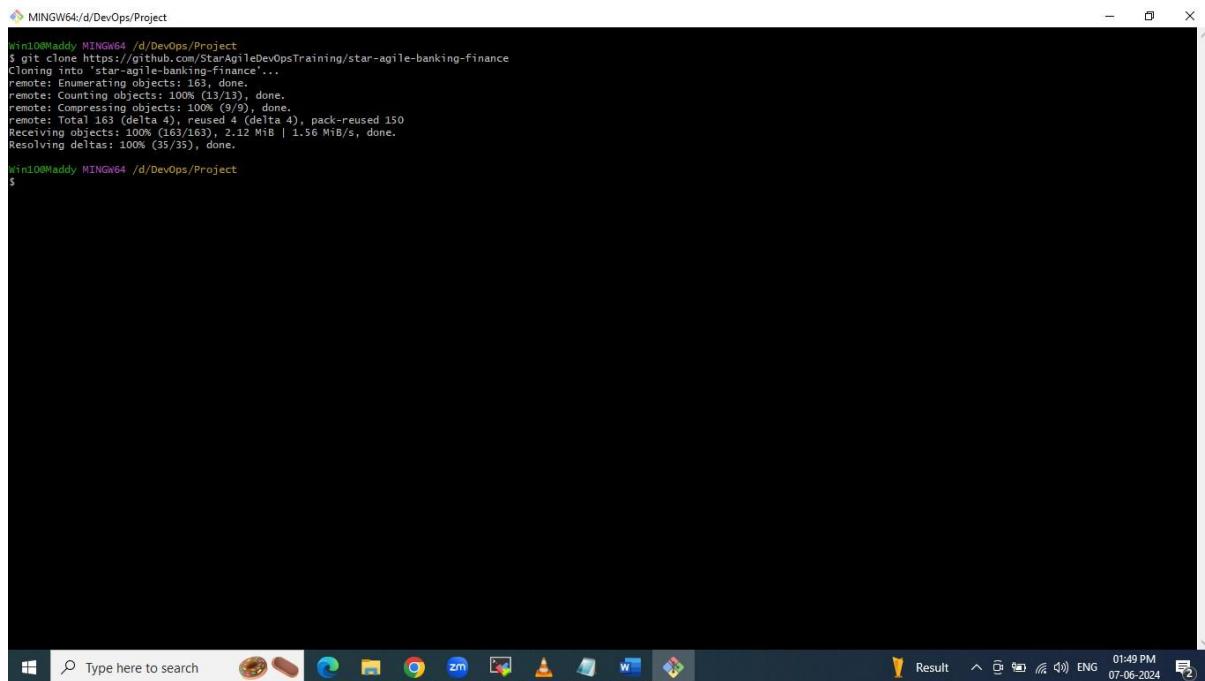


Name: Sagar Kulkarni

Batch: DevOps - SA2401021

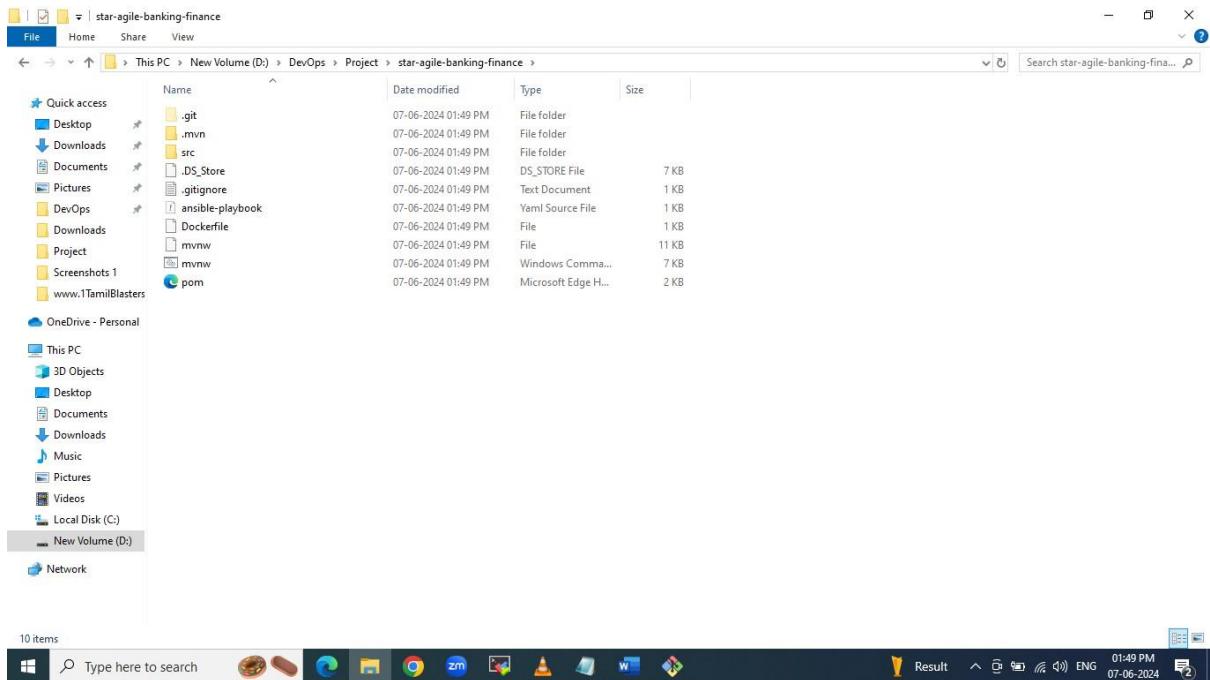
Banking and Finance Domain Project

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.

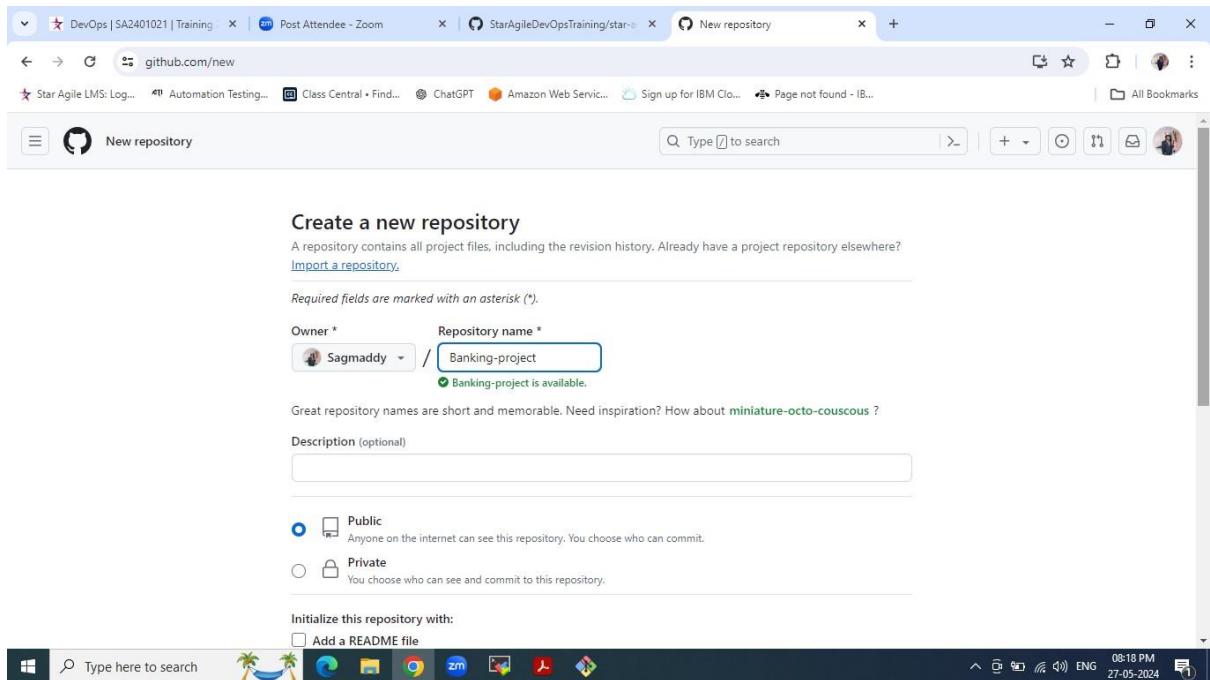


```
MINGW64/d/DevOps/Project
$ git clone https://github.com/StarAgileDevOpsTraining/star-agile-banking-Finance
Cloning into 'star-agile-banking-Finance'...
remote: Enumerating objects: 163, done.
remote: Counting objects: 100% (13/13), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 163 (delta 4), reused 4 (delta 4), pack-reused 150
Receiving objects: 100% (163/163), 2.12 MiB | 1.56 MiB/s, done.
Resolving deltas: 100% (35/35), done.
$
```

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



Repository is cloned into local system.

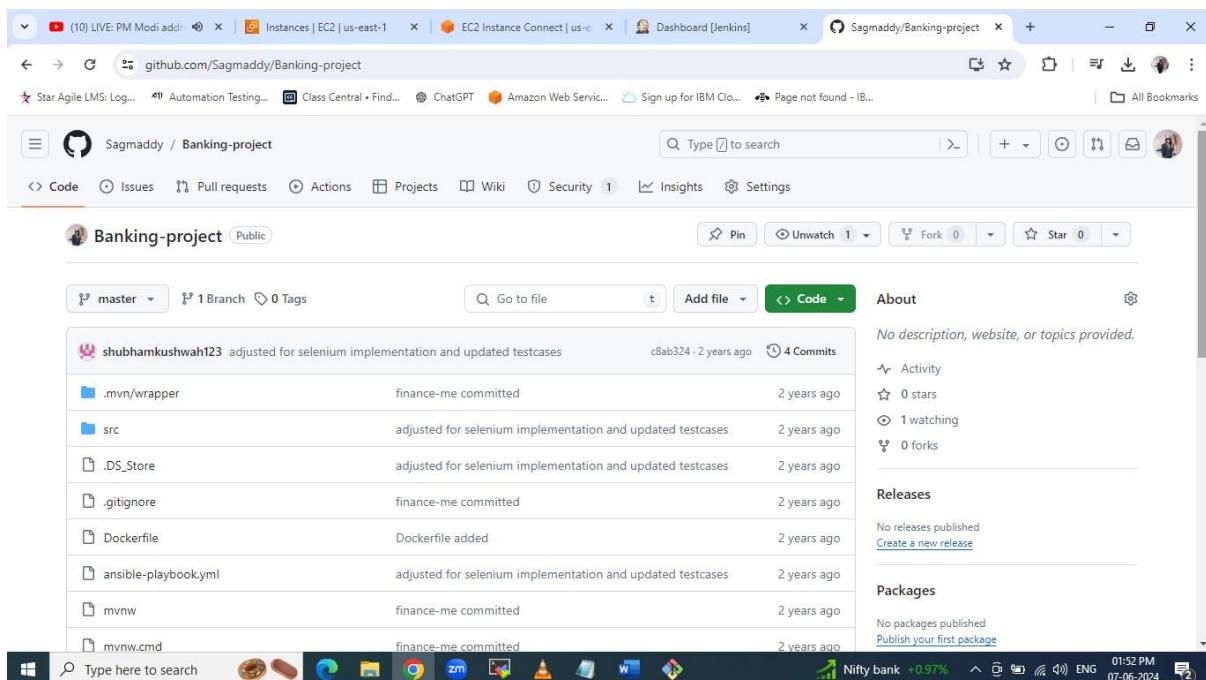


Created a new repo Banking-project

```
MINGW64:/d/DevOps/Project/star-agile-banking-finance
$ git remote remove origin
$ git remote add origin https://github.com/Sagmaddy/Banking-project
$ git push -u origin master
Enumerating objects: 163, done.
Counting objects: 100% (163/163), done.
Delta compression using up to 4 threads.
Compressing objects: 100% (100/100), done.
Writing objects: 100% (163/163), 2.12 MiB | 83.00 KiB/s, done.
Total 163 (delta 35), reused 163 (delta 35), pack-reused 0
remote: Resolving deltas: 100% (35/35), done.
To https://github.com/Sagmaddy/Banking-project
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
$ |
```

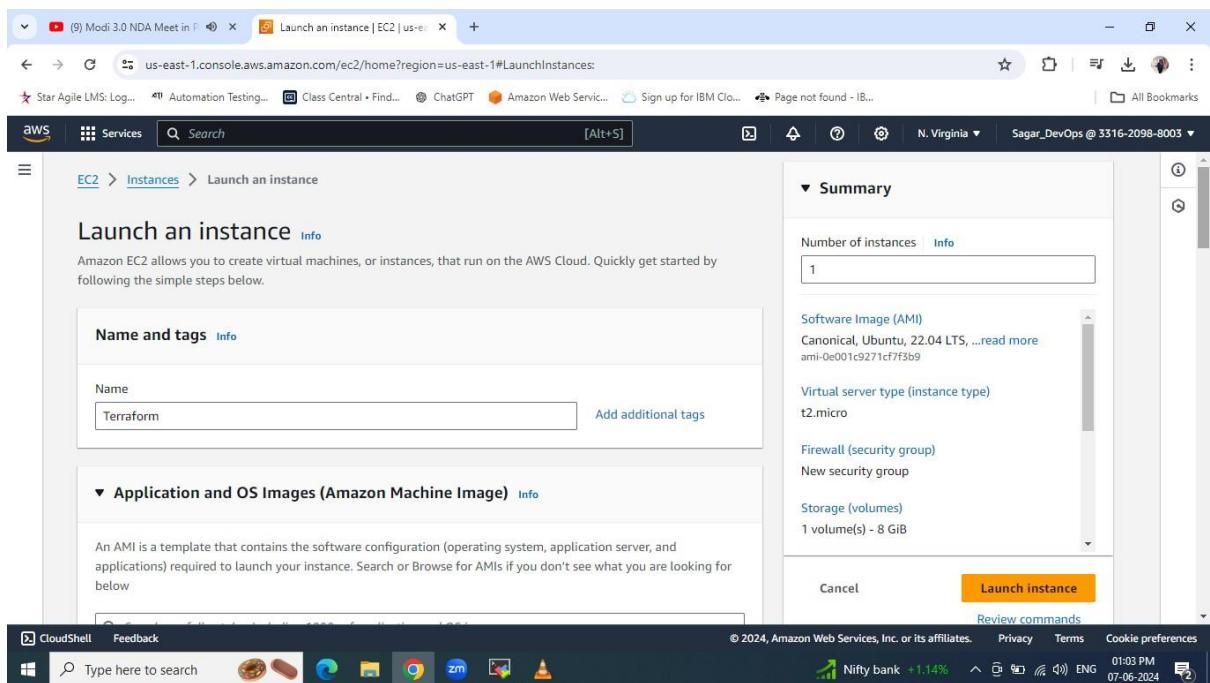
Removed the previous origin using git remote remove origin and added my repo as origin using git remote add origin and my repo URL.

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.

The screenshot shows the AWS Cloud Console interface. In the top navigation bar, there are several tabs and links. On the left, the 'Services' menu is open, showing various options like CloudWatch, Lambda, and Step Functions. The main content area is titled 'Quick Start' and displays a grid of AMI icons for Amazon Linux, macOS, Ubuntu, Windows, Red Hat, and SUSE. Below this, a specific AMI is selected: 'Ubuntu Server 22.04 LTS (HVM), SSD Volume Type'. The details show it's 'Free tier eligible' and includes Canonical, Ubuntu, 22.04 LTS, amd64 jammy image build on 2024-04-11. The 'Architecture' dropdown is set to '64-bit (x86)'. To the right, the 'Summary' section provides details about the instance: 1 instance, Software Image (AMI) Canonical, Ubuntu, 22.04 LTS, Virtual server type (instance type) t2.micro, Firewall (security group) New security group, and Storage (volumes) 1 volume(s) - 8 GiB. At the bottom right of the summary section is a prominent orange 'Launch instance' button.

Selected ubuntu 22.04 server.

This screenshot continues from the previous one, showing the configuration of the selected instance type. The 'Instance type' section shows the 't2.micro' option, which is described as 'Family: t2, 1 vCPU, 1 GiB Memory, Current generation: true'. It also lists base pricing for Windows, SUSE, RHEL, and Linux. Below this, the 'Key pair (login)' section is shown, where a new key pair named 'Banking' is being created. The 'Summary' section on the right remains the same, detailing 1 instance, Software Image (AMI) Canonical, Ubuntu, 22.04 LTS, Virtual server type (instance type) t2.micro, Firewall (security group) New security group, and Storage (volumes) 1 volume(s) - 8 GiB. The 'Launch instance' button is still visible at the bottom right.

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

Instances (1/1) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Available
Terraform	i-03c096e2b6f1b6cfb	Running	t2.micro	Initializing	View alarms	us-east-1

i-03c096e2b6f1b6cfb (Terraform)

Details Status and alarms Monitoring Security Networking Storage Tags

Instance summary Info

Instance ID	Public IPv4 address	Private IPv4 addresses
i-03c096e2b6f1b6cfb (Terraform)	52.204.100.247 open address	172.31.17.227

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Terraform instance is up and running.

```
root@ip-172-31-17-227:/home/ubuntu# terraform --version
Terraform v1.8.5
on linux_amd64
root@ip-172-31-17-227:/home/ubuntu#
```

i-03c096e2b6f1b6cfb (Terraform)

PublicIPs: 52.204.100.247 PrivateIPs: 172.31.17.227

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Installed terraform in the instance.

A screenshot of a Windows desktop environment. At the top, there's a taskbar with several pinned icons. Below the taskbar, a browser window is open to the AWS CloudShell interface. The URL bar shows the connection to an EC2 instance. The main content area of the browser displays a Terraform configuration file named 'ec2.tf'. The configuration defines two 'aws_instance' resources, each with specific parameters like AMI, instance type, and tags. Below the configuration, it shows the instance ID 'i-03c096e2b6f1b6cfb' and its public and private IP addresses. A separate CloudShell terminal window is visible at the bottom, showing the command 'terraform init' being run in a root shell on the EC2 instance.

```
resource "aws_instance" "web" {
  ami = "ami-0e001c9271cf7f3b9"
  instance_type = "t2.micro"
  count = 2
  key_name = "Banking"
  tags = {
    Name = "Master"
  }
}

"ec2.tf" 11L, 153B
```

i-03c096e2b6f1b6cfb (Terraform)
PublicIPs: 54.90.161.167 PrivateIPs: 172.31.17.227

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

A screenshot of a Windows desktop environment, similar to the previous one. The taskbar and browser setup are identical. The CloudShell terminal window at the bottom shows the command 'terraform init' being run again, this time with a different timestamp. The output of the command is visible in the terminal window.

```
root@ip-172-31-17-227:/home/ubuntu# terraform init
```

i-03c096e2b6f1b6cfb (Terraform)
PublicIPs: 52.204.100.247 PrivateIPs: 172.31.17.227

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

```
Initializing the backend...
Initializing provider plugins...
  - Finding latest version of hashicorp/aws...
  - Installing hashicorp/aws v5.53.0...
  - Installed hashicorp/aws v5.53.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

root@ip-172-31-17-227:/home/ubuntu#
```

i-03c096e2b6f1b6cfb (Terraform)
PublicIPs: 52.204.100.247 PrivateIPs: 172.31.17.227

Terraform is successfully initialized.

```
root@ip-172-31-17-227:/home/ubuntu# terraform apply
```

i-03c096e2b6f1b6cfb (Terraform)
PublicIPs: 52.204.100.247 PrivateIPs: 172.31.17.227

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

```
Enter a value: yes

aws instance.web[0]: Creating...
aws instance.web[1]: Creating...
aws instance.web[0]: Still creating... [10s elapsed]
aws instance.web[1]: Still creating... [10s elapsed]
aws instance.web[0]: Still creating... [20s elapsed]
aws instance.web[1]: Still creating... [20s elapsed]
aws instance.web[0]: Creation complete after 22s [id=i-0ce1e47618bf7cb69]
aws instance.web[0]: Still creating... [30s elapsed]
aws instance.web[0]: Still creating... [40s elapsed]
aws instance.web[0]: Still creating... [50s elapsed]
aws instance.web[0]: Still creating... [1m0s elapsed]
aws instance.web[0]: Still creating... [1m10s elapsed]
aws instance.web[0]: Still creating... [1m20s elapsed]
aws instance.web[0]: Still creating... [1m30s elapsed]
aws instance.web[0]: Still creating... [1m40s elapsed]
aws instance.web[0]: Creation complete after 1m42s [id=i-01e874fd90bc71b9d]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
root@ip-172-31-17-227:/home/ubuntu#
```

i-03c096e2b6f1b6cfb (Terraform)
PublicIPs: 52.204.100.247 PrivateIPs: 172.31.17.227

2 t2.medium ubuntu machines are created.

Instances (2/3) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Available
Master	i-0ce1e47618bf7cb69	Running	t2.medium	Initializing	View alarms +	us-east-1
Terraform	i-03c096e2b6f1b6cfb	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1
Master	i-01e874fd90bc71b9d	Running	t2.medium	Initializing	View alarms +	us-east-1

2 instances selected

Monitoring

CPU utilization (%) Network in (bytes) Network out (bytes) Network packets in (bytes)

2 t2.medium ubuntu machines are named Master.

Instances (1/3) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Available
Master	i-0ce1e47618bf7cb69	Running	t2.medium	2/2 checks passed	View alarms +	us-east-1
Terraform	i-03c096e2b6f1b6cfb	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1
Ansible-host	i-01e874fd90bc71b9d	Running	t2.medium	Initializing	View alarms +	us-east-1

i-01e874fd90bc71b9d (Ansible-host)

Details | Status and alarms | Monitoring | Security | Networking | Storage | Tags

Instance summary

Instance ID: i-01e874fd90bc71b9d (Ansible-host) | Public IPv4 address: 3.93.62.178 | Private IPv4 addresses: 172.31.37.35 | Public IPv4 DNS: 3.93.62.178 | Instance state: Running

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

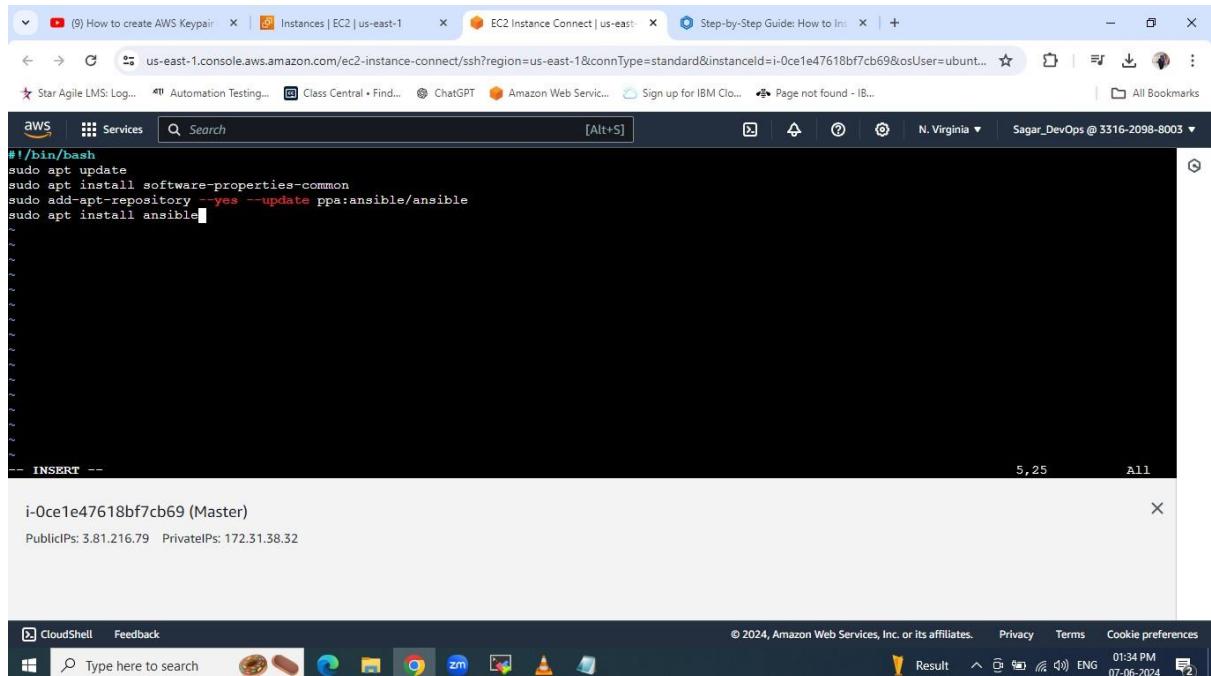
```

root@ip-172-31-42-243:/home/ubuntu# java --version
openjdk 17.0.10 2024-01-16
OpenJDK Runtime Environment (build 17.0.10+7-Ubuntu-122.04.1)
OpenJDK 64-Bit Server VM (build 17.0.10+7-Ubuntu-122.04.1, mixed mode, sharing)
root@ip-172-31-42-243:/home/ubuntu# mvn --version
Apache Maven 3.6.3
Maven home: /usr/share/maven
Java version: 17.0.10, vendor: Private Build, runtime: /usr/lib/jvm/java-17-openjdk-amd64
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "6.5.0-1017-aws", arch: "amd64", family: "unix"
root@ip-172-31-42-243:/home/ubuntu# ansible --version
ansible [core 2.16.7]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['root/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /root/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0] (/usr/bin/python3)
  jinja version = 3.0.3
  libyaml = True
root@ip-172-31-42-243:/home/ubuntu#

```

i-018a2d6a1711f4b95 (Master)
PublicIPs: 35.153.183.36 PrivateIPs: 172.31.42.243

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

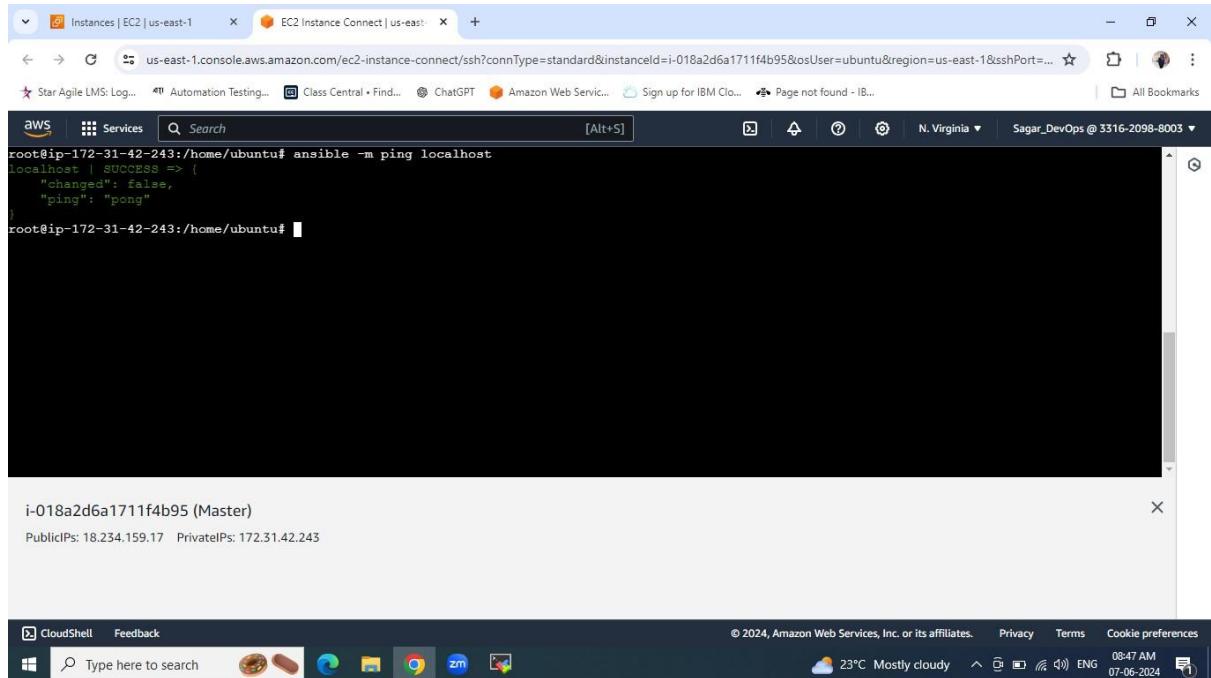


```
#!/bin/bash
sudo apt update
sudo apt install software-properties-common
sudo add-apt-repository --yes --update ppa:ansible/ansible
sudo apt install ansible
```

-- INSERT --

i-0ce1e47618bf7cb69 (Master)
PublicIPs: 3.81.216.79 PrivateIPs: 172.31.38.32

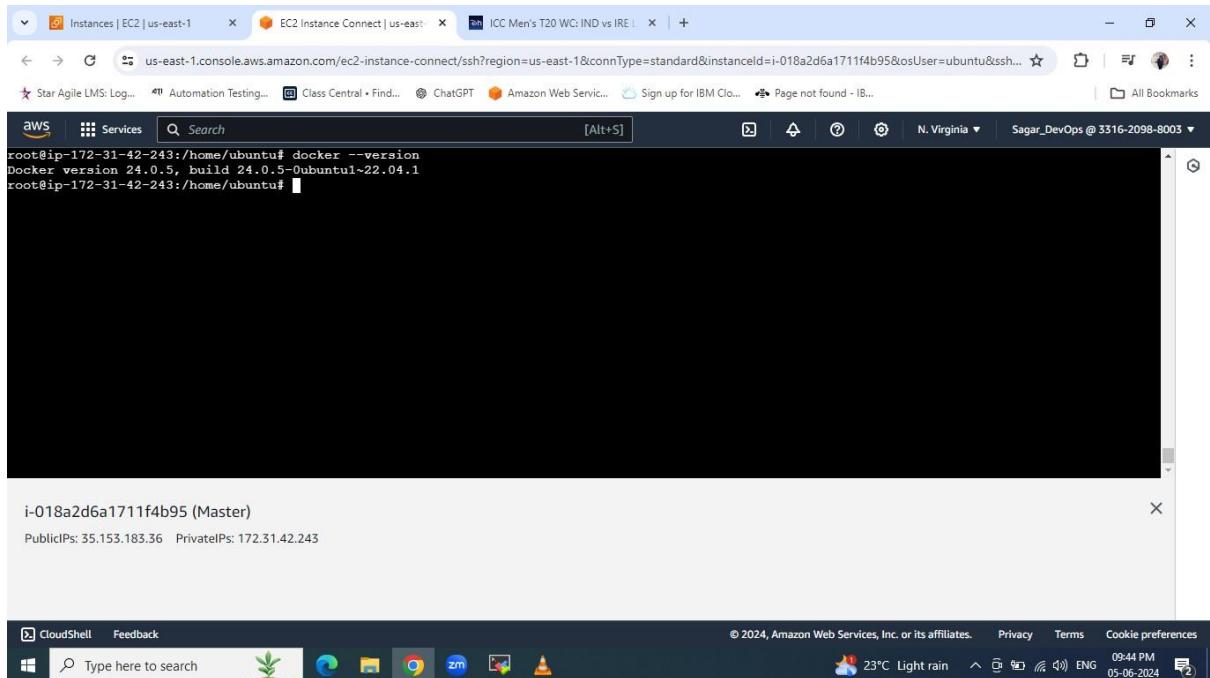
Commands to install Ansible in the machine.



```
root@ip-172-31-42-243:/home/ubuntu# ansible -m ping localhost
localhost | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
root@ip-172-31-42-243:/home/ubuntu#
```

i-018a2d6a1711f4b95 (Master)
PublicIPs: 18.234.159.17 PrivateIPs: 172.31.42.243

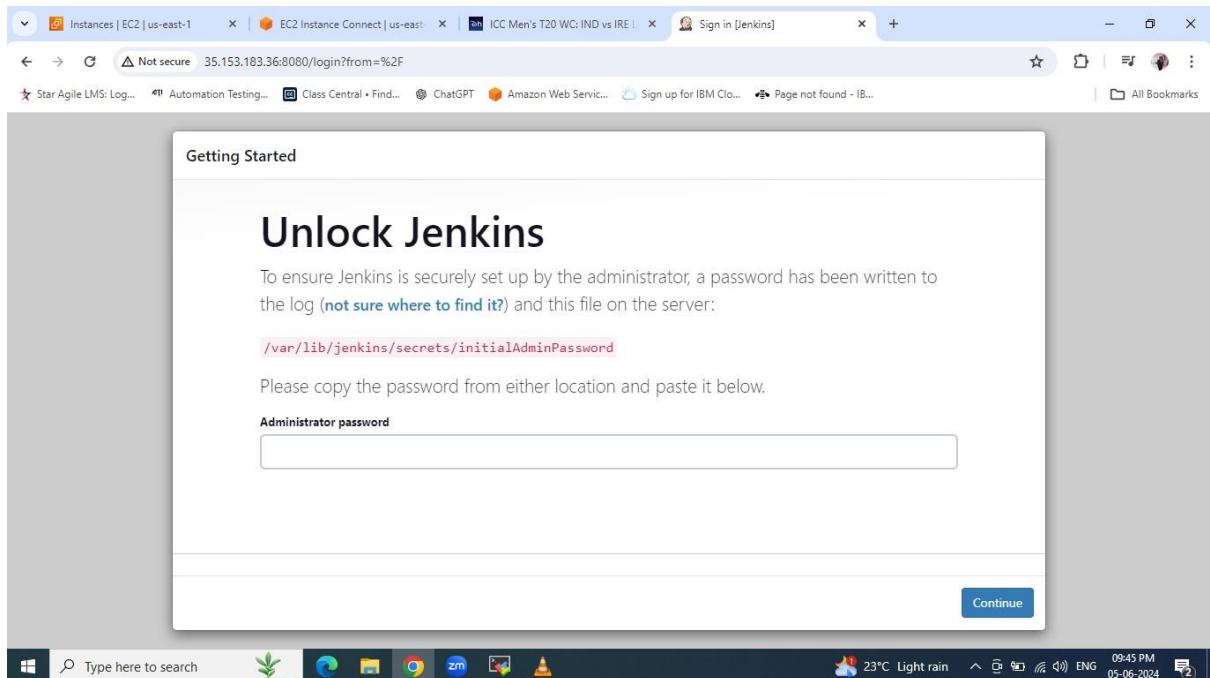
Ansible is installed and running in the machine.



```
root@ip-172-31-42-243:/home/ubuntu# docker --version
Docker version 24.0.5, build 24.0.5-Ubuntu1~22.04.1
root@ip-172-31-42-243:/home/ubuntu#
```

i-018a2d6a1711f4b95 (Master)
PublicIPs: 35.153.183.36 PrivateIPs: 172.31.42.243

Docker is installed in Master machine.



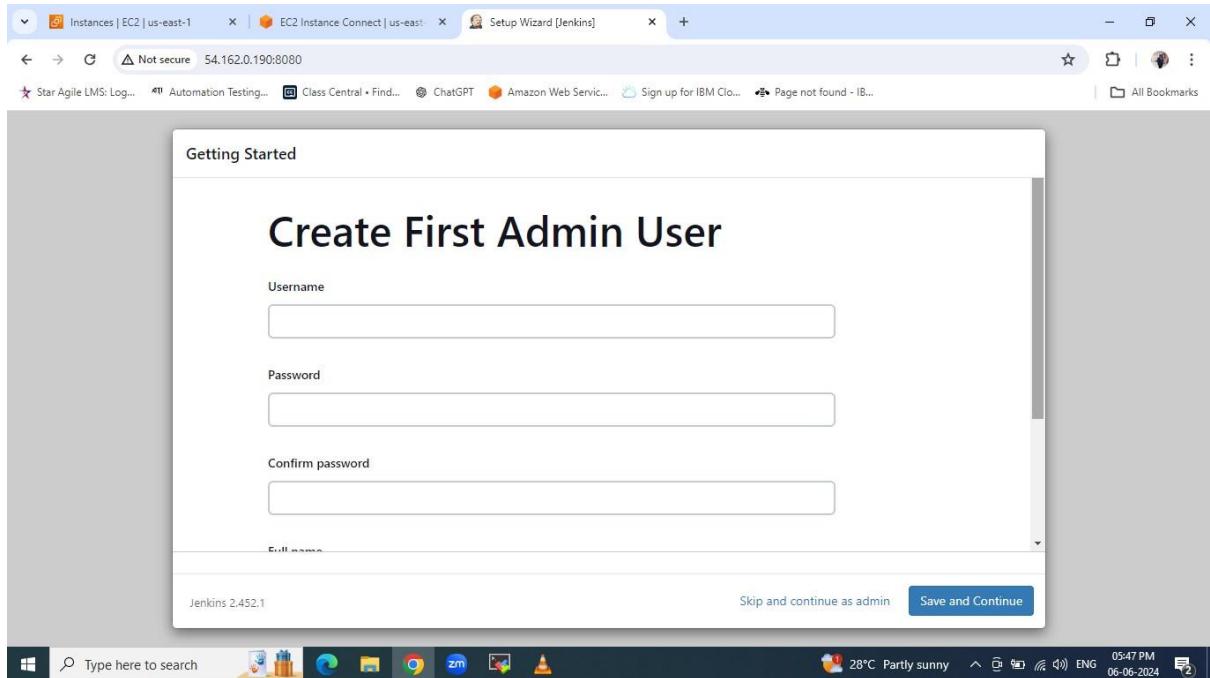
The screenshot shows a browser window with the URL `35.153.183.36:8080/login?from=%2F`. The page title is "Getting Started" and the main heading is "Unlock Jenkins". The text on the page reads: "To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server: `/var/lib/jenkins/secrets/initialAdminPassword`". Below this, there is a placeholder text area labeled "Administrator password" with a "Continue" button at the bottom right.

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

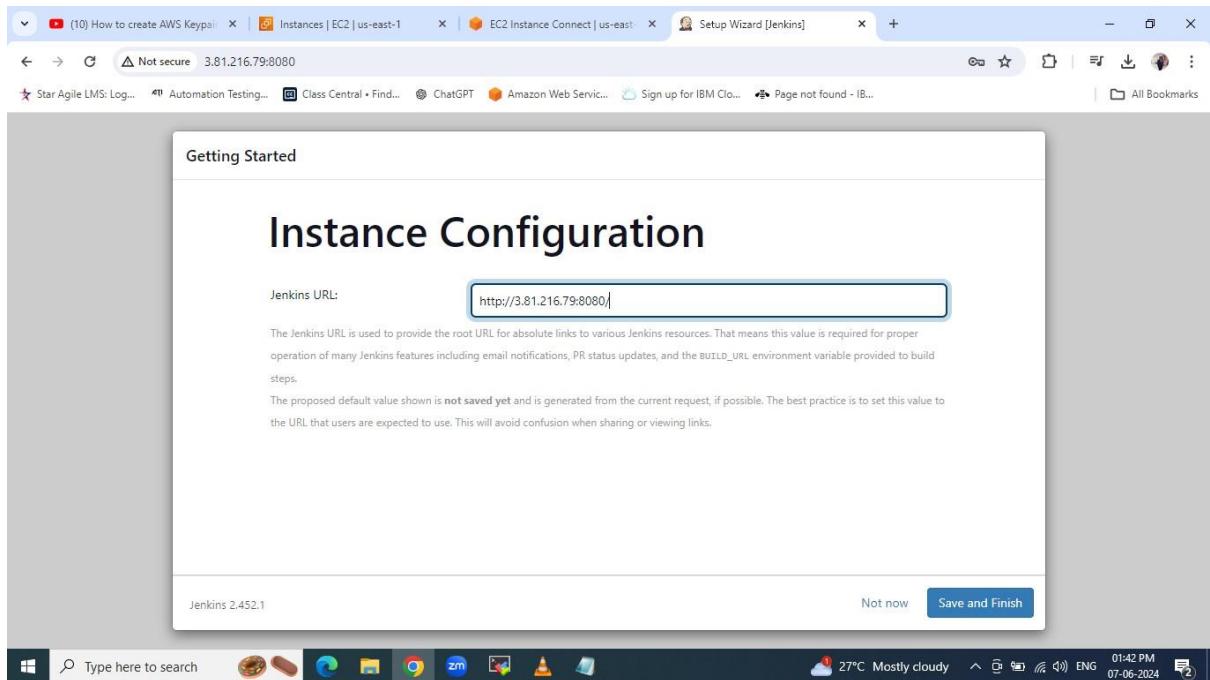
```
root@ip-172-31-42-243:/home/ubuntu# cat /var/lib/jenkins/secrets/initialAdminPassword
c96ff5d40b7542088dd9b932d4eac1d5
root@ip-172-31-42-243:/home/ubuntu#
```

i-018a2d6a1711f4b95 (Master)
Public IPs: 54.162.0.190 Private IPs: 172.31.42.243

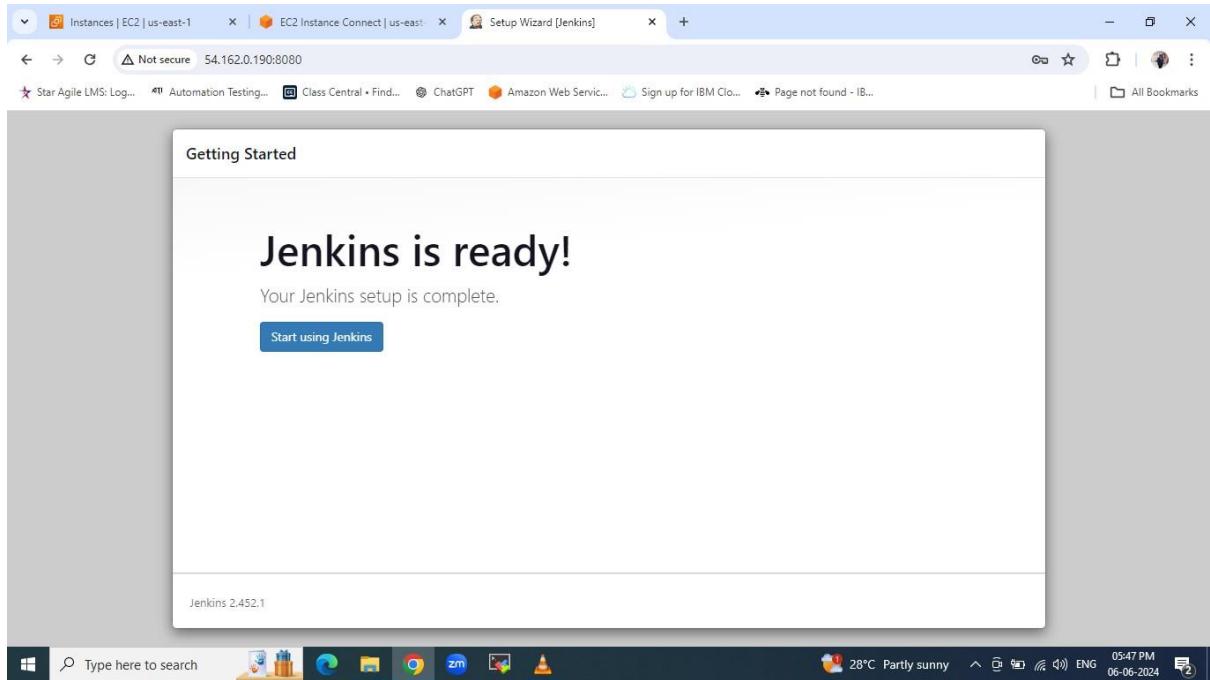
Using above command copied Administrator password.



Setup the Admin user.



Verified the Jenkins instance URL.



Jenkins is ready to use.

```
root@ip-172-31-42-243:/home/ubuntu# sudo usermod -aG docker jenkins
root@ip-172-31-42-243:/home/ubuntu# service jenkins restart
root@ip-172-31-42-243:/home/ubuntu#
```

i-018a2d6a1711f4b95 (Master)
PublicIPs: 54.162.0.190 PrivateIPs: 172.31.42.243

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.

```
GNU nano 6.2 /etc/sudoers.tmp *

# User alias specification
# Cmnd alias specification
# User privilege specification
root    ALL=(ALL:ALL) ALL
jenkins ALL=(ALL:ALL) NOPASSWD: ALL
# Members of the admin group may gain root privileges
%admin  ALL=(ALL) ALL

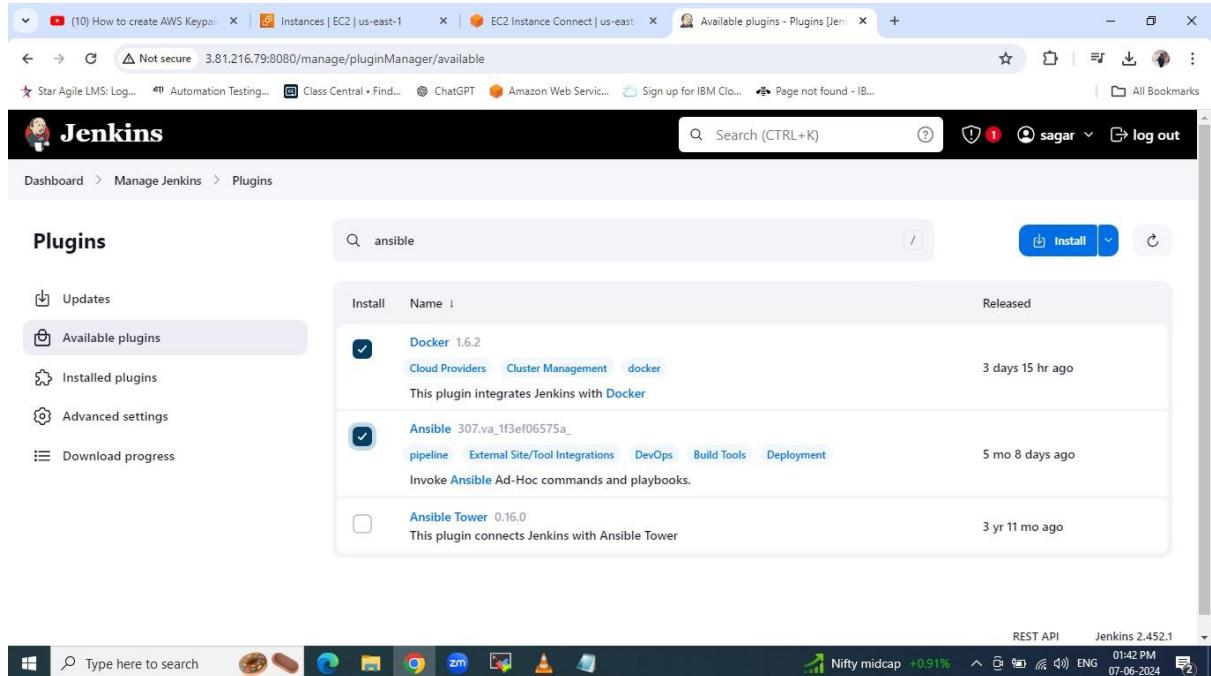
# Allow members of group sudo to execute any command
%sudo  ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "@include" directives:
@includedir /etc/sudoers.d

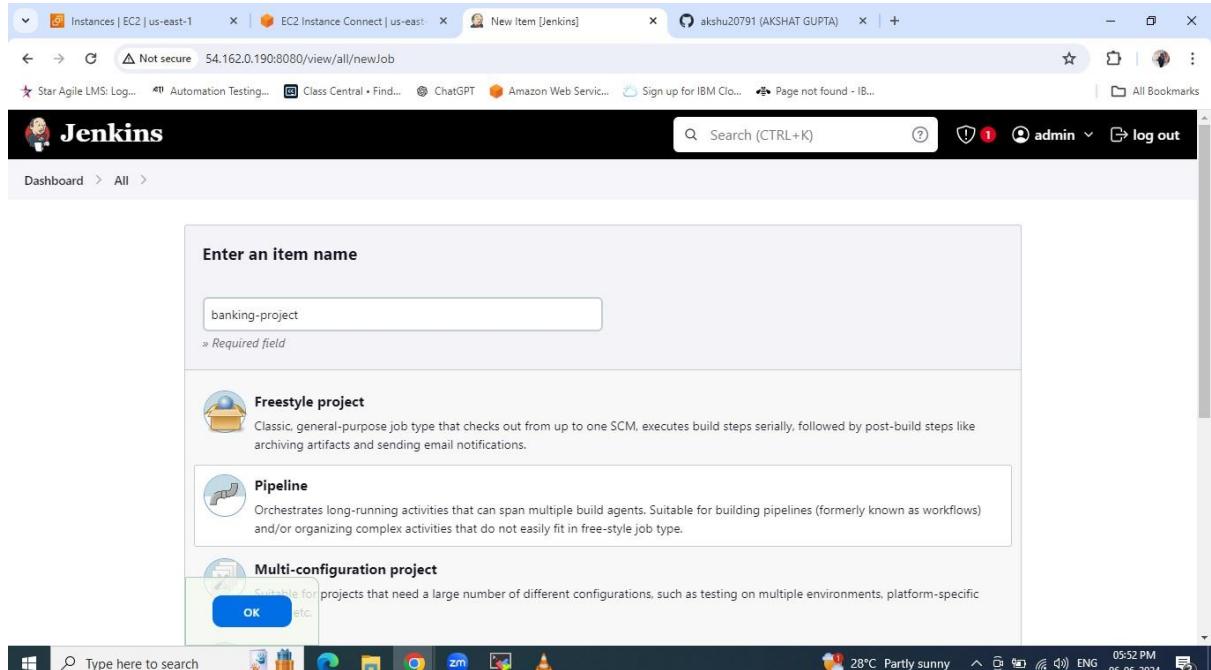
^G Help      ^C Write Out   ^W Where Is   ^R Cut          ^I Execute   ^C Location   M-U Undo   M-A Set Mark   M-J To Bracket
^X Exit      ^R Read File  ^V Replace   ^T Paste        ^J Justify   ^L Go To Line  M-B Redo   M-C Copy      M-G Where Was

i-0ce1e47618bf7cb69 (Master)
PublicIPs: 3.81.216.79 PrivateIPs: 172.31.38.32
```

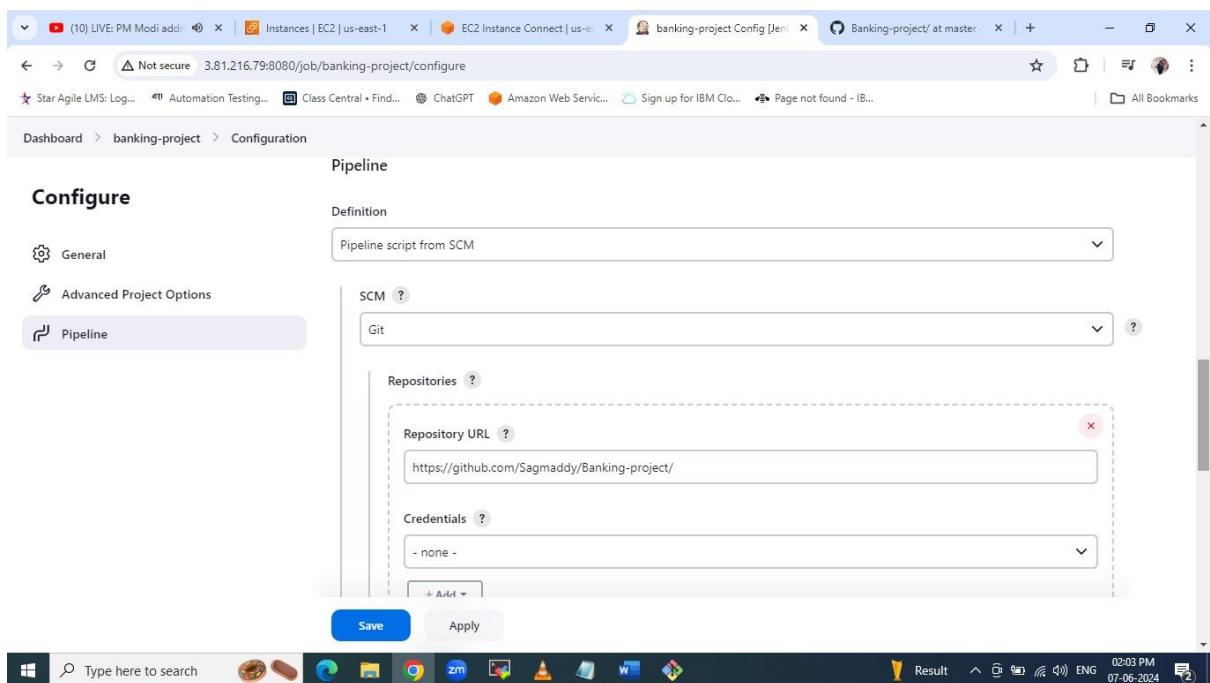
In the visudo file, add an additional line so that Jenkins doesn't require a password to execute Docker commands.



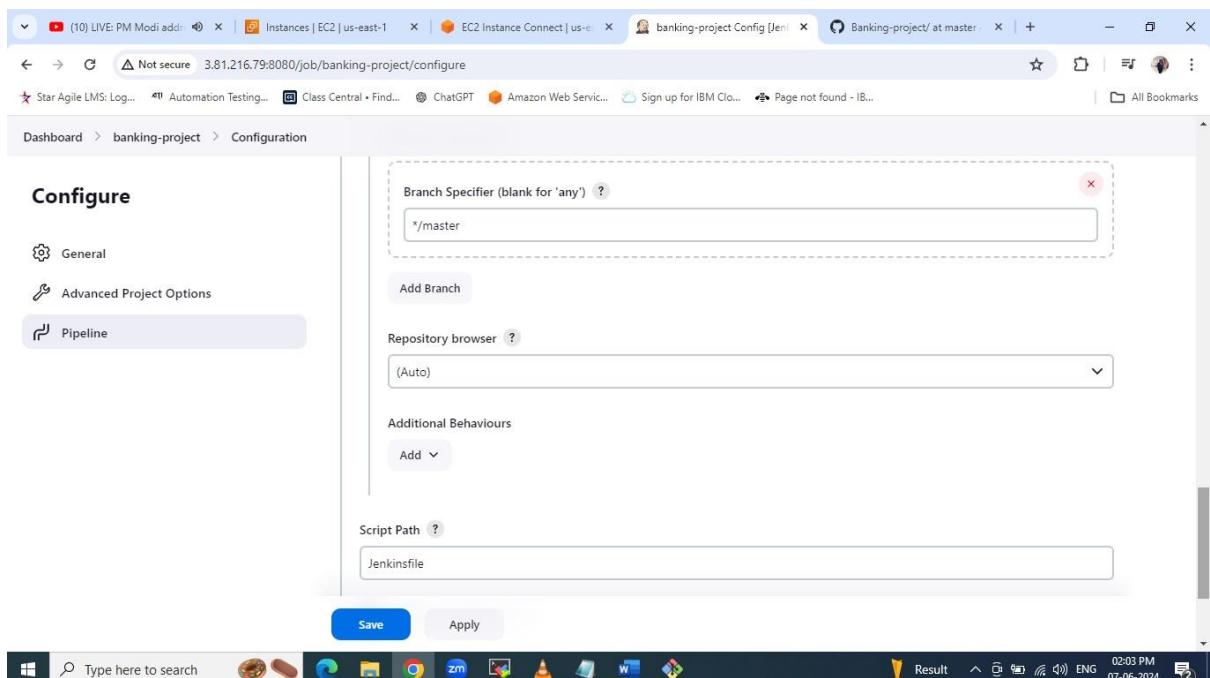
After setting up Jenkins, go to "Manage Jenkins" and install the Docker and Ansible plugins.



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

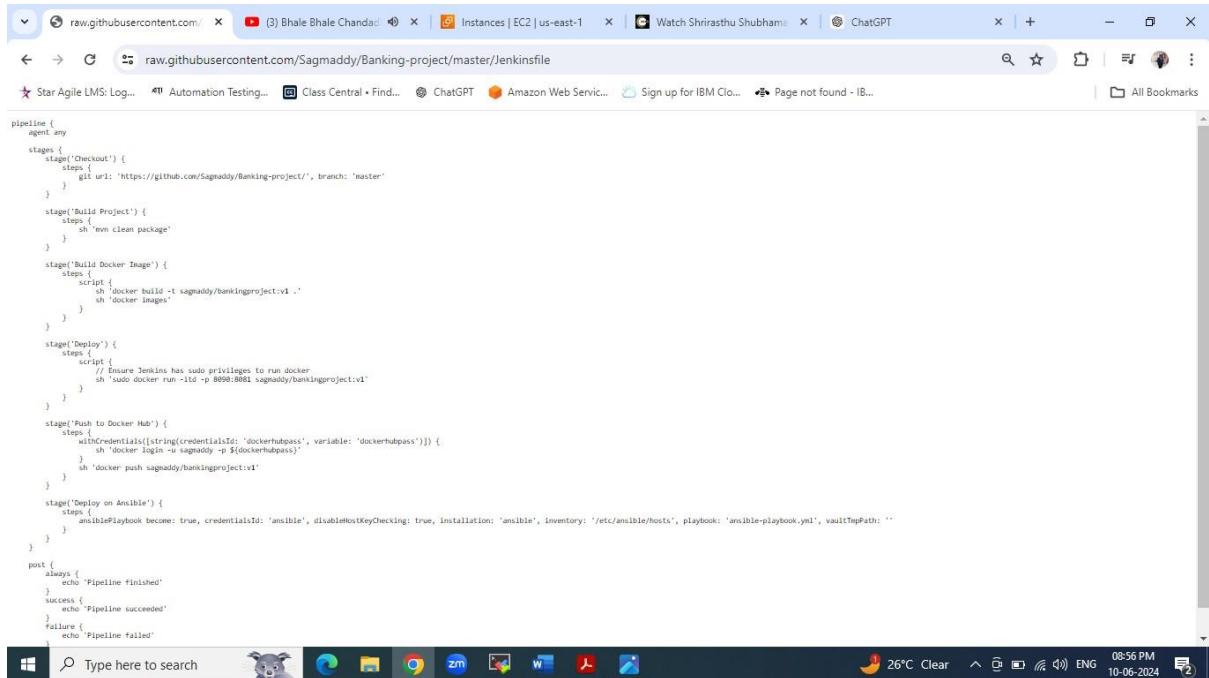


Go to pipeline step and select Pipeline script from SCM i.e., from Git. Provided the git URL so that Jenkins will fetch the project from above URL.



Select master branch and Script path Jenkinsfile.

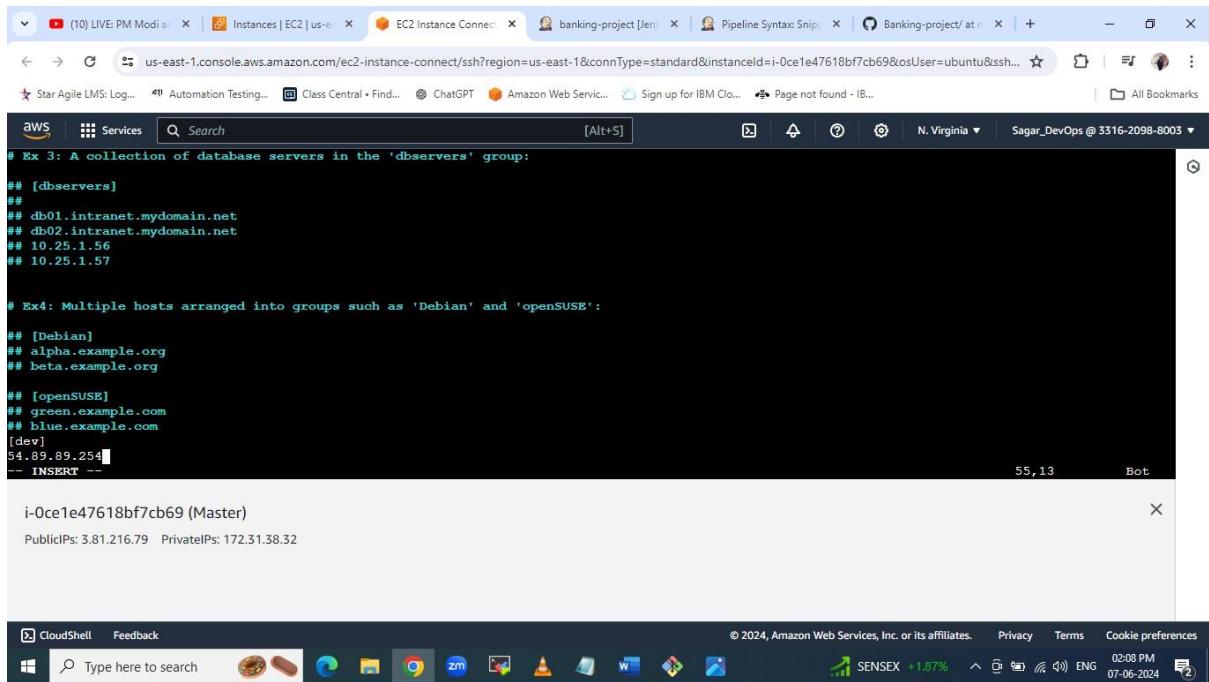
A Jenkinsfile is a text file that contains the definition of a Jenkins pipeline and is used to describe the stages and steps of your continuous integration and continuous delivery (CI/CD) pipeline.



The screenshot shows a Microsoft Edge browser window with multiple tabs open. The active tab displays the Jenkinsfile for the 'Sagmaddy/Banking-project' repository on GitHub. The Jenkinsfile is a Groovy script defining a pipeline with several stages: Checkout, Build Project, Build Docker Image, Deploy, Push to Docker Hub, Deploy on Ansible, and a final post section. The pipeline uses Jenkins credentials to log in to Docker Hub and pushes the Docker image to it. It also deploys the image to an Ansible host using a playbook located at '/etc/ansible/hosts'. The Jenkinsfile ends with a post section that echoes 'Pipeline finished' for success or failure.

```
pipeline {
    agent any
    stages {
        stage('Checkout') {
            steps {
                git url: 'https://github.com/Sagmaddy/Banking-project/', branch: 'master'
            }
        }
        stage('Build Project') {
            steps {
                sh 'mvn clean package'
            }
        }
        stage('Build Docker Image') {
            steps {
                script {
                    sh 'docker build -t sagmaddy/bankingproject:v1 .'
                    sh 'docker images'
                }
            }
        }
        stage('Deploy') {
            steps {
                script {
                    sh "Ensure Jenkins has sudo privileges to run docker"
                    sh "sudo docker run -itd -p 8090:8083 sagmaddy/bankingproject:v1"
                }
            }
        }
        stage('Push to Docker Hub') {
            steps {
                withCredentials([string(credentialsId: 'dockerhubpass', variable: 'dockerhubpass')]) {
                    sh 'docker login -u sagmaddy -p ${dockerhubpass}'
                    sh 'docker push sagmaddy/bankingproject:v1'
                }
            }
        }
        stage('Deploy on Ansible') {
            steps {
                ansiblePlaybook become: true, credentialsId: 'ansible', disableHostKeyChecking: true, installation: 'ansible', inventory: '/etc/ansible/hosts', playbook: 'ansible-playbook.yml', vaultTmpPath: ''
            }
        }
    }
    post {
        always {
            echo 'Pipeline finished'
        }
        success {
            echo 'Pipeline succeeded'
        }
        failure {
            echo 'Pipeline failed'
        }
    }
}
```

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



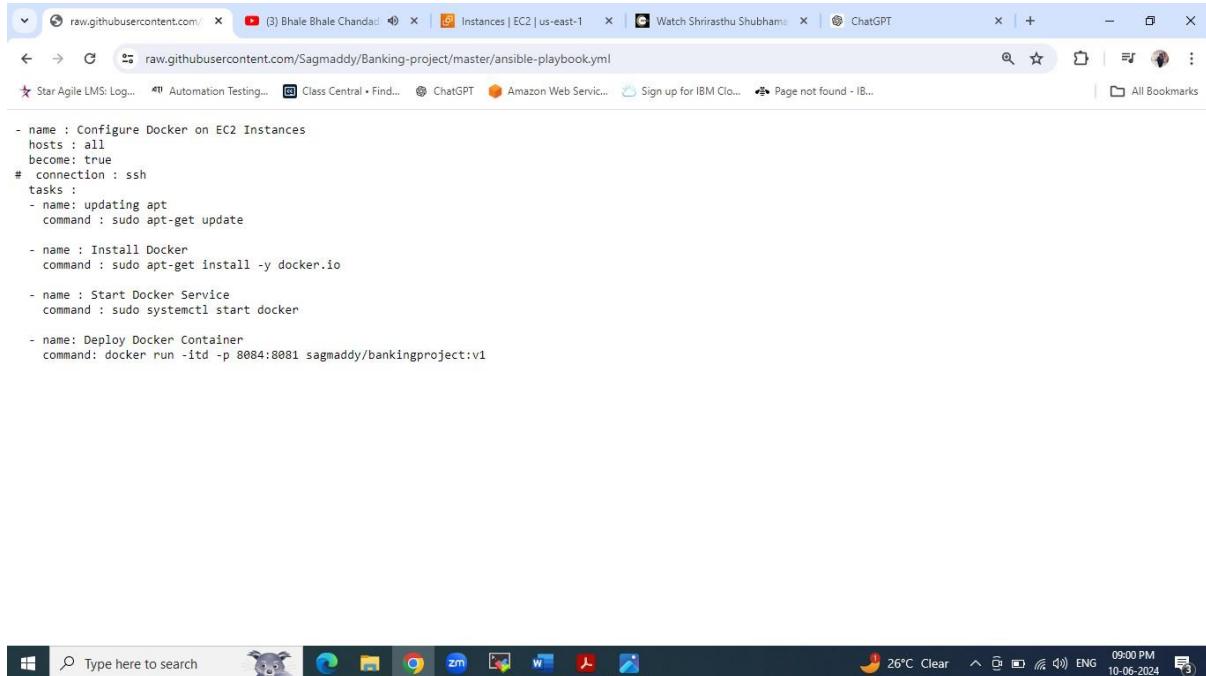
The screenshot shows an AWS CloudShell session. The terminal window displays an Ansible inventory file named 'inventory'. The file defines two groups: 'dbservers' and 'Debian'. The 'dbservers' group contains four hosts: 'db01.intranet.mydomain.net', 'db02.intranet.mydomain.net', '10.25.1.56', and '10.25.1.57'. The 'Debian' group contains three hosts: 'alpha.example.org', 'beta.example.org', and 'green.example.com'. A 'blue.example.com' host is also listed under the 'Debian' group. A 'dev' group is defined with one host: '54.89.89.254'. The 'openSUSE' group is empty. The CloudShell interface includes a search bar, navigation buttons, and status indicators for the session.

```
## Ex 3: A collection of database servers in the 'dbservers' group:
## [dbservers]
## db01.intranet.mydomain.net
## db02.intranet.mydomain.net
## 10.25.1.56
## 10.25.1.57

## Ex4: Multiple hosts arranged into groups such as 'Debian' and 'openSUSE':
## [Debian]
## alpha.example.org
## beta.example.org

## [openSUSE]
## green.example.com
## blue.example.com
[dev]
54.89.89.254
-- INSERT --
```

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.



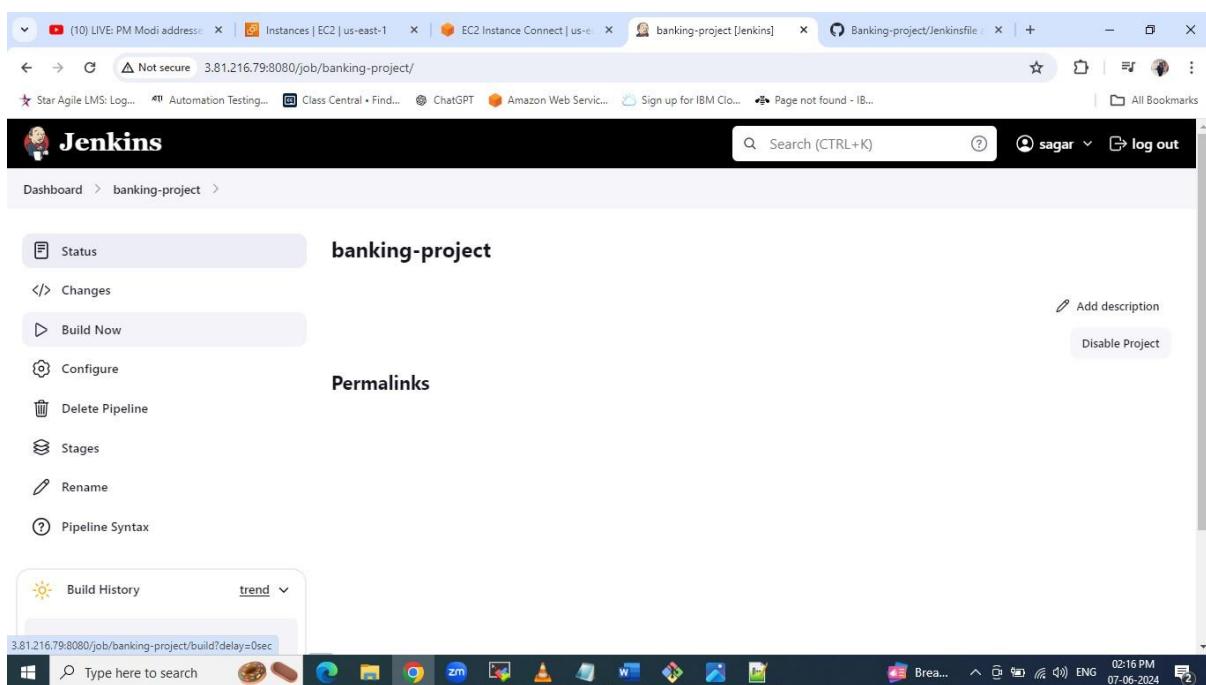
```
- name : Configure Docker on EC2 Instances
  hosts : all
  become: true
# connection : ssh
  tasks :
    - name: updating apt
      command : sudo apt-get update

  - name : Install Docker
    command : sudo apt-get install -y docker.io

  - name : Start Docker Service
    command : sudo systemctl start docker

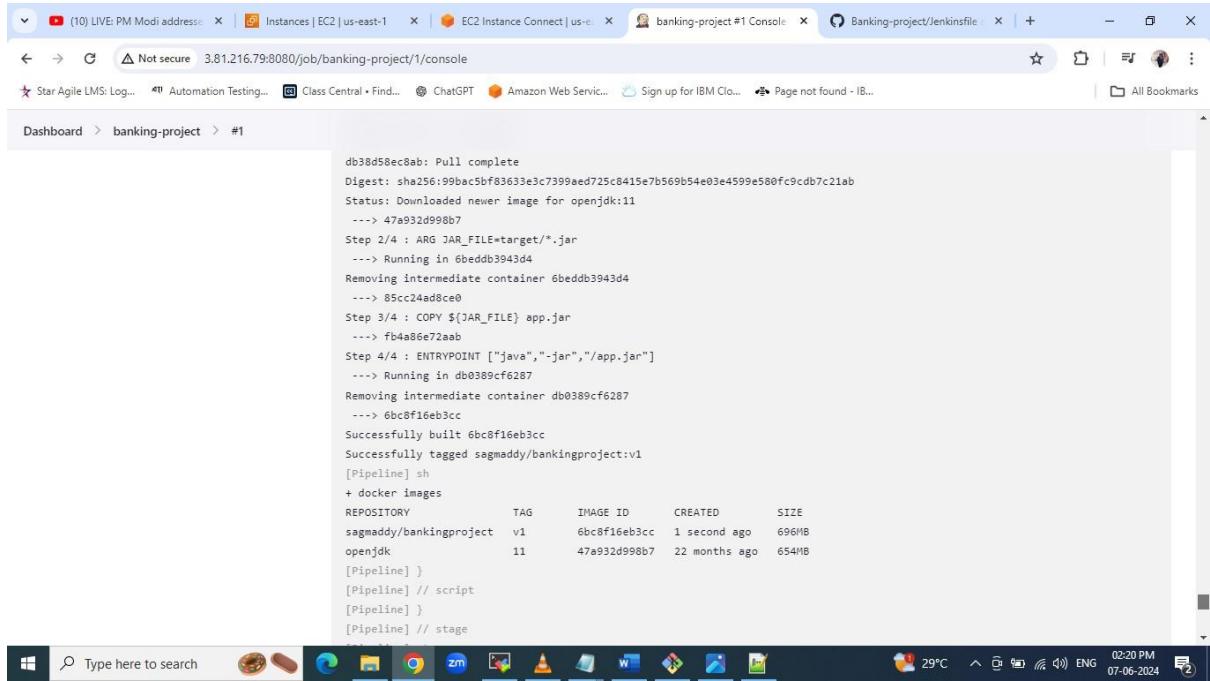
  - name: Deploy Docker Container
    command: docker run -itd -p 8084:8081 sagmaddy/bankingproject:v1
```

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



The screenshot shows a Jenkins dashboard for the 'banking-project'. The main navigation bar includes 'Dashboard', 'banking-project', and 'Permalinks'. On the left, there are links for 'Status', 'Changes', 'Build Now', 'Configure', 'Delete Pipeline', 'Stages', 'Rename', and 'Pipeline Syntax'. A 'Build History' section is visible at the bottom left. The taskbar at the bottom of the screen displays various application icons and system status indicators.

Build the pipeline project by clicking build now.



```
db38d58ec8ab: Pull complete
Digest: sha256:99bac5bf83633e3c7399aed725c8415e7b569b54e03e4599e580fc9cdb7c21ab
Status: Downloaded newer image for openjdk:11
--> 47a932d998b7
Step 2/4 : ARG JAR_FILE=target/*.jar
--> Running in 6beddb3943d4
Removing intermediate container 6beddb3943d4
--> 85cc24ad8ce0
Step 3/4 : COPY ${JAR_FILE} app.jar
--> fbaa86e72aab
Step 4/4 : ENTRYPOINT ["java","-jar","/app.jar"]
--> Running in db0389cf6287
Removing intermediate container db0389cf6287
--> 6bc8f16eb3cc
Successfully built 6bc8f16eb3cc
Successfully tagged sagmaddy/bankingproject:v1
[Pipeline] sh
+ docker images
REPOSITORY          TAG      IMAGE ID   CREATED        SIZE
sagmaddy/bankingproject  v1      6bc8f16eb3cc  1 second ago  696MB
openjdk              11      47a932d998b7  22 months ago  654MB
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] }
```

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.

```
[Pipeline] stage
[Pipeline] { (Deploy)
[Pipeline] script
[Pipeline] {
[Pipeline] sh
+ sudo docker run -itd -p 8090:8081 sagmaddy/bankingproject:v1
e4ba80f12e9d87d1f48885dabfb94af95ca68387da2186851890b6d2b5ad9a74
[Pipeline]
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Push to Docker Hub)
[Pipeline] withCredentials
Masking supported pattern matches of $dockerhubpass
[Pipeline] {
[Pipeline] sh
+ docker login -u sagmaddy -p ****
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
WARNING! Your password will be stored unencrypted in /var/lib/jenkins/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[Pipeline] }
```

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

sagmaddy/bankingproject

Updated about 9 hours ago

This repository does not have a description INCOMPLETE

This repository does not have a category INCOMPLETE

Docker commands

To push a new tag to this repository:

```
docker push sagmaddy/bankingproject:tagname
```

Tags

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
v1		Image	---	9 hours ago

[See all](#)

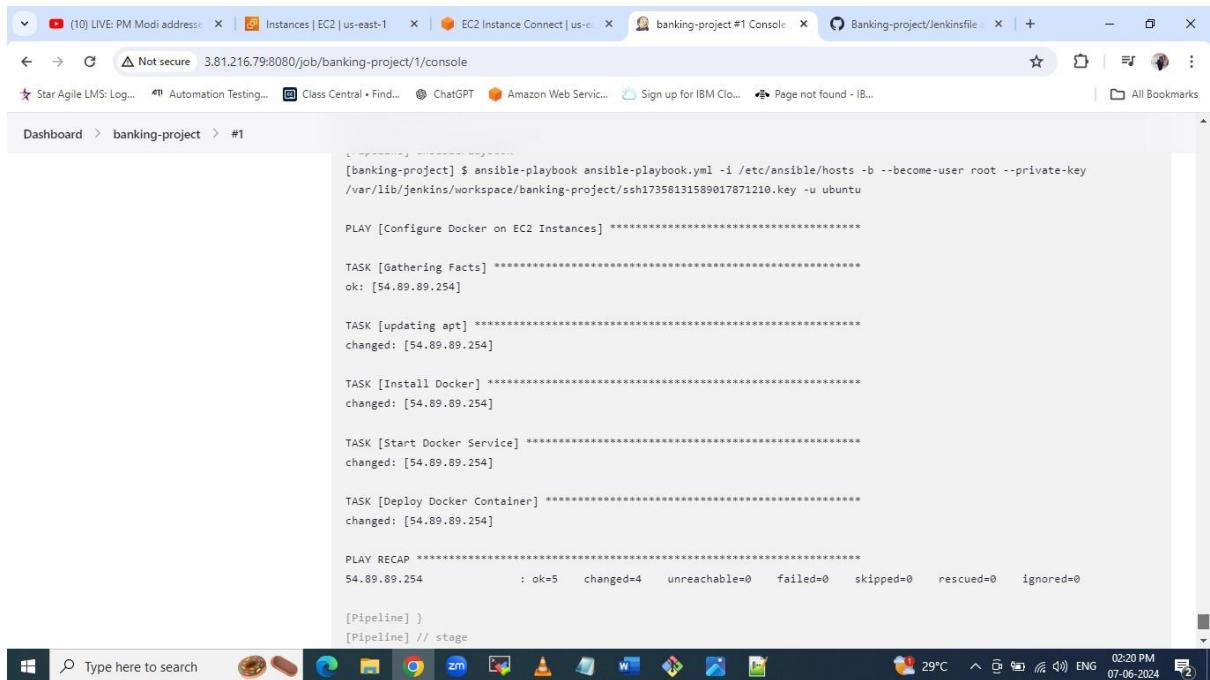
Automated Builds

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available with Pro, Team and Business subscriptions. [Read more about automated builds](#).

Upgrade

The image is pushed in docker hub repo.



The screenshot shows a Windows desktop environment. At the top is the taskbar with various icons. Below it is a browser window titled "banking-project #1 Console". The main content of the browser shows the output of an Ansible playbook execution:

```
[banking-project] $ ansible-playbook ansible-playbook.yml -i /etc/ansible/hosts -b --become-user root --private-key /var/lib/jenkins/workspace/banking-project/ssh17358131589017871210.key -u ubuntu

PLAY [Configure Docker on EC2 Instances] *****

TASK [Gathering Facts] *****
ok: [54.89.89.254]

TASK [updating apt] *****
changed: [54.89.89.254]

TASK [Install Docker] *****
changed: [54.89.89.254]

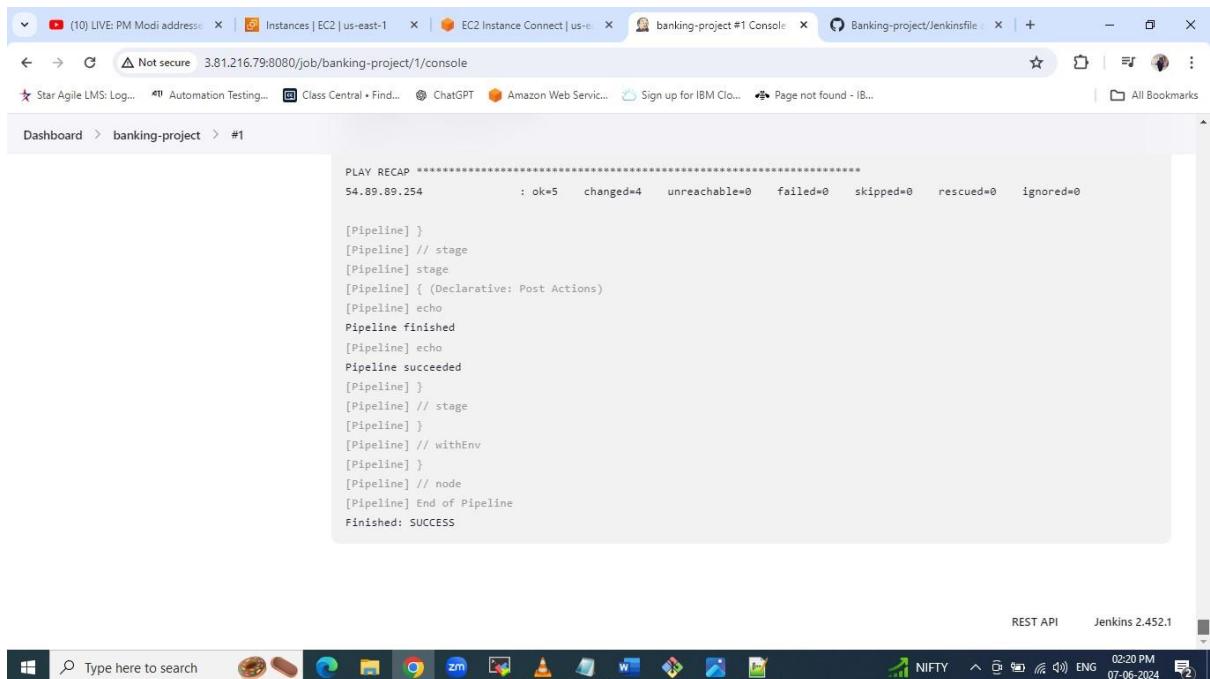
TASK [Start Docker Service] *****
changed: [54.89.89.254]

TASK [Deploy Docker Container] *****
changed: [54.89.89.254]

PLAY RECAP *****
54.89.89.254      : ok=5    changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[Pipeline] }
[Pipeline] // stage
```

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

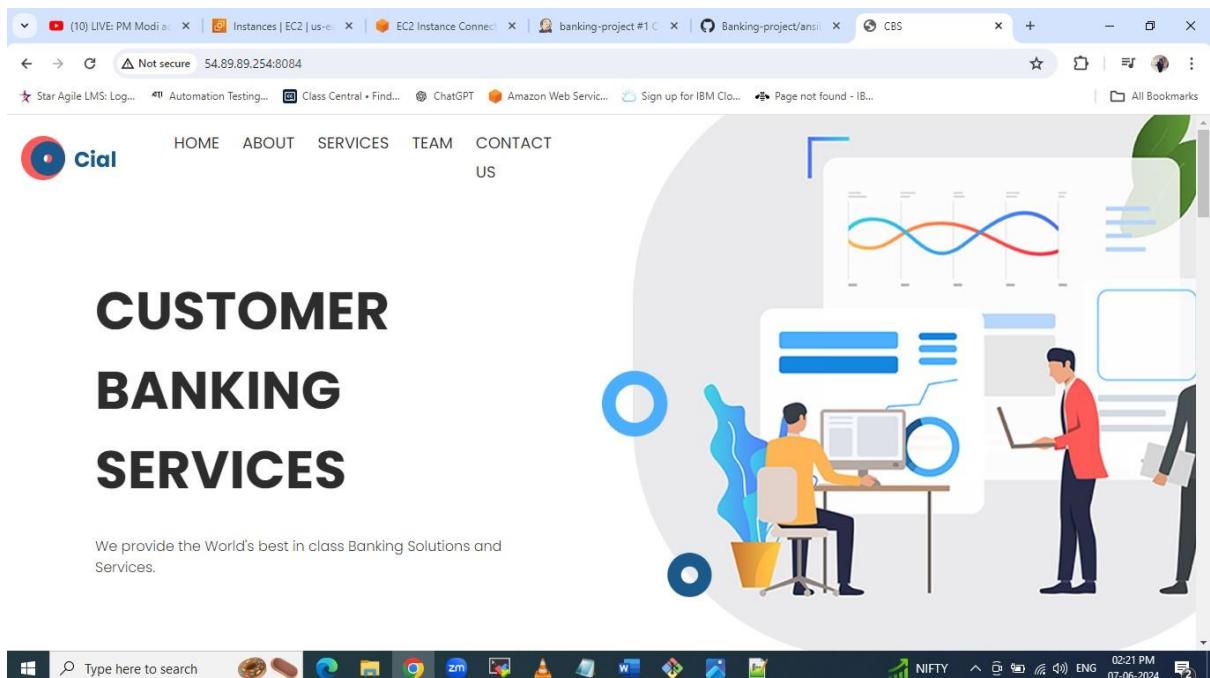


The screenshot shows a Windows desktop environment. At the top is the taskbar with various icons. Below it is a browser window titled "banking-project #1 Console". The main content of the browser shows the output of an Ansible playbook execution, specifically focusing on the PLAY RECAP and pipeline stages:

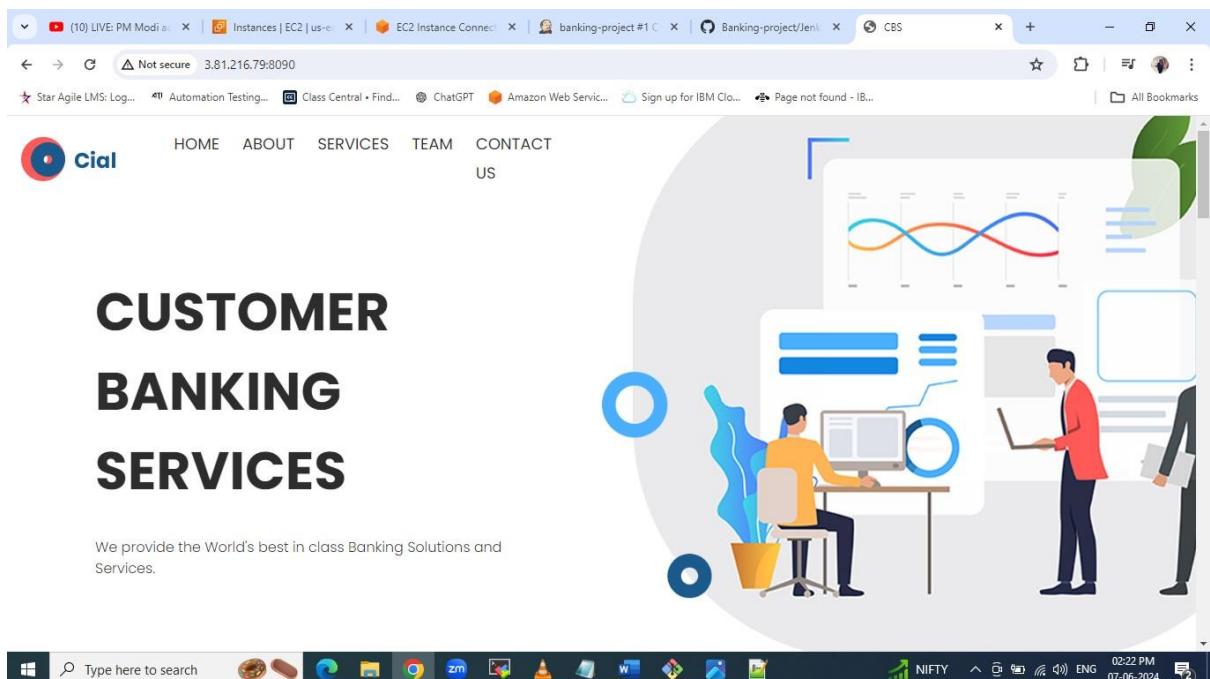
```
PLAY RECAP *****
54.89.89.254      : ok=5    changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] echo
Pipeline finished
[Pipeline] echo
Pipeline succeeded
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

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.

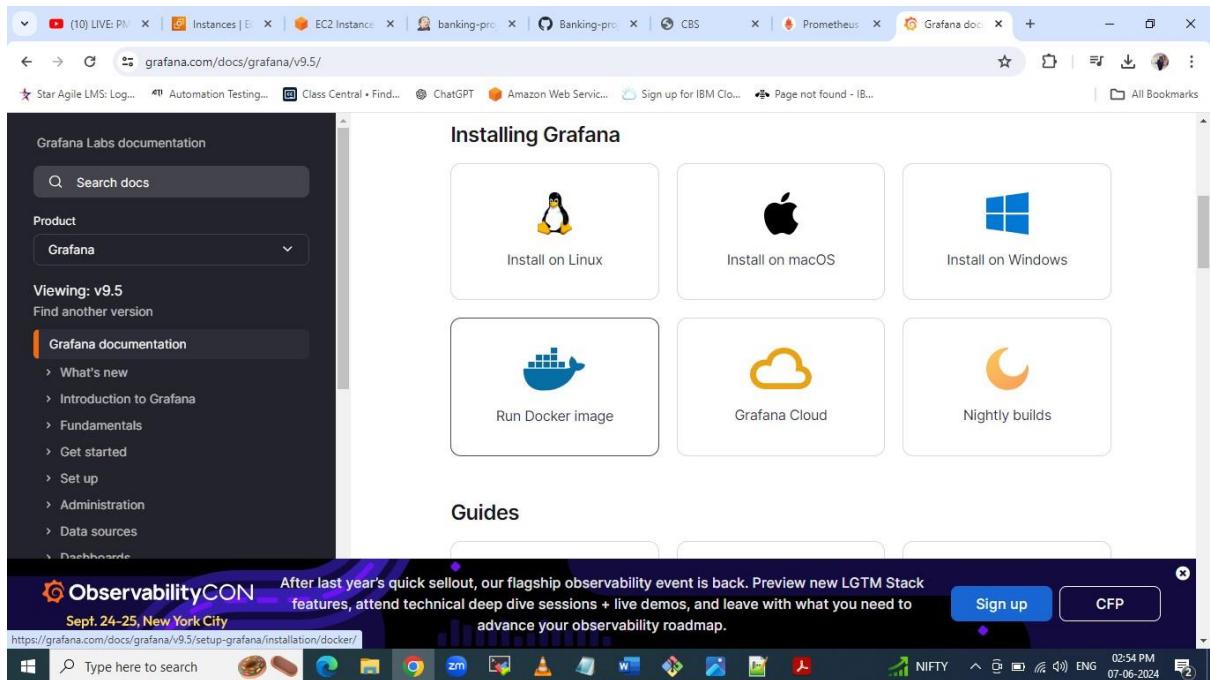
We have initialised the docker and deployed in Jenkinsfile and ansible-playbook.

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

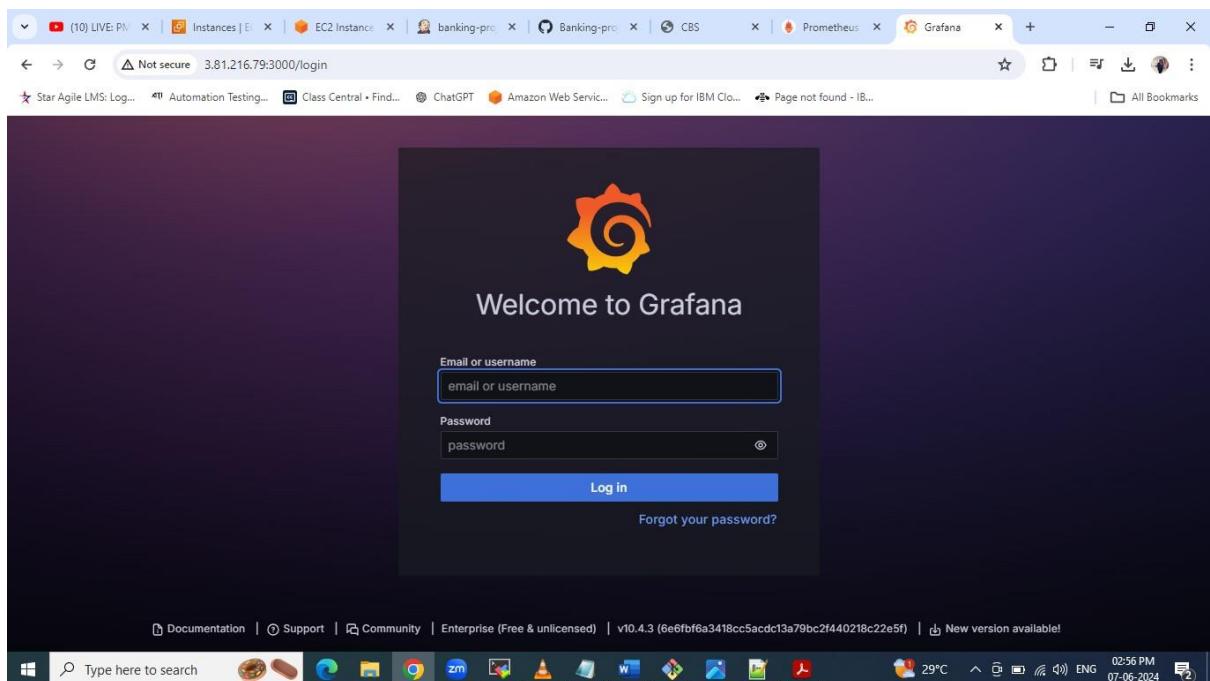
The screenshot shows a web browser window with multiple tabs open. The active tab is titled "Prometheus Tim" and displays the Prometheus Graph interface. The URL in the address bar is `3.81.216.79:9090/graph?g0.expr=&g0.tab=1&g0.display_mode=lines&g0.show_exemplars=0&g0.range_input=1h`. The browser's status bar indicates "Not secure". Below the address bar, there are several bookmarked links: "Star Agile LMS: Log...", "Automation Testing...", "Class Central • Find...", "ChatGPT", "Amazon Web Servic...", "Sign up for IBM Clo...", and "Page not found - IB...". The main content area of the browser shows the Prometheus interface. At the top of the interface, there are several checkboxes: "Use local time" (unchecked), "Enable query history" (unchecked), "Enable autocomplete" (checked), "Enable highlighting" (checked), and "Enable linter" (checked). Below these is a search bar with the placeholder "Expression (press Shift+Enter for newlines)". To the right of the search bar are buttons for "Table" (disabled), "Graph" (selected), "Alerts", "Status", and "Help". Further down, there is a button labeled "Execute". A message "No data queried yet" is displayed in the main panel. At the bottom of the interface, there are buttons for "Add Panel" and "Remove Panel".



Prometheus is installed and running on 9010 port.

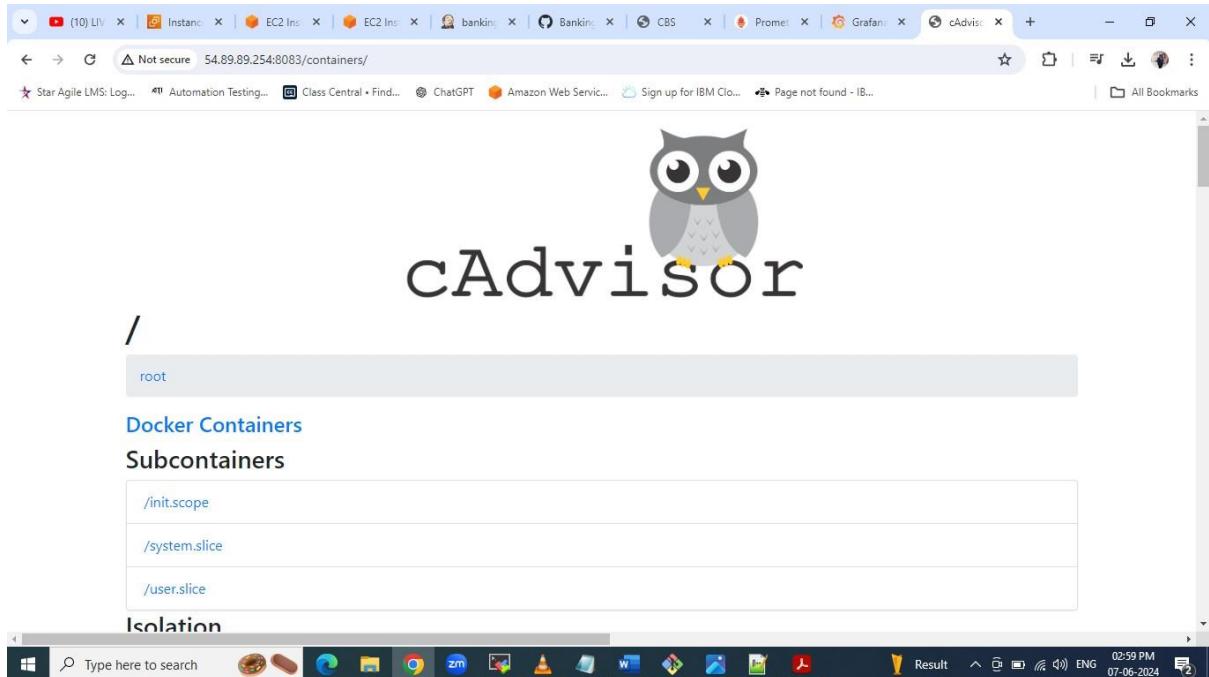


Grafana is installed by running docker image.



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.

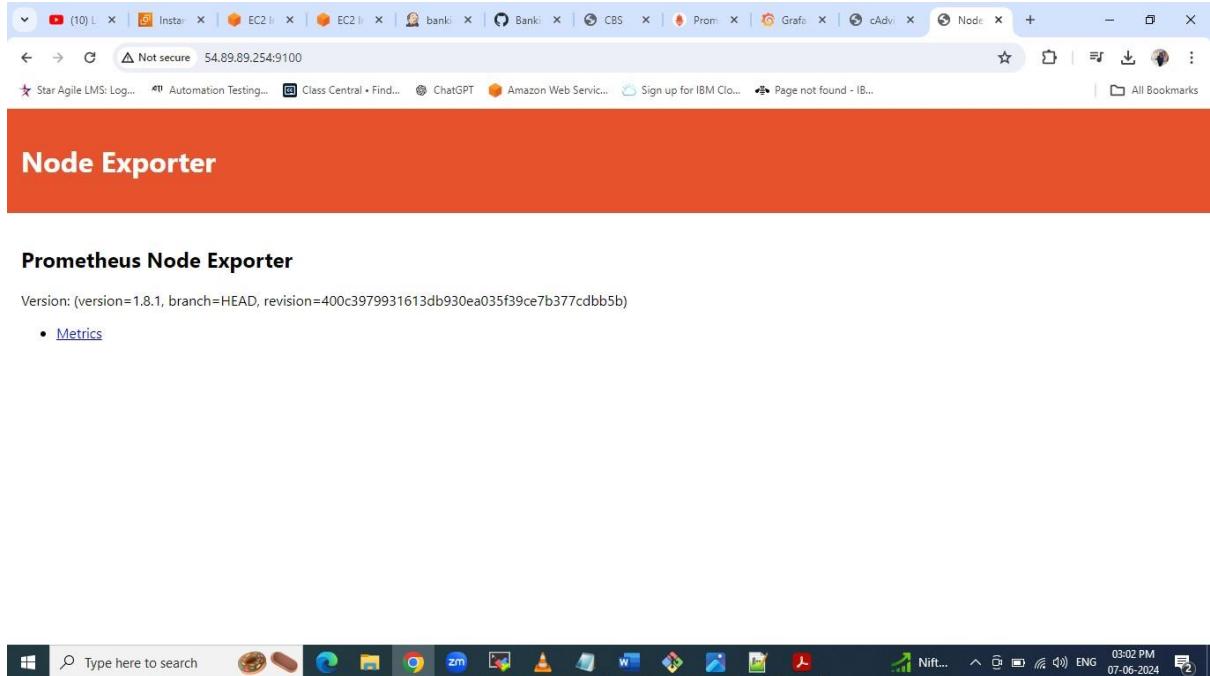


cAdvisor is installed on another machine and running on 8083 port.

```
root@ip-172-31-37-35:/home/ubuntu# sudo docker pull quay.io/prometheus/node-exporter
Using default tag: latest
latest: Pulling from prometheus/node-exporter
9fa9226be034: Pull complete
1617e25568b2: Pull complete
432171469b8e: Pull complete
Digest: sha256:faf12a57eff607176d5c363d8bb08dfbf636b36ac3cb5613a202f3c61a6631
Status: Downloaded newer image for quay.io/prometheus/node-exporter:latest
quay.io/prometheus/node-exporter:latest
root@ip-172-31-37-35:/home/ubuntu# sudo docker run -d -p 9100:9100 --name=node-exporter quay.io/prometheus/node-exporter
c618a76221734591bcbb0c03a1d0cd407b8bc88913bcf49b5ac45fd123476093
root@ip-172-31-37-35:/home/ubuntu#
```

i-01e874fd90bc71b9d (Ansible-host)
PublicIPs: 54.89.89.254 PrivateIPs: 172.31.37.35

NodeExporter is installed running docker image.



NodeExporter is installed and running on 9100 port.

```

{
  "metrics-addr" : "0.0.0.0:9323",
  "experimental" : true
}

i-0ce1e47618bf7cb69 (Master)
PublicIPs: 52.90.225.167 PrivateIPs: 172.31.38.32

```

CloudShell Feedback

Type here to search

Nifty smiccap +1.36% 12:49 PM 10-06-2024

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.

```

# HELP builder_builds_failed_total Number of failed image builds
# TYPE builder_builds_failed_total counter
builder_builds_failed_total{reason="build_canceled"} 0
builder_builds_failed_total{reason="build_target_not_reachable_error"} 0
builder_builds_failed_total{reason="command_not_supported_error"} 0
builder_builds_failed_total{reason="dockerfile_empty_error"} 0
builder_builds_failed_total{reason="dockerfile_syntax_error"} 0
builder_builds_failed_total{reason="error_processing_commands_error"} 0
builder_builds_failed_total{reason="invalid_onbuild_arguments_error"} 0
builder_builds_failed_total{reason="unknown_instruction_error"} 0
# HELP builder_builds_triggered_total Number of triggered image builds
# TYPE builder_builds_triggered_total counter
builder_builds_triggered_total 0
# HELP engine_daemon_container_actions_seconds The number of seconds it takes to process each container action
# TYPE engine_daemon_container_actions_seconds histogram
engine_daemon_container_actions_seconds_bucket{action="changes",le="0.005"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="0.01"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="0.025"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="0.05"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="0.1"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="0.25"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="0.5"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="1"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="2.5"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="10"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="+inf"} 1
engine_daemon_container_actions_seconds_sum{actions="changes"} 0
engine_daemon_container_actions_seconds_count{actions="changes"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="0.005"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="0.01"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="0.025"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="0.05"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="0.1"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="0.25"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="0.5"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="1"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="2.5"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="5"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="10"} 1

```

Type here to search

Nifty smiccap +1.36% 12:50 PM 10-06-2024

We can see the docker metrics via 9323 port.

```
ubuntu@ip-172-31-38-32:~$ ls
ansible.sh jenkins.sh prometheus-2.53.0-rc.0.linux-amd64
ubuntu@ip-172-31-38-32:~$ cd prometheus-2.53.0-rc.0.linux-amd64/
ubuntu@ip-172-31-38-32:~/prometheus-2.53.0-rc.0.linux-amd64$ ls
LICENSE NOTICE  consoles  data  prometheus  prometheus.yml  promtool
ubuntu@ip-172-31-38-32:~/prometheus-2.53.0-rc.0.linux-amd64$ vi prometheus.yml
ubuntu@ip-172-31-38-32:~/prometheus-2.53.0-rc.0.linux-amd64$ sudo vi prometheus.yml
ubuntu@ip-172-31-38-32:~/prometheus-2.53.0-rc.0.linux-amd64$
```

The screenshot shows a terminal window within an AWS CloudShell interface. The user has run a Jenkins script and listed the contents of the Prometheus distribution directory. They then opened the Prometheus configuration file for editing.

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

```
# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: "prometheus"
    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

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

  - job_name: "NodeExporter"
    static_configs:
      - targets: ["3.81.17.89:9100"]

  - job_name: "cAdvisor"
    static_configs:
      - targets: ["3.81.17.89:8080"]
```

The screenshot shows the Prometheus configuration file being edited. The user has added three new scrape configurations: 'NodeExporter' and 'cAdvisor' which point to the local host at port 9100, and 'prometheus' which points to the local host at port 9090.

Endpoints are added in Prometheus.yml file.

The screenshot shows a web browser window with multiple tabs open. The active tab is titled "Prometheus". The page displays three tables, each representing a target endpoint:

- NodeExporter (1/1 up)**: Shows one instance at `http://3.81.17.89:9100/metrics` labeled as UP.
- cAdvisor (1/1 up)**: Shows one instance at `http://3.81.17.89:8088/metrics` labeled as UP.
- prometheus (1/1 up)**: Shows one instance at `http://localhost:9090/metrics` labeled as UP.

Each table includes columns for Endpoint, State, Labels, Last Scrape, Scrape Duration, and Error. The labels column shows specific metrics like `instance="3.81.17.89:9100"`, `job="NodeExporter"`, etc. The last scrape time and duration are also listed.

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

The screenshot shows a web browser window with multiple tabs open. The active tab is titled "Grafana". The page has a dark theme and features a "Welcome to Grafana" message. It includes several informational panels:

- Basic**: A panel with instructions for setting up Grafana.
- TUTORIAL**: A panel titled "DATA SOURCE AND DASHBOARDS" with a sub-section "Grafana fundamentals". It describes how to set up Grafana for the first time.
- DATA SOURCES**: A panel with the heading "Add your first data source" and a "Learn how in the docs" link.
- DASHBOARDS**: A panel with the heading "Create your first dashboard" and a "Learn how in the docs" link.

At the bottom, there are sections for "Dashboards" and "Latest from the blog". The address bar shows the URL `52.90.225.167:3000/?utm_source=grafana_gettingstarted`.

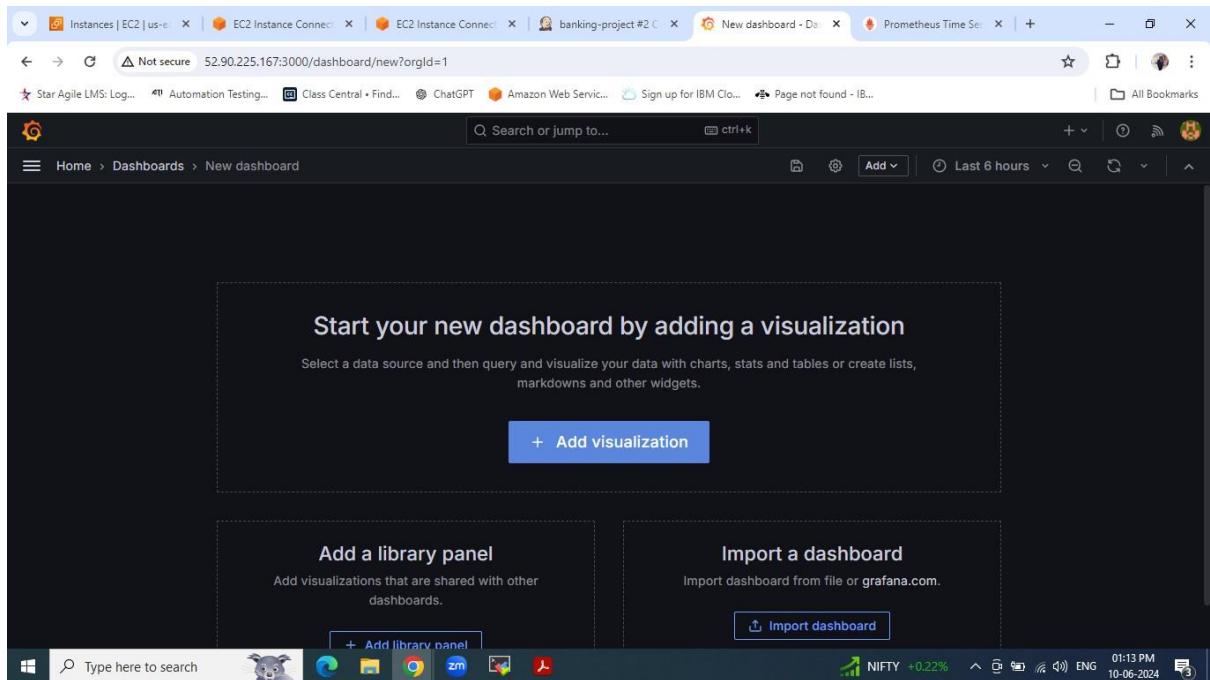
Now to go Grafana and add the data source to visualise the metrics.

The screenshot shows the Grafana interface with the URL `http://52.90.225.167:9090/` entered in the 'Prometheus server URL' field. The 'Authentication' section is set to 'No Authentication'. The status bar at the bottom indicates a successful connection with a green checkmark and the message 'Successfully queried the Prometheus API.'

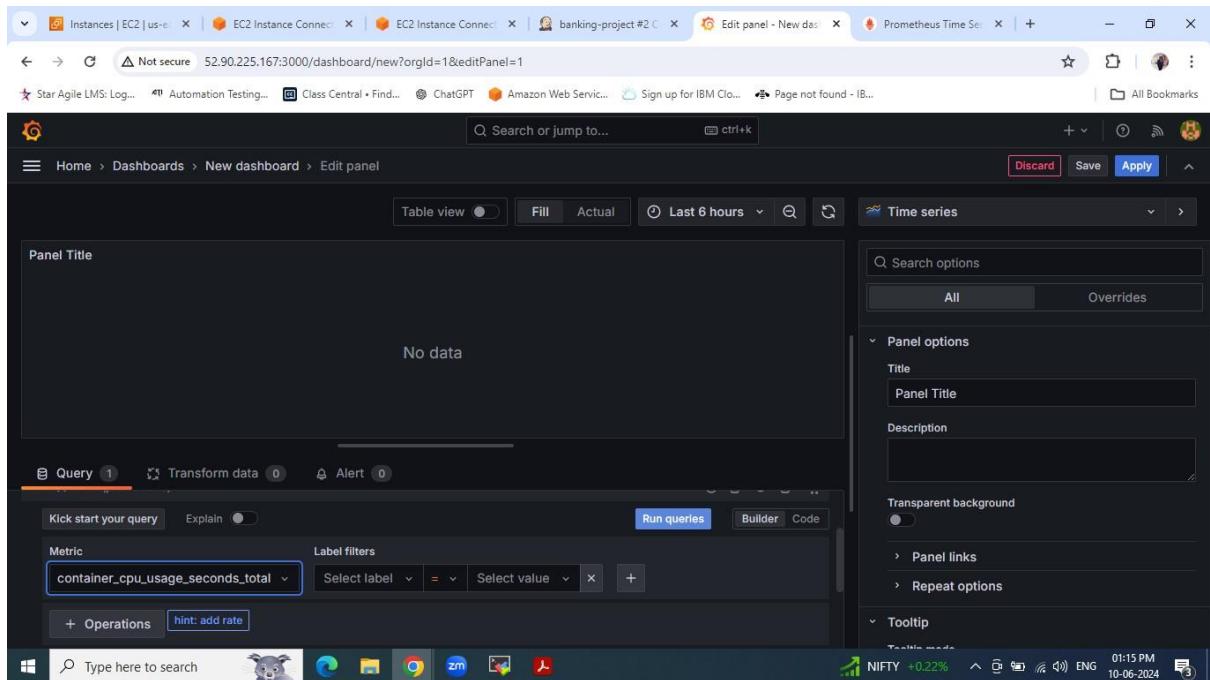
To connect Grafana with Prometheus add Prometheus URL in Grafana.

The screenshot shows the Grafana interface with the URL `http://52.90.225.167:9090/` entered in the 'Prometheus server URL' field. The 'Authentication' section is set to 'No Authentication'. A success message at the bottom states: 'Successfully queried the Prometheus API. Next, you can start to visualize data by [building a dashboard](#), or by querying data in the [Explore view](#)'.

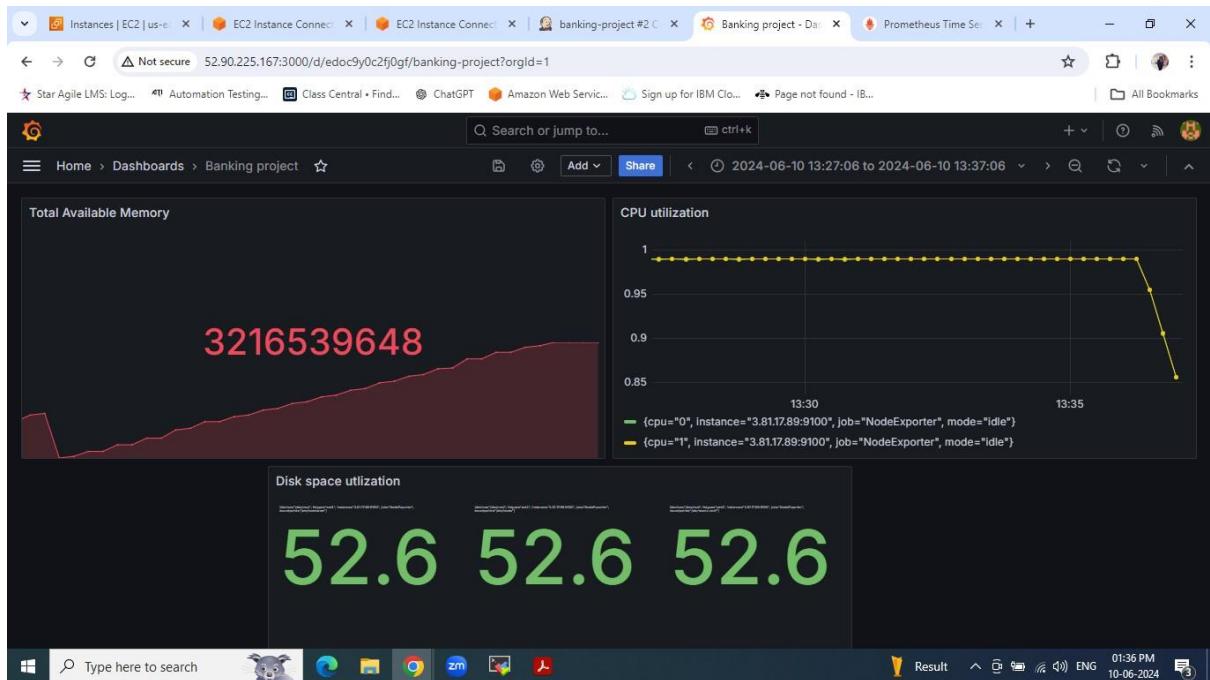
Tested the connection and it successful. We can build the dashboard now.



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.