***AWS DEVOPS***

***INTERNSHIP***

***MAIN PROJECT – 1***

**DEPLOY 3 TIER ARCHITECTURE IN AWS USING TERRAFORM**

**WHAT IS TERRAFORM?**

Terraform is an open-source infrastructure as a code (IAC) tool that allows to create, manage & deploy the production-ready environment. Terraform codifies cloud APIs into declarative configuration files. Terraform can manage both existing service providers and custom in-house solutions.



**Objective of the project:**

The primary goal of this project is to design and deploy a scalable and resilient 3-tier architecture on Amazon Web Services (AWS) using Terraform. The 3-tier architecture comprises a web tier, an application tier, and a database tier, each residing in its dedicated network segment. This setup ensures modularity, scalability, and separation of concerns.

3-TIER ARCHITECTURE:

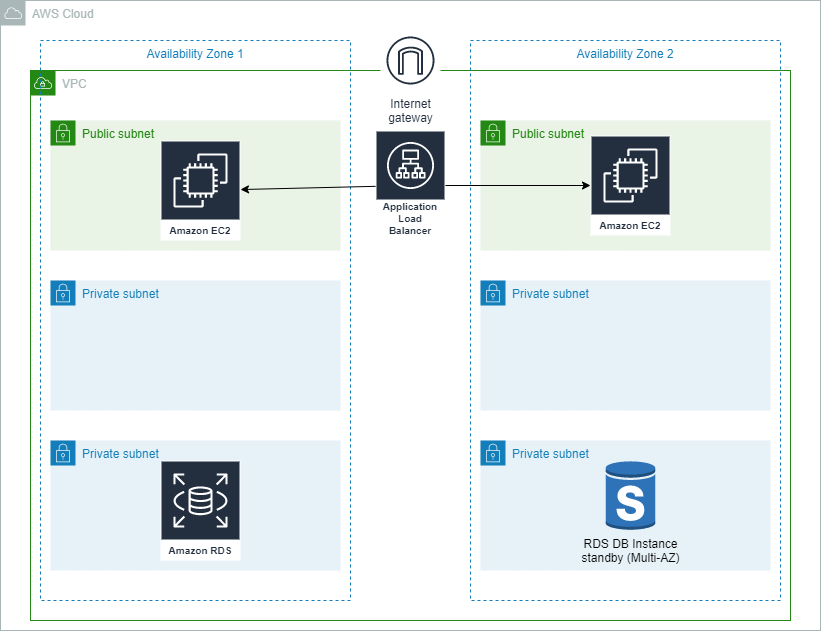
A 3-tier architecture is a software architecture pattern that divides an application into three interconnected components, or tiers, each responsible for specific functionalities. This separation enhances modularity, scalability, and maintainability of the application

The 3-tiers are:

1. Presentation Tier

2. Application Tier

3. Database Tier



Presentation Tier (User Interface Tier):

* Also known as the "front end" or "client tier."
* Responsible for presenting information to users and receiving user inputs.
* Typically includes the user interface (UI), graphical elements, and user interaction components.
* Interfaces with the application tier to request and display data.

Application Tier (Logic Tier or Middle Tier):

* Often referred to as the "business logic" or "server-side" tier.
* Contains the application logic that processes user requests, performs computations, and manages data flow.
* Implements business rules and processes to ensure proper application functionality.
* Interacts with both the presentation tier and the data tier.

Data Tier (Persistence Tier or Backend):

* Handles data storage, retrieval, and management.
* Stores and retrieves data from databases, file systems, or other data storage mechanisms.
* Ensures data integrity, security, and consistency.
* The data tier is responsible for managing the application's data, and it communicates with the application tier to perform necessary operations.

**KEY FEATURES**:

**Scalability**:

Utilizes Auto Scaling Groups to automatically adjust the number of instances based on demand.

**High Availability:**

Implements multi-AZ deployment for the database tier to ensure availability in case of a failure in one availability zone.

**Security**:

Configures security groups to control inbound and outbound traffic, enforcing the principle of least privilege.

**Load Balancing**:

Implements an Application Load Balancer (ALB) to evenly distribute incoming web traffic across multiple web servers.

**Maintainability**:

Separating the application into distinct tiers makes it easier to troubleshoot, update, and maintain. Each tier can be developed, tested, and updated independently.

**TOOLS AND TECHNOLOGIES**:

**AWS Services**:

VPC, EC2, RDS, ALB, Auto Scaling, Subnets, Security Groups.

**Infrastructure as Code (IaC**):

Utilizes Terraform for defining and provisioning AWS infrastructure.

**PREREQUISITES:**

AWS Account Setup:

Create and configure an AWS account to access AWS services.

Tool Installation:

Install the AWS CLI and Terraform on your local machine.

IAM User and Permissions:

Set up an IAM user with appropriate permissions for Terraform.

Networking Plan:

Plan the VPC, subnets, and security groups for your architecture.

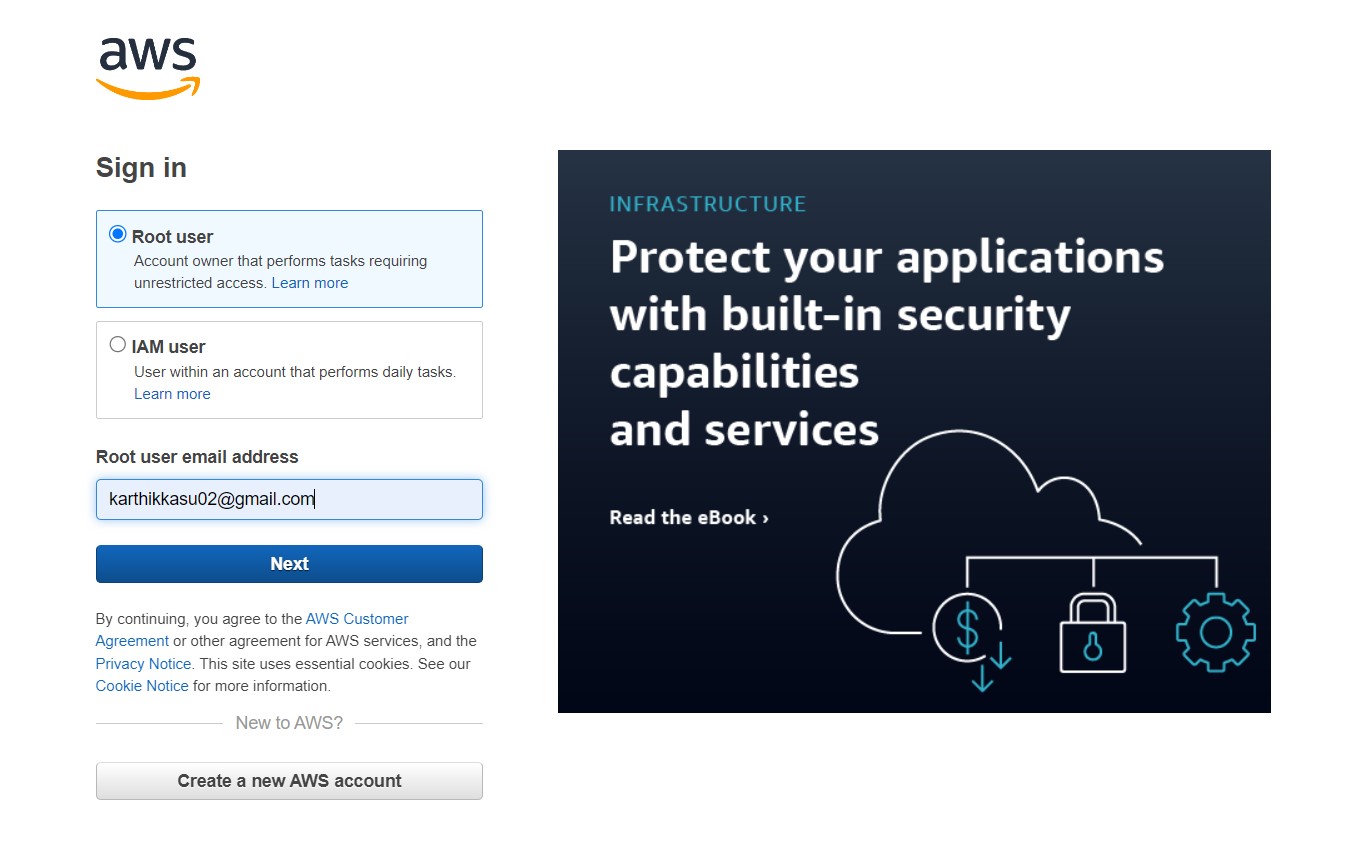
Resource Configuration:

Determine AMIs, database configurations, SSH key pairs, and prepare Terraform files for EC2 instances, RDS, and other resources.

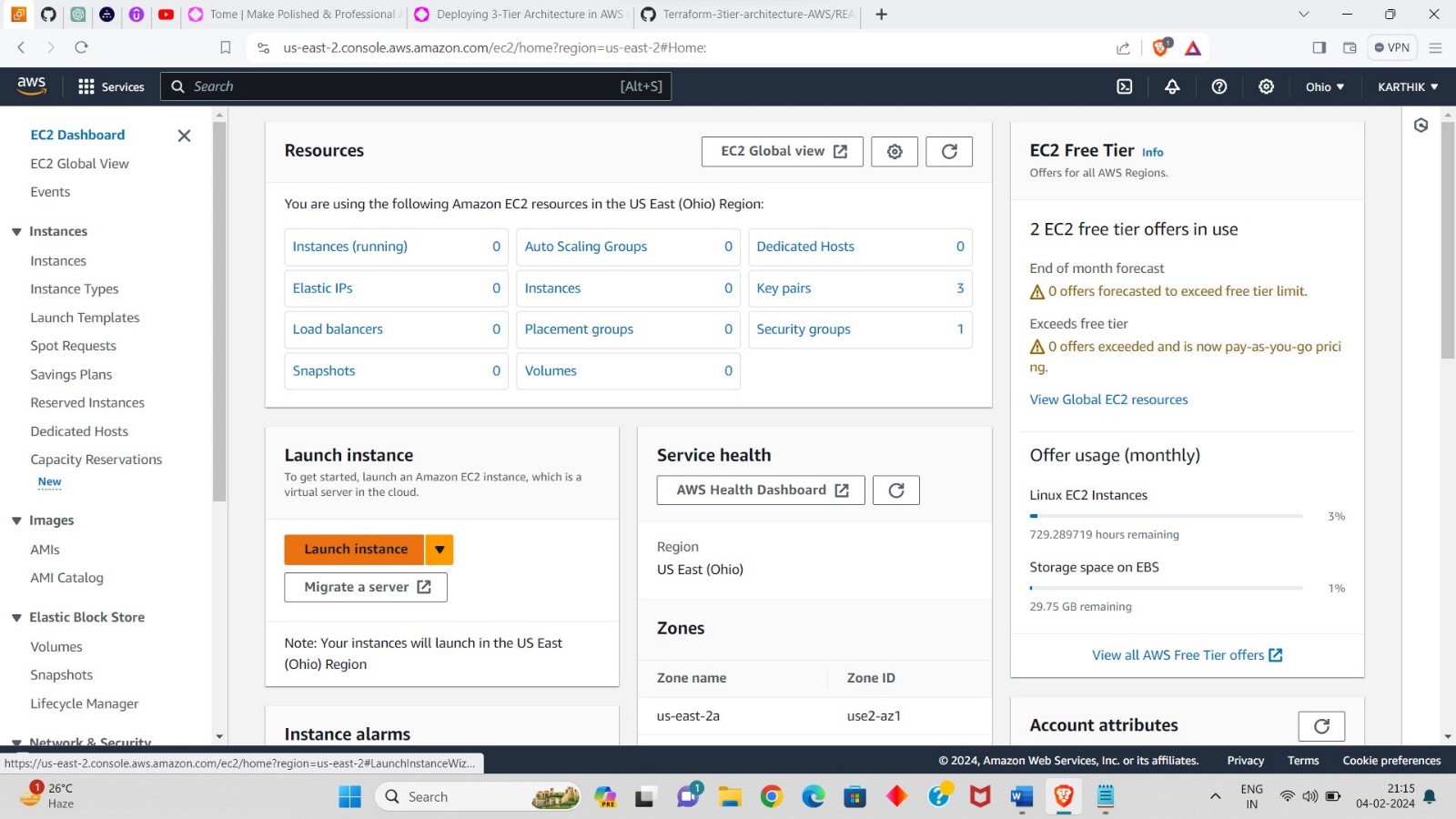
**PROCEDURE**

* Firstly, open your AWS Management Console and login to your aws account.
* Login with your console credentials.

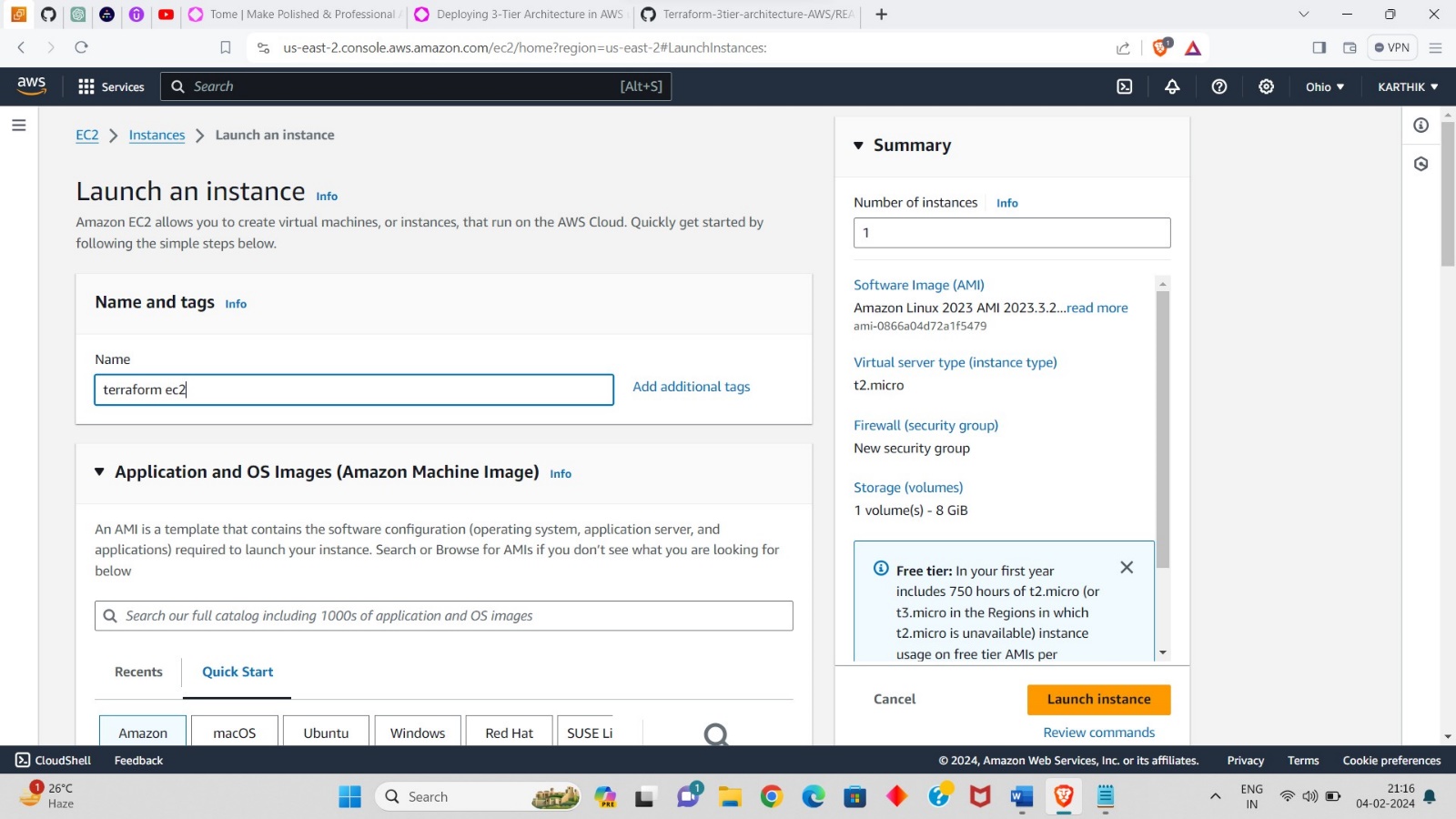
**We have to create an instance:**



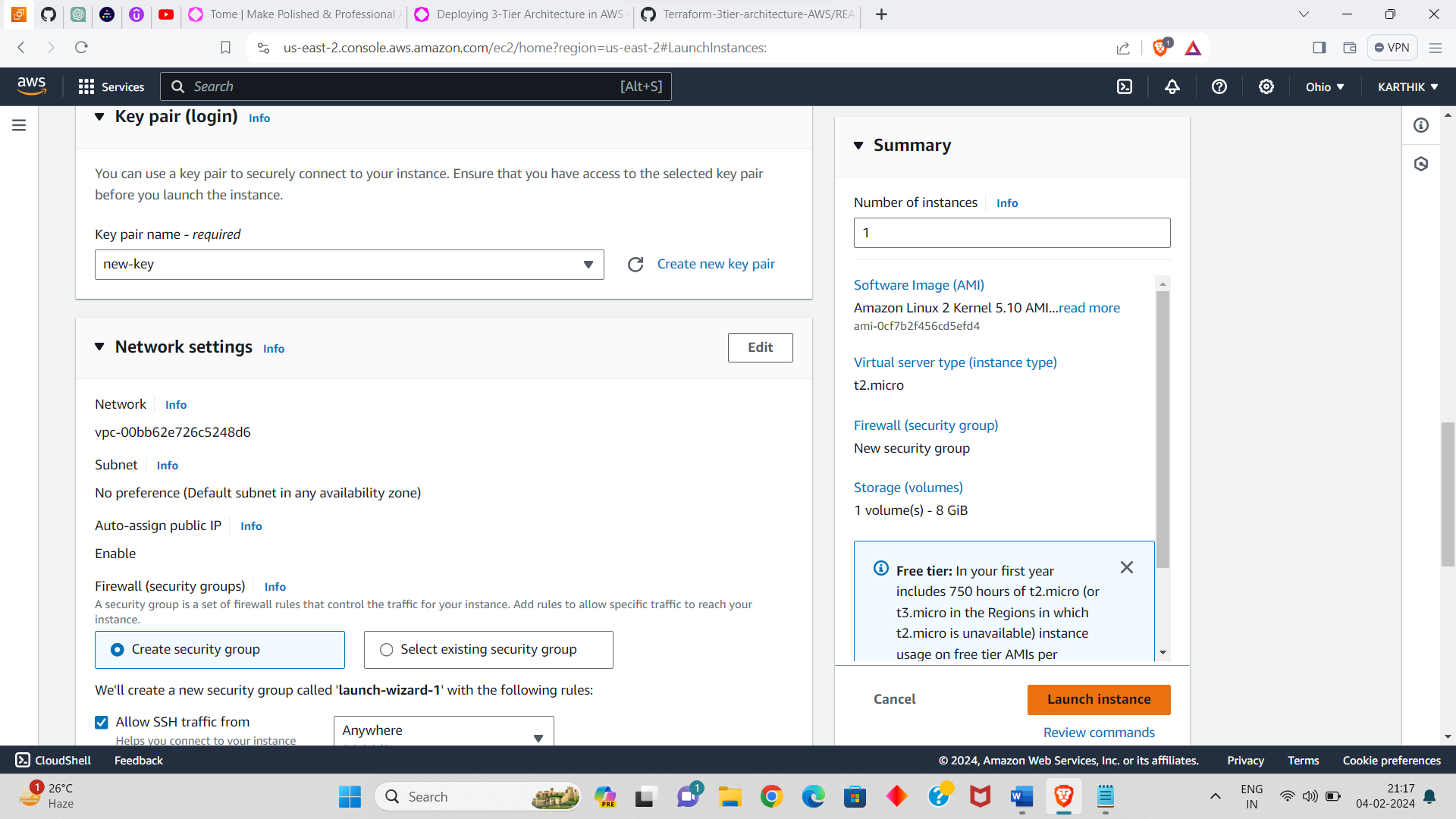
* Navigate to the EC2 Dashboard, and then go to instance.



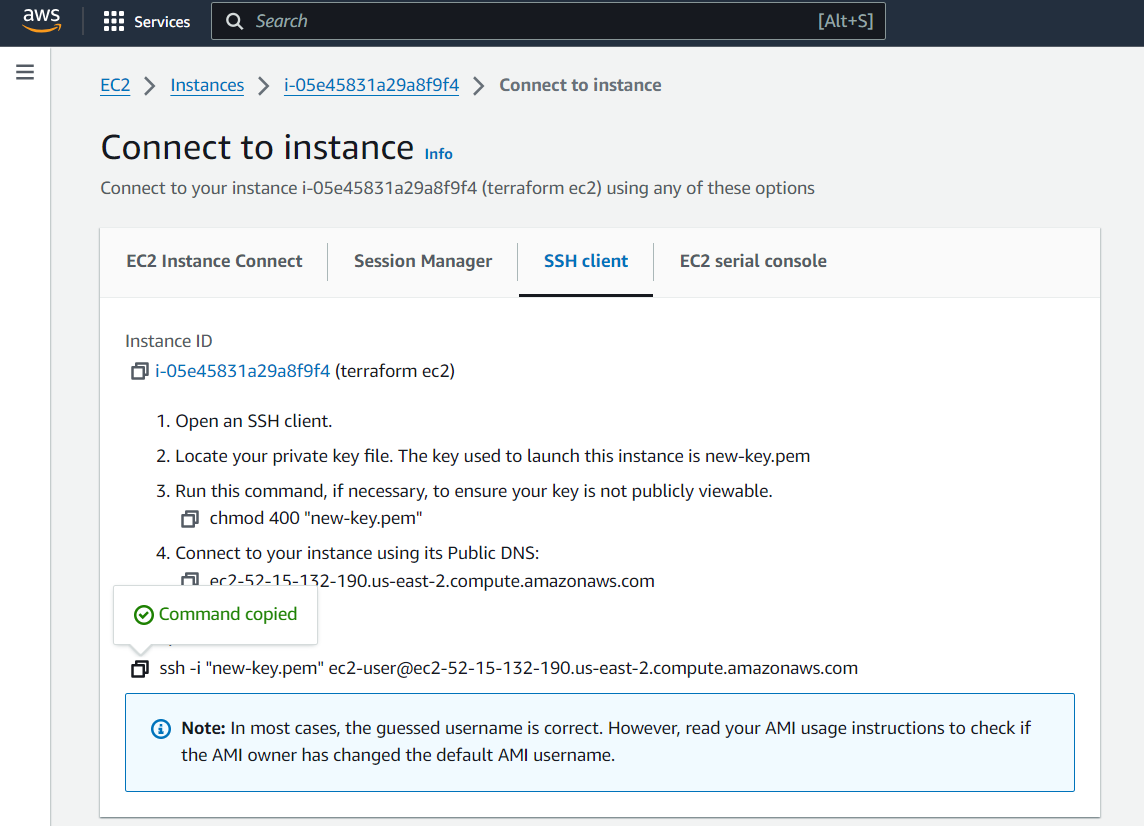
* Click on Launch instance, it will open the instance launch panel.



* You have to give the name to the instance.
* Then select the amazon AMI and instance type as “t2 micro”.
* Select the existing key pair if you have, if not create a key pair.



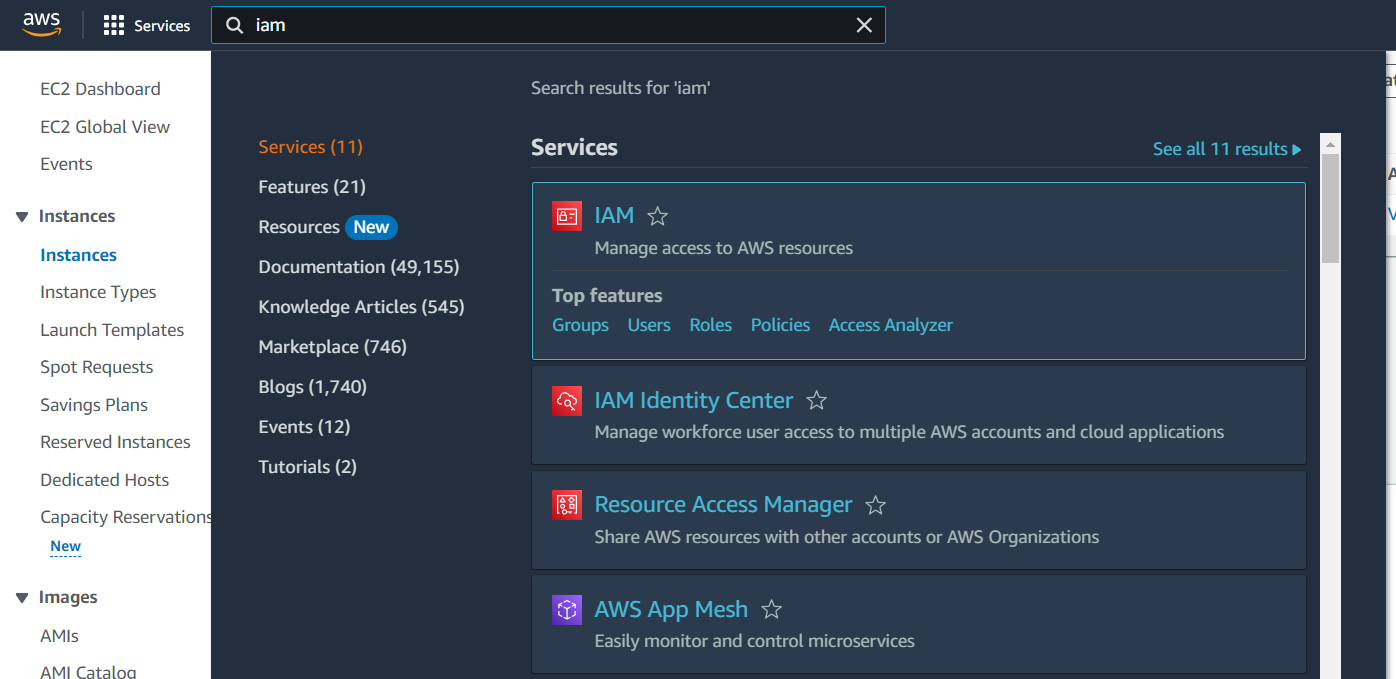
* Leave network setting as default.
* And click on launch instance.
* Not we have to connect instance to terminal through ssh.
* Open Git Bash and go the path where the key pair is stored.



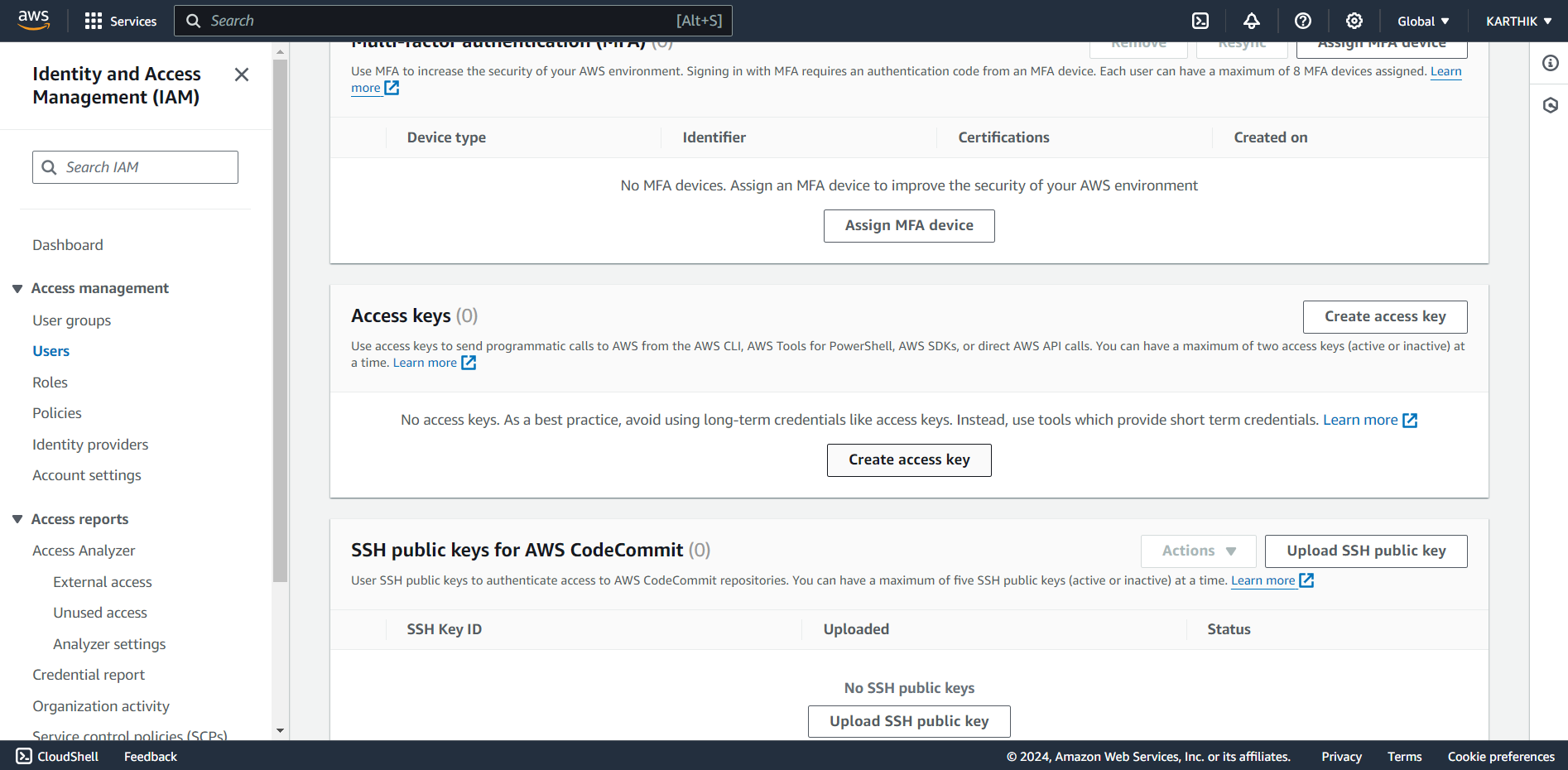
* Copy the ssh client and past and run in the terminal.
* The instance is connected to terminal.

**CREATING ACCESS AND SECURITY KEY FOR USER:**

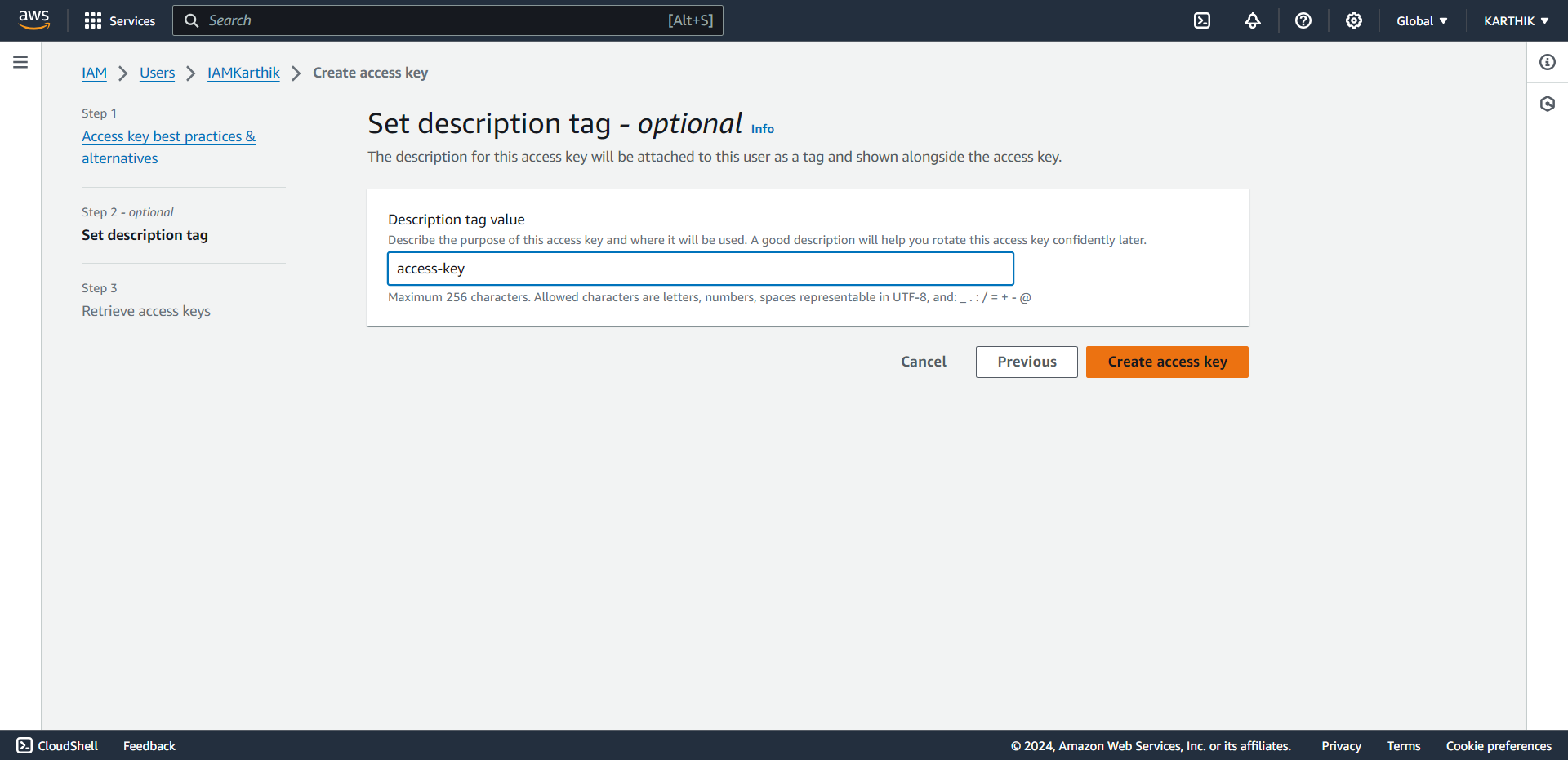
* I Navigated to the IAM services in my console.
* And goes into user I have.



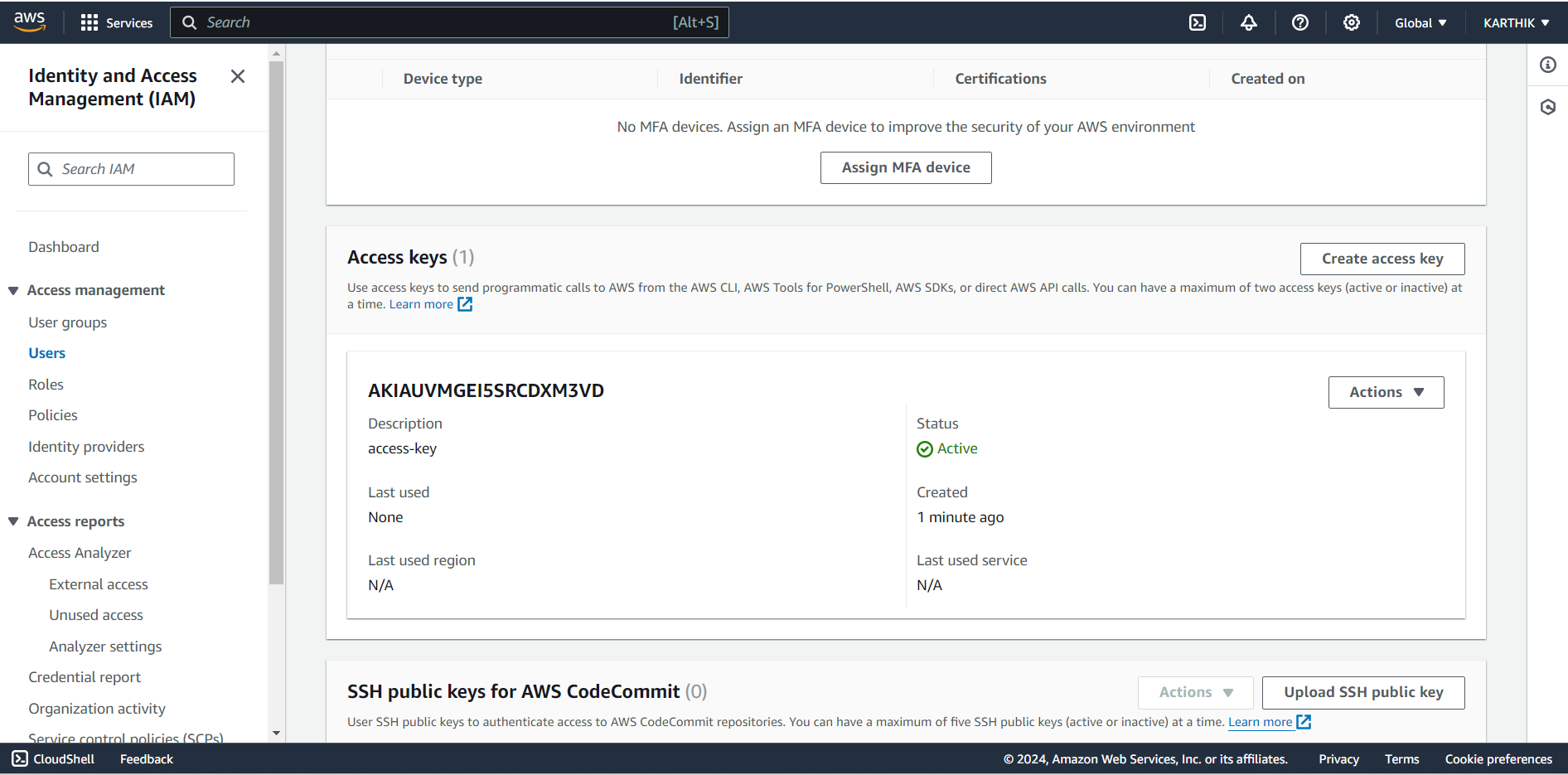
* Go into security credentials scroll down you will get Access key section, in that click on create access key.



* You will open a space there i selected “cli”, and click on next.
* In next section I given name to the access key.



* Click on create access key.
* The access key is created successfully,
* There you will get the user name and password.
* You have to store those two, that will used in the aws configure.
* That user name and password will not get again if you closed.

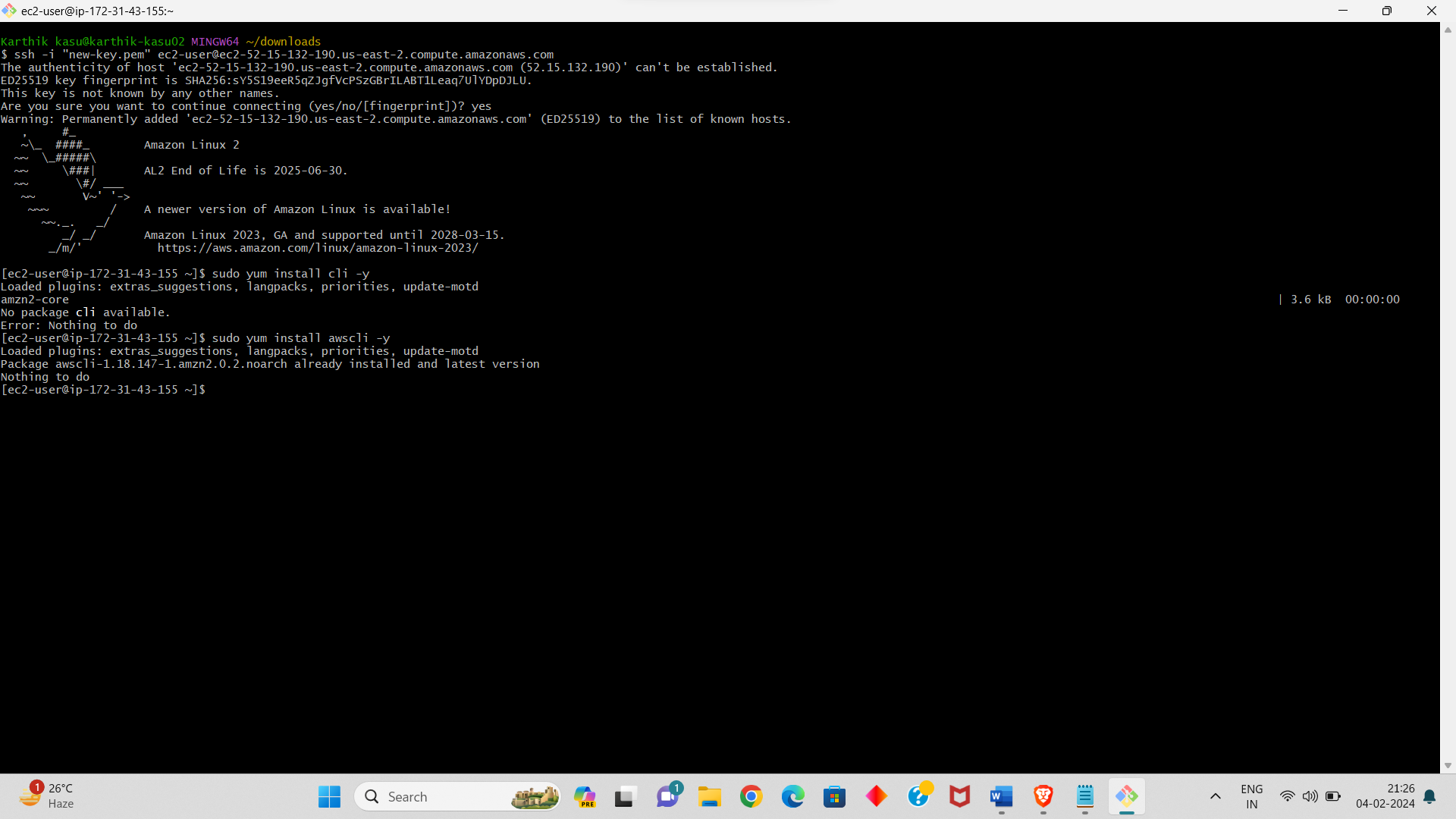


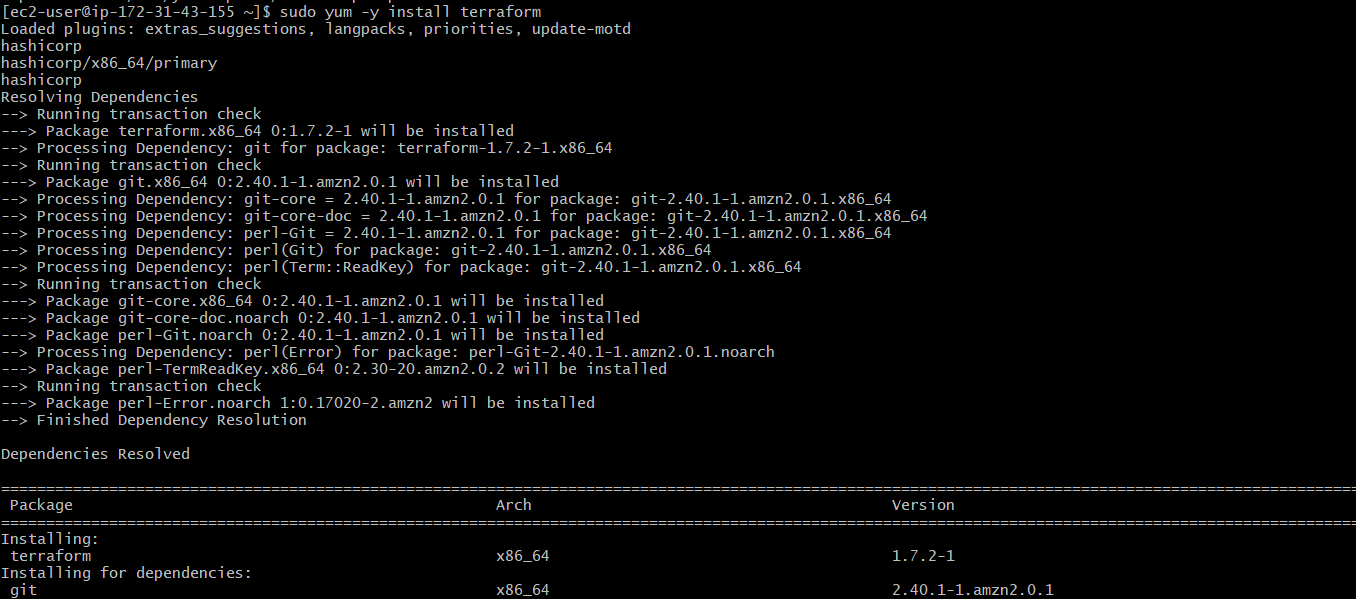
* Here you can see that the access key is created and it is in active status.
* Now we will open the terminal and install “awscli” and terraform.
* Install CLI with command “sudo yum install awscli -y”
* Install terraform with the commands

“sudo yum install -y yum-utils shadow-utils”

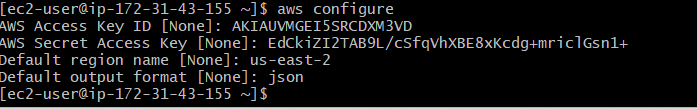
“sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo”

“sudo yum -y install terraform”





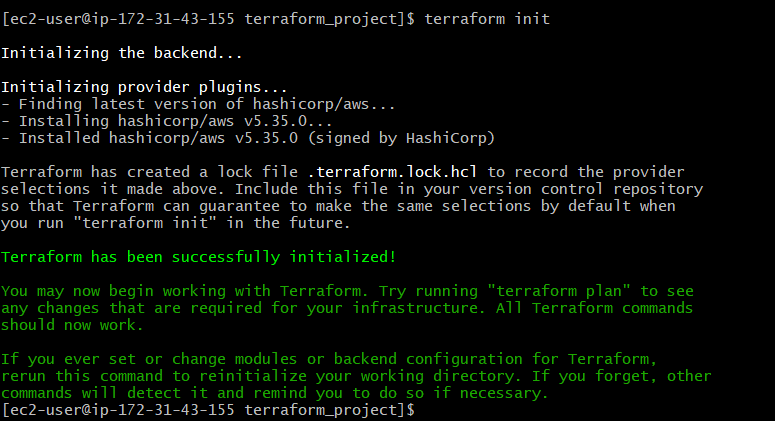
* We successfully installed the “terraform” and “awscli” in the terminal.
* Now we have to configure into our aws account with the access key and security key.
* For this use command “aws configure”.
* First, we will give access key and then security key and then region we want to use.



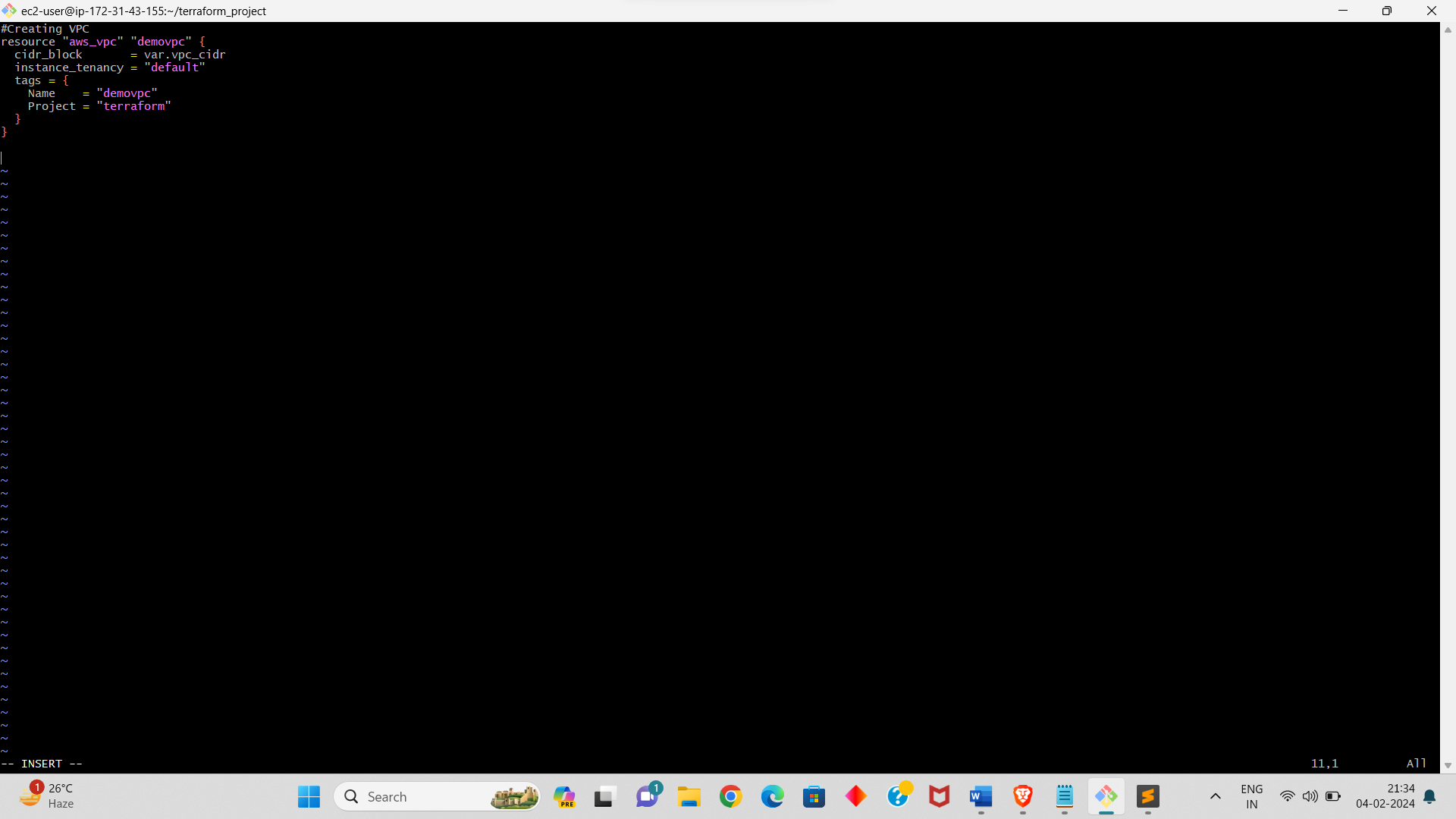
* We have to create each file for the each network a directory using “mkdir <name of the directory>”
* Go into the directory using “cd <name of the directory>”
* Now we have to create a **vpc.tf**  named file for VPC.



* Now initialize the terraform using “**terraform init**”.

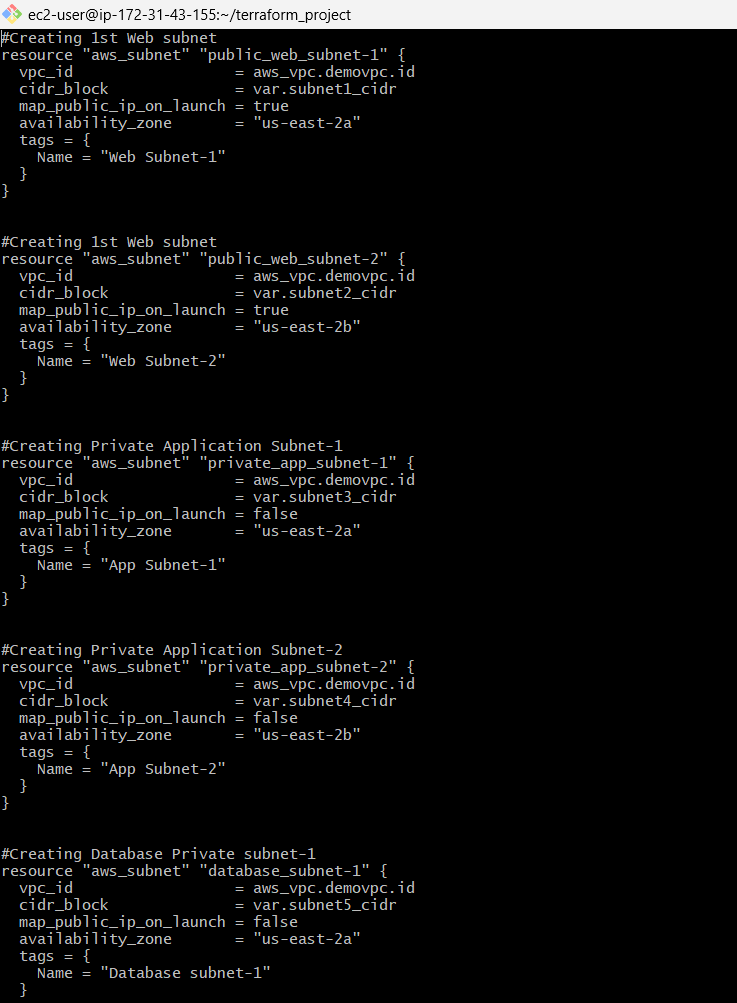


* Open the **vpc.tf** and write the below script for creating the VPC in the file.



* Now we have to format the file using “**terraform fmt**” this is done to convert the script into canonical format and style.
* And validate the file using “**terraform validate**”.
* We have to run these two commands after creating each terraform file, this will give the error if we done anything in the script.
* Now create a “**subnet.tf”** named file and write the script for creating SUBNETS.
* We have to write the script for creating 6 subnets.
* In that 2 for public subnets and 4 for private subnets, this 2 for web and another 4 for application and database of each 2.
* And attach the subnets for VPC.
* Give the CIDR rages for each subnet without overlapping.

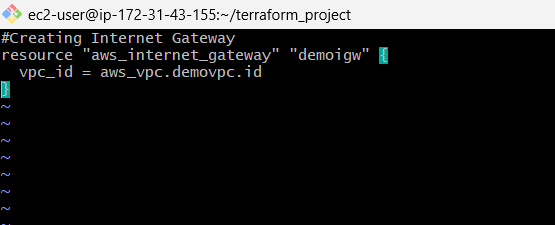
And give the availability zone



* Create a “**variable.tf”** for the variable in the we are giving the CIDR ranges for vpc and subnets.



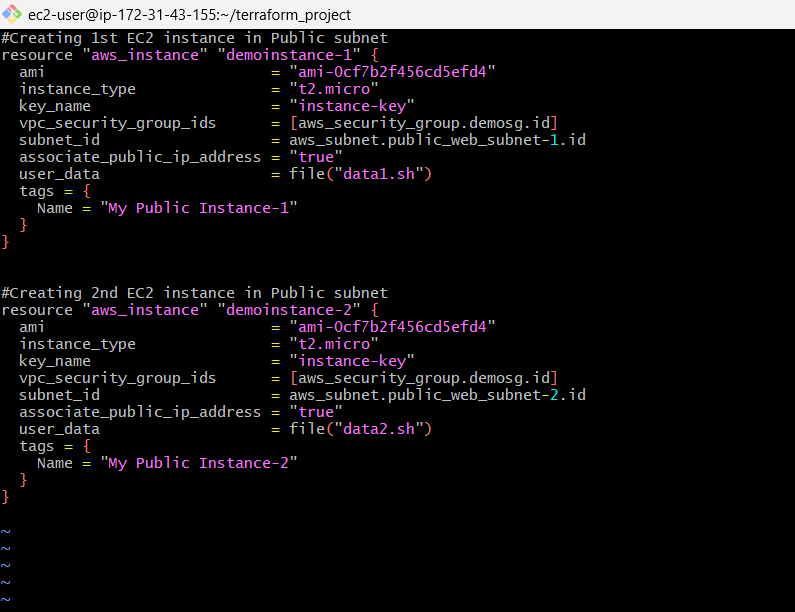
* Write the above script which have one vpc and 6 subnets.
* In that we clearly mentained the subnets and cidr ranges.
* next we have to create “**igw.tf**” file for creating INTERNET GATEWAY .



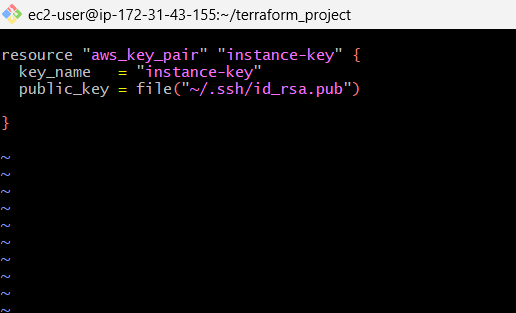
* in that we mentained the script for name and vpc id.
* After creating each file, we have to run the format command and validate command to check the errors.
* Create a “**routetable.tf”** file for creating ROUTE TABLE.



* The script is given above, in that script we give the script for creating route table and associating the route to subnets.
* Now create “**ec2.tf**”named file for creating EC2 INSTANCES.



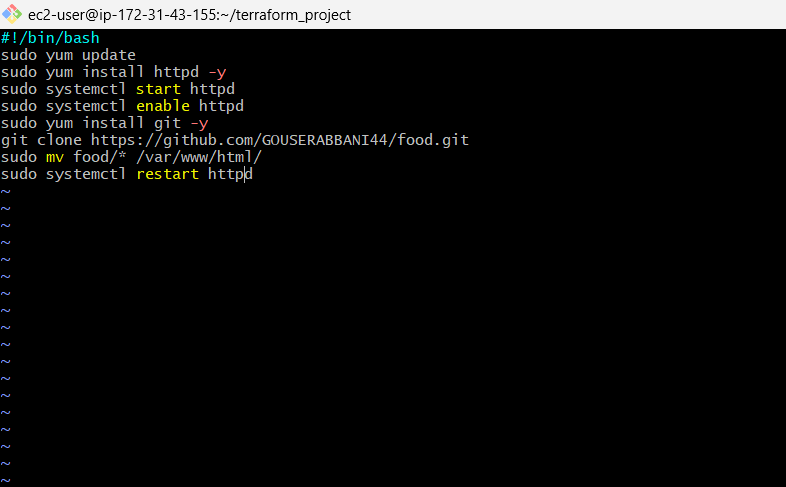
* In that we have to provide AMI id, instance type, vpc id, subnet id, user data, etc.,
* You can copy the script above for creating ec2 instances.
* Next we create a “**keypair.tf**” named file for path for the key pair.



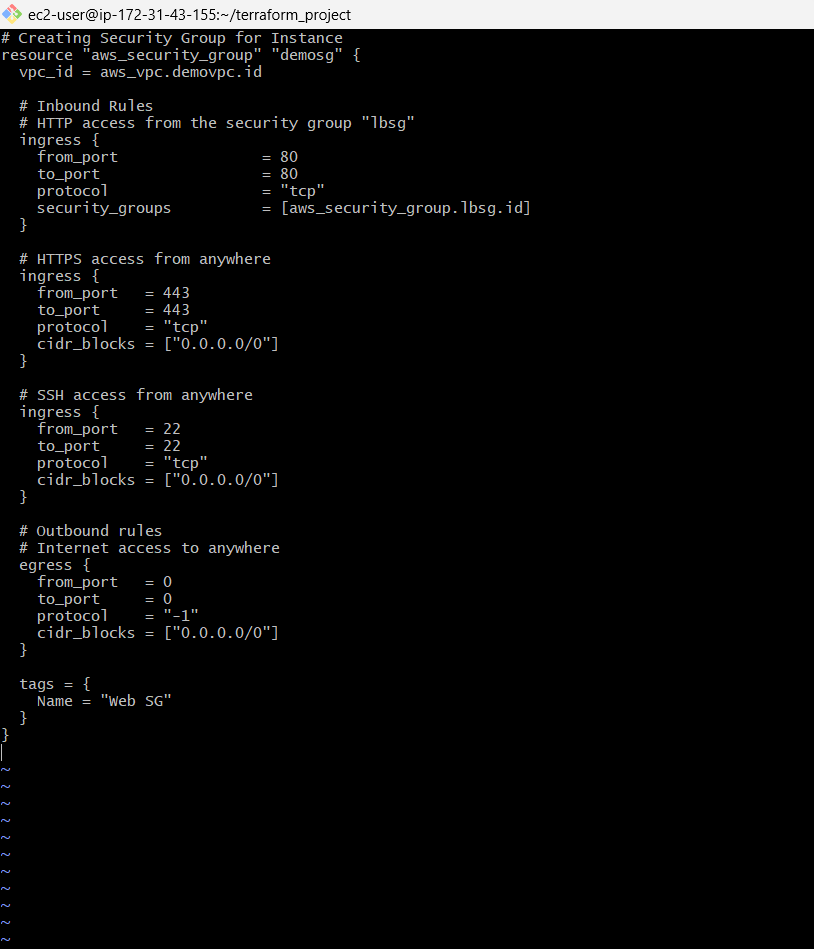
* In that file we mention the public key file path, which we created the key pair in the terminal using command

“**ssh-keygen -t rsa -b 2048 -f instance-key**”.

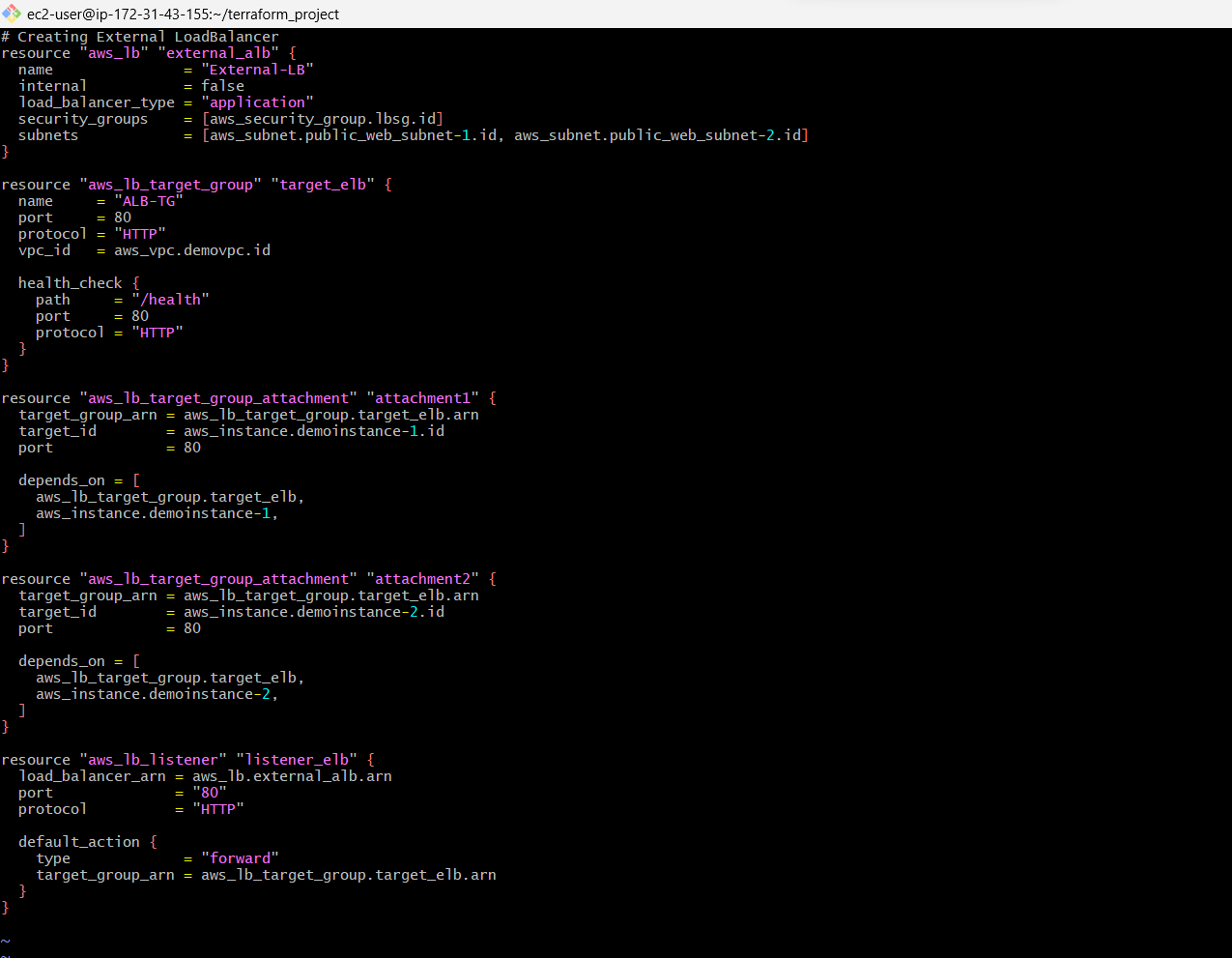
* Now we have to create a file “data1.sh”.



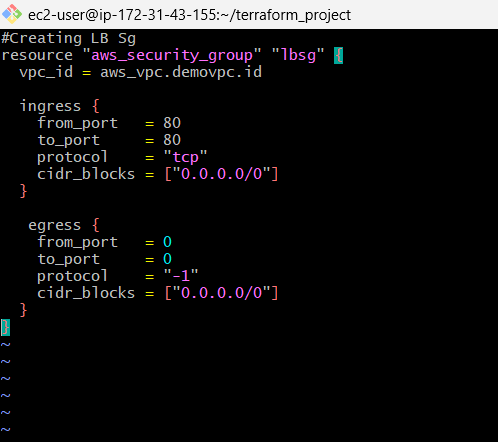
* In that file we have to write the commands that we want execute our 1st instances we created.
* The script is about installing httpd and git, start and enable the httpd, and git clone.
* Next we have to create the “data2.sh” file for the 2nd instance.
* In that file we have to write the commands that we want execute our 2nd instances we created.
* Now create a terraform file for creating SECURITY GROUP of our instance. File name “**ec2sg.tf**”.



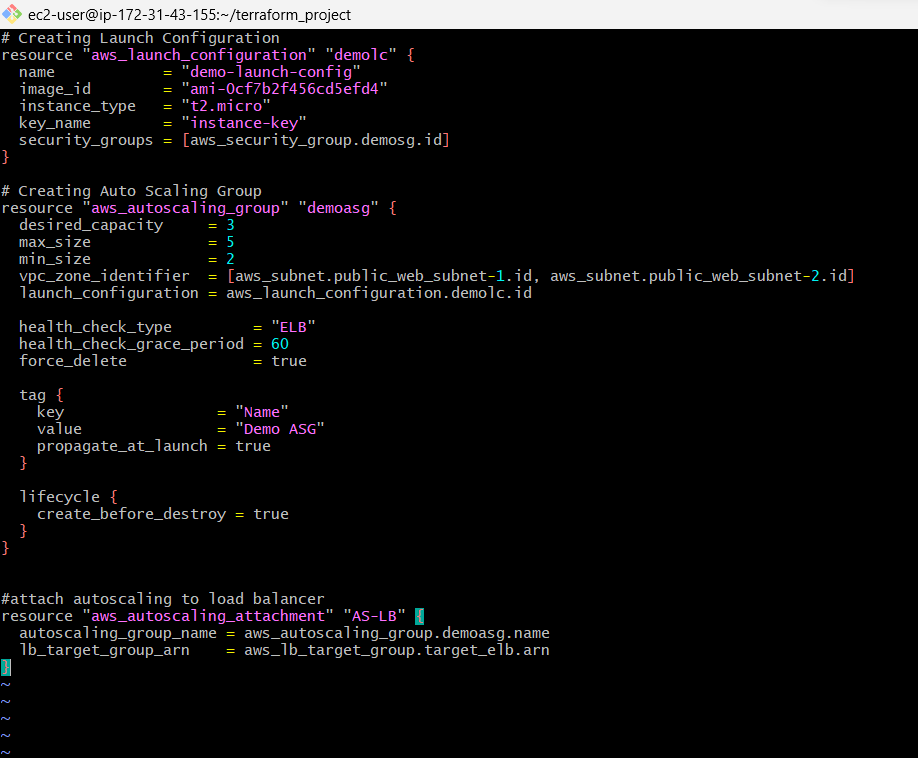
* In that file we have to give scripts for the inbound and out bound rules.
* And give the form and to ports and the cidr blocks also.
* Next we have to create a terraform file “**elb.tf**” for creating LOAD BALANCE AND TARGET GROUP.
* In that we have to mention the attachments for target group and load balancer.
* And we have to create script for listener also.
* Give port Forword to the target group.



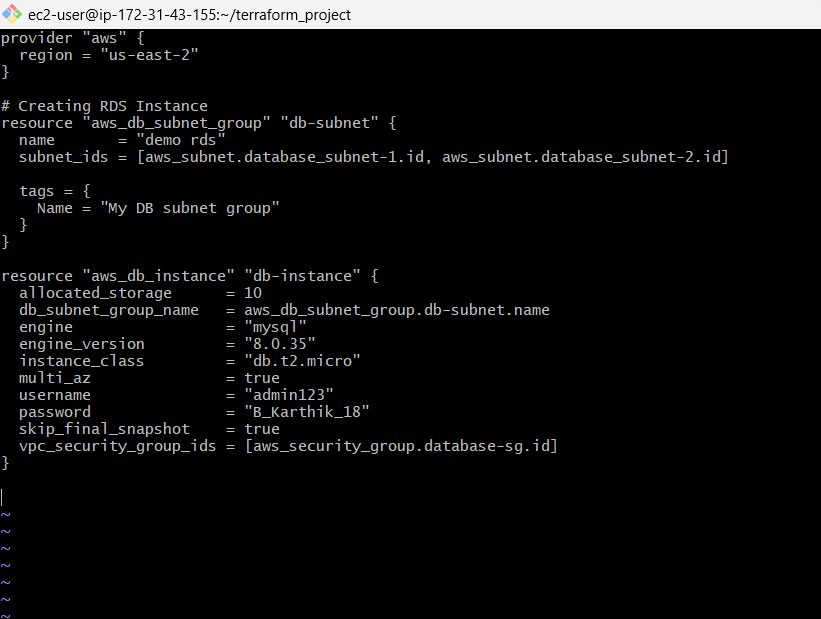
* Now create a “**lgsg.tf**” file for creating LOADBALANCER SECURITY GROUP.



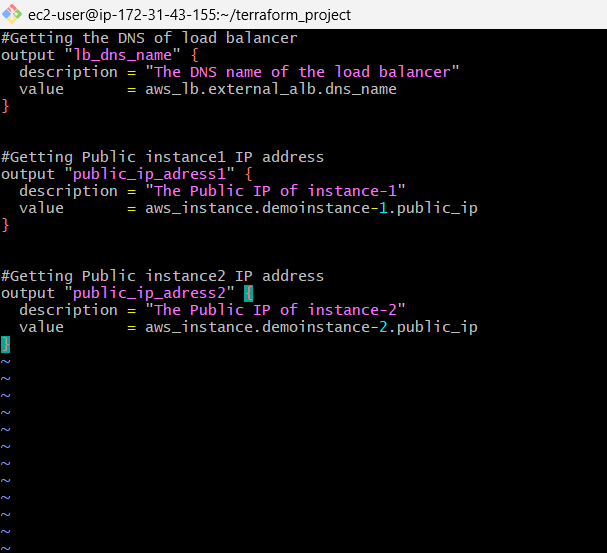
* Write the above script in that file.
* That file consists of the script of inbound and outbound rules of load balancer sg.
* Now create a terraform file for create AUTO SCALING AND LAUCH CONFIGURATION.
* In that we mention the script for creating LC with name, ami id, instance type, and security group.
* You have to write the below script for that.



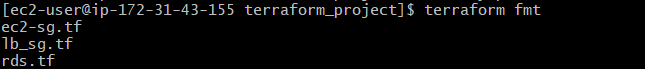
* We also added script for attaching auto scaling to load balancer also.
* And added the health check type, and health check grace period.
* Don’t forgot to rum the format and validate command to check the errors after create each terraform file.
* This helps you to at the final apply part.
* Now create a terraform file named “**rds.tf**” file for creating RDS instance and subnet for RDS.



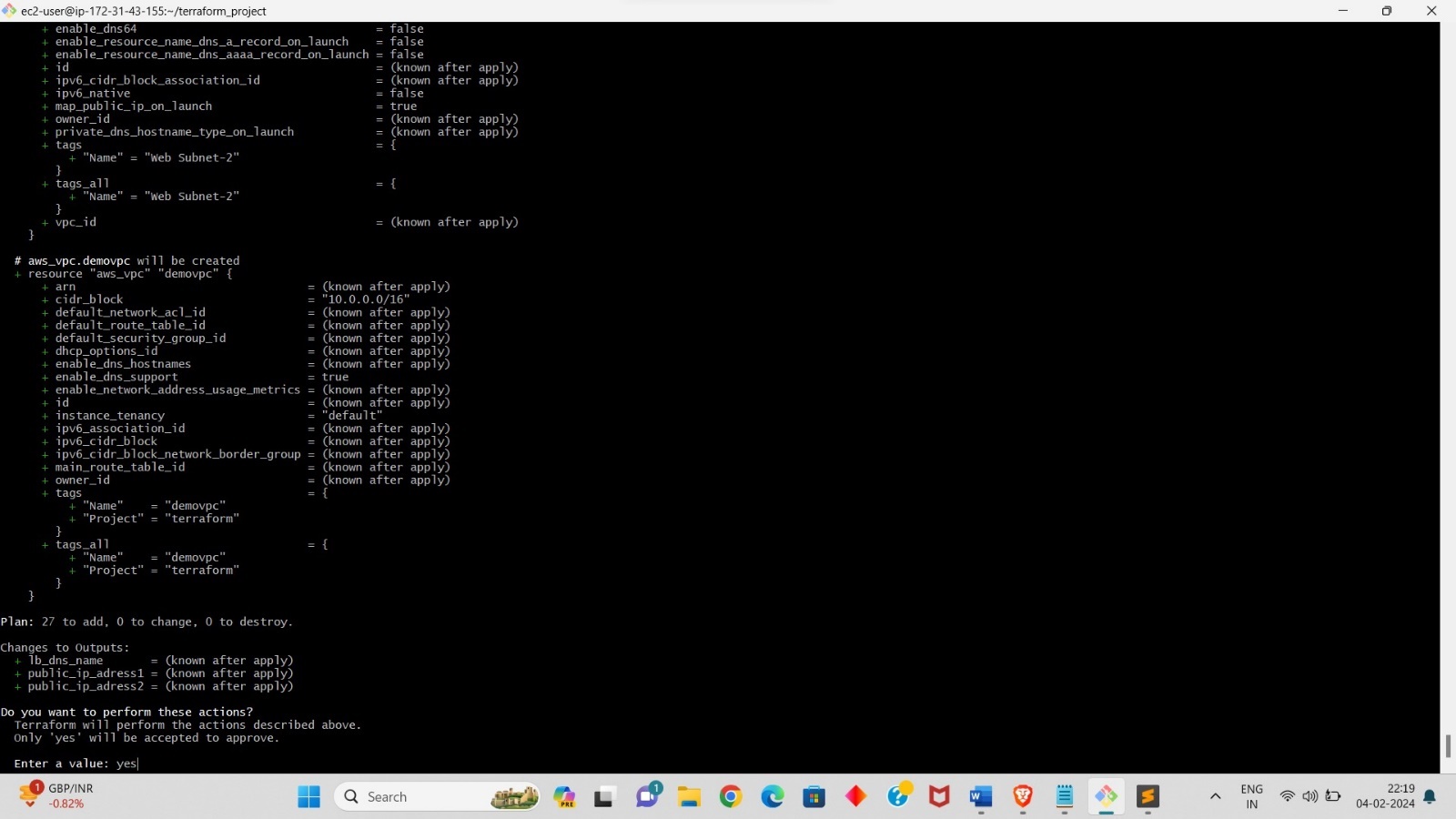
* The rds instance is created in a private subnet.
* In that we have to give the database engine, engine version.
* And give the username and password for our database.
* Store you username and password for opening in the terminal.
* Create a terraform file of “**outputs.tf**” this file is out outputs.



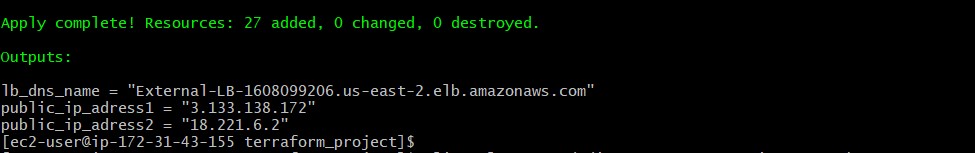
* Write the above script in the outputs.tf file.
* In this file we mention the output script for DNS of Load balancer, and public id of our two instances that we created.
* This file gives the outputs after we created the aws architecture.
* Now execute these commands “**terraform fmt**” to set all the script into canonical format and style.
* Run “**terraform validate**” command for validate the script and this command give the error if we did any.

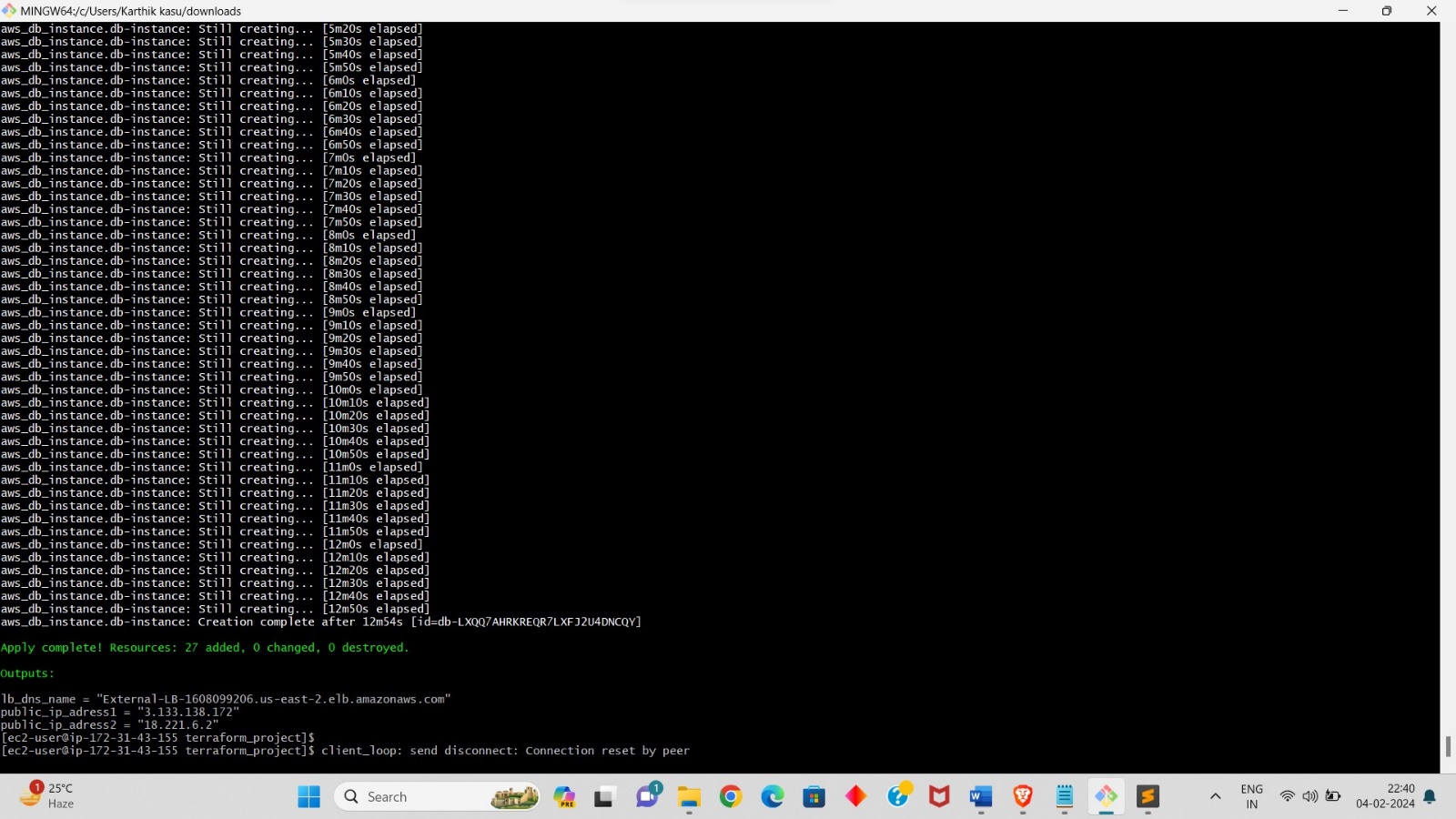


* Execute the command “**terraform plan**” to show what about to happen.
* When we execute the command “**terraform apply**”it starts creating the resources in your AWS account.

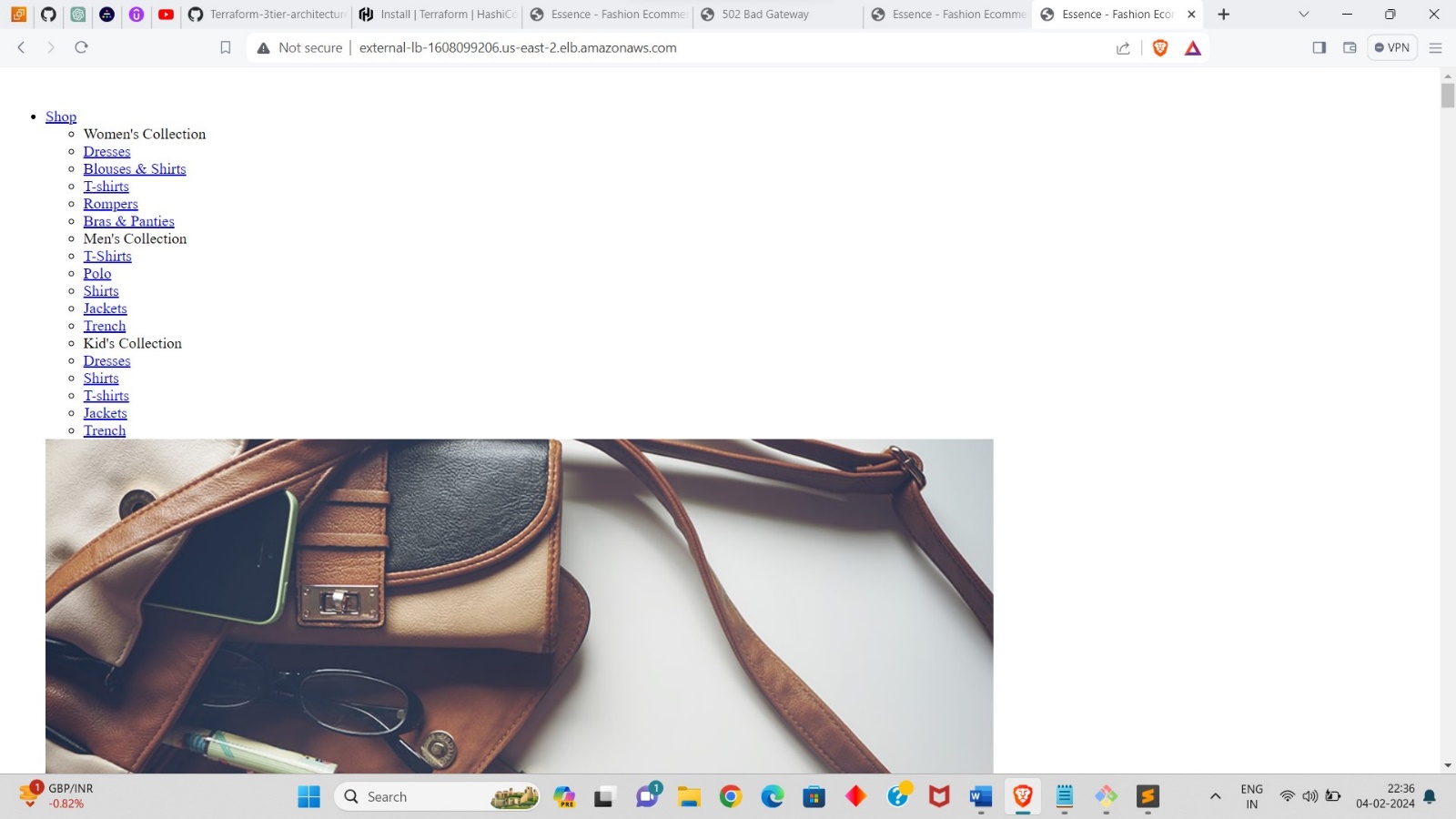


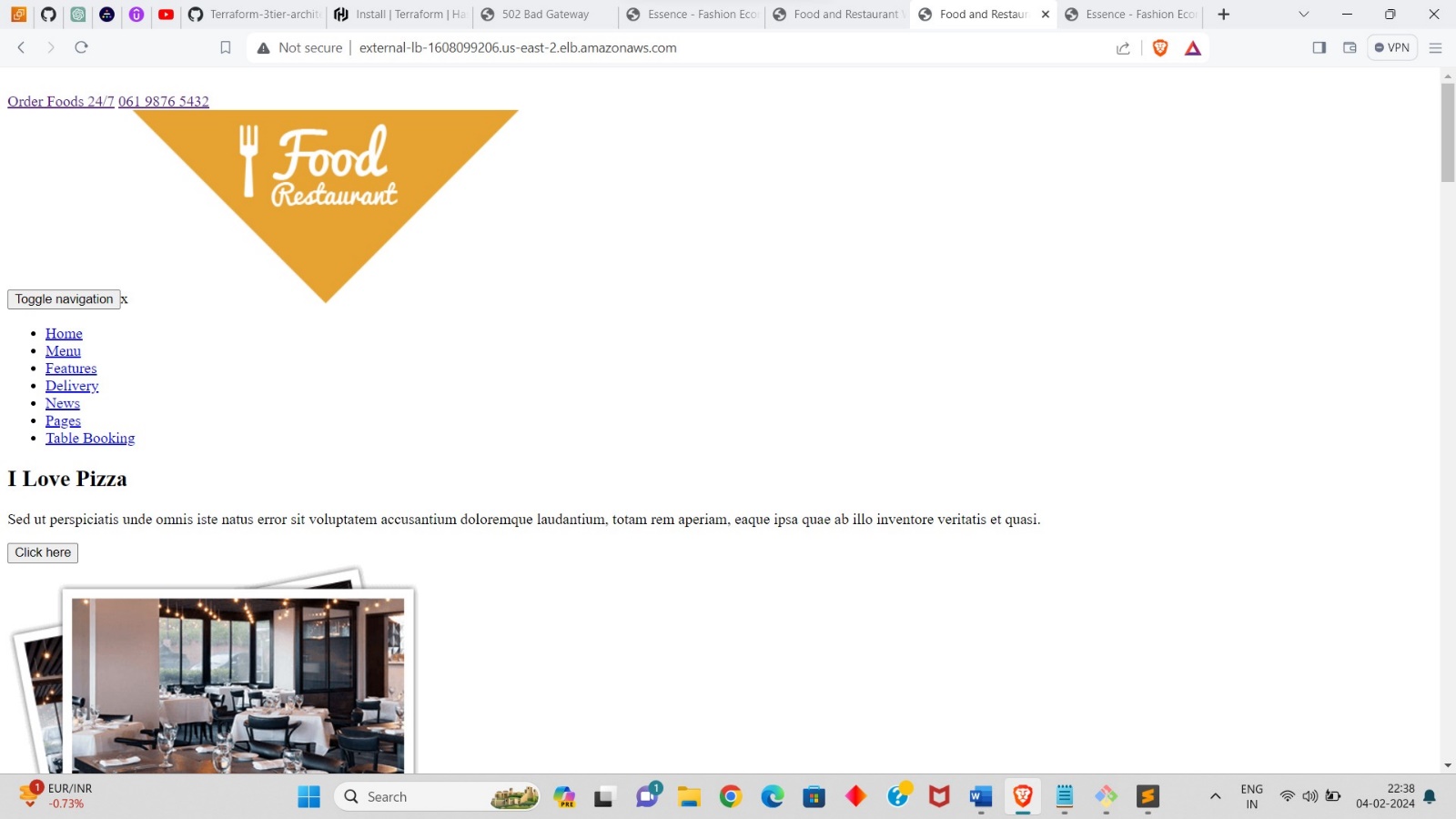
* It asks you permission whether to create or not.
* We have to type “yes” if you want to create.
* We have to wait for some time for creating our resources.
* After some time it will give you the outputs that we want in the outputs.
* And check in the aws account whether all our resources is created are not.





* Copy the DNS link of Load Balance given in the outputs and browse it in the browser to check the application are running are not.
* Click on refresh few times and you will see that the applications are shuffling one after other.

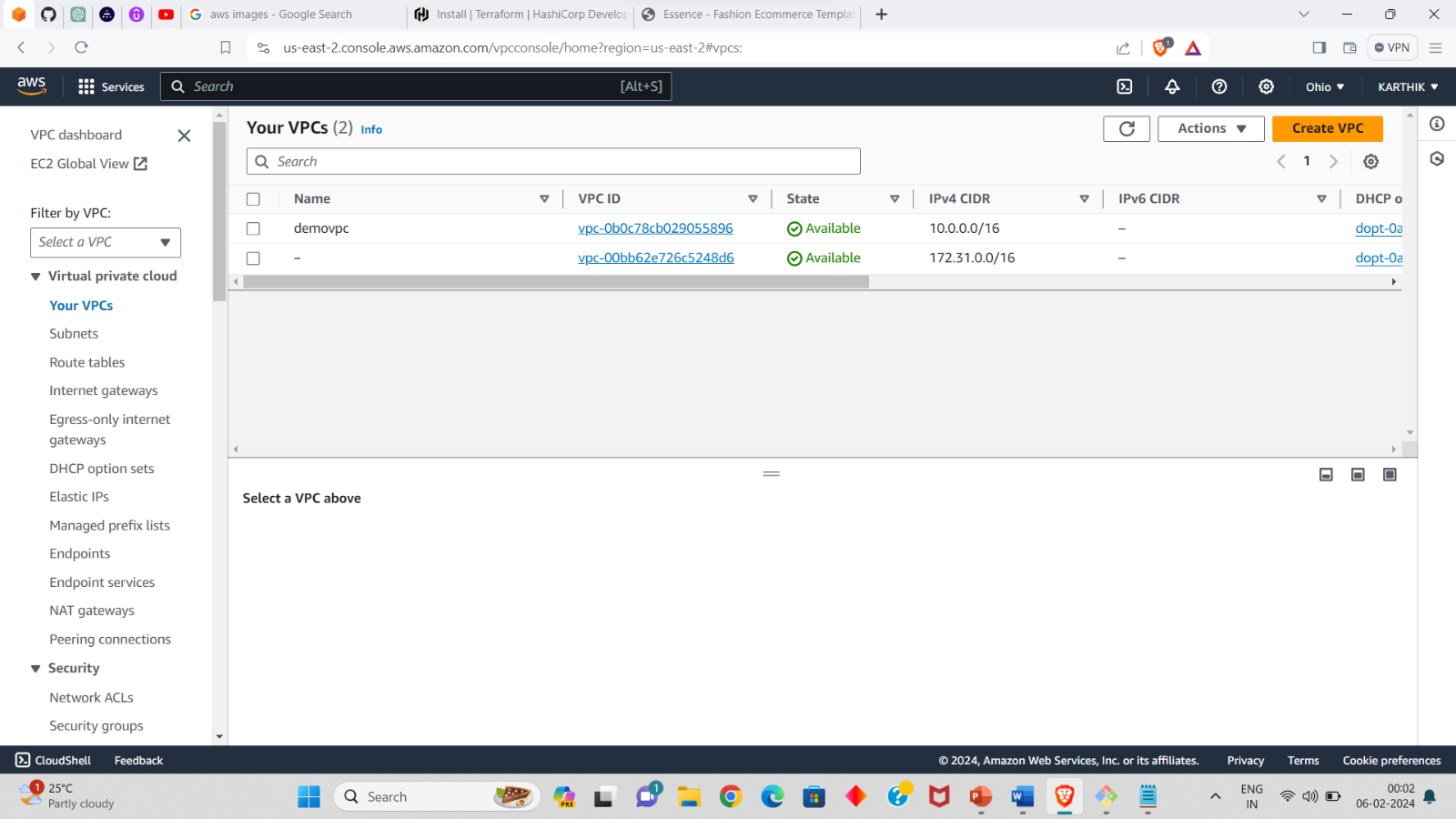




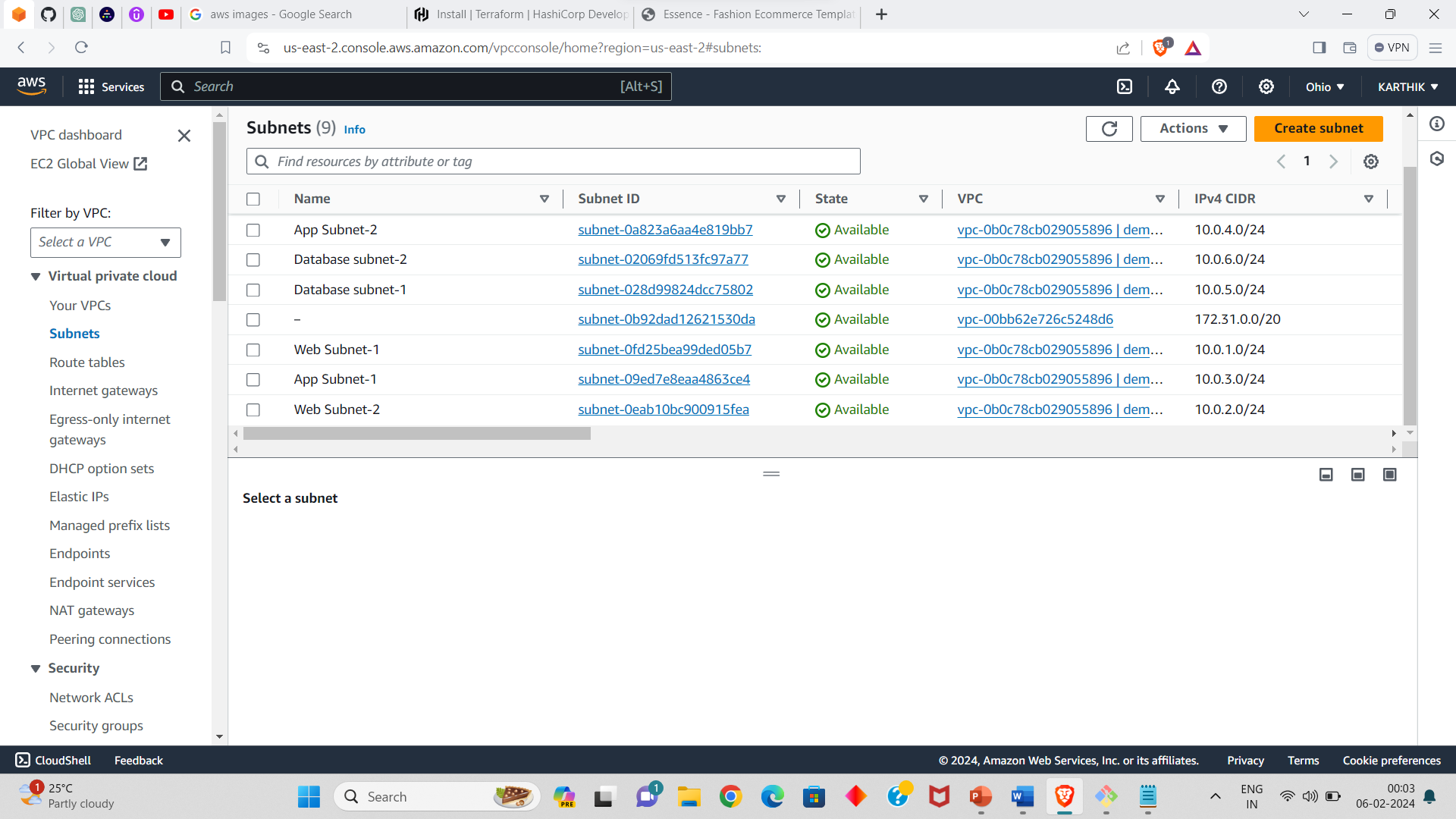
* By this our deploy of 3 tier architecture in aws is completed.
* Execute the command “**terraform destroy**” to destroy or delete your created resources.

RESOURCES THAT CREATED IN AWS:

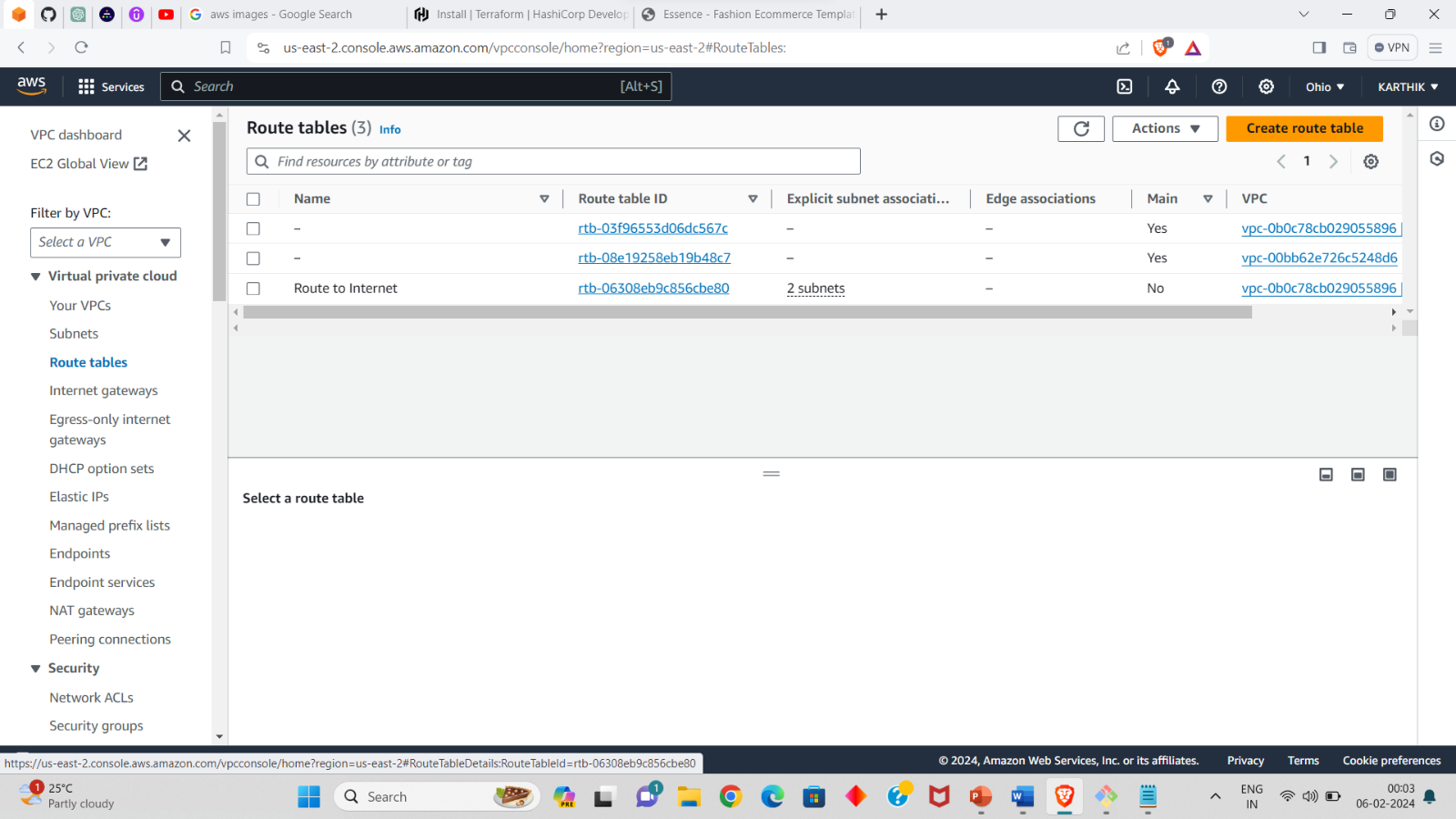
* DEMO NAMED VPC



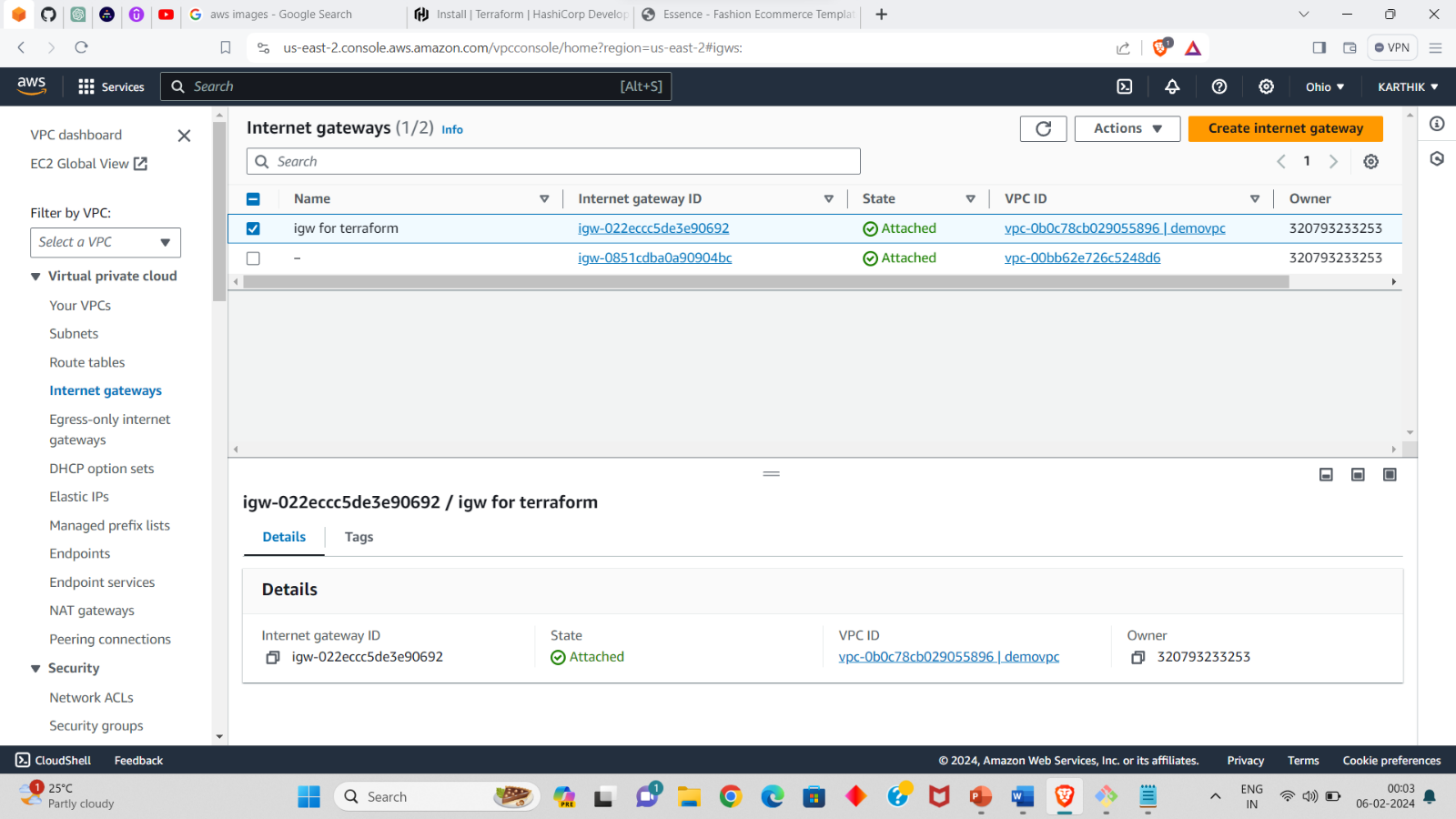
* 6 SUBNETS, 2 FOR EACH TIER:



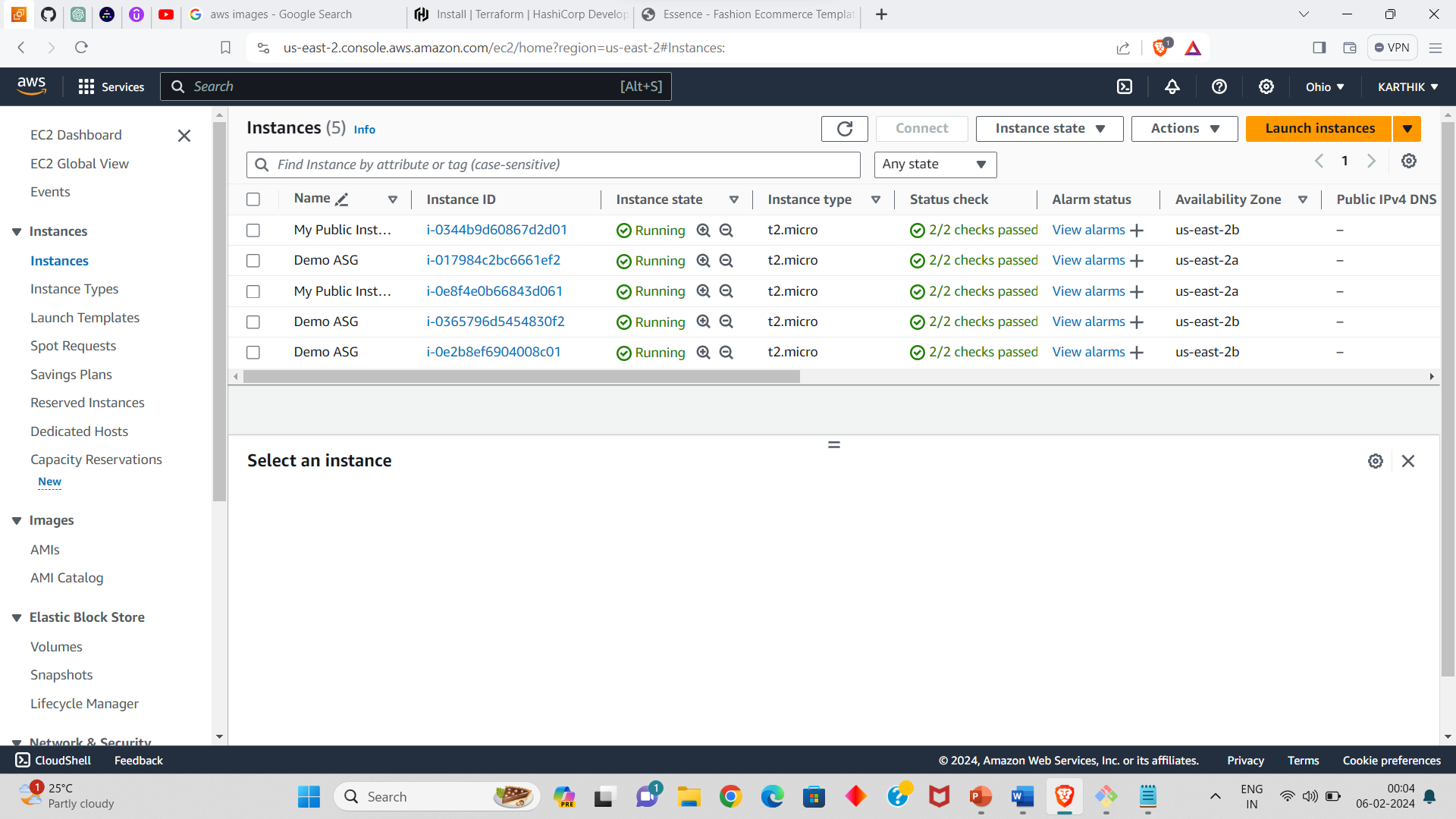
* ROUTE TABLE:



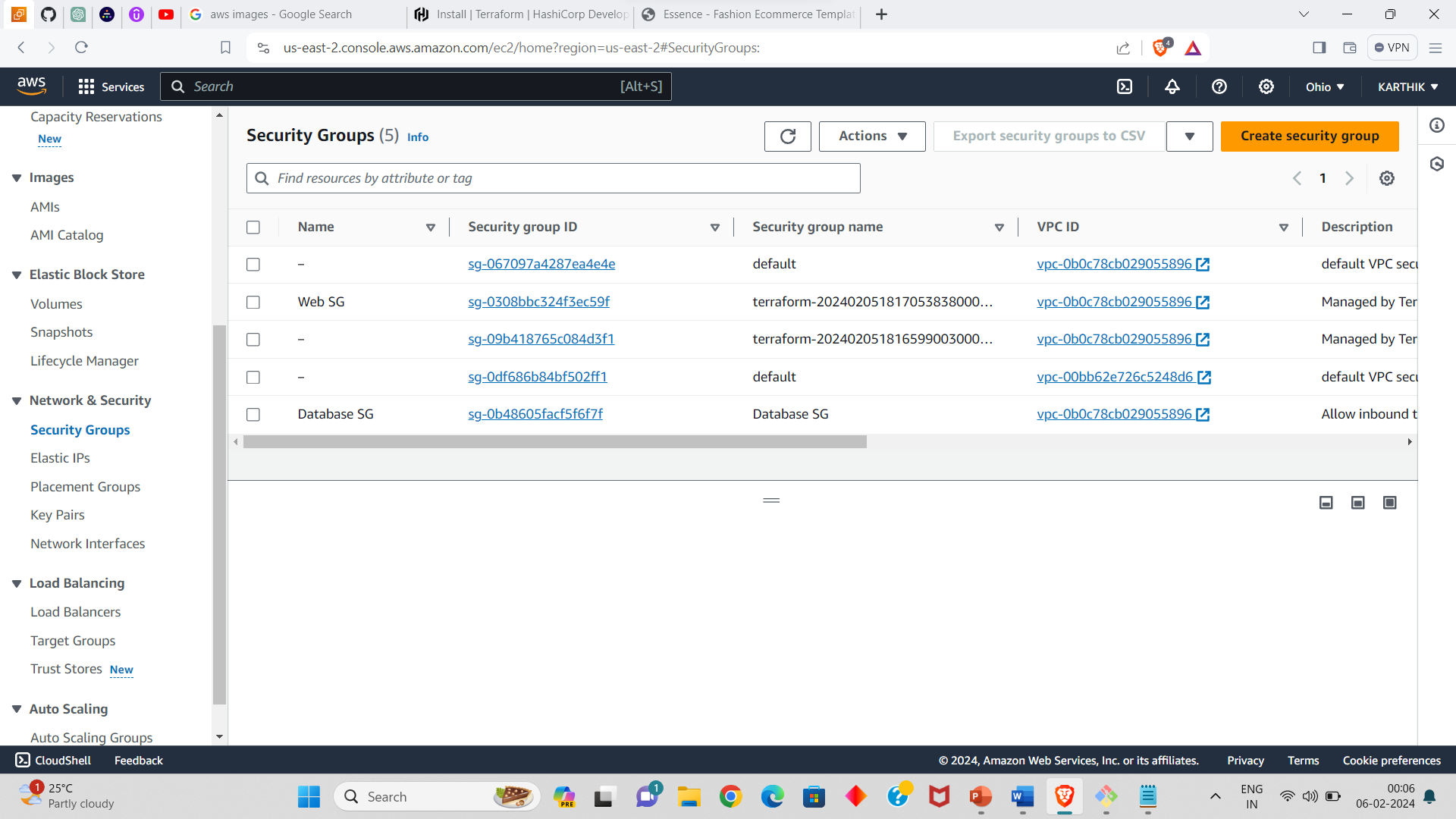
* INTERNET GATEWAY:



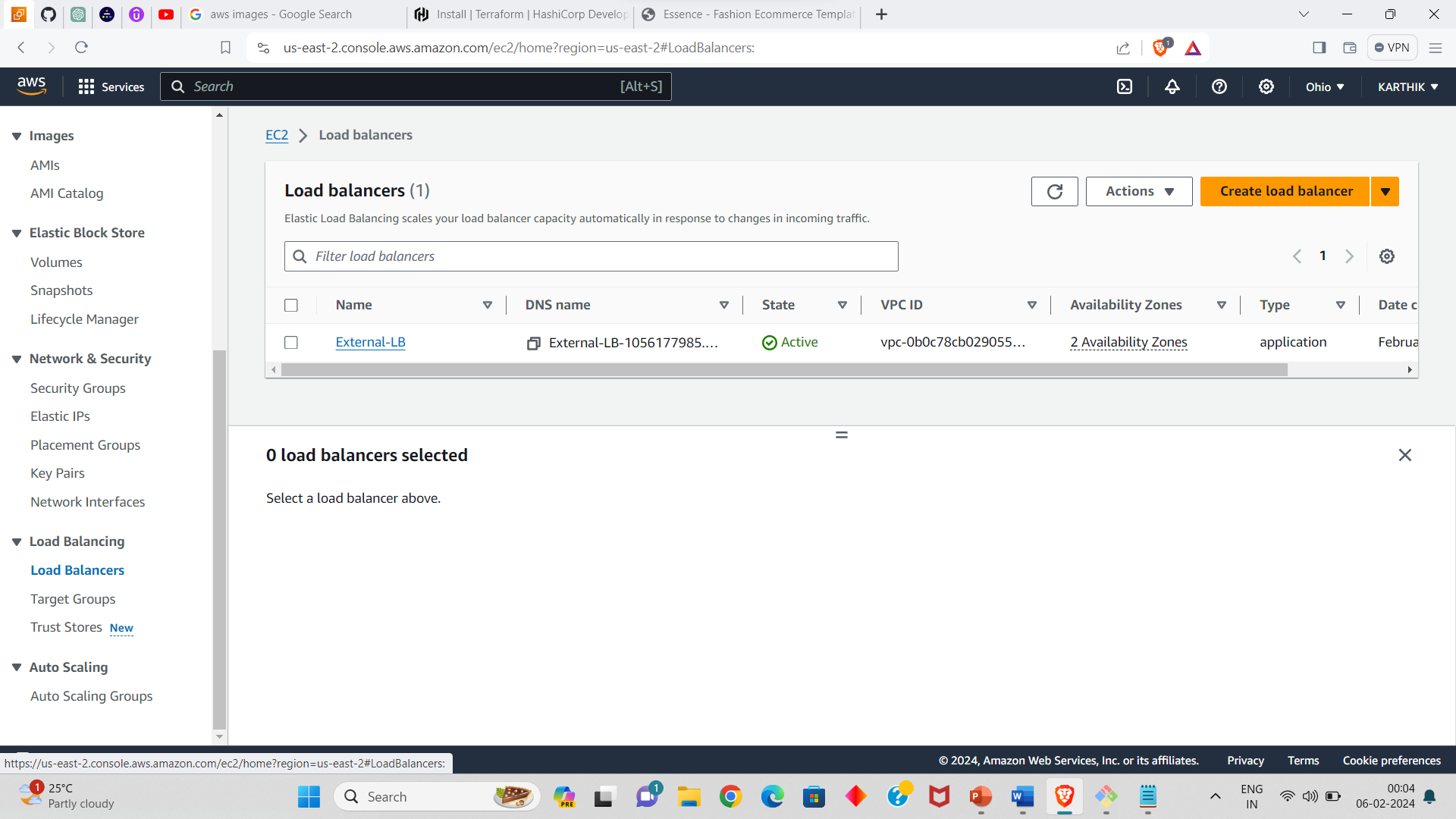
* INSTANCES:



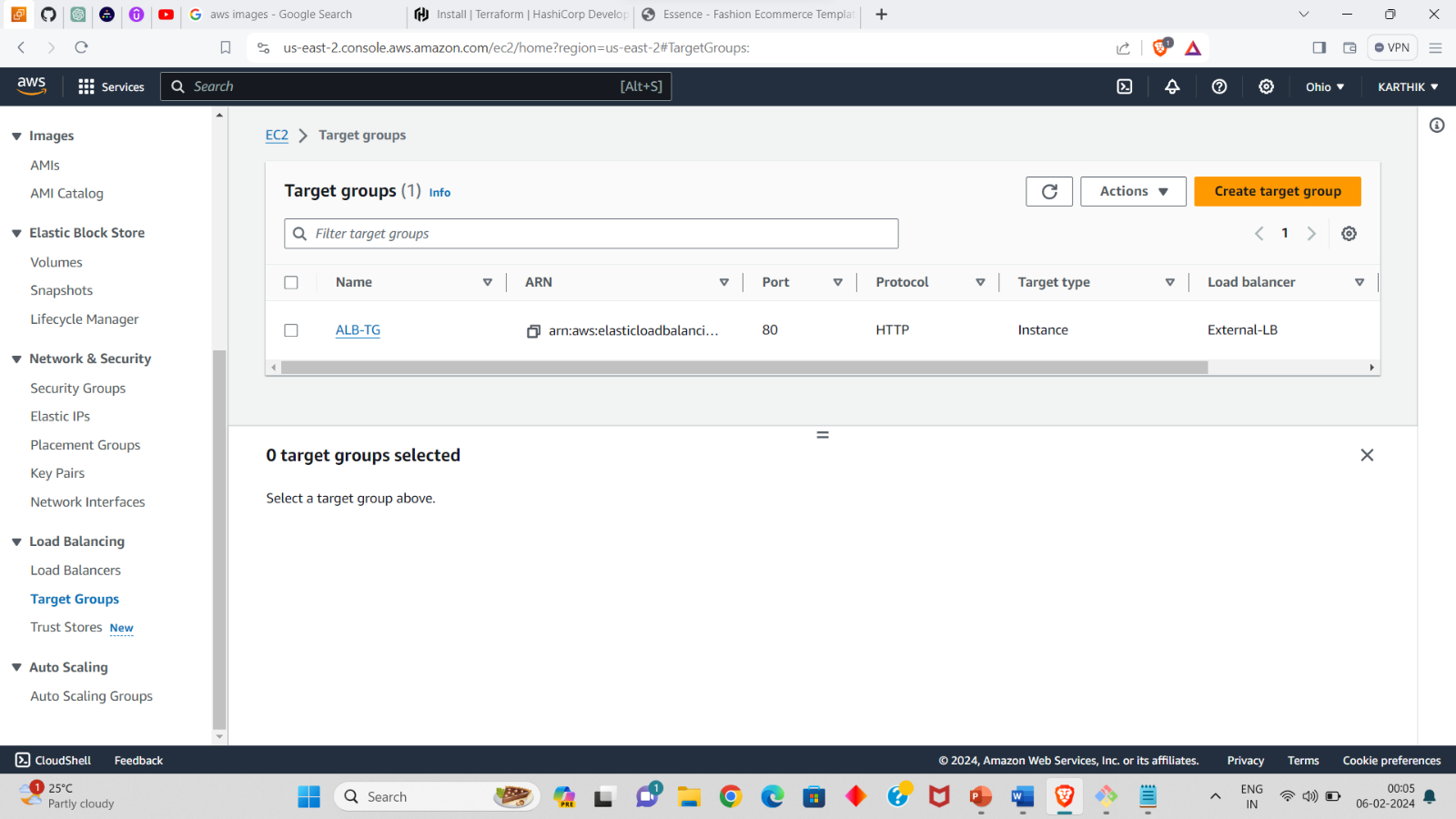
* SECUTIRY GORUPS:



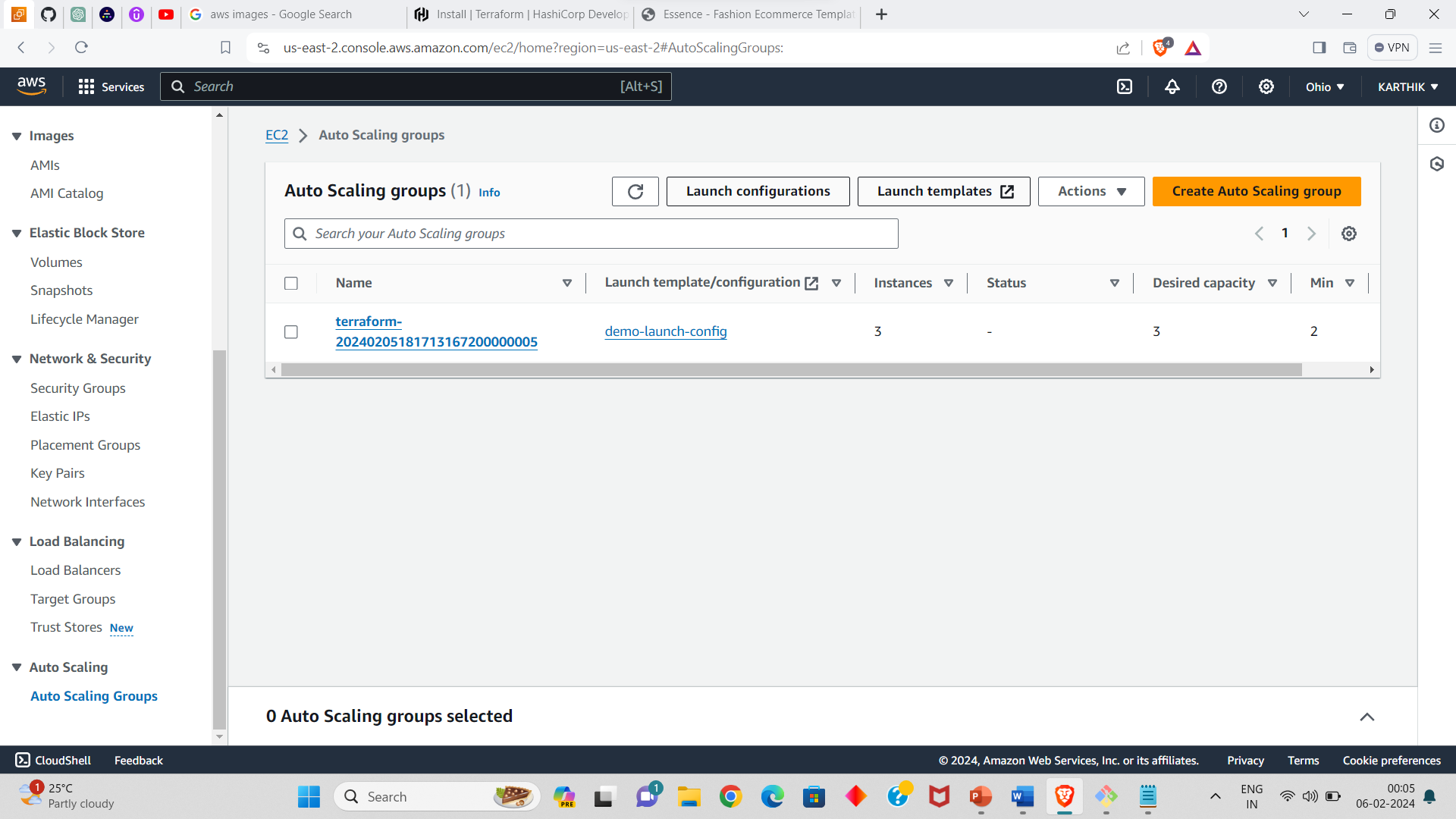
* LOAD BALANCER:



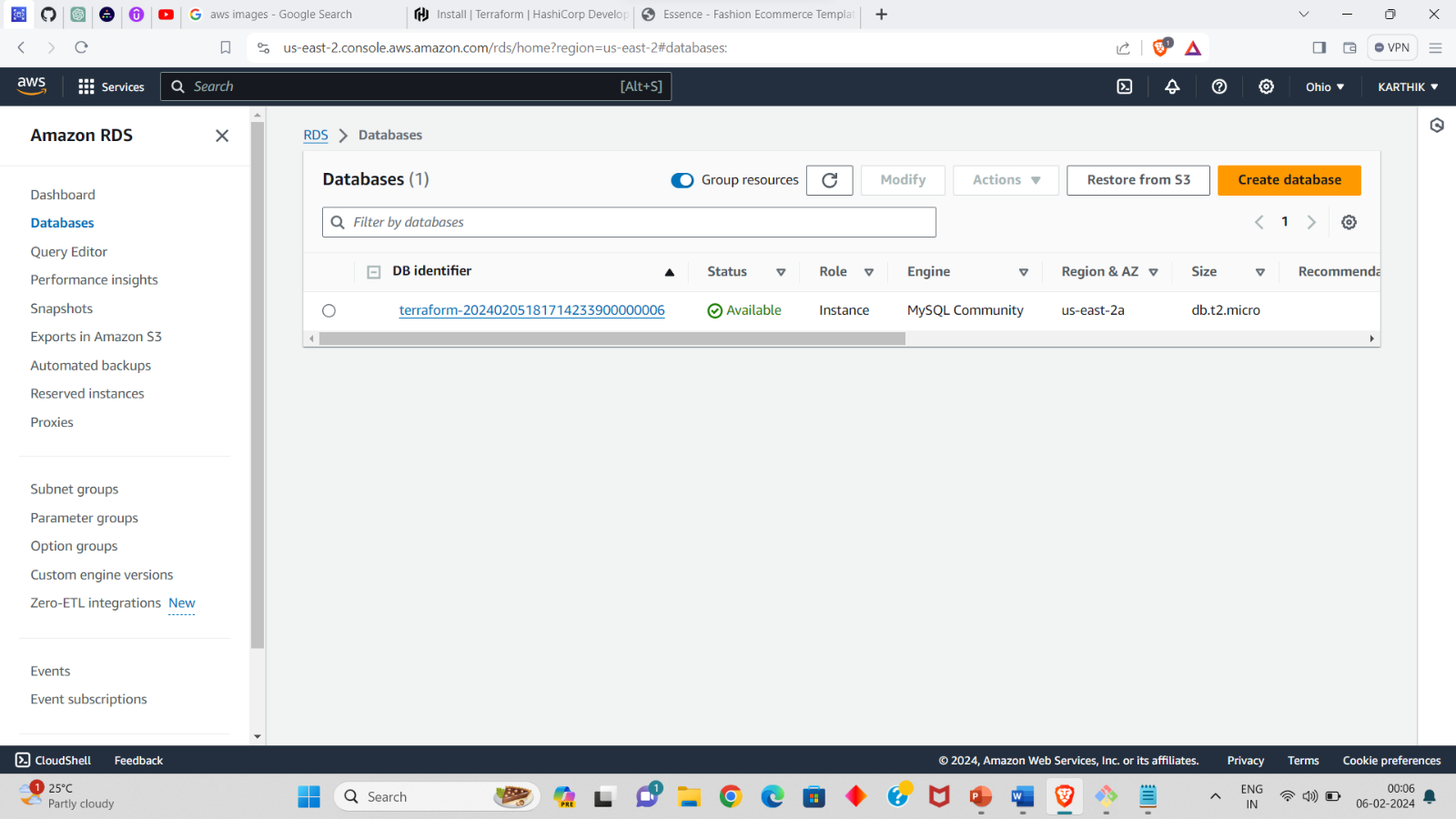
* TARGET GROUP:



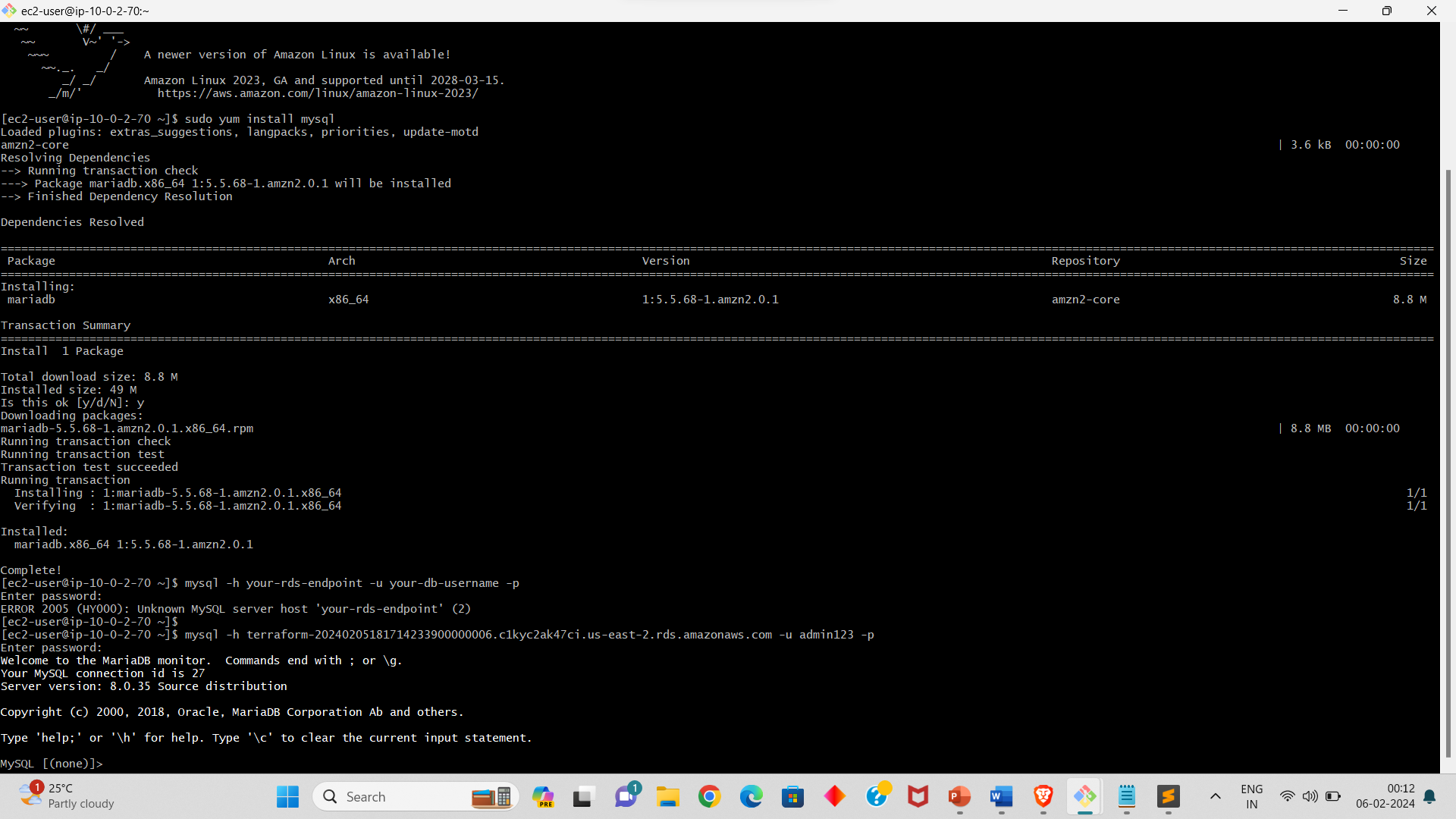
* AUTO SCALING GROUP:



* RDS:



* CONNECTED RDS TO EC2 INSTANCE:



RESOURCE:

* Here is the link of my GITHUB repository, you can get the scripts of the resources that we created above.

<https://github.com/KARTHIK0418/terraform_project.git>