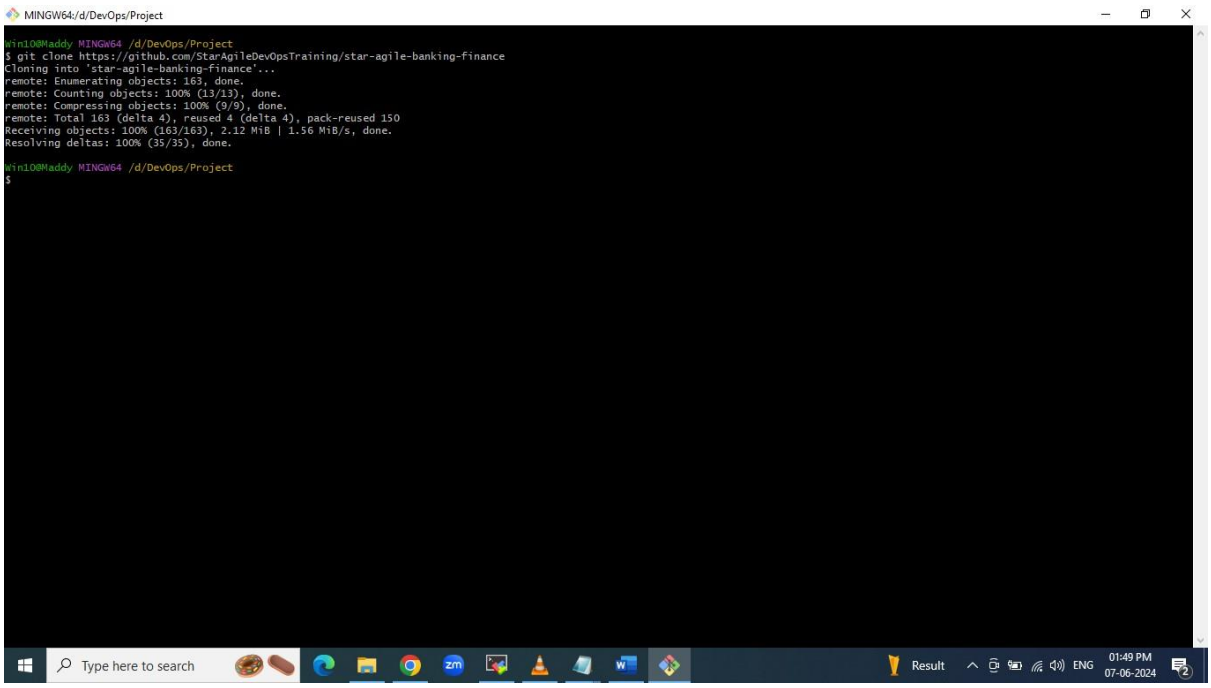
Name: Sagar Kulkarni

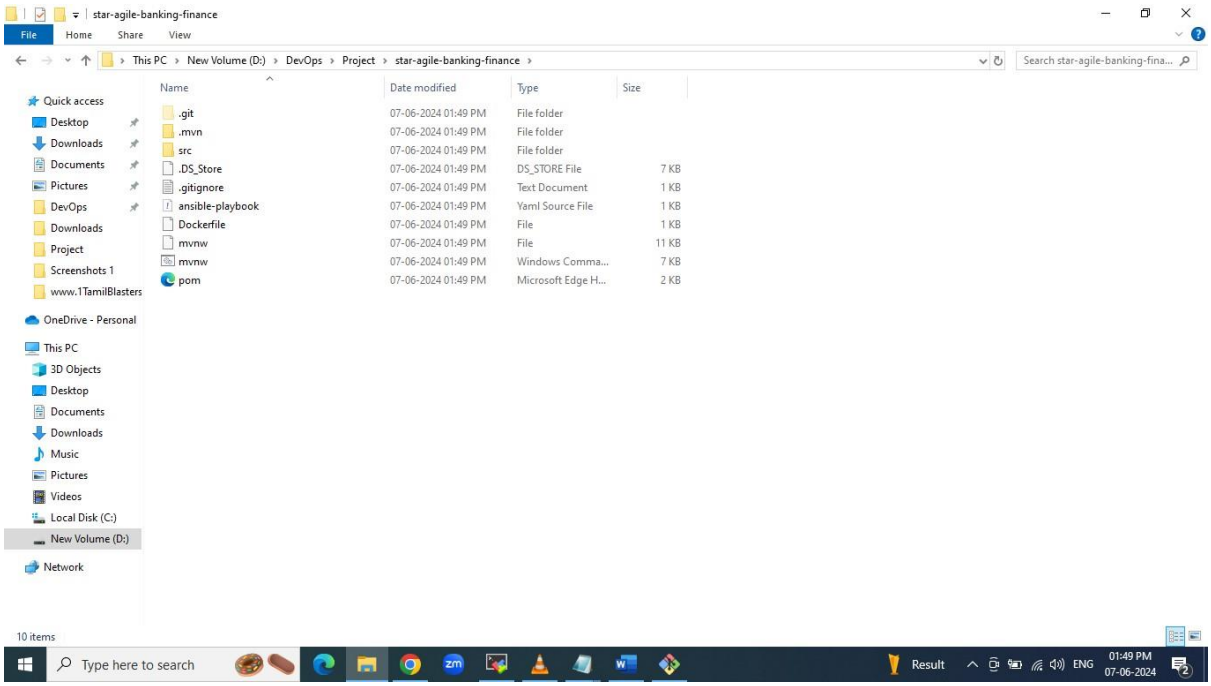
Batch: DevOps - SA2401021

Banking and Finance Domain Project

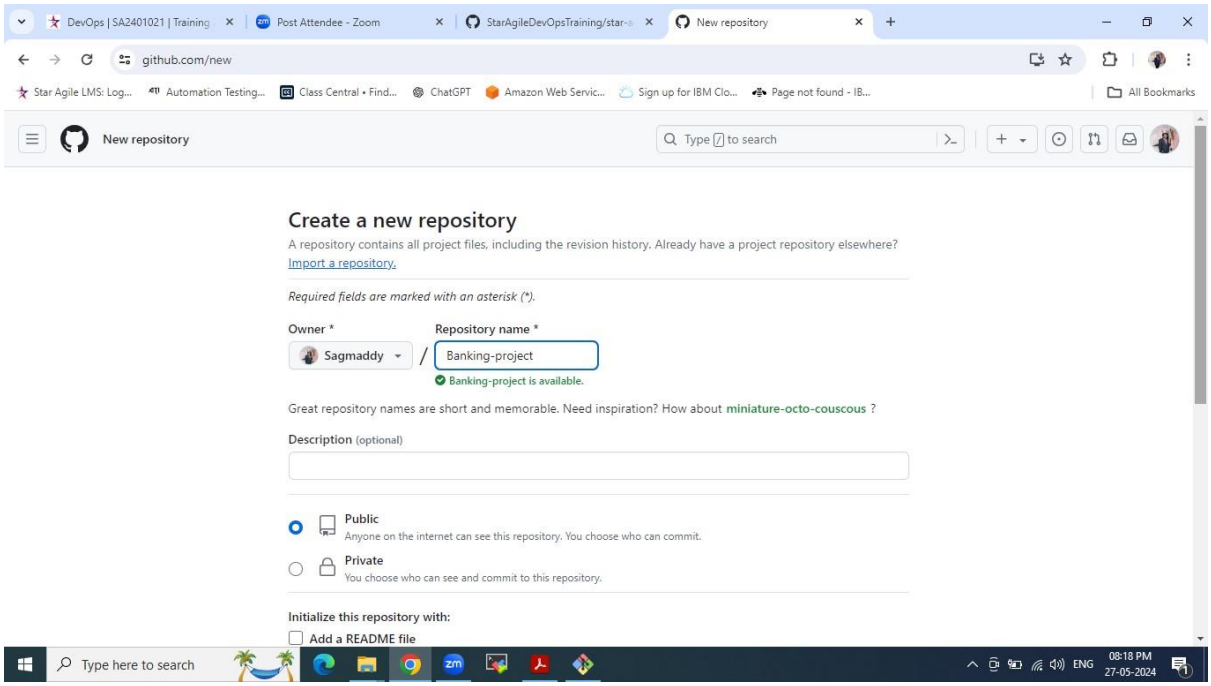
The "FinanceMe" project is a comprehensive initiative aimed at modernizing the application architecture of FinanceMe, a leading global provider of banking and financial services based in Germany. Originally utilizing a monolithic application structure, FinanceMe faced significant challenges in managing infrastructure, deployments, and scalability as the company expanded. To address these issues, the project transitioned to a microservice architecture with integrated DevOps practices, leveraging AWS for cloud services. Key objectives included enhancing software delivery speed, improving quality and reliability, and reducing feedback time between development and testing. The project involved developing a microservice using Spring Boot and an in-memory H2 database, exposing multiple API endpoints, implementing CI/CD pipelines with tools such as Git, Jenkins, Docker, Ansible, Selenium, Terraform, Prometheus, and Grafana, and ensuring automated monitoring and reporting for continuous integration and deployment.



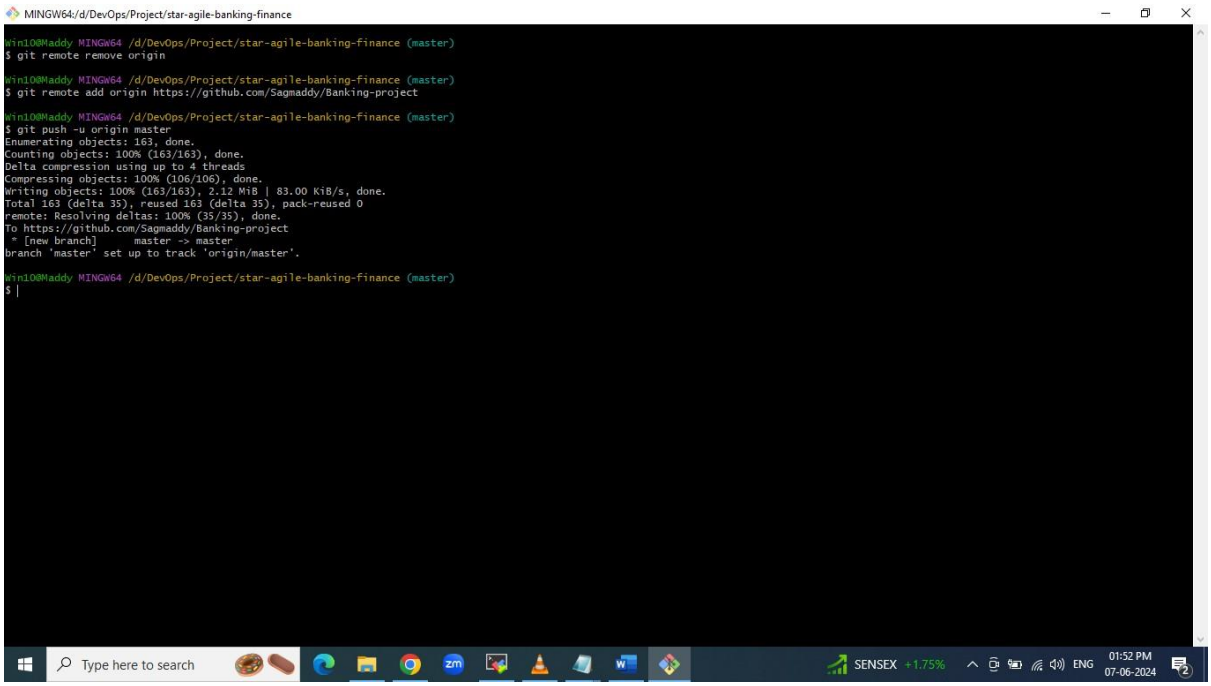
Clone the finance me project into local system using git clone command.



Repository is cloned into local system.

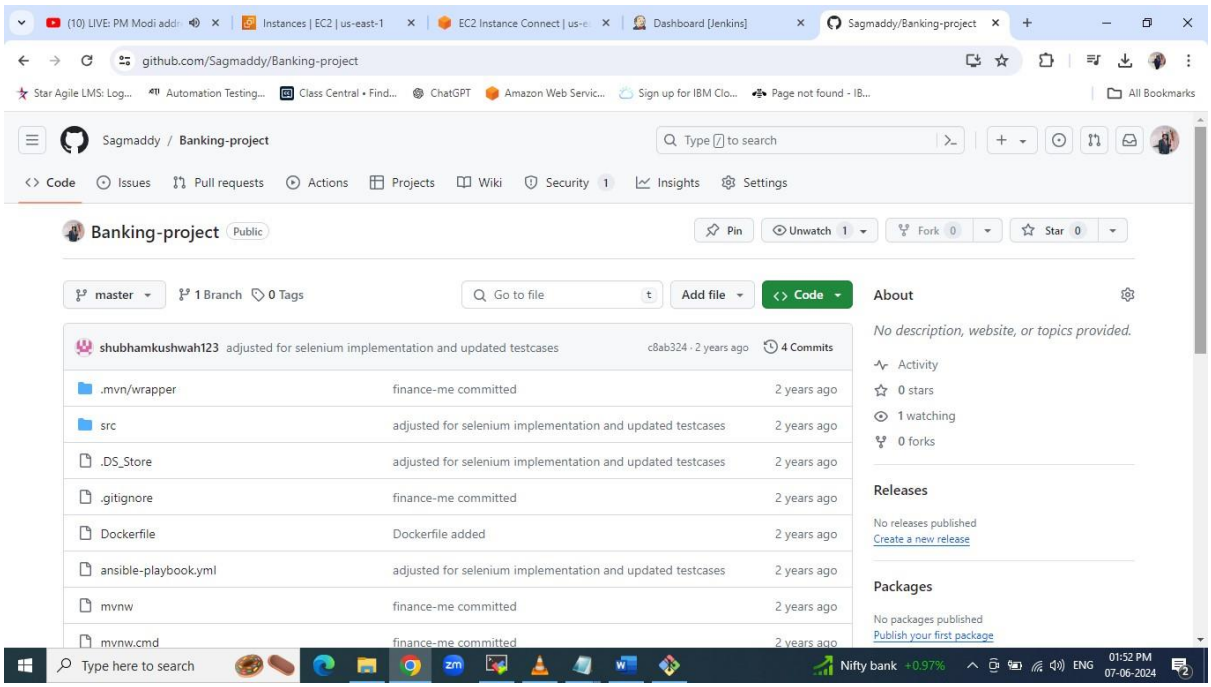


Created a new repo Banking-project



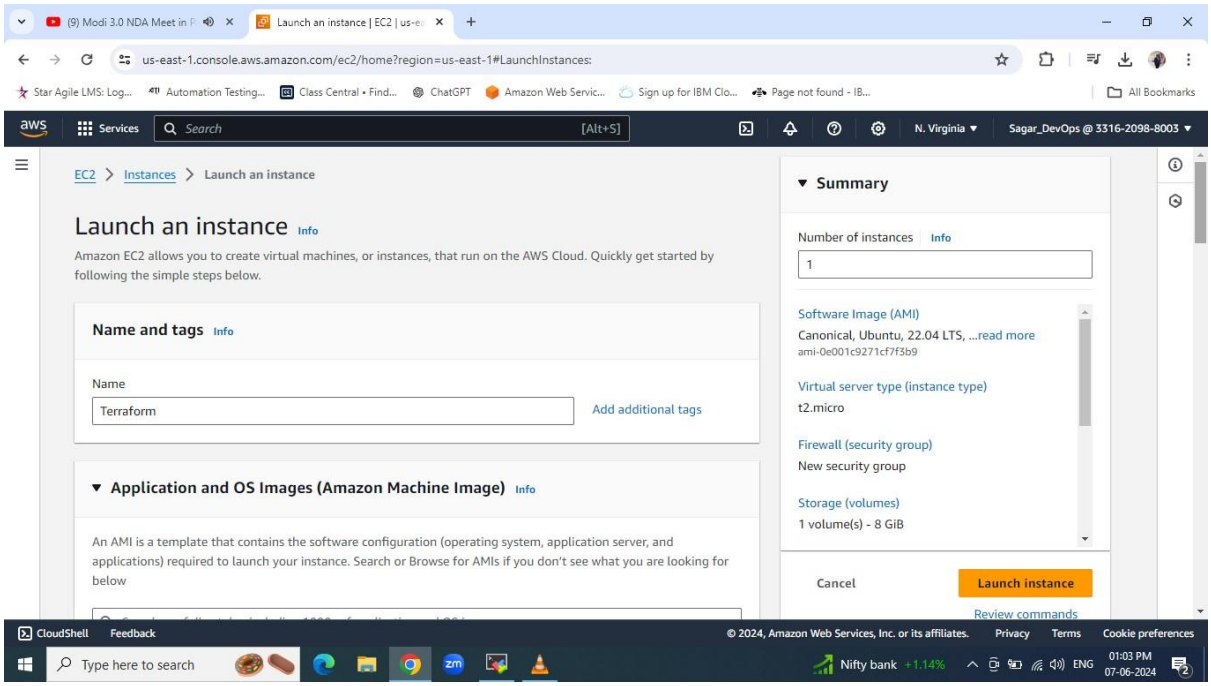
Removed the previous origin using git remote remove origin and added my repo as origin using git remote add origin and my repo URL.

Pushed the files into my repo using git push -u origin master.

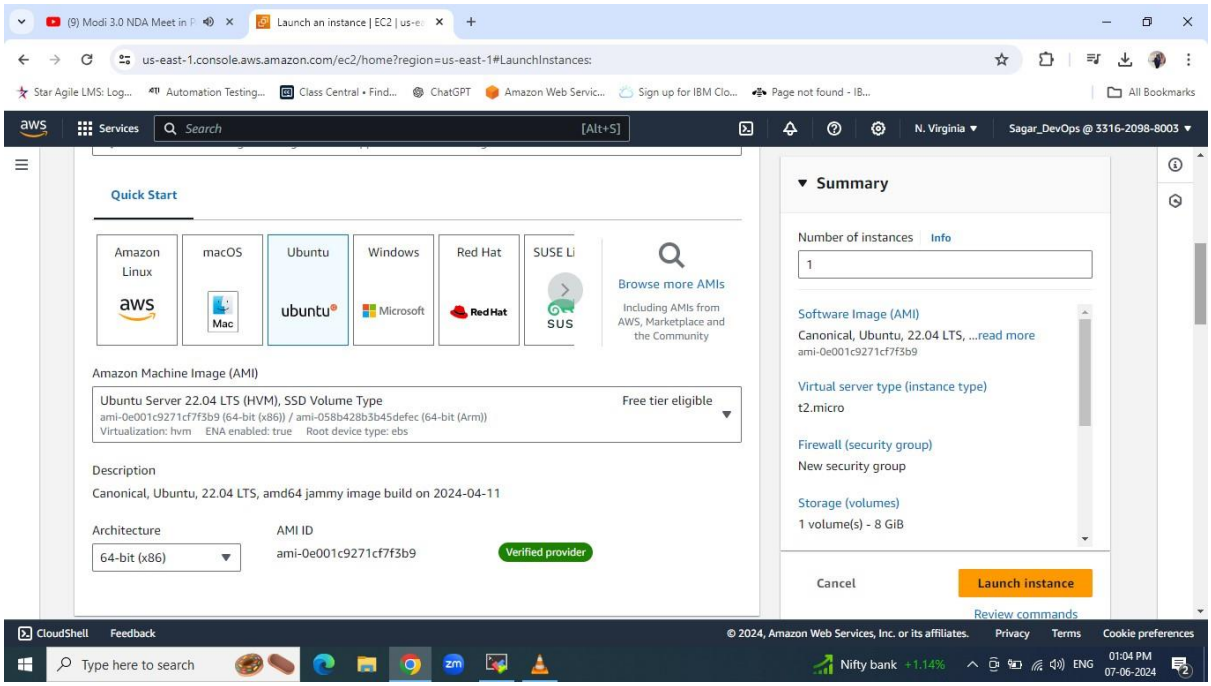


all the files are added into my repo Banking-project.

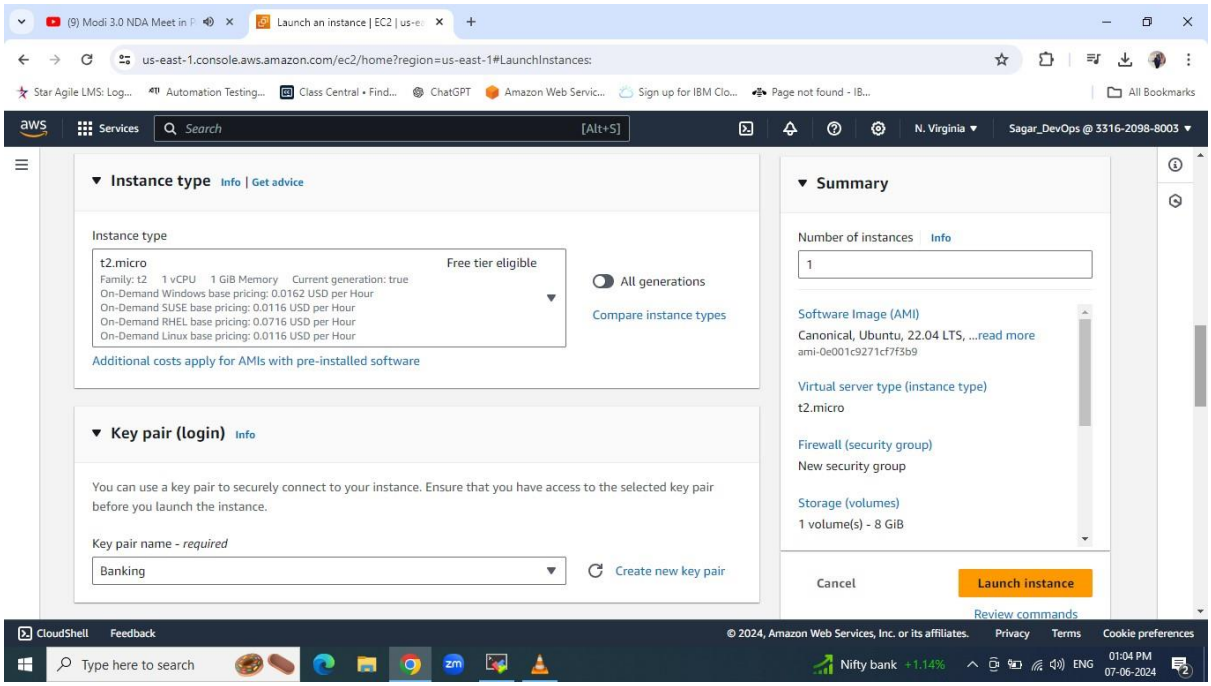
Launched the t2.micro instance and installed terraform to create the infrastructure for this project. Created two t2.medium instances for Jenkins, Docker, ansible and ansible-host (to host the application).



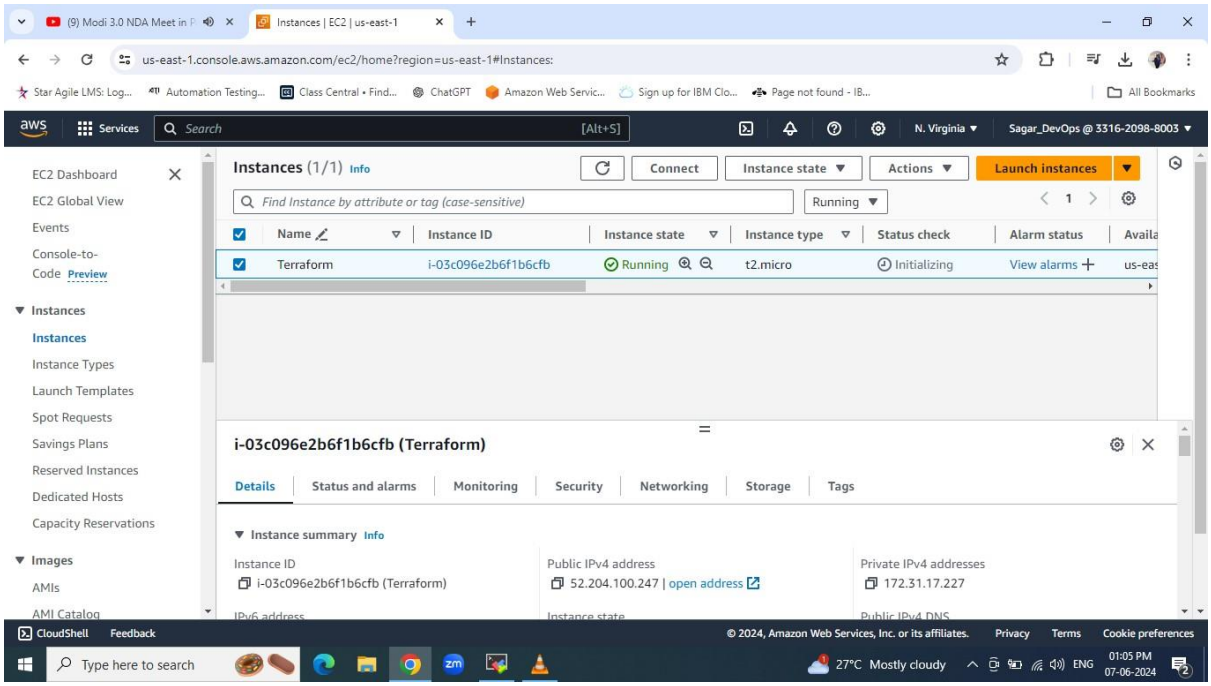
Named the instance as Terraform.



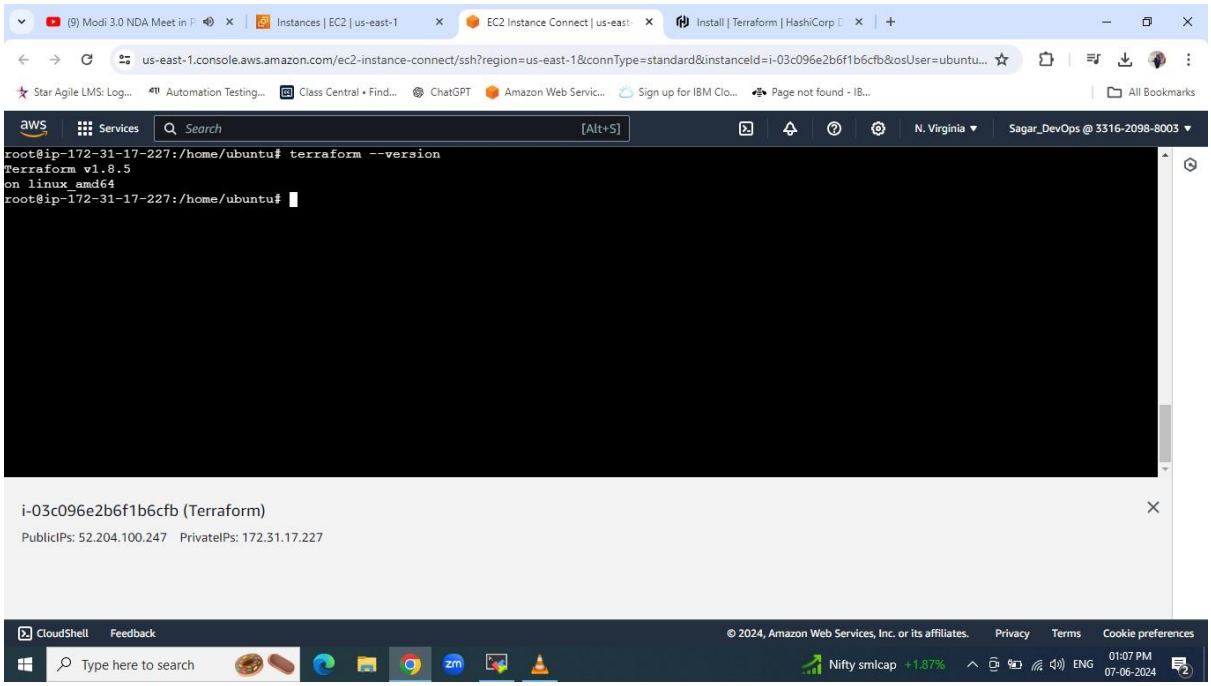
Selected ubuntu 22.04 server.



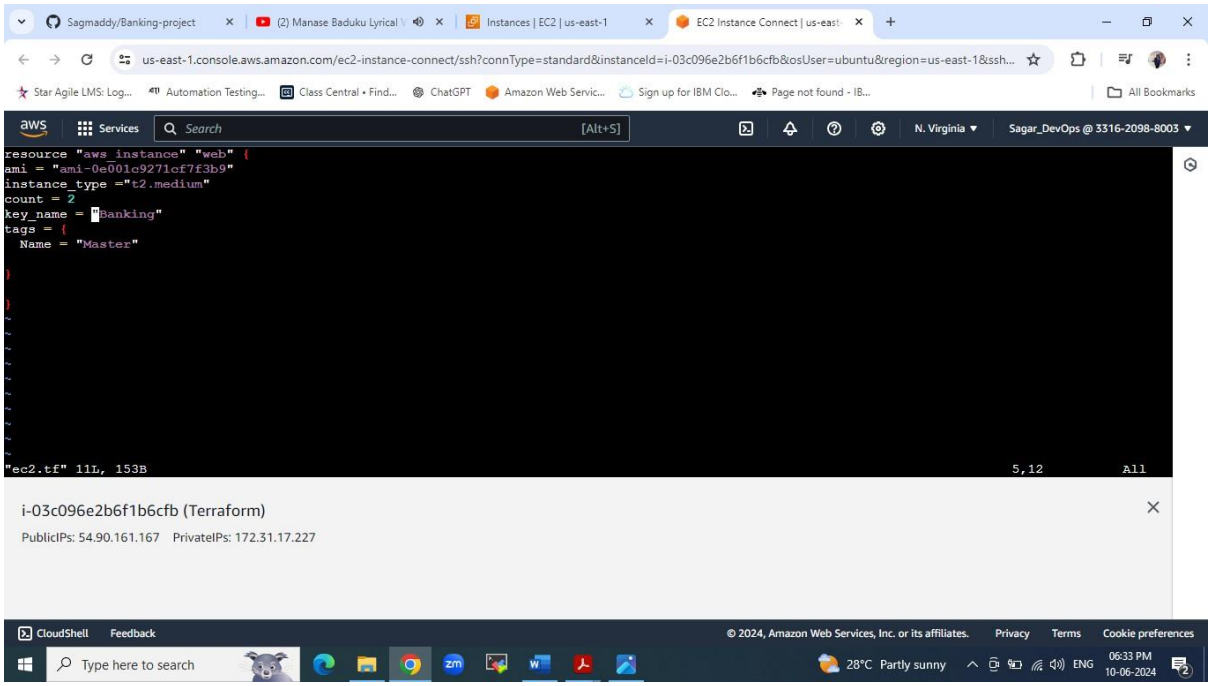
Instance type is t2.micro and created new key pair called Banking.



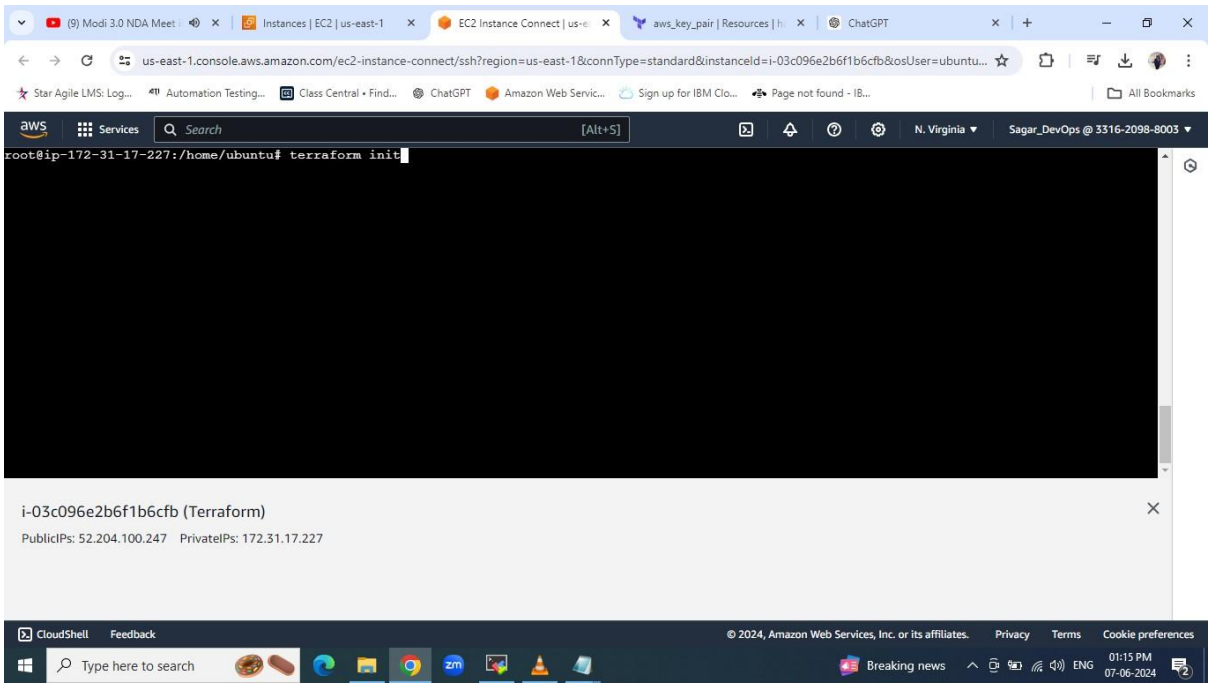
Terraform instance is up and running.



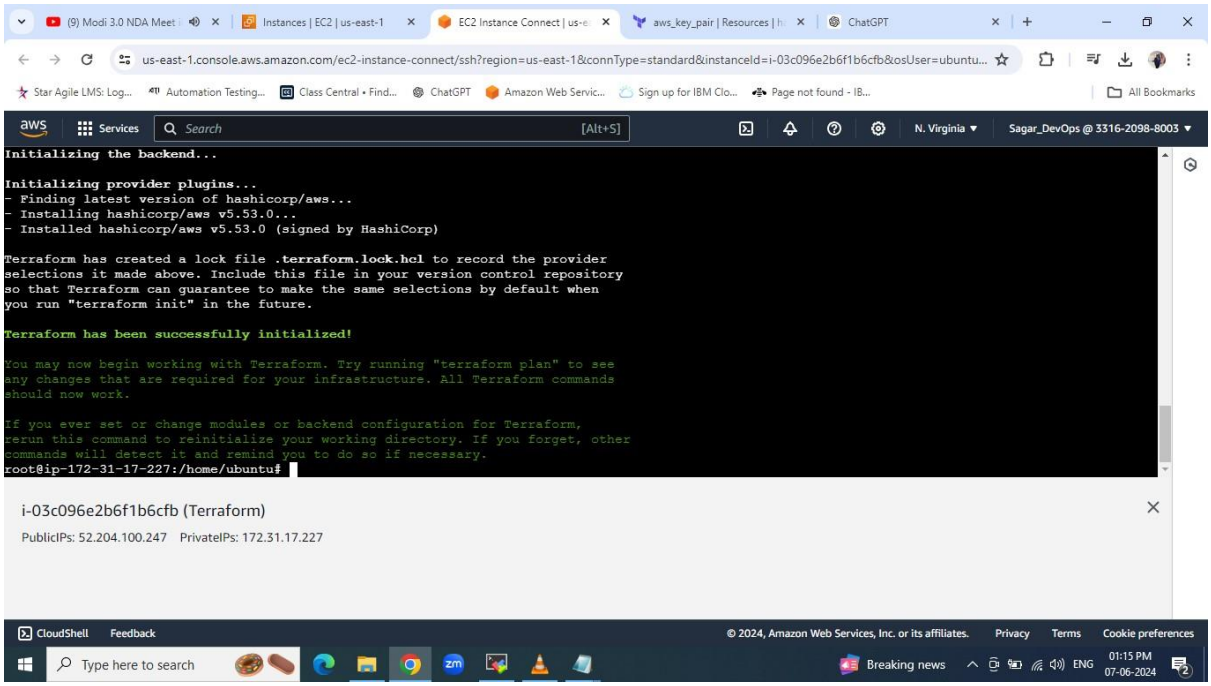
Installed terraform in the instance.



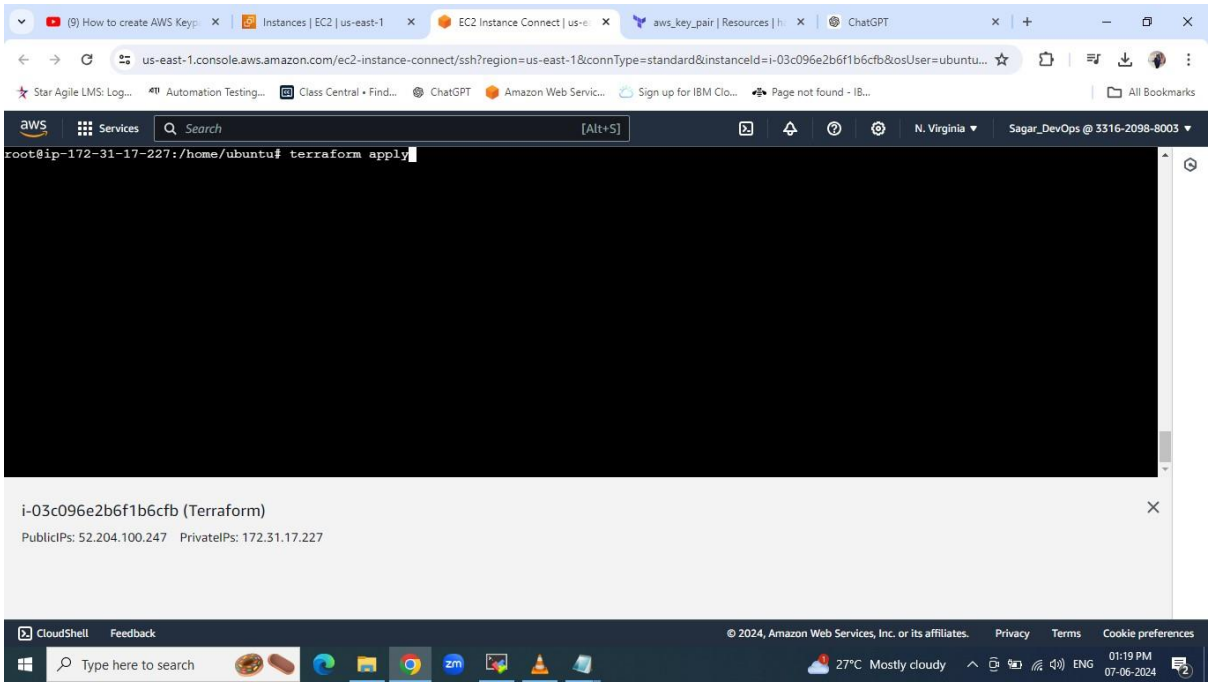
Created ec2.tf to create 2 t2.micro ubuntu instance named Master.



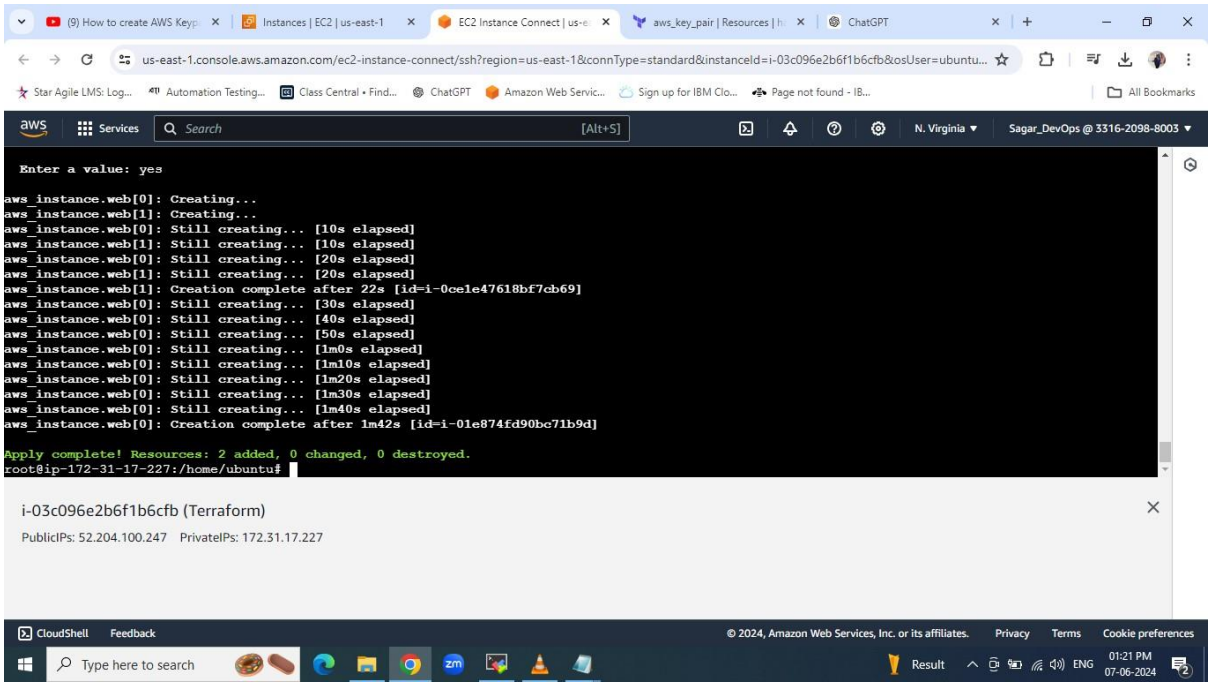
Using terraform init, initialised the working directory and downloads the necessary plugins to setup the infrastructure.



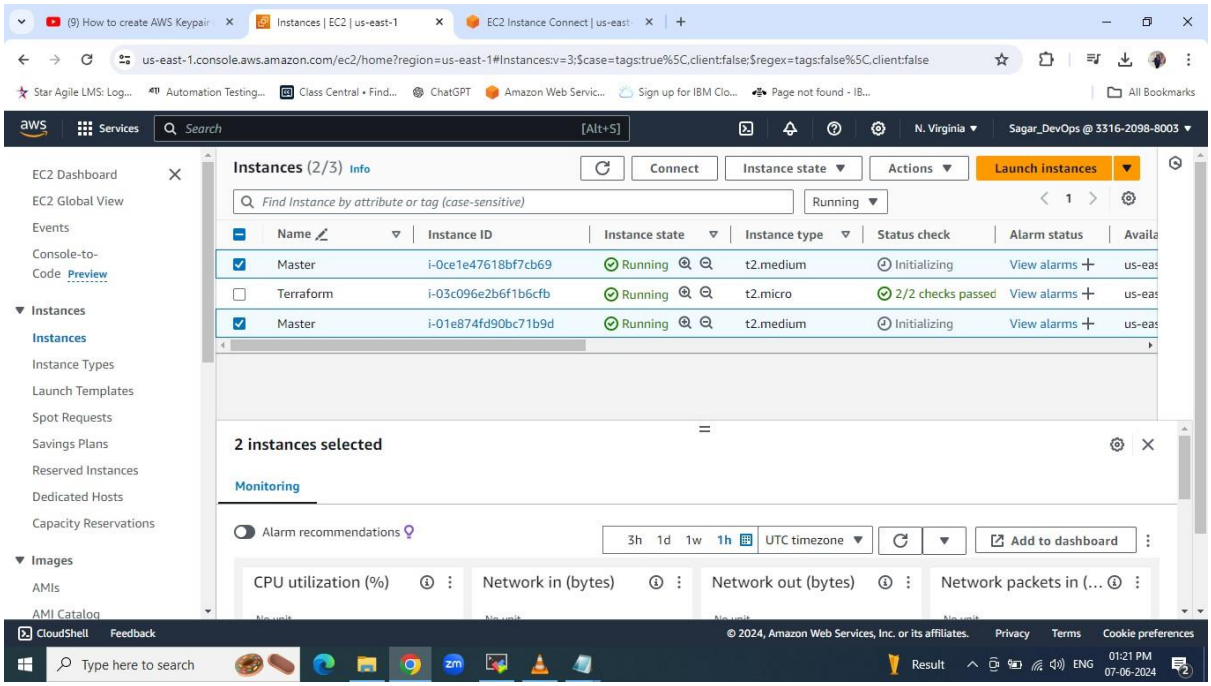
Terraform is successfully initialized.



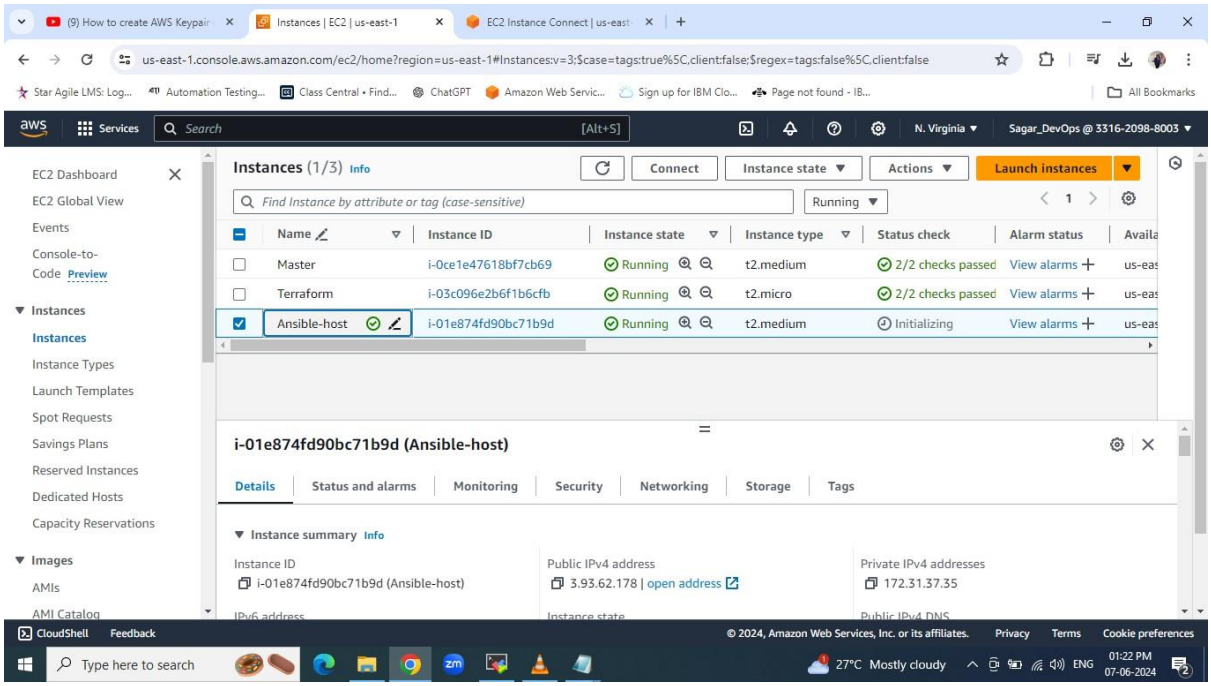
Using Terraform apply, created the infrastructure which is defined in the configuration file.



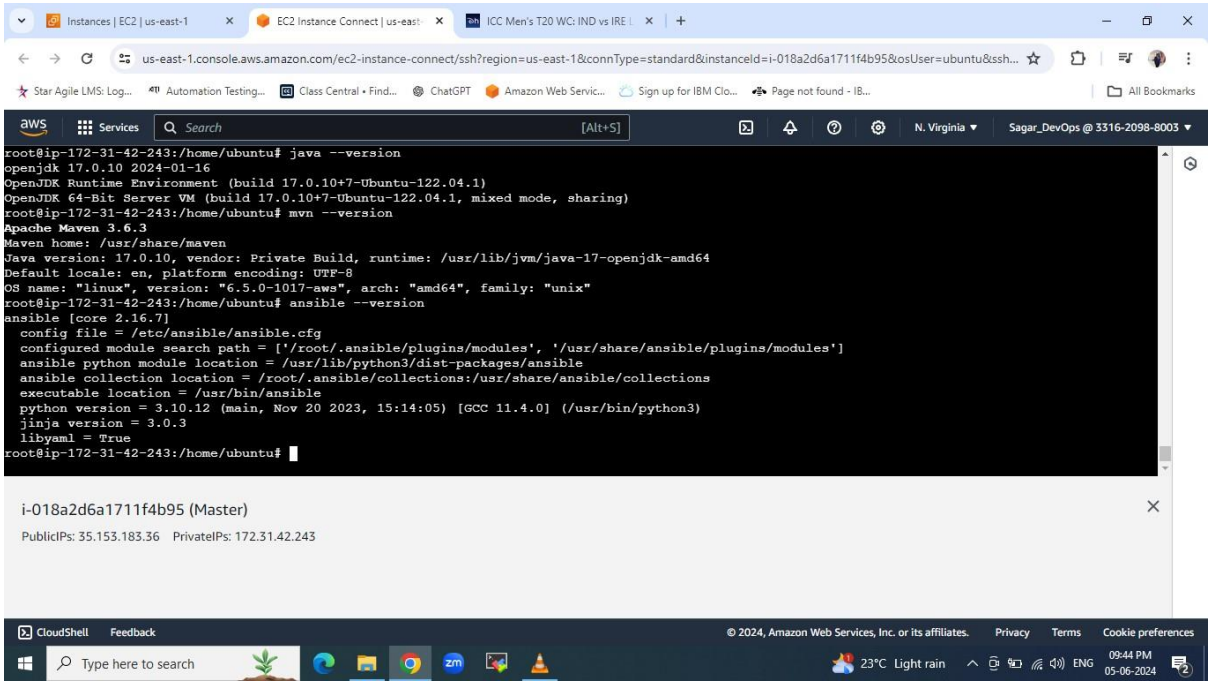
2 t2.medium ubuntu machines are created.



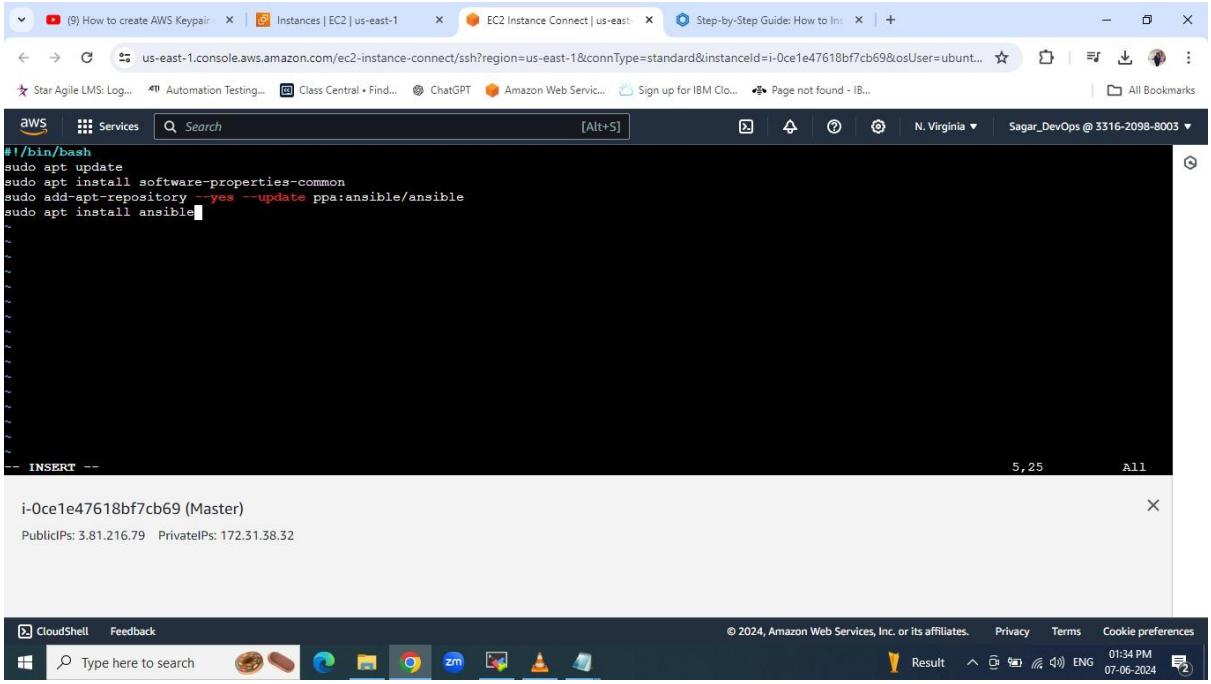
2 t2.medium ubuntu machines are named Master.



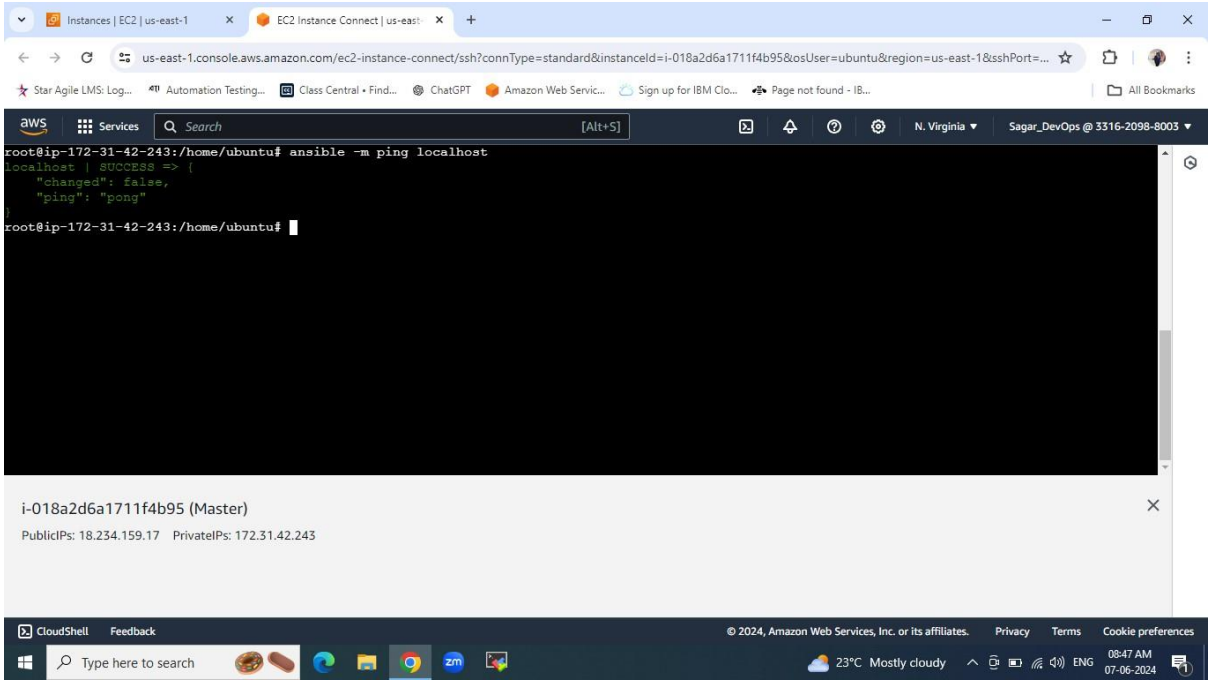
Renamed 1 instance to Ansible-host where the application will be deployed.



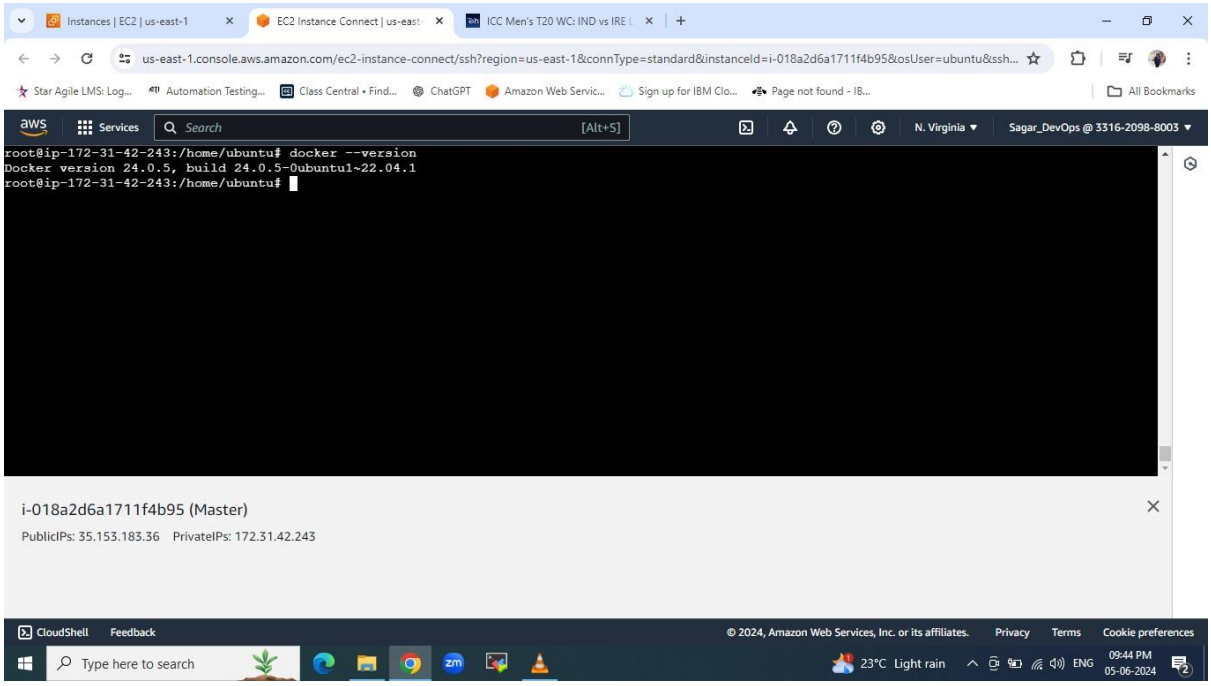
Installed java, maven, Docker, Jenkins and Ansible in Master machine.
Verified the installation using respective commands.



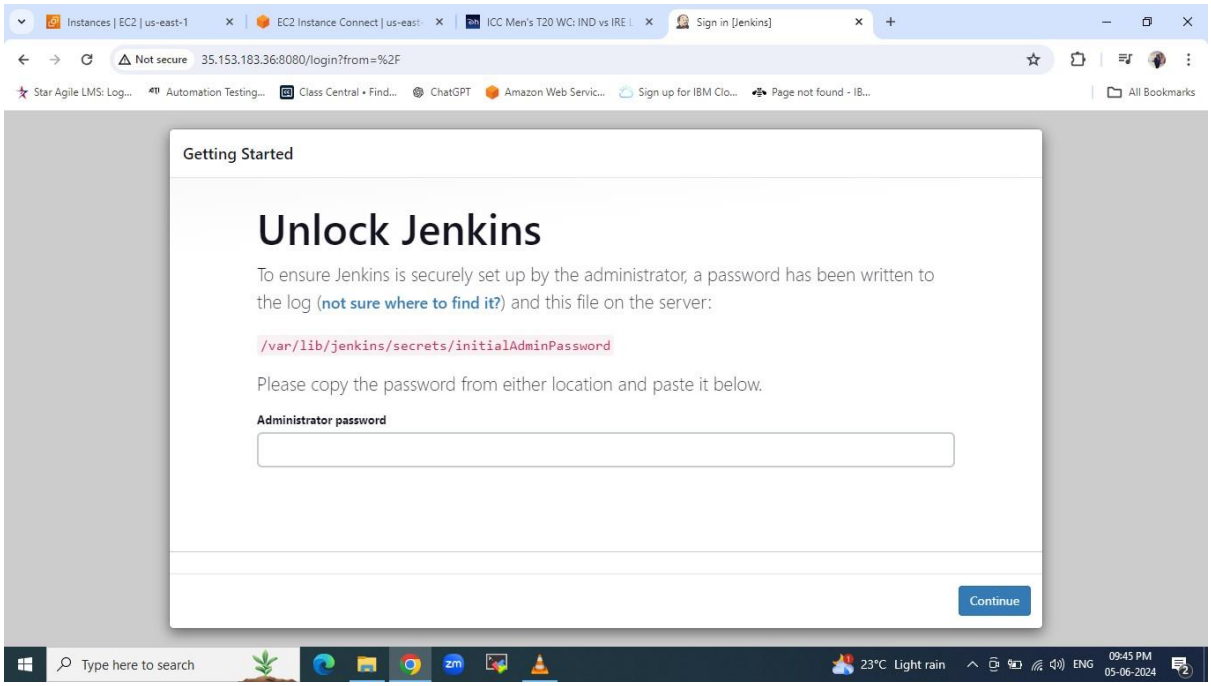
Commands to install Ansible in the machine.



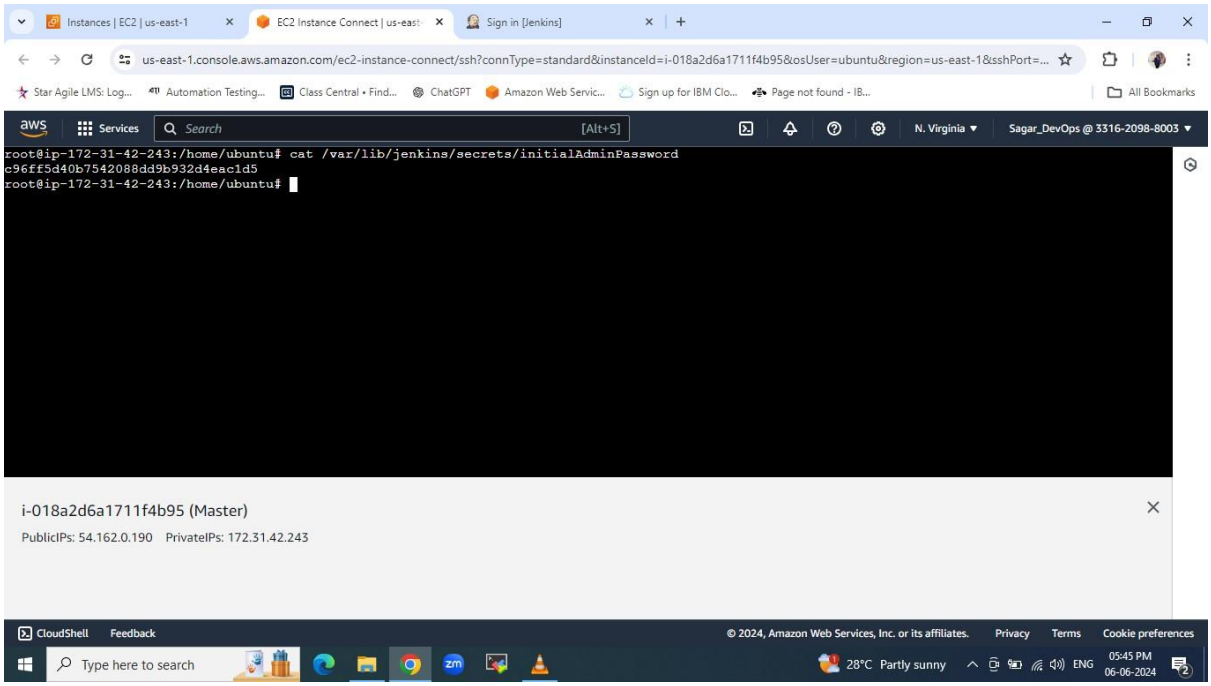
Ansible is installed and running in the machine.



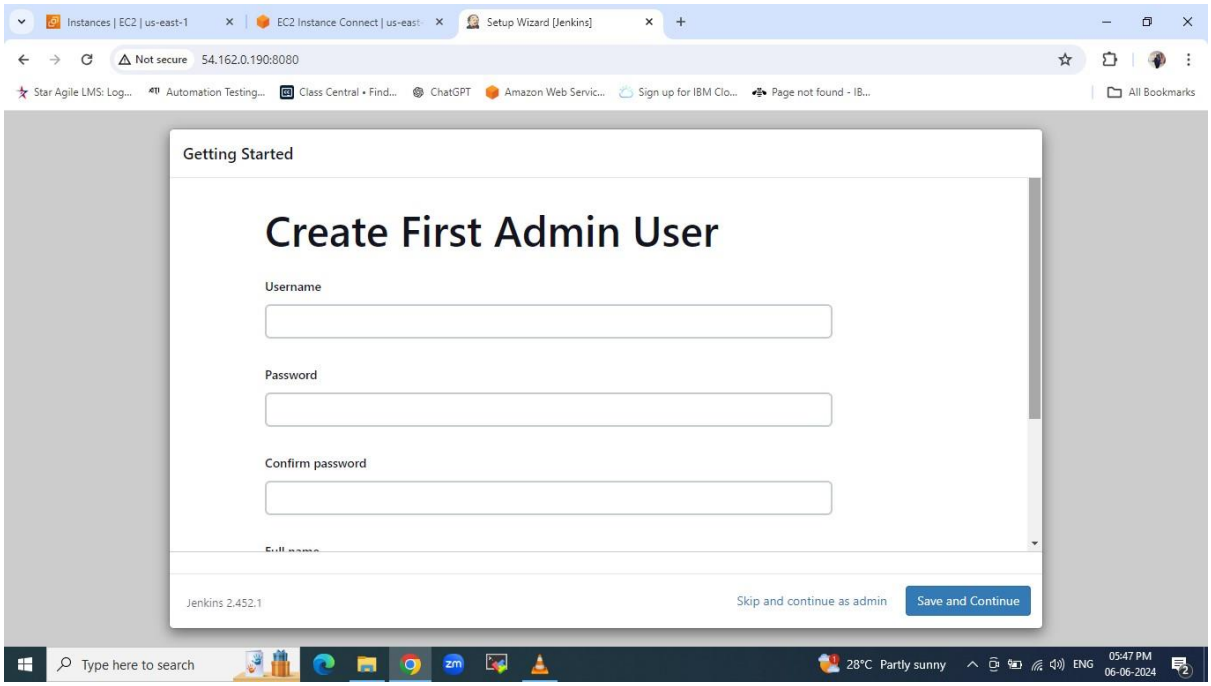
Docker is installed in Master machine.



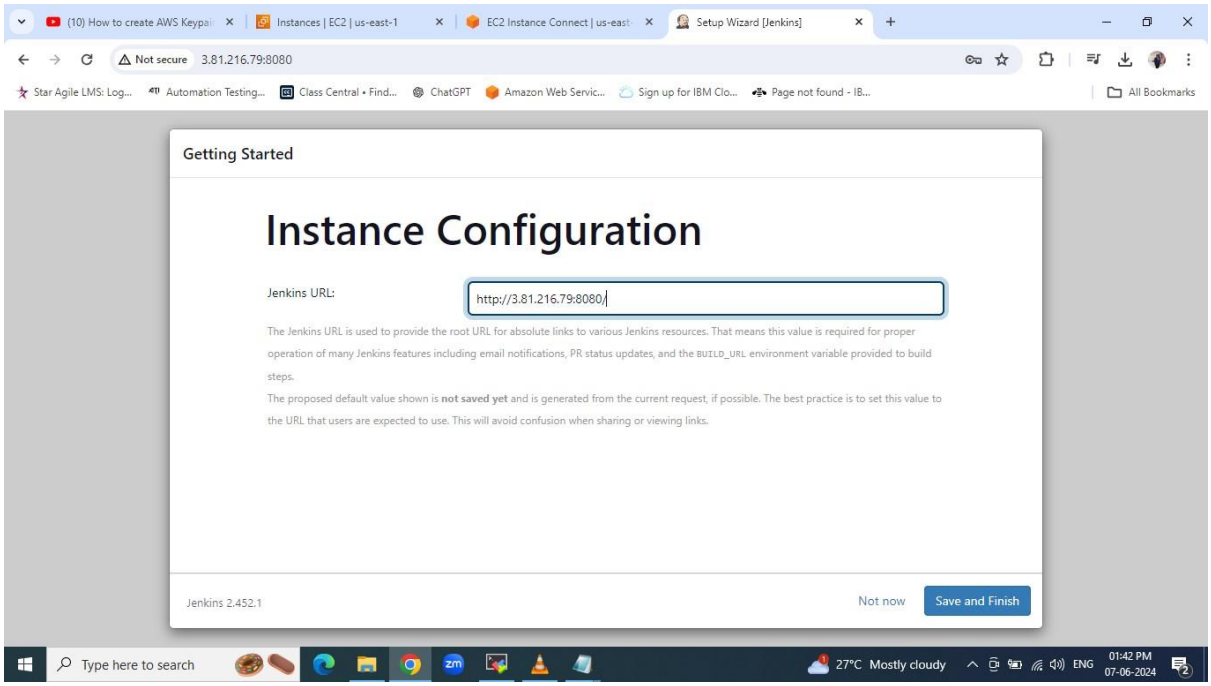
Jenkins is installed and verified using public Ip with 8080 port.



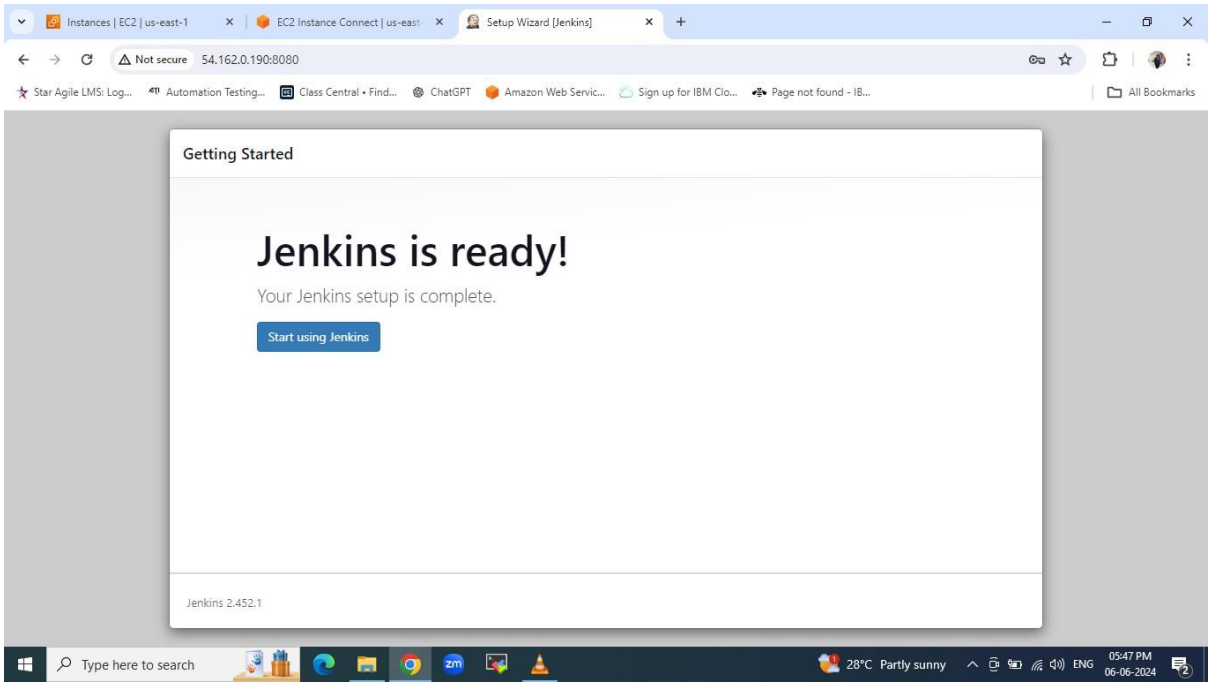
Using above command copied Administrator password.



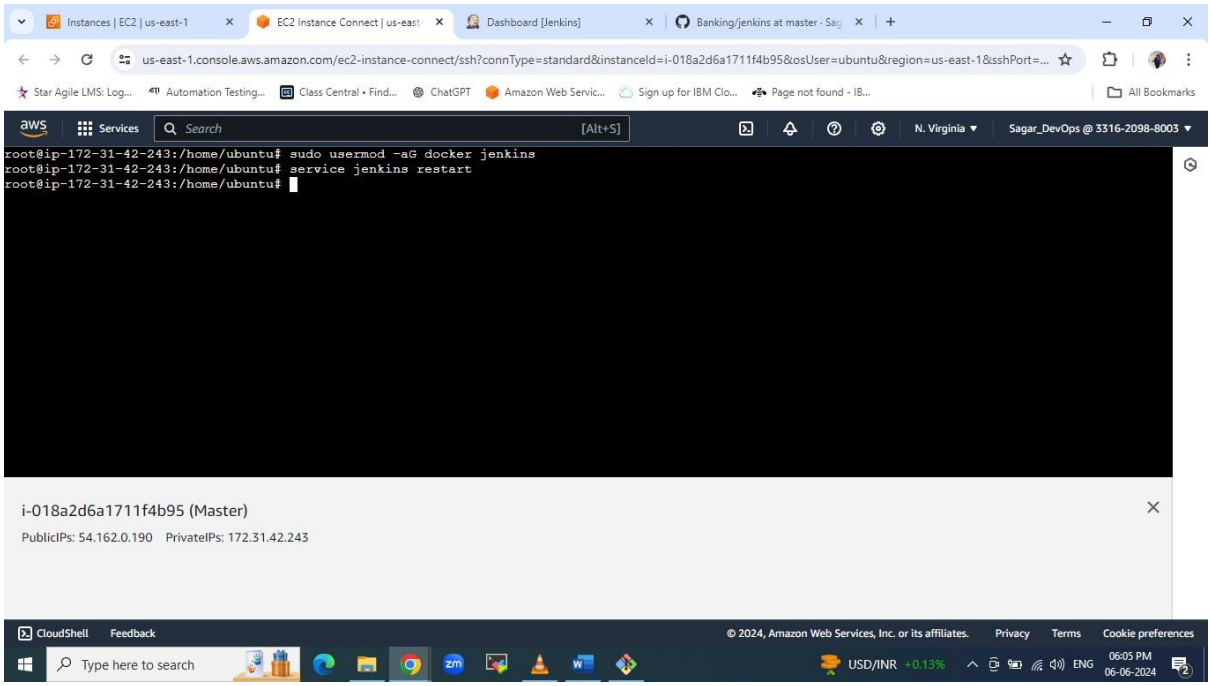
Setup the Admin user.



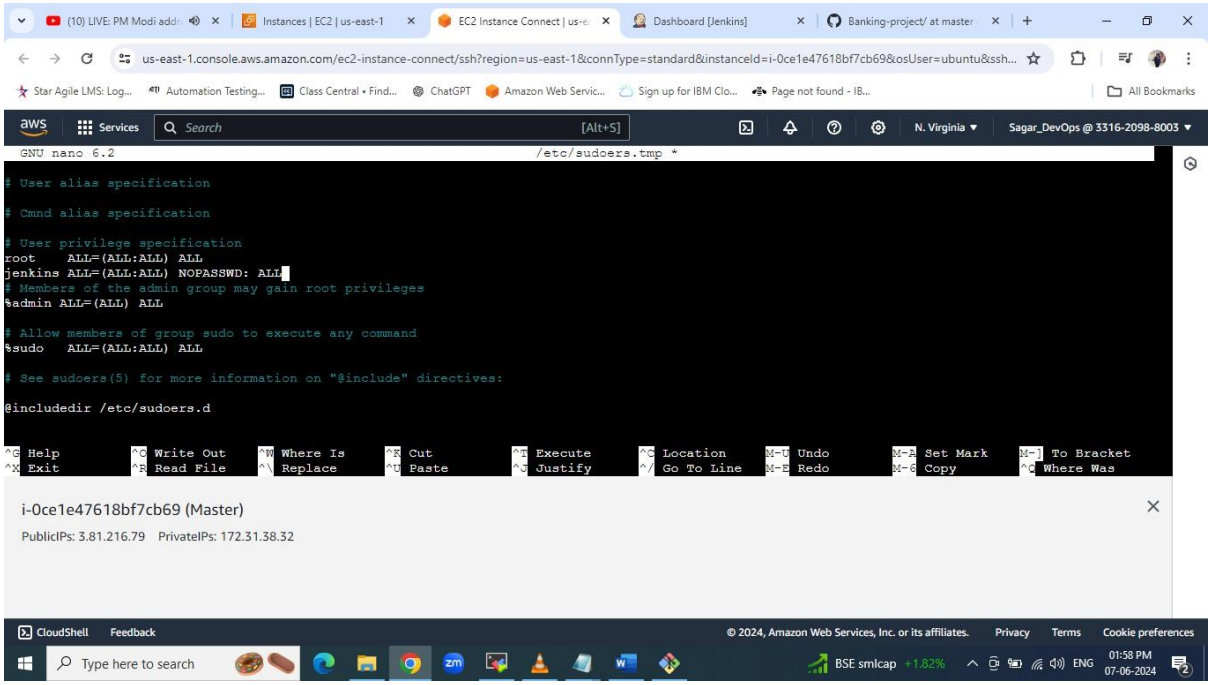
Verified the Jenkins instance URL.



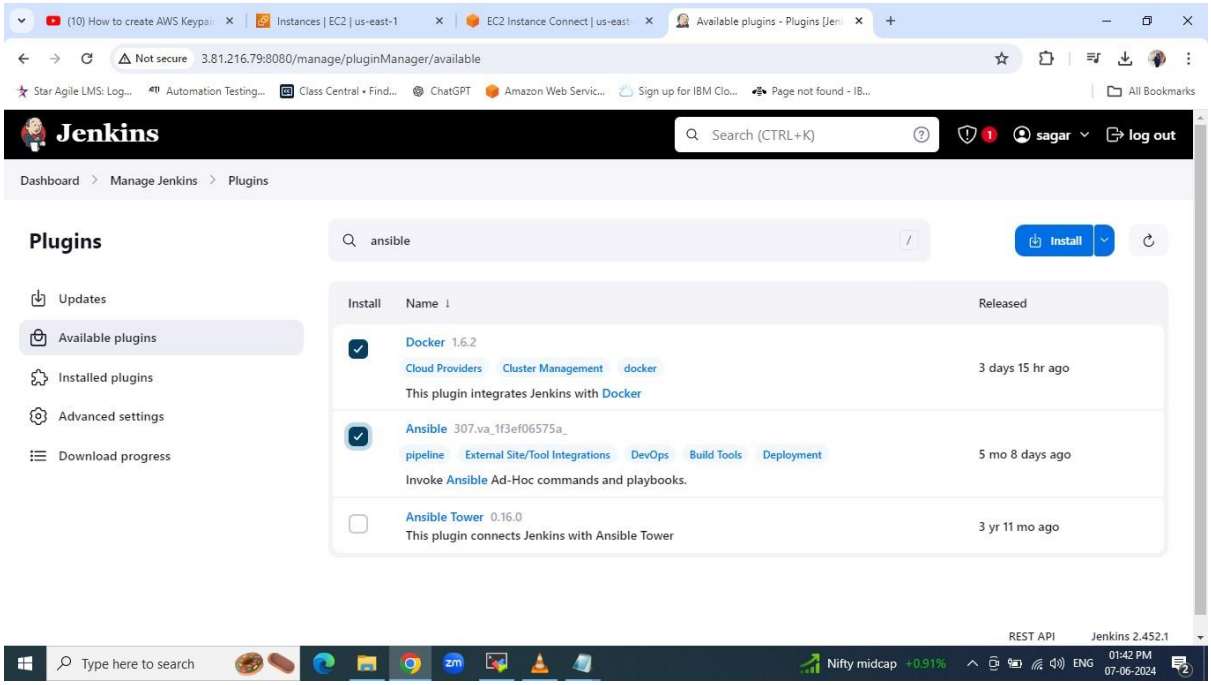
Jenkins is ready to use.



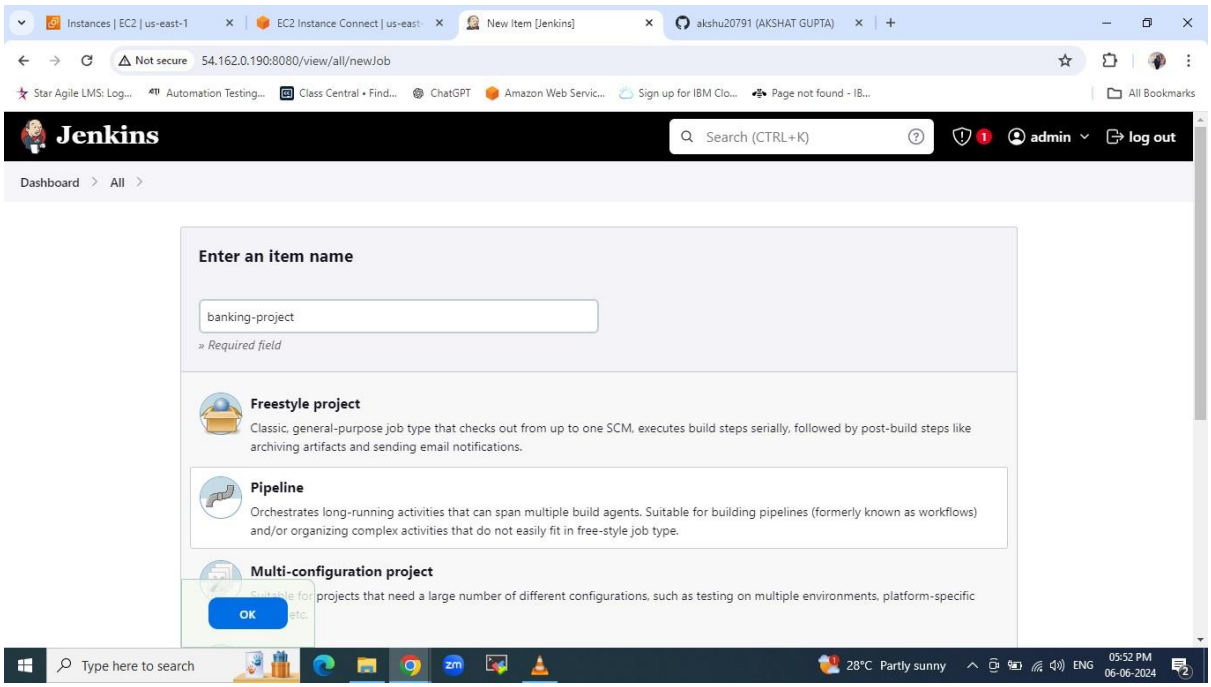
Ran the above command to add the Jenkins user to the docker group. This is done to allow ‘Jenkins’ user to execute the Docker commands without using sudo. This command is usually used in CI/CD environments where Jenkins needs to build docker images or run docker containers.



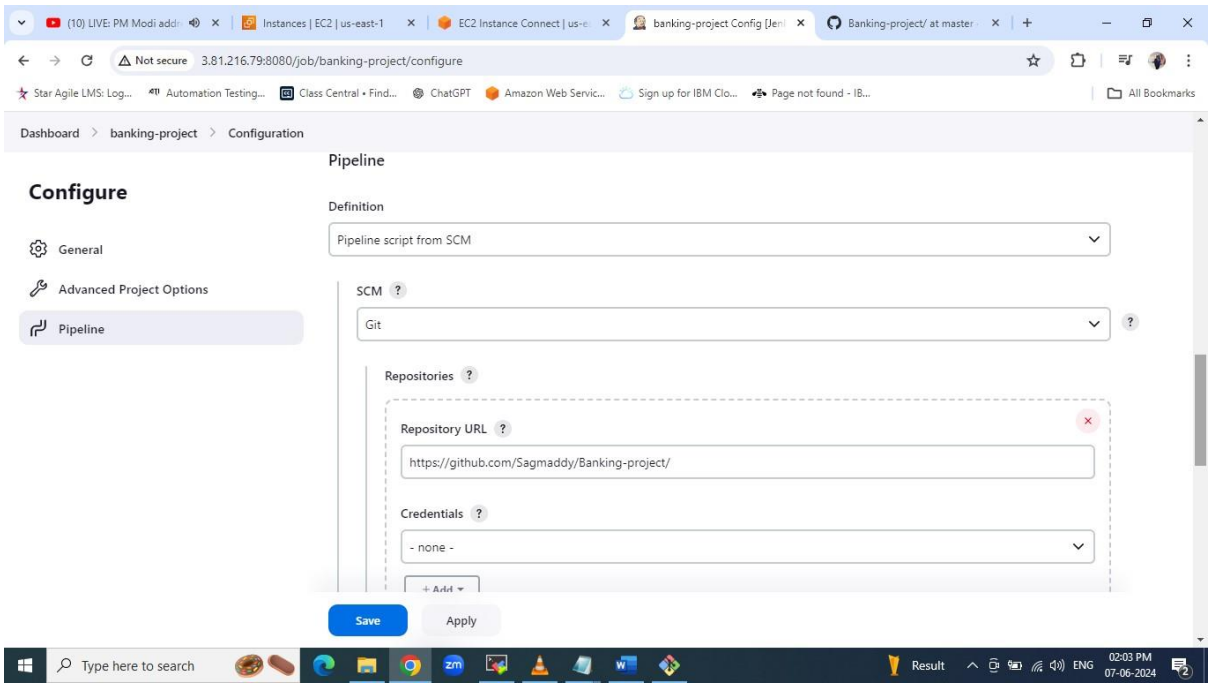
In the visudo file, add an additional line so that Jenkins doesn't require a password to execute Docker commands.



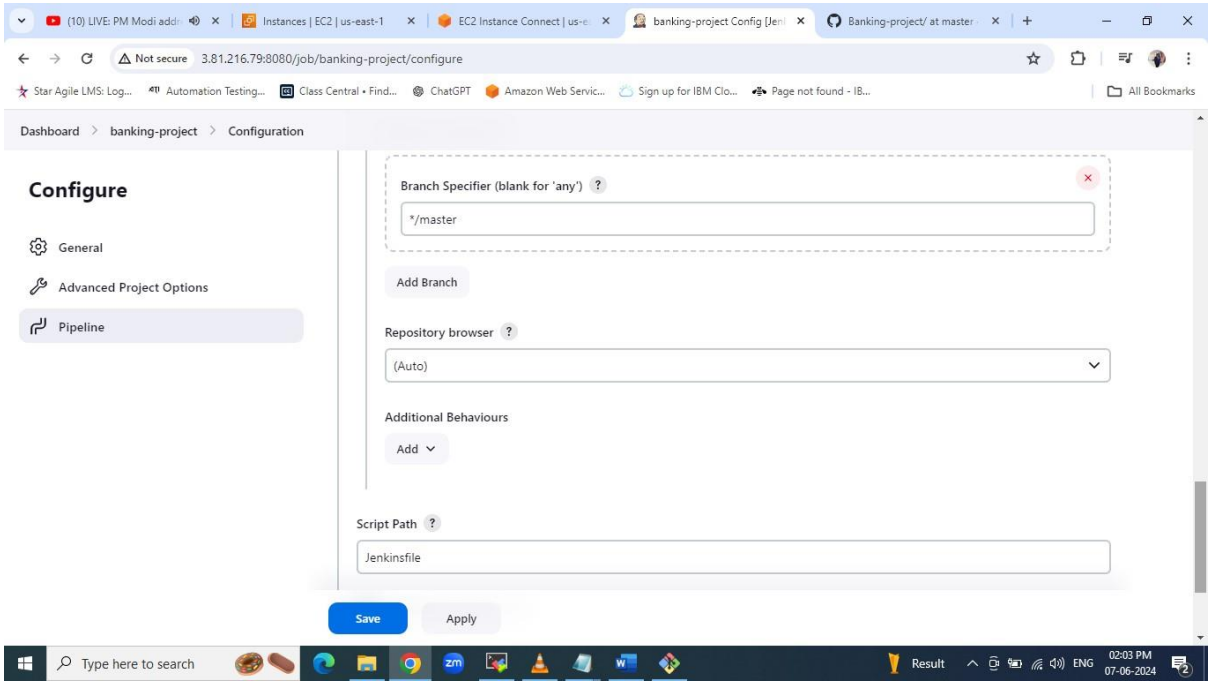
After setting up Jenkins, go to "Manage Jenkins" and install the Docker and Ansible plugins.



Click on new item and create a pipeline project named Banking-project.

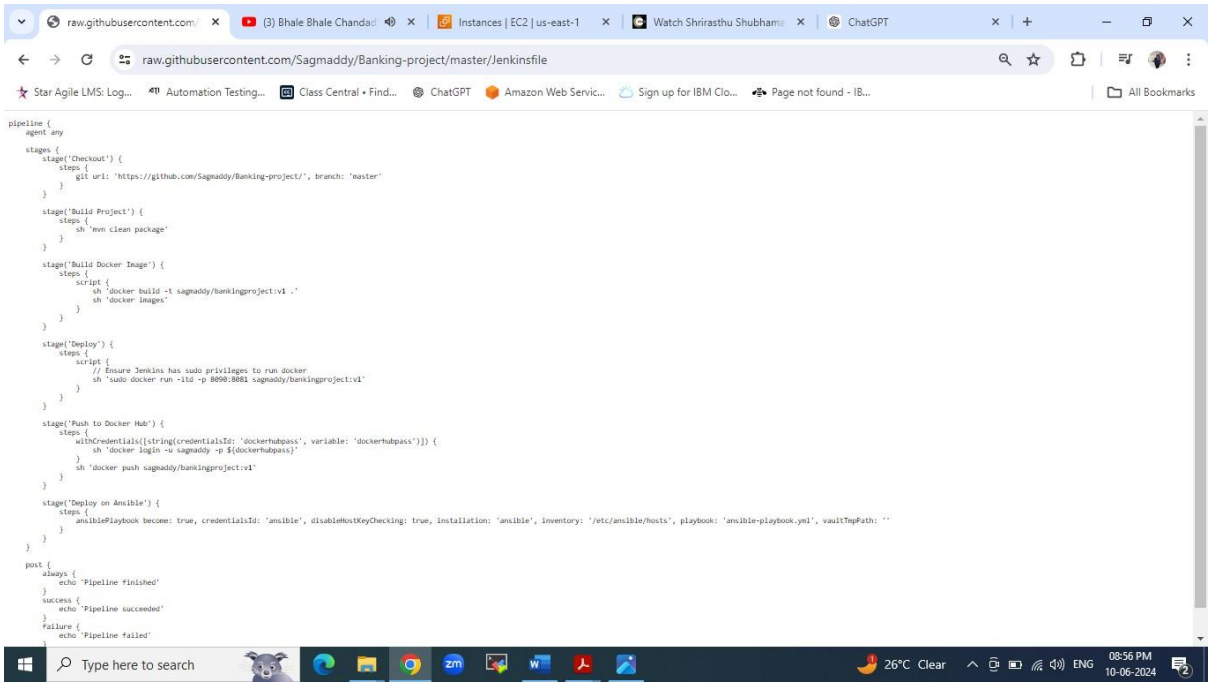


Go to pipeline step and select Pipeline script from SCM i.e., from Git. Provided the git URL so that Jenkins will fetch the project from above URL.

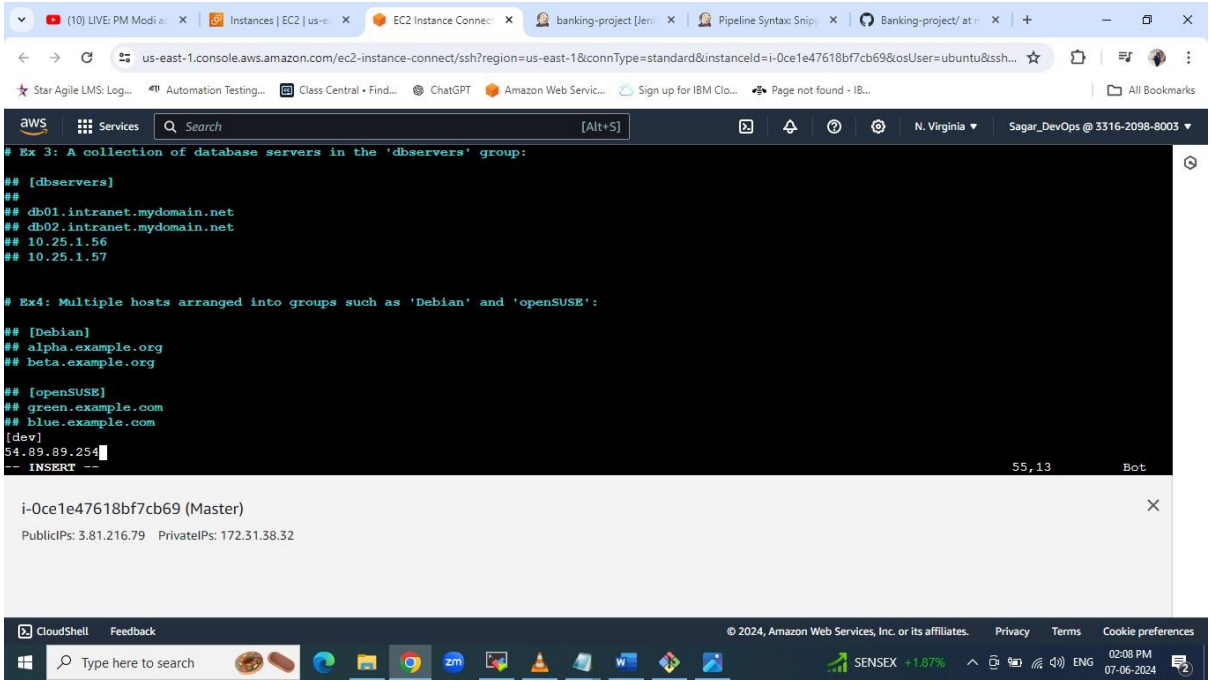


Select master branch and Script path Jenkinsfile.

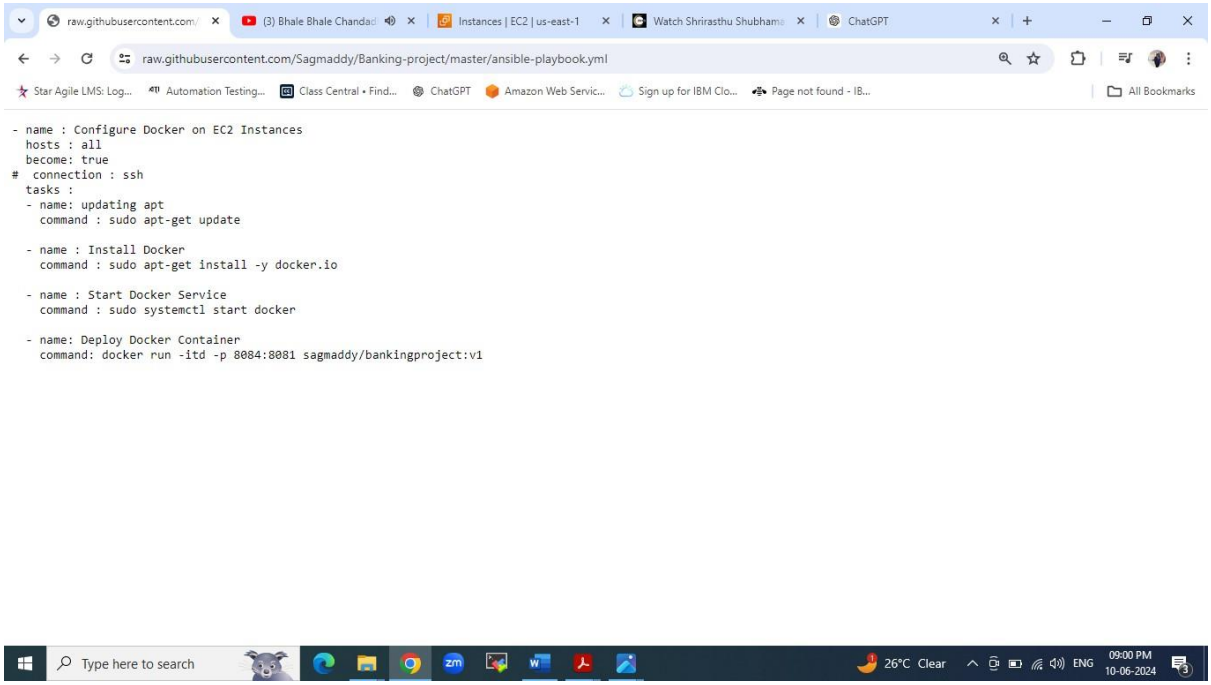
A Jenkinsfile is a text file that contains the definition of a Jenkins pipeline and is used to describe the stages and steps of your continuous integration and continuous delivery (CI/CD) pipeline.



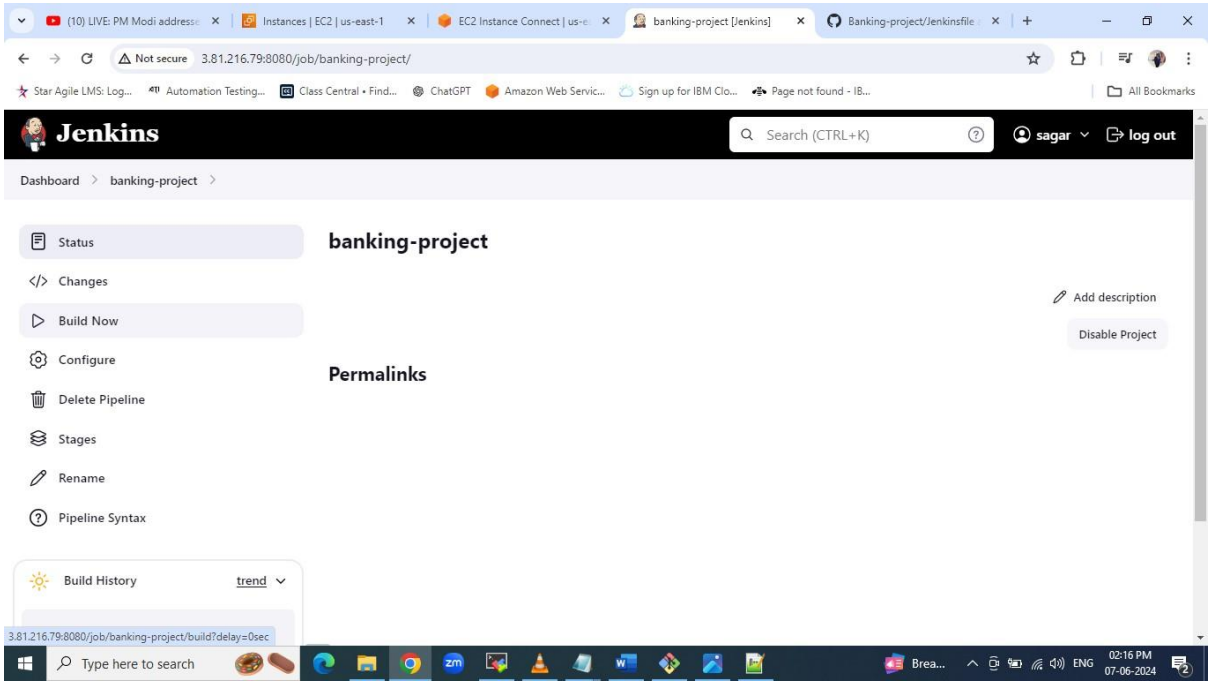
This is the Jenkinsfile of this project. The clear code is in the git repo.



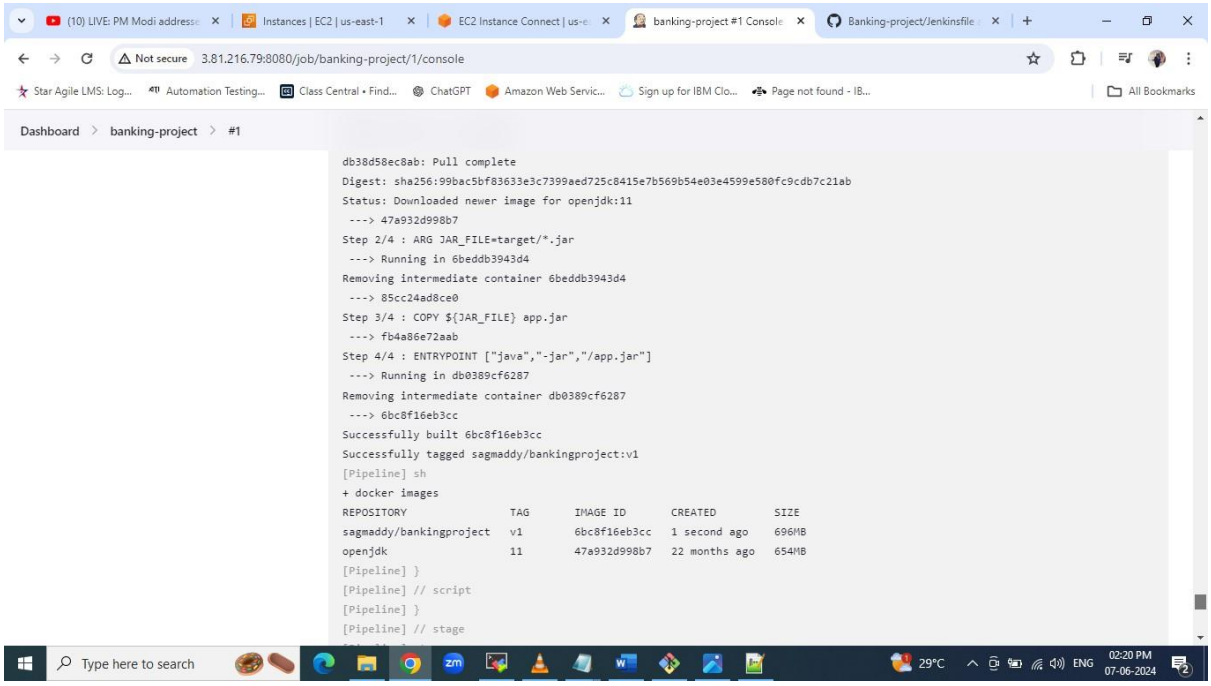
To deploy the application in the ansible-host machine, add IP address of the machine in hosts file which you can find under /etc/ansible/hosts path.



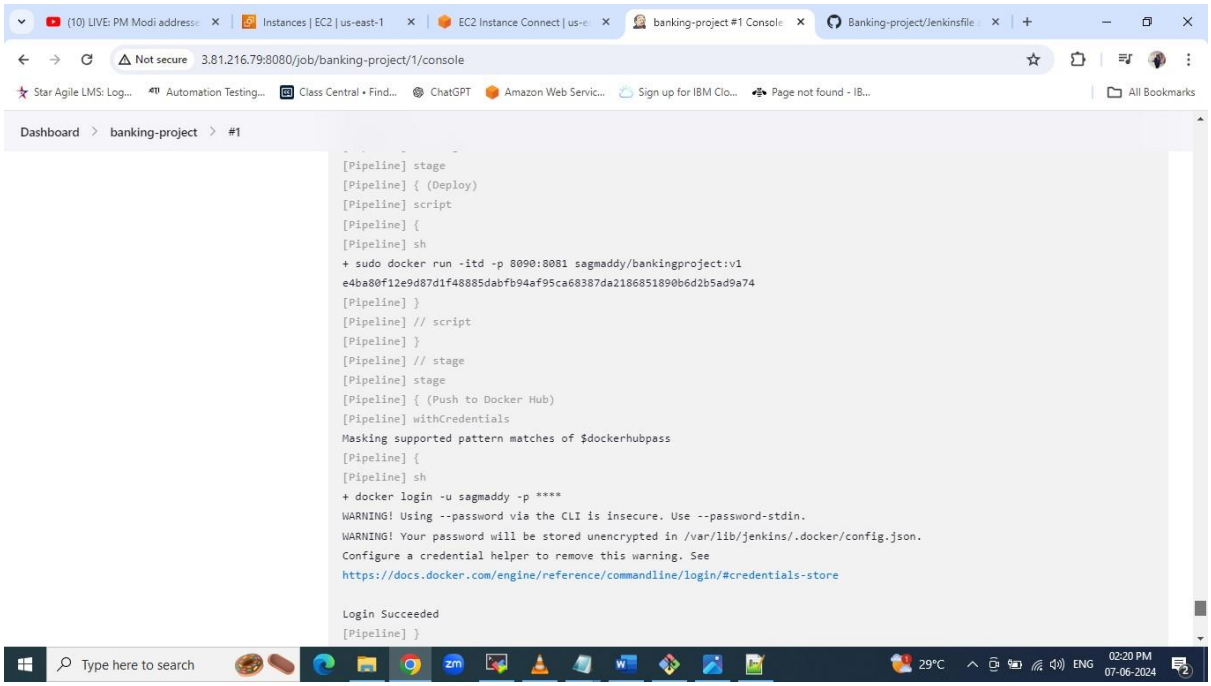
Created ansible-playbook to install docker and deploy the container in host machine. The application is deployed in 8084 port.



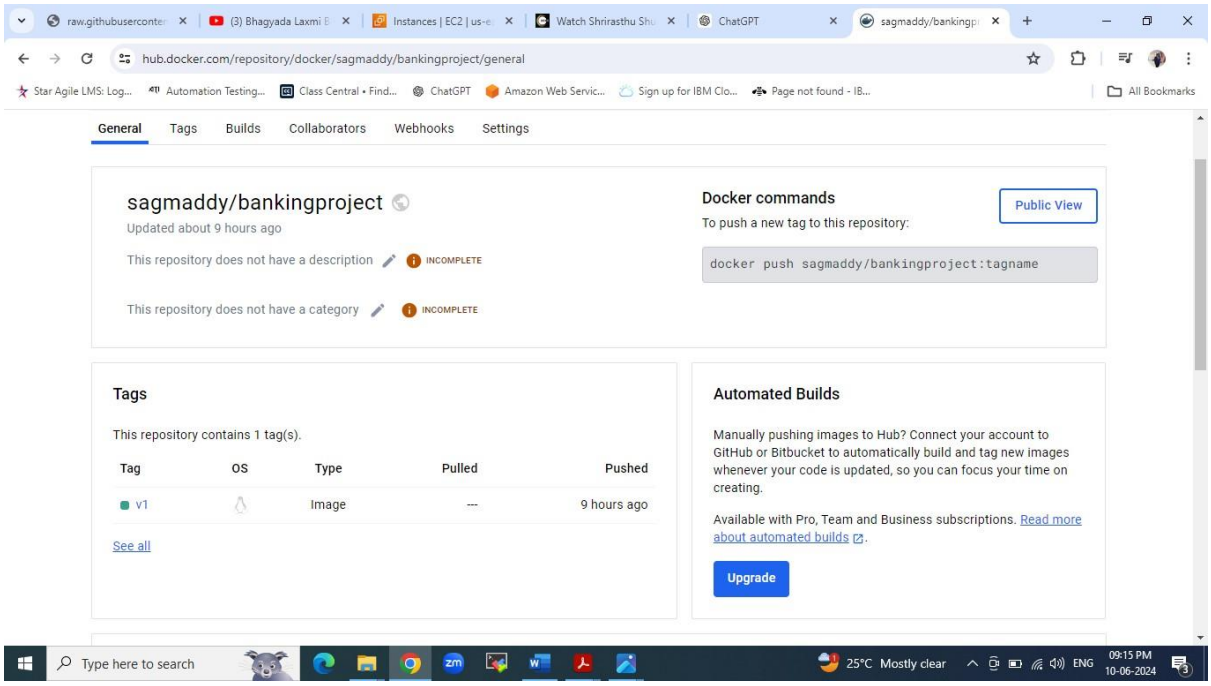
Build the pipeline project by clicking build now.



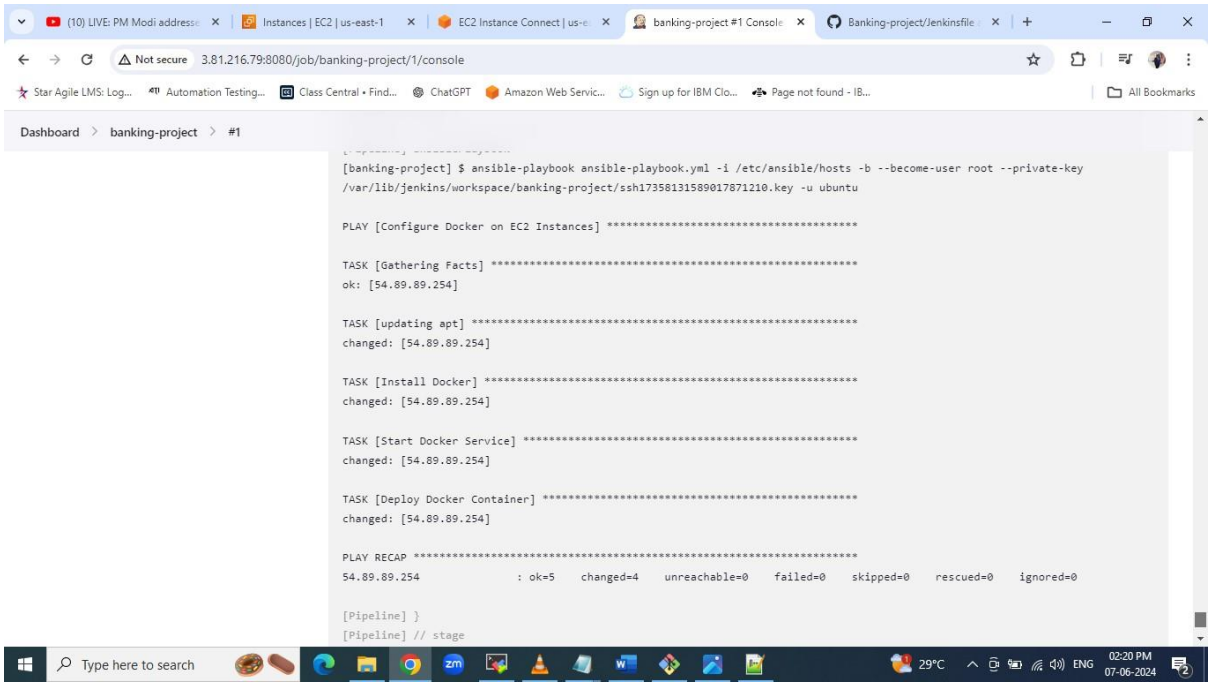
Project is successfully built. The docker file is executed and pulled the OpenJDK image from the repo and ran the jar file and image is built successfully and tagged with sagmaddy docker hub repo.



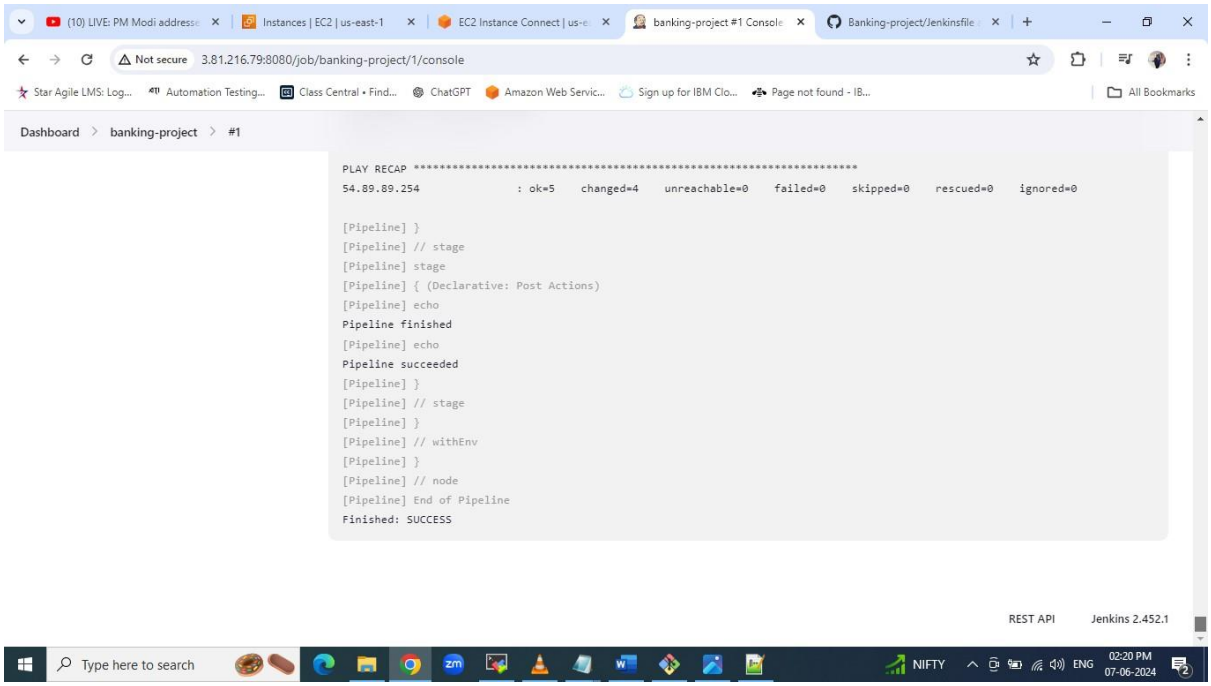
The container is created and running in 8090 port. The Docker hub is successfully logged in and image is pushed in Docker hub repo.



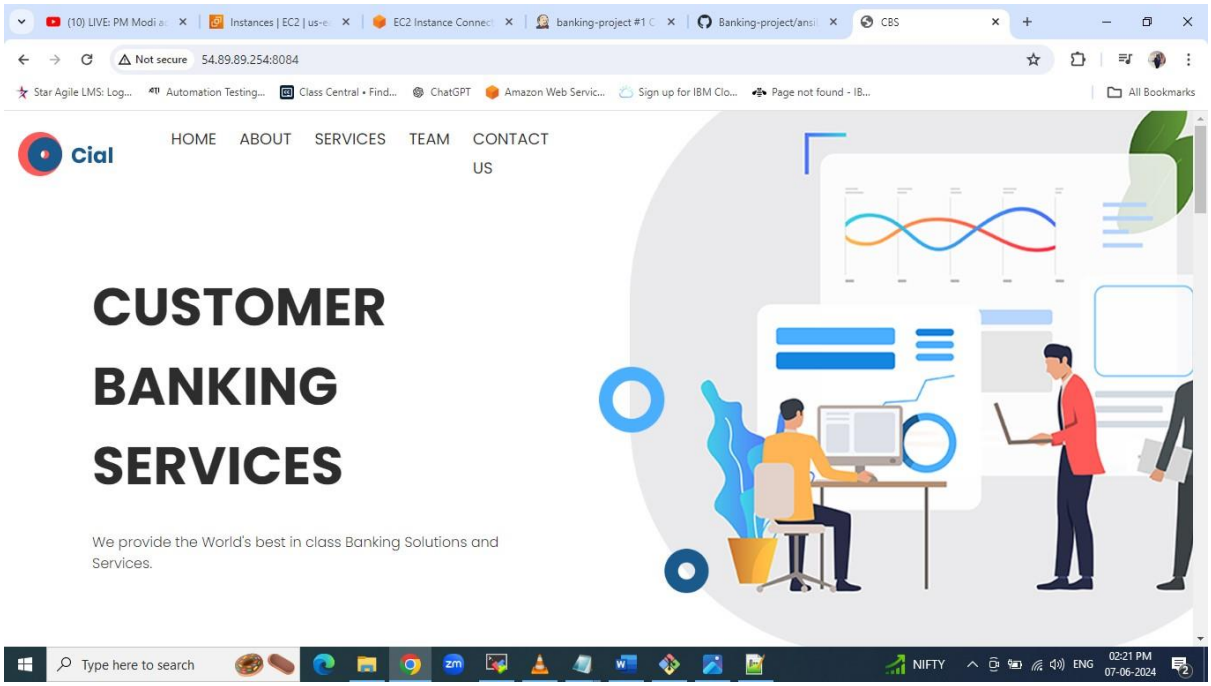
The image is pushed in docker hub repo.



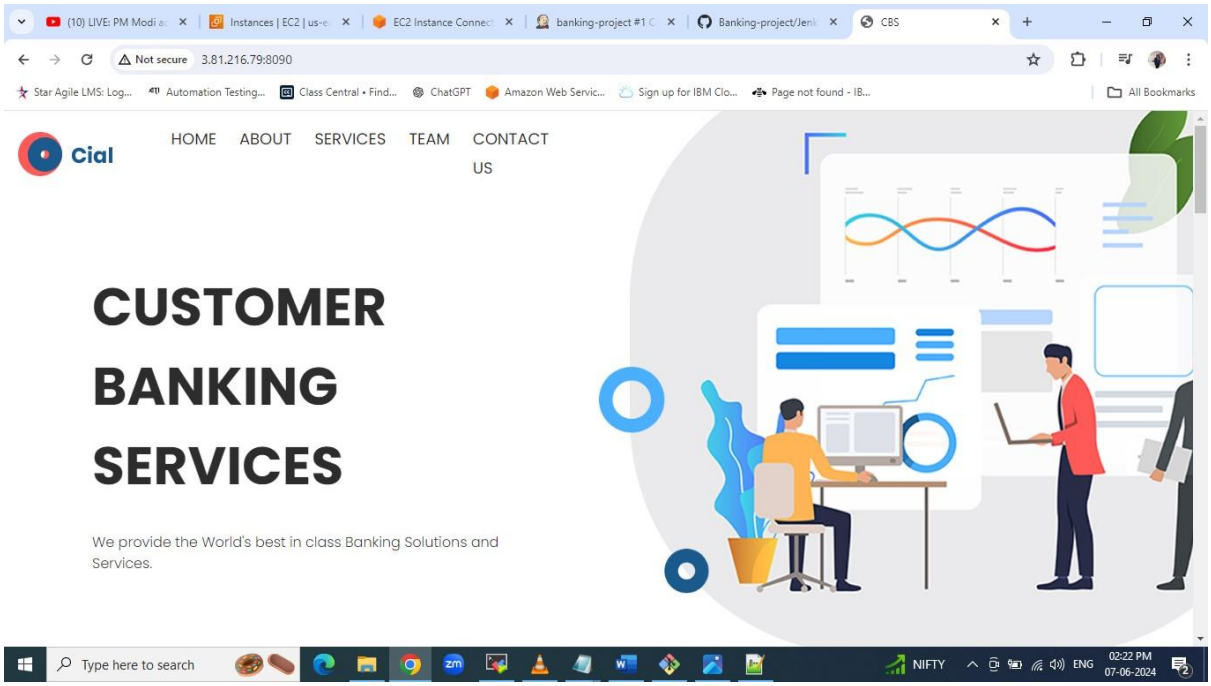
The ansible playbook is executed and installed docker and started the docker service and deployed the docker container. It is deployed in host machine.



Post actions to confirm the pipeline is finished and succeeded.



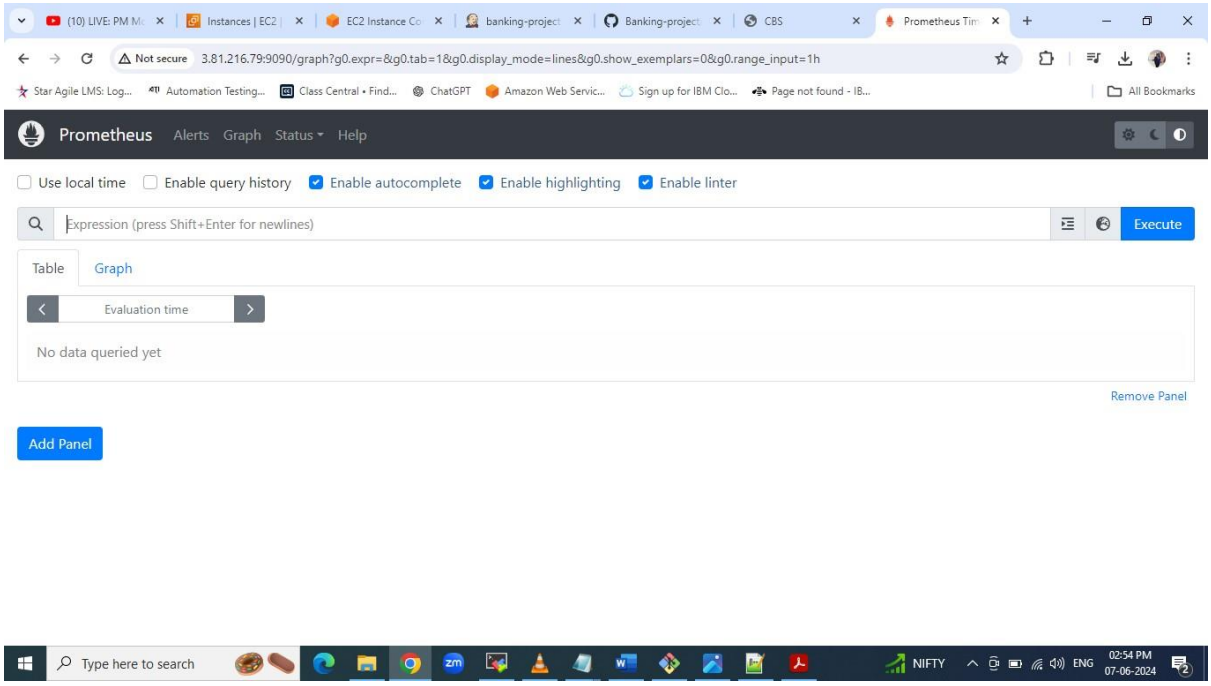
The application is deployed successfully and running in host machine 8084 port.



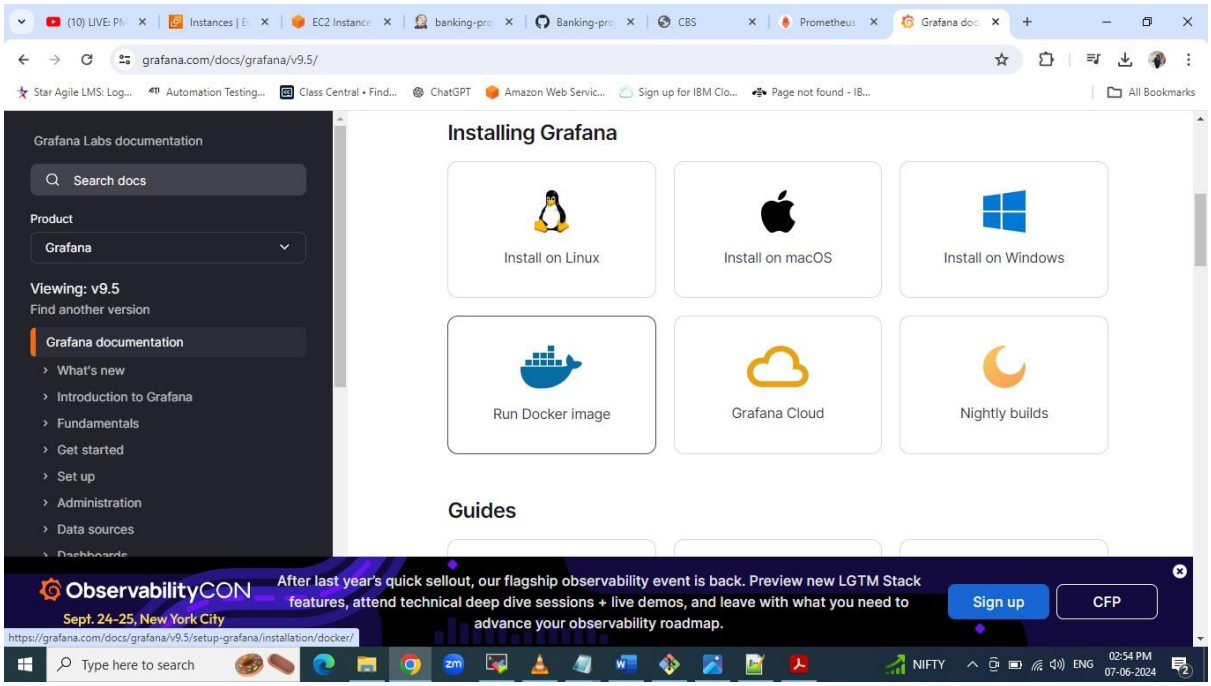
The application is also running in master machine in 8090 port.

We have initialised the docker and deployed in Jenkinsfile and ansible-playbook.

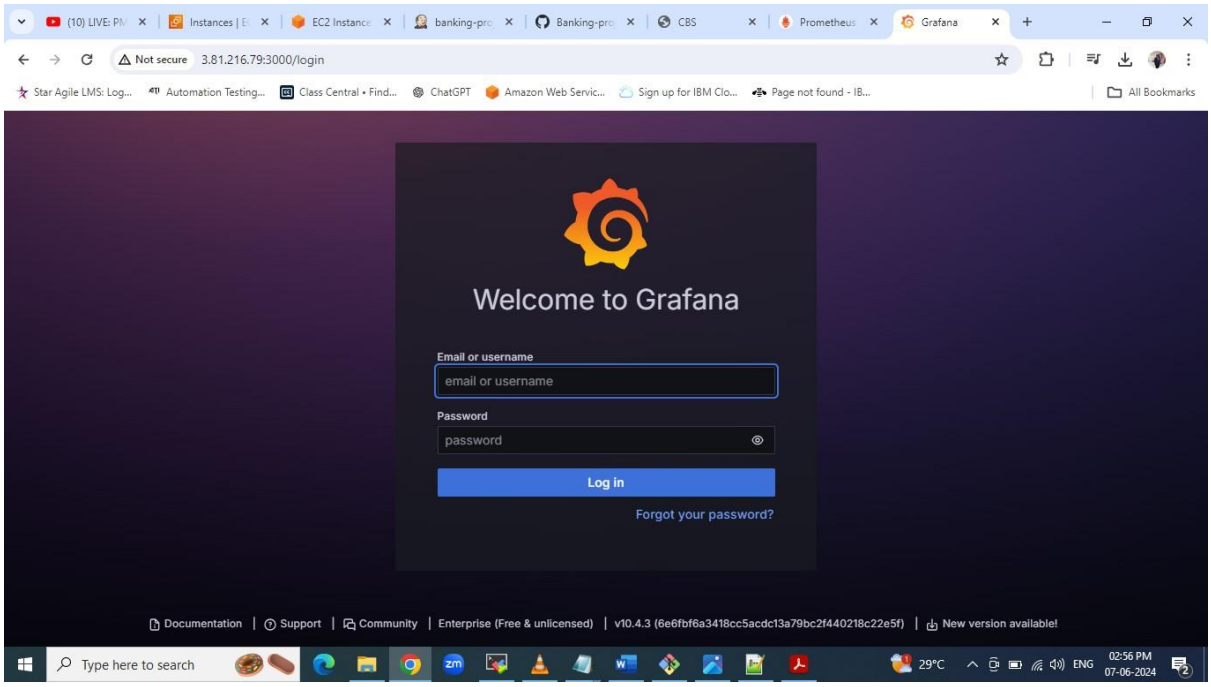
Installed Prometheus and Grafana to continuously monitoring the server configured the Grafana to display a dashboard of metrics.



Prometheus is installed and running on 9010 port.

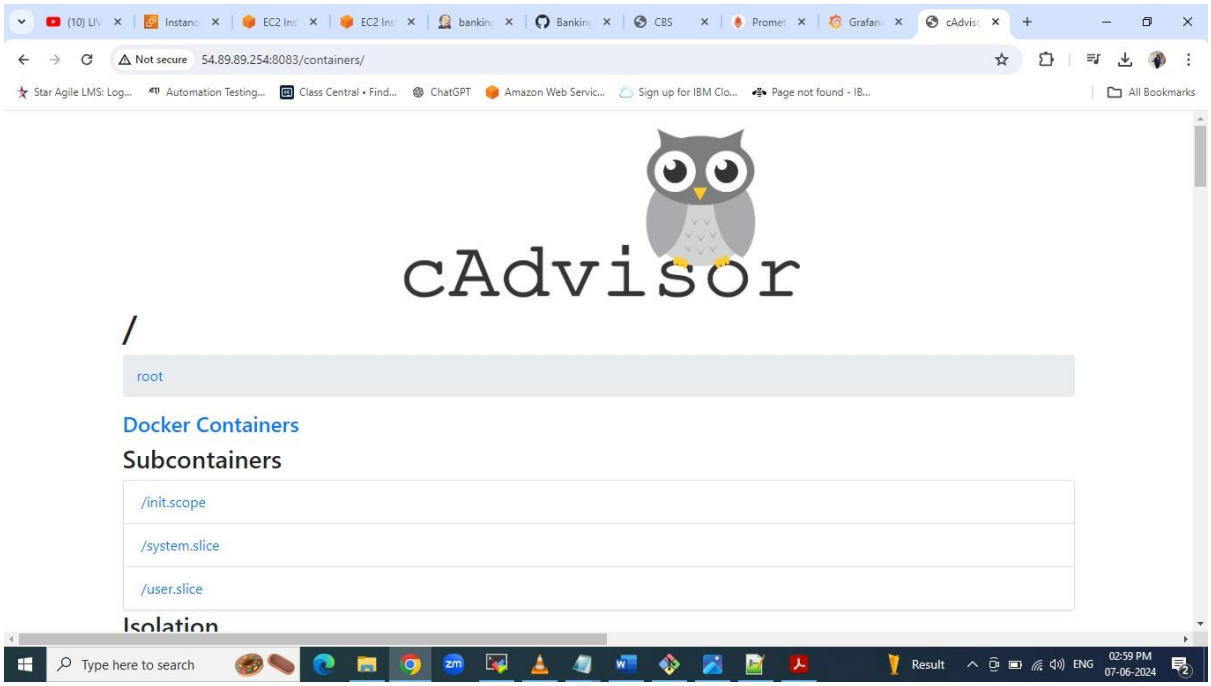


Grafana is installed by running docker image.

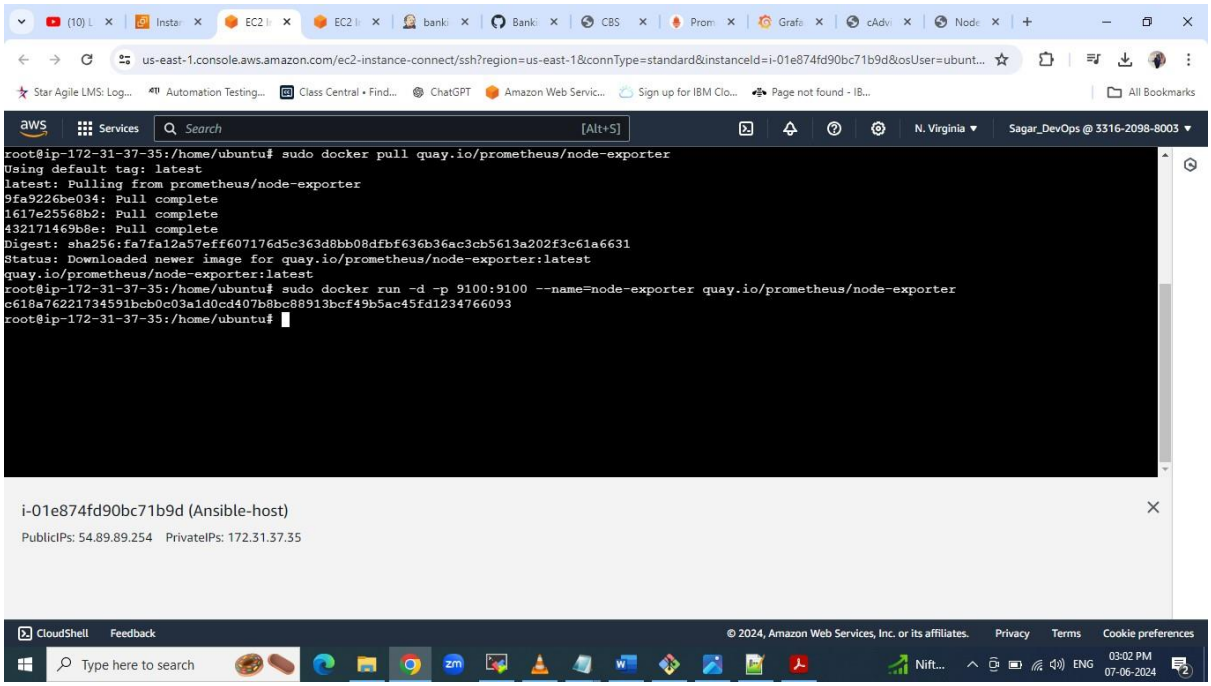


Grafana is installed and running on 3000 port.

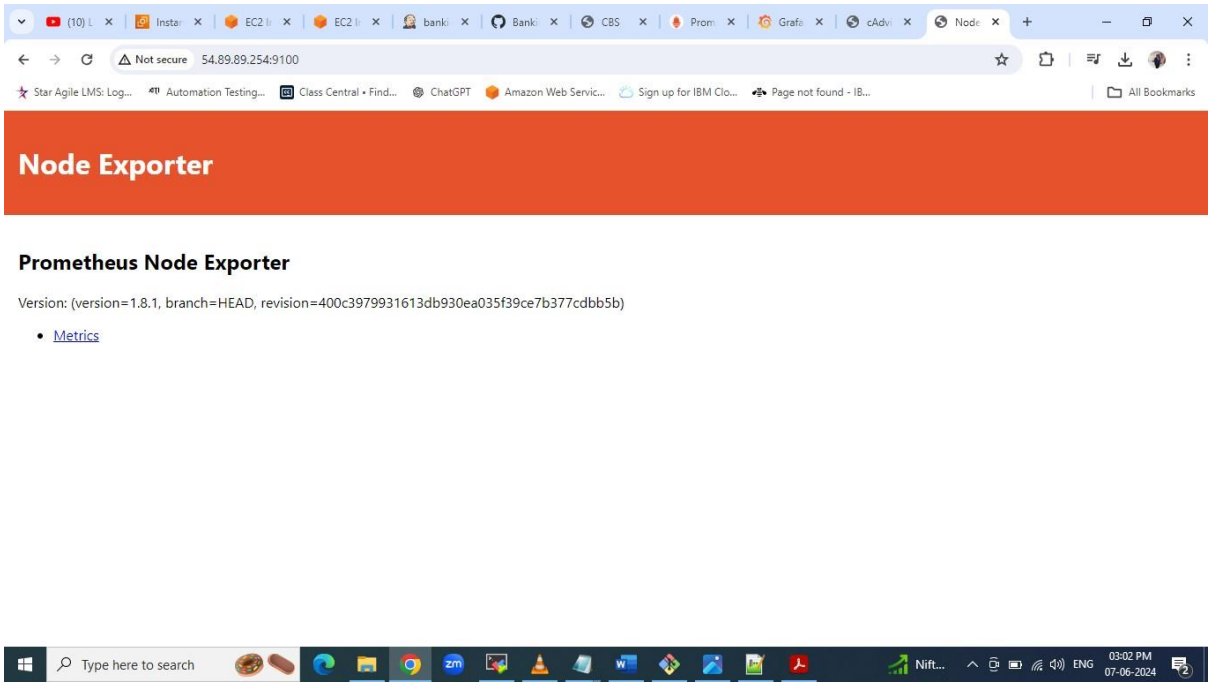
cAdvisor and Node Exporter are essential for comprehensive monitoring and visualization in containerized environments because they provide detailed metrics that Prometheus can scrape and Grafana can visualize.



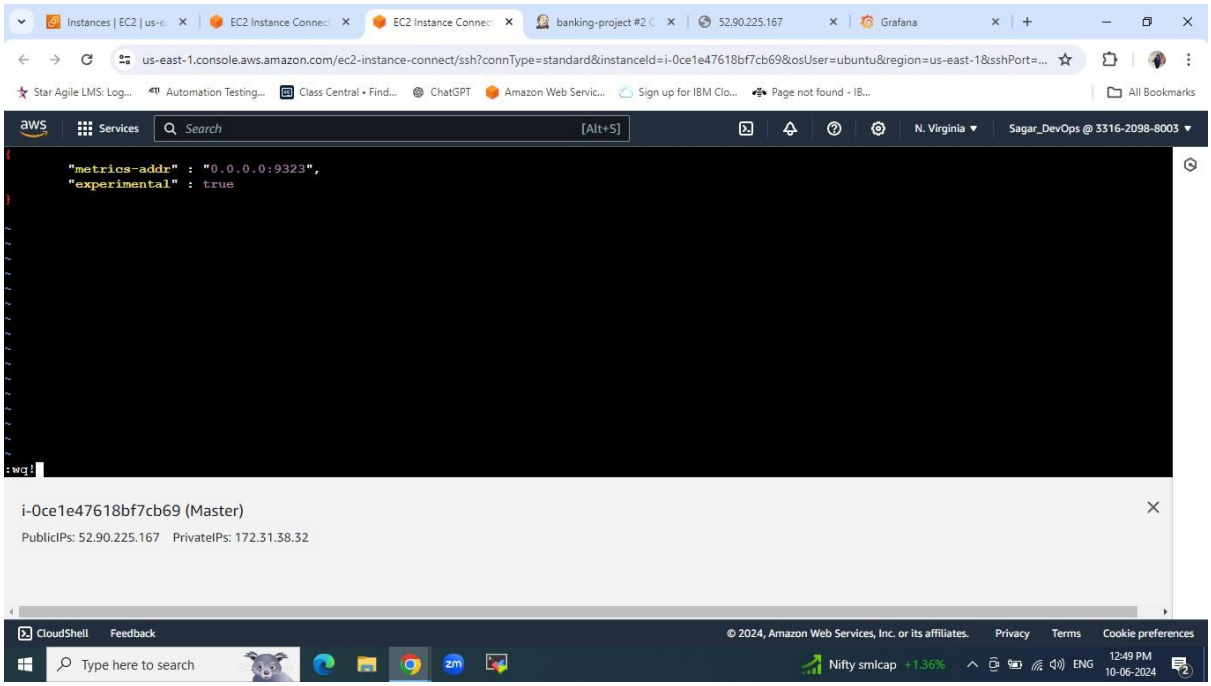
cAdvisor is installed on another machine and running on 8083 port.



NodeExporter is installed running docker image.

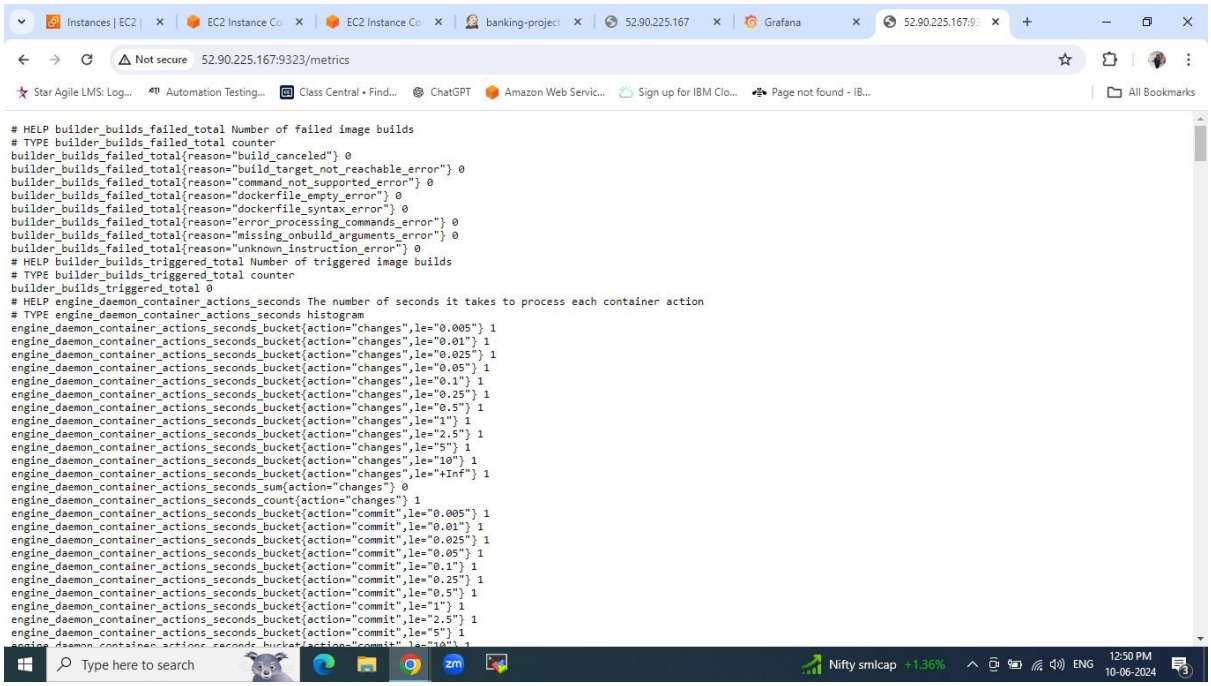


NodeExporter is installed and running on 9100 port.

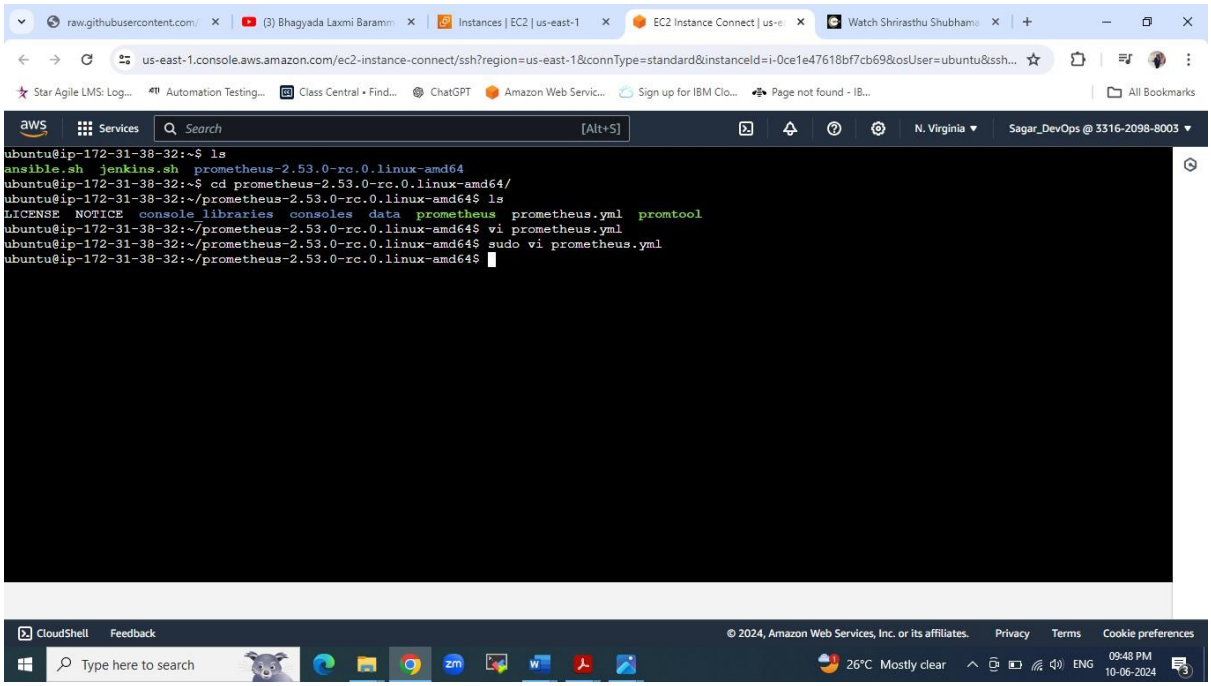


We need to tell the docker as well that Prometheus would be tracking the docker via port 9323.

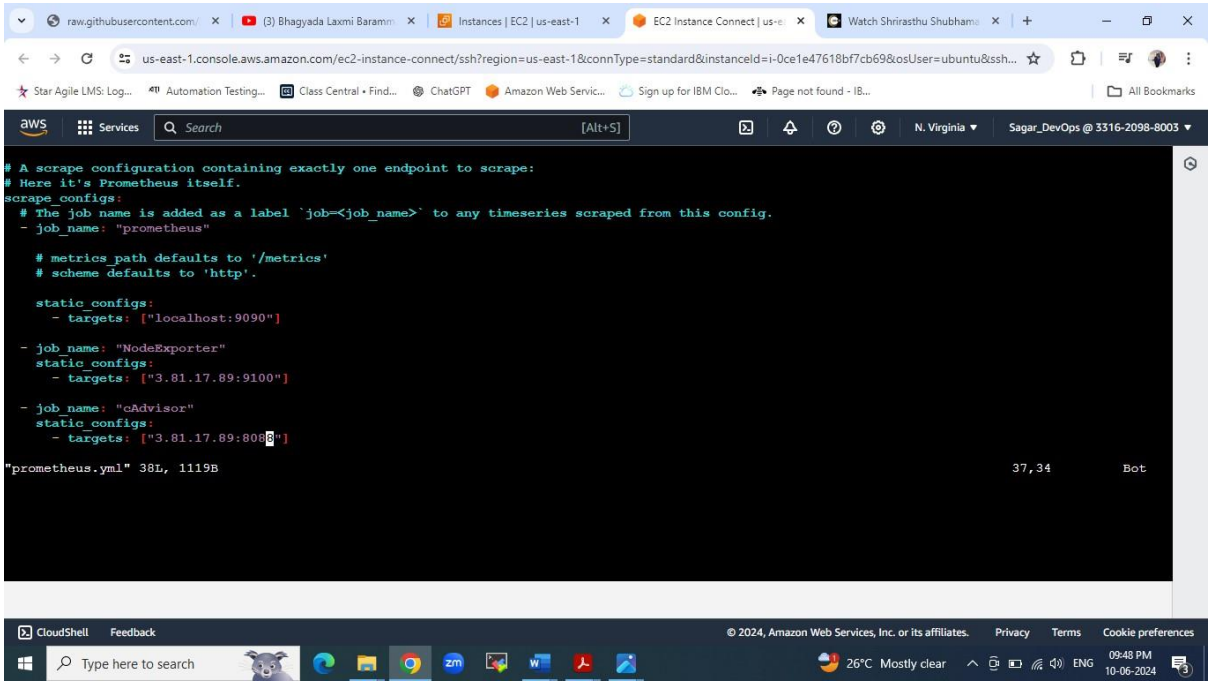
Go to vi /etc/docker/daemon.json and add the above code and save.



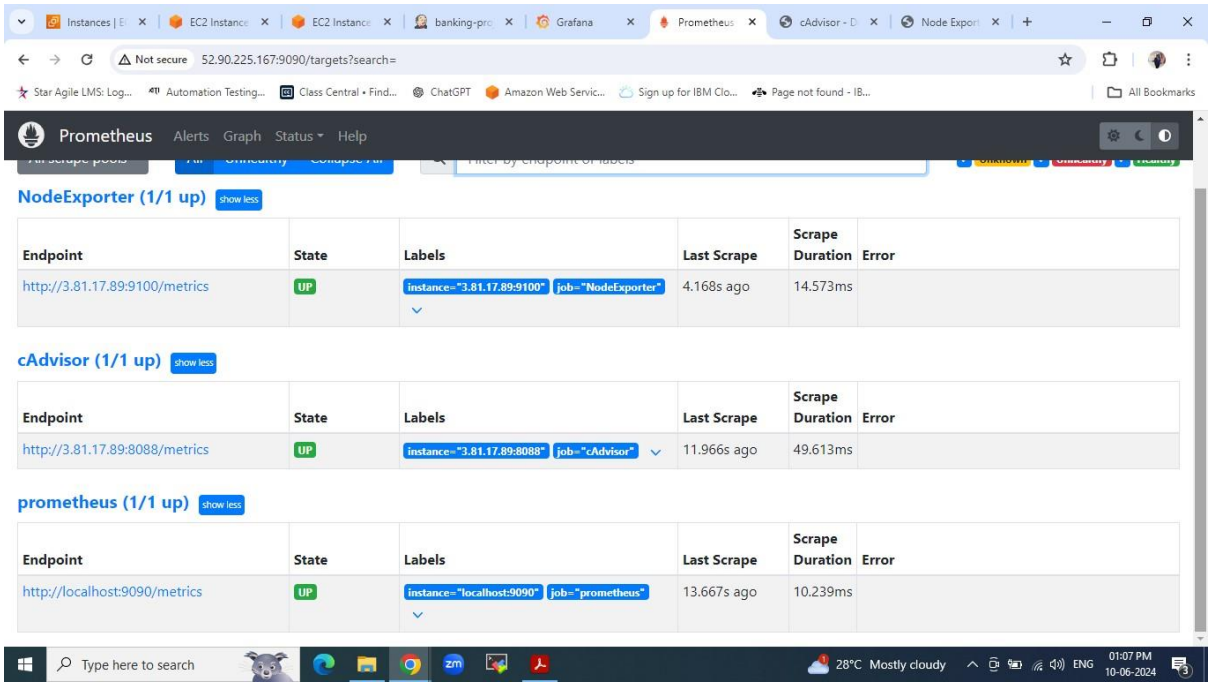
We can see the docker metrics via 9323 port.



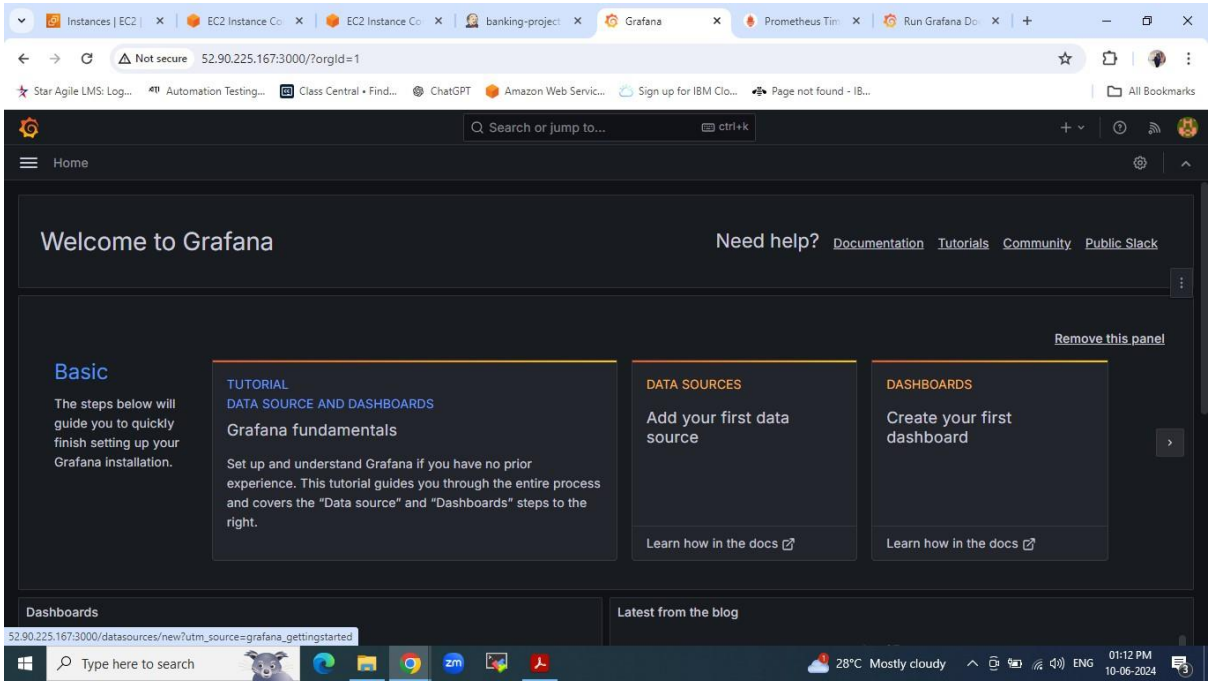
Go to Prometheus.yml and add NodeExporter and cAdvisor endpoints.



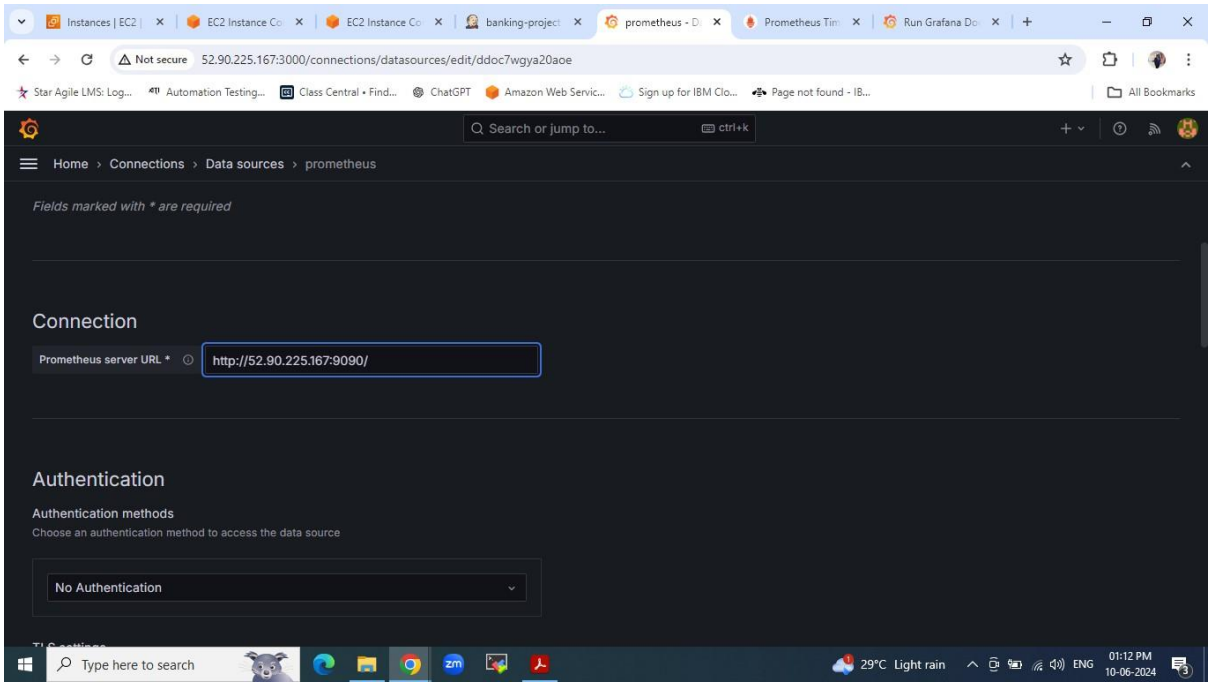
Endpoints are added in Prometheus.yml file.



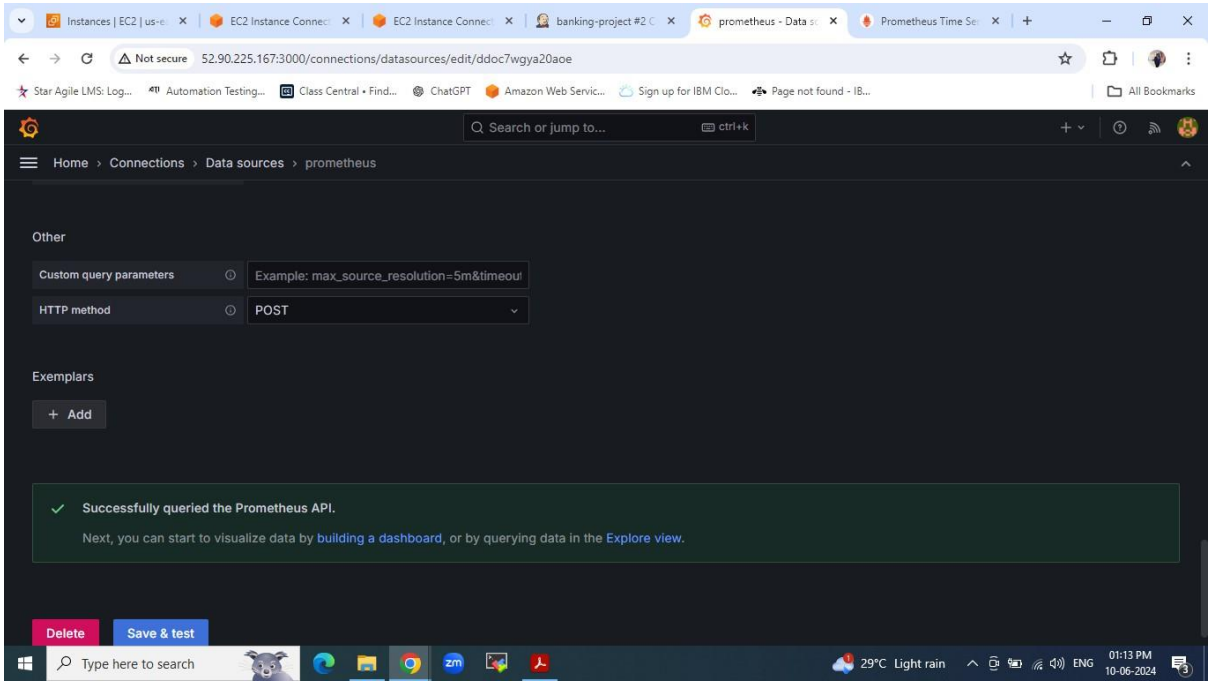
All the endpoints are added in Prometheus and we can access them in target tab.



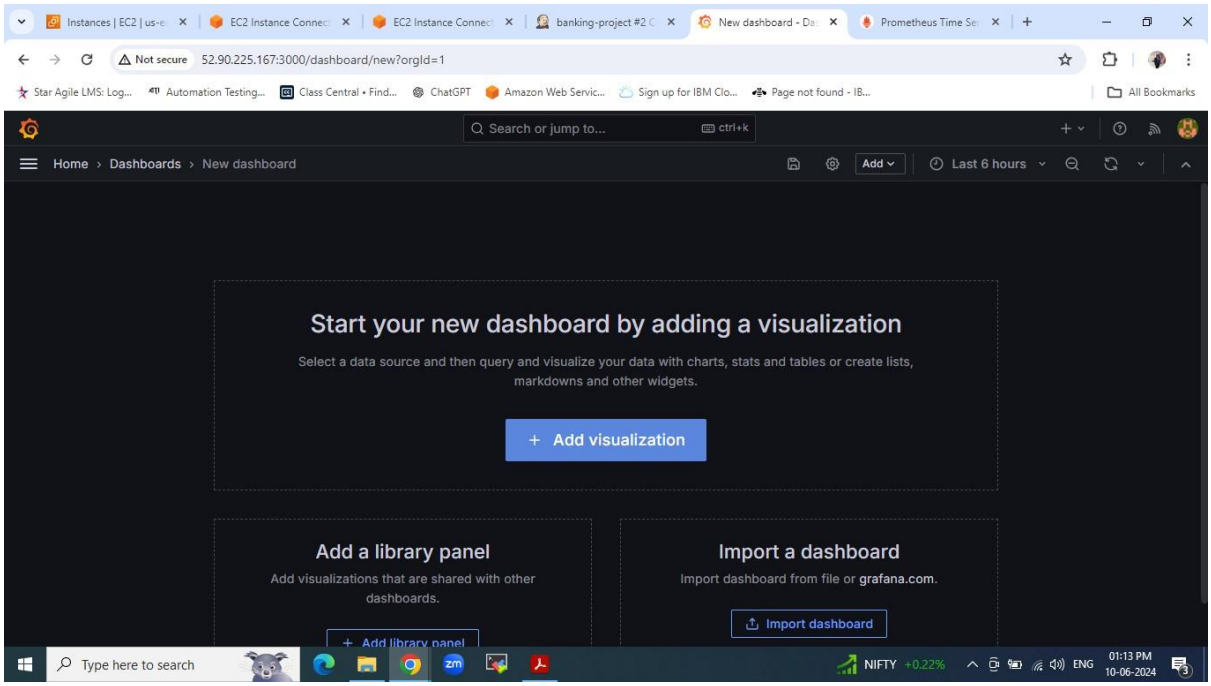
Now to go Grafana and add the data source to visualise the metrics.



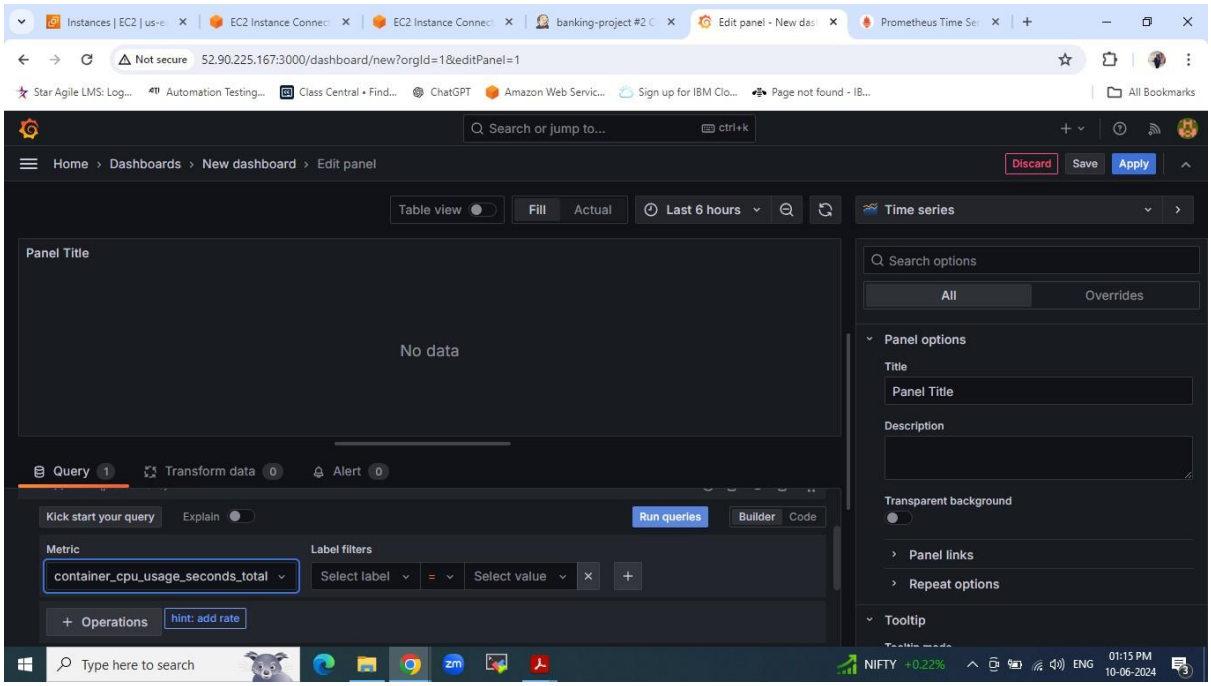
To connect Grafana with Prometheus add Prometheus URL in Grafana.



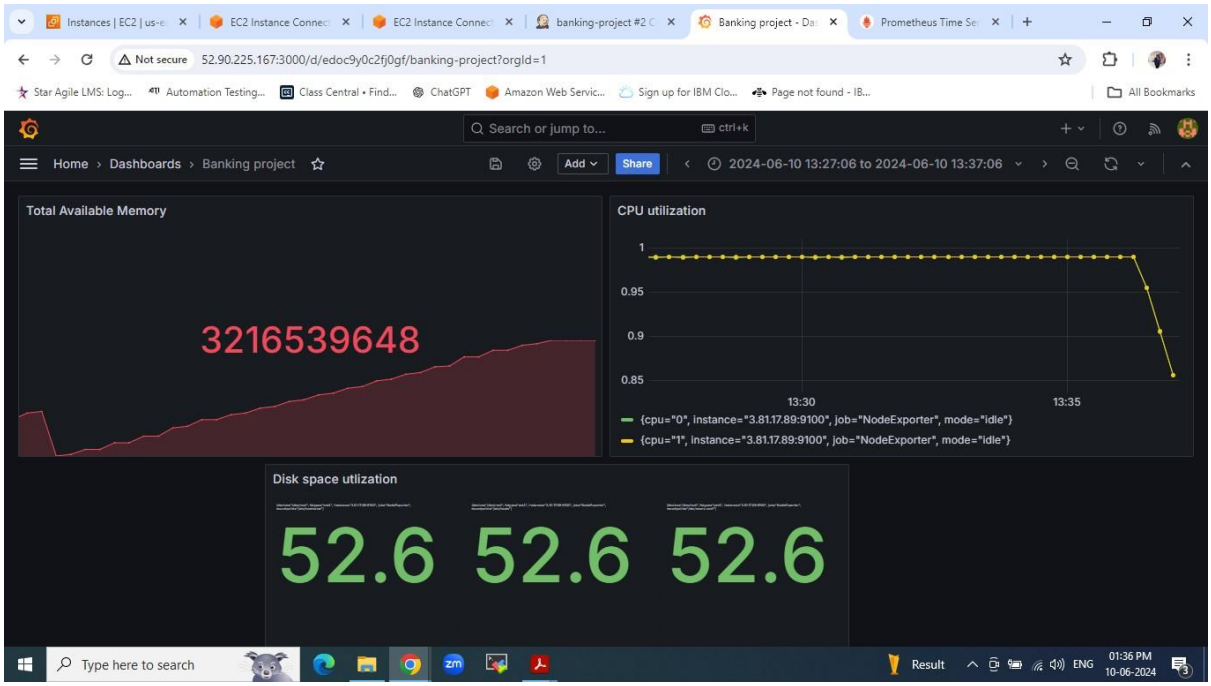
Tested the connection and it successful. We can build the dashboard now.



Click on Add visualization.



Go to metrics tab and add queries which we want to visualize.



Total Available Memory: The large number (3216539648) likely represents the available memory in bytes. The graph below the number indicates the memory usage trend over time. It appears to be increasing, suggesting that memory usage is steadily rising.

CPU Utilization: This section displays the CPU utilization for two CPUs (cpu="0" and cpu="1"). The yellow line represents the percentage of CPU utilization over time, showing a consistent usage near 95-100% before a sudden drop around 13:35. The high utilization indicates the CPUs were heavily used until the sudden drop, which might indicate a process termination or a system issue.

Disk Space Utilization: This section shows the disk space utilization percentages. Both values are 52.6%, indicating that about half of the disk space is currently used. The graph would help track changes in disk usage over time, but here, it simply shows a static value.

The conclusion of the FinanceMe project demonstrates the successful implementation of a comprehensive DevOps pipeline, transforming a monolithic banking application into a scalable microservice architecture using Spring Boot and an in-memory H2 database. The automation of the CI/CD process using tools like Git, Maven, Jenkins, Docker, Ansible, Selenium, and Terraform ensures efficient code compilation, testing, packaging, and deployment. Continuous monitoring with Prometheus and Grafana enhances the application's reliability and performance by providing real-time metrics. This project showcases the significant improvements in software delivery speed, quality, and feedback loops, addressing the initial challenges faced by FinanceMe and exemplifying a robust, automated, and scalable solution for modern banking applications.