

3-tier Infrastructure with Terraform and Ansible

Overview

This project involves creating a scalable and highly available infrastructure on AWS using Terraform for infrastructure provisioning and Ansible for configuration management. The setup includes a VPC with subnets, NAT gateway, internet gateway, EC2 instances, security groups, a load balancer, RDS instance, CloudWatch for monitoring, SNS for notifications, and Route 53 for DNS management. The EC2 instances are configured using Ansible to host a web application with a backend and frontend server setup.

Prerequisites

Before proceeding with the infrastructure setup and configuration, ensure the following prerequisites are met on your local Ubuntu machine:

- **Create SSH Key Pair:**

- Generate an SSH key pair and import the public key into your AWS account.

```
#ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

- **AWS CLI Installation:**

- Install AWS CLI.
- Configure AWS CLI with your access key and secret key.
- Install Terraform.
- Install Ansible.

Tools and Technologies

- **Infrastructure Provisioning:** Terraform
- **Configuration Management:** Ansible, shell scripting
- **Secrets Management:** Terraform Vault, Ansible Vault
- **Cloud Provider:** AWS
- **Services Used:** VPC, EC2, RDS, S3, CloudWatch, SNS, Route 53, Load Balancer

Three-Tier Architecture

- **Presentation Tier:** Managed by the frontend server (NGINX) which handles user interaction.
- **Application Tier:** Managed by the backend server (Apache) which handles business logic.
- **Data Tier:** Managed by the RDS instance which handles data storage and management.

Infrastructure Setup with Terraform

1. VPC Creation

- Create a new VPC with CIDR 192.168.0.0/20.
- Create public and private subnets in different availability zones.
- Enable auto-assign public IP on the public subnet.
- Create and attach an internet gateway to the VPC.
- Create a route table, add a route to the internet gateway, and associate it with the public subnet

- Create a NAT gateway in the public subnet and update the route table for the private subnet to route traffic through the NAT gateway.

2. EC2

- Launch an EC2 instances in the public and private subnet.
- Create security groups for the instances with appropriate inbound rules (SSH, HTTP, MYSQL/Aurora, ALL ICMP, Custom TCP on port 8080).
- Create an Application Load Balancer and configure it to balance traffic to the private instance.
- Create a launch template and auto-scaling group to manage the EC2 instances in the private subnet.

1. RDS:

- Create parameter groups and subnet groups for RDS.
- Launch an RDS instance using the created groups

2. CloudWatch and SNS:

- Set up CloudWatch for monitoring and SNS for notifications.

3. Route 53:

- Create a Route 53 hosted zone.
- Add a new A record with a simple routing policy.
- Create a health check for the DNS.

Configuration Management

Shell script :

- I wrote a shell script to create an inventory file with the IP addresses of the running instances.
- I saved the terraform output in Json format and then I used that output from variable in Ansible playbook.

Ansible :

1. Private and Public Instances Configuration:

- Use Ansible to install Apache on the backend server (private instance).
- Store the WAR file of the application in the webapps folder and the sql-connector.jar file in the lib folder.
- Configure context.xml

2. Frontend Configuration:

- Install Nginx on the frontend server (public instance).
- Configure Nginx to proxy pass requests to the backend server load balancer DNS.

3. Database Configuration:

- Ansible Playbook to Create Database
- Use the RDS endpoint stored in the JSON file to create a database

4. Securing Credentials:

- Use Terraform Vault and Ansible Vault to secure database passwords.

Terraform and ansible Script:

https://github.com/Sameerpatlekar/project/tree/main/terraform_project/3-tier

Setup terraform vault on ec2 instance

Install Vault on the EC2 instance

To install Vault on the EC2 instance, you can use the following steps:

Install gpg

```
sudo apt update && sudo apt install gpg
```

Download the signing key to a new keyring

```
wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --  
dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg
```

Verify the key's fingerprint

```
gpg --no-default-keyring --keyring /usr/share/keyrings/hashicorp-  
archive-keyring.gpg --fingerprint
```

Add the HashiCorp repo

```
echo "deb [arch=$(dpkg --print-architecture) signed-  
by=/usr/share/keyrings/hashicorp-archive-keyring.gpg]  
https://apt.releases.hashicorp.com $(lsb_release -cs) main" |  
sudo tee /etc/apt/sources.list.d/hashicorp.list
```

```
sudo apt update
```

Finally, Install Vault

```
sudo apt install vault
```

Configure Terraform to read the secret from Vault.

Detailed steps to enable and configure AppRole authentication in HashiCorp Vault:

1. **Enable AppRole Authentication:**

To enable the AppRole authentication method in Vault, you need to use the Vault CLI or the Vault HTTP API.

Using Vault CLI:

Run the following command to enable the AppRole authentication method:

```
vault auth enable approle
```

This command tells Vault to enable the AppRole authentication method.

2. **Create an AppRole:**

We need to create policy first,

```
vault policy write terraform - <<EOF
```

```
path "*" {  
  capabilities = ["list", "read"]  
}
```

```
path "secrets/data/*" {  
  capabilities = ["create", "read", "update", "delete", "list"]  
}
```

```
path "kv/data/*" {  
  capabilities = ["create", "read", "update", "delete", "list"]  
}
```

EOF

Now you'll need to create an AppRole with appropriate policies and configure its authentication settings. Here are the steps to create an AppRole:

a. Create the AppRole:

```
vault write auth/approle/role/terraform \
    token_policies=terraform
```

/*Optional

```
secret_id_ttl=10m \
    token_num_uses=10 \
    token_ttl=20m \
    token_max_ttl=30m \
    secret_id_num_uses=40 \
```

*/

3. Generate Role ID and Secret ID:

After creating the AppRole, you need to generate a Role ID and Secret ID pair. The Role ID is a static identifier, while the Secret ID is a dynamic credential.

a. Generate Role ID:

You can retrieve the Role ID using the Vault CLI:

```
vault read auth/approle/role/my-approle/role-id
```

Save the Role ID for use in your Terraform configuration.

b. Generate Secret ID:

To generate a Secret ID, you can use the following command:

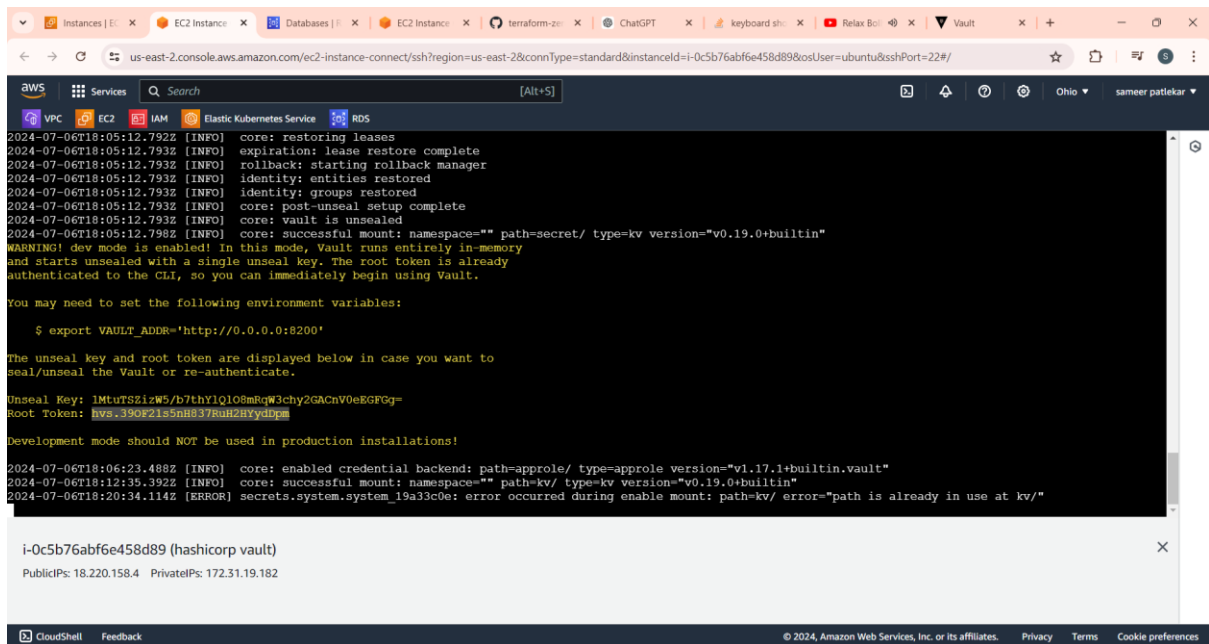
`vault write -f auth/approle/role/my-approle/secret-id`

Start Vault.

To start Vault, you can use the following command:

`vault server -dev -dev-listen-address="0.0.0.0:8200"`

Copy the root token



The screenshot shows a terminal window with the following content:

```
2024-07-06T18:05:12.792Z [INFO] core: restoring leases
2024-07-06T18:05:12.793Z [INFO] expiration: lease restore complete
2024-07-06T18:05:12.793Z [INFO] rollback: starting rollback manager
2024-07-06T18:05:12.793Z [INFO] identity: entities restored
2024-07-06T18:05:12.793Z [INFO] identity: groups restored
2024-07-06T18:05:12.793Z [INFO] core: post-unseal setup complete
2024-07-06T18:05:12.793Z [INFO] core: vault is unsealed
2024-07-06T18:05:12.793Z [INFO] core: successful mount: namespace="" path=secret/ type=kv version="v0.19.0+builtin"
WARNING! dev mode is enabled! In this mode, Vault runs entirely in-memory
and starts unsealed with a single unseal key. The root token is already
authenticated to the CLI, so you can immediately begin using Vault.

You may need to set the following environment variables:

$ export VAULT_ADDR='http://0.0.0.0:8200'

The unseal key and root token are displayed below in case you want to
seal/unseal the Vault or re-authenticate.

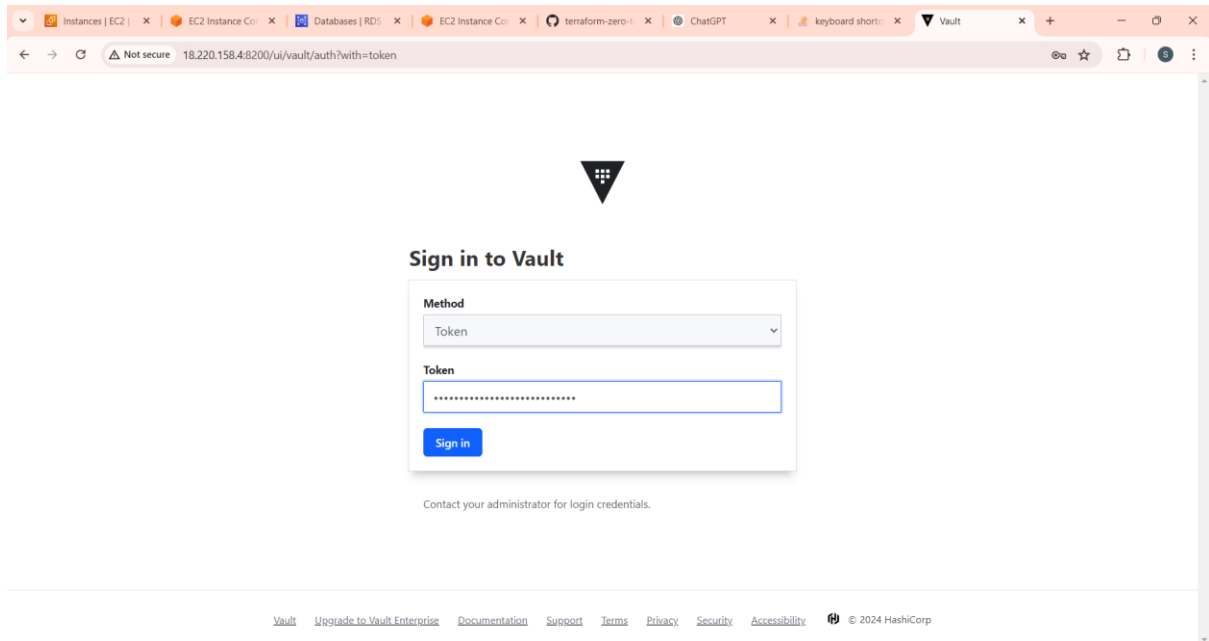
Unseal Key: 1MtutSZizW5/b7thYlQl08mRqW3chy2GAcnV0eEGFGg-
Root Token: hvs.390F2155nH83/RuH2Hyvdpm

Development mode should NOT be used in production installations!

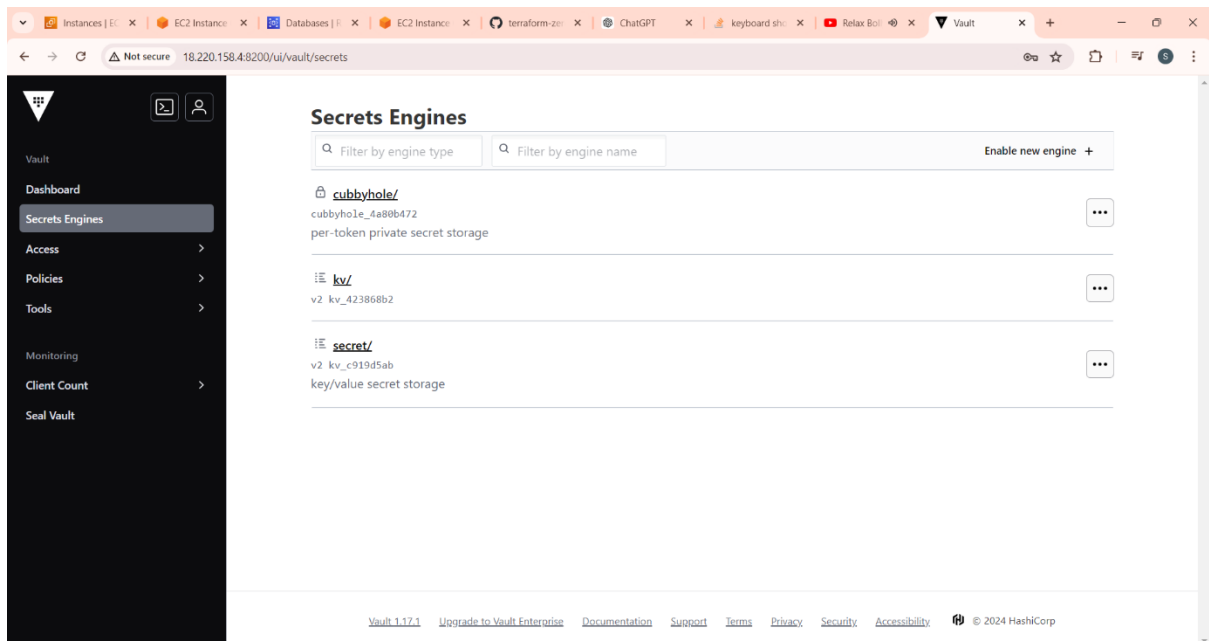
2024-07-06T18:06:23.488Z [INFO] core: enabled credential backend: path=approle/ type=approle version="v1.17.1+builtin.vault"
2024-07-06T18:12:35.392Z [INFO] core: successful mount: namespace="" path=kv/ type=kv version="v0.19.0+builtin"
2024-07-06T18:20:34.114Z [ERROR] secrets.system.system_19a33c0e: error occurred during enable mount: path=kv/ error="path is already in use at kv/"

i-0c5b76abf6e458d89 (hashicorp vault)
PublicIPs: 18.220.158.4 PrivateIPs: 172.31.19.182
```

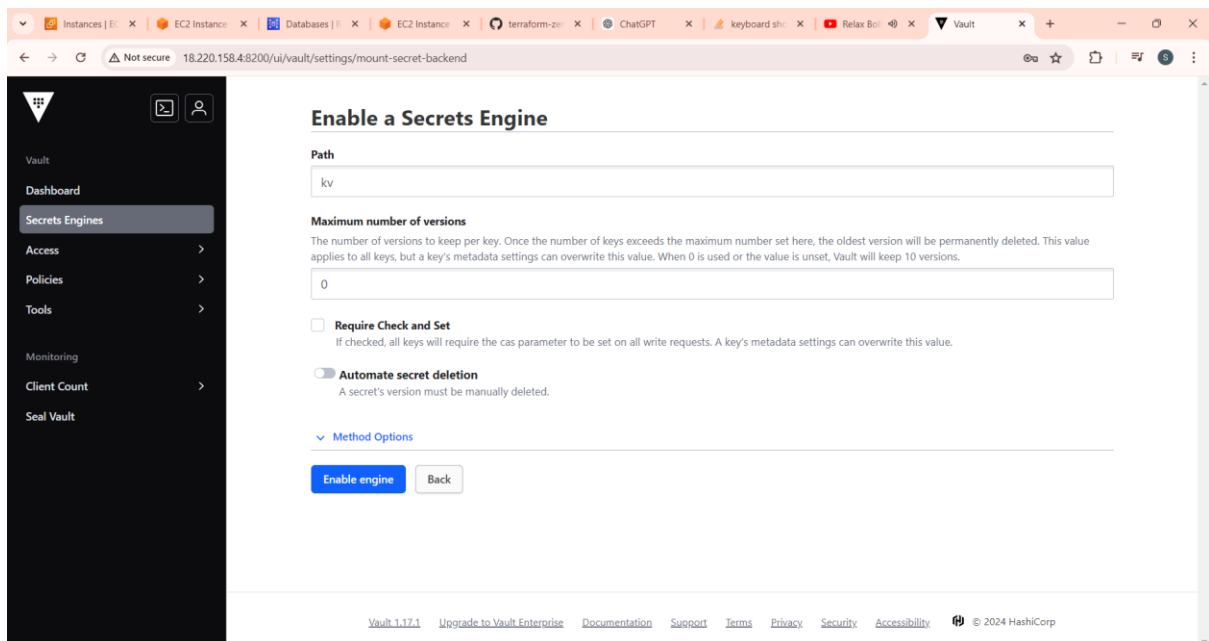
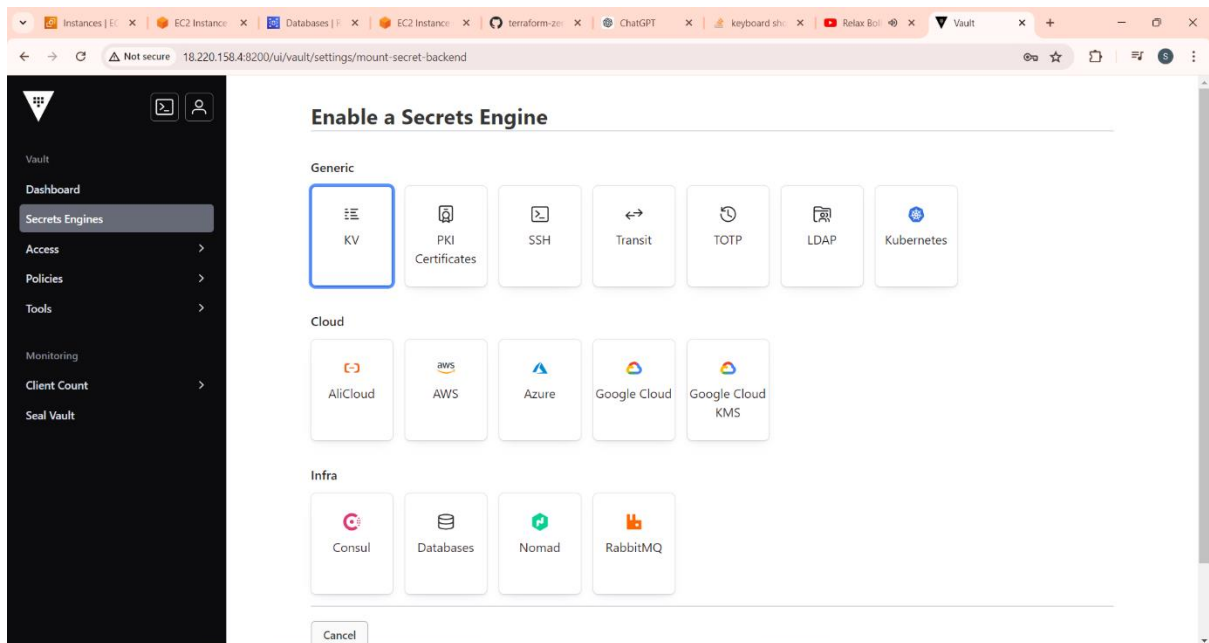

Open chrome enter ip address with port 8200 then paste token and then hit



In the Dashboard of the terraform vault go to secrets engines and click on enable new engine



Select engine



Setup Ansible Vault

1. Create a password for vault

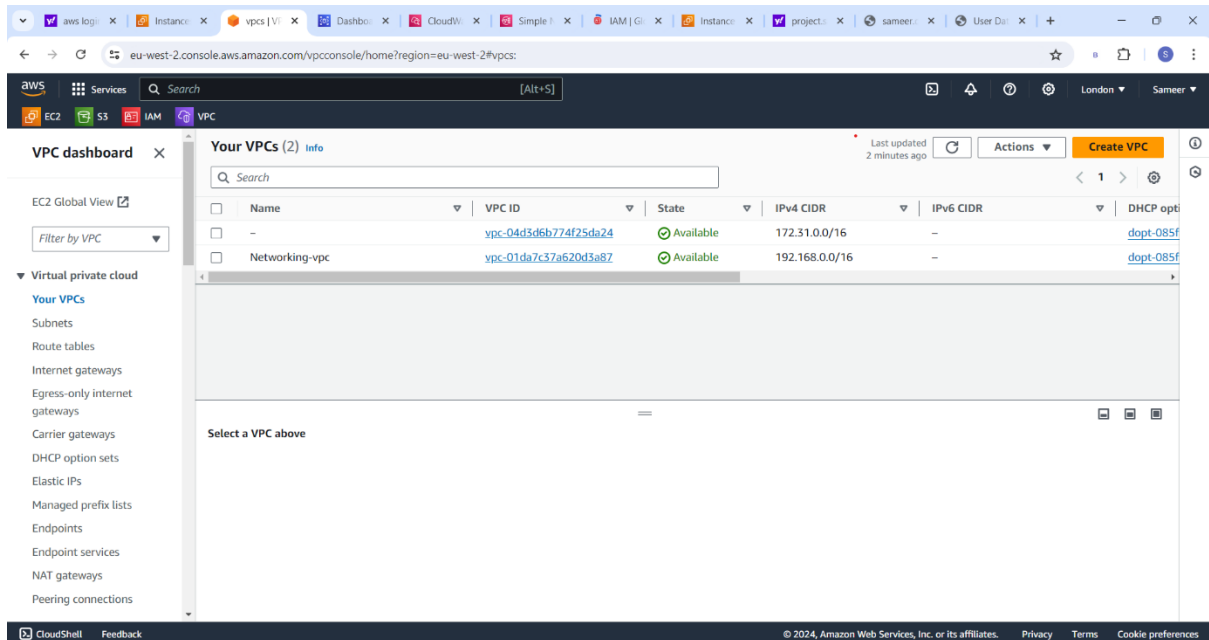
`openssl rand -base64 2048 > vault.pass`

2. Add your AWS credentials using the below vault command

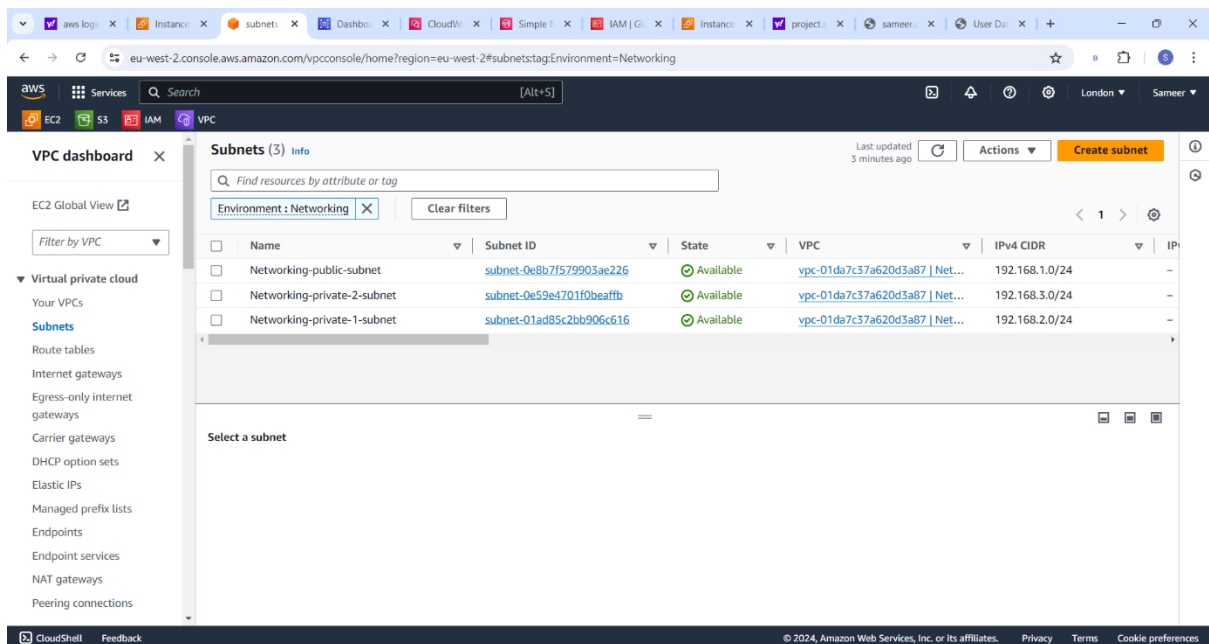
ansible-vault create group_vars/all/pass.yml --vault-password-file vault.pass

Infrastructure Setup with terraform

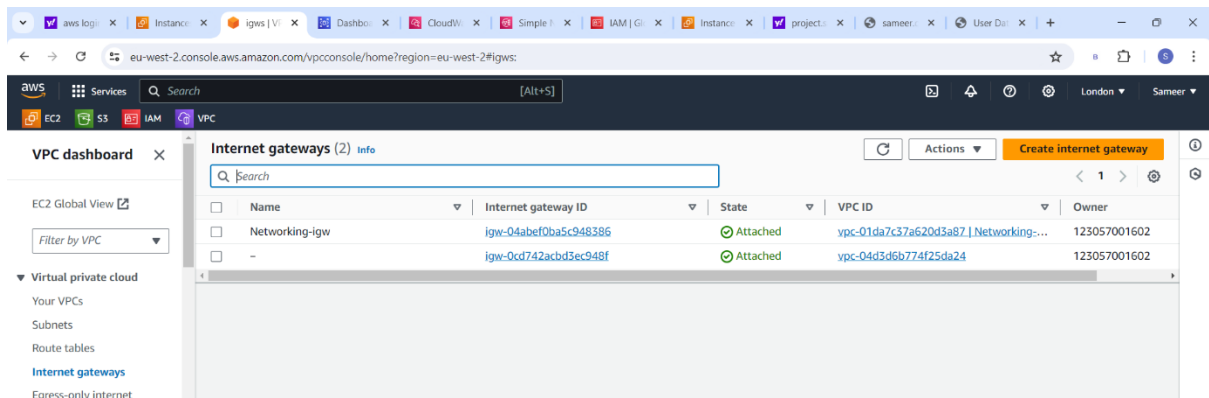
Create a new VPC with CIDR 192.168.0.0/20.



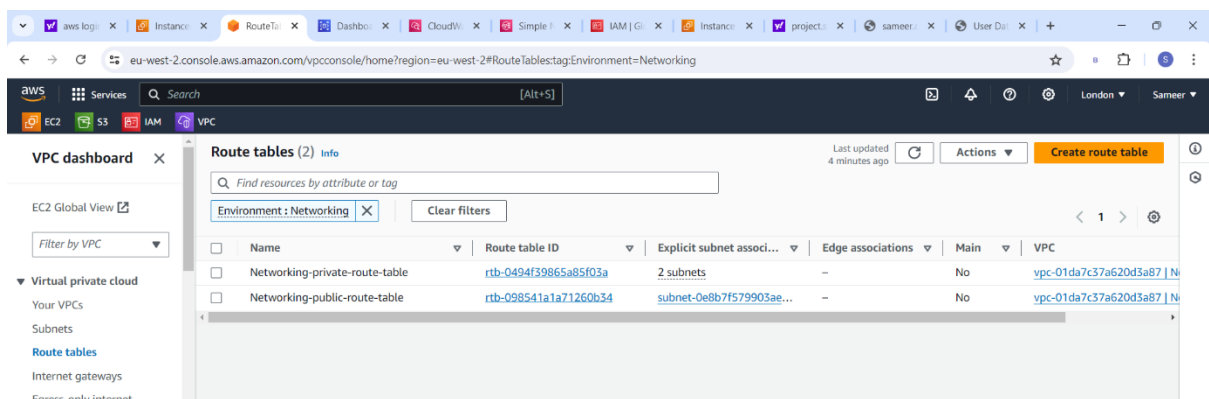
Create public and private subnets in different availability zones.
Enable auto-assign public IP on the public subnet.



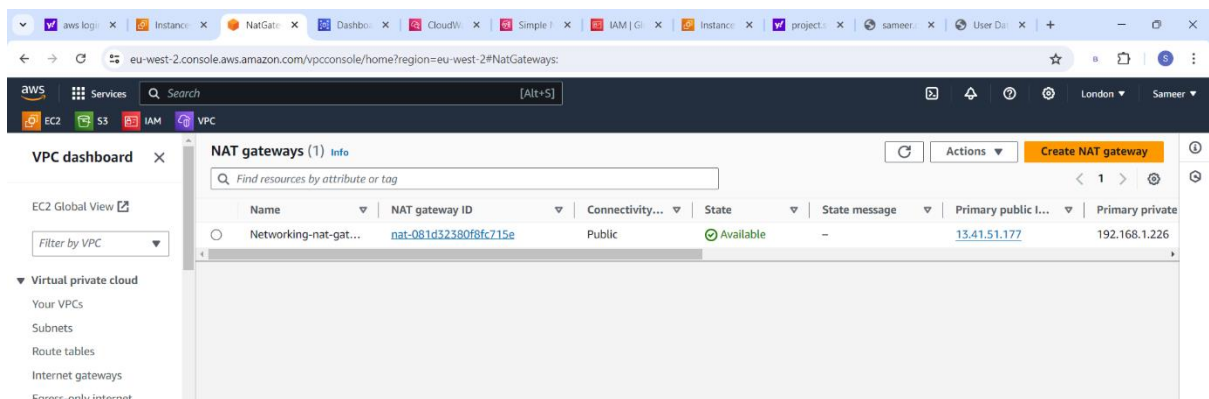
Create and attach an internet gateway to the VPC.



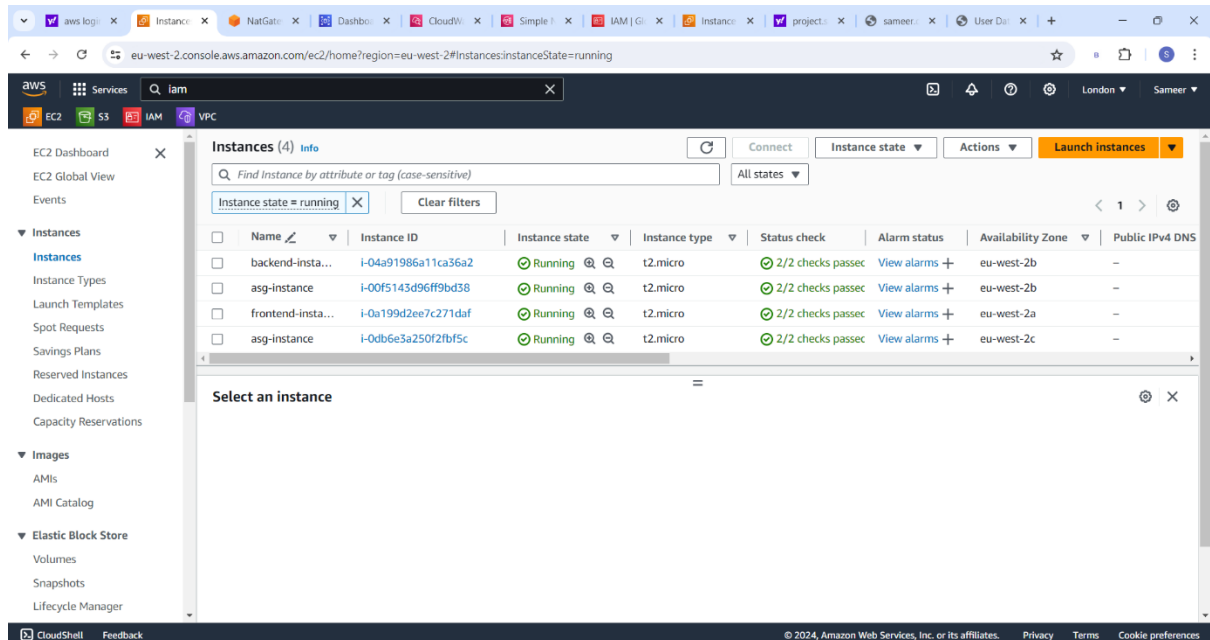
Create a route table, add a route to the internet gateway, and associate it with the public and private subnet



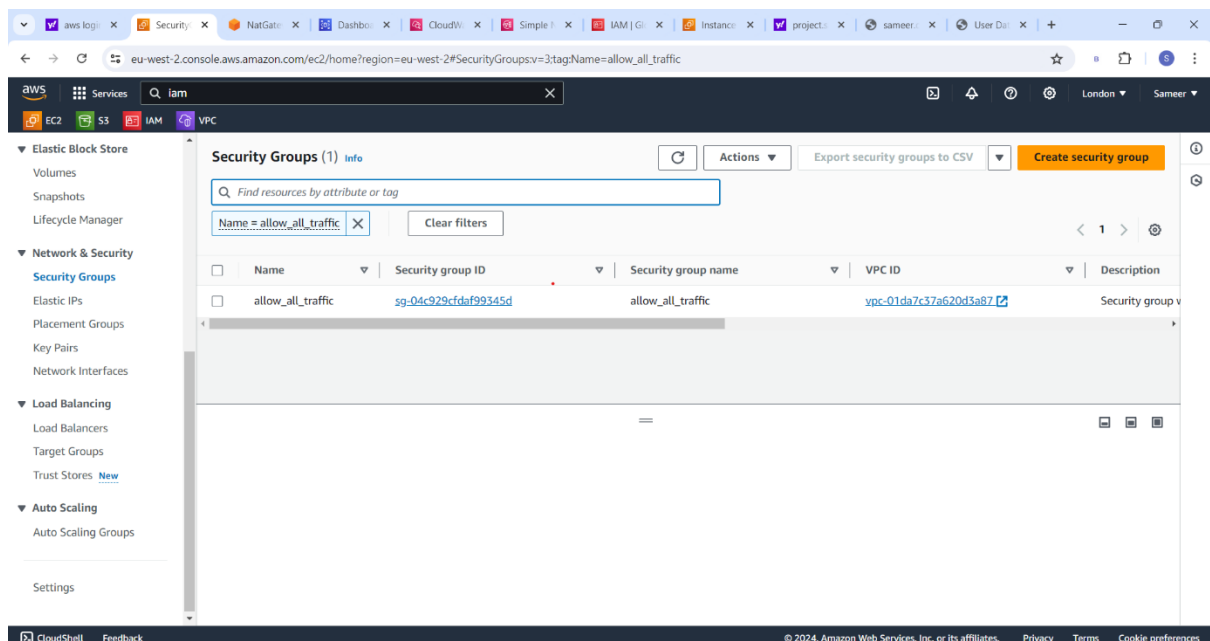
Create a NAT gateway in the public subnet and update the route table for the private subnet to route traffic through the NAT gateway.



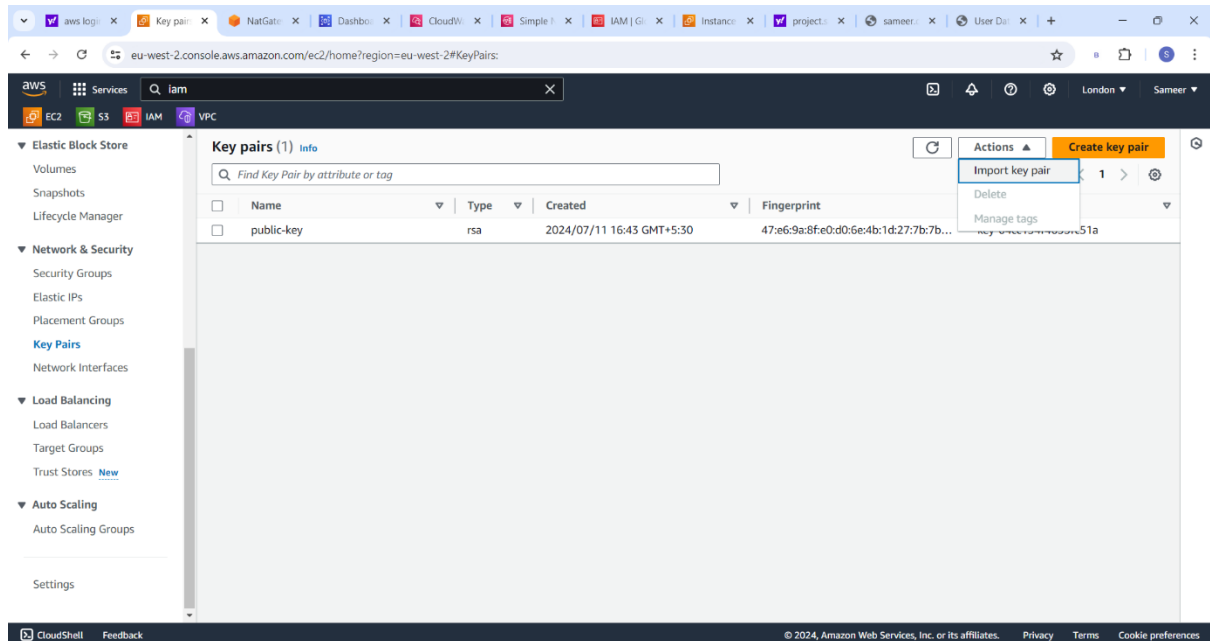
Launch an EC2 instances in the public and private subnet.



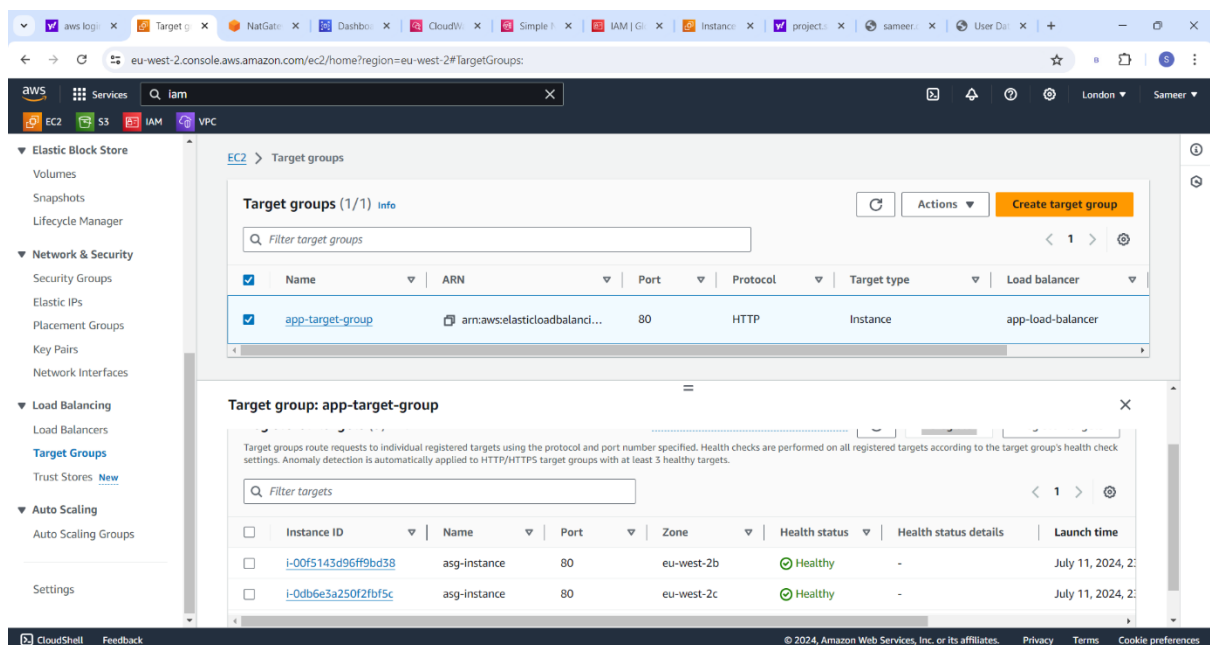
Create security groups for the instances with appropriate inbound rules (SSH, HTTP, MYSQL/Aurora, ALL ICMP, Custom TCP on port 8080).



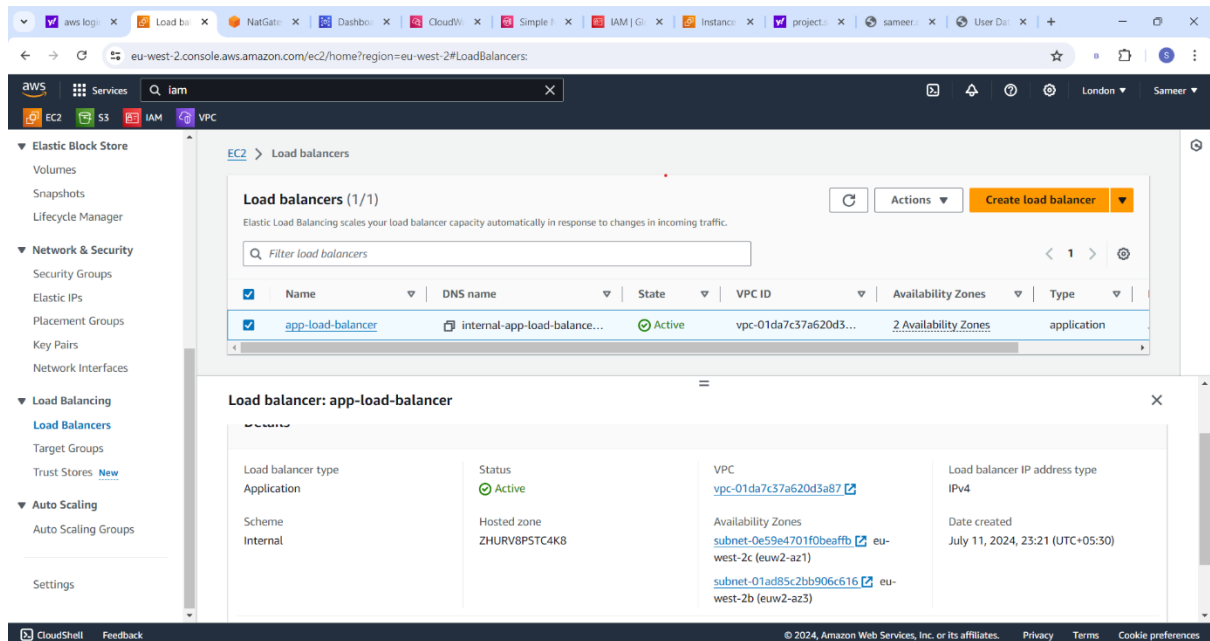
Import public key of my local machine



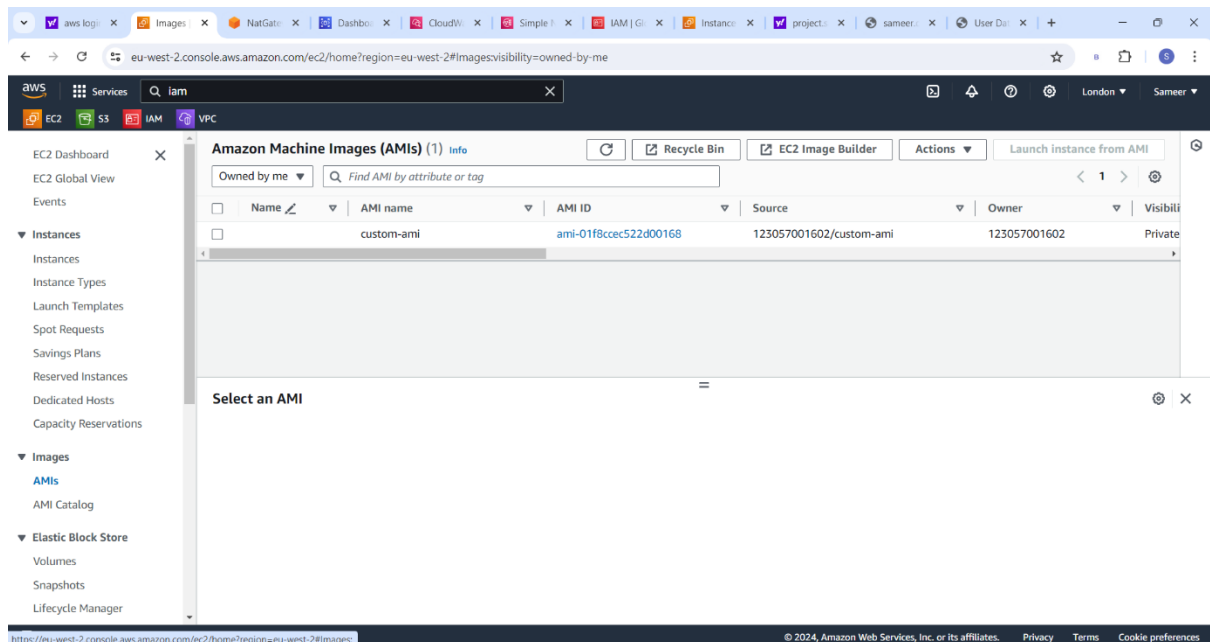
Create target group with private instance



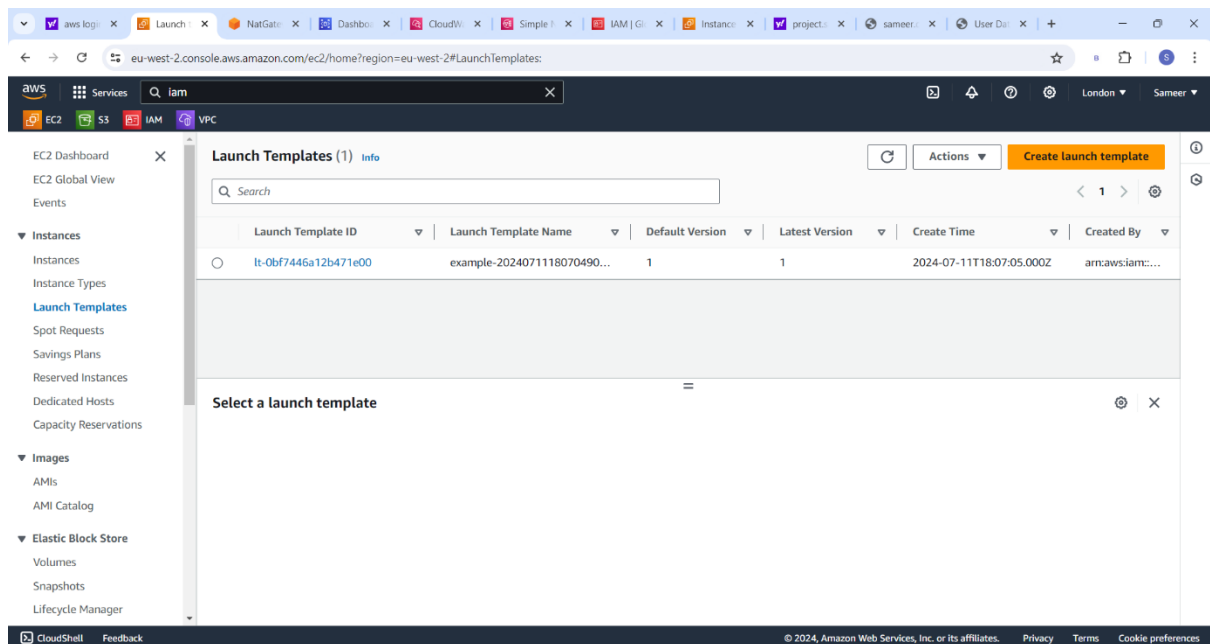
Create an Application Load Balancer and configure it to balance traffic to the private instance.



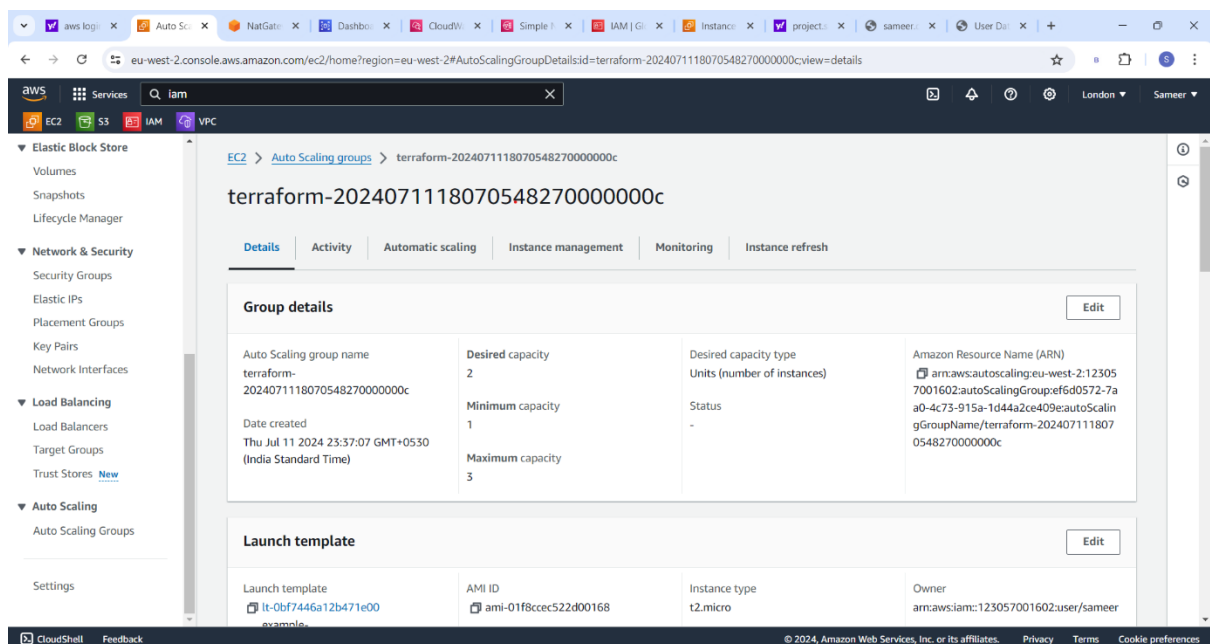
Create AMI of private instance



Create launch template with existing AMI



Create a launch template and auto-scaling group to manage the EC2 instances in the private subnet.



Create subnet groups for RDS

The screenshot shows the AWS Management Console for the 'rds_subnet_group' resource. The left sidebar contains the 'Amazon RDS' navigation menu with options like Dashboard, Databases, Query Editor, Performance insights, Snapshots, Exports in Amazon S3, Automated backups, Reserved instances, Proxies, Subnet groups, Parameter groups, Option groups, Custom engine versions, and Zero-ETL integrations. The main content area displays the 'Subnet group details' for 'rds_subnet_group'. It shows the VPC ID as 'vpc-01da7c37a620d3a87', the ARN as 'arn:aws:rds:eu-west-2:123057001602:subgrp:rds_subnet_group', and the supported network types as 'IPv4'. The description is 'Managed by Terraform'. Below this, the 'Subnets (2)' section shows a table with two subnets:

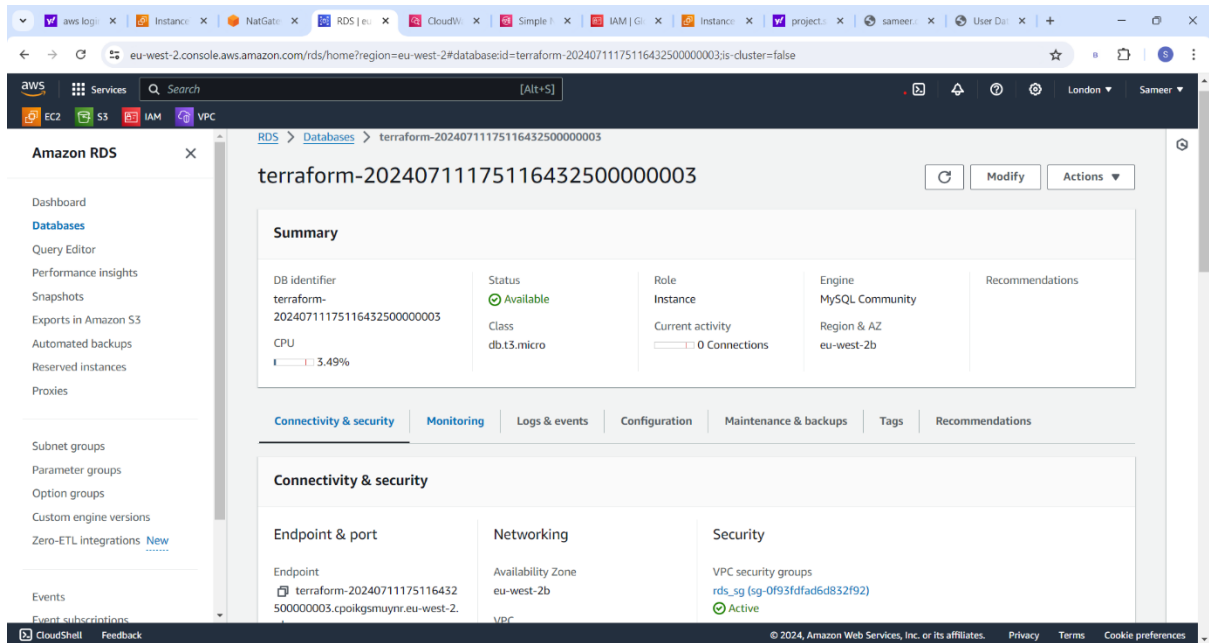
Availability zone	Subnet ID	CIDR block
eu-west-2c	subnet-0e59e4701f0beaffb	192.168.3.0/24
eu-west-2b	subnet-01ad85c2bb906c616	192.168.2.0/24

Create parameter groups .

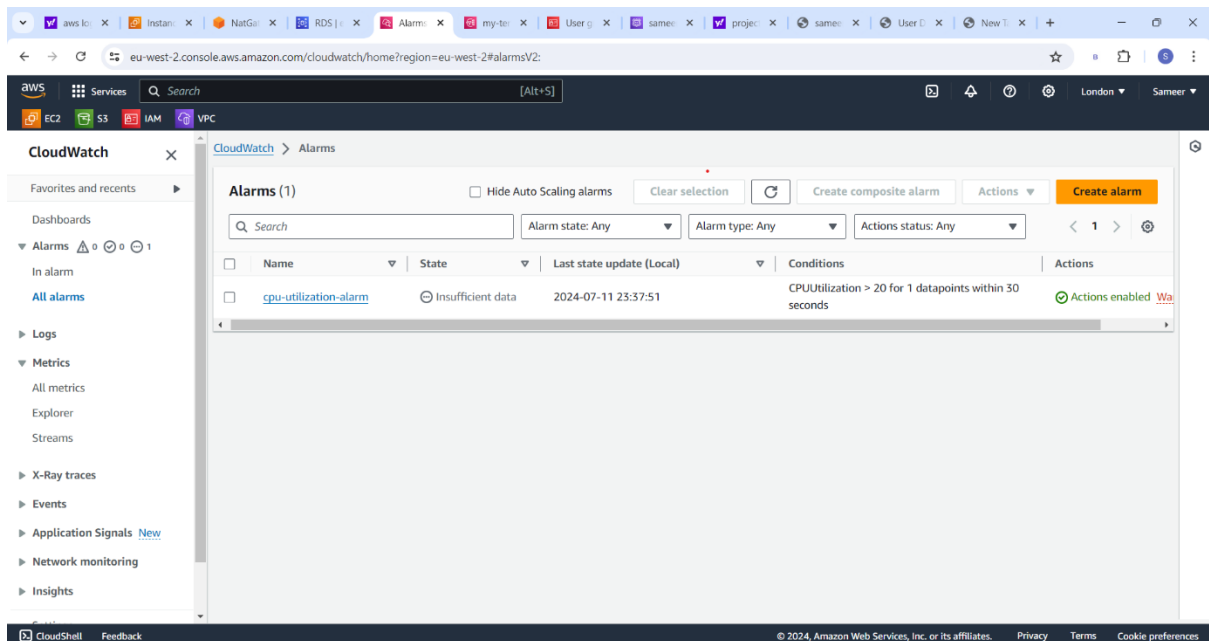
The screenshot shows the AWS Management Console for the 'Parameter groups' resource. The left sidebar contains the 'Amazon RDS' navigation menu. The main content area displays the 'Parameter groups' page with tabs for 'Custom' and 'Default'. The 'Custom' tab is active, showing a list of custom parameter groups. There is a search bar with the placeholder text 'Filter by custom parameter groups' and a 'Create parameter group' button. Below the search bar, a table lists the custom parameter groups:

Name	Family	Type	Description	ARN
my-db-parameter-group	mysql8.0	DB instance parameter group	My custom DB parameter group for MySQL 5.7	arn:aws:rds:eu-west-2:123057001602:subgrp:rds_subnet_group

Launch an RDS instance using the created groups



Create Alarm with CloudWatch



Create SNS topic

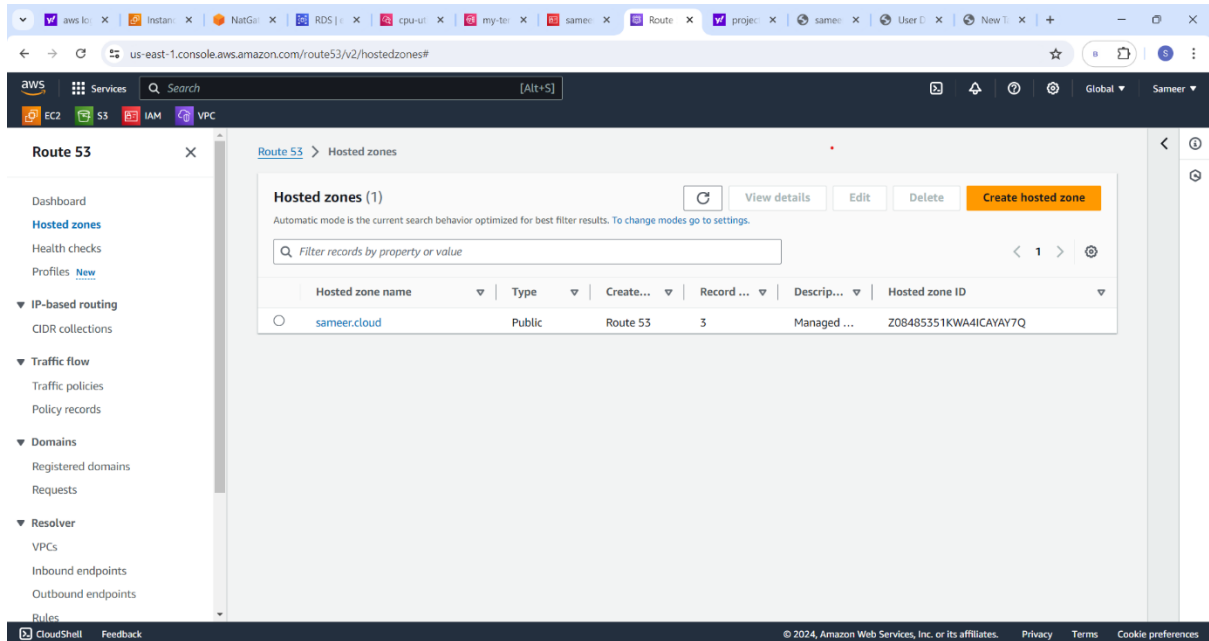
The screenshot shows the AWS Management Console for the 'my-terraform-project-sns' topic. The left sidebar contains navigation links for Dashboard, Topics, Subscriptions, Mobile, Push notifications, Text messaging (SMS), and Origination numbers. The main content area displays the topic details, including the Name 'my-terraform-project-sns', Display name '-', ARN 'arn:aws:sns:eu-west-2:123057001602:my-terraform-project-sns', Topic owner '123057001602', and Type 'Standard'. Below the details, there are tabs for Subscriptions, Access policy, Data protection policy, Delivery policy (HTTP/S), Delivery status logging, Encryption, and Tags. The Subscriptions tab is active, showing a table with one subscription. The subscription has ID 'b205c742-1f83-48bc-9805-56e4...', Endpoint 'sameer.gcp.jam@gmail.com', Status 'Confirmed', and Protocol 'EMAIL'. Buttons for Edit, Delete, Request confirmation, Confirm subscription, and Create subscription are visible.

ID	Endpoint	Status	Protocol
b205c742-1f83-48bc-9805-56e4...	sameer.gcp.jam@gmail.com	Confirmed	EMAIL

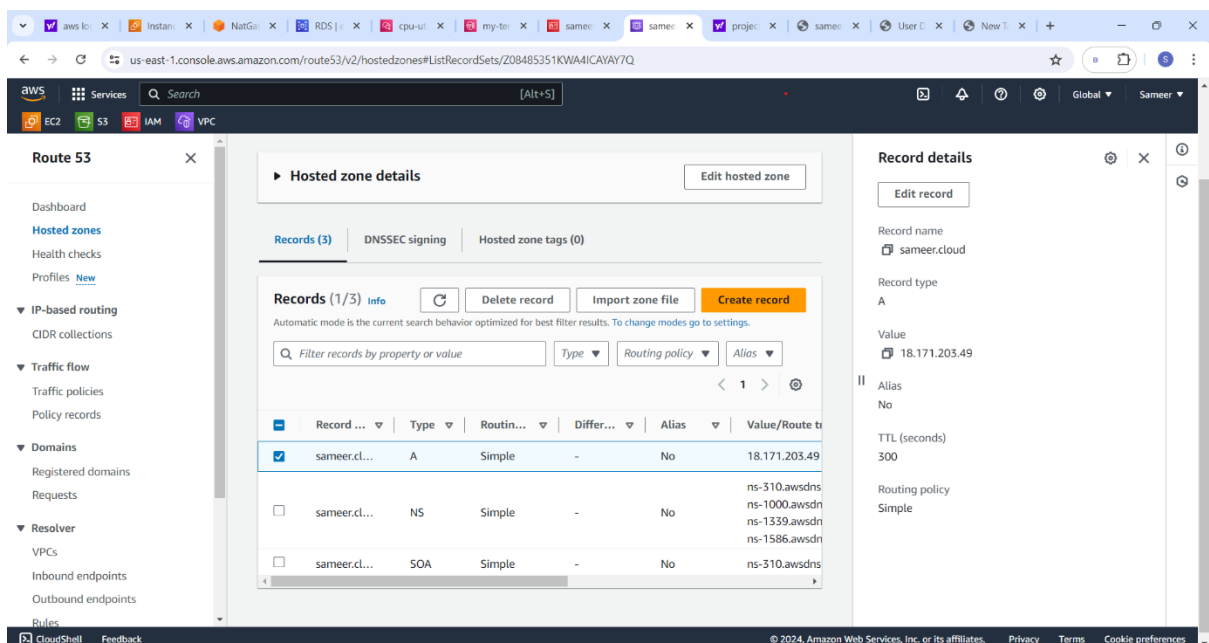
Gmail for confirmation from aws

The screenshot shows a Gmail inbox with an email titled 'AWS Notification - Subscription Confirmation'. The email is from 'AWS Notifications' and is dated 'Thu, Jul 11, 9:23 PM (2 hours ago)'. The body of the email states: 'You have chosen to subscribe to the topic: arn:aws:sns:eu-west-2:123057001602:my-terraform-project-sns. To confirm this subscription, click or visit the link below: [Confirm subscription](#)'. Below the email, there are buttons for Reply, Forward, and a smiley face icon. At the bottom of the screen, there is a notification bar that says 'Enable desktop notifications for Gmail. OK No thanks X'.

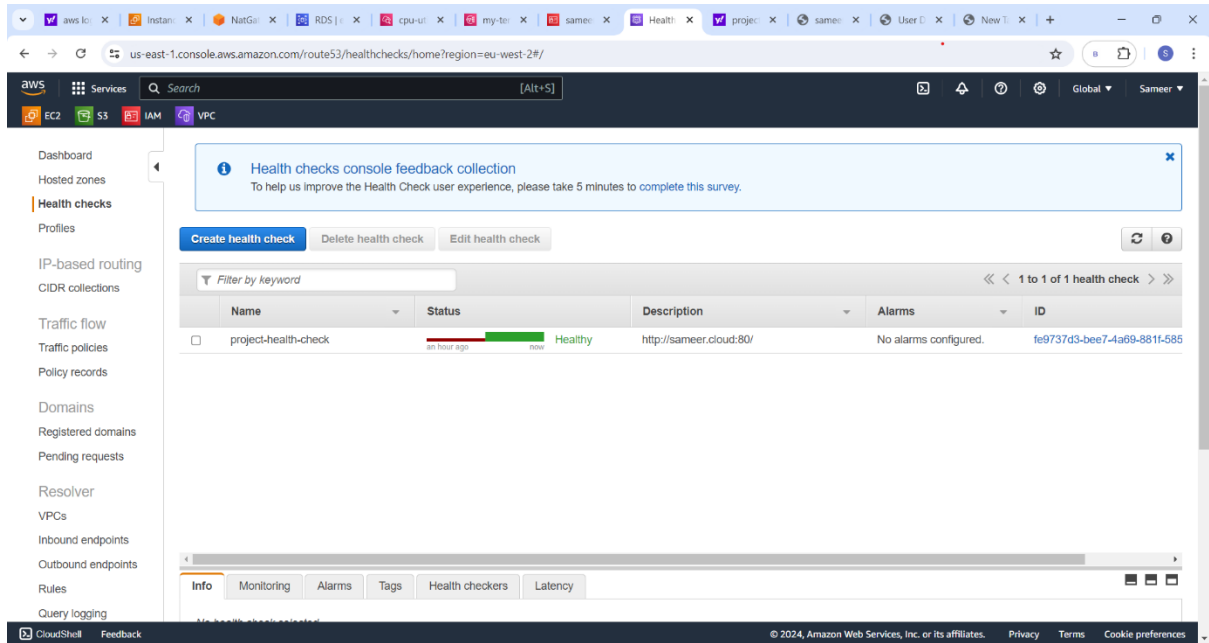
Create a Route 53 hosted zone.



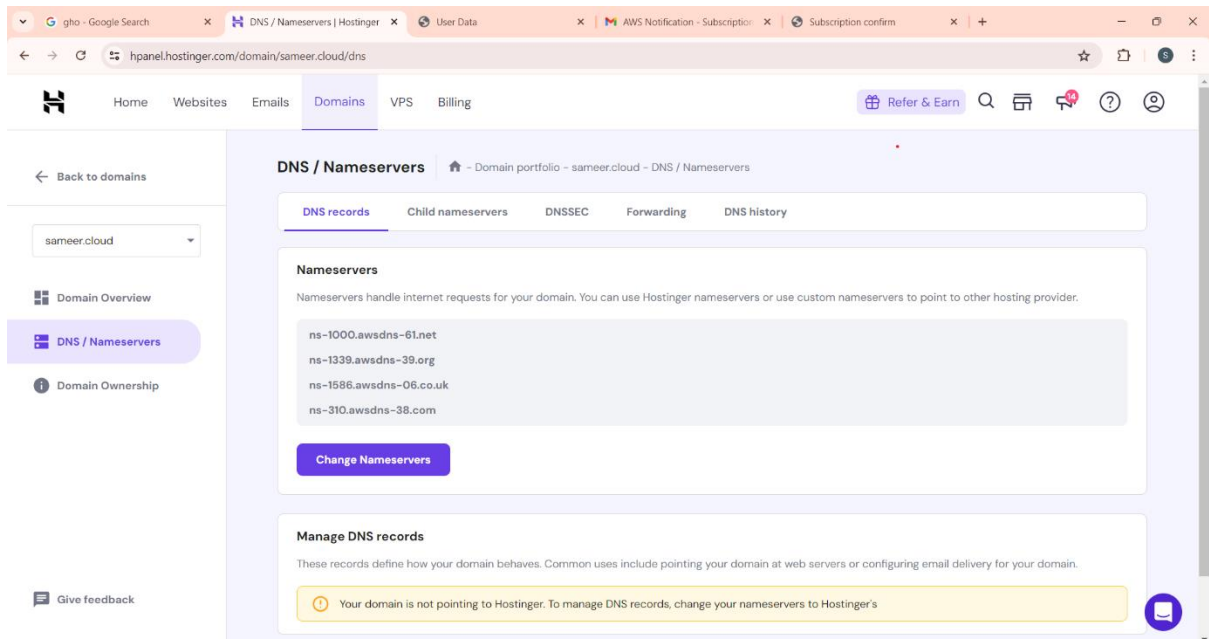
Add a new A record with a simple routing policy.



Create a health check for the DNS.



Add name server from route53 hoststage zone



Here we go, we see the application on internet.

aws lo x

Instan x

NatGw x

RDS [x

cpu-ut x

my-ter x

samee x

samee x

projec x

User D x

User D x

New T x

+ x

← → ↻ Not secure sameer.cloud ☆ 📄 📁 ⚙

Student Registration Form

Student Name

sameer

Student Address

asoli

Student Age

23

Student Qualification

bsc

Student Percentage

8.14

Year Passed

2023

register

Database is stored successfully.

aws lo x

Instan x

NatGw x

RDS [x

cpu-ut x

my-ter x

samee x

samee x

projec x

samee x

User D x

New T x

+ x

← → ↻ Not secure sameer.cloud/student/viewStudents ☆ 📄 📁 ⚙

[Register Student](#)

Students List

Student ID	StudentName	Student Addr	Student Age	Student Qualification	Student Percentage	Student Year Passed	Edit	Delete
1	l	fh	vb	b	fh	fh	edit	delete
2	l	fh	vb	b	fh	fh	edit	delete
3	sameer	asoli	23	bsc	8.14	2023	edit	delete