Module Name : Capstone Project

Project name : Banking and Finance Domain

Submitted by : Rahul R

On : 26-05-2025

Business challenge/requirement

As soon as the developer pushes the updated code on the GIT master branch, the code should be checked out, compiled, tested, packaged and containerized. A new test-server should be provisioned using terraform and should be automatically configured using Ansible with all the required software's and as soon as the server is available, the application must be deployed to the test-server automatically.

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

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

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

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

Tech stack Required to use:

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

✓ Maven – For Continuous Build

✓ Jenkins - For continuous integration and continuous deployment

✓ Docker - For deploying containerized applications

✓ Ansible - Configuration management tools

✓ Selenium - For automating tests on the deployed web application

✓ Terraform - For creation of infrastructure.

✓ Prometheus and Grafana – For Automated Monitoring and Report Visualization

Steps for project :

1. Launch 4 Instance with terraform

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

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

Authenticate..tf

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

Variable .tf

variable.tf
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-0053980059cdab306"
}

Main .tf

main.tf
resource "aws_instance" "banking" {
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 = 28
volume_type = "gp3"
delete_on_termination = true
}
tags = {
Name = [
"Jenkins-host",
"prod-1",
"test-1",
"monitoring"
][count.index]
}
}

Output.tf

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

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

```
Microsoft Windows [Version 10.0.26100.4061]
(c) Microsoft Corporation. All rights reserved.

D:\course\banking-script>terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.98.0

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.

D:\course\banking-script>
```

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

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

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

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

Ansible Setup and configuration with other instances

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

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

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

"# from PubkeyAuthentication "

In the same file do down and remove " # from PasssswordAuthentication "

And the save and exit the file

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

Do all the file changes in all the other nodes

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

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

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

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

```
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-172-31-0-90:/home/ubuntu# vi /etc/ssh/sshd_config
root@ip-172-31-0-90:/home/ubuntu# ^C
root@ip-172-31-0-90:/home/ubuntu# vi /etc/ssh/sshd_config.d/60-cloudimg-settings.conf
root@ip-172-31-0-90:/home/ubuntu# ^C
root@ip-172-31-0-90:/home/ubuntu# service ssh restart
root@ip-172-31-0-90:/home/ubuntu# visudo
root@ip-172-31-0-90:/home/ubuntu# su - devops
devops@ip-172-31-0-90:~$ 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:MftjsAxbTsu2rYv530Pux2wtUPS43tJyiiXYPtFCAKM devops@ip-172-31-0-90
The key's randomart image is:
+--[ED25519 256]--+
|        o.       |
|       . ..   .  |
|      E o  . . o |
|         +  . o .|
|      . S  . o . |
|       O = o= o  |
|      . B =ooB.+ |
|       + + +++X =|
|      o.=+o.==.* |
+----[SHA256]-----+
devops@ip-172-31-0-90:~$
```

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

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

Do this for other instance too using the private ip

```
devops@ip-172-31-0-90:~$ ssh-copy-id devops@172.31.14.153
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/devops/.ssh/id_ed25519.pub"
The authenticity of host '172.31.14.153 (172.31.14.153)' can't be established.
ED25519 key fingerprint is SHA256:85UWpHtDHXpHSXayT9ORmmKpmQqcTChqyLeoLVSWRCk.
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 the new keys
devops@172.31.14.153's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'devops@172.31.14.153'"
and check to make sure that only the key(s) you wanted were added.

devops@ip-172-31-0-90:~$ ssh-copy-id devops@172.31.1.196
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/devops/.ssh/id_ed25519.pub"
The authenticity of host '172.31.1.196 (172.31.1.196)' can't be established.
ED25519 key fingerprint is SHA256:bVe7mm1jR0BHnf0QP/s7ayC5lnynYkCOduSWOy8wYKQ.
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 the new keys
devops@172.31.1.196's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'devops@172.31.1.196'"
and check to make sure that only the key(s) you wanted were added.

devops@ip-172-31-0-90:~$
```

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

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

Then set user required name and password

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

Also install ansible plugin

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

And then restart the Jenkins service

Configure the ansible path in the jenkins

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

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

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

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

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

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

Add the private ip of the production server

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

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

Add the triggers github poll scm and poll scm

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

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

Jenkins pipeline script

```
pipeline {
  agent any
  stages {
    stage('Checkout Code') {
      steps {
        git url: 'https://github.com/rahul-reddy16/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 reddyrahul/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 reddyrahul/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: ''
            }
        }
```

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

Check the pipeline build status it gets executed

Now after clicking the link ur able to access the appilications on the machine

Install Prometheus and Grafana in monitoring instance "check systemctl status grafana and Prometheus
Install node exporter in all the nodes :test 1, prod1

Prometheus and Grafana setup Now in master machine
Step 1: Update your system packages
sudo apt-get update && sudo apt-get upgrade -y
sudo yum update -y

Step 2: Create a dedicated user for Node Exporter
sudo useradd --no-create-home --shell /bin/false node_exporter

Step 3: Download the latest Node Exporter binary
Change to a temporary directory and download Node Exporter (replace version with latest if needed):
cd /usr/src
sudo wget https://github.com/prometheus/node_exporter/releases/download/v1.8.0/node_exporter-1.8.0.linux-amd64.tar.gz

Step 4: Extract the downloaded archive
sudo tar -xvf node_exporter-1.8.0.linux-amd64.tar.gz

Step 5: Move the Node Exporter binary to /usr/local/bin and set ownership
sudo mv node_exporter-1.8.0.linux-amd64/node_exporter /usr/local/bin/
sudo chown node_exporter:node_exporter /usr/local/bin/node_exporter

Step 6: Create a systemd service file for Node Exporter
Open the service file for editing:
sudo nano /etc/systemd/system/node_exporter.service
Paste the following content:
[Unit]
Description=Prometheus Node Exporter
Wants=network-online.target
After=network-online.target

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

Restart=on-failure

[Install]
WantedBy=multi-user.target

Save and exit (Ctrl+O, Enter, Ctrl+X).

Step 7: Reload systemd, enable and start Node Exporter service
sudo systemctl daemon-reload
sudo systemctl enable node_exporter
sudo systemctl start node_exporter

Step 8: Verify Node Exporter is running
sudo systemctl status node_exporter
You should see the service active and running

Step 9: Confirm Node Exporter metrics are accessible
Open a browser or use curl to check metrics endpoint:
curl http://localhost:9100/metrics

You should see a list of metrics output, confirming Node Exporter is working correctly.

This procedure installs Node Exporter as a system service running under a dedicated user, ensuring proper permissions and automatic startup on boot. The default port is 9100.

Install and Configure Prometheus

Step 1: Create Prometheus user and directories

sudo useradd --no-create-home --shell /bin/false prometheus
sudo mkdir /etc/prometheus
sudo mkdir /var/lib/prometheus
sudo chown prometheus:prometheus /etc/prometheus
sudo chown prometheus:prometheus /var/lib/Prometheus

Step 2: Download and extract Prometheus

cd /usr/local/src
sudo wget https://github.com/prometheus/prometheus/releases/download/v2.45.0/prometheus-2.45.0.linux-amd64.tar.gz
sudo tar xvf prometheus-2.45.0.linux-amd64.tar.gz
sudo mv prometheus-2.45.0.linux-amd64 prometheus-files

Step 3: Move binaries and set ownership

sudo cp prometheus-files/prometheus /usr/local/bin/
sudo cp prometheus-files/promtool /usr/local/bin/
sudo chown prometheus:prometheus /usr/local/bin/prometheus
sudo chown prometheus:prometheus /usr/local/bin/promtool

Step 4: Copy configuration files and set ownership

sudo cp -r prometheus-files/consoles /etc/prometheus
sudo cp -r prometheus-files/console_libraries /etc/prometheus
sudo cp prometheus-files/prometheus.yml /etc/prometheus/
sudo chown -R prometheus:prometheus /etc/Prometheus

Step 5: Edit Prometheus configuration to monitor Node Exporter

Edit /etc/prometheus/prometheus.yml and add the following scrape config under scrape_configs:
scrape_configs
In node exporter configure Prometheus .yaml file add :

```
 job_name: "node_exporter"
 static_configs:
  - targets: - "public-ip-prod:9100"
             - "public-ip-test:9100"
```

Part 2: Install and Configure Grafana

Step 1: Install required dependencies

sudo apt-get install -y apt-transport-https software-properties-common wget

Step 2: Add Grafana GPG key and repository

sudo wget -q -O /usr/share/keyrings/grafana.key https://apt.grafana.com/gpg.key
echo "deb [signed-by=/usr/share/keyrings/grafana.key] https://packages.grafana.com/oss/deb stable main" | sudo tee /etc/apt/sources.list.d/grafana.list

Step 3: Update package list and install Grafana

sudo apt-get update
sudo apt-get install -y Grafana

Step 4: Enable and start Grafana service

sudo systemctl enable --now grafana-server
sudo systemctl status grafana-server
Step 5: Access Grafana web UI
Open your browser and go to:

http://localhost:3000
Default login credentials:

- Username: admin
- Password: admin

After installation access the grafana using http://localhost:3000

You will be prompted to change the password on first login.

Part 3: Configure Grafana to use Prometheus as a Data Source

1. Log in to Grafana UI.
2. Go to **Configuration** (gear icon) > **Data Sources** > **Add data source**.
3. Select **Prometheus**.
4. Set the URL to:

text
http://localhost:9090

Note : I have missed the screenshot of prometheus

Now go to Grafana website in Grafana go to connection >in connection > click data sources in data source add the dat source as Prometheus then add the website link in the url,then click save

After adding the data sources ,go to the dashboard creation,create new dashboard
In new dashboard add new panel in new panel,name it as the disk_utilization
Disk_utilization: 100 * (1 - (node_filesystem_avail_bytes{fstype=~"ext4|xfs"} / node_filesystem_size_bytes{fstype=~"ext4|xfs"}))

cpu_utilization add the code : 100 - (avg by(instance) (rate(node_cpu_seconds_total{mode="idle"}[5m])) * 100)

Memory utilization= 100 * (1 - ((node_memory_MemAvailable_bytes) / (node_memory_MemTotal_bytes)))

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