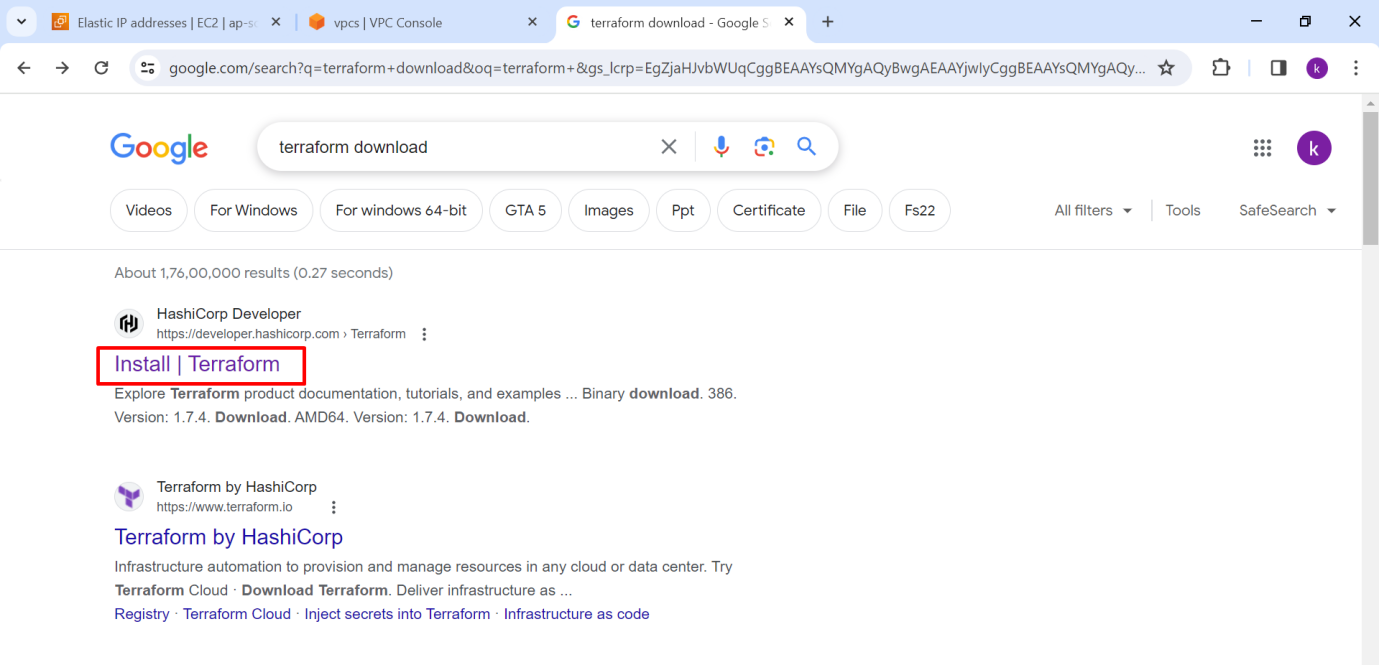
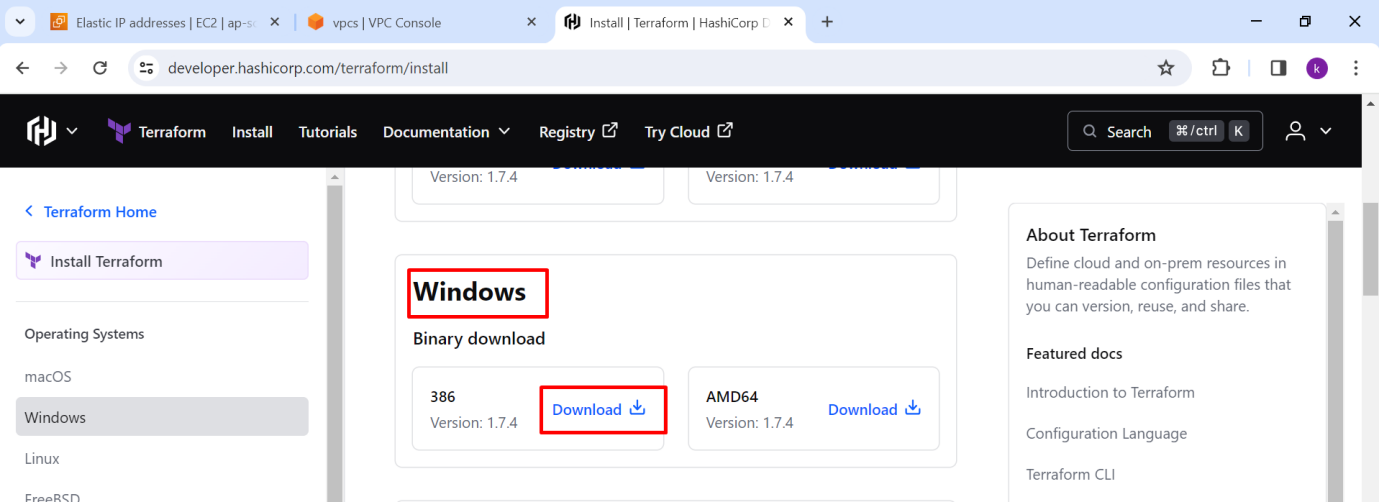
**Private server hosting (apache2) using Load balancer(AWS) with Terraform**

**Pre Requirements :-**

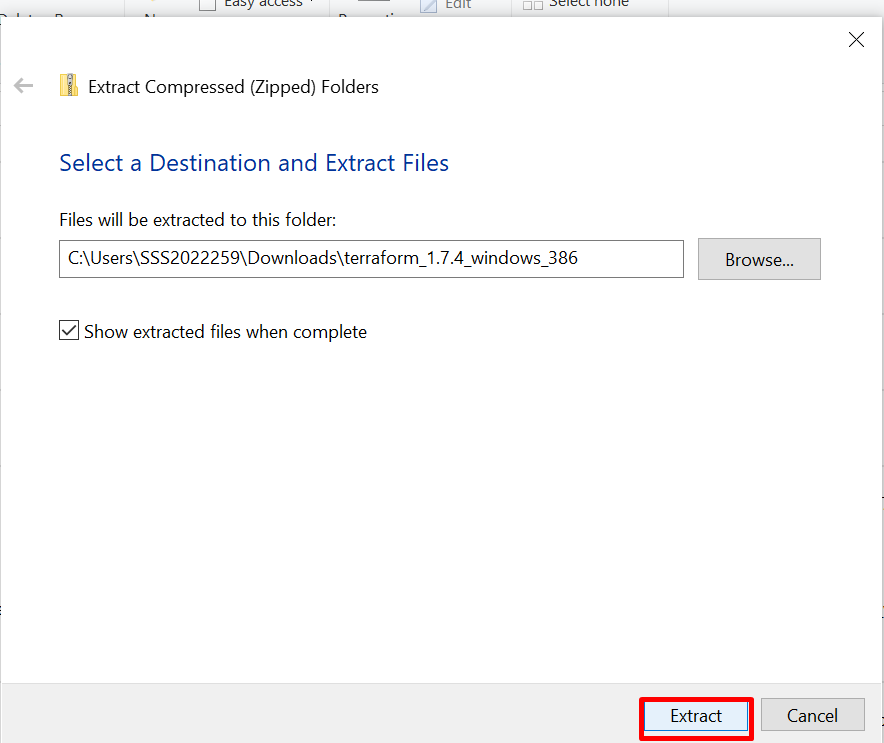
1. Install Terraform
2. AWS Account
3. AWS and Terraform configure with local server.

Install Terraform in local server (Windows)

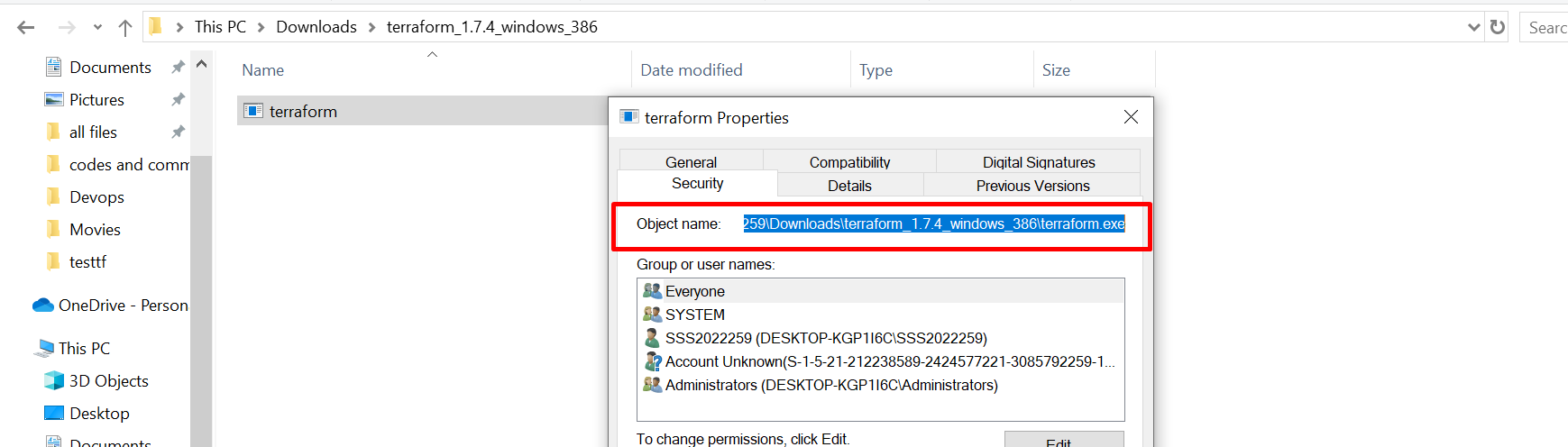




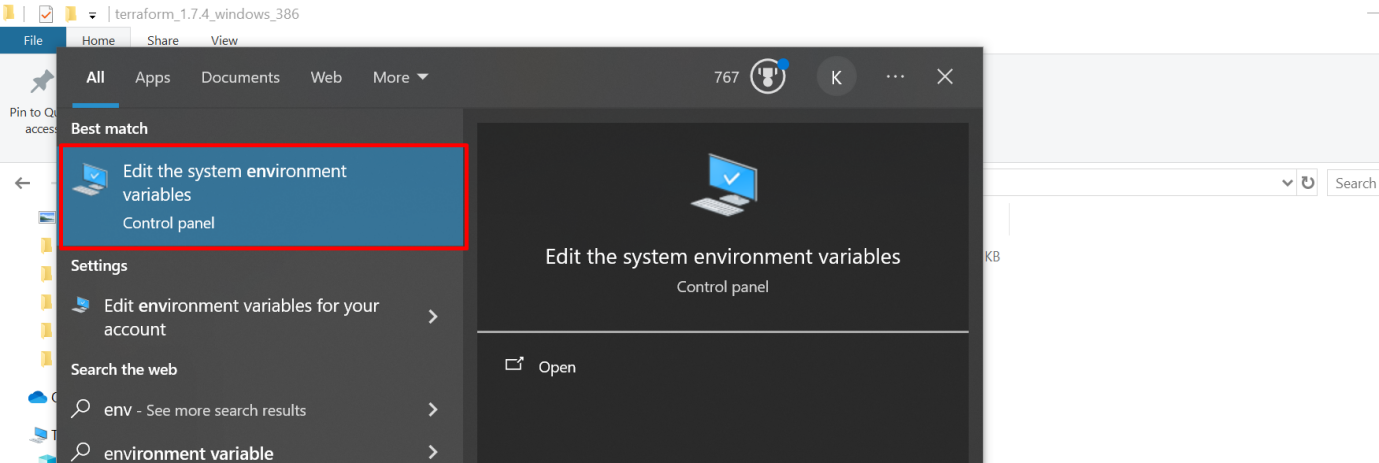
After download the Terraform file then extract the Terraform zip file.



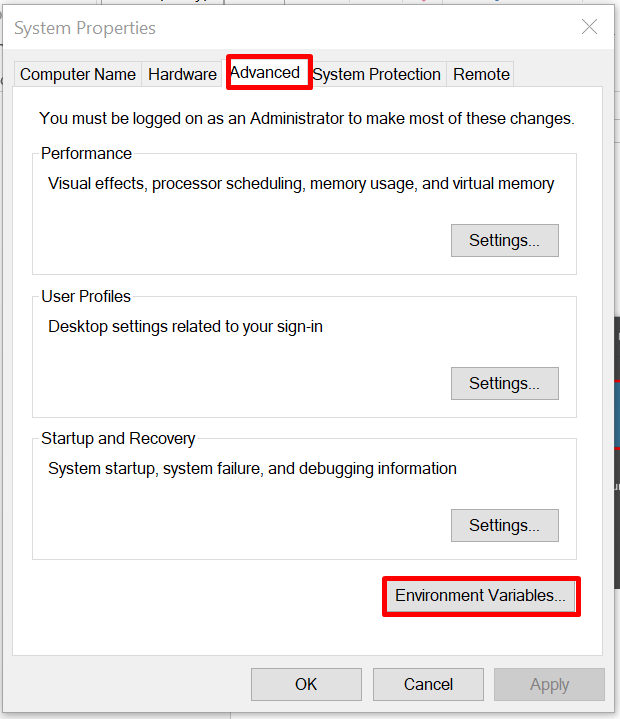
After extract and **copy** the file **path**.



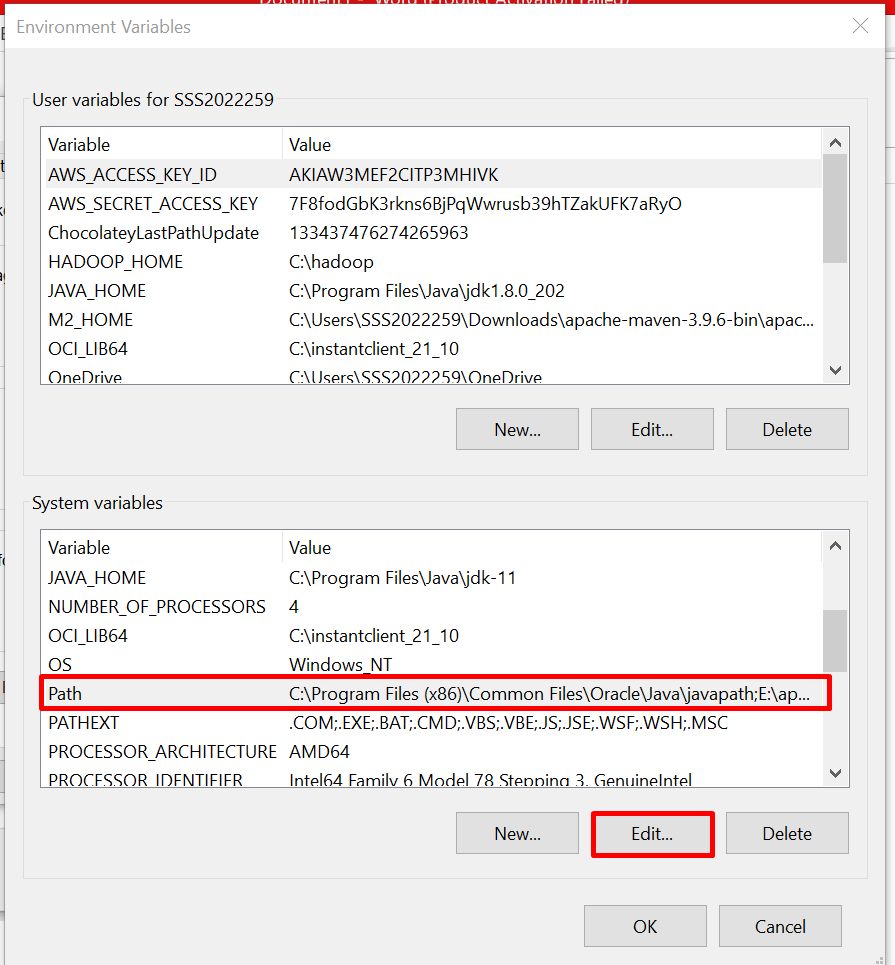
Next, open “**Edit the system environment variables**” in local server.



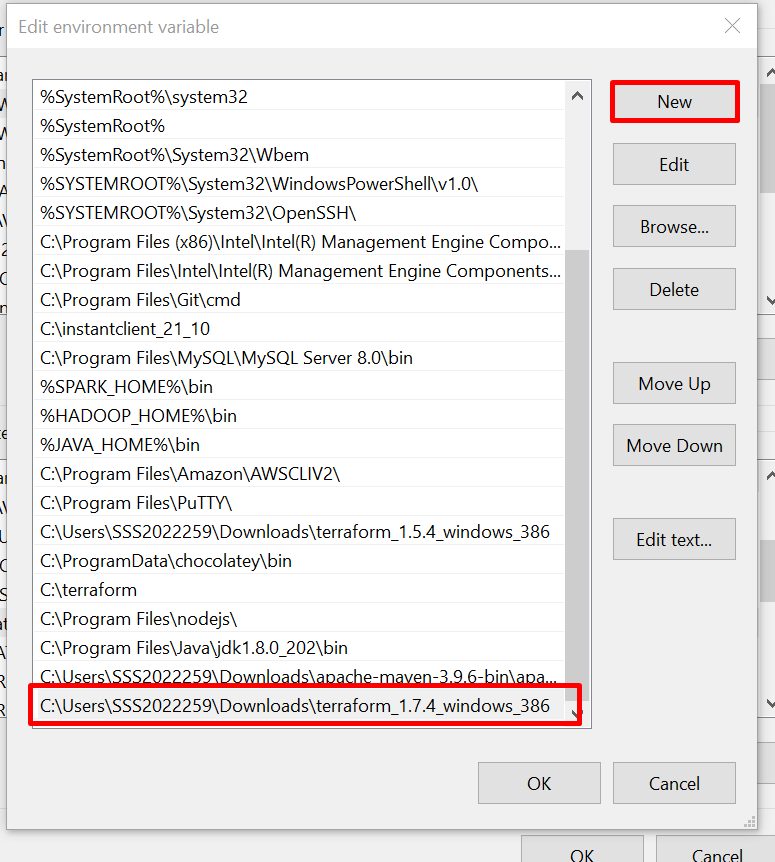
Then select the **Advanced** and go to **Environment variables**.



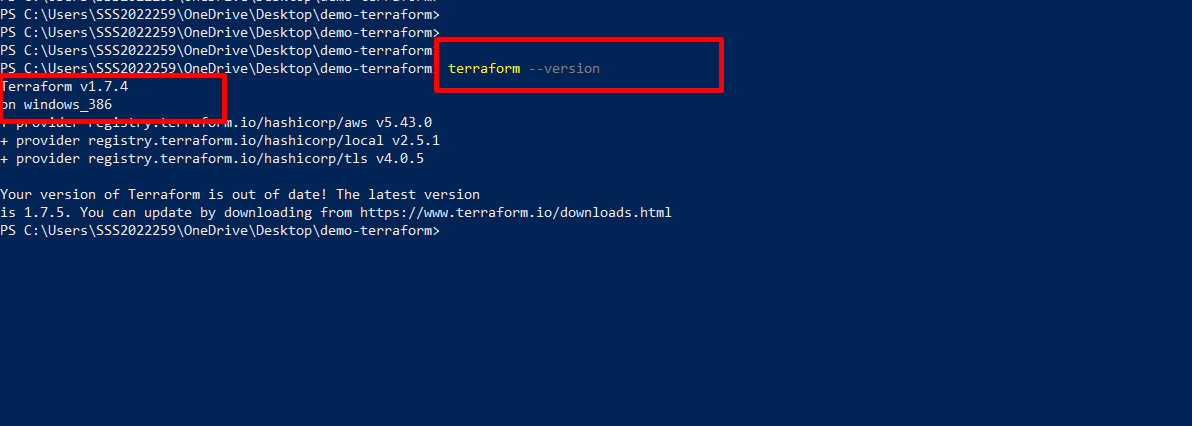
Now in **system variables** choose the **path** and then click on **edit**.



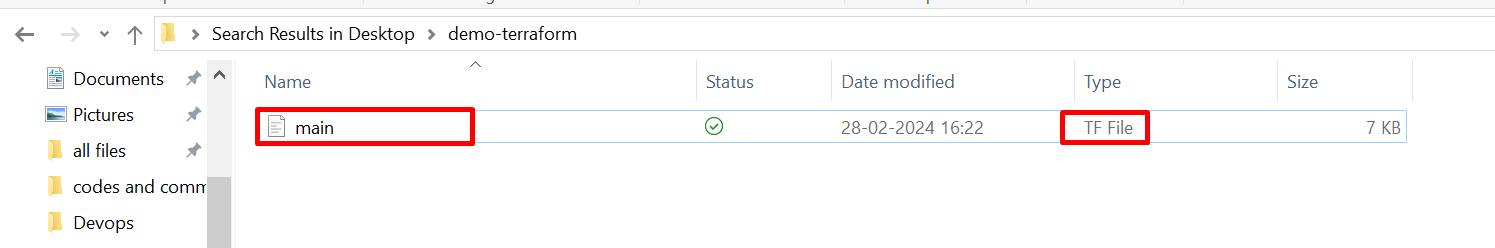
Create **New** path and **paste** the Terraform location, then click on **OK** for save the environment variable.



Open Command Line Interface **(CLI)** for checking Terraform status  
  
Then execute the command “**terraform –-version”** it shows the Terraform version as shown below image

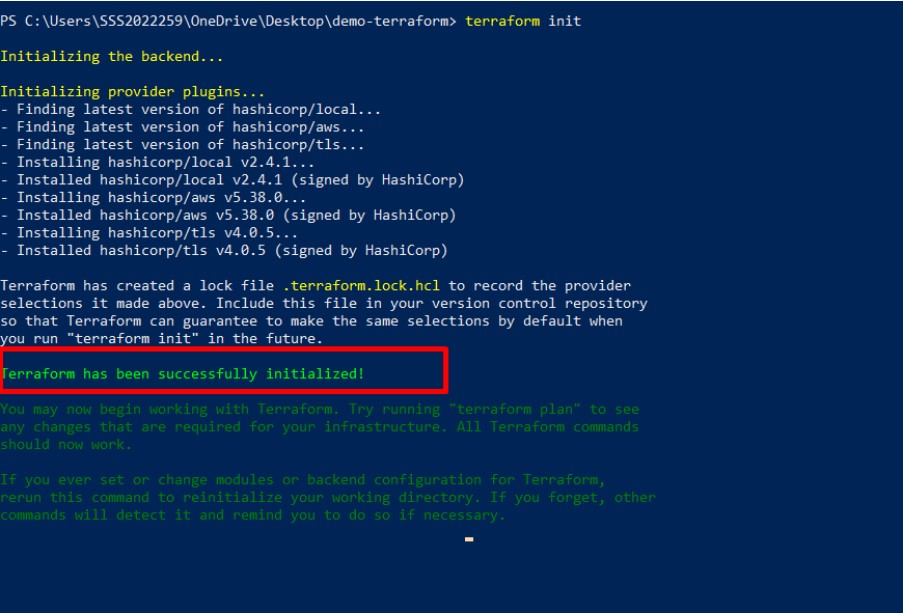


Now we write a code in terraform as our requirements.

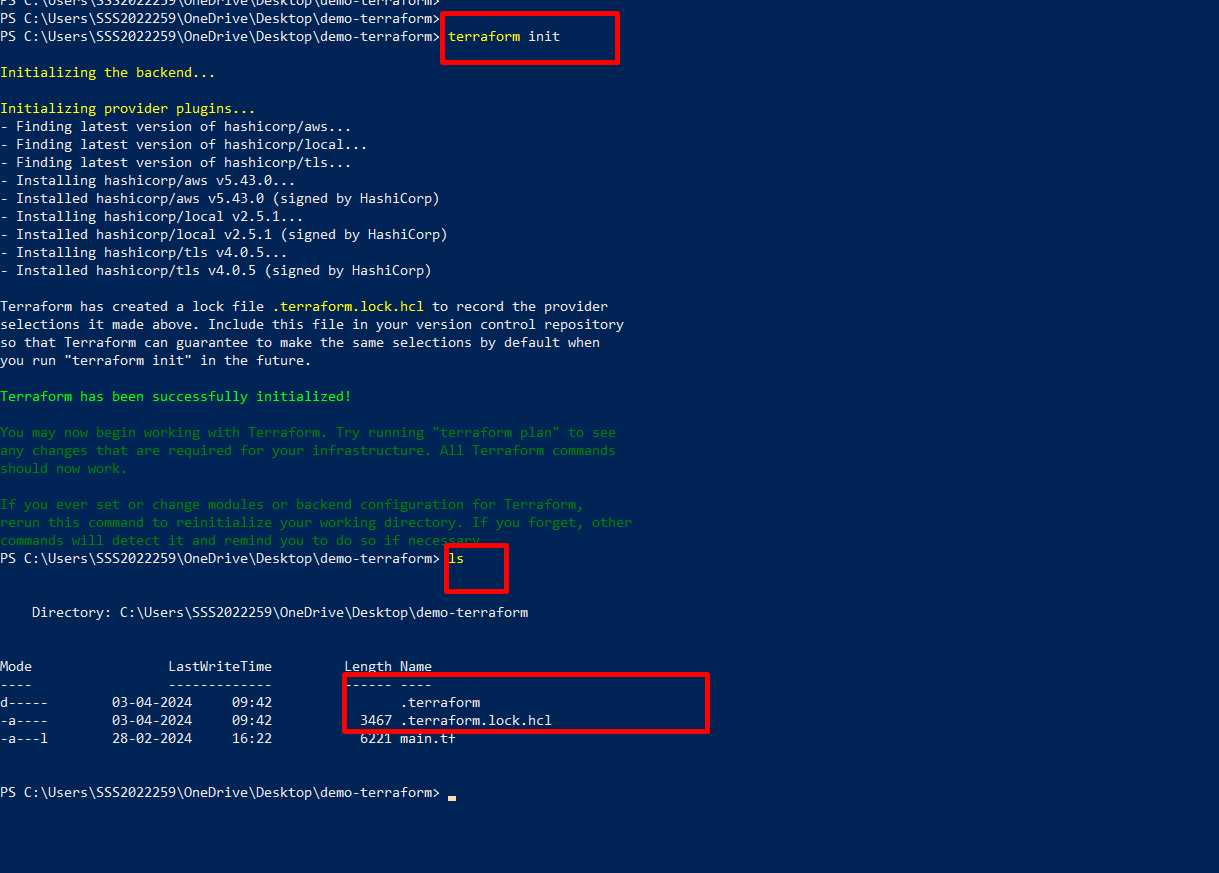


Now we go to Terraform file path on CLI

Execute this command “**terraform --init”** for install required plugins and dependencies.

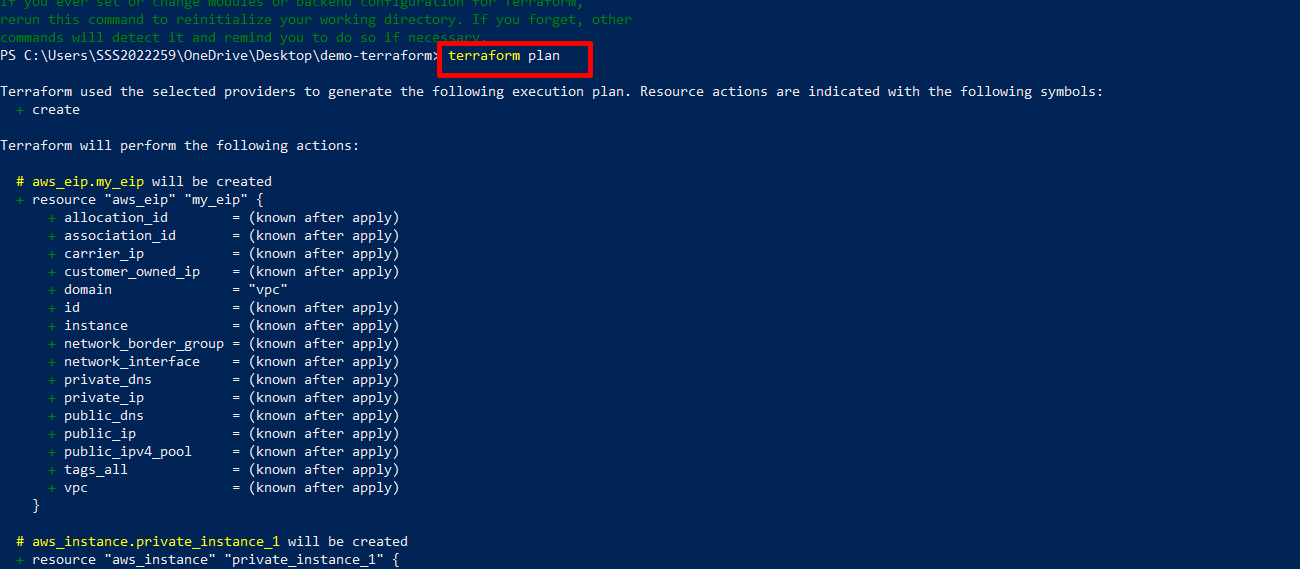


If you require you can check the file path after terraform init

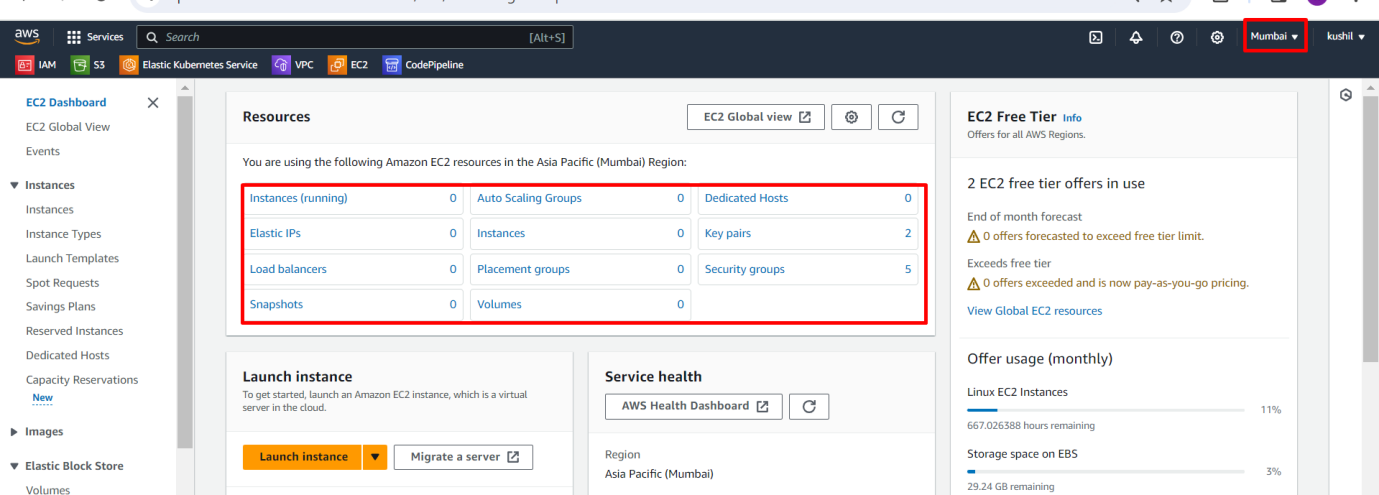


Now execute “**terraform plan”** command.

The **terraform plan** command used to creates an execution plan, which lets you preview the changes that Terraform plans to make to our infrastructure.

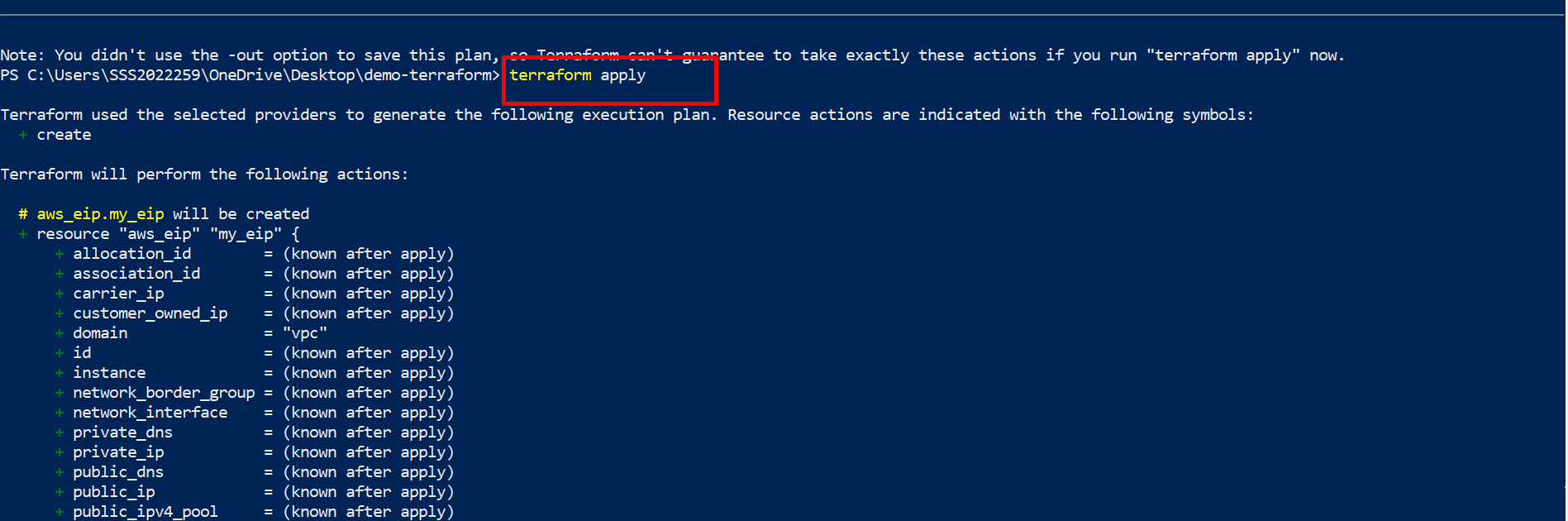


We check our AWS console once.We take **Mumbai(ap-south-1)** region and instances and Load balancers are not created .

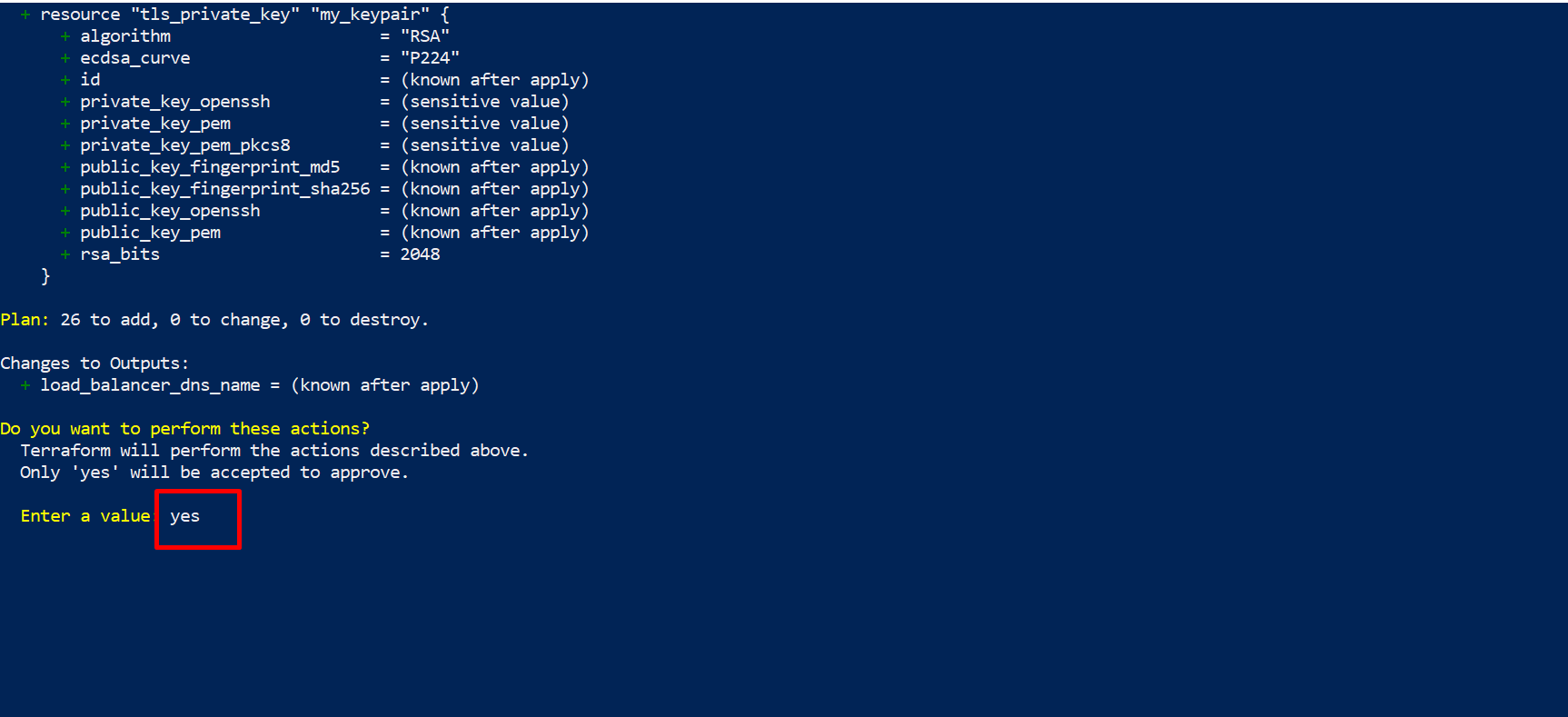


Now Execute code using “**terraform apply”** command.

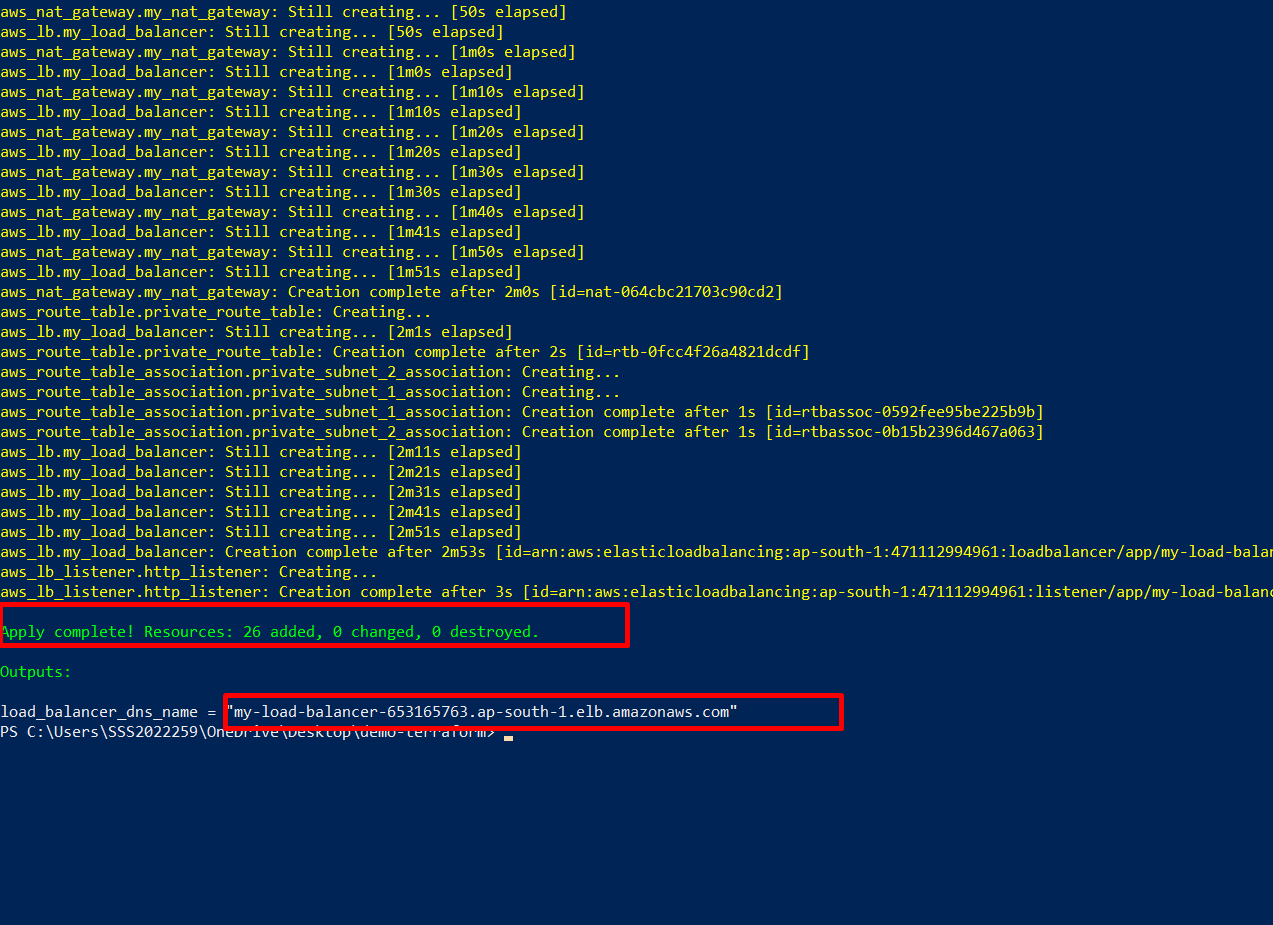
The **terraform apply** command executes the actions proposed in a terraform plan . It is used to deploy our infrastructure.



After execute terraform apply it will ask continue the perform actions or not, If we want to perform actions then Enter value **yes** otherwise **no**. Here we enter **yes**.

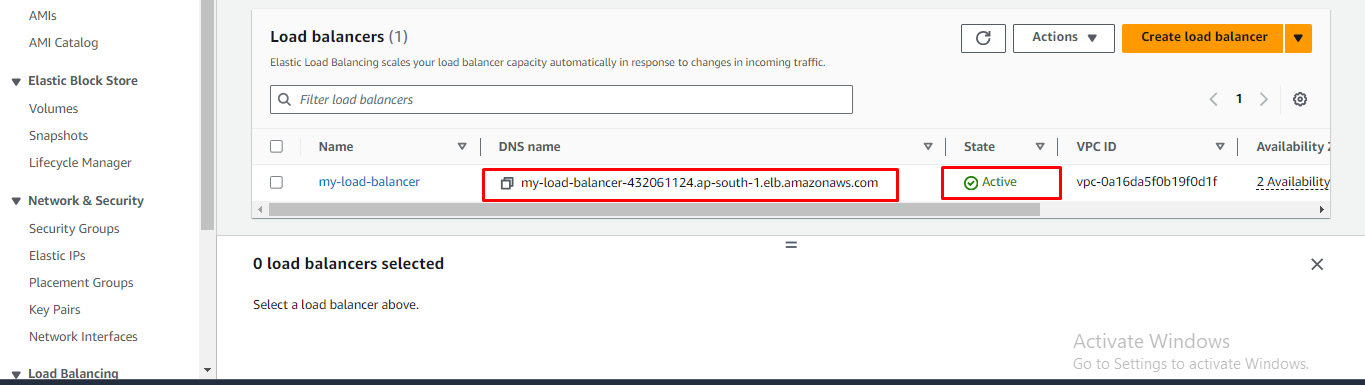


Next creating process are displays and successfully execute our main.tf file.now we copy load balancer DNS name and paste in browser.

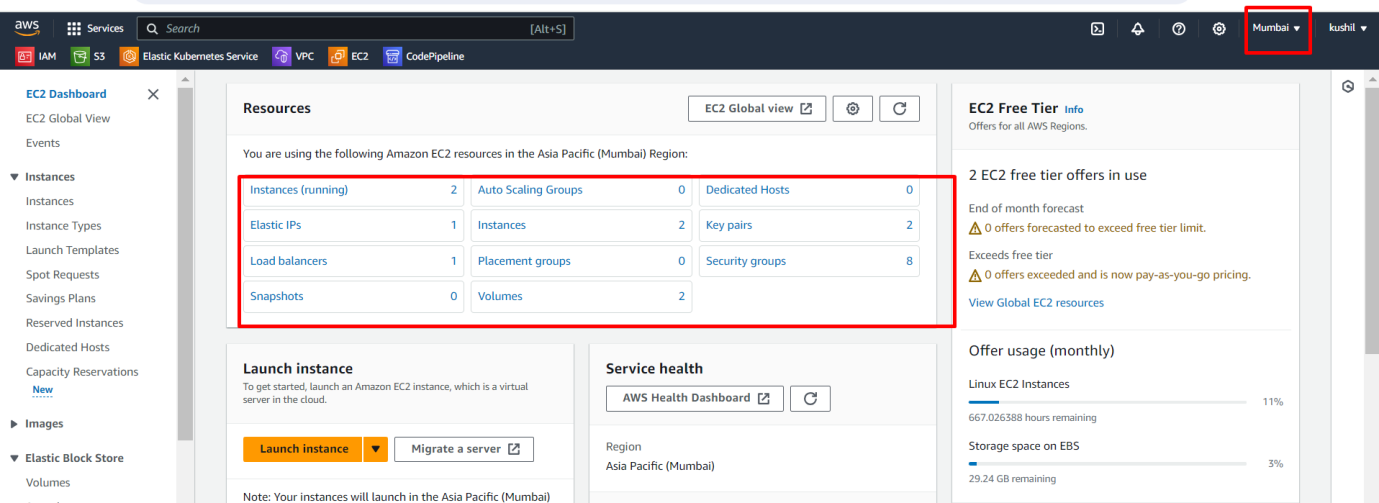


2nd way:-

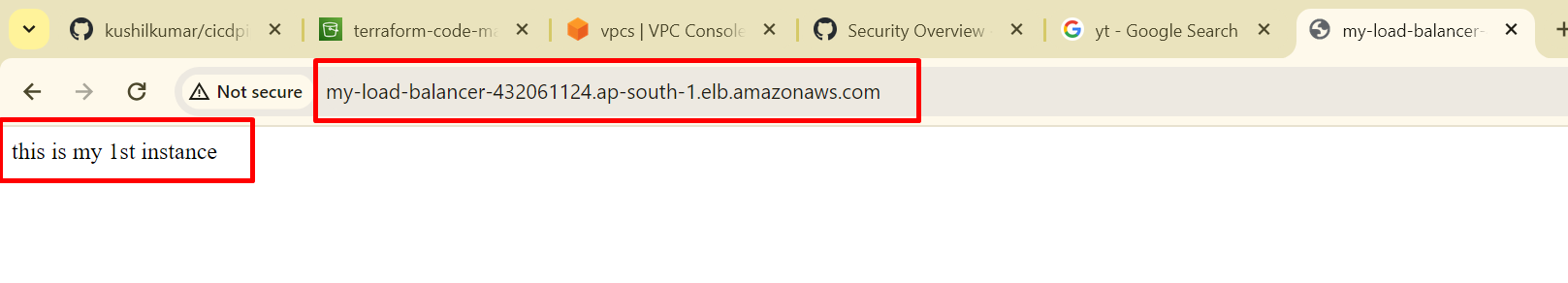
We can go to the Load Balancer service in AWS then copy the ALB DNS name.



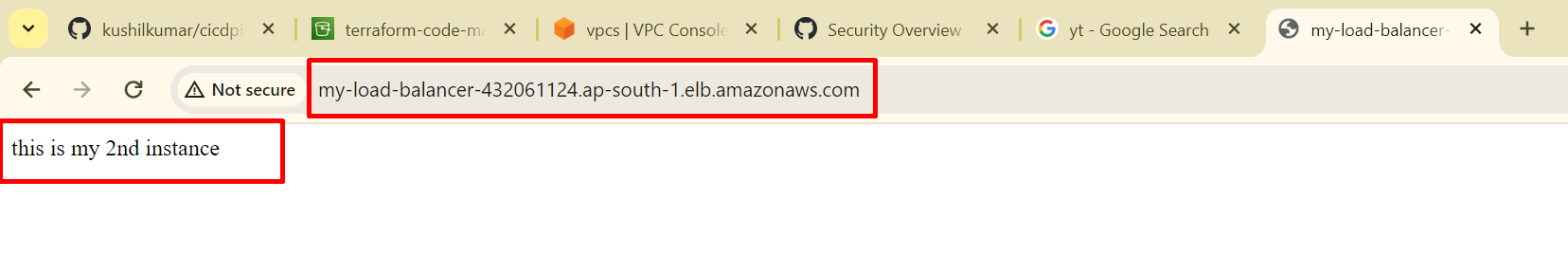
Here all required services are created successfully.



Copy and paste Load Balancer DNS Name in browser.



Once refresh then the request transfer to another server this is called Load Balancer.



**Terraform code link :-**

<https://github.com/kushilkumar/test.git>

In Terraform file we can add AWS credentials like Access\_key and Secret\_key.

**Use Cases For Private Server Hosting:-**

**Secure Application Hosting**: Organizations can host sensitive applications that require a high level of security and compliance in a private server environment. This allows them to have full control over access, data encryption, and network configuration.

**Database Hosting**: Private server hosting is often used for hosting databases that contain sensitive or confidential information. With private server hosting, organizations can implement strict access controls and encryption to protect their data.

**Internal Applications:** Companies can host internal applications and services that are not intended to be accessed by the public or external users in a private server environment.

**Development and Testing Environments:** Private server hosting can be used to set up development and testing environments that replicate production configurations while keeping them isolated from the public internet.

**Compliance and Regulatory Requirements:** Organizations subject to specific compliance requirements, such as HIPAA, PCI-DSS, or GDPR, can use private server hosting to ensure that their cloud infrastructure meets these standards.

**High-Performance Computing (HPC):** Private server hosting can be beneficial for running HPC workloads that require dedicated resources and low-latency network connections.

**Elastic Load Balancer (ELB) Used For:**

Elastic Load Balancing (ELB) is a service provided by AWS that automatically distributes incoming application traffic across multiple targets, such as Amazon EC2 instances, containers, and IP addresses, in one or more Availability Zones. ELB offers several use cases to improve the availability, fault tolerance, and scalability of your applications: