

**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

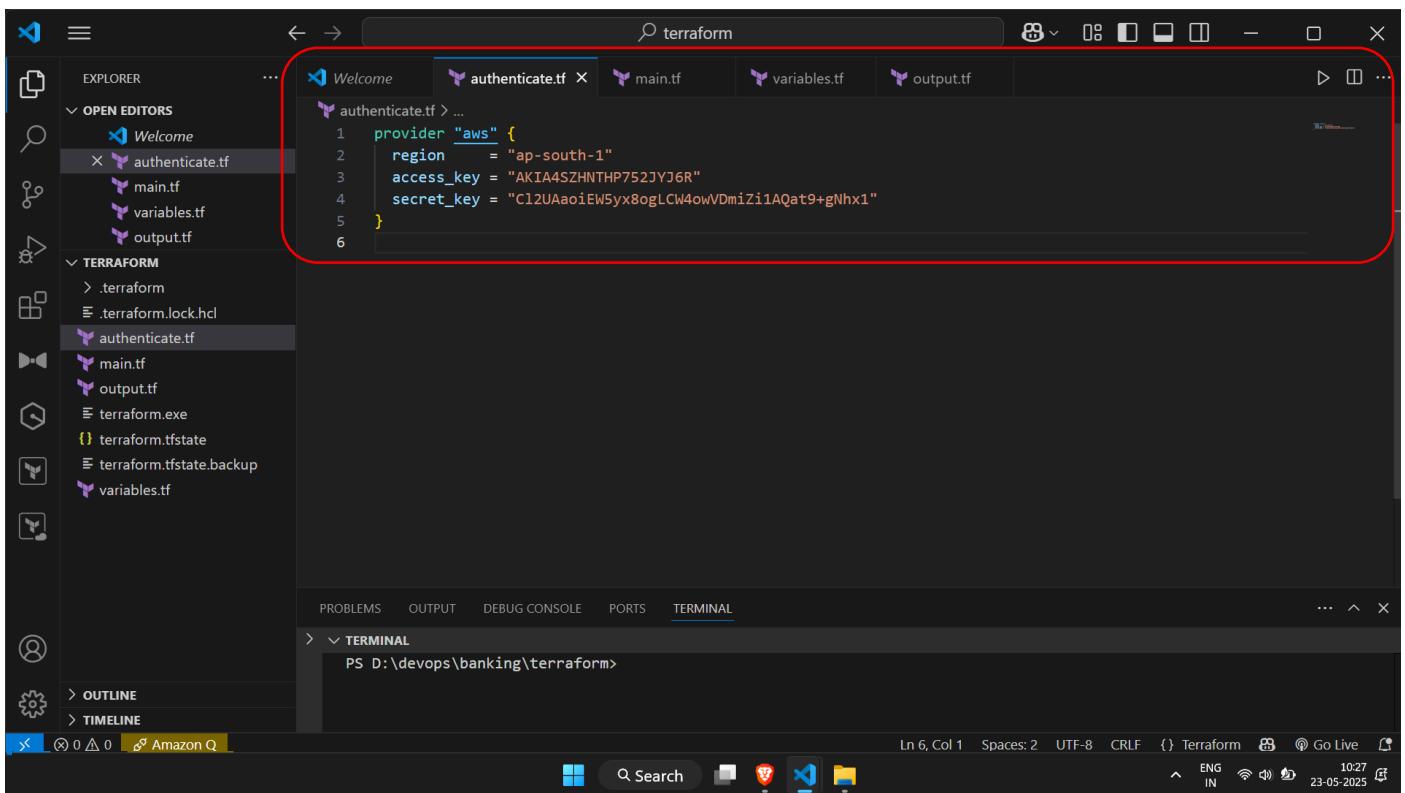
## Steps for project :

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

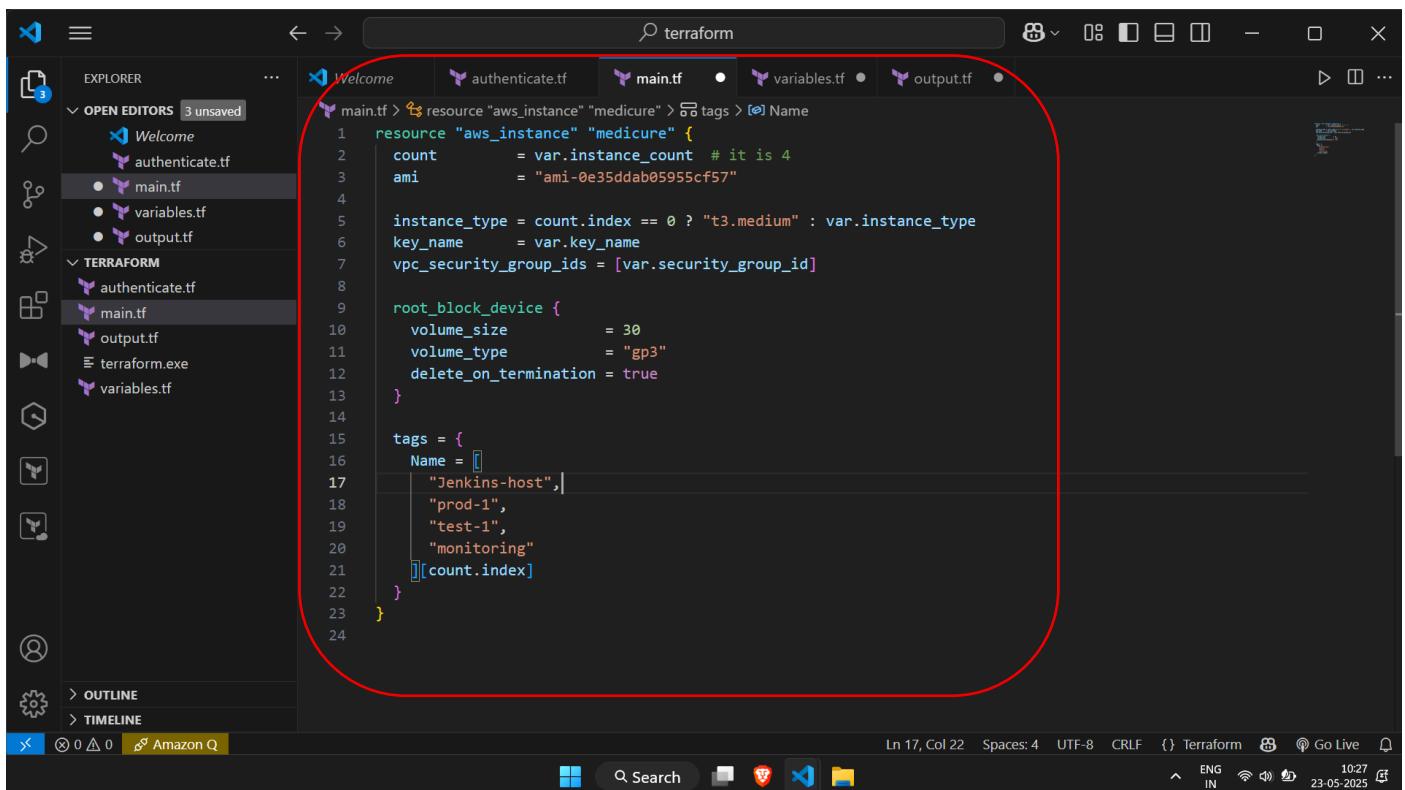
Authenticate..tf



```
provider "aws" {
  region      = "ap-south-1"
  access_key = "AKIA4SZHNTHP752JYJ6R"
  secret_key = "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"
}
```

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



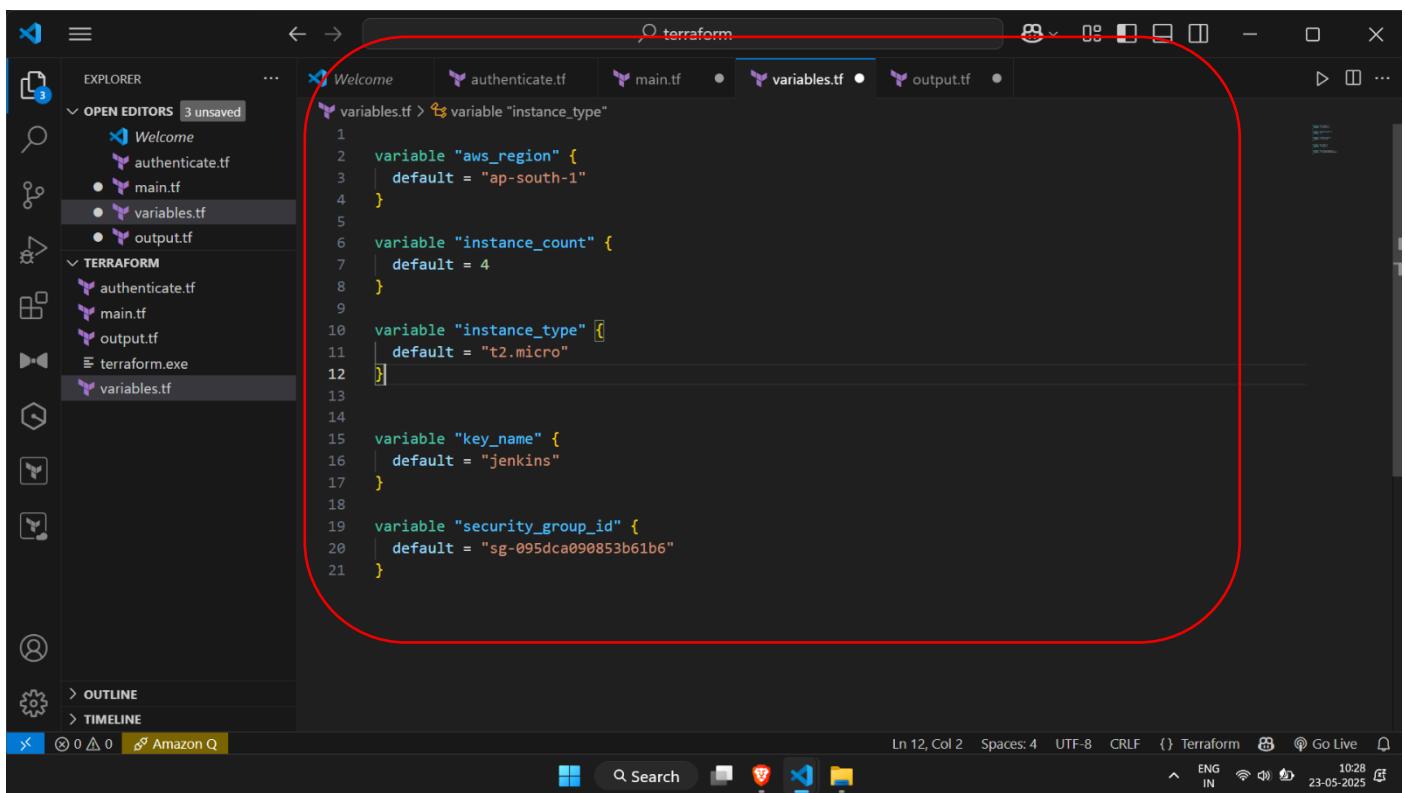
VS Code interface showing the `main.tf` file. The code defines an AWS instance named "medicure" with specific configuration and tags.

```
resource "aws_instance" "medicure" {
    count      = var.instance_count # it is 4
    ami        = "ami-0e35ddab05955cf57"
    instance_type = count.index == 0 ? "t3.medium" : var.instance_type
    key_name   = var.key_name
    vpc_security_group_ids = [var.security_group_id]

    root_block_device {
        volume_size      = 30
        volume_type      = "gp3"
        delete_on_termination = true
    }

    tags = [
        Name = [
            "Jenkins-host",
            "prod-1",
            "test-1",
            "monitoring"
        ][count.index]
    ]
}
```

## Variables.tf



VS Code interface showing the `variables.tf` file. It defines several variables with their default values.

```
variable "aws_region" {
    default = "ap-south-1"
}

variable "instance_count" {
    default = 4
}

variable "instance_type" [
    default = "t2.micro"
]

variable "key_name" {
    default = "jenkins"
}

variable "security_group_id" {
    default = "sg-095dca090853b61b6"
}
```

## Output.tf

The screenshot shows the VS Code interface with the 'output.tf' file open in the main editor. A red circle highlights the code block. The code defines two outputs:

```
output "instance_name_to_public_ip" {
  description = "Map of instance names to their public IP addresses"
  value = {
    for instance in aws_instance.medicure :
      | instance.tags["Name"] => instance.public_ip
  }
}
output "instance_ids" [
  description = "IDs of all Medicure instances"
  value       = [for instance in aws_instance.medicure : instance.id]
]
```

The left sidebar shows the project structure with files: Welcome, authenticate.tf, main.tf, variables.tf, and output.tf.

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

The screenshot shows the VS Code interface with the terminal tab open, displaying the output of the 'terraform init' command. A red circle highlights the terminal window. The output shows the initialization process:

```
D:\devops\banking>terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.98.0...
- Installed hashicorp/aws v5.98.0 (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!
```

The terminal also includes instructions and a note at the bottom:

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,

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 .

The screenshot shows the VS Code interface with the following details:

- EXPLORER:** Shows files: Welcome, authenticate.tf, main.tf (selected), variables.tf, output.tf.
- TERRAFORM:** Shows files: .terraform, .terraform.lock.hcl, authenticate.tf, main.tf, output.tf, terraform.exe, terraform.tfstate, variables.tf.
- TERMINAL:** Shows the output of a Terraform command:

```
Apply complete! Resources: 4 added, 0 changed, 0 destroyed.

Outputs:

instance_ids = [
  "i-0fd00900c0b474dc7",
  "i-0190e328b68cb7a80",
  "i-0c37391b3cb34c94e",
  "i-0773c6911cee509cf",
]
instance_name_to_public_ip = {
  "Jenkins-host" = "52.66.242.193"
  "monitoring" = "3.110.168.33"
  "prod-1" = "13.203.210.228"
  "test-1" = "3.109.56.235"
}

D:\devops\banking\terraform>
```
- STATUS BAR:** Shows Ln 5, Col 45, Spaces: 4, UTF-8, CRLF, {} Terraform, Go Live, ENG IN, 23-05-2025, 10:40.

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

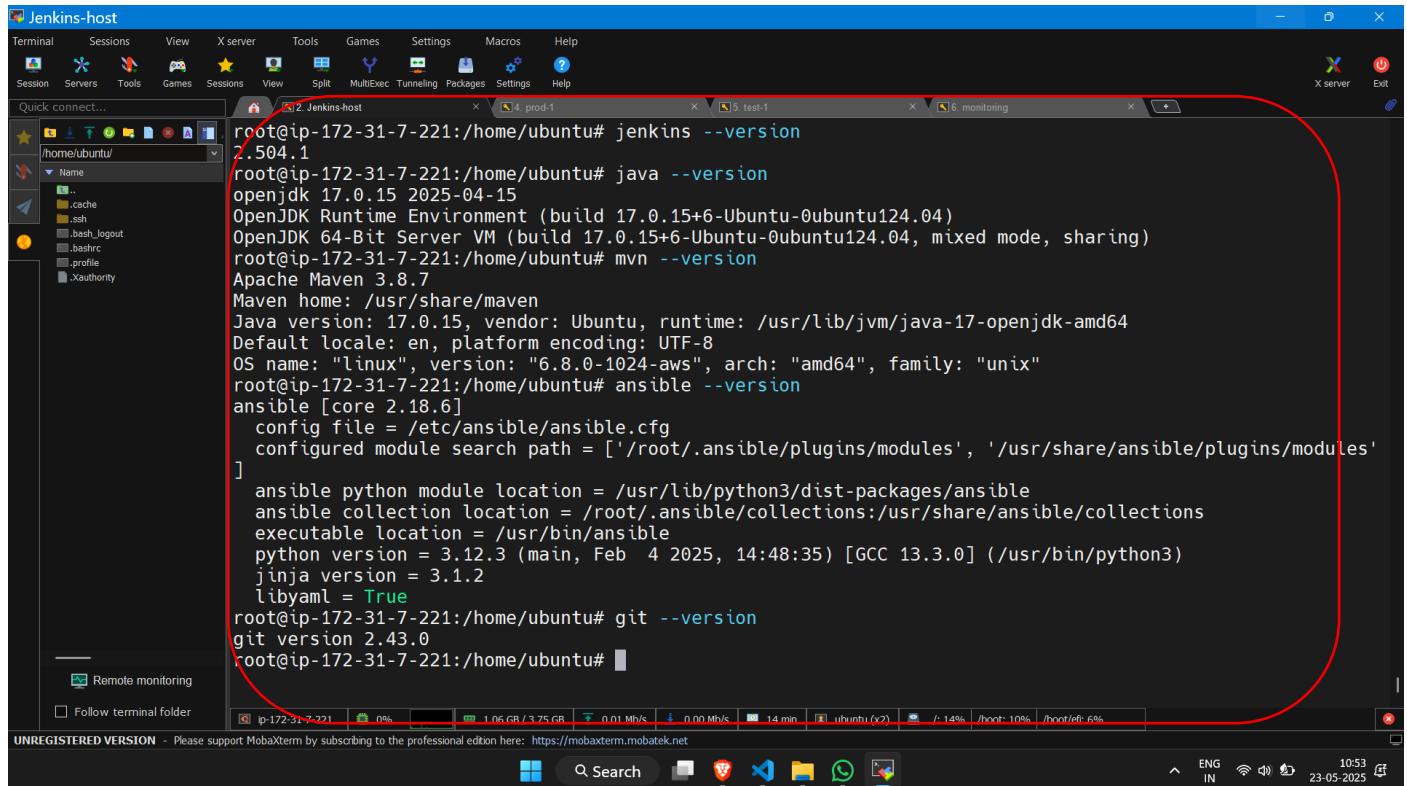
The screenshot shows the AWS EC2 Instances page with the following details:

- Left Sidebar:** EC2 > Instances, showing sections for Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, and Elastic Block Store.
- Instances Table:** Shows 4 instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
prod-1	i-0190e328b68cb7a80	Running	t2.micro	Initializing	View alarms +	ap-south-1b	ec2-13-2
Jenkins-host	i-0fd00900c0b474dc7	Running	t3.medium	Initializing	View alarms +	ap-south-1b	ec2-52-6
test-1	i-0c37391b3cb34c94e	Running	t2.micro	Initializing	View alarms +	ap-south-1b	ec2-3-10
monitoring	i-0773c6911cee509cf	Running	t2.micro	Initializing	View alarms +	ap-south-1b	ec2-3-11
- Bottom Bar:** Shows CloudShell, Feedback, © 2025, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, Cookie preferences, ENG IN, 23-05-2025, 10:40.

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

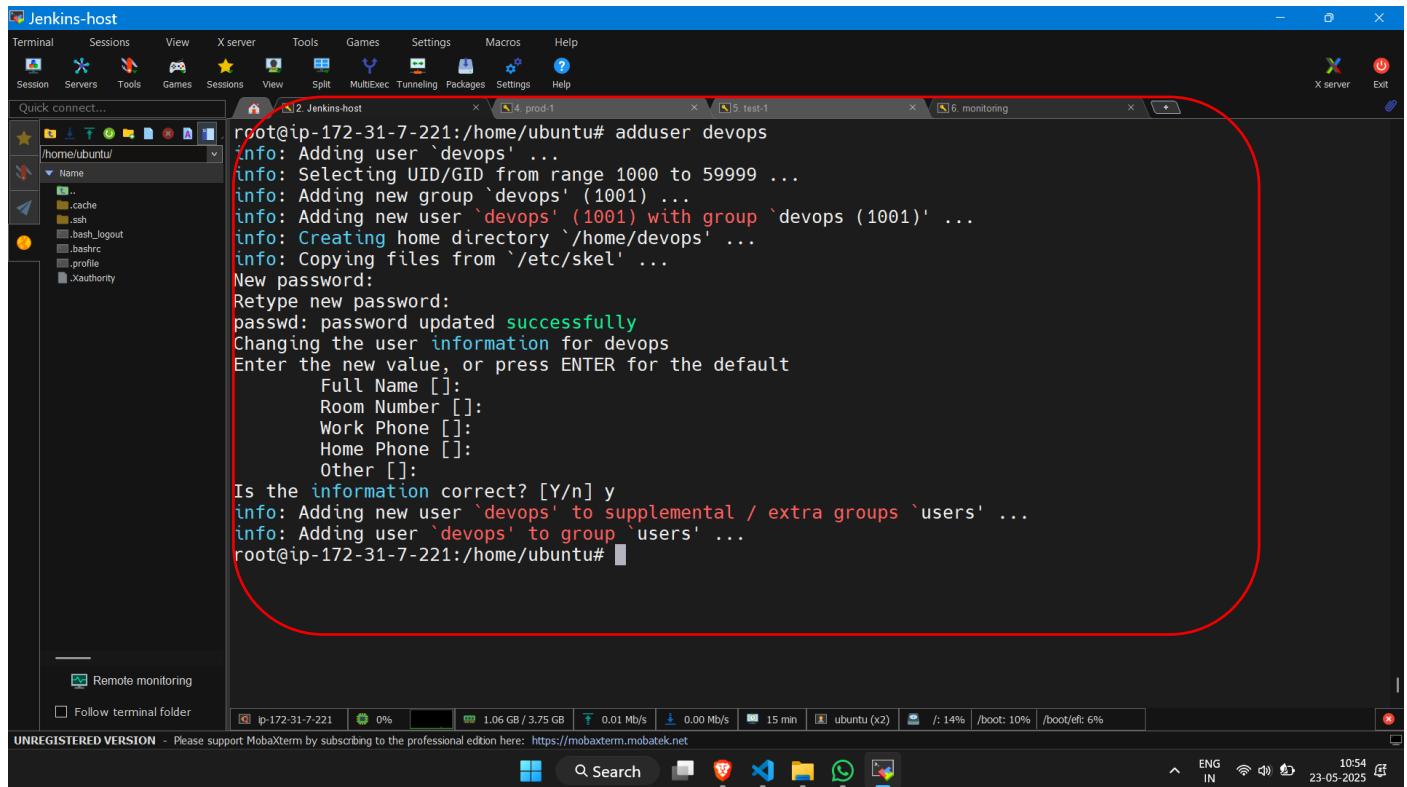
And install **Jenkins**, **Java**, **Maven**, **ansible**, **git** and also **Docker** this instances will be the Jenkins host and also the ansible controller node .



```
root@ip-172-31-7-221:/home/ubuntu# jenkins --version
2.504.1
root@ip-172-31-7-221:/home/ubuntu# java --version
openjdk 17.0.15 2025-04-15
OpenJDK Runtime Environment (build 17.0.15+6-Ubuntu-0ubuntu124.04)
OpenJDK 64-Bit Server VM (build 17.0.15+6-Ubuntu-0ubuntu124.04, mixed mode, sharing)
root@ip-172-31-7-221:/home/ubuntu# mvn --version
Apache Maven 3.8.7
Maven home: /usr/share/maven
Java version: 17.0.15, vendor: Ubuntu, runtime: /usr/lib/jvm/java-17-openjdk-amd64
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "6.8.0-1024-aws", arch: "amd64", family: "unix"
root@ip-172-31-7-221:/home/ubuntu# ansible --version
ansible [core 2.18.6]
  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 = 3.12.3 (main, Feb  4 2025, 14:48:35) [GCC 13.3.0] (/usr/bin/python3)
  jinja version = 3.1.2
  libyaml = True
root@ip-172-31-7-221:/home/ubuntu# git --version
git version 2.43.0
root@ip-172-31-7-221:/home/ubuntu#
```

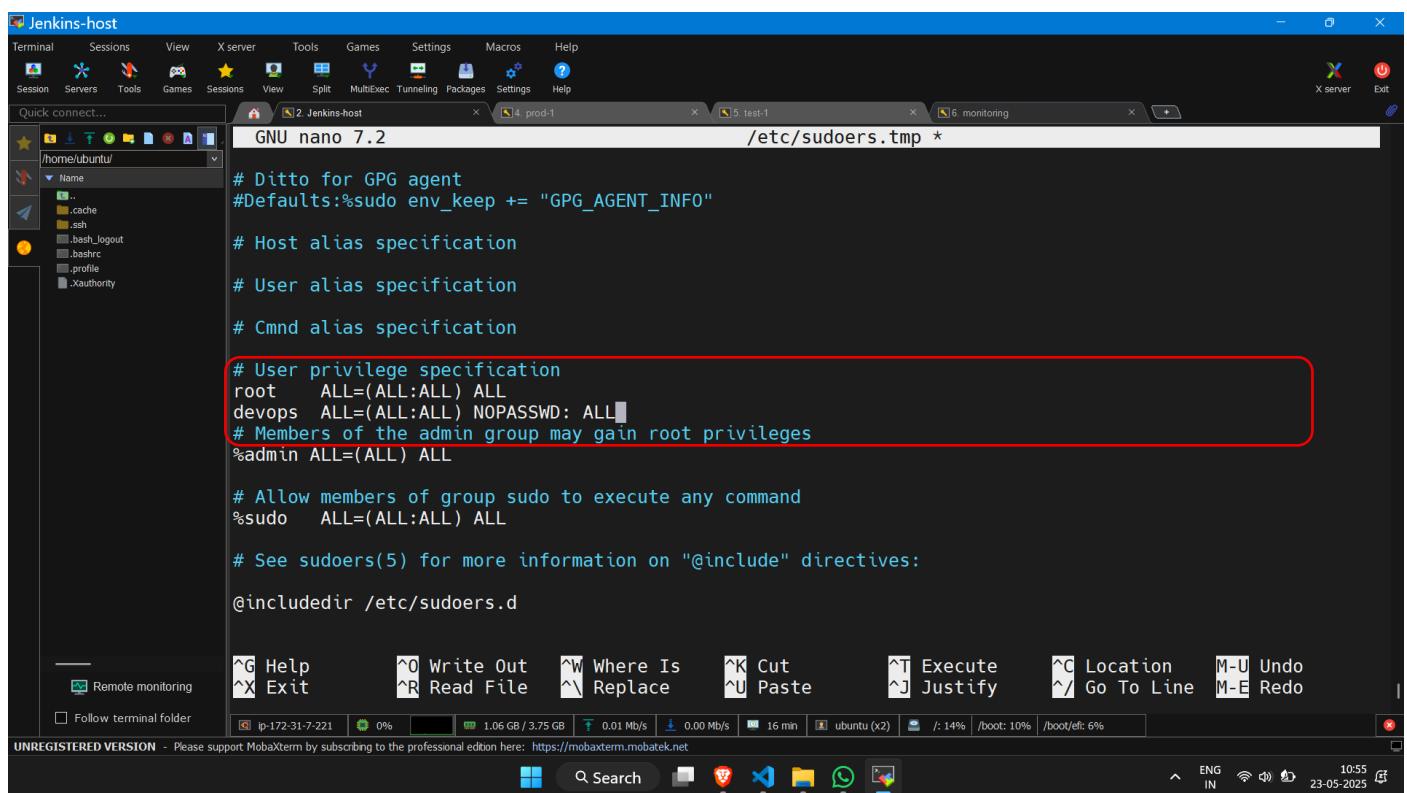
## Ansible Setup and configuration with other instances

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)



```
root@ip-172-31-7-221:/home/ubuntu# adduser devops
info: Adding user `devops' ...
info: Selecting UID/GID from range 1000 to 59999 ...
info: Adding new group `devops' (1001) ...
info: Adding new user `devops' (1001) with group `devops (1001)' ...
info: Creating home directory `/home/devops' ...
info: Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for devops
Enter the new value, or press ENTER for the default
  Full Name []:
  Room Number []:
  Work Phone []:
  Home Phone []:
  Other []:
Is the information correct? [Y/n] y
info: Adding new user `devops' to supplemental / extra groups `users' ...
info: Adding user `devops' to group `users' ...
root@ip-172-31-7-221:/home/ubuntu#
```

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



```
GNU nano 7.2 /etc/sudoers.tmp *
# Ditto for GPG agent
Defaults:env_keep += "GPG_AGENT_INFO"

# Host alias specification

# User alias specification

# Cmnd alias specification

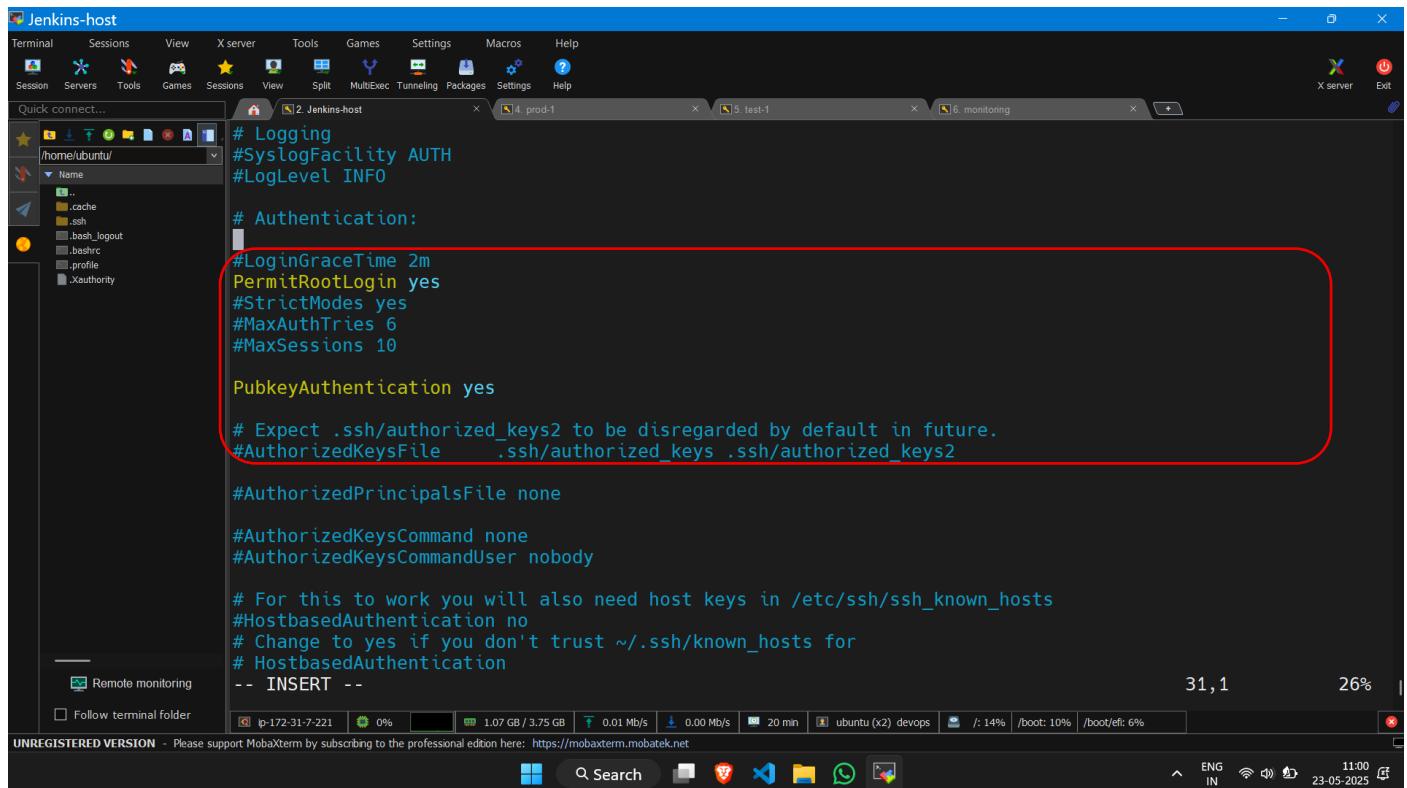
# User privilege specification
root    ALL=(ALL:ALL) ALL
devops  ALL=(ALL:ALL) NOPASSWD: ALL
# Members of the admin group may gain root privileges
%admin  ALL=(ALL) ALL

# 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
```

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 “ PermitRootLogin permission to yes “ and and remove

“# from PubkeyAuthentication “



```
# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:
#
#LoginGraceTime 2m
PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes

# Expect .ssh/authorized_keys2 to be disregarded by default in future.
#AuthorizedKeysFile      .ssh/authorized_keys .ssh/authorized_keys2

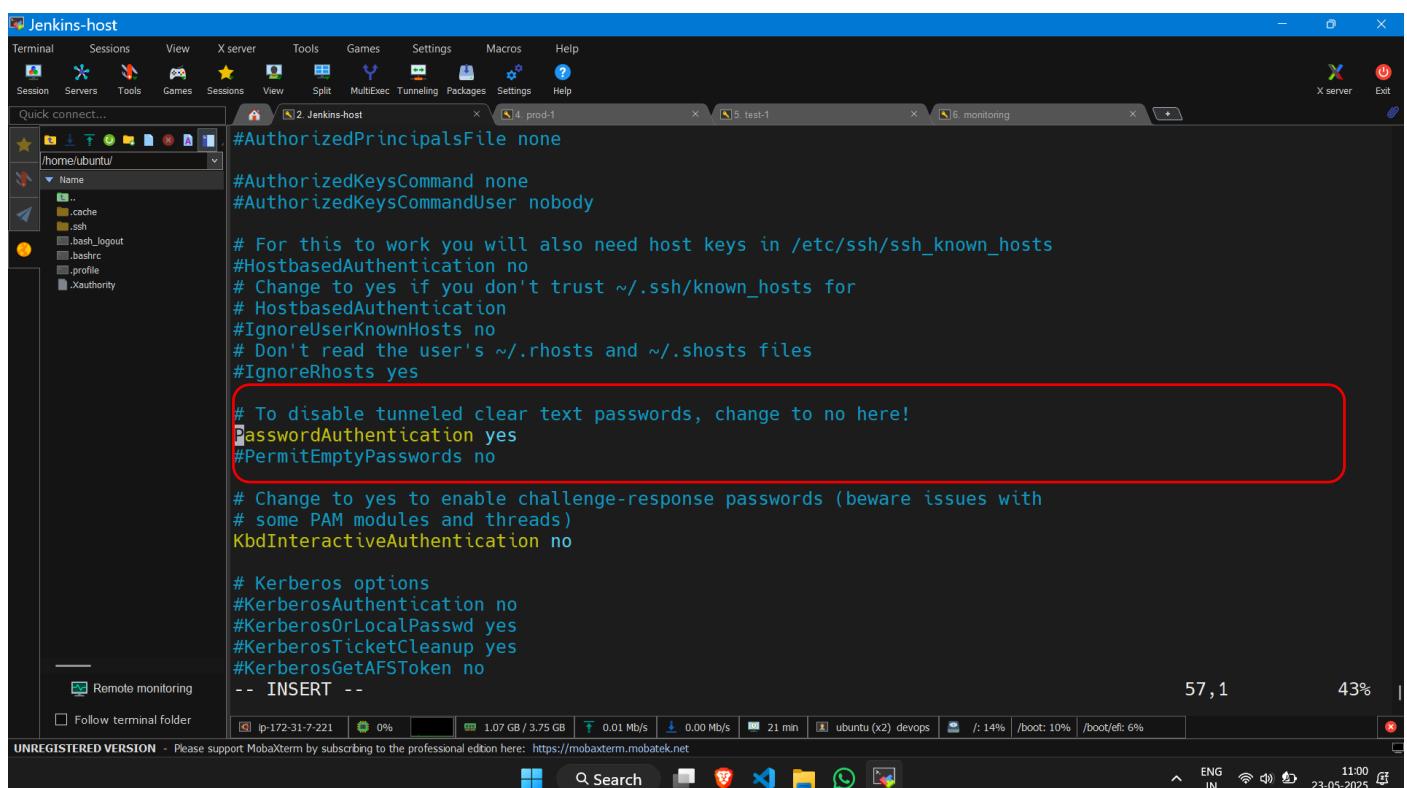
#AuthorizedPrincipalsFile none

#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody

# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
-- INSERT --
```

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

And the save and exit the file



```
#AuthorizedPrincipalsFile none

#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody

# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

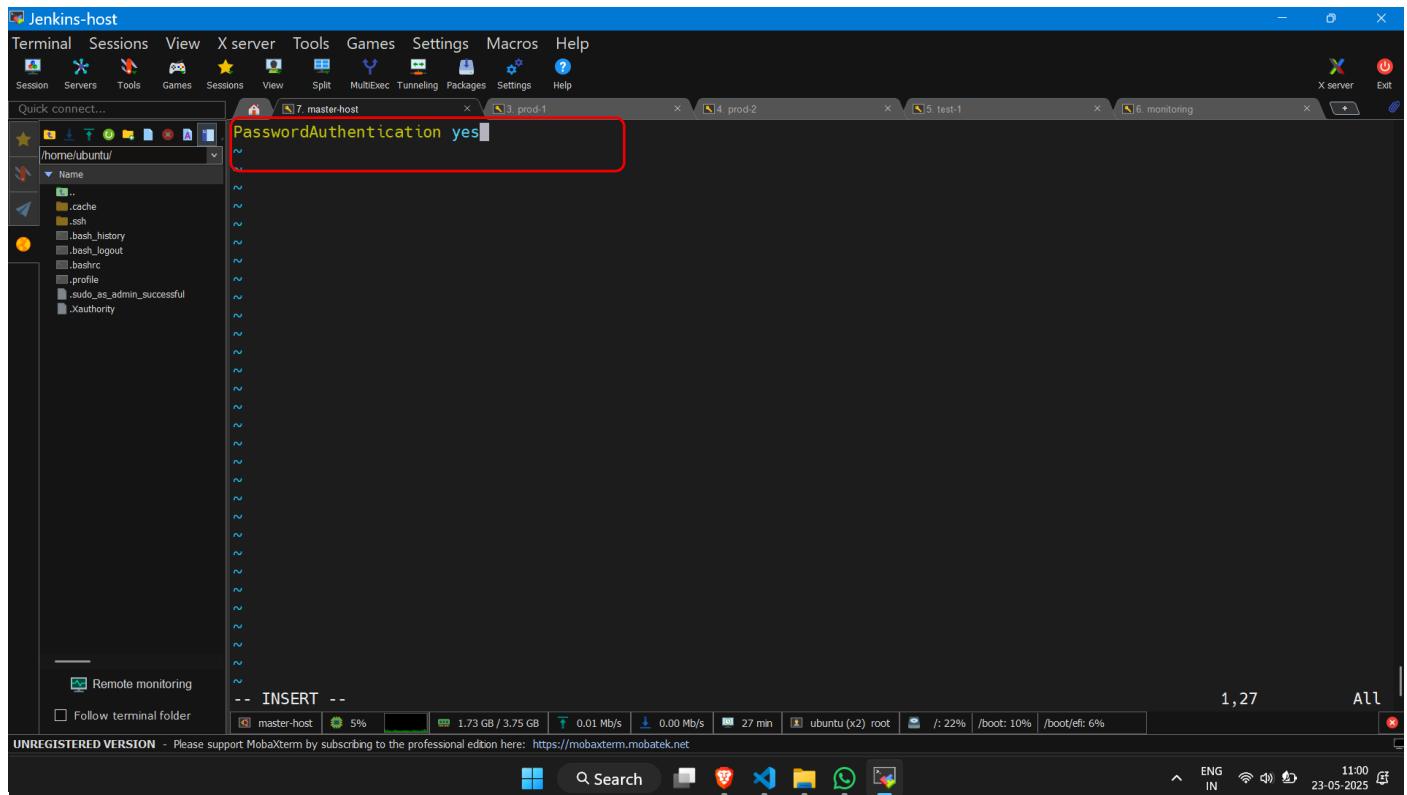
# To disable tunneled clear text passwords, change to no here!
#PasswordAuthentication yes
#PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
KbdInteractiveAuthentication no

# Kerberos options
#KerberosAuthentication no
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes
#KerberosGetAFSToken no
-- INSERT --
```

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”

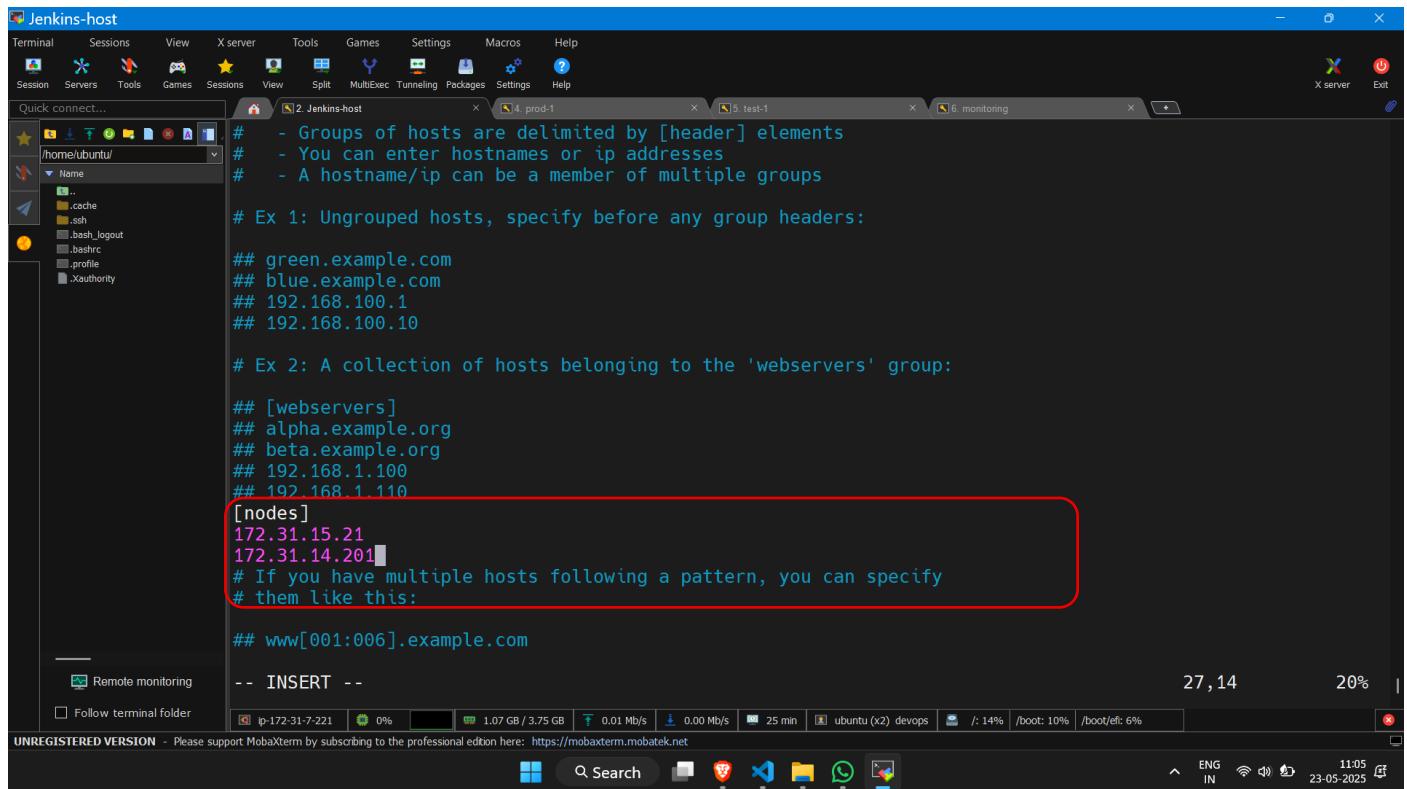
Do all the file changes in all the other nodes



```
>PasswordAuthentication yes
```

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 ”



```
# Groups of hosts are delimited by [header] elements
# - You can enter hostnames or ip addresses
#   - A hostname/ip can be a member of multiple groups

# Ex 1: Ungrouped hosts, specify before any group headers:

## green.example.com
## blue.example.com
## 192.168.100.1
## 192.168.100.10

# Ex 2: A collection of hosts belonging to the 'webservers' group:

## [webservers]
## alpha.example.org
## beta.example.org
## 192.168.1.100
## 192.168.1.110
[nodes]
172.31.15.21
172.31.14.201
# If you have multiple hosts following a pattern, you can specify
# them like this:

## www[001:006].example.com
```

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 screenshot shows a terminal session titled "Jenkins-host" in MobaXterm. The user is running the command "ssh-keygen". The output shows the key pair being generated, including the public key path (/home/devops/.ssh/id\_ed25519.pub) and the SHA256 fingerprint.

```
root@ip-172-31-7-221:/home/ubuntu# su - devops
devops@ip-172-31-7-221:~$ ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/devops/.ssh/id_ed25519):
Created directory '/home/devops/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/devops/.ssh/id_ed25519
Your public key has been saved in /home/devops/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:xEEL2Nhz6dE3AyrVx4ECd65i2UrsMt08XTbIib6goPlw devops@ip-172-31-7-221
The key's randomart image is:
+--[ED25519 256]--+
|          o+o+.
|          o = +x0.
|          o.ooo=.
|          *oo+ .
|          . S+ =
|          E ...* + .
|          .. o* + .
|          .o . o+= .
|          .oo .. +=o
+---[SHA256]-----+
devops@ip-172-31-7-221:~$
```

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\_instances> “

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

The screenshot shows a terminal session titled "Jenkins-host" in MobaXterm. The user is running "ssh-copy-id" to transfer their public key to another host. They are prompted for confirmation and asked to enter the target host's password.

```
devops@ip-172-31-7-221:~/.ssh$ ssh-copy-id devops@172.31.14.201
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/devops/.ssh/id_ed25519.pub"
The authenticity of host '172.31.14.201 (172.31.14.201)' can't be established.
ED25519 key fingerprint is SHA256:bdtqsdIU3J0KmuZTTBLzHiNHoVBkoo0d5wruZ+/N9Rg.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already
installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install t
he new keys
devops@172.31.14.201's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'devops@172.31.14.201'"
and check to make sure that only the key(s) you wanted were added.

devops@ip-172-31-7-221:~/.ssh$
```

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

```
devops@ip-172-31-7-221:~$ ansible all -a "touch file1"
[WARNING]: Platform linux on host 172.31.14.201 is using the discovered Python interpreter at
/usr/bin/python3.12, but future installation of another Python interpreter could change the meaning of
that path. See https://docs.ansible.com/ansible-
core/2.18/reference_appendices/interpreter_discovery.html for more information.
172.31.14.201 | CHANGED | rc=0 >>

[WARNING]: Platform linux on host 172.31.15.21 is using the discovered Python interpreter at
/usr/bin/python3.12, but future installation of another Python interpreter could change the meaning of
that path. See https://docs.ansible.com/ansible-
core/2.18/reference_appendices/interpreter_discovery.html for more information.
172.31.15.21 | CHANGED | rc=0 >>

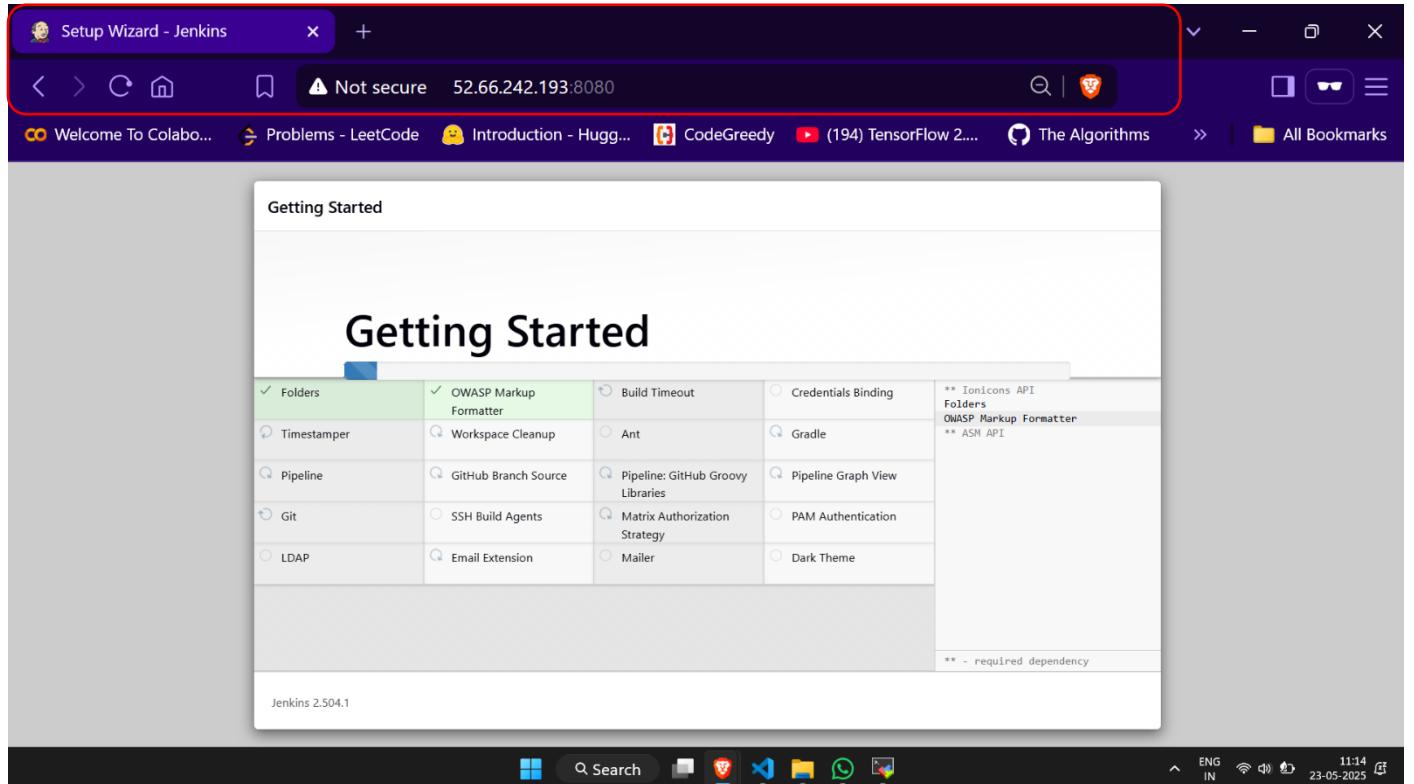
devops@ip-172-31-7-221:~$ ansible all -a "ls"
[WARNING]: Platform linux on host 172.31.14.201 is using the discovered Python interpreter at
/usr/bin/python3.12, but future installation of another Python interpreter could change the meaning of
that path. See https://docs.ansible.com/ansible-
core/2.18/reference_appendices/interpreter_discovery.html for more information.
172.31.14.201 | CHANGED | rc=0 >>
file1
[WARNING]: Platform linux on host 172.31.15.21 is using the discovered Python interpreter at
/usr/bin/python3.12, but future installation of another Python interpreter could change the meaning of
that path. See https://docs.ansible.com/ansible-
core/2.18/reference_appendices/interpreter_discovery.html for more information.
172.31.15.21 | CHANGED | rc=0 >>
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

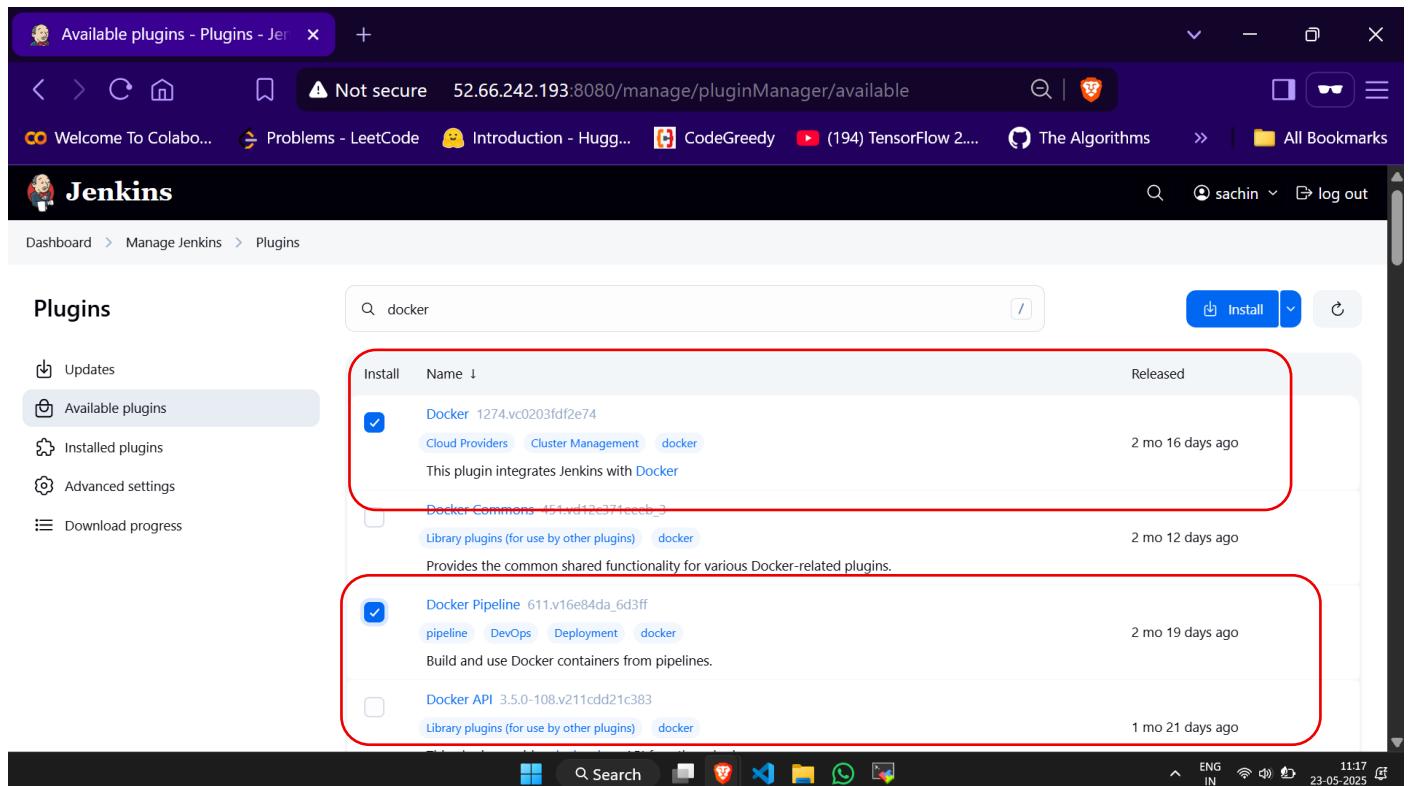
Now access the Jenkins application using the public ip of the host machine on the port 8080 ensuring that the security group is configured correctly which allows 8080 port in the inbound rules

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

Then set user required name and password



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



Also install ansible plugin

The screenshot shows the Jenkins plugin manager interface. In the top navigation bar, there are several links: Available plugins - Jenkins, Welcome To Colabo..., Problems - LeetCode, Introduction - Hugg..., CodeGreedy, (194) TensorFlow 2..., The Algorithms, and All Bookmarks. The user is logged in as sachin. Below the navigation, the Jenkins logo is displayed. The main content area is titled "Plugins" and has a search bar with the query "ansible". A red box highlights the "Available plugins" section, which lists the "Ansible" plugin by pipeline (version 524.v9fa\_a\_4c989224). The description for the Ansible plugin states: "Invoke Ansible Ad-Hoc commands and playbooks." Another plugin, "Ansible Tower" (version 0.17.0), is also listed with the description: "This plugin connects Jenkins with Ansible Tower". On the left sidebar, there are links for Updates, Available plugins (which is selected and highlighted in red), Installed plugins, Advanced settings, and Download progress. At the bottom right, it says REST API Jenkins 2.504.1.

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

And then restart the Jenkins service

The screenshot shows a terminal session in MobaXterm. The title bar indicates the session is connected to "Jenkins-host". The terminal window displays the following commands being run:

```
root@ip-172-31-7-221:/home/devops# sudo usermod -aG docker jenkins
root@ip-172-31-7-221:/home/devops# sudo systemctl restart jenkins
root@ip-172-31-7-221:/home/devops#
```

A red box highlights the command "root@ip-172-31-7-221:/home/devops# sudo usermod -aG docker jenkins". The terminal also shows a file tree on the left side under "/home/ubuntu/" and various system status indicators at the bottom.

Configure the ansible path in the jenkins

Tools - Jenkins

Not secure 52.66.242.193:8080/manage/configureTools/

Welcome To Colabo... Problems - LeetCode Introduction - Hugg... CodeGreedy (194) TensorFlow 2.... The Algorithms > All Bookmarks

Dashboard > Manage Jenkins > Tools

Add Ansible

**Ansible**

Name: ansible

Path to ansible executables directory: /usr/bin

Install automatically ?

Add Ansible

Save Apply

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

Jenkins-host

Terminal Sessions View X server Tools Games Settings Macros Help

Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help

Quick connect...

root@ip-172-31-7-221:/home/devops# cd .ssh

root@ip-172-31-7-221:/home/devops/.ssh# ls

id\_ed25519 id\_ed25519.pub known\_hosts known\_hosts.old

root@ip-172-31-7-221:/home/devops/.ssh# cat id\_ed25519

-----BEGIN OPENSSH PRIVATE KEY-----

b3B1bnNzaC1rZXktqjEAAAAABG5vbmlUA5AAEbm9uZQAAAAAAAAABAAAAMwAAAAtzc2gtZWQyNTUx0QAACDvoGN/Rchzetc8LMfxBxbmH0xd5IUTFDT0lrNdq7cwaQAAAKCGGIBlhhiAZQAAAAtzc2gtZWQyNTUx0QAACDvoGN/Rchzetc8LMfxBxbmH0xd5IUTFDT0lrNdq7cwaQAAAEcnax8BmeiOJnxh0WUJvn0v87WyJj/0JslfXwkdbgMUH+++gY39FyHN62ryUx/EHFuYc7F3khRMUNM6Ws12rtzBpAAAAFmRldm9wc0BpcC0xNzItMzEtNy0yMjEBAgMEBQYH

-----END OPENSSH PRIVATE KEY-----

root@ip-172-31-7-221:/home/devops/.ssh#

Remote monitoring

Follow terminal folder

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

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 .

The screenshot shows the Jenkins 'New credentials' configuration page. A red circle highlights the 'Kind' dropdown set to 'SSH Username with private key'. Other fields include 'Scope' (Global), 'ID' (ansible-ssh), 'Description' (empty), 'Username' (devops), and a checked 'Treat username as secret' checkbox. Under 'Private Key', the 'Enter directly' radio button is selected, and a text area contains the copied private key:

```
-----BEGIN OPENSSH PRIVATE KEY-----
b3B1bnNzaC1rZXktdjEAAAABG5vbmUAAAEBm9uZQAAAAAAAABAAAA%AAAAtzc2gtZN
QyNTUxQAAACDVoGN/Rchzetq8JHfxBxmH0xd51UTFDt0lNdq7cw@QAAKCGGIB1hhIA
```

A blue 'Create' button is at the bottom.

This screenshot shows the same Jenkins 'New credentials' configuration page, but the 'Enter directly' radio button under 'Private Key' is now unselected, and the text area is empty. Instead, a 'Key' input field is present, containing the same copied private key:

```
-----BEGIN OPENSSH PRIVATE KEY-----
b3B1bnNzaC1rZXktdjEAAAABG5vbmUAAAEBm9uZQAAAAAAAABAAAA%AAAAtzc2gtZN
QyNTUxQAAACDVoGN/Rchzetq8JHfxBxmH0xd51UTFDt0lNdq7cw@QAAKCGGIB1hhIA
```

The 'Create' button is at the bottom.

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

The screenshot shows the Jenkins 'New credentials' configuration page. A red box highlights the 'Username with password' section. Inside this section, the 'Scope' dropdown is set to 'Global (Jenkins, nodes, items, all child items, etc)'. The 'Username' field contains 'sachinbacha', and the 'ID' field contains 'dockercreds'. The 'Create' button is visible at the bottom.

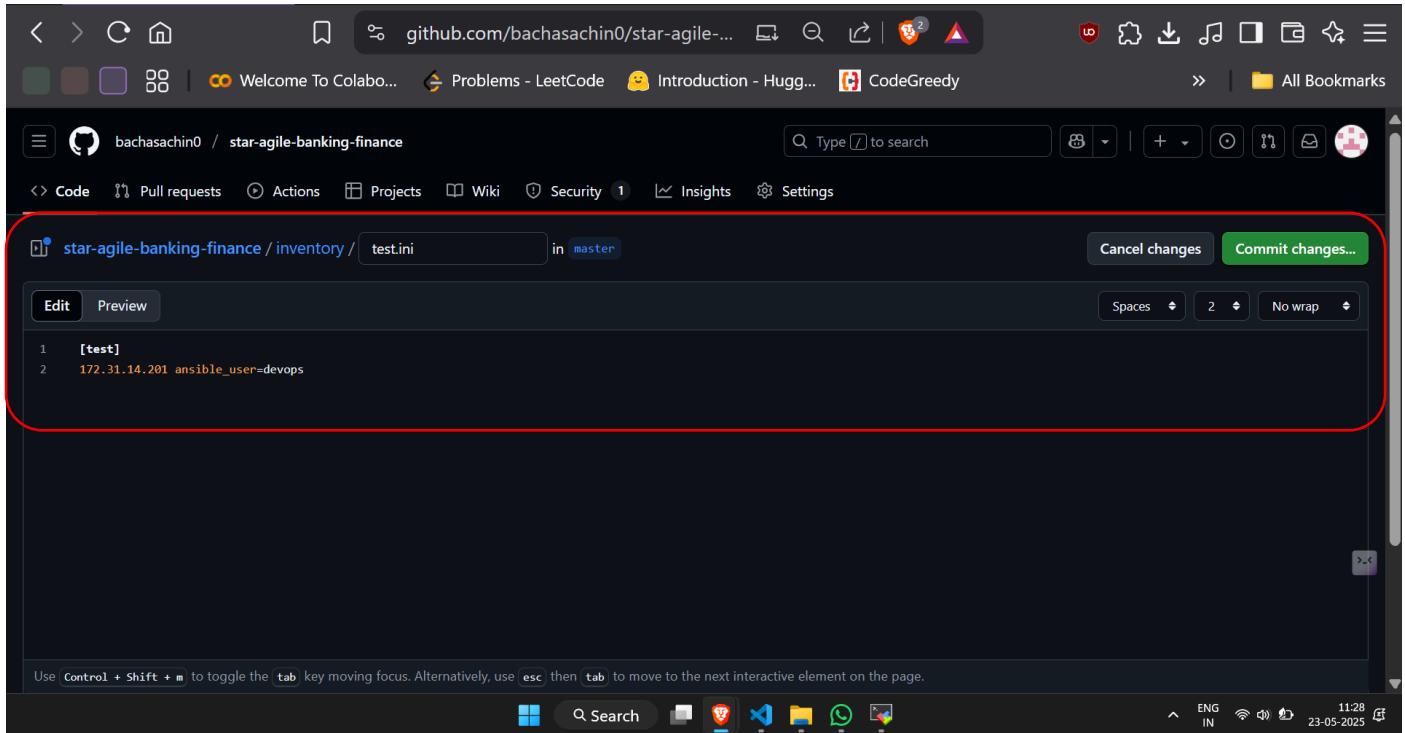
The screenshot shows the Jenkins 'Global credentials (unrestricted)' list page. A red box highlights the table displaying the added credentials. The table has columns for ID, Name, Kind, and Description. It shows two entries: 'ansible-ssh' (Name: devops, Kind: SSH Username with private key) and 'dockercreds' (Name: sachinbacha/\*\*\*\*\*, Kind: Username with password). The 'Add Credentials' button is located at the top right of the table area.

ID	Name	Kind	Description
ansible-ssh	devops	SSH Username with private key	
dockercreds	sachinbacha/*****	Username with 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.

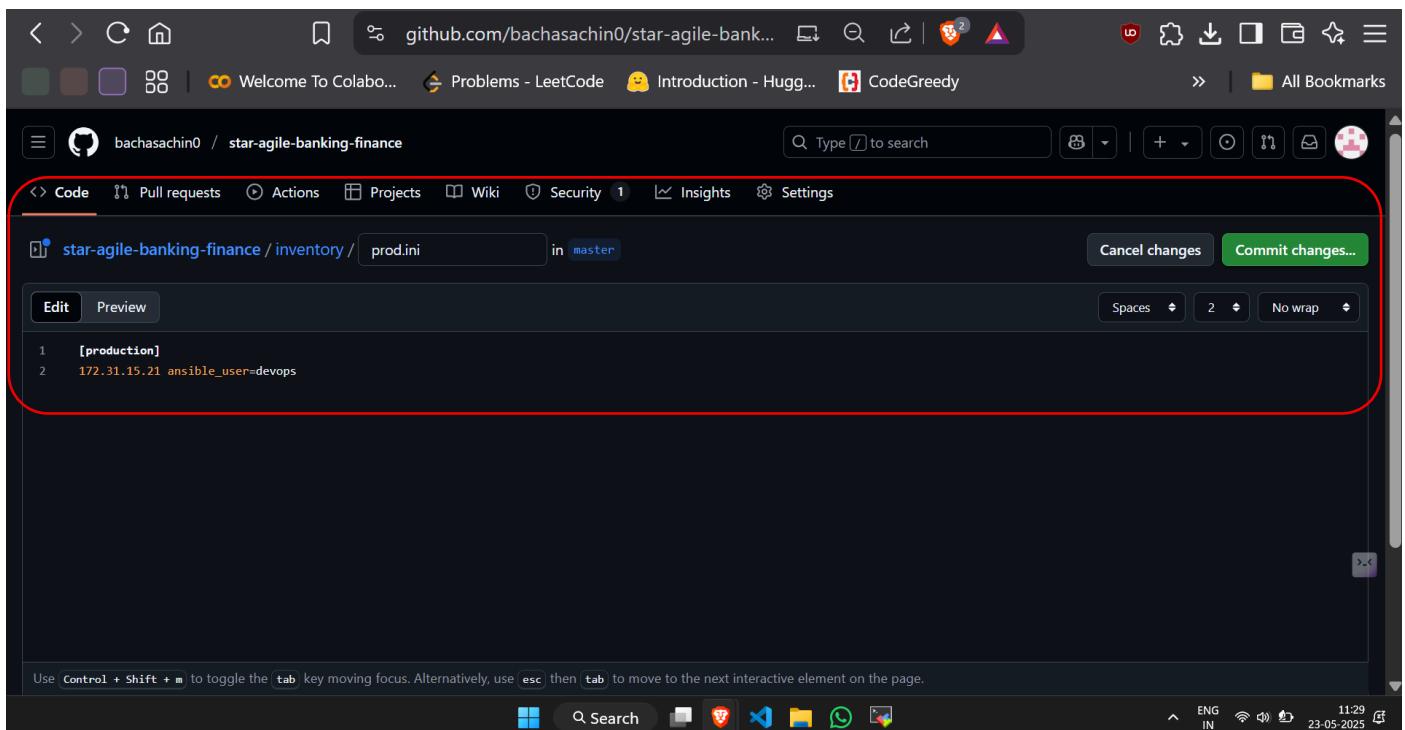


The screenshot shows a GitHub code editor interface. The URL in the address bar is `github.com/bachasachin0/star-agile-banking-finance`. The repository name is `star-agile-banking-finance`. The file being edited is `test.ini` in the `master` branch. The code in the editor is:

```
[test]
172.31.14.201 ansible_user=devops
```

At the top right of the editor, there are two buttons: `Cancel changes` and `Commit changes...`. Below the editor, there are buttons for `Edit` and `Preview`. At the bottom of the editor area, there are buttons for `Spaces`, `2`, and `No wrap`.

Add the private ip of the production server



The screenshot shows a GitHub code editor interface. The URL in the address bar is `github.com/bachasachin0/star-agile-banking-finance`. The repository name is `star-agile-banking-finance`. The file being edited is `prod.ini` in the `master` branch. The code in the editor is:

```
[production]
172.31.15.21 ansible_user=devops
```

At the top right of the editor, there are two buttons: `Cancel changes` and `Commit changes...`. Below the editor, there are buttons for `Edit` and `Preview`. At the bottom of the editor area, there are buttons for `Spaces`, `2`, and `No wrap`.

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

New Item

Enter an item name

banking-pipeline

Select an item type

**Pipeline** Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

OK

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

banking-pipeline Config - Jenkins

Description

This is a Banking project CICD pipeline

Plain text Preview

Discard old builds ?

Do not allow concurrent builds

Do not allow the pipeline to resume if the controller restarts

GitHub project

Project url ?

https://github.com/bachasachin/star-agile-banking-finance

Save Apply

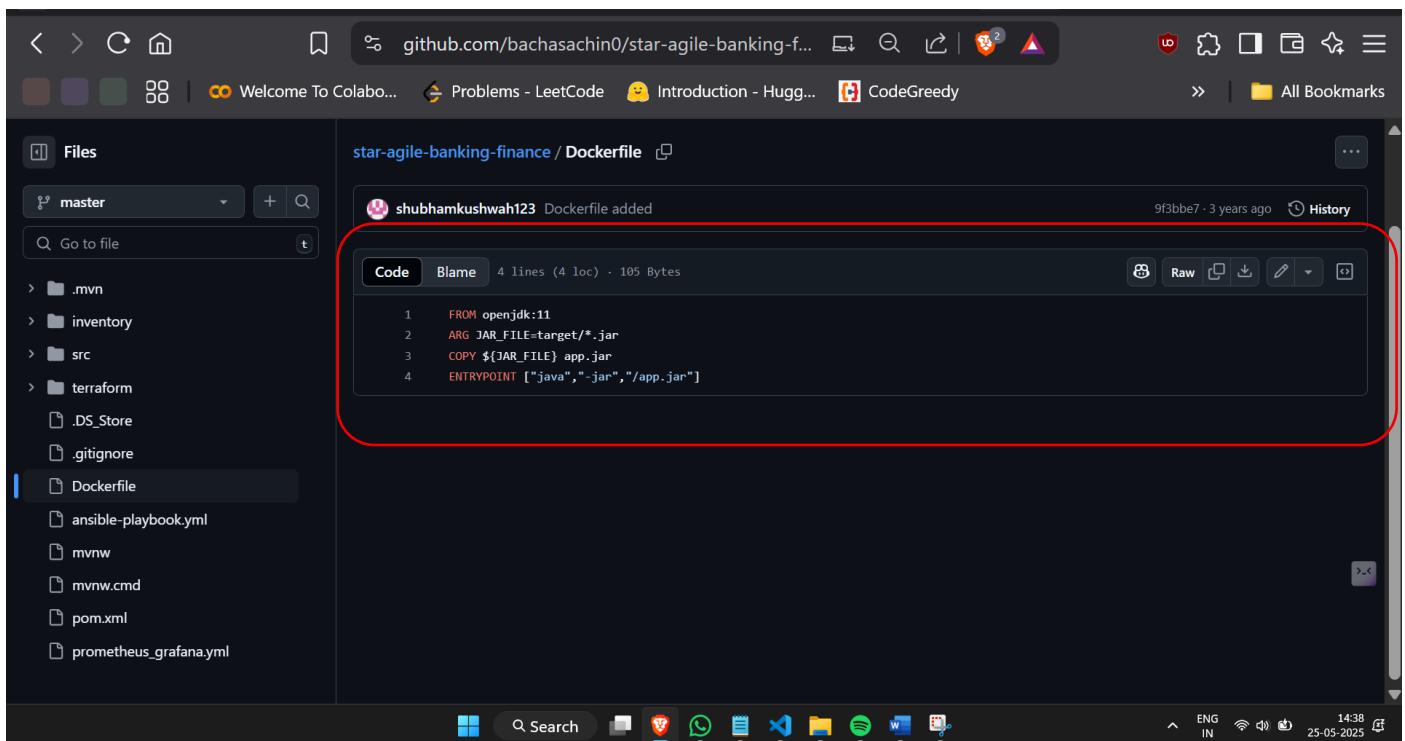
Add the triggers github poll scm and poll scm

The screenshot shows the Jenkins configuration page for a pipeline named 'banking-pipeline'. The 'Triggers' section is highlighted with a red box. It contains options for 'Build after other projects are built', 'Build periodically', 'GitHub hook trigger for GITScm polling' (which is checked), and 'Poll SCM' (which is also checked). Below these is a 'Schedule' field containing 'H \* \* \* \*'. A note below the schedule says: 'Would last have run at Friday, May 23, 2025 at 5:38:00 AM Coordinated Universal Time; would next run at Friday, May 23, 2025 at 6:38:00 AM Coordinated Universal Time.' At the bottom are 'Save' and 'Apply' buttons.

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

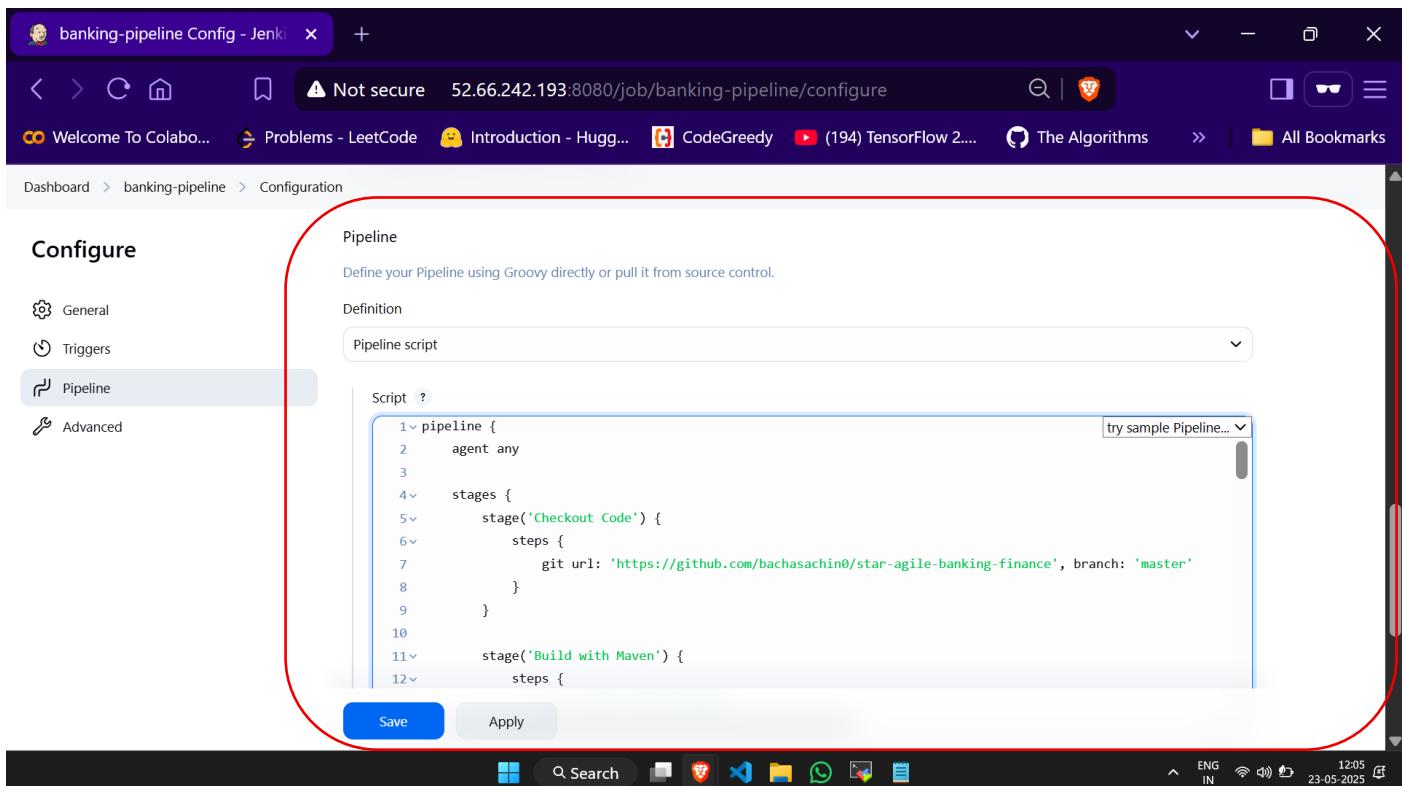
The screenshot shows a GitHub repository page for 'star-agile-banking-finance'. The left sidebar shows files: '.mvn', 'inventory', 'src', 'terraform', '.DS\_Store', '.gitignore', 'Dockerfile', 'ansible-playbook.yml' (which is selected and highlighted with a red box), 'mvnw', 'mvnw.cmd', 'pom.xml', and 'prometheus\_grafana.yml'. The main area shows the 'ansible-playbook.yml' file content:

```
1  ---
2  - name: Configure Docker on EC2 Instances
3  hosts: all
4  become: true
5  connection: ssh
6  tasks:
7
8      - name: Remove conflicting containerd packages
9        apt:
10          name:
11              - containerd
12              - containerd.io
13          state: absent
14          purge: yes
15          autoremove: yes
16
17      - name: Update apt cache
18        apt:
19            update_cache: yes
```



```
FROM openjdk:11
ARG JAR_FILE=target/*.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
```

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 .



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

## Jenkins pipeline script

```
pipeline {
    agent any
    stages {
        stage('Checkout Code') {
            steps {
                git url: 'https://github.com/bachasachino/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

The screenshot shows the Docker Hub interface for the repository `sachinbacha/banking`. The left sidebar shows the user's personal space with options like Repositories, Collaborations, Settings, Default privacy, Notifications, Billing, Usage, Pulls, and Storage. The main content area displays the repository details, including the latest tag (`latest`) which was pushed 1 minute ago. To the right, there are Docker commands for pushing a new tag and a public view link. A sidebar advertisement for `buildcloud` is visible, mentioning Docker Build Cloud and shared caching.

Check the pipeline build status it gets executed

The screenshot shows the Jenkins pipeline interface for the `banking-pipeline`. The left sidebar provides navigation options like Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, GitHub, Stages, Rename, Pipeline Syntax, GitHub Hook Log, and Polling Log. The main content area displays the `Stage View` for the `banking-pipeline`, showing the execution times for various stages: Checkout Code (1s), Build with Maven (24s), Build Docker Image (16s), Push Docker Image (21s), Deploy to Test Using Ansible (1min 9s), Wait for Test App to Be Ready (5s), Run Selenium Tests on Test machine (11s), Check Build Status (202ms), Deploy to Production (1min 9s), Print App URLs (358ms), and Declarative Post Actions (117ms). A summary at the bottom indicates an average stage time of 1min 46s. A red box highlights the Stage View table and the overall pipeline summary.

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

The screenshot shows a Jenkins pipeline interface. At the top, there are tabs for 'banking-pipeline - Jenkins' and 'sachinbacha/banking | Docker Hub'. The main content area is titled 'Stage Logs (Print App URLs)'. It contains a log entry:

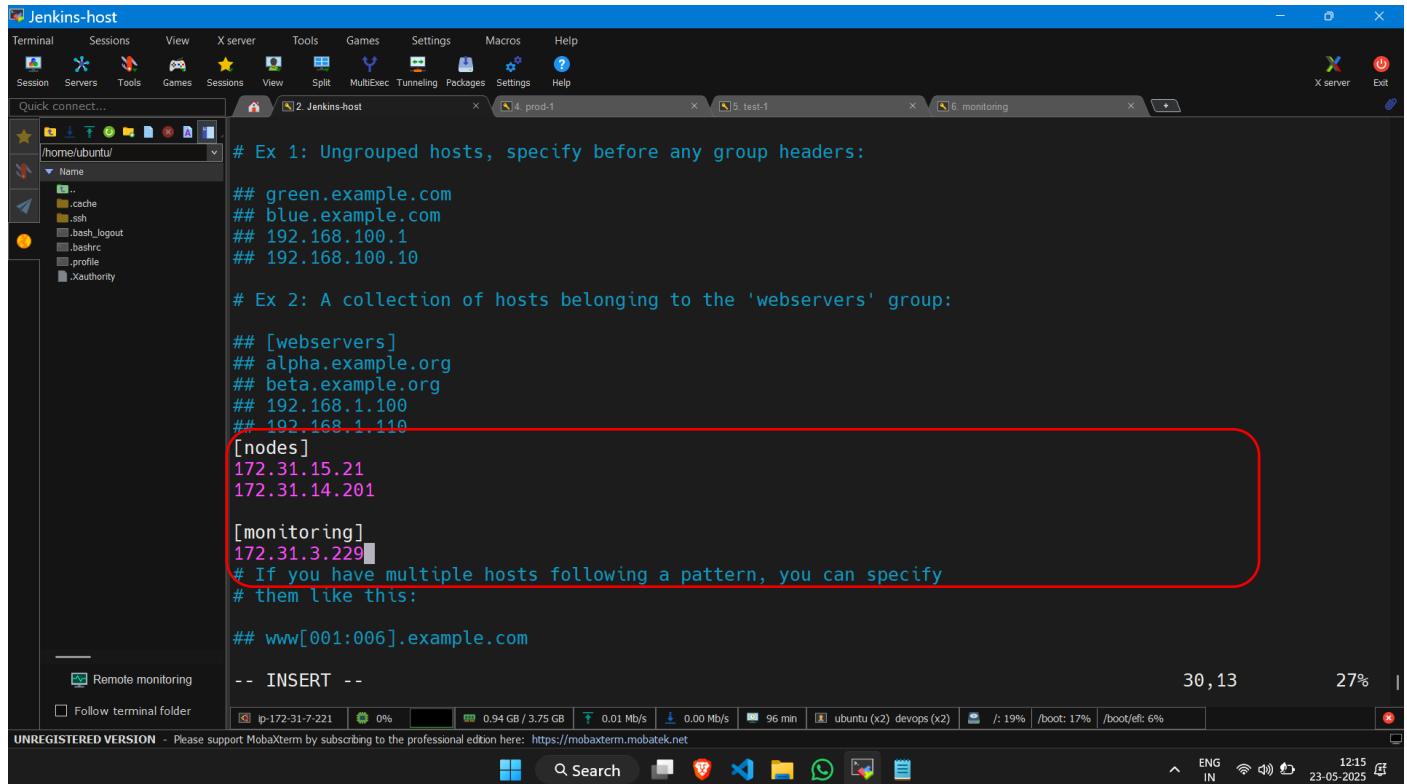
```
Shell Script -- echo "👉 Test App: http://3.109.56.235:8084" echo "👉 Production App: http://13.203.210.228:8084" (self time 282ms)
+ echo 👉 Test App: http://3.109.56.235:8084
👉 Test App: http://3.109.56.235:8084
+ echo 👉 Production App: http://13.203.210.228:8084
👉 Production App: http://13.203.210.228:8084
```

Below the logs is a 'Stage View' section with a timeline. The timeline shows various stages: Checkout Code, Build with Maven, Build Docker Image, Push Docker Image, Deploy to Test Using Ansible, Wait for Test App to Be Ready, Run Selenium Tests on Test machine, Check Build Status, Deploy to Production, Print App URLs, and Declarative: Post Actions. Each stage has a color-coded bar indicating its duration. A red box highlights the 'Print App URLs' stage.

Now after clicking the link you are able to access the application on the machine

The screenshot shows a web browser window with the address bar containing '13.203.210.228:8084'. The page itself is for 'Customer Banking Services' (CBS). The CBS logo is at the top left. The main heading is 'CUSTOMER BANKING SERVICES'. Below it, a subtext reads: 'We provide the World's best in class Banking Solutions and Services.' To the right, there is a stylized illustration of two people working with a computer and a laptop, surrounded by charts and graphs. A red box highlights the URL in the address bar.

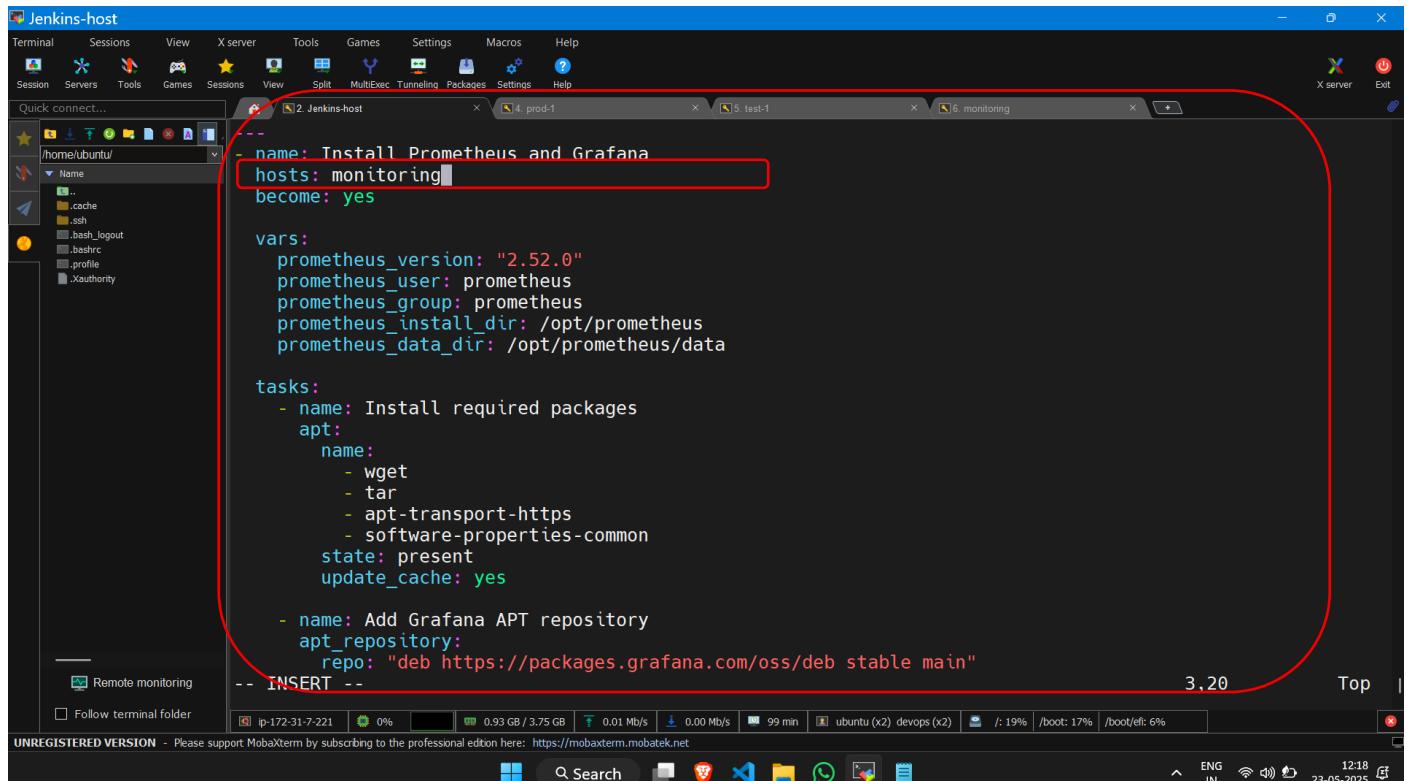
Now also connect the monitoring instance to the ansible host to install Prometheus and Grafana and configure them from the controller node itself . add the private ip of the monitoring node in to the hosts inventory file



The screenshot shows a terminal window titled "Jenkins-host" in MobaXterm. The window contains a configuration file for hosts inventory. A red box highlights the section where the IP address 172.31.3.229 is listed under the [monitoring] group. The file includes comments explaining host specification patterns.

```
# Ex 1: Ungrouped hosts, specify before any group headers:  
## green.example.com  
## blue.example.com  
## 192.168.100.1  
## 192.168.100.10  
  
# Ex 2: A collection of hosts belonging to the 'webservers' group:  
## [webservers]  
## alpha.example.org  
## beta.example.org  
## 192.168.1.100  
## 192.168.1.110  
  
[nodes]  
172.31.15.21  
172.31.14.201  
  
[monitoring]  
172.31.3.229  
# If you have multiple hosts following a pattern, you can specify  
# them like this:  
  
## www[001:006].example.com  
  
-- INSERT --  
ip-172-31-7-221 0% 0.94 GB / 3.75 GB 0.01 Mb/s 0.00 Mb/s 96 min ubuntu (x2) devops (x2) /: 19% /boot: 17% /boot/efi: 6%
```

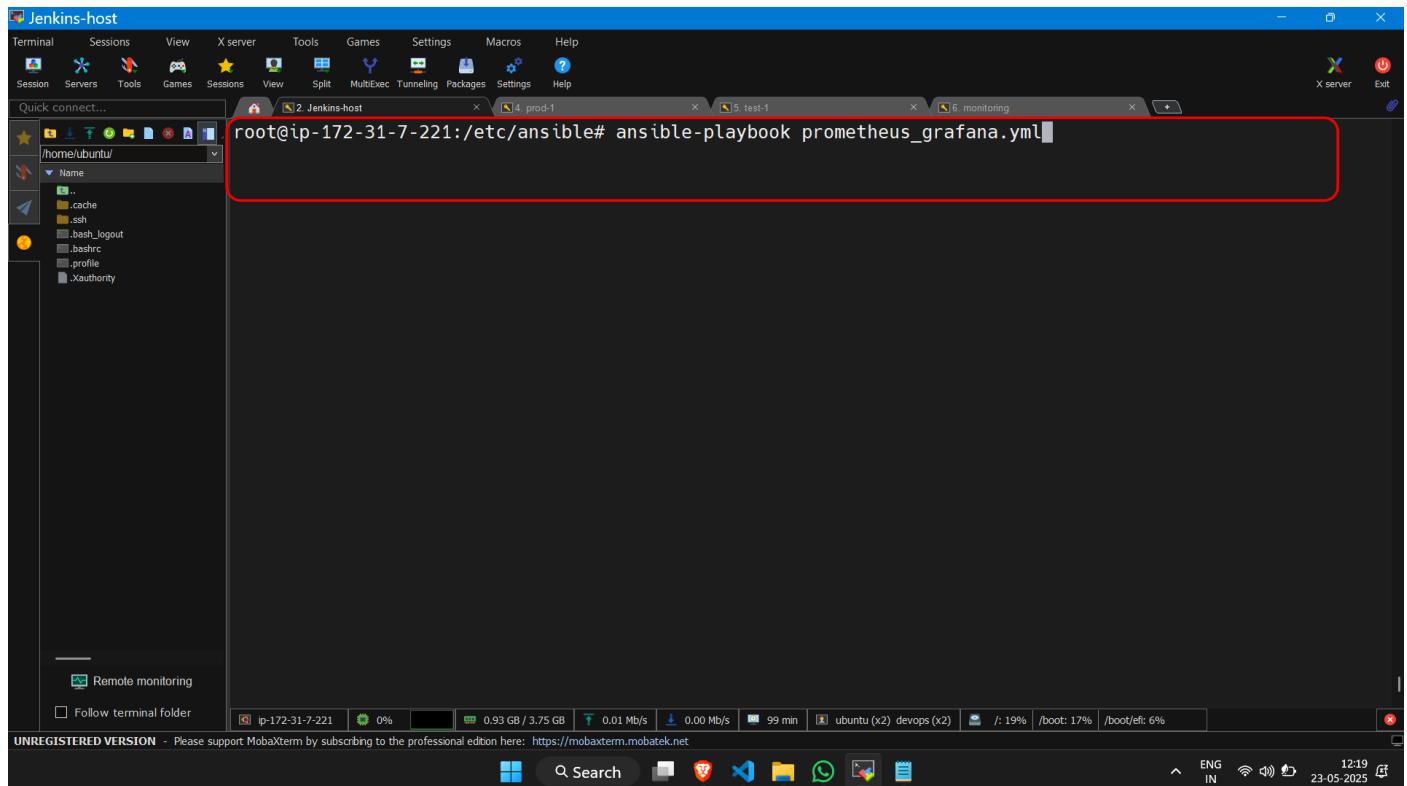
Now write a ansible-playbook to install Prometheus and Grafana into the target machines



The screenshot shows a terminal window titled "Jenkins-host" in MobaXterm. The window contains an Ansible playbook. A red box highlights the "hosts: monitoring" line, indicating the target hosts for the play. The playbook defines variables for Prometheus installation and lists tasks for package installation and repository addition.

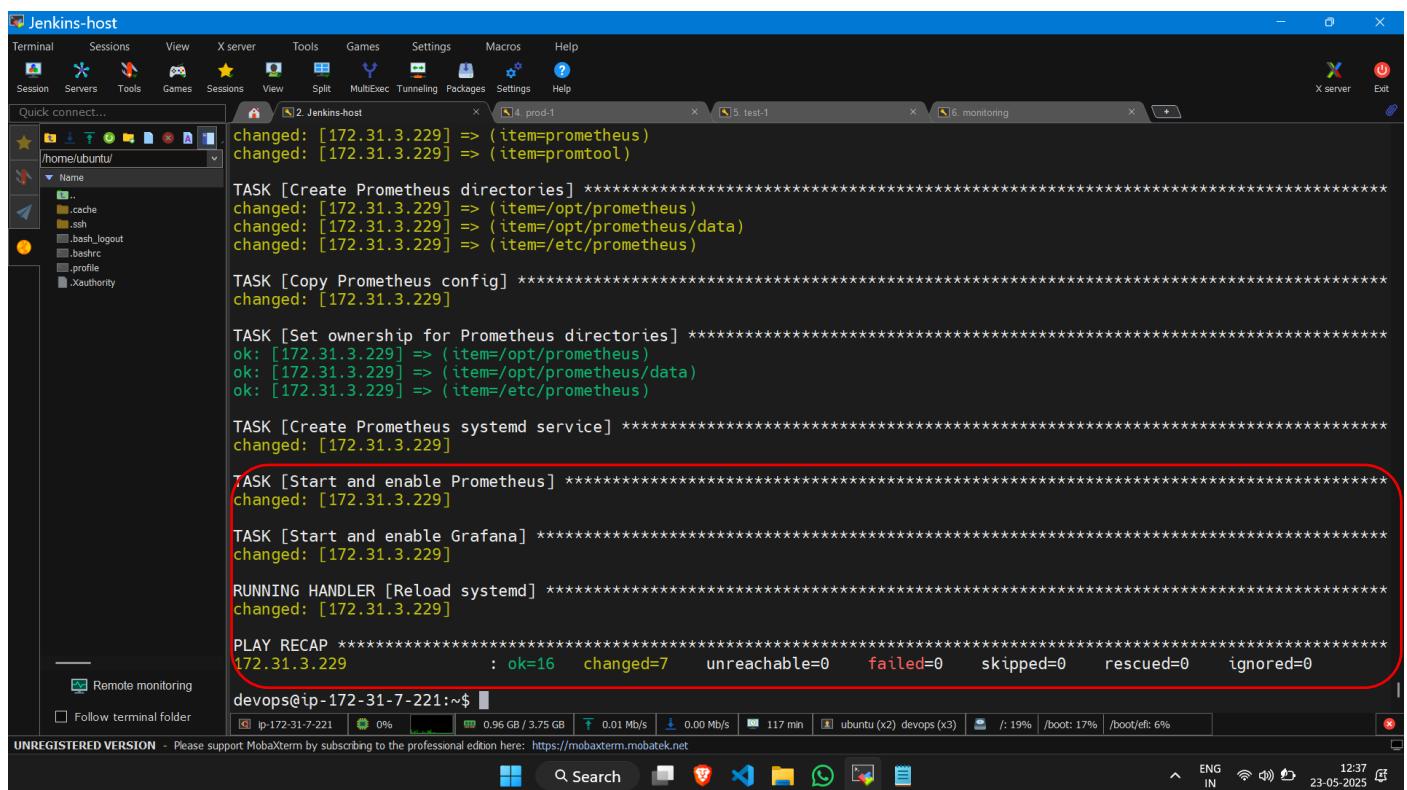
```
name: Install Prometheus and Grafana  
hosts: monitoring  
become: yes  
  
vars:  
  prometheus_version: "2.52.0"  
  prometheus_user: prometheus  
  prometheus_group: prometheus  
  prometheus_install_dir: /opt/prometheus  
  prometheus_data_dir: /opt/prometheus/data  
  
tasks:  
  - name: Install required packages  
    apt:  
      name:  
        - wget  
        - tar  
        - apt-transport-https  
        - software-properties-common  
      state: present  
      update_cache: yes  
  
  - name: Add Grafana APT repository  
    apt_repository:  
      repo: "deb https://packages.grafana.com/oss/deb stable main"  
-- INSERT --  
ip-172-31-7-221 0% 0.93 GB / 3.75 GB 0.01 Mb/s 0.00 Mb/s 99 min ubuntu (x2) devops (x2) /: 19% /boot: 17% /boot/efi: 6%
```

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



```
root@ip-172-31-7-221:/etc/ansible# ansible-playbook prometheus_grafana.yml
```

Now Prometheus and Grafana will get installed into the monitoring machine



```
changed: [172.31.3.229] => (item=prometheus)
changed: [172.31.3.229] => (item=promtool)

TASK [Create Prometheus directories] *****
changed: [172.31.3.229] => (item=/opt/prometheus)
changed: [172.31.3.229] => (item=/opt/prometheus/data)
changed: [172.31.3.229] => (item=/etc/prometheus)

TASK [Copy Prometheus config] *****
changed: [172.31.3.229]

TASK [Set ownership for Prometheus directories] *****
ok: [172.31.3.229] => (item=/opt/prometheus)
ok: [172.31.3.229] => (item=/opt/prometheus/data)
ok: [172.31.3.229] => (item=/etc/prometheus)

TASK [Create Prometheus systemd service] *****
changed: [172.31.3.229]

TASK [Start and enable Prometheus] *****
changed: [172.31.3.229]

TASK [Start and enable Grafana] *****
changed: [172.31.3.229]

RUNNING HANDLER [Reload systemd] *****
changed: [172.31.3.229]

PLAY RECAP *****
172.31.3.229      : ok=16   changed=7    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

devops@ip-172-31-7-221:~$
```

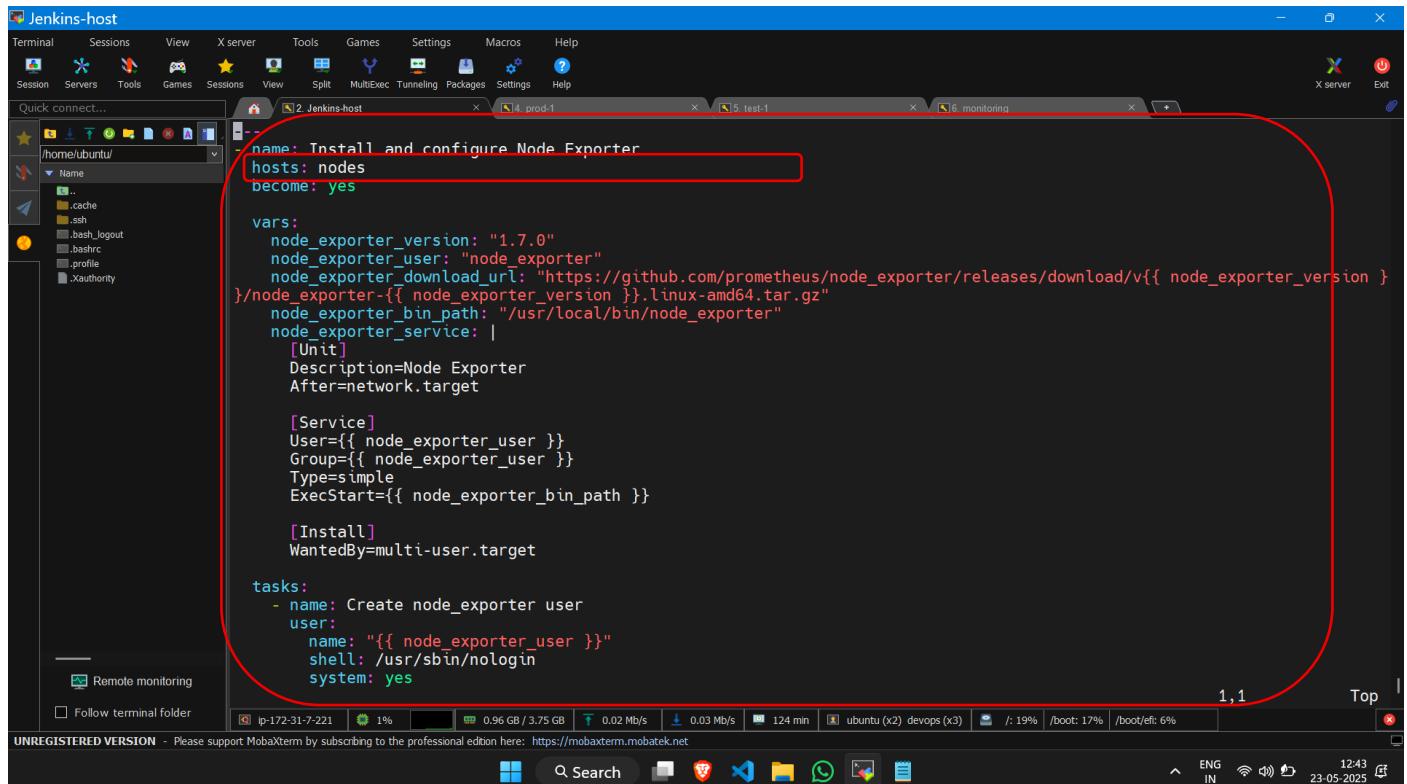
Now access the Prometheus at public ip of the monitoring machine on port 9090 to check if they are installed and running

The screenshot shows a browser window with the URL `3.110.168.33:9090/graph?g0.expr=&g0.tab=1&g0.display_mode=lin...`. The title bar says "Prometheus Time Series". The interface includes a navigation bar with "Prometheus", "Alerts", "Graph", "Status", and "Help". Below it are checkboxes for "Use local time", "Enable query history", "Enable autocomplete" (checked), "Enable highlighting" (checked), and "Enable linter". A search bar contains the placeholder "Expression (press Shift+Enter for newlines)". A large panel below displays the message "No data queried yet". At the bottom right of the panel is a "Remove Panel" button. A blue "Add Panel" button is located at the bottom left. The status bar at the bottom right shows "12:38 23-05-2025".

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

The screenshot shows a browser window with the URL `3.110.168.33:3000/?orgId=1&from=now-6h&to=now&timezone=br...`. The title bar says "Home - Dashboards". The interface includes a navigation bar with "Home", "Dashboards", and "Home". A search bar is present. The main area features a "Welcome to Grafana" message and a "Need help?" section with links to "Documentation", "Tutorials", "Community", and "Public Slack". On the left, a "Basic" section provides instructions for setting up Grafana. To the right, there are three panels: "TUTORIAL" (with sections for "DATA SOURCE AND DASHBOARDS" and "Grafana fundamentals"), "DATA SOURCES" (with a link to "Add your first data source"), and "DASHBOARDS" (with a link to "Create your first dashboard"). A "Remove this panel" link is located in the top right corner of the main content area. The status bar at the bottom right shows "12:40 23-05-2025".

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



```

name: Install and configure Node Exporter
hosts: nodes
become: yes

vars:
  node_exporter_version: "1.7.0"
  node_exporter_user: "node_exporter"
  node_exporter_download_url: "https://github.com/prometheus/node_exporter/releases/download/v{{ node_exporter_version }}/{{ node_exporter_version }}.linux-amd64.tar.gz"
  node_exporter_bin_path: "/usr/local/bin/node_exporter"
  node_exporter_service: |
    [Unit]
    Description=Node Exporter
    After=network.target

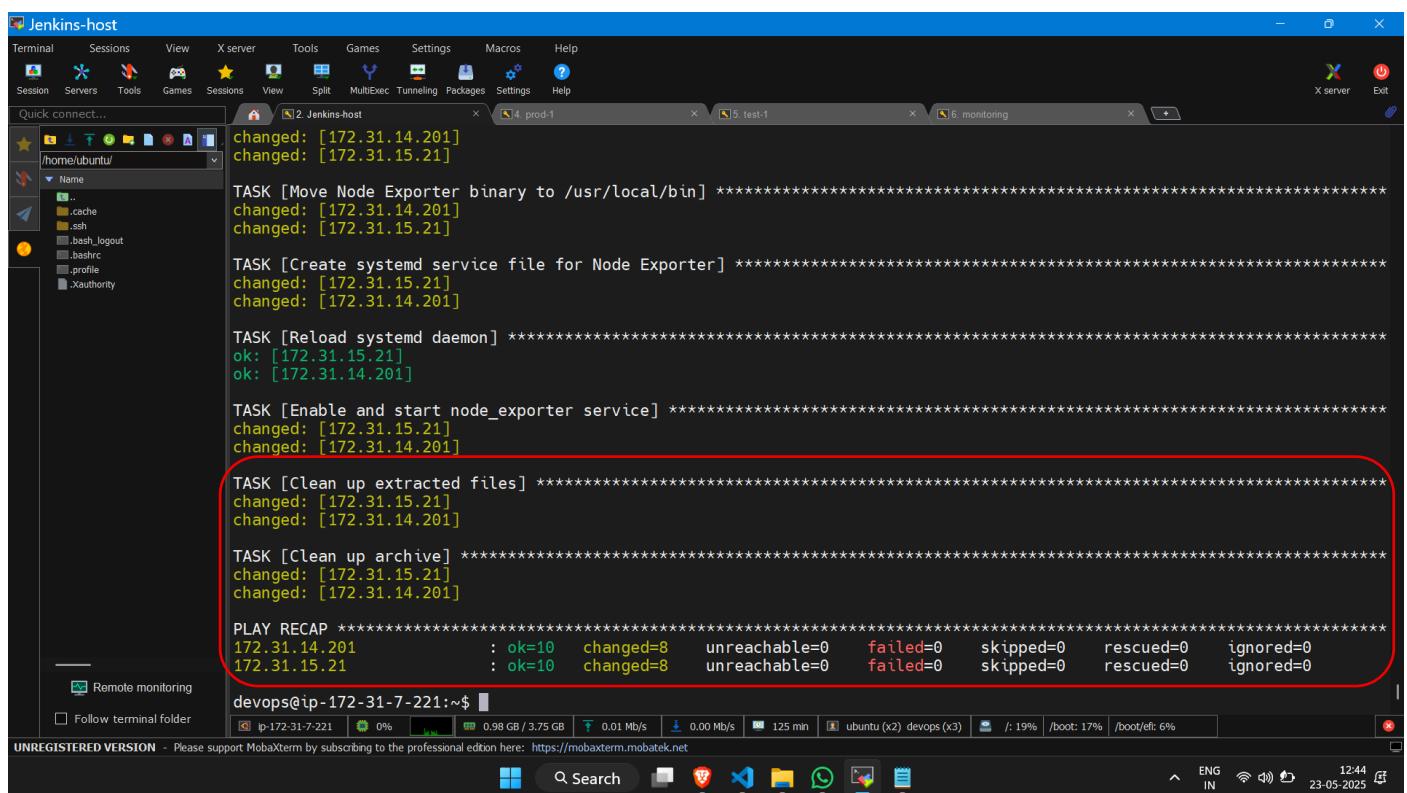
    [Service]
    User={{ node_exporter_user }}
    Group={{ node_exporter_user }}
    Type=simple
    ExecStart={{ node_exporter_bin_path }}

    [Install]
    WantedBy=multi-user.target

tasks:
- name: Create node_exporter user
  user:
    name: "{{ node_exporter_user }}"
    shell: /usr/sbin/nologin
    system: yes

```

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.



```

changed: [172.31.14.201]
changed: [172.31.15.21]

TASK [Move Node Exporter binary to /usr/local/bin] ****
changed: [172.31.14.201]
changed: [172.31.15.21]

TASK [Create systemd service file for Node Exporter] ****
changed: [172.31.15.21]
changed: [172.31.14.201]

TASK [Reload systemd daemon] ****
ok: [172.31.15.21]
ok: [172.31.14.201]

TASK [Enable and start node_exporter service] ****
changed: [172.31.15.21]
changed: [172.31.14.201]

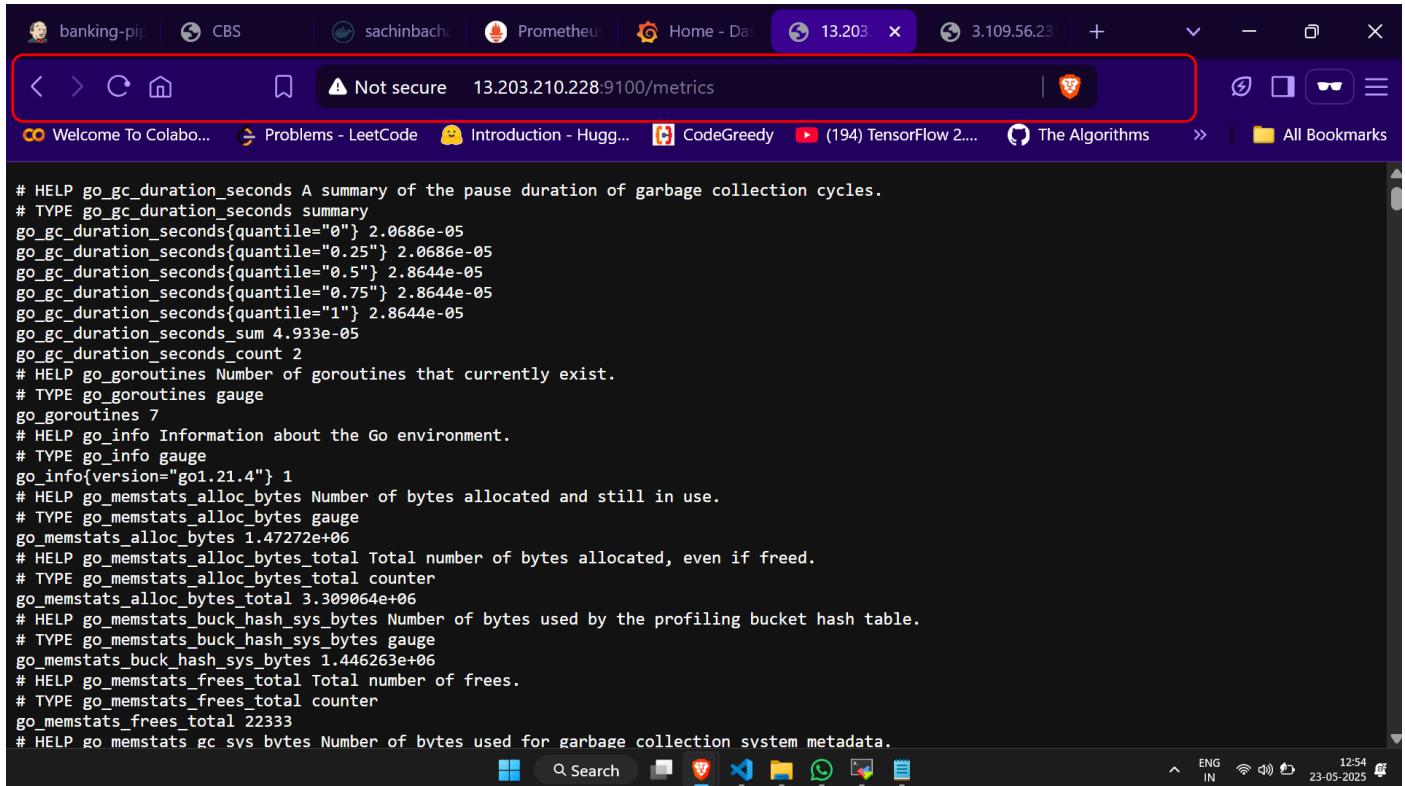
TASK [Clean up extracted files] ****
changed: [172.31.15.21]
changed: [172.31.14.201]

TASK [Clean up archive] ****
changed: [172.31.15.21]
changed: [172.31.14.201]

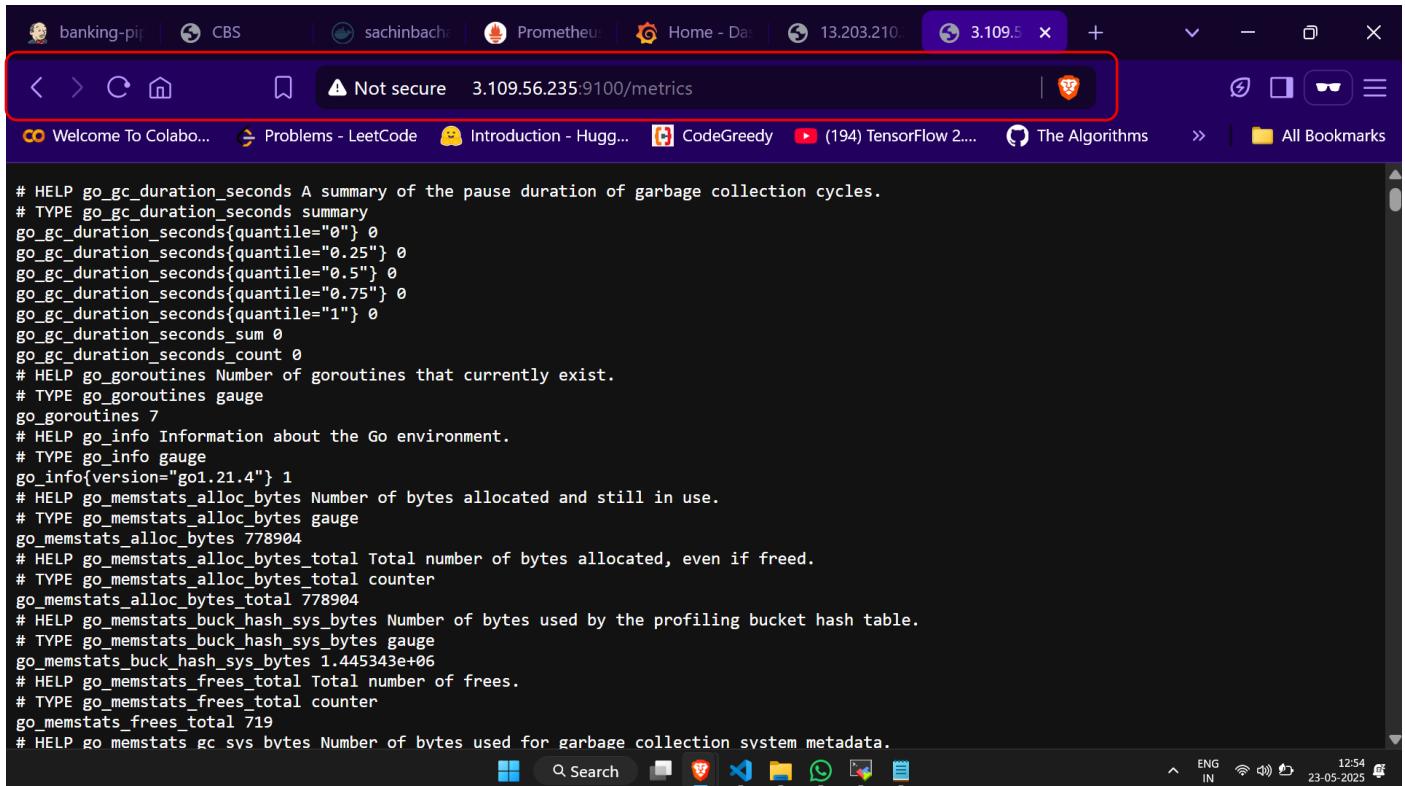
PLAY RECAP ****
172.31.14.201 : ok=10    changed=8      unreachable=0    failed=0      skipped=0      rescued=0      ignored=0
172.31.15.21  : ok=10    changed=8      unreachable=0    failed=0      skipped=0      rescued=0      ignored=0

```

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



```
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 2.0686e-05
go_gc_duration_seconds{quantile="0.25"} 2.0686e-05
go_gc_duration_seconds{quantile="0.5"} 2.8644e-05
go_gc_duration_seconds{quantile="0.75"} 2.8644e-05
go_gc_duration_seconds{quantile="1"} 2.8644e-05
go_gc_duration_seconds_sum 4.933e-05
go_gc_duration_seconds_count 2
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 7
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.21.4"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 1.47272e+06
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 3.309064e+06
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 1.446263e+06
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 22333
# HELP go_memstats_gc_ssvs_bytes Number of bytes used for garbage collection system metadata.
```



```
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 0
go_gc_duration_seconds{quantile="0.25"} 0
go_gc_duration_seconds{quantile="0.5"} 0
go_gc_duration_seconds{quantile="0.75"} 0
go_gc_duration_seconds{quantile="1"} 0
go_gc_duration_seconds_sum 0
go_gc_duration_seconds_count 0
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 7
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.21.4"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 778904
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 778904
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 1.445343e+06
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 719
# HELP go_memstats_gc_ssvs_bytes Number of bytes used for garbage collection system metadata.
```

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

The screenshot shows a terminal window titled 'Jenkins-host' in MobaXterm. The terminal displays an Ansible playbook for configuring a Node Exporter scrape job in a Prometheus configuration file. A red box highlights the 'vars' section where two IP addresses are defined for 'node\_exporter\_targets'. The 'tasks' section includes a task to add the job to the Prometheus config file and another to print the updated file.

```
name: Add Node Exporter scrape job and show config
hosts: monitoring
become: yes

vars:
  node_exporter_targets:
    - '13.203.210.228:9100'
    - '3.109.56.235:9100'

  node_exporter_scrape_job: |
    - job_name: 'node_exporter'
      scrape_interval: 15s
      static_configs:
        - targets: {{ node_exporter_targets | to_nice_yaml(indent=12) | indent(12, false) }}

tasks:
  - name: Add Node Exporter scrape job to prometheus.yml
    blockinfile:
      path: /etc/prometheus/prometheus.yml
      marker: "# {mark} ANSIBLE MANAGED NODE EXPORTER SCRAPE JOB"
      insertafter: '^scrape_configs:'
      block: "{{ node_exporter_scrape_job }}"
    notify:
      - Reload Prometheus

  - name: Show contents of prometheus.yml after changes
    shell: cat /etc/prometheus/prometheus.yml
    register: prometheus_config_contents

  - name: Print prometheus.yml
    shell: cat /etc/prometheus/prometheus.yml
```

Run the playbook and the Prometheus file gets added with the ip address of the node\_exporter ip's

The screenshot shows the terminal output of the Ansible playbook execution. It starts with a warning about Python interpreter discovery. The output then shows the tasks being run: adding the scrape job to the Prometheus config file and printing the updated file. A red box highlights the printed Prometheus configuration file, which now includes the two IP addresses defined in the playbook's variables. The final output shows a recap of the play.

```
[WARNING]: Platform linux on host 172.31.3.229 is using the discovered Python interpreter at /usr/bin/python3.12, but future installation of another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-core/2.18/reference_appendices/interpreter_discovery.html for more information.
ok: [172.31.3.229]

TASK [Add Node Exporter scrape job to prometheus.yml] *****
changed: [172.31.3.229]

TASK [Show contents of prometheus.yml after changes] *****
changed: [172.31.3.229]

TASK [Print prometheus.yml] *****
ok: [172.31.3.229] => {
  "msg": "# my global config\\nglobal:\\n  scrape_interval: 15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.\\n  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.\\n  # scrape_timeout is set to the global_default (10s).\\n\\# Alertmanager configuration\\nalerting:\\n  alertmanagers:\\n    - static_configs:\\n      - targets:\\n          # - alertmanager:9093\\n          # Load rules once and periodically evaluate them according to the global 'evaluation_interval'.\\nrule_files:\\n  # - \"first_rules.yml\"\\n  # - \"second_rules.yml\"\\n\\# A scrape configuration containing exactly one endpoint to scrape:\\n  # Here it's Prometheus itself.\\nscrape_configs:\\n    # BEGIN ANSIBLE MANAGED NODE EXPORTER SCRAPE JOB\\n    - job_name: 'node_exporter'\\n      scrape_interval: 15s\\n      static_configs:\\n        - targets: - 13.203.210.228:9100\\n          - 3.109.56.235:9100\\n        # END ANSIBLE MANAGED NODE EXPORTER SCRAPE JOB\\n      # The job name is added as a label 'job=<job_name>' to any timeseries scraped from this config.\\n      # - job_name: \"prometheus\"\\n      # metrics_path defaults to '/metrics'\\n      # scheme defaults to 'http'\\n      static_configs:\\n        - targets: [\"localhost:9090\"]\\n    }\\n\\nRUNNING HANDLER [Reload Prometheus] *****
changed: [172.31.3.229]

PLAY RECAP *****
172.31.3.229 : ok=5    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

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

The screenshot shows the Prometheus Targets page. At the top, there are filters for 'All', 'Unhealthy', and 'Collapse All'. A search bar is present, along with buttons for 'Unknown', 'Unhealthy', and 'Healthy'. Below this, there are two sections: 'node\_exporter (2/2 up)' and 'prometheus (0/1 up)'. Each section has a 'show less' button. The 'node\_exporter' section contains two entries:

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
<a href="http://13.203.210.228:9100/metrics">http://13.203.210.228:9100/metrics</a>	UP	instance="13.203.210.228:9100" job="node_exporter"	7.942s ago	94.611ms	
<a href="http://3.109.56.235:9100/metrics">http://3.109.56.235:9100/metrics</a>	UP	instance="3.109.56.235:9100" job="node_exporter"	7.883s ago	93.349ms	

The 'prometheus' section is currently empty. The bottom right corner shows system status: ENG IN, 23-05-2025, 13:40.

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 .

The screenshot shows the Grafana Data sources page. On the left, there is a sidebar with options like Home, Bookmarks, Starred, Dashboards, Explore, Drilldown, Alerting, Connections, Data sources (which is selected), and Administration. The main area shows a 'Connections > Data sources > prometheus' path. A 'Connection' section is highlighted with a red box, containing a 'Prometheus server URL \*' input field with the value 'http://3.110.168.33:9090/'. Below this is an 'Authentication' section, which is also highlighted with a red box. The bottom right corner shows system status: ENG IN, 23-05-2025, 13:41.

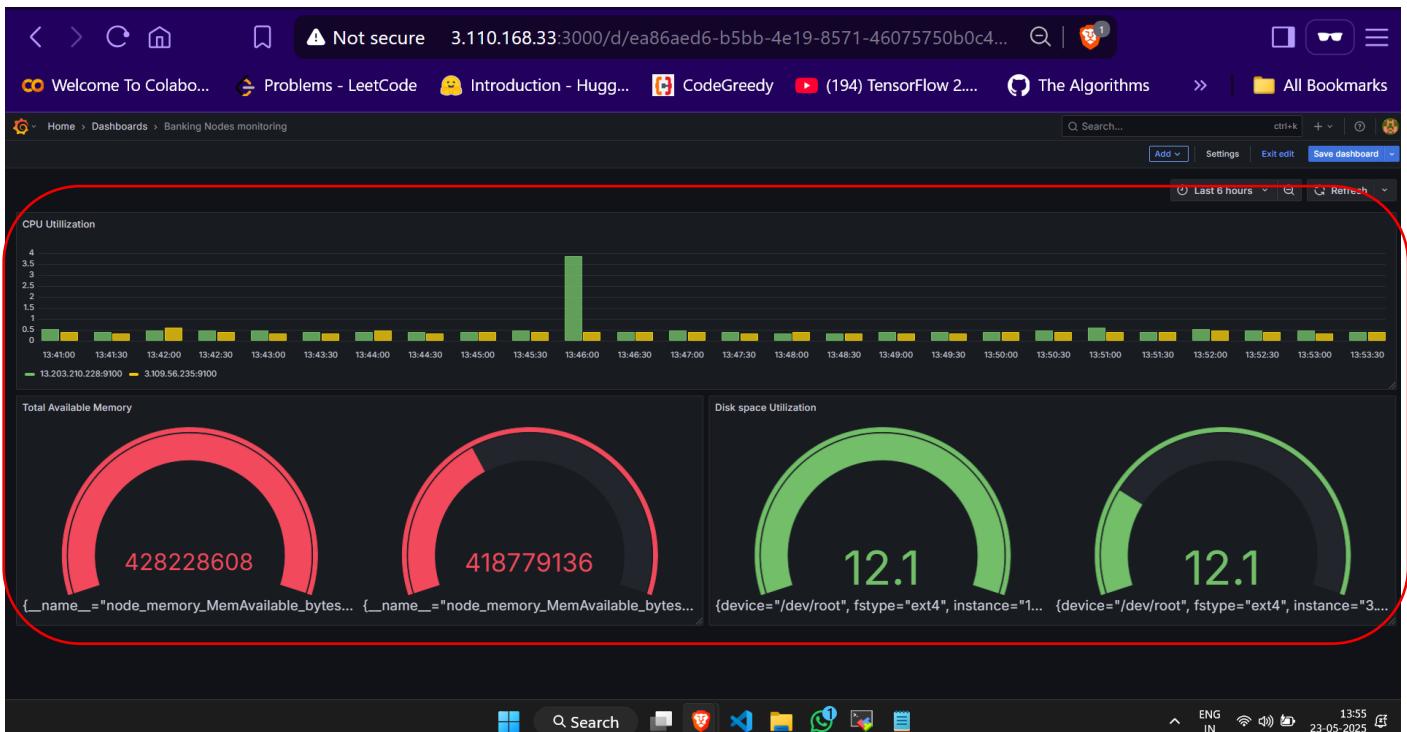
Now create new visualization in the Grafana and create a panel for monitoring the total memory available for both the nodes

The screenshot shows a Grafana dashboard titled "Banking Nodes monitoring". It contains two gauge panels, each with a red arc and a central value. The left gauge has a value of 428228608 and the right one has 418779136. Both gauges are labeled "Total Available Memory". Below the gauges is a query editor for Prometheus. The query is: `{__name__="node_memory_MemAvailable_bytes", instance="13.203.210.228:9100", ...} {__name__="node_memory_MemAvailable_bytes", instance="3.109.56.235:9100", ...}`. The data source is set to "prometheus". On the right side of the dashboard, there is a panel configuration sidebar with options for "Title" (set to "Total Available Memory"), "Value options" (set to "All values"), and "Calculation" (set to "Last"). The bottom right corner of the dashboard shows the date and time as 23-05-2025 13:54.

Now create another panel for CPU utilization

The screenshot shows a Grafana dashboard titled "Banking Nodes monitoring". It contains a bar chart panel titled "CPU Utilization". The chart displays CPU usage over time, with green bars representing one node and yellow bars representing another. The x-axis shows time intervals from 13:41:00 to 13:51:00. A single large green bar is visible around 13:46:00. Below the chart is a query editor for Prometheus. The query is: `100 - (avg_by(instance) (irate(node_cpu_seconds_total{mode='idle'}{5m})) * 100)`. The data source is set to "prometheus". On the right side of the dashboard, there is a panel configuration sidebar with options for "Title" (set to "CPU Utilization"), "Bar chart" (with "X Axis" set to "First string or time field" and "Orientation" set to "Horizontal"), and "X-axis labels minimum spacing" (set to "Large"). The bottom right corner of the dashboard shows the date and time as 23-05-2025 13:52.

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 ipaddress

