

Project -1: Certification Project – Finance Me Banking and Finance Domain

Problems in traditional method implementation:

Building Complex Monolithic Application is difficult.

Manual efforts to test various components/modules of the project

Incremental builds are difficult to manage, test and deploy.

It was not possible to scale up individual modules independently.

Creation of infrastructure and configure it manually is very time consuming

Continuous manual monitoring the application is quite challenging.

In the GitHub repository we have the necessary codes for our project.

<https://github.com/Sonal0409/banking-finance-Project1.git>

To overcome the traditional method issues, using the above code creating a DevOps CICD pipeline which includes.

GIT and Github -> Maintain source code

Terraform to create a EC2 instance

Ansible -> Setup to install required packages

Jenkins -> Create Build Pipeline and integrate polISCM

Docker -> Use docker to build images and push to dockerhub

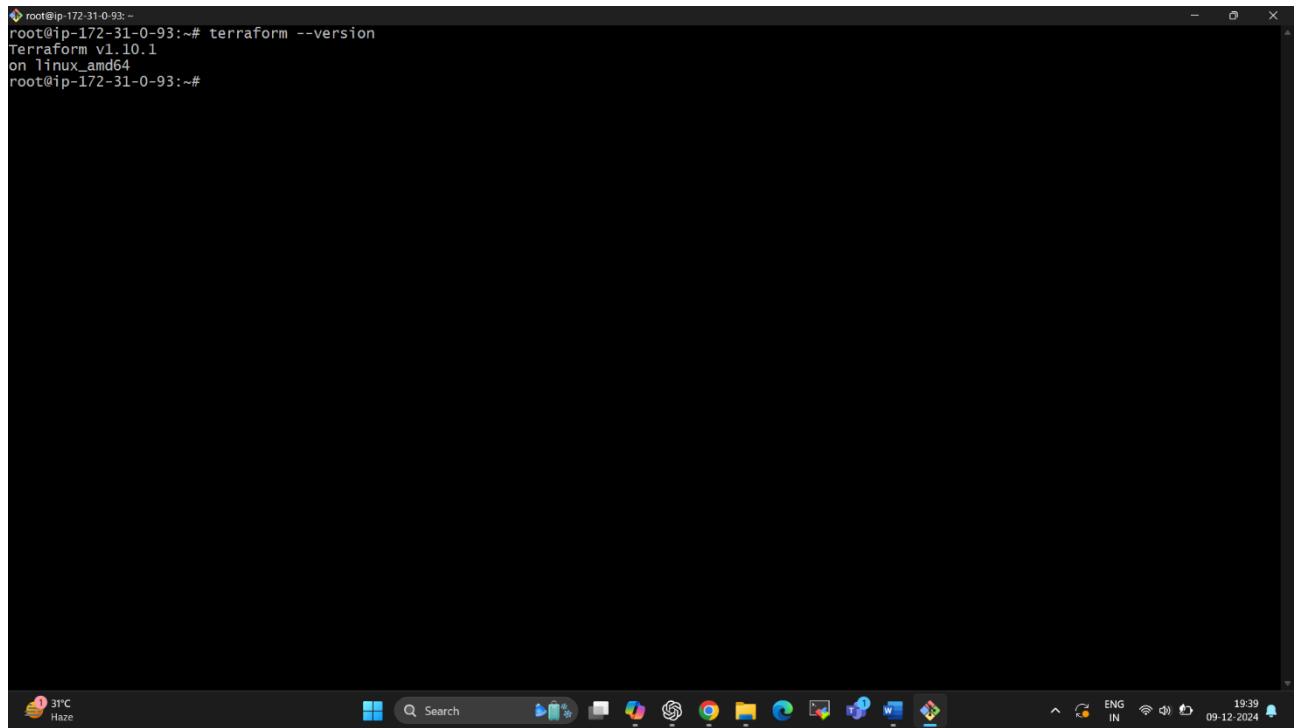
Prometheus and Grafana - monitor Jenkins pipeline

STEPS:

Create a VM and install Terraform on it

OS: ubuntu 22 → Instance type: t2.micro → Connecting to the instance →

Then, install terraform on it.



```
root@ip-172-31-0-93:~# terraform --version
Terraform v1.10.1
on linux_amd64
root@ip-172-31-0-93:~#
```

Creating Accesskey and secret key in AWS for terraform to connect:

Once Accesskey and secret key is created in AWS – IAM

Go and configure it on the terraform VM by the command ‘aws configure’

The screenshot shows the AWS IAM Users page. A green success message at the top states "User created successfully" and "You can view and download the user's password and email instructions for signing in to the AWS Management Console." Below this, a table lists a single user named "terraform1". The table columns include User name, Path, Group, Last activity, MFA, Password age, and Console last sign-in. The "terraform1" row has a checkmark in the User name column and is highlighted in blue. The "Create user" button is visible on the right side of the table.

The screenshot shows the "Create access key" step of the AWS IAM process. It starts with "Step 1: Access key best practices & alternatives" (selected), followed by "Step 2 - optional: Set description tag" and "Step 3: Retrieve access keys". The main content area is titled "Access key best practices & alternatives" and includes a note about avoiding long-term credentials. It then lists five "Use case" options with radio buttons:

- Command Line Interface (CLI)
You plan to use this access key to enable the AWS CLI to access your AWS account.
- Local code
You plan to use this access key to enable application code in a local development environment to access your AWS account.
- Application running on an AWS compute service
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.
- Third-party service
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.
- Application running outside AWS
You plan to use this access key to authenticate workloads running in your data center or other infrastructure outside of AWS that needs to access your AWS resources.

Writing terraform configuration file to create an EC2 server and installing ansible on it.

```
root@ip-172-31-0-93:~/myproject
provider "aws" {
region = "us-east-1"
shared_credentials_files = ["~/.aws/credentials"]
}
resource "tls_private_key" "mykey" {
algorithm = "RSA"
}
resource "aws_key_pair" "aws_key" {
key_name   = "web-key"
public_key = tls_private_key.mykey.public_key_openssh
provisioner "local-exec" {
command = "echo ${tls_private_key.mykey.private_key_openssh} > ./web-key.pem"
}
}
resource "aws_vpc" "s1-vpc" {
cidr_block = "10.0.0.0/16"
tags = {
Name = "s1-vpc"
}
}
resource "aws_subnet" "subnet-1"{
vpc_id = aws_vpc.s1-vpc.id
cidr_block = "10.0.1.0/24"
depends_on = [aws_vpc.s1-vpc]
map_public_ip_on_launch = true
tags = {
Name = "s1-subnet"
}
}
resource "aws_route_table" "s1-route-table"{
vpc_id = aws_vpc.s1-vpc.id
-- INSERT --

```



```
root@ip-172-31-0-93:~/myproject
resource "aws_security_group" "s1-sg" {
name      = "s1-sg"
vpc_id   = aws_vpc.s1-vpc.id
dynamic  "ingress" {
for_each = var.sg_ports
iterator = port
content{
from_port    = port.value
to_port      = port.value
protocol     = "tcp"
cidr_blocks  = ["0.0.0.0/0"]
}
}
egress {
from_port    = 0
to_port      = 0
protocol     = "-1"
cidr_blocks  = ["0.0.0.0/0"]
}
}
resource "aws_instance" "myec2" {
ami          = "ami-053b12d3152c0cc71"
instance_type = "t2.medium"
key_name     = "web-key"
subnet_id    = aws_subnet.subnet-1.id
security_groups = [aws_security_group.s1-sg.id]
tags = {
Name = "finance me-EC2"
}
provisioner "remote-exec" {
connection {
type    = "ssh"
user    = "ubuntu"
private_key = tls_private_key.mykey.private_key_pem
host    = self.public_ip
}
}
inline = [
"sudo apt update",
"sudo apt install software-properties-common",
"sudo add-apt-repository --yes --update ppa:ansible/ansible",
"sudo apt install ansible -y"
]
}
-- INSERT --

```



```
root@ip-172-31-0-93:~# mkdir myproject
root@ip-172-31-0-93:~# cd myproject
root@ip-172-31-0-93:~/myproject# vim terra1.tf
root@ip-172-31-0-93:~/myproject# terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/tls...
- Finding latest version of hashicorp/aws...
+ Installing hashicorp/tls v4.0.6 (signed by HashiCorp)
  - Installed hashicorp/tls v4.0.6 (signed by HashiCorp)
- Installing hashicorp/aws v5.80.0
  - Installed hashicorp/aws v5.80.0 (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-172-31-0-93:~/myproject# |
```

Terraform works at the directory level

Run terraform init command in the terraform directory to initialize and connect to the aws instance.

```
root@ip-172-31-0-93: ~myproject
aws_instance.myec2 (remote-exec): Reading package lists... Done
aws_instance.myec2 (remote-exec): Building dependency tree... 0%
aws_instance.myec2 (remote-exec): Building dependency tree... 50%
aws_instance.myec2 (remote-exec): Building dependency tree... Done
aws_instance.myec2 (remote-exec): Reading state information... 0%
aws_instance.myec2 (remote-exec): Reading state information... 0%
aws_instance.myec2 (remote-exec): Reading package lists... Done
aws_instance.myec2 (remote-exec): 40 packages can be upgraded. Run 'apt list --upgradable' to see them.
aws_instance.myec2 (remote-exec): Reading package lists... 0%
aws_instance.myec2 (remote-exec): Reading package lists... 100%
aws_instance.myec2 (remote-exec): Building dependency tree... 0%
aws_instance.myec2 (remote-exec): Building dependency tree... 0%
aws_instance.myec2 (remote-exec): Building dependency tree... 50%
aws_instance.myec2 (remote-exec): Building dependency tree... 50%
aws_instance.myec2 (remote-exec): Building dependency tree... Done
aws_instance.myec2 (remote-exec): Reading state information... 0%
aws_instance.myec2 (remote-exec): Reading state information... 0%
aws_instance.myec2 (remote-exec): Reading state information... Done
aws_instance.myec2 (remote-exec): The following additional packages will be installed:
aws_instance.myec2 (remote-exec):   python3-software-properties
aws_instance.myec2 (remote-exec): The following packages will be upgraded:
aws_instance.myec2 (remote-exec):   python3-software-properties
aws_instance.myec2 (remote-exec):   software-properties-common
aws_instance.myec2 (remote-exec): 2 upgraded, 0 newly installed, 0 to remove and 38 not upgraded.
aws_instance.myec2 (remote-exec): Need to get 44.1 kB of archives.
aws_instance.myec2 (remote-exec): After this operation, 0 B of additional disk space will be used.
aws_instance.myec2 (remote-exec): Do you want to continue? [Y/n]

aws_instance.myec2: Still creating... [50s elapsed]
aws_instance.myec2: Still creating... [1m0s elapsed]
aws_instance.myec2: Still creating... [1m10s elapsed]
aws_instance.myec2: Still creating... [1m20s elapsed]
aws_instance.myec2: Still creating... [1m30s elapsed]
aws_instance.myec2: Still creating... [1m40s elapsed]
aws_instance.myec2: Still creating... [1m50s elapsed]
aws_instance.myec2: Still creating... [2m0s elapsed]
aws_instance.myec2: Still creating... [2m10s elapsed]
aws_instance.myec2: Still creating... [2m20s elapsed]
aws_instance.myec2: Still creating... [2m30s elapsed]
aws_instance.myec2: Still creating... [2m40s elapsed]
aws_instance.myec2: Still creating... [2m50s elapsed]
aws_instance.myec2: Still creating... [3m0s elapsed]
aws_instance.myec2: Still creating... [3m10s elapsed]
aws_instance.myec2: Still creating... [3m20s elapsed]
```

Install all the tools for CICD

Go the us-east-1 region:

connect to the created t2. medium EC2 instance which has Ansible installed on it → Connect using EC2 instance connect → Add Jenkins key to the server → Make Jenkins apt repo.

The screenshot shows the AWS EC2 Instances page. The left sidebar includes sections for EC2 Global View, Events, Instances (with sub-options like Instances Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations), Images (AMIs, AMI Catalog), Elastic Block Store (Volumes, Snapshots, Lifecycle Manager), and Network & Security (Security Groups, Elastic IPs). The main content area displays 'Instances (1/1) Info'. A search bar at the top allows finding instances by attribute or tag. Below it is a table with columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and a 'Pu' column. One row is selected for 'finance me-EC2' (i-0ccbf77210f0c6a58), which is running, t2.medium, and has 2/2 checks passed. At the bottom, there's a detailed view for the selected instance, showing sections for Details, Status and alarms, Monitoring, Security, Networking, Storage, and Tags. The 'Details' tab is active, showing the instance summary with fields like Instance ID, Public IPv4 address, Private IPv4 addresses, and Instance type.

```
root@ip-10-0-1-13:~# ansible --version
ansible [core 2.17.7]
  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
  executable collection location = /root/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.12.3 (main, Sep 11 2024, 14:17:37) [GCC 13.2.0] (/usr/bin/python3)
  jinja version = 3.1.2
  libyaml = True
root@ip-10-0-1-13:~#
```

i-02dba1144b3ada87b (finance me-EC2)

PublicIPs: 3.234.213.150 PrivateIPs: 10.0.1.13



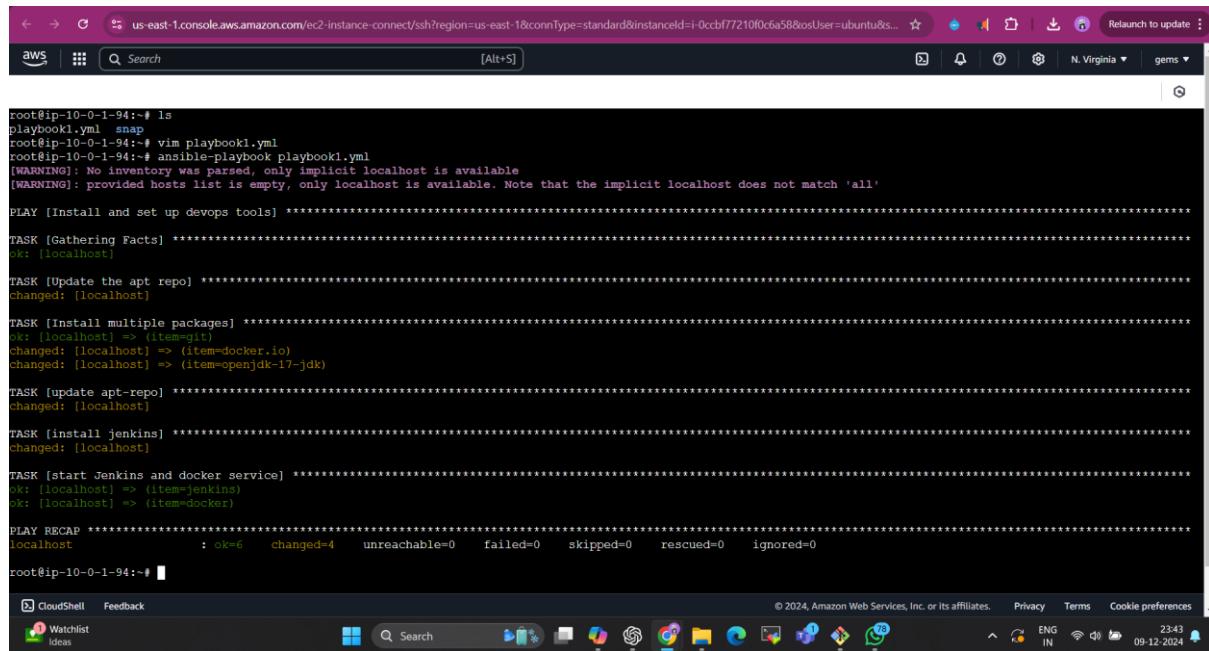
Creating an ansible playbook to setup required packages
[openjdk,Jenkins,docker] in the local hosts server → Save the file and run it.

Starting the Jenkins and docker service.

```
name: Install and set up devops tools
hosts: localhost
become: true
tasks:
- name: Update the apt repo
  command: apt-get update
- name: Install multiple packages
  package: name={{item}} state=present
  loop:
    - git
    - docker.io
    - openjdk-17-jdk
- name: update apt-repo
  command: sudo apt-get update
- name: install jenkins
  command: sudo apt-get install jenkins -y
- name: start Jenkins and docker service
  service: name={{item}} state=started
  loop:
    - jenkins
    - docker
  
```

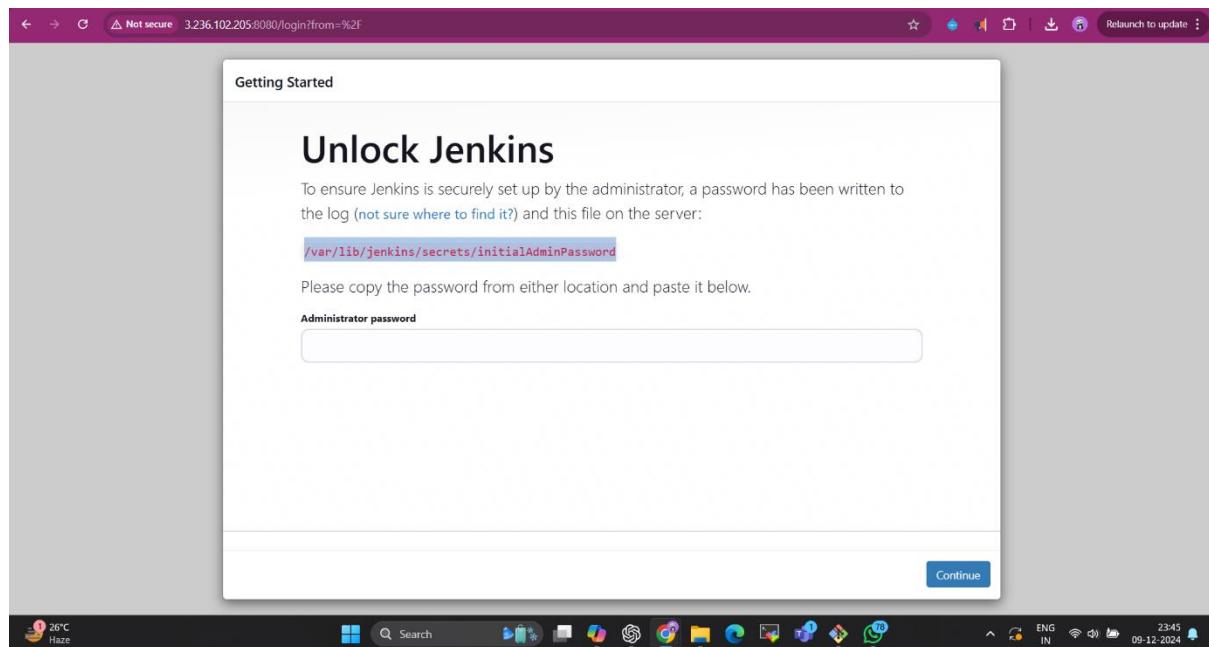
"playbook1.yml" 23L, 531B

2



```
root@ip-10-0-1-94:~# ls
playbook1.yml  snap
root@ip-10-0-1-94:~# vim playbook1.yml
root@ip-10-0-1-94:~# ansible-playbook playbook1.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'
PLAY [Install and set up devops tools] ****
TASK [Gathering Facts] ****
  ok: [localhost]
TASK [Update the apt repo] ****
  changed: [localhost]
TASK [Install multiple packages] ****
  ok: [localhost] => (item=git)
  changed: [localhost] => (item=docker.io)
  changed: [localhost] => (item=openjdk-17-jdk)
TASK [update apt-repo] ****
  changed: [localhost]
TASK [install jenkins] ****
  changed: [localhost]
TASK [start Jenkins and docker service] ****
  ok: [localhost] => (item=jenkins)
  ok: [localhost] => (item=docker)
PLAY RECAP ****
localhost          : ok=6   changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
root@ip-10-0-1-94:~#
```

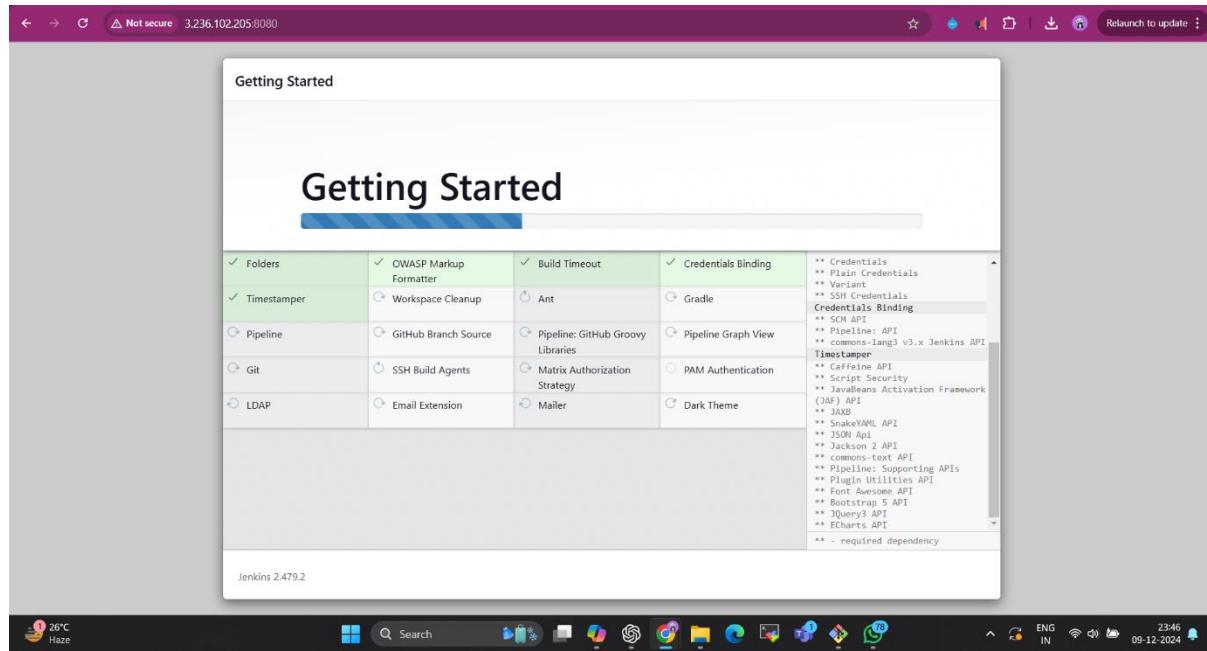
Accessing Jenkins server at the port no:8080



CICD pipeline:

Setup Jenkins dashboard and login to the Jenkins Dashboard.

Setup maven and ansible in tools section of Jenkins.

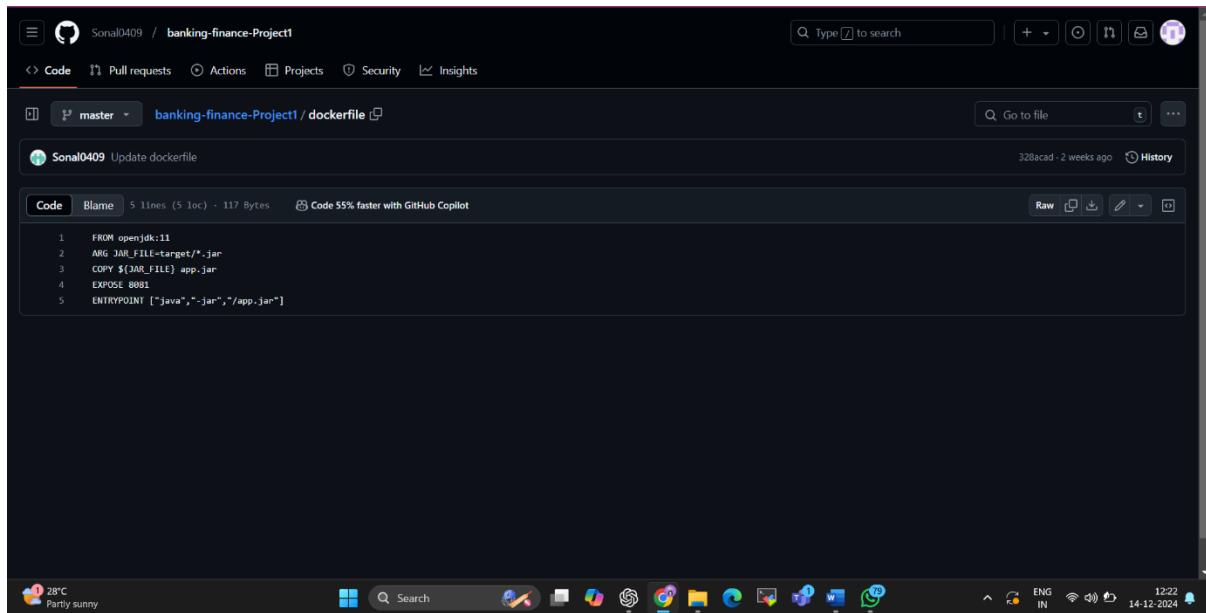


Creating a docker file to containerize the necessary packages to run the application and save it in the git hub repo.

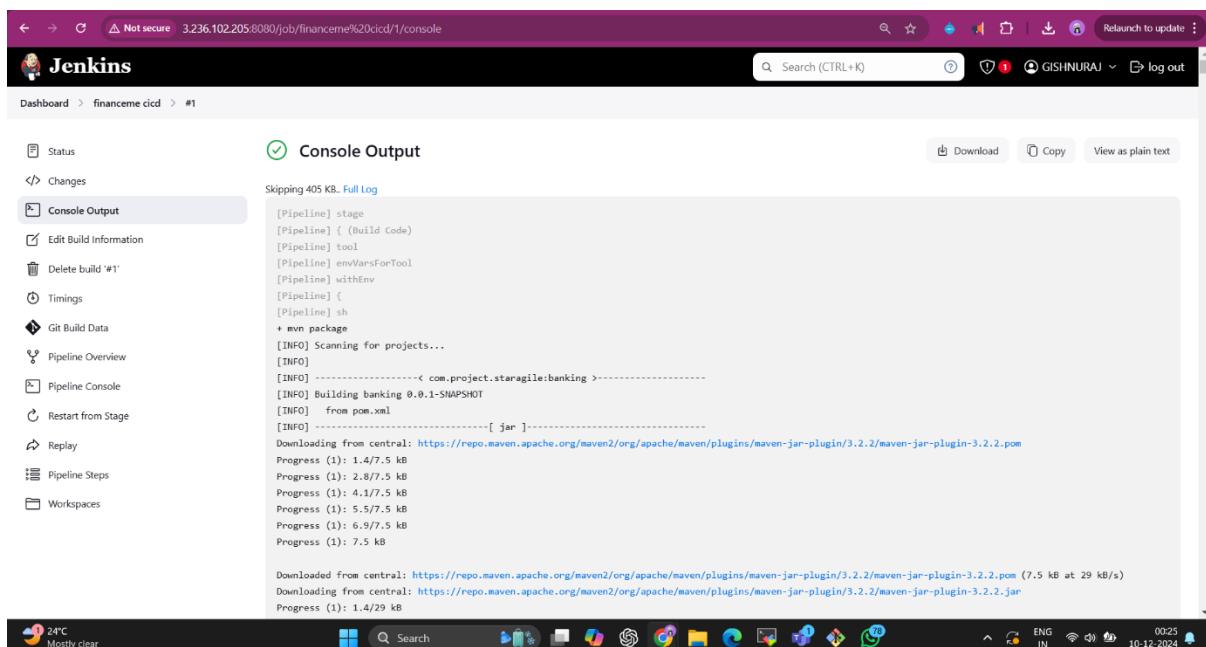
Providing permission for Jenkins to run the docker commands.

This command will allow Jenkins to run docker commands

```
chmod -R 777 /var/run/docker.sock
```

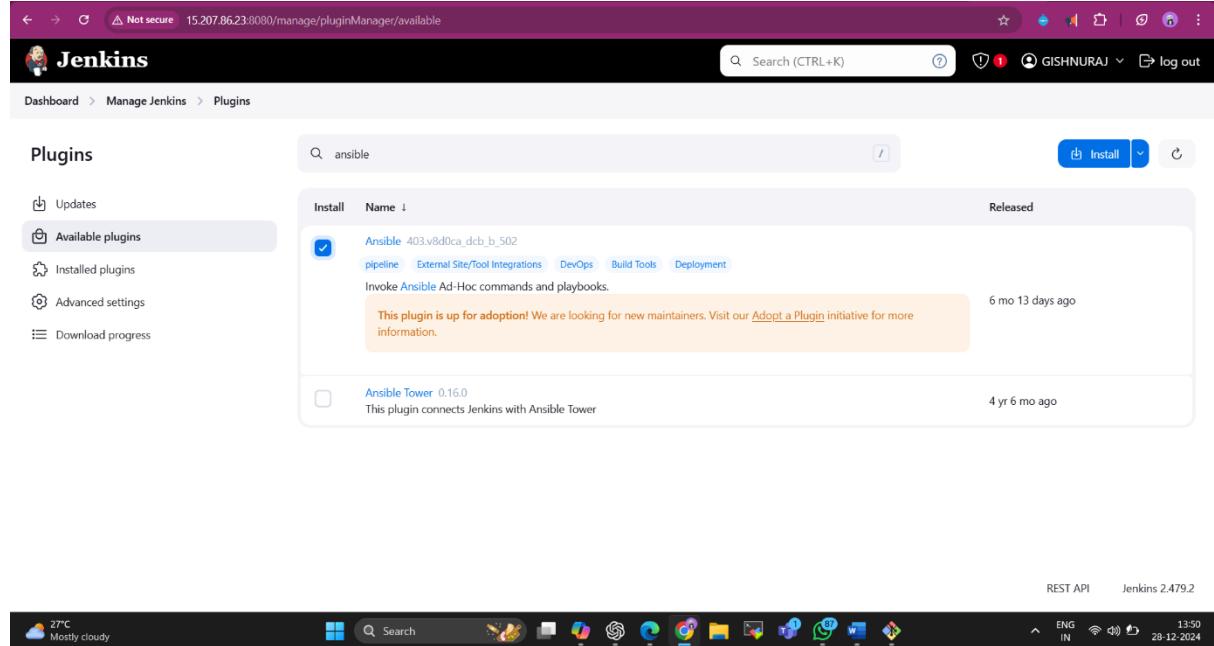


Creating a Pipeline job to
fetch code from GitHub → test and build the code using Maven commands →
build Docker image → creating an infrastructure (test server) using Terraform
→ managing configurations using Ansible playbook to deploy the containerized
application.

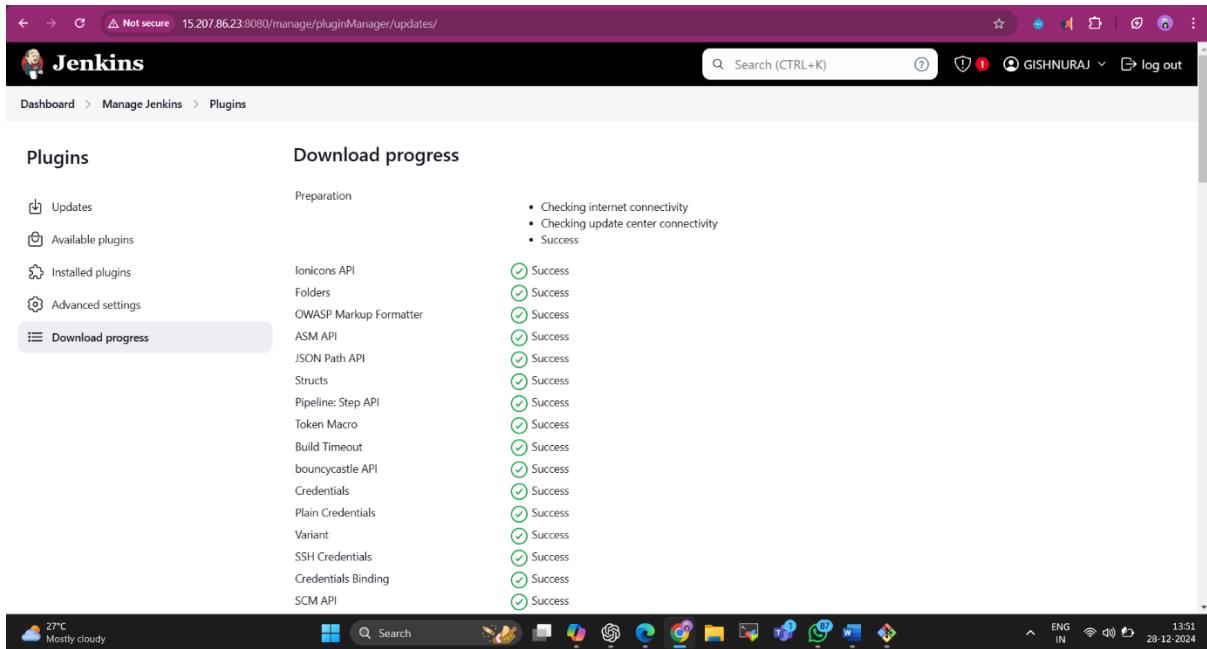


This cicd pipeline includes the various following actions:

Installing necessary plugins (ansible,terraform)



The screenshot shows the Jenkins plugin manager interface. The URL is 15.207.86.23:8080/manage/pluginManager/available. A search bar at the top contains the text "ansible". Below it, a table lists available plugins. The first row, "Ansible 403.v8d0ca_dcb_b_502", has a checked "Install" button. The second row, "Ansible Tower 0.16.0", has an unchecked "Install" button. The interface includes navigation links like "Dashboard", "Manage Jenkins", and "Plugins". On the left, there's a sidebar with "Updates", "Available plugins" (which is selected), "Installed plugins", "Advanced settings", and "Download progress". At the bottom, there are links for "REST API" and "Jenkins 2.479.2". The taskbar at the bottom of the screen shows various application icons and the date/time as 28-12-2024.



The screenshot shows the Jenkins plugin manager interface. The URL is 15.207.86.23:8080/manage/pluginManager/updates/. The "Download progress" section is highlighted in the sidebar. It displays a table of updates with their status. Most items show a green checkmark and the word "Success". The table includes rows for "Ionicons API", "Folders", "OWASP Markup Formatter", "ASM API", "JSON Path API", "Structs", "Pipeline: Step API", "Token Macro", "Build Timeout", "bouncycastle API", "Credentials", "Plain Credentials", "Variant", "SSH Credentials", "Credentials Binding", and "SCM API". The interface includes navigation links like "Dashboard", "Manage Jenkins", and "Plugins". On the left, there's a sidebar with "Updates", "Available plugins", "Installed plugins", "Advanced settings", and "Download progress" (which is selected). At the bottom, there are links for "REST API" and "Jenkins 2.479.2". The taskbar at the bottom of the screen shows various application icons and the date/time as 28-12-2024.

The screenshot shows the Jenkins 'Ansible installations' configuration page. At the top, there is a header bar with the URL 'Not secure 15.207.86.23:8080/manage/configureTools/'. Below the header, the breadcrumb navigation shows 'Dashboard > Manage Jenkins > Tools'. The main content area is titled 'Ansible installations' and contains a form for adding a new Ansible installation. The form fields are: 'Name' (set to 'ansible'), 'Path to ansible executables directory' (set to '/usr/bin/'), and a checkbox for 'Install automatically' which is unchecked. Below the form is a 'Save' button and an 'Apply' button. In the bottom right corner of the window, it says 'Jenkins 2.479.2'.

In the tools section configuring the ansible and the maven for the cicd pipeline.

The screenshot shows the Jenkins 'Available Plugins' page. At the top, there is a header bar with the URL 'Not secure 15.207.86.23:8080/manage/pluginManager/available'. Below the header, the breadcrumb navigation shows 'Dashboard > Manage Jenkins > Plugins'. The main content area is titled 'Plugins' and includes a search bar with the term 'terra'. A sidebar on the left has links for 'Updates', 'Available plugins' (which is selected), 'Installed plugins', 'Advanced settings', and 'Download progress'. The central area lists the 'Terraform' plugin, version 1.0.10, which was released 4 years and 10 months ago. The plugin description states: 'This plugin provides a build wrapper for Terraform to launch and destroy infrastructure.' There is an 'Install' button with a checkmark next to it. In the bottom right corner of the window, it says 'REST API Jenkins 2.479.2'.

Installing terraform ,git, ansible, docker in the server.

```
ubuntu@ip-172-31-0-13:~$ terraform --version
Terraform v1.10.3
on linux_amd64
ubuntu@ip-172-31-0-13:~$ |
```

The screenshot shows a terminal window on a Windows desktop. The terminal output is as follows:

```
ubuntu@ip-172-31-0-13:~$ terraform --version
Terraform v1.10.3
on linux_amd64
ubuntu@ip-172-31-0-13:~$ |
```

The desktop taskbar at the bottom shows various icons for applications like File Explorer, Edge, and FileZilla. The system tray indicates the date as 28-12-2024.

Writing a terraform configuration file to create the test server infrastructure .

In jenkins server adding the AWS credentials to make changes in the aws console.

Also adding the IAM role to provide access to the aws account.

The screenshot shows the 'Modify IAM role' page in the AWS Management Console. The URL is ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#ModifyIAMRole:instanceId=i-02fe44794617e9811. The instance ID is listed as 'I-02fe44794617e9811 (jenkins server)'. The 'IAM role' section shows a dropdown menu with the following options:

- jenkinsterra (selected)
- No IAM Role
- ec2noderole
- jenkinsterra (selected)

At the bottom right are 'Cancel' and 'Update IAM role' buttons. The status bar at the bottom of the browser window shows the date as 28-12-2024.

```
1 provider "aws" {
2   region = "ap-south-1"
3 }
4
5 resource "aws_instance" "test_server" {
6   ami           = "ami-053b12d3152c0cc71" # Ubuntu 22.04 AMI
7   instance_type = "t2.micro"
8
9   tags = {
10     Name = "Test-Server"
11   }
12
13   provisioner "local-exec" {
14     command = "echo ${aws_instance.test_server.public_ip} > test_server_ip.txt"
15   }
16 }
```

New credentials

Kind

Secret text

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Secret

.....

ID ?

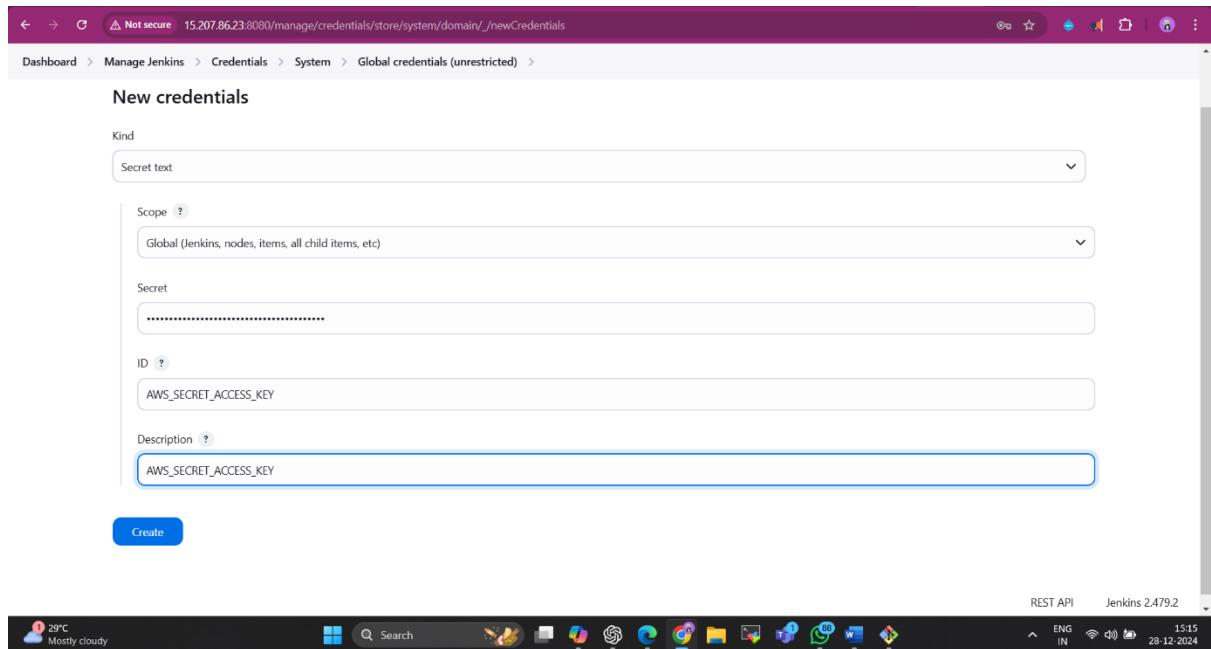
AWS_ACCESS_KEY_ID

Description ?

AWS_ACCESS_KEY_ID

Create





Writing the ansible playbook to make configuration changes in the test server (deploying an application).

Write and save the playbook in the github repository (playbook , inventory).

Also connecting to the test server and make configuration changes can be through ssh keys. These keys will be stored as a credential in the Jenkins server to make use of it.

Not secure 15.207.86.23:8080/manage/credentials/store/system/domain/ /

Jenkins

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

Search (CTRL+K)

GISHNURAJ log out

ID	Name	Kind	Description
AWS_ACCESS_KEY_ID	AWS_ACCESS_KEY_ID	Secret text	AWS_ACCESS_KEY_ID
AWS_SECRET_ACCESS_KEY	AWS_SECRET_ACCESS_KEY	Secret text	AWS_SECRET_ACCESS_KEY

Global credentials (unrestricted)

+ Add Credentials

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
AWS_ACCESS_KEY_ID	AWS_ACCESS_KEY_ID	Secret text	AWS_ACCESS_KEY_ID
AWS_SECRET_ACCESS_KEY	AWS_SECRET_ACCESS_KEY	Secret text	AWS_SECRET_ACCESS_KEY

Icon: S M L

REST API Jenkins 2.479.2



```
root@ip-172-31-0-13:~  
ubuntu@ip-172-31-0-13:~$ sudo su  
root@ip-172-31-0-13:/home/ubuntu# cd  
root@ip-172-31-0-13:# adduser ansible  
Fatal: The user 'ansible' already exists.  
root@ip-172-31-0-13:# su -ansible  
su: invalid option -- 'a'  
Try 'su --help' for more information.  
root@ip-172-31-0-13:# sudo su -ansible  
su: invalid option -- '-'  
Try 'su --help' for more information.  
root@ip-172-31-0-13:# ansible --version  
ansible 2.12.0  
  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, Nov  6 2024, 18:32:19) [GCC 13.2.0] (/usr/bin/python3)  
  jinja version = 3.1.2  
  libyaml = True  
root@ip-172-31-0-13:# ssh-keygen -t rsa -b 2048 -f ~/.ssh/jenkins_key |
```

26°C Partly cloudy

Search

ENG IN 23:02 28-12-2024

Jenkins Global credentials (unrestricted)

ID	Name	Kind	Description
AWS_ACCESS_KEY_ID	AWS_ACCESS_KEY_ID	Secret text	AWS_ACCESS_KEY_ID
AWS_SECRET_ACCESS_KEY	AWS_SECRET_ACCESS_KEY	Secret text	AWS_SECRET_ACCESS_KEY
ansible_ssh_key	ubuntu	SSH Username with private key	

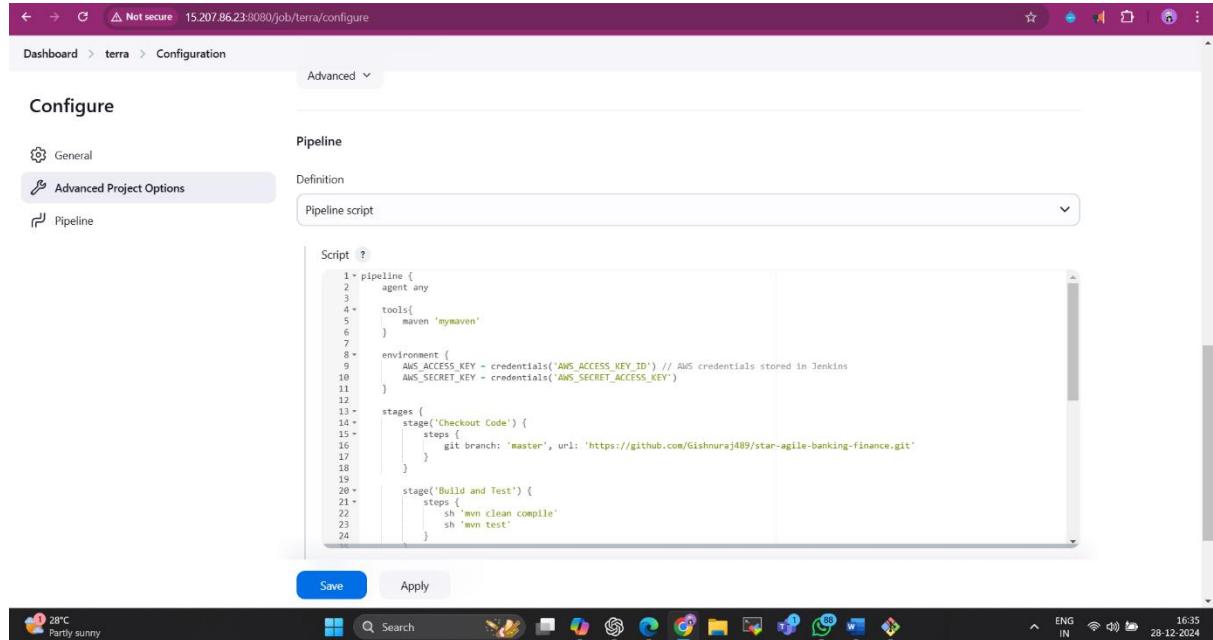
Icon: S M L REST API Jenkins 2.479.2

Code 55% faster with GitHub Copilot

```
1 - name : Configure Docker on EC2 Instances
2   hosts : all
3   become: true
4   connection : ssh
5   tasks :
6     - name: updating apt
7       command : sudo apt-get update
8
9     - name : Install Docker
10       command : sudo apt-get install -y docker.io
11
12    - name : Start Docker Service
13      command : sudo systemctl start docker
14
15    - name: Deploy Docker Container
16      command: docker run -itd -p 8084:8081 gms079903/myproject0:1
17
```

Type ⌘ to search ENG IN 28-12-2024

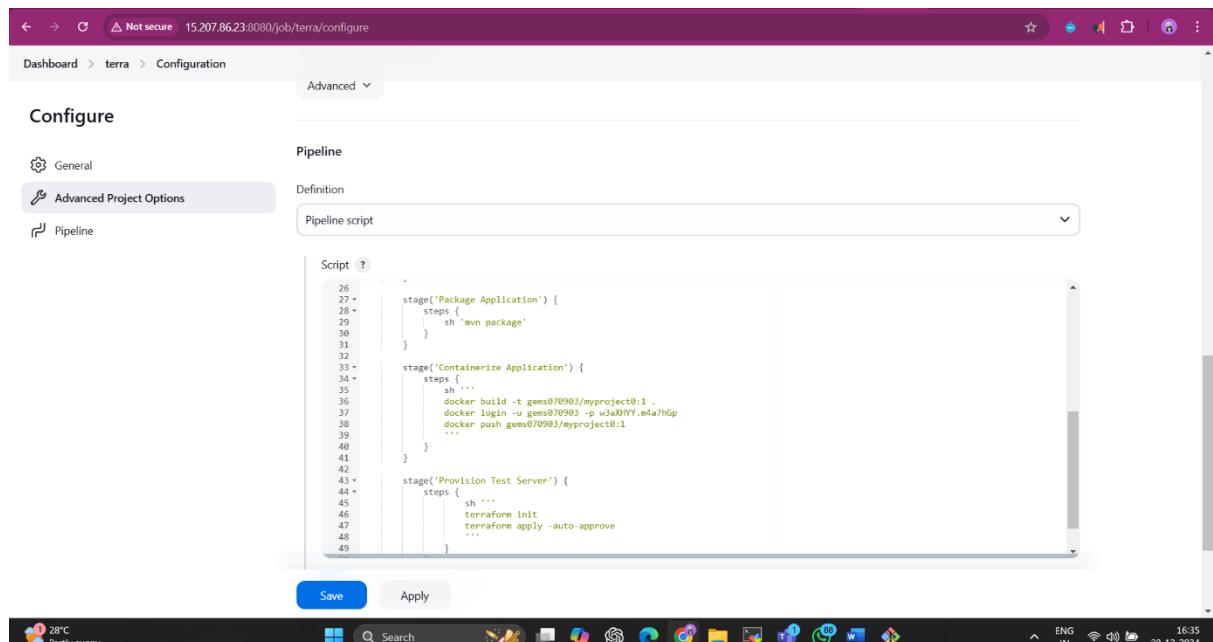
Finally, integrating terraform and ansible in the pipeline scripts.



The screenshot shows the Jenkins Pipeline configuration page for a project named 'terra'. The 'Pipeline' tab is selected under 'Definition'. The script content is as follows:

```
1+ pipeline {
2+     agent any
3+
4+     tools{
5+         maven 'mymaven'
6+
7+
8+     environment {
9+         AWS_ACCESS_KEY = credentials('AWS_ACCESS_KEY_ID') // AWS credentials stored in Jenkins
10+        AWS_SECRET_KEY = credentials('AWS_SECRET_ACCESS_KEY')
11+
12+
13+
14+     stages {
15+         stage('Checkout Code') {
16+             steps {
17+                 git branch: 'master', url: 'https://github.com/Gishnuraj489/star-agile-banking-finance.git'
18+             }
19+
20+         stage('Build and Test') {
21+             steps {
22+                 sh 'mvn clean compile'
23+                 sh 'mvn test'
24+             }
25+         }
26+
27+     }
28+
29+
30+
31+
32+
33+
34+
35+
36+
37+
38+
39+
40+
41+
42+
43+
44+
45+
46+
47+
48+
49+
```

At the bottom, there are 'Save' and 'Apply' buttons. The system tray at the bottom shows a weather icon (28°C, Partly sunny), a search bar, and various application icons.



The screenshot shows the Jenkins Pipeline configuration page for the same project 'terra'. The 'Pipeline' tab is selected under 'Definition'. The script content has been expanded to show more stages and steps:

```
1+ pipeline {
2+     agent any
3+
4+     tools{
5+         maven 'mymaven'
6+
7+
8+     environment {
9+         AWS_ACCESS_KEY = credentials('AWS_ACCESS_KEY_ID') // AWS credentials stored in Jenkins
10+        AWS_SECRET_KEY = credentials('AWS_SECRET_ACCESS_KEY')
11+
12+
13+
14+     stages {
15+         stage('Checkout Code') {
16+             steps {
17+                 git branch: 'master', url: 'https://github.com/Gishnuraj489/star-agile-banking-finance.git'
18+             }
19+
20+         stage('Build and Test') {
21+             steps {
22+                 sh 'mvn clean compile'
23+                 sh 'mvn test'
24+             }
25+         }
26+         stage('Package Application') {
27+             steps {
28+                 sh 'mvn package'
29+             }
30+         }
31+
32+
33+
34+
35+
36+
37+
38+
39+
40+
41+
42+
43+
44+
45+
46+
47+
48+
49+
```

At the bottom, there are 'Save' and 'Apply' buttons. The system tray at the bottom shows a weather icon (28°C, Partly sunny), a search bar, and various application icons.

Not secure 15.207.86.23:8080/job/financeme%20cicd/configure

Dashboard > financeme cicd > Configuration

Configure

Advanced

General

Advanced Project Options

Pipeline

Definition

Pipeline script

Script ?

```
52 ~
53 ~
54 ~
55 stage('Configure Test Server with Ansible') {
56     steps {
57         ansiblePlaybook(
58             playbook: 'ansible-playbook.yml',
59             inventory: 'inventory',
60             credentialsId: 'ansible_ssh_key'
61         )
62     }
63 }
```

try sample Pipeline...

Use Groovy Sandbox ?

Pipeline Syntax

Save Apply

REST API Jenkins 2.479.2

A screenshot of a Jenkins pipeline configuration page. The pipeline script defines a stage named 'Configure Test Server with Ansible' that runs an Ansible playbook. The script uses variables like 'ansible-playbook.yml', 'inventory', and 'credentialsId'. A 'Use Groovy Sandbox' checkbox is checked.

Not secure 3.236.102.205:8080/job/financeme%20cicd/

Jenkins

Dashboard > financeme cicd >

Status Changes Build Now Configure Delete Pipeline Stages Rename Pipeline Syntax Git Polling Log

financeme cicd

Permalinks

- Last build (#1), 1 min 45 sec ago
- Last stable build (#1), 1 min 45 sec ago
- Last successful build (#1), 1 min 45 sec ago
- Last completed build (#1), 1 min 45 sec ago

Add description

Builds

Filter /

Today #1 6:53 PM

REST API Jenkins 2.479.2

A screenshot of a Jenkins pipeline job page. It shows the build history with four entries: 'Last build (#1)', 'Last stable build (#1)', 'Last successful build (#1)', and 'Last completed build (#1)'. Navigation links for the pipeline include 'Status', 'Changes', 'Build Now', 'Configure', 'Delete Pipeline', 'Stages', 'Rename', 'Pipeline Syntax', and 'Git Polling Log'. The Jenkins logo is at the top left.

Thus the test server is created ,then using ansible → docker is installed and then the application is deployed in it.

The screenshot shows the AWS CloudShell interface. On the left, the AWS Management Console sidebar is visible with sections for Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images (AMIs, AMI Catalog), Elastic Block Store (Volumes, Snapshots, Lifecycle Manager), and Network & Security (CloudShell, Feedback). The main content area displays the 'Instances (1/2) Info' page. It lists two instances: 'jenkins server' (Instance ID: i-0fe44794617e9811, State: Running, Type: t2.medium, Status: 2/2 checks passed, Alarm status: View alarms +, Availability Zone: ap-south-1a, Public IP: ec2-15-20-11-111) and 'Test-Server' (Instance ID: i-0bd24ce2284f60a05, State: Running, Type: t2.micro, Status: Initializing, Alarm status: View alarms +, Availability Zone: ap-south-1a, Public IP: ec2-13-23-11-111). Below this, a detailed view for 'i-0bd24ce2284f60a05 (Test-Server)' is shown with tabs for Details, Status and alarms, Monitoring, Security, Networking, Storage, and Tags. The Details tab shows the instance ID, Public IP (13.232.115.148), Private IP (172.31.0.21), and a screenshot of the Windows desktop environment with various icons. The status bar at the bottom indicates the date (28-12-2024) and time (19:16).

The screenshot shows a terminal session within the AWS CloudShell. The command history shows:

```
root@ip-172-31-0-21:~# docker --version
Docker version 26.1.3, build 26.1.3-0ubuntu1~24.04.1
root@ip-172-31-0-21:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
e2ab0f847c68        gems070903/myproject:0.1   "java -jar /app.jar"   37 seconds ago    Up 36 seconds   0.0.0.0:8084->8081/tcp, :::8084->8081/tcp   beautiful_vohard
root@ip-172-31-0-21:~#
```

Below the terminal, the instance summary shows:

i-0bd24ce2284f60a05 (Test-Server)

Public IPs: 13.232.115.148 Private IPs: 172.31.0.21

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences ENG IN 19:30 28-12-2024

← → ⌛ Not secure 13.232.115.148:8084 ⋮

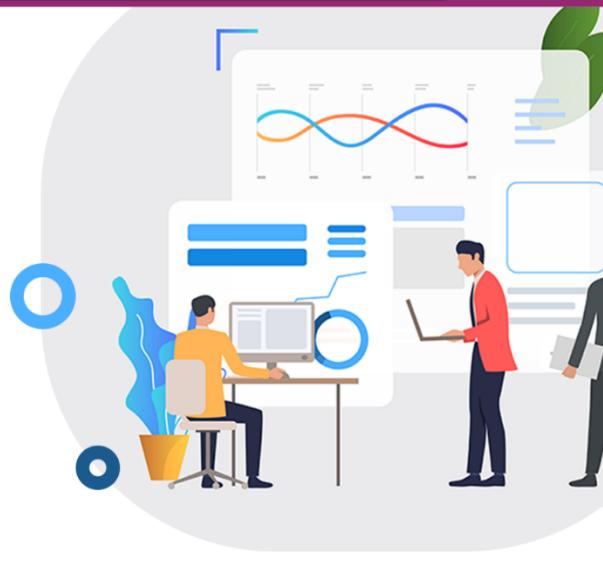
 Cial

HOME ABOUT SERVICES TEAM CONTACT US

CUSTOMER BANKING SERVICES

We provide the World's best in class Banking Solutions and Services.

[READ MORE](#) [CONTACT US](#)



28°C Haze

Search

ENG IN 19:31 28 12 2024

← → ⌛ Not secure 13.232.115.148:8084 ⋮

OUR TEAM AND EXPERTS

It is a long established fact that a reader will be distracted by the readable content of a



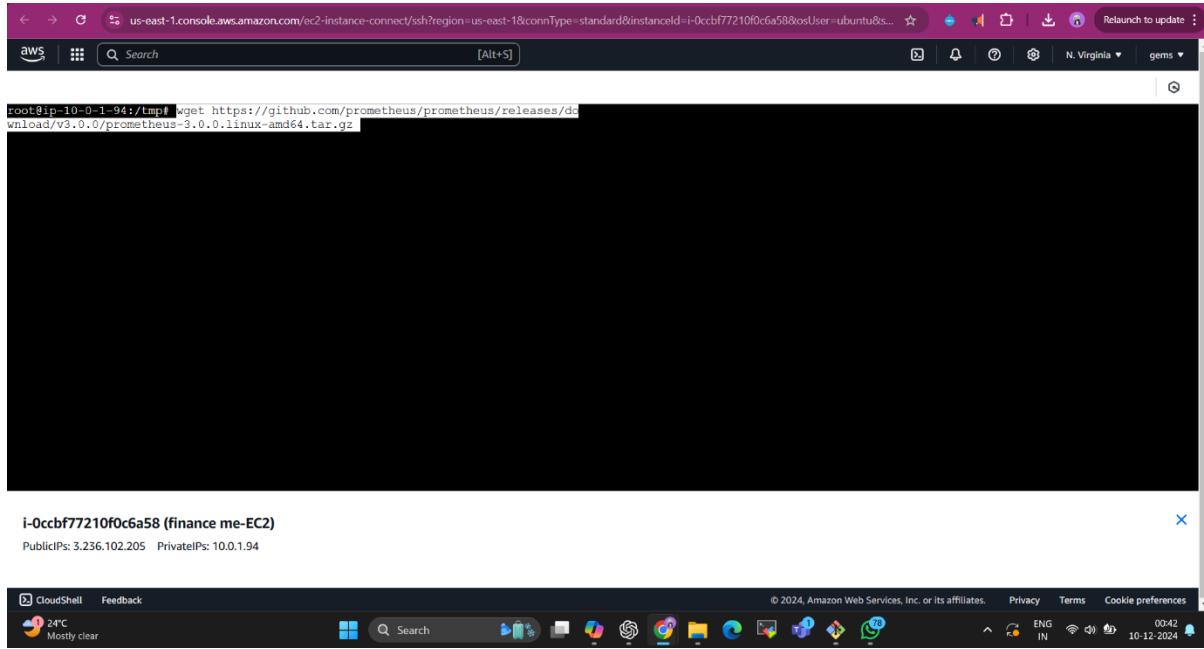
Readable	Content	Readable	Content
Follow Us	Follow Us	Follow Us	Follow Us
  	  	  	  

28°C Haze

Search

ENG IN 19:31 28 12 2024

Then, install the Prometheus metrics plugin for the continuous monitoring of the Jenkins job. → then restart the Jenkins server.



```
root@ip-10-0-1-94:/tmp# wget https://github.com/prometheus/prometheus/releases/download/v3.0.0/prometheus-3.0.0.linux-amd64.tar.gz
```

i-0ccbf77210f0c6a58 (finance me-EC2)
PublicIPs: 3.236.102.205 PrivateIPs: 10.0.1.94

Monitoring of this Jenkins job using Prometheus and Grafana:

Installing and configuring Prometheus and Grafana as monitoring and visualization tools.

STEPS:

Create a system user for Prometheus using below commands → Create the directories in which we will be storing our configuration files and libraries

Move the configuration file and set the owner to the Prometheus → Move the binaries and set the owner → Creating the Prometheus service file in the directory name: **/etc/systemd/system/prometheus.service**

```
root@monitoring:/tmp/prometheus-3.0.0.linux-amd64
GNU nano 7.2                               /etc/systemd/system/prometheus.service *

[Unit]
Description=Prometheus
Wants=network-online.target
After=network-online.target

[Service]
User=prometheus
Group=prometheus
Type=simple
ExecStart=/usr/local/bin/prometheus \
--config.file /etc/prometheus/prometheus.yml \
--storage.tsdb.path /var/lib/prometheus/ \
--web.console.templates=/etc/prometheus/consoles \
--web.console.libraries=/etc/prometheus/console_libraries

[Install]
WantedBy=multi-user.target

File Name to Write: /etc/systemd/system/prometheus.service
^G Help      M-D DOS Format      M-A Append      M-B Backup File
^C Cancel    M-M Mac Format     M-P Prepend     ^T Browse
```

```

root@monitoring:/tmp/prometheus-3.0.0.linux-amd64
● prometheus.service - Prometheus
   Loaded: loaded (/etc/systemd/system/prometheus.service; enabled; preset: enabled)
     Active: active (running) since Sun 2024-12-08 06:46:04 UTC; 22s ago
       Main PID: 1335 (prometheus)
          Tasks: 7 (limit: 4676)
        Memory: 18.9M (peak: 19.0M)
         CPU: 94ms
        CGroup: /system.slice/prometheus.service
                  └─1335 /usr/local/bin/prometheus --config.file /etc/prometheus/prometheus.yml --storage.tsdb.path >

Dec 08 06:46:04 monitoring prometheus[1335]: time=2024-12-08T06:46:04.232Z level=INFO source=head.go:723 msg="R>
Dec 08 06:46:04 monitoring prometheus[1335]: time=2024-12-08T06:46:04.233Z level=INFO source=head.go:795 msg="W>
Dec 08 06:46:04 monitoring prometheus[1335]: time=2024-12-08T06:46:04.233Z level=INFO source=head.go:832 msg="W>
Dec 08 06:46:04 monitoring prometheus[1335]: time=2024-12-08T06:46:04.236Z level=INFO source=main.go:1260 msg=>>
Dec 08 06:46:04 monitoring prometheus[1335]: time=2024-12-08T06:46:04.236Z level=INFO source=main.go:1263 msg=>>
Dec 08 06:46:04 monitoring prometheus[1335]: time=2024-12-08T06:46:04.236Z level=INFO source=main.go:1446 msg=>>
Dec 08 06:46:04 monitoring prometheus[1335]: time=2024-12-08T06:46:04.240Z level=INFO source=main.go:1485 msg=>>
Dec 08 06:46:04 monitoring prometheus[1335]: time=2024-12-08T06:46:04.240Z level=INFO source=main.go:1495 msg=>>
Dec 08 06:46:04 monitoring prometheus[1335]: time=2024-12-08T06:46:04.240Z level=INFO source=main.go:1224 msg=>>
Dec 08 06:46:04 monitoring prometheus[1335]: time=2024-12-08T06:46:04.240Z level=INFO source=manager.go:168 msg=>>

root@monitoring:/tmp/prometheus-3.0.0.linux-amd64#
root@monitoring:/tmp/prometheus-3.0.0.linux-amd64# wget https://github.com/prometheus/node_exporter/releases/download/v1.8.2/node_exporter-1.8.2.linux-amd64.tar.gz
--2024-12-08 06:47:54-- https://github.com/prometheus/node_exporter/releases/download/v1.8.2/node_exporter-1.8.2.linux-amd64.tar.gz
Resolving github.com (github.com)... 20.207.73.82
Connecting to github.com (github.com)|20.207.73.82|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/9524057/a7e04f41-5543-40e2-9060-26fefef32bb4b?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=releaseassetproduction%2F20241208%2Fus-east-1%2Faws4_request&X-Amz-Date=20241208T064754Z&X-Amz-Expires=300&X-Amz-Signature=0835713b7862698eb39181667f3825de7f7661891c81a9f15dd11d27e56205b&X-Amz-SignedHeaders=host&response-content-disposition=attachment%3B%20filename%3Dnode_exporter-1.8.2.linux-amd64.tar.gz&response-content-type=application%2Foctet-stream [following]
--2024-12-08 06:47:54-- https://objects.githubusercontent.com/github-production-release-asset-2e65be/9524057/a7e04f41-5543-40e2-9060-26fefef32bb4b?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=releaseassetproduction%2F20241208%2Fus-east-1%2Faws4_request&X-Amz-Date=20241208T064754Z&X-Amz-Expires=300&X-Amz-Signature=0835713b7862698eb39181667f3825de7f7661891c81a9f15dd11d27e56205b&X-Amz-SignedHeaders=host&response-content-disposition=attachment%3B%20filename%3Dnode_exporter-1.8.2.linux-amd64.tar.gz&response-content-type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.109.133, 185.199.110.133, 185.199.111.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.109.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10676343 (10M) [application/octet-stream]
Saving to: 'node_exporter-1.8.2.linux-amd64.tar.gz'

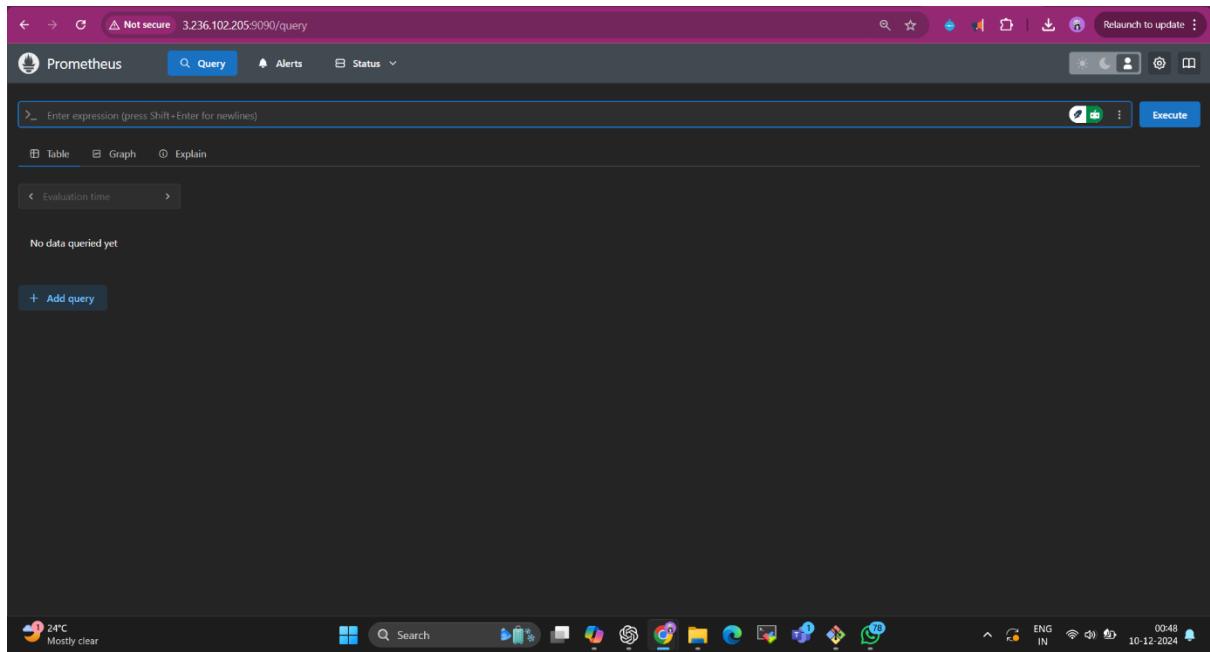
node_exporter-1.8.2.linux-a 100%[=====] 10.18M 18.2MB/s in 0.6s

2024-12-08 06:47:56 (18.2 MB/s) - 'node_exporter-1.8.2.linux-amd64.tar.gz' saved [10676343/10676343]

root@monitoring:/tmp/prometheus-3.0.0.linux-amd64# sudo tar xvfz node_exporter-*.*-amd64.tar.gz
node_exporter-1.8.2.linux-amd64/
node_exporter-1.8.2.linux-amd64/NOTICE
node_exporter-1.8.2.linux-amd64/node_exporter
node_exporter-1.8.2.linux-amd64/LICENSE
root@monitoring:/tmp/prometheus-3.0.0.linux-amd64# sudo mv node_exporter-*.*-amd64/node_exporter /usr/local/bin/
root@monitoring:/tmp/prometheus-3.0.0.linux-amd64# sudo useradd -rs /bin/false node_exporter

```

Reload systemd → Start and enable Prometheus service → Now accessing Prometheus in your browser in the PORT NO:9090.



Installing Node Exporter on Ubuntu 22.04 LTS → Move the binary file of node exporter to /usr/local/bin location → Create a node_exporter user to run the node exporter service → Create a Custom Node Exporter Service in the name: **/etc/systemd/system/node_exporter.service.**

```

root@ip-10-0-1-94:/tmp/prometheus-3.0.0.linux-amd64# sudo nano /etc/systemd/system/node_exporter.service
root@ip-10-0-1-94:/tmp/prometheus-3.0.0.linux-amd64# sudo cat /etc/systemd/system/node_exporter.service
[Unit]
Description=Node Exporter
After=network.target

[Service]
User=node_exporter
Group=node_exporter
Type=simple
ExecStart=/usr/local/bin/node_exporter

[Install]
WantedBy=multi-user.target
root@ip-10-0-1-94:/tmp/prometheus-3.0.0.linux-amd64#

```

i-0ccbf77210fc6a58 (finance me-EC2)
PublicIPs: 3.236.102.205 PrivateIPs: 10.0.1.94

Reload the system → Start, enable node exporter → update our with jobs [node exporter, jenkin jobs] configuration file:
/etc/prometheus/prometheus.yml

```

# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=` to any timeseries scraped from these targets.
  - job_name: "prometheus"
    #告警
    - __meta_prometheus_sd_file: "file:///etc/prometheus/prometheus_sd.conf"
    - __meta_prometheus_sd_replace: true
    - __meta_prometheus_sd_exit_mark: true
    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.
    static_configs:
      - targets: ["localhost:9090"]
    job_name: "Node_Exporter"
    scrape_interval: 5s
    static_configs:
      - targets: ["3.236.102.205:9100"]
    job_name: "jenkins"
    metrics_path: "/prometheus"
    static_configs:
      - targets: ["3.236.102.205:8080"]

```

i-0ccbf77210fc6a58 (finance me-EC2)
PublicIPs: 3.236.102.205 PrivateIPs: 10.0.1.94

installing the Prometheus metric plugin so that the jenkin jobs can be monitored continuously.

The screenshot shows the Jenkins plugin manager interface. The search bar at the top contains the text "prome". Below the search bar, there is a list of plugins:

- Artifactory** 4.0.8 (Released 5 mo 1 day ago): A plugin for Jenkins that allows build jobs to deploy artifacts and resolve dependencies from Artifactory.
- Prometheus metrics** 801.v98e119dBeeda (Released 3 days 12 hr ago): A plugin for Jenkins that exposes an endpoint for Prometheus monitoring.
- Build With Parameters** 76.v982db_f78962 (Released 2 yr 0 mo ago): A plugin that allows users to provide parameters for a build in the URL.
- promoted builds** 965.vcdla_cba_e099bf (Released 3 mo 13 days ago): A plugin that implements a "promoted build" feature.
- artifact-promotion** 0.5.2 (Released 2 yr 0 mo ago): A plugin for Jenkins that integrates with Artifactory.

[Note: After changes any configuration file we need to restart our Prometheus]

The screenshot shows the Prometheus query results for the metric `jenkins_executor_count_history`. The results are displayed in a table format:

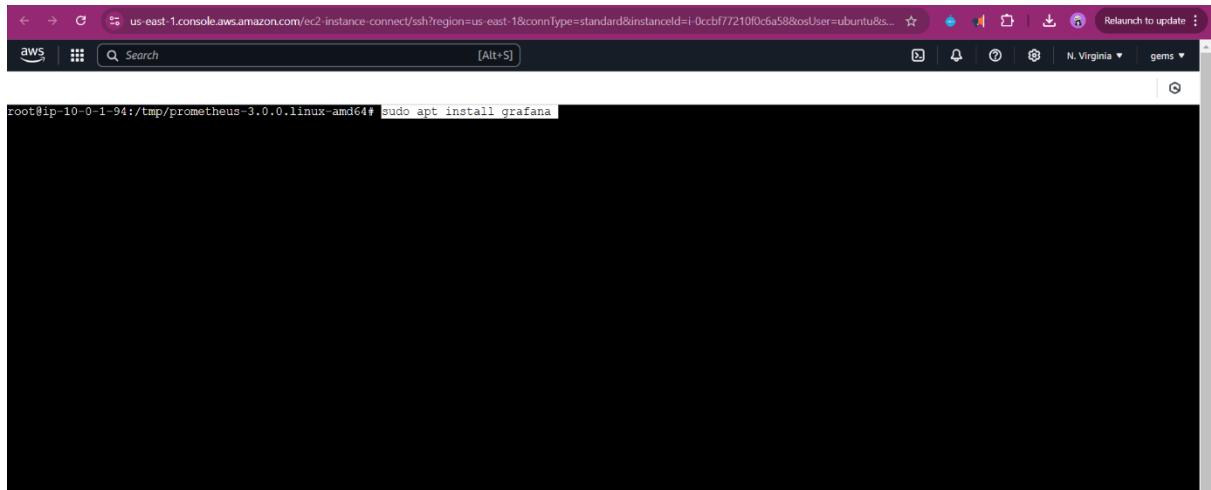
Series	Value
jenkins_executor_count_history{instance="3.236.102.205:8080",job="jenkins",quantile="0.51"}	2
jenkins_executor_count_history{instance="3.236.102.205:8080",job="jenkins",quantile="0.75"}	2
jenkins_executor_count_history{instance="3.236.102.205:8080",job="jenkins",quantile="0.95"}	2
jenkins_executor_count_history{instance="3.236.102.205:8080",job="jenkins",quantile="0.99"}	2
jenkins_executor_count_history{instance="3.236.102.205:8080",job="jenkins",quantile="0.999"}	2

The screenshot shows the Prometheus web interface at the URL `3.236.102.205:9090/targets`. The page displays three targets: Node_Exporter, Jenkins, and prometheus. Each target has its endpoint, labels, last scrape time, and state (all are UP). The interface includes navigation tabs for Query, Alerts, and Status > Target health, along with various search and filter options.

Target	Endpoint	Labels	Last scrape	State
Node_Exporter	http://3.236.102.205:9100/metrics	instance="3.236.102.205:9100" job="Node_Exporter"	4.822s ago	11ms UP
Jenkins	http://3.236.102.205:8080/prometheus	instance="3.236.102.205:8080" job="jenkins"	4.871s ago	9ms UP
prometheus	http://localhost:9090/metrics	instance="localhost:9090" job="prometheus"	314ms ago	4ms UP

Installing Grafana on Ubuntu 22.04 LTS with necessary packages.

Once Grafana is installed, use `systemctl` command to start the Grafana server → we can access Grafana server in the port no:3000.



root@ip-10-0-1-94:/tmp/prometheus-3.0.0.linux-amd64# sudo apt install grafana

i-0ccbf77210f0c6a58 (finance me-EC2)
Public IPs: 3.236.102.205 Private IPs: 10.0.1.94

CloudShell Feedback 24C Mostly clear © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences ENG IN 01:07 10-12-2024

Clicking Add data source and select Prometheus → For the URL, enter <http://3.236.102.205:9090> and click Save and test .

Finally, adding Dashboard for a better view in Grafana.

Click On Dashboard → Import Dashboard then paste the code 9964 and click on load to see the Jenkins and its jobs performance metrices.

Not secure 3.236.102.205:3000/dashboard/import

Home > Dashboards > Import dashboard

Import dashboard

Import dashboard from file or Grafana.com

Upload dashboard JSON file
Drag and drop here or click to browse
Accepted file types: json, txt

Find and import dashboards for common applications at grafana.com/dashboards

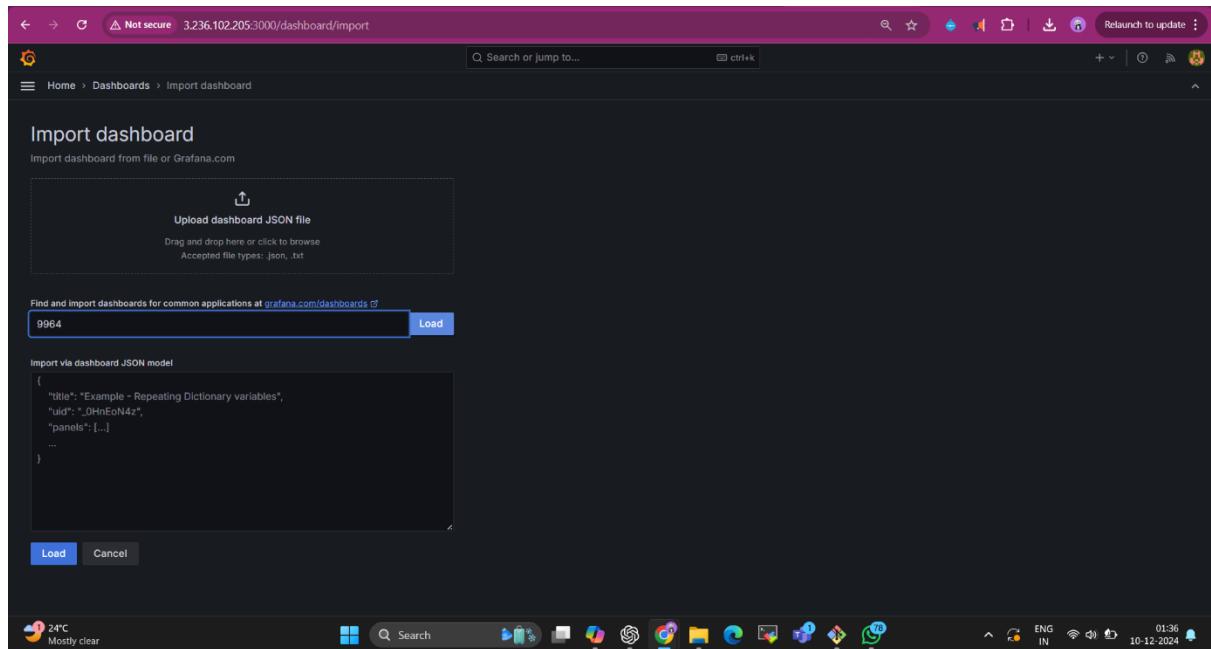
9964 Load

Import via dashboard JSON model

```
{
  "title": "Example - Repeating Dictionary variables",
  "uid": "_OHnEoN4z",
  "panels": [...]
}
```

Load Cancel

24°C Mostly clear Search ENG IN 01:36 10-12-2024



Not secure 3.236.102.205:3000/dashboard/import

Home > Dashboards > Import dashboard

Import dashboard

Import dashboard from file or Grafana.com

Importing dashboard from [Grafana.com](#)

Published by haryan
Updated on 2023-08-24 15:04:53

Options

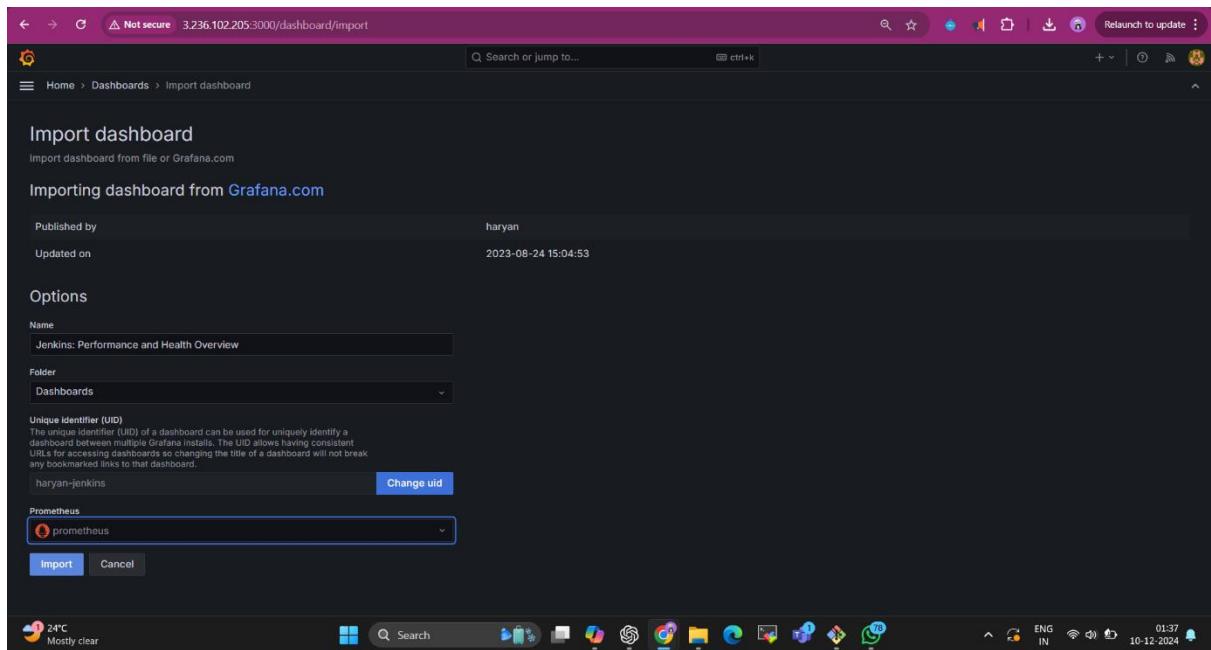
Name Jenkins: Performance and Health Overview
Folder Dashboards

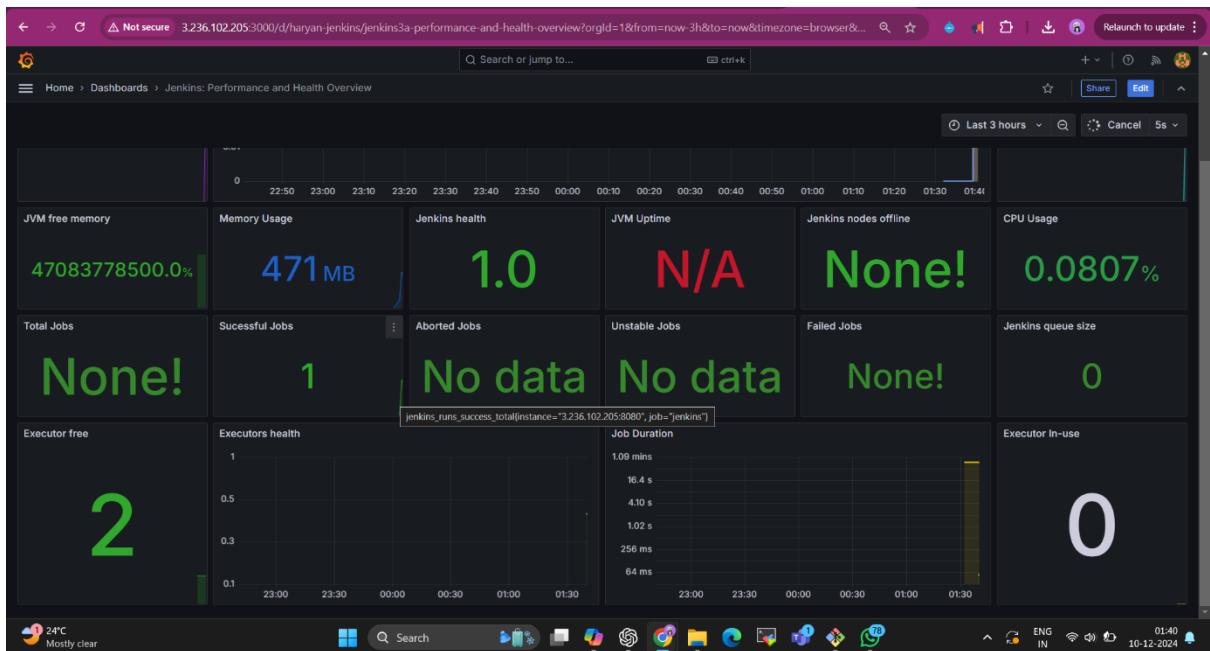
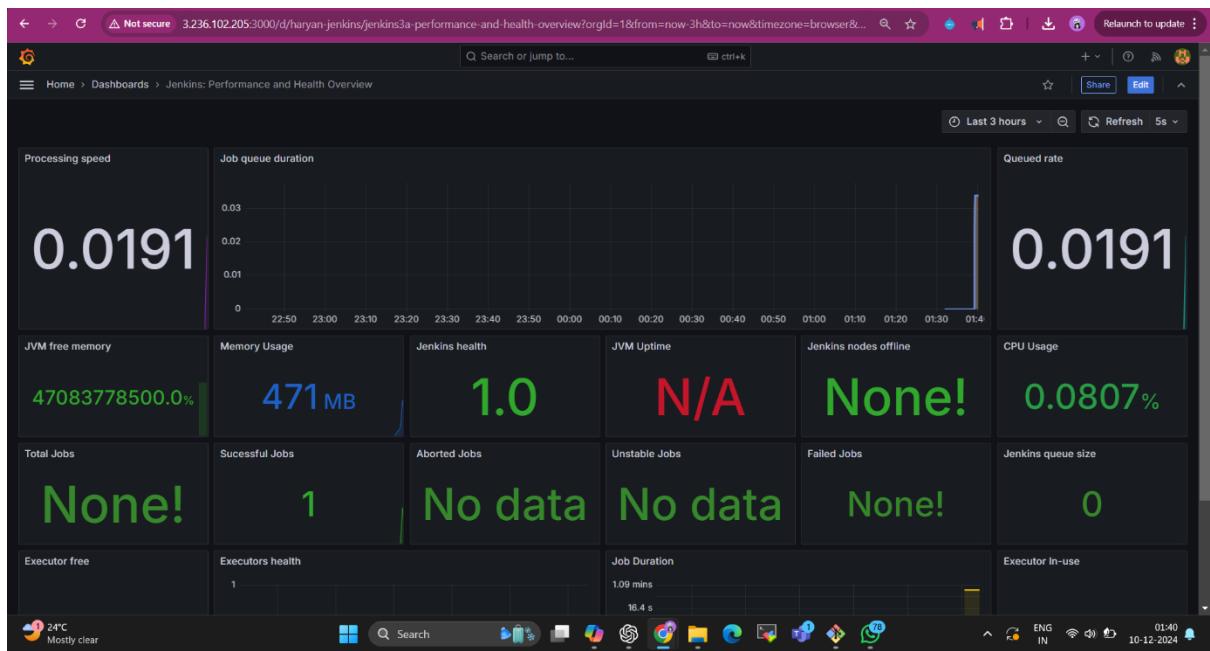
Unique Identifier (UID)
The unique identifier (UID) of a dashboard can be used for uniquely identify a dashboard between multiple Grafana installs. The UID allows having consistent URLs for accessing dashboards so changing the title of a dashboard will not break any bookmarked links to that dashboard.
haryan-jenkins Change uid

Prometheus
prometheus

Import Cancel

24°C Mostly clear Search ENG IN 01:37 10-12-2024





Not secure 3.236.102.205:3000/dashboards

Search or jump to... ⌘ ctrl+k

Home > Dashboards

Dashboards

Create and manage dashboards to visualize your data

Search for dashboards and folders

Filter by tag Starred

Name Tags

Name	Tags
Jenkins: Performance and Health Overview	Jenkins, Performance, Health
Node Exporter Full	linux

24°C Mostly clear

Search

ENG IN

10-12-2024 01:40

CONCLUSION:

As we built the CICD pipeline, if there are any changes committed in the Git hub repo it immediately triggers the pipeline, then the updated code (application) gets deployed through docker containers. And, we can monitor the Jenkins job (pipeline) continuously through Prometheus & visualize the data in Grafana.

Whenever there is a commit in the repo, it automatically triggers the pipeline(#2build).

In the previous Jenkins dash board you can see the successful job count is "1".

Thus, the continuous monitoring tool updated the successful job count as "2".

