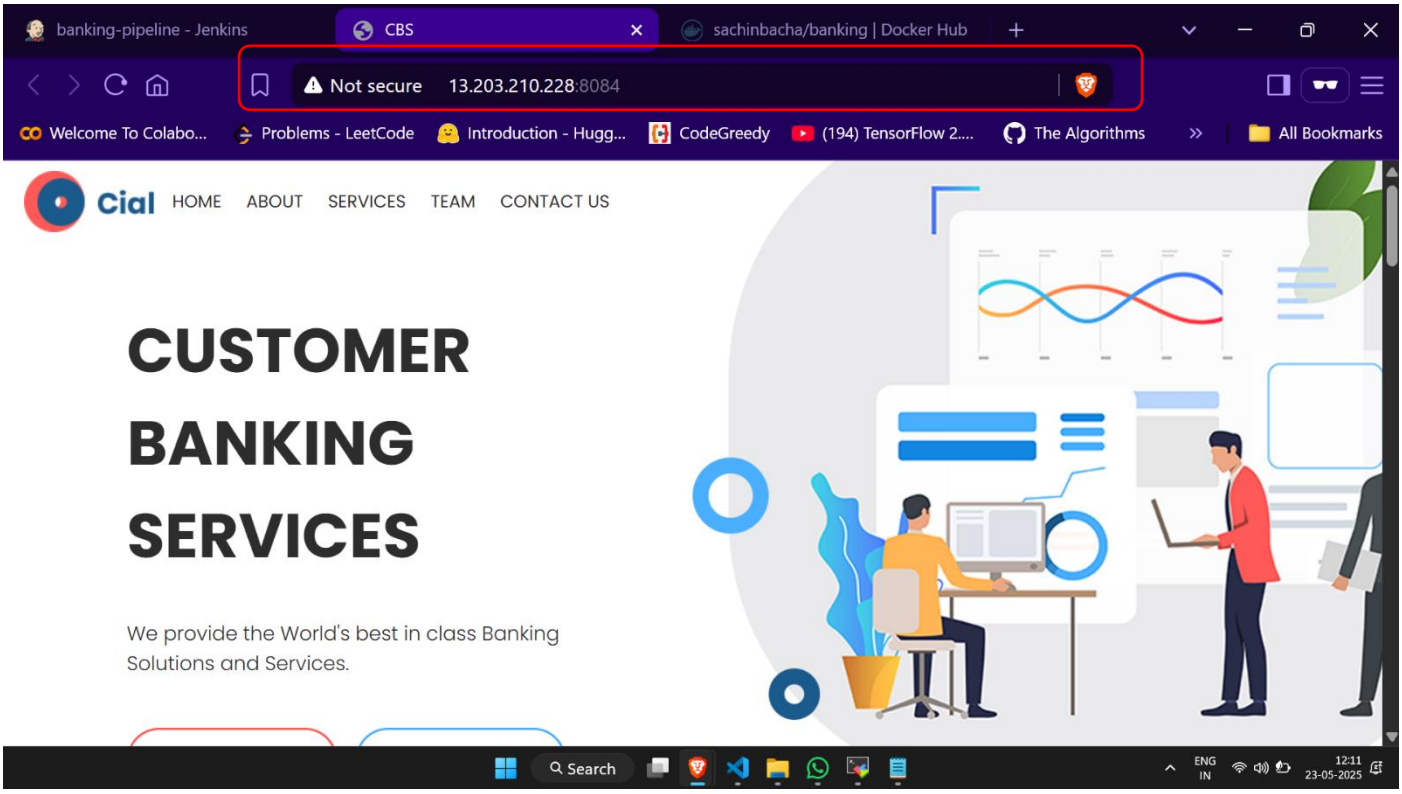
Check the pipeline build status it gets executed



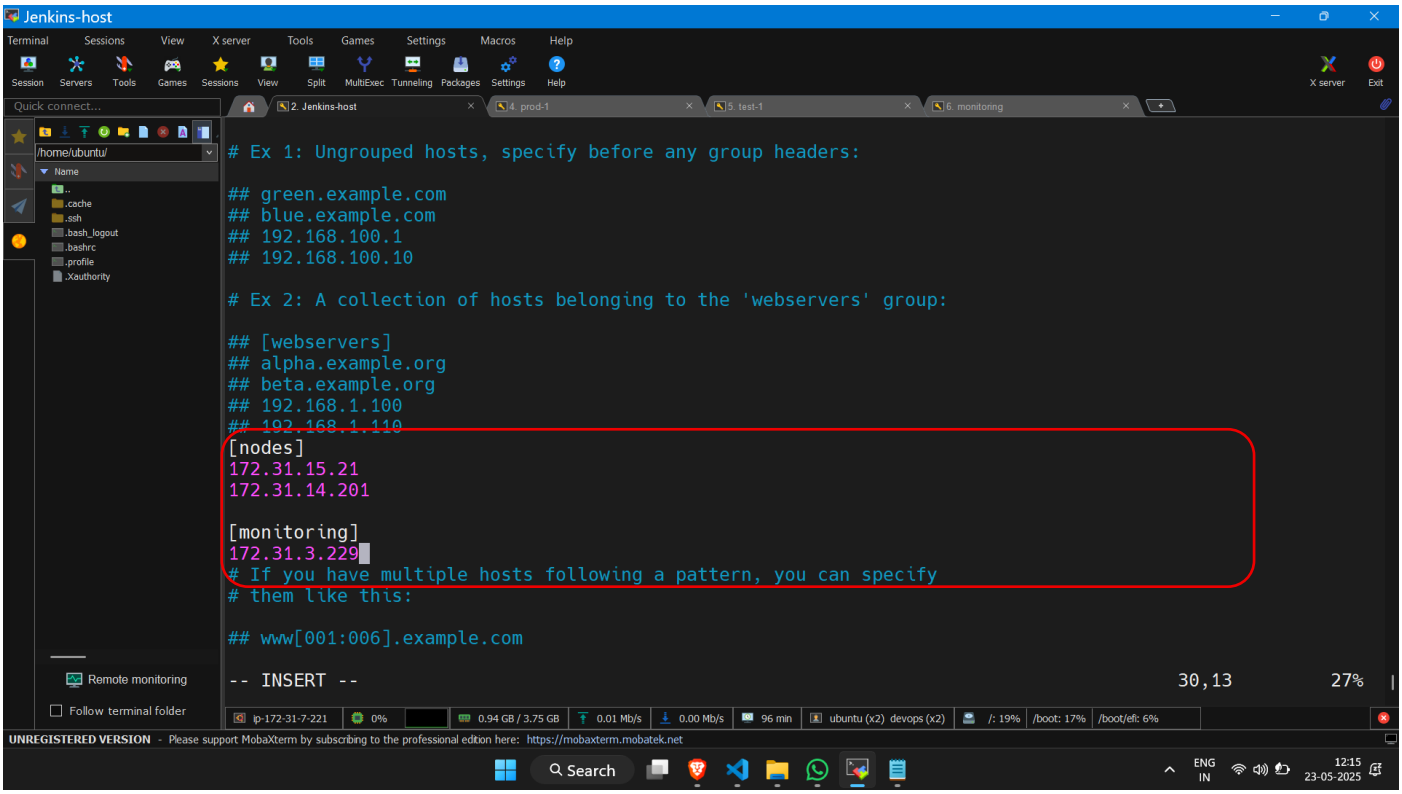
Click on the print app url section logs to get the links



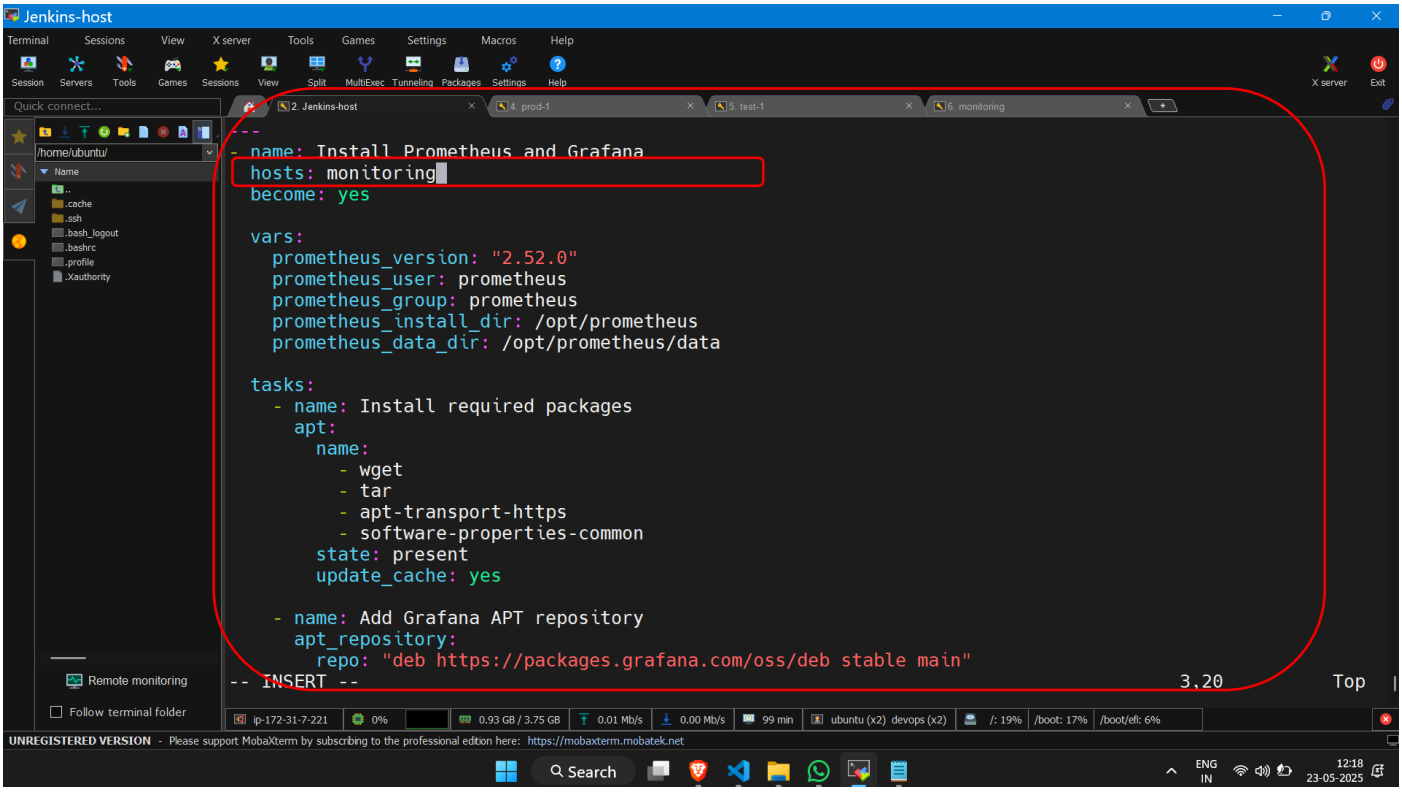
Now after clciking the link your are able to access the application on the machine



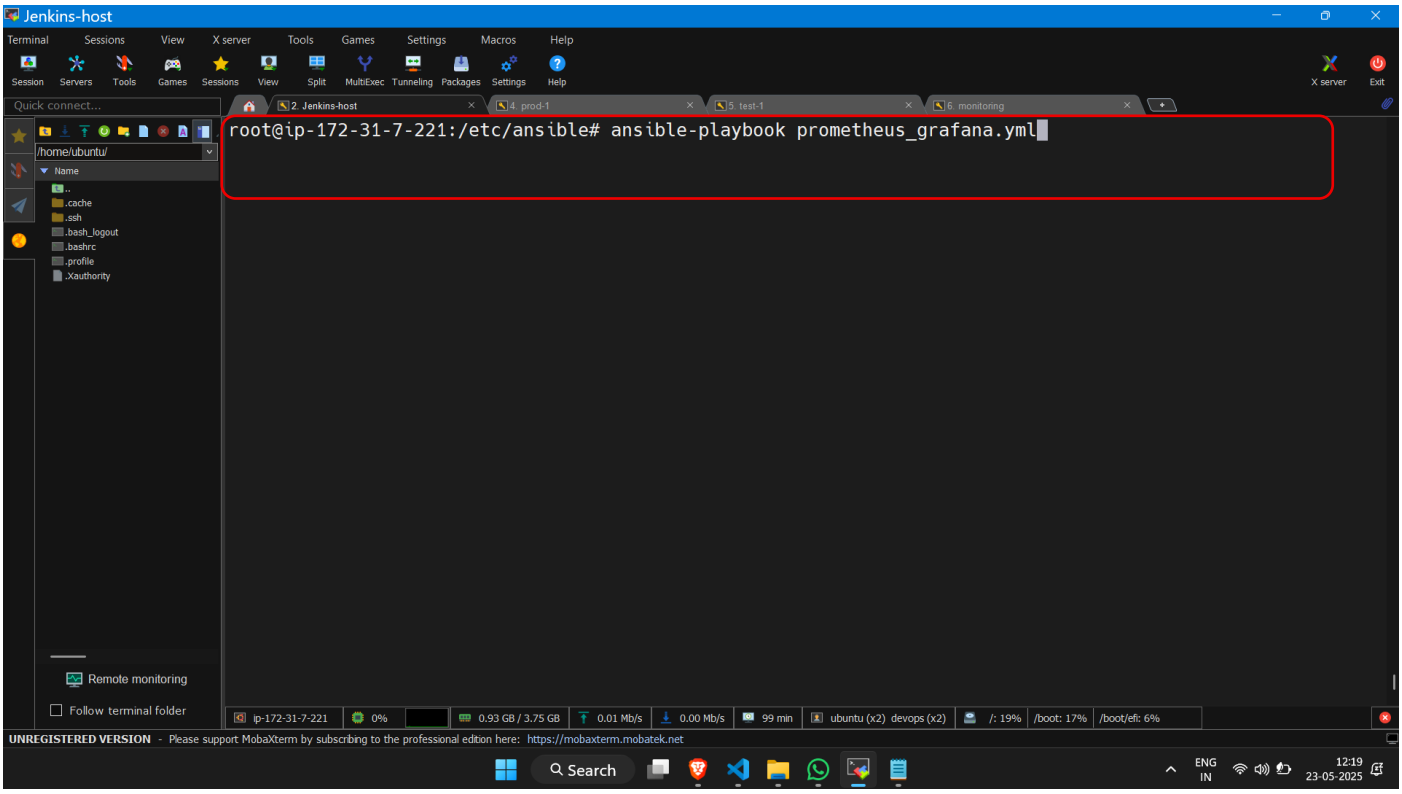
Now also connect the monitoring instance to the ansible host to install Prometheus and Grafana and configure them from the controller node itself . add the private ip of the monitoring node in to the hosts inventory file



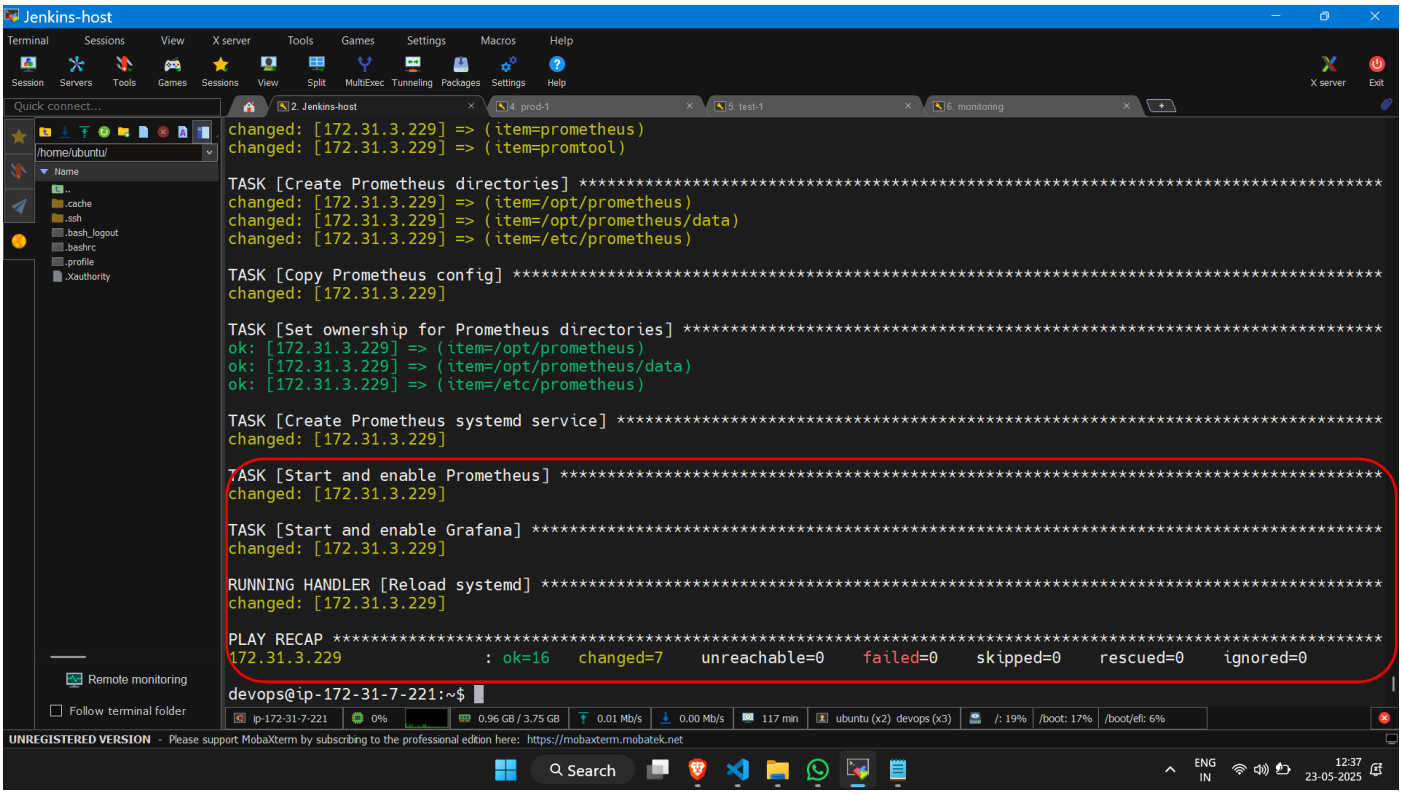
Now write a ansible-playbook to install Prometheus and Grafana into the target machines



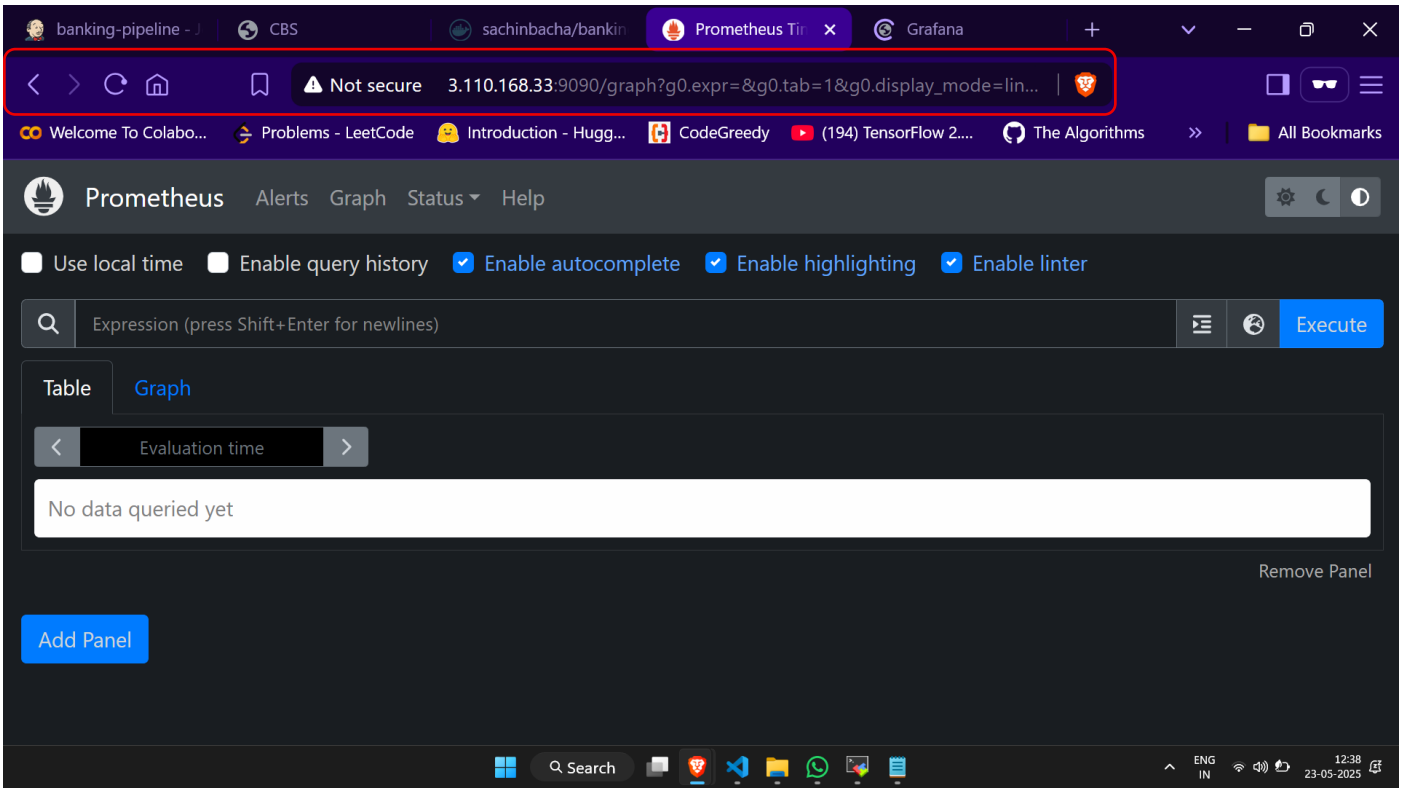
Now run the ansible playbook using the command ansible-playbook <name_of_the_file.yml>



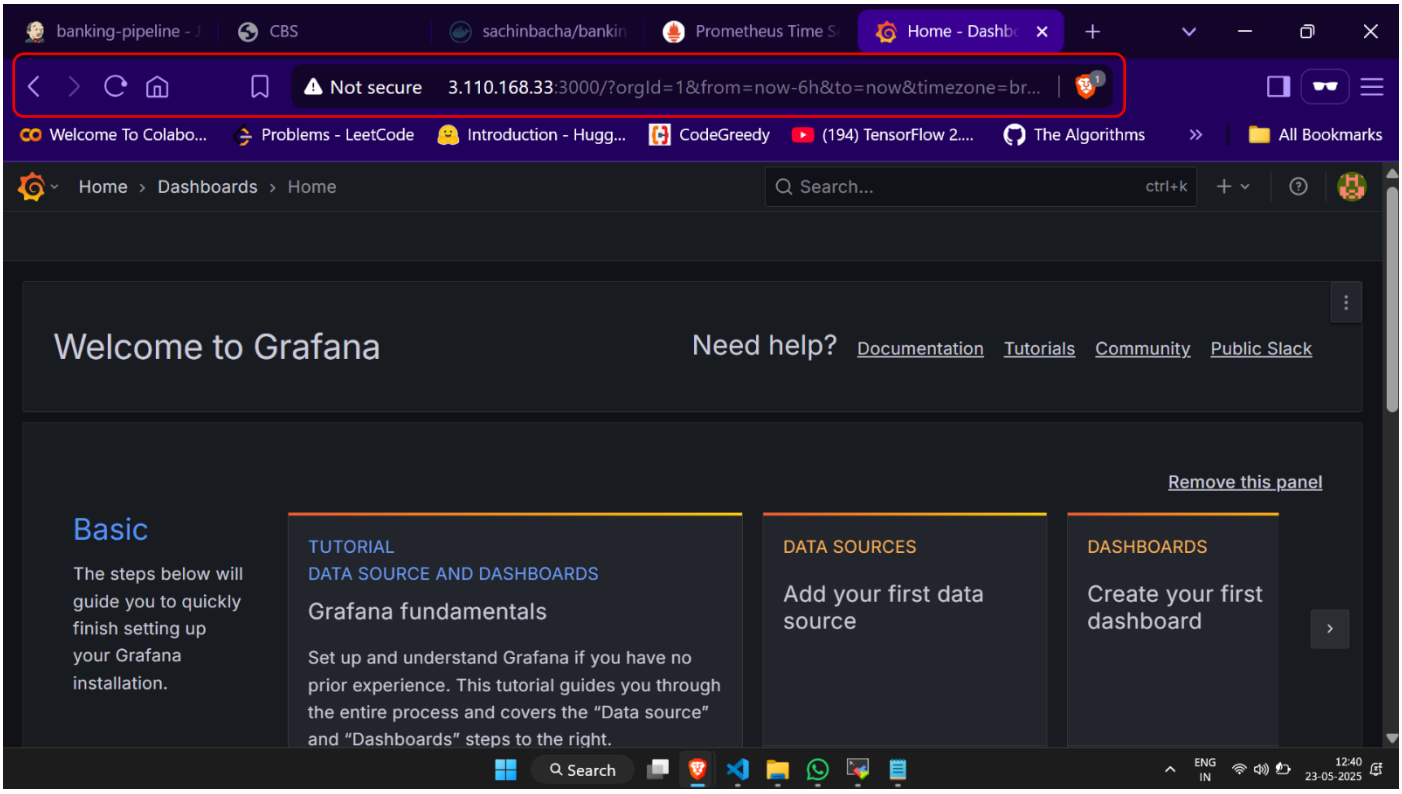
Now Prometheus and Grafana will get installed into the monitoring machine



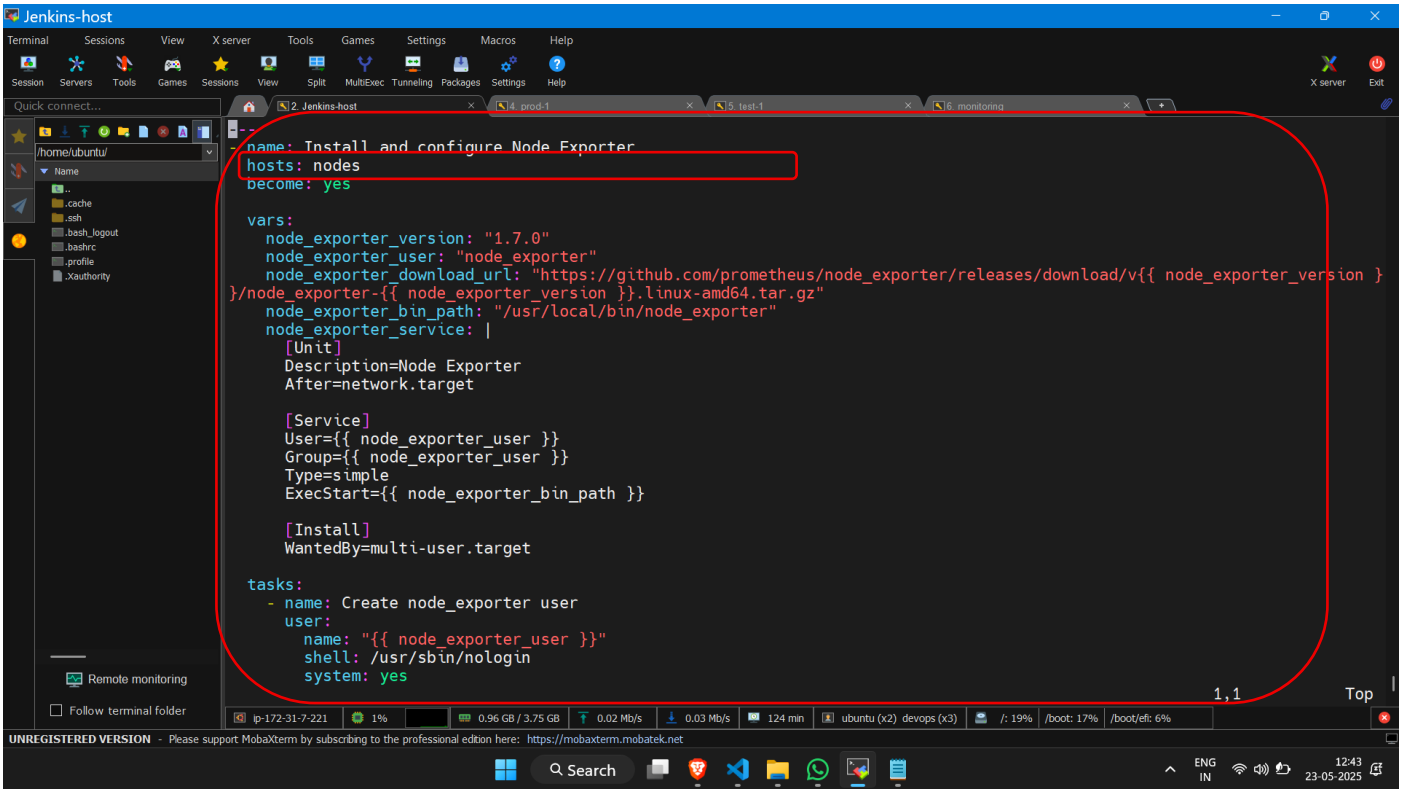
Now access the Prometheus at public ip of the monitoring machine on port 9090 to check if they are installed and running



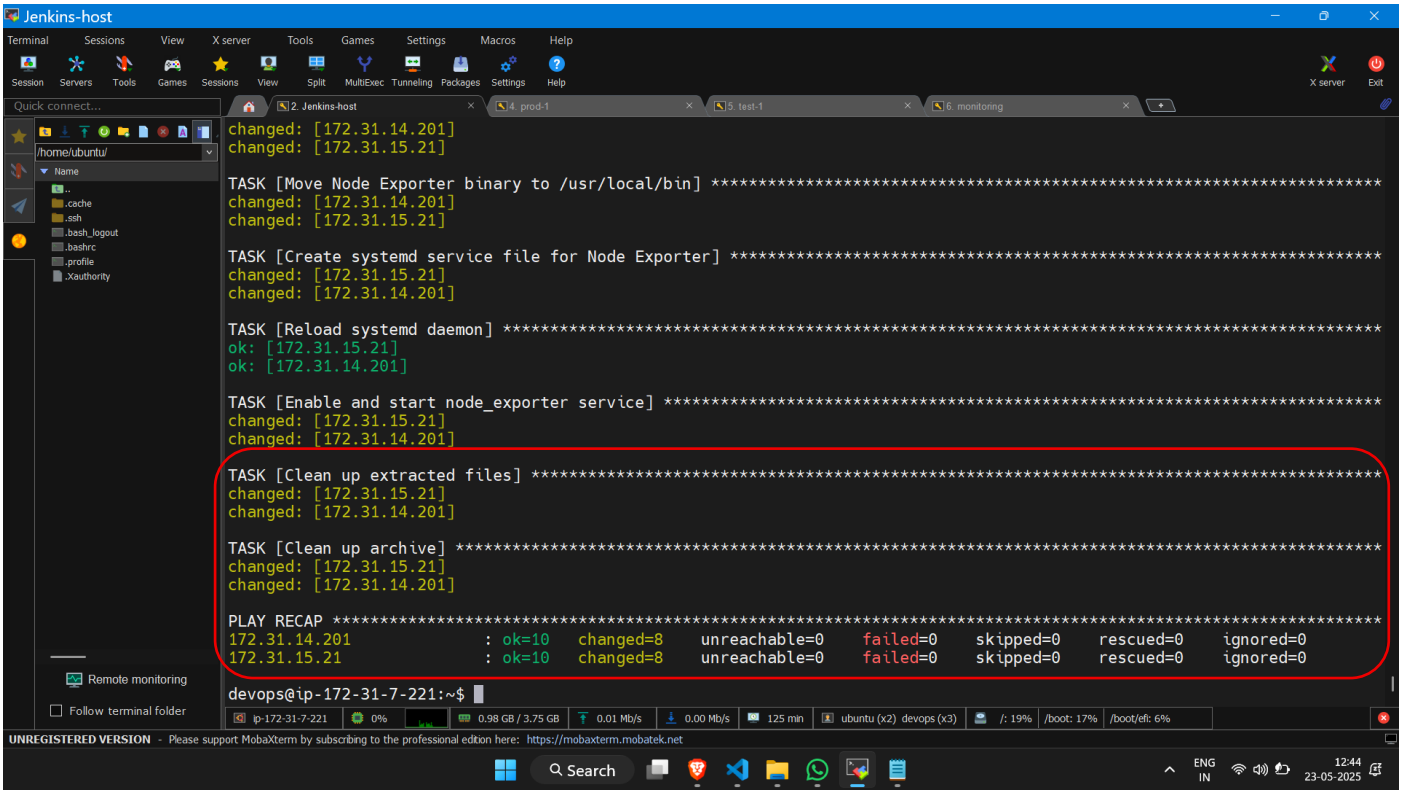
Also access the Grafana on the public ip of the monitoring machine on the port 3000 to check if it is running properly



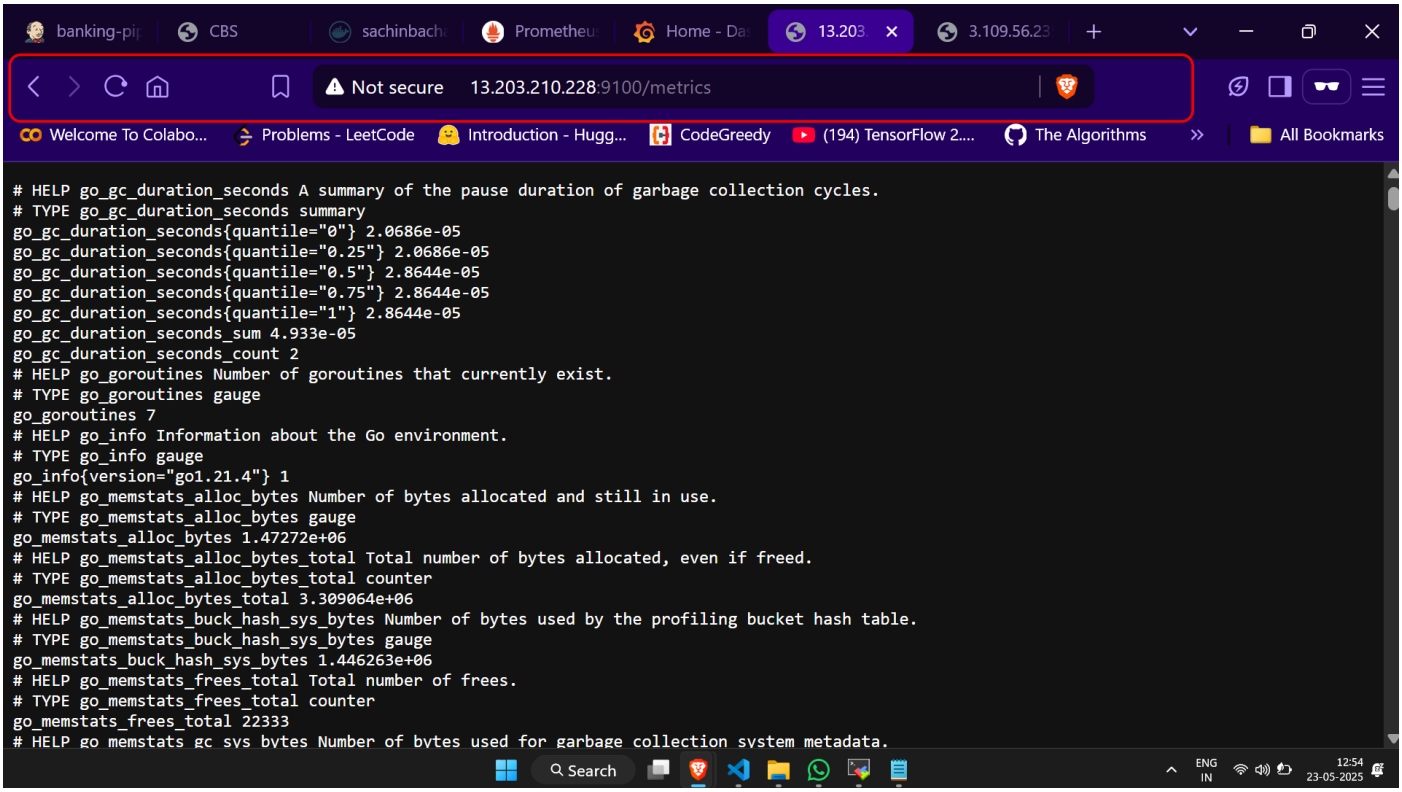
Now write an ansible playbook to install the node exporter into the production and test machines and run them . under the hosts mention the target group as nodes which has been given in the hosts inventory file of ansible . So it will install the node exporter into the only those target machine

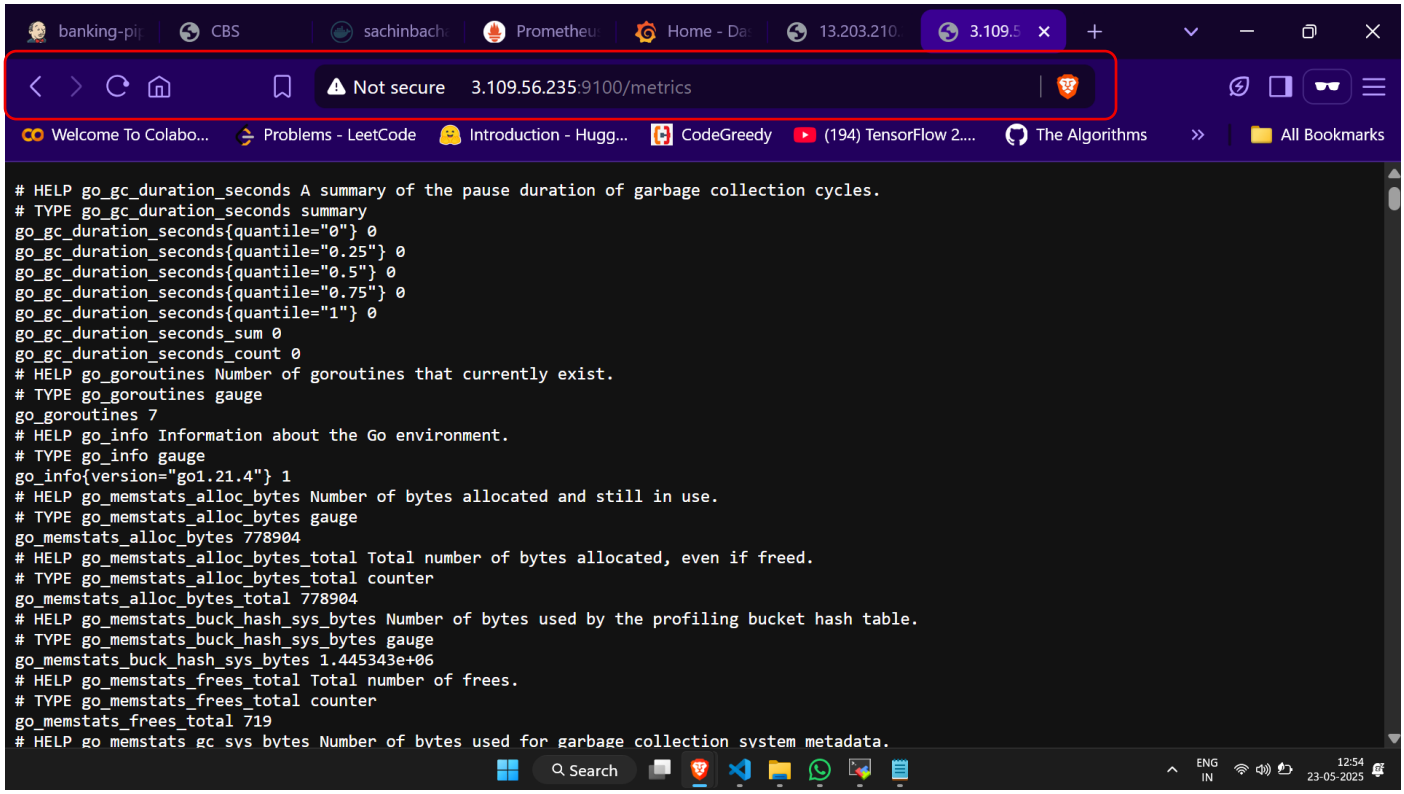


Now run the playbook so the node exporter will get installed into the machines at a time this will reduce installation and setup of node exporter on both the machines.

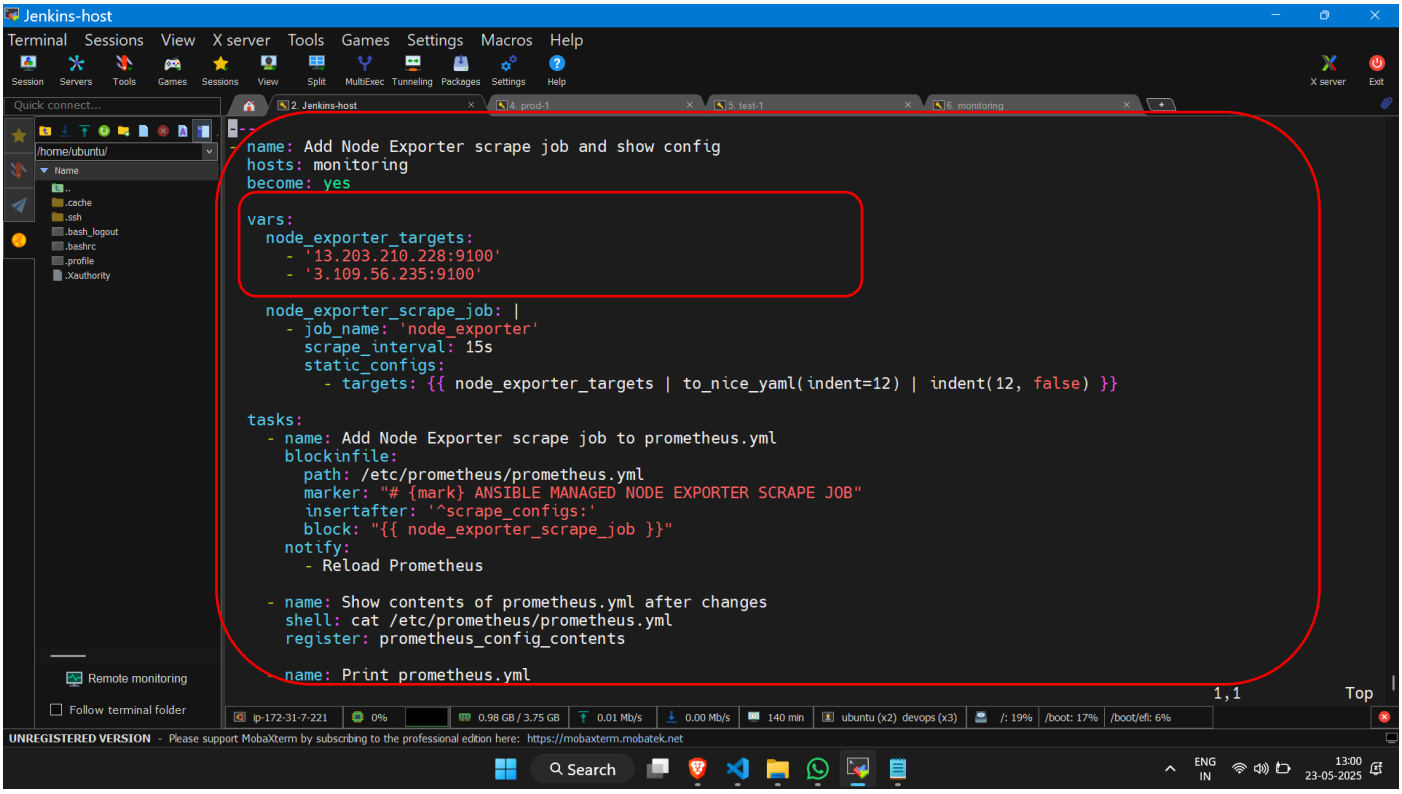


Now verify if the node exporter is installed into the node properly by accessing the public ip on port 9100 on route metrics

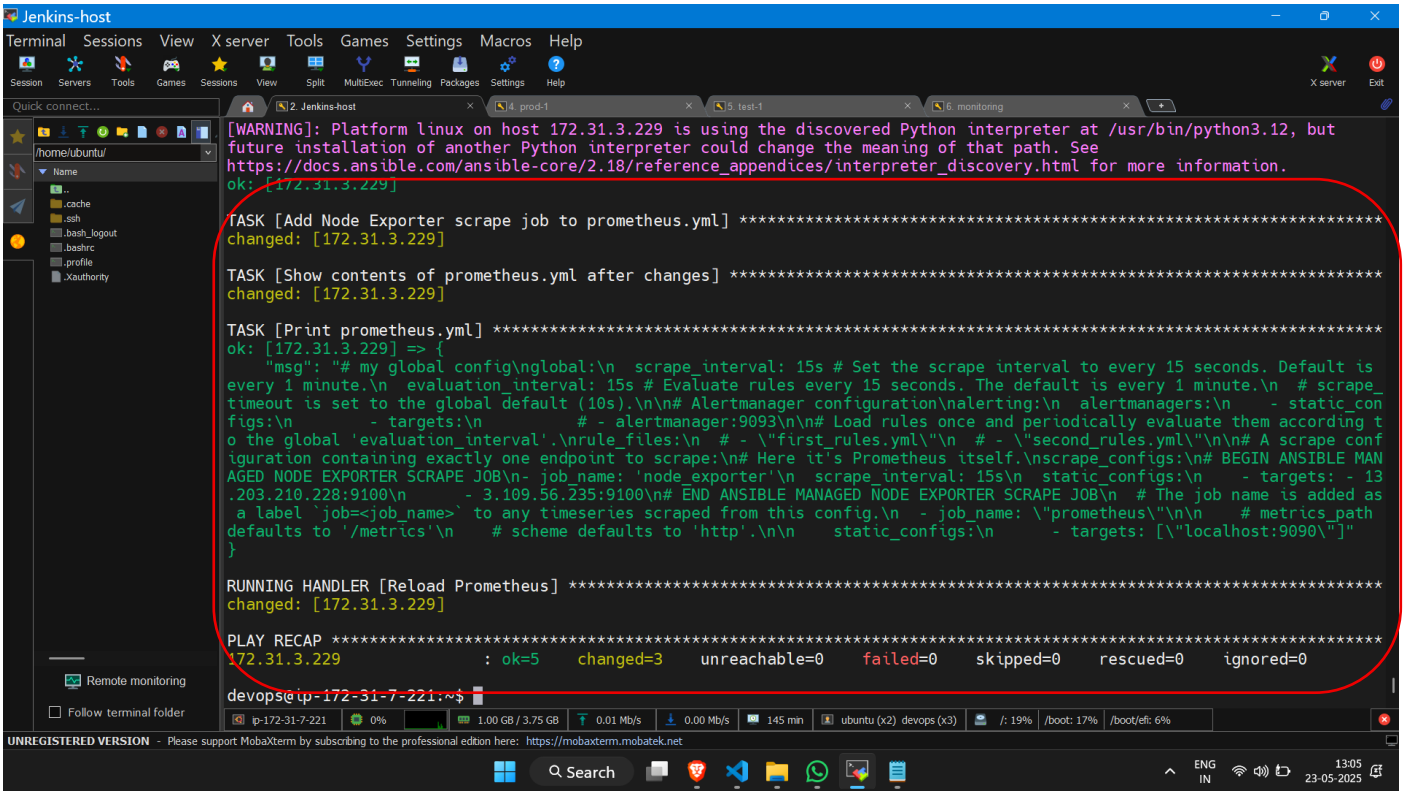




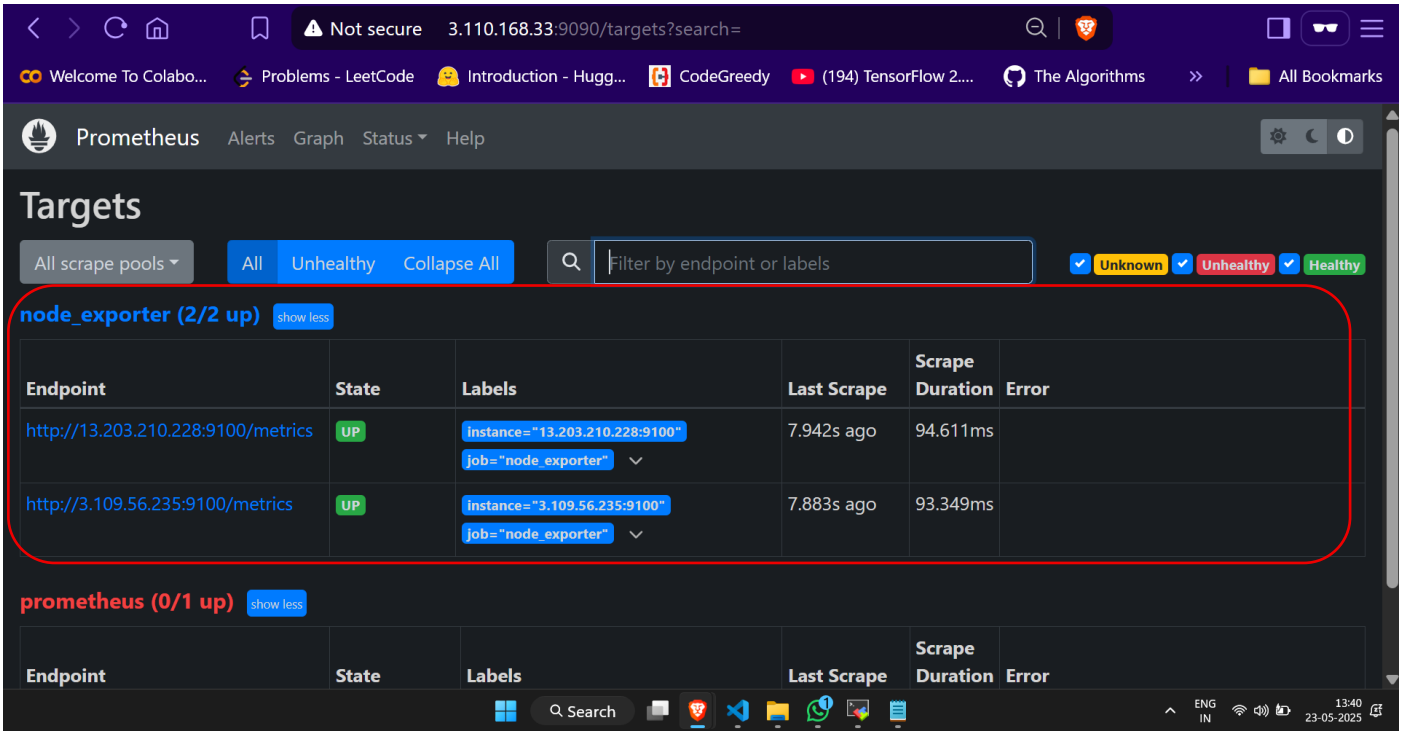
Now Configure the ansible playbook to configure the Prometheus.yml file with the target nodes to be added on to the file



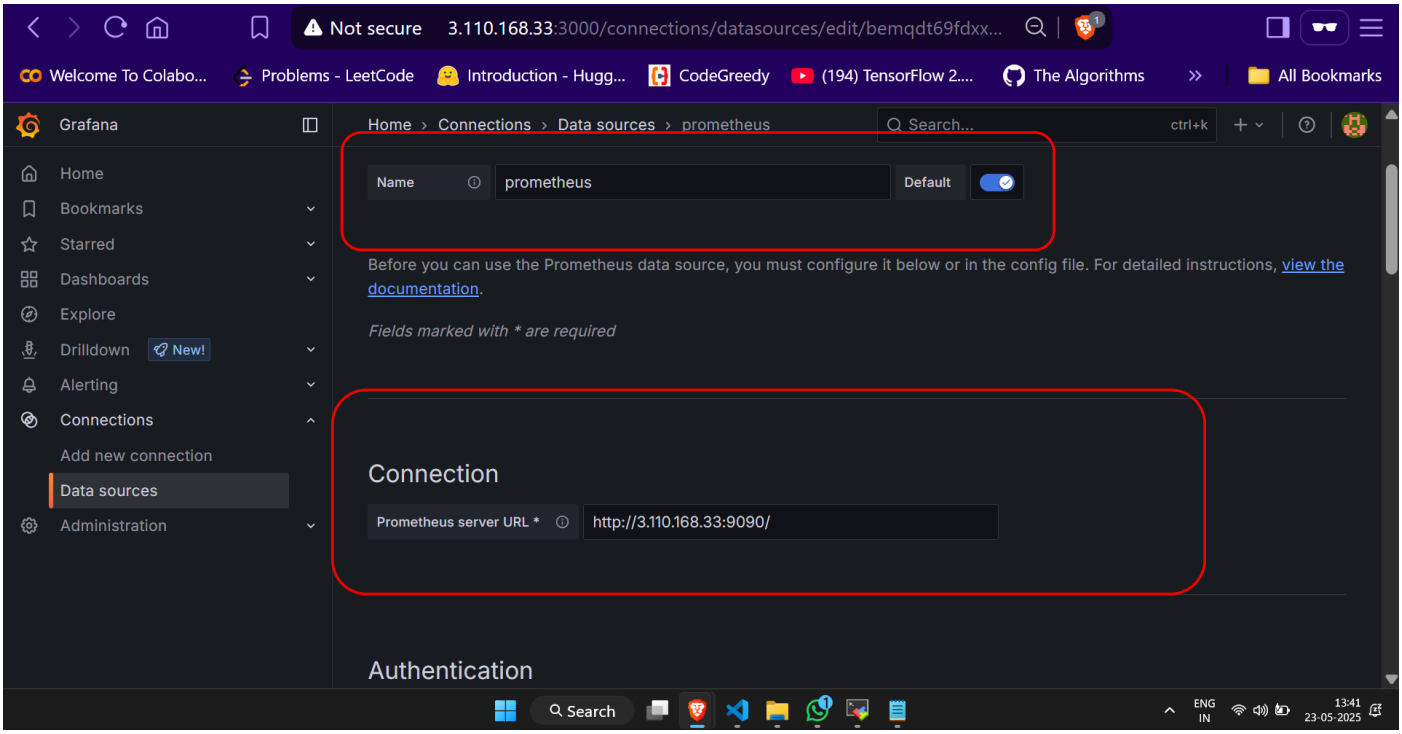
Run the playbook and the Prometheus file gets added with the ip address of the node_exporter ip's



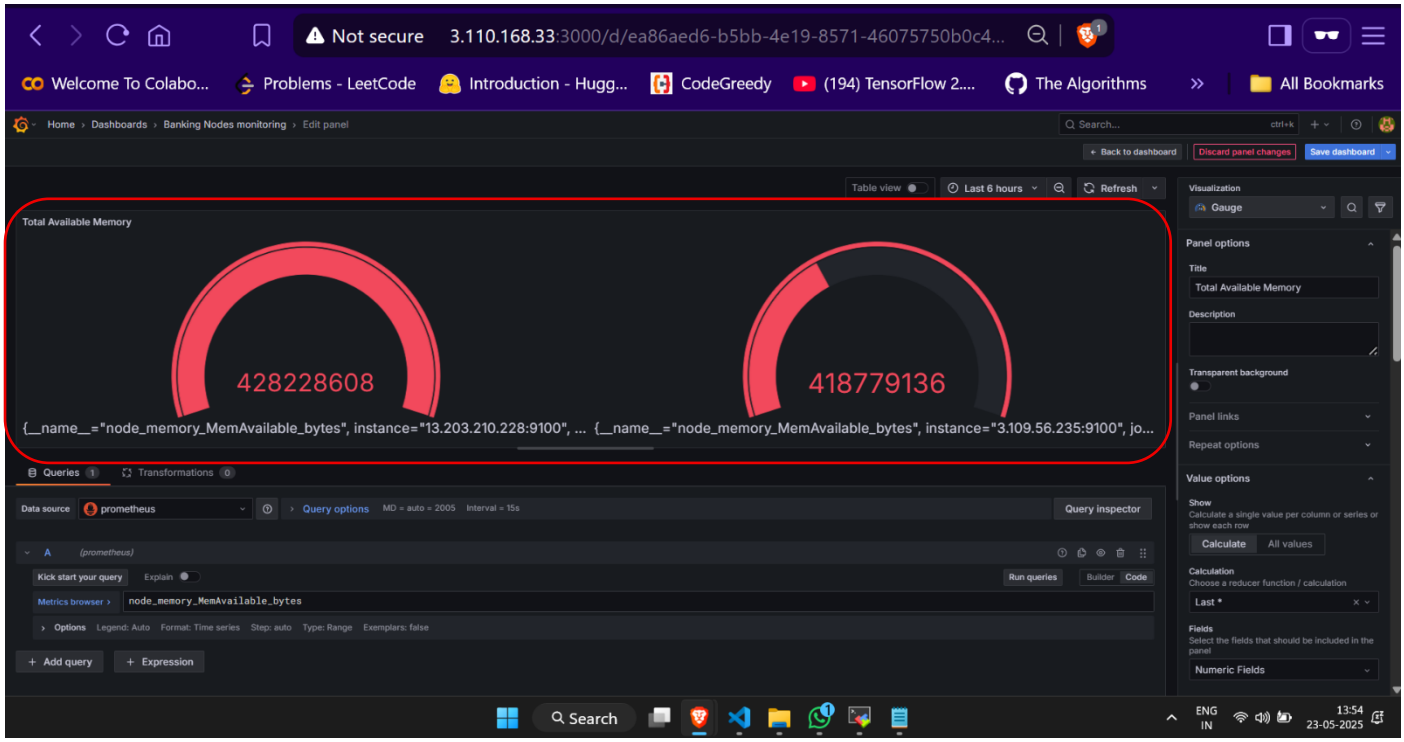
Now check the Prometheus targets and verify if the metrics nodes are added succesfully



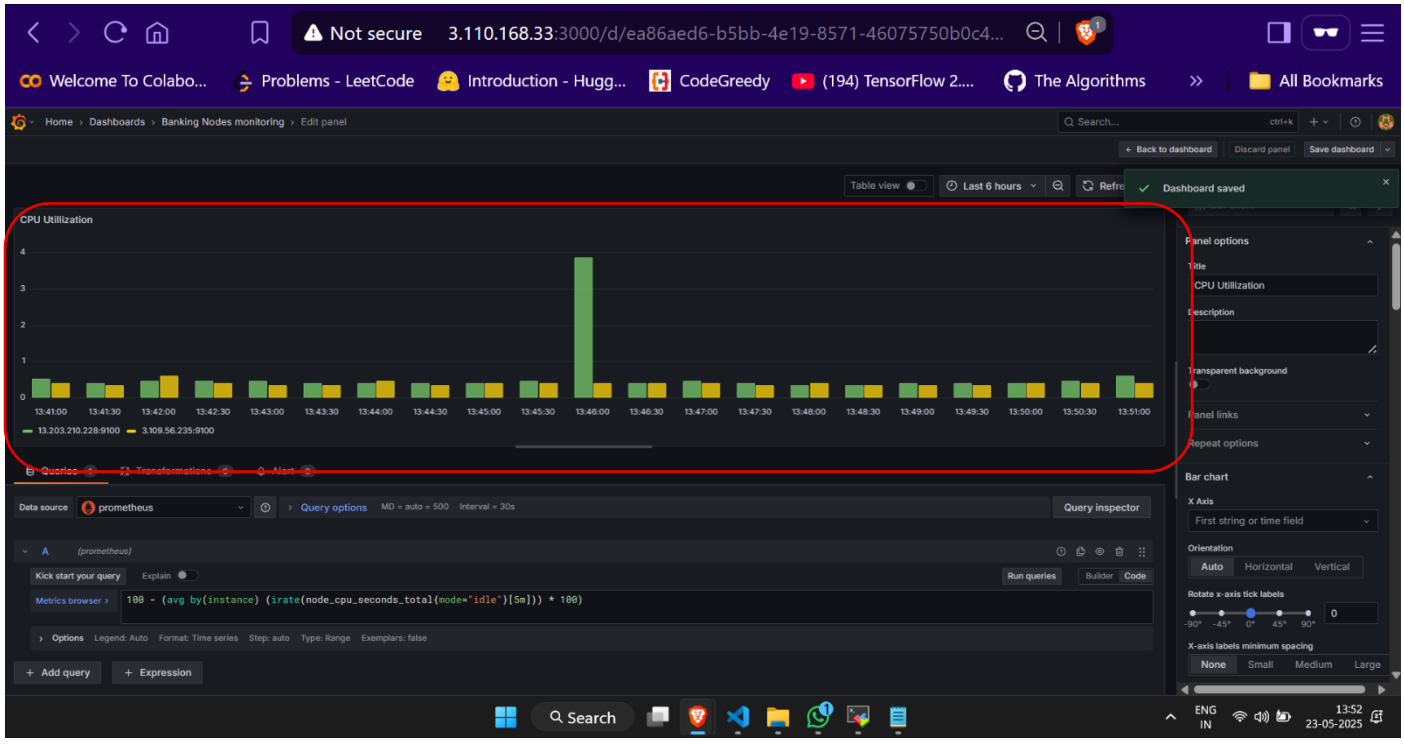
Now connect the data source of Prometheus into Grafana on the ipaddress on port 9090 and add the data source Now we are good to go with the monitoring visualizations .



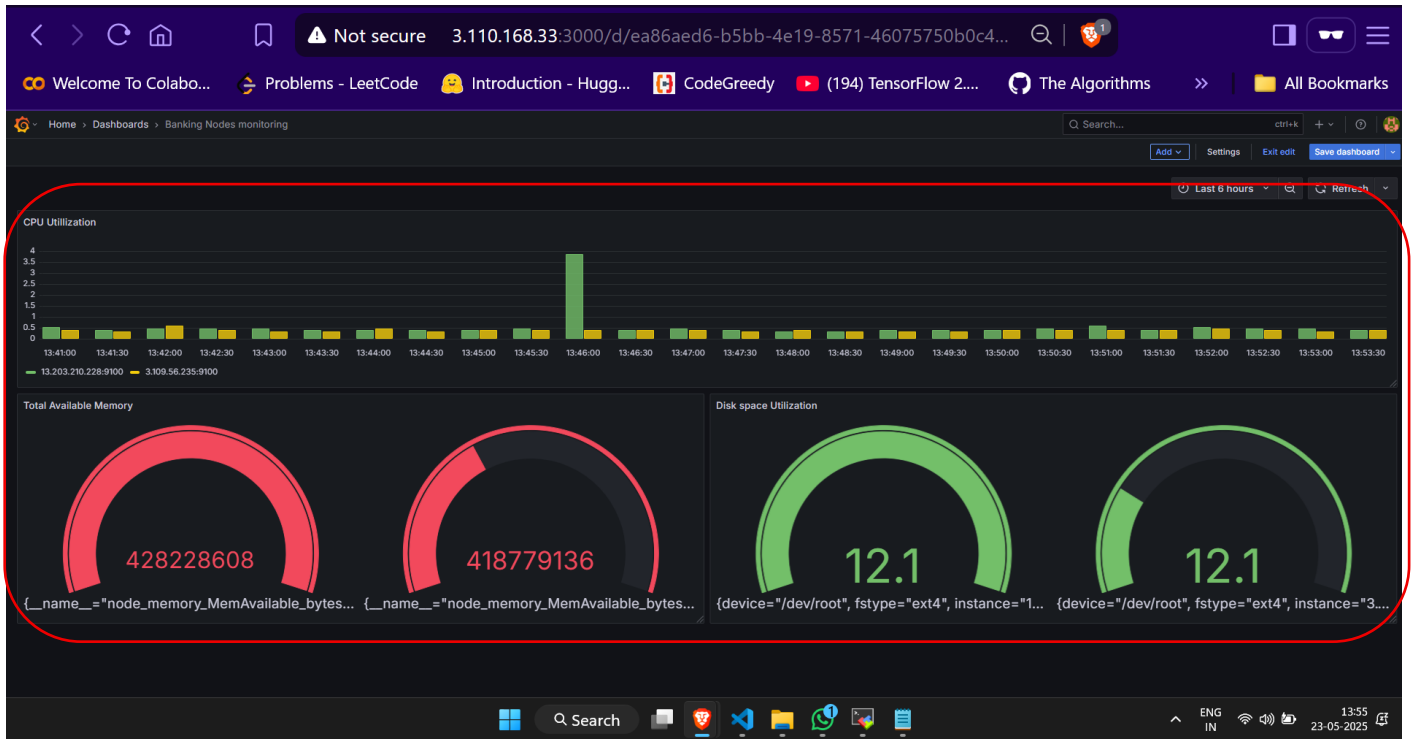
Now create new visualization in the Grafana and create a panel for monitoring the total memory available for both the nodes



Now create another panel for CPU utilization



Create a comprehensive dashboard to monitor CPU Utilization, Total available memory , and Disk space utilization



Also add the github webhook trigger to trigger for any changes in the repository on the Jenkins ipadress