

**VIRTUAL PRIVATE CLOUD USING TERRAFORM**

**SUBMITTED BY**

**TERRIFIC TERRAFORMS (77)**

**B.V.S.S.SUSHMA(TEAM LEADER)**

**T.KAVYA SRI**

**V.N.S.SRINITHA**

**Y.PUSHPA MOULIKA**

**UNDER THE ESTEEMED GUIDANCE OF**

**BOBBY SIR**

**SURYA ASHOK SIR**

**INTRODUCTION:**

Terraform and VPC allow you to create and manage your VPC resources in a programmatic and automated way. This allows you to quickly and easily provision and manage VPC resources, while also reducing the risk of human error and enabling greater consistency across your infrastructure.

**A Virtual Private Cloud (VPC)** is a virtual network environment that provides a secure and isolated space for resources within the Amazon Web Services (AWS) cloud. VPCs can be customized to meet specific needs, and can be connected to on-premises data centers or other VPCs using VPN or Direct Connect.

**Terraform** is an open-source infrastructure as code (IaC) tool that allows you to define and manage your infrastructure as code. Terraform enables you to create, modify, and delete infrastructure resources in a safe and repeatable way, ensuring consistency and reducing the potential for human error.

**Subnet:** A subnet, or subnetwork, is a segmented piece of a larger network. More specifically, subnets are a logical partition of an IP network into multiple, smaller network segments. The Internet Protocol (IP) is the method for sending data from one computer to another over the internet. Each computer, or host, on the internet has at least one IP address as a unique identifier**.**

**Routetable:** A route table contains a set of rules, called routes, that determine where network traffic from your subnet or gateway is directed**.**

**Internet Gateway:** An Internet gateway is a network "node" that connects two different networks that use different protocols (rules) for communicating. In the most basic terms, an Internet gateway is where data stops on its way to or from other networks.

**NAT Gateway:** A NAT gateway is a Network Address Translation (NAT) service. You can use a NAT gateway so that instances in a private subnet can connect to services outside your VPC but external services cannot initiate a connection with those instances**.**

In essence, using Terraform to create and manage a VPC enables you to define your infrastructure as code, enabling you to rapidly and consistently create and manage your cloud infrastructure.

**SCOPE OF THE PROJECT**

The scope of a Virtual Private Cloud (VPC) using Terraform project is to create a secure and isolated network environment in the cloud. The project includes configuring the VPC, subnets, route tables, internet gateways, security groups, NAT gateways, and VPN connections using infrastructure as code. The goal is to provide a reliable, scalable, and customizable infrastructure that can be easily managed and updated. Overall, the project scope is to create a flexible and secure cloud-based networking solution that can meet the requirements of various applications and workloads.

**PURPOSE OF THE PROJECT**

The purpose of a Virtual Private Cloud (VPC) using Terraform project is to create a secure, flexible, and cost-effective networking infrastructure in the cloud for running applications and workloads. The project achieves this by defining and managing the VPC resources as code, which enables easy and efficient deployment, management, and updating of the infrastructure.

**TOOLS USED**

There are several tools used in a Virtual Private Cloud (VPC) using Terraform project to create and manage the networking infrastructure in the cloud. These tools include

**1.Terraform**: Terraform is an open-source infrastructure as code tool used to create, manage, and update the VPC and its resources.

**2.Amazon Virtual Private Cloud (VPC):** Amazon VPC is a service that allows you to create a virtual network environment in the cloud.

**3.AWS CLI**: It allows users to interact with AWS services and resources using commands in the terminal. This can be useful for tasks such as configuring AWS credentials, managing EC2 instances, and performing other AWS resource management tasks.

**4.Visual Studio Code (VS Code)**: Visual studio code can be used for creating and managing a Virtual Private Cloud (VPC) using Terraform.

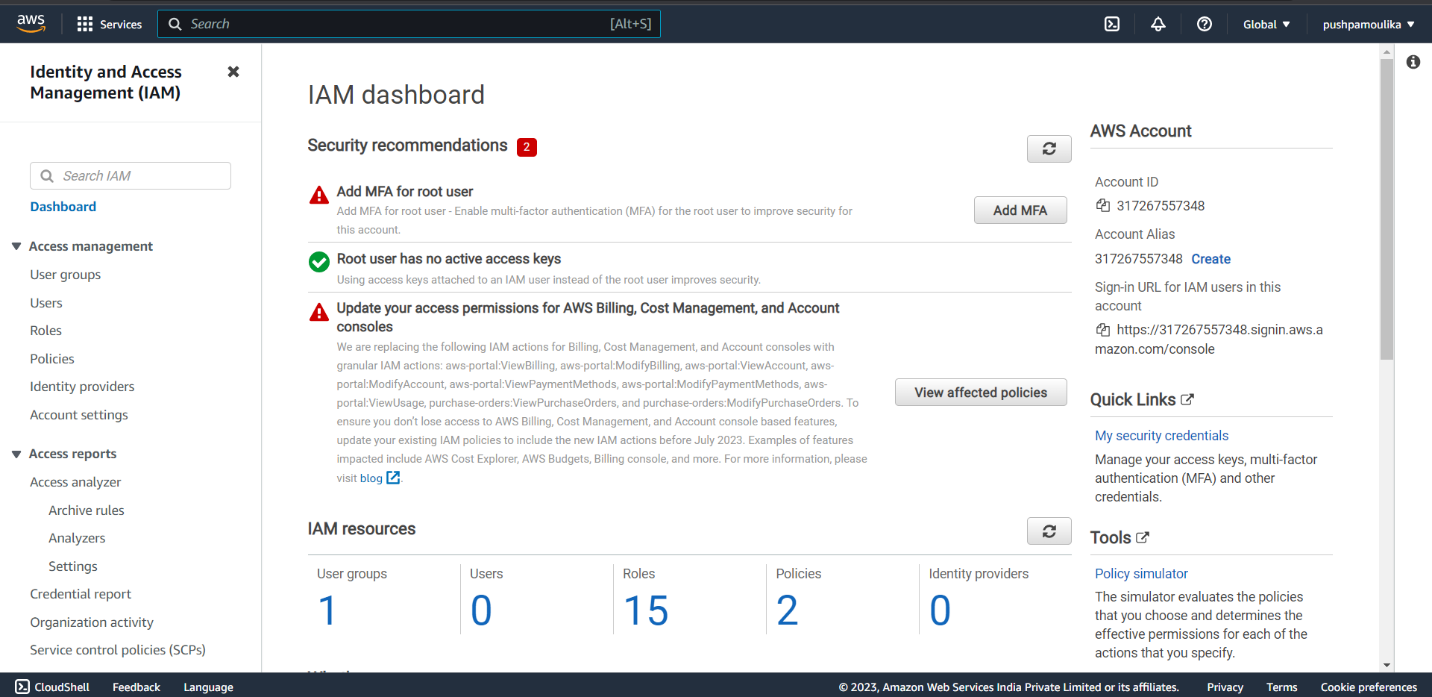
**TECHNOLOGIES USED**

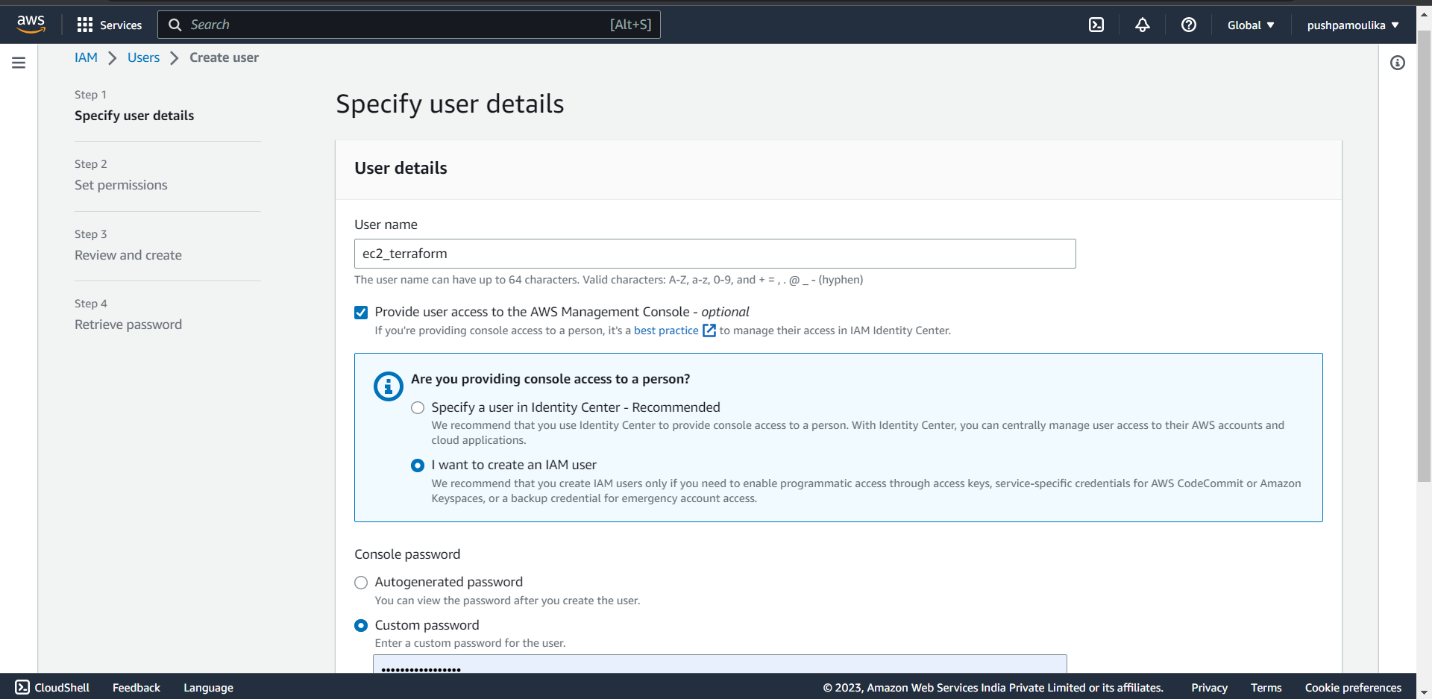
The technology used in a Virtual Private Cloud (VPC) using Terraform includes AWS, Amazon VPC and IAM. Terraform is used as an infrastructure as code (IaC) tool to define and manage the VPC and its resources. Other tools and technologies may also be used to automate the deployment and management of the VPC using Terraform.

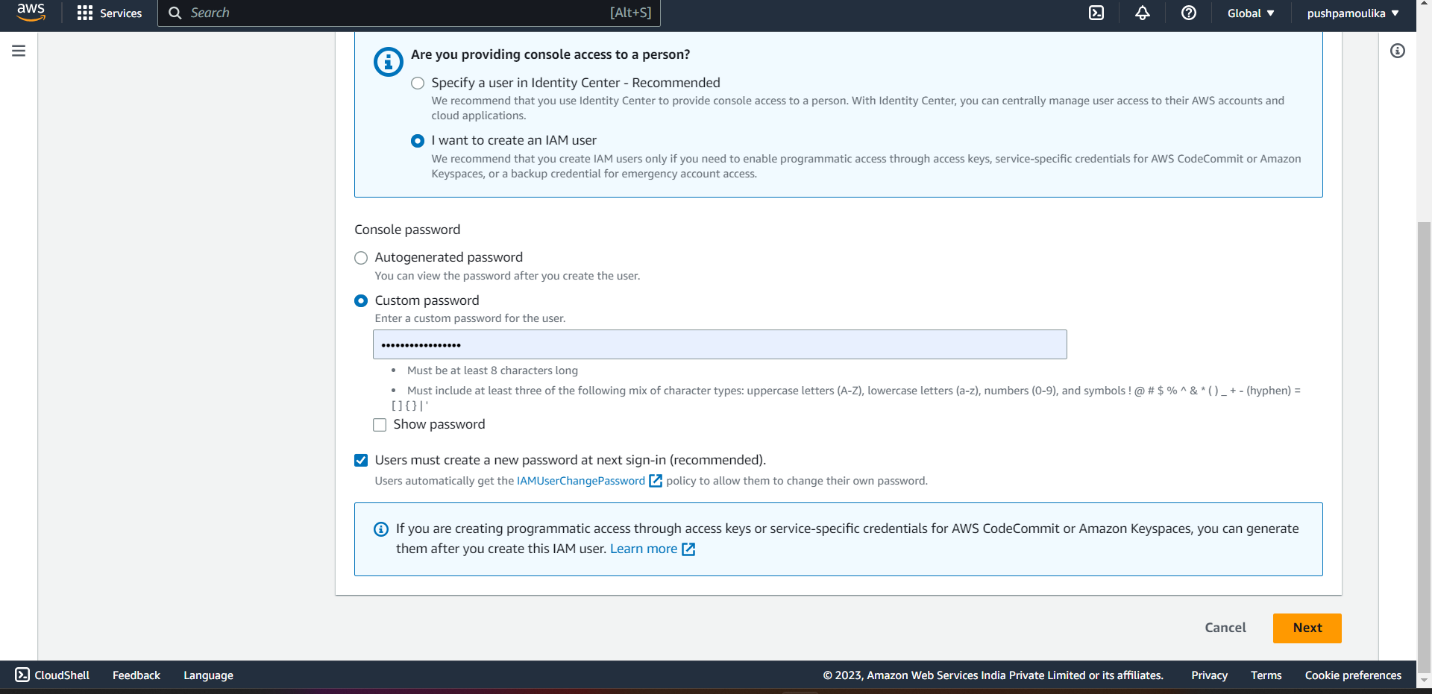
**PROOF OF CONCEPT**

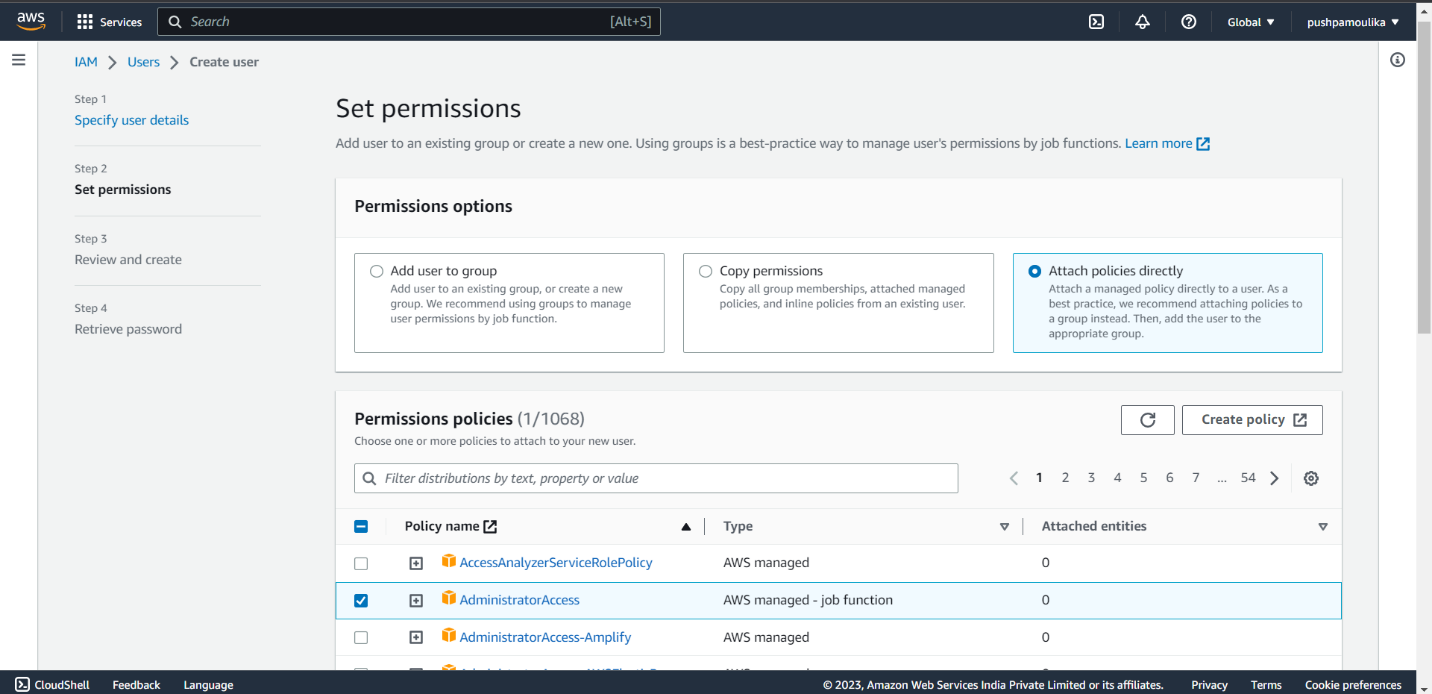
The proof of concept (POC) for a Virtual Private Cloud (VPC) using Terraform typically involves creating a simple VPC with a few resources to demonstrate the basic functionality and feasibility of using Terraform for infrastructure as code (IaC).

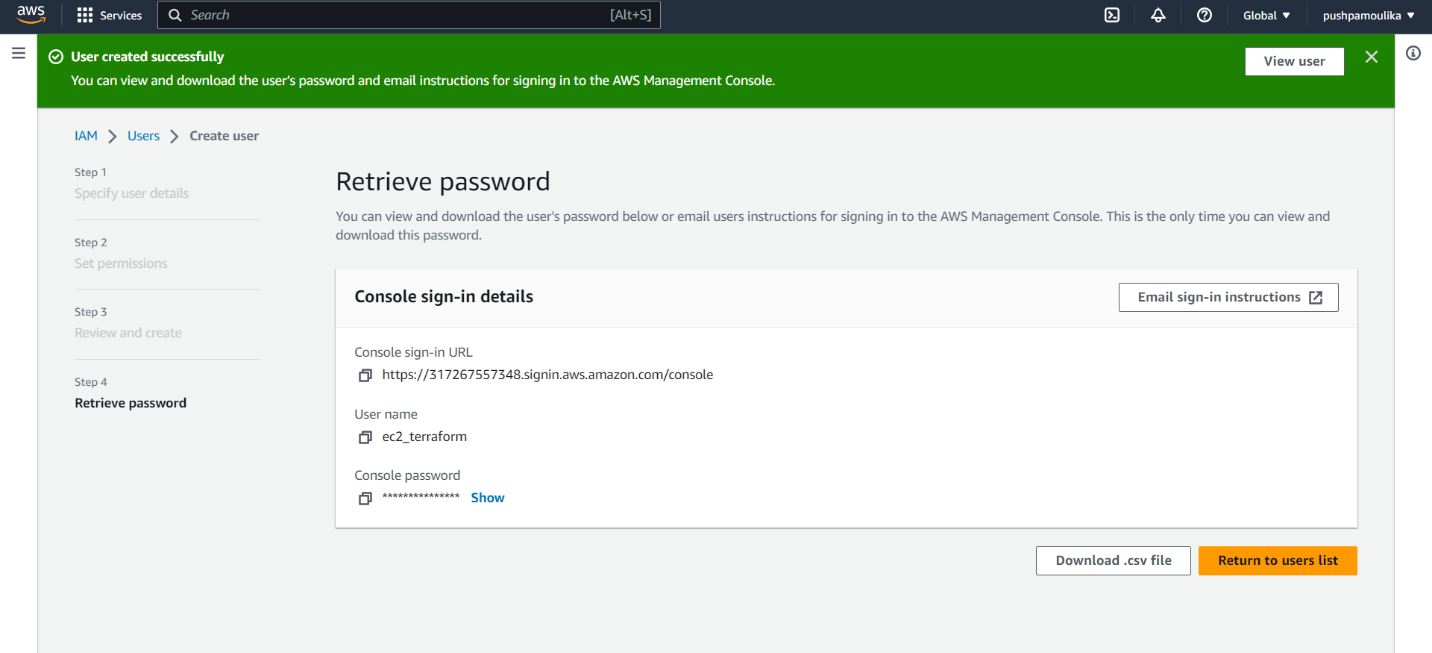
**STEP1:** first we need to install terraform and aws cli in our local system. Now open aws account and create IAM user with administrator access and go to security credentials and create access key and secret key and click on download csv file.

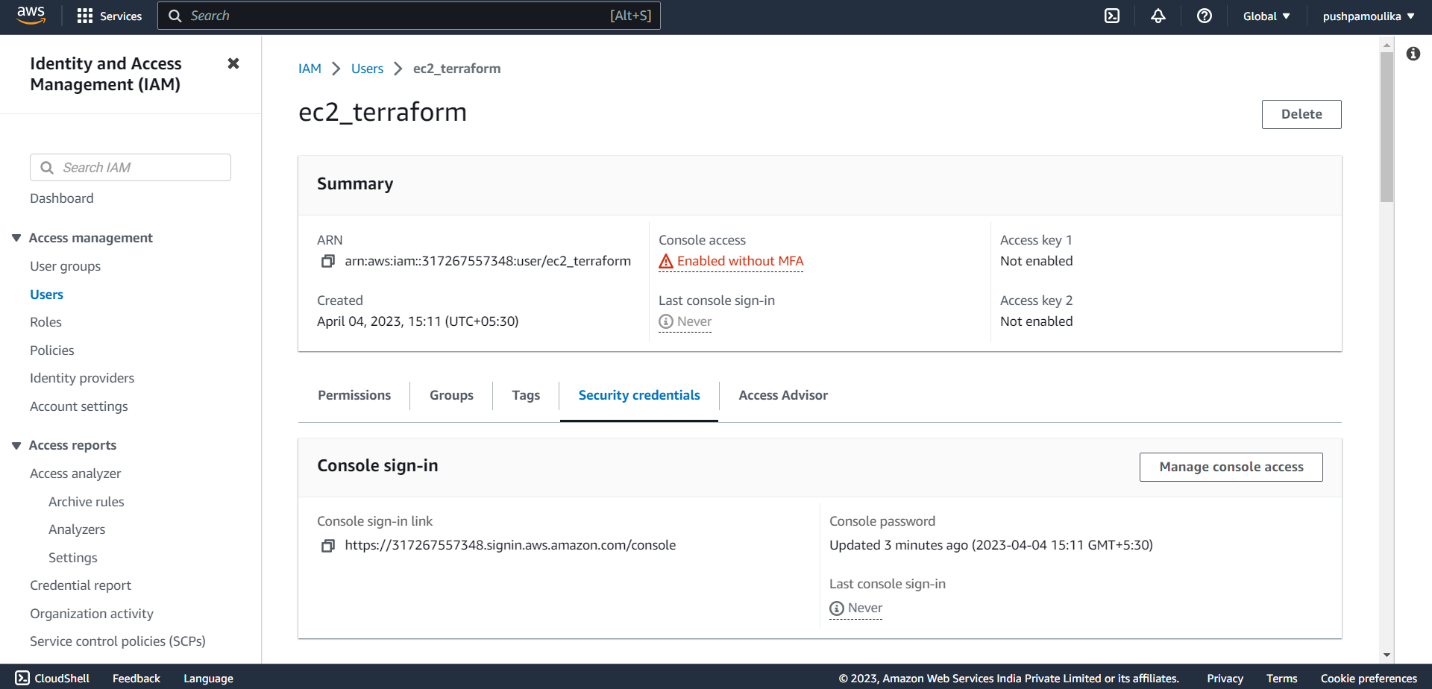


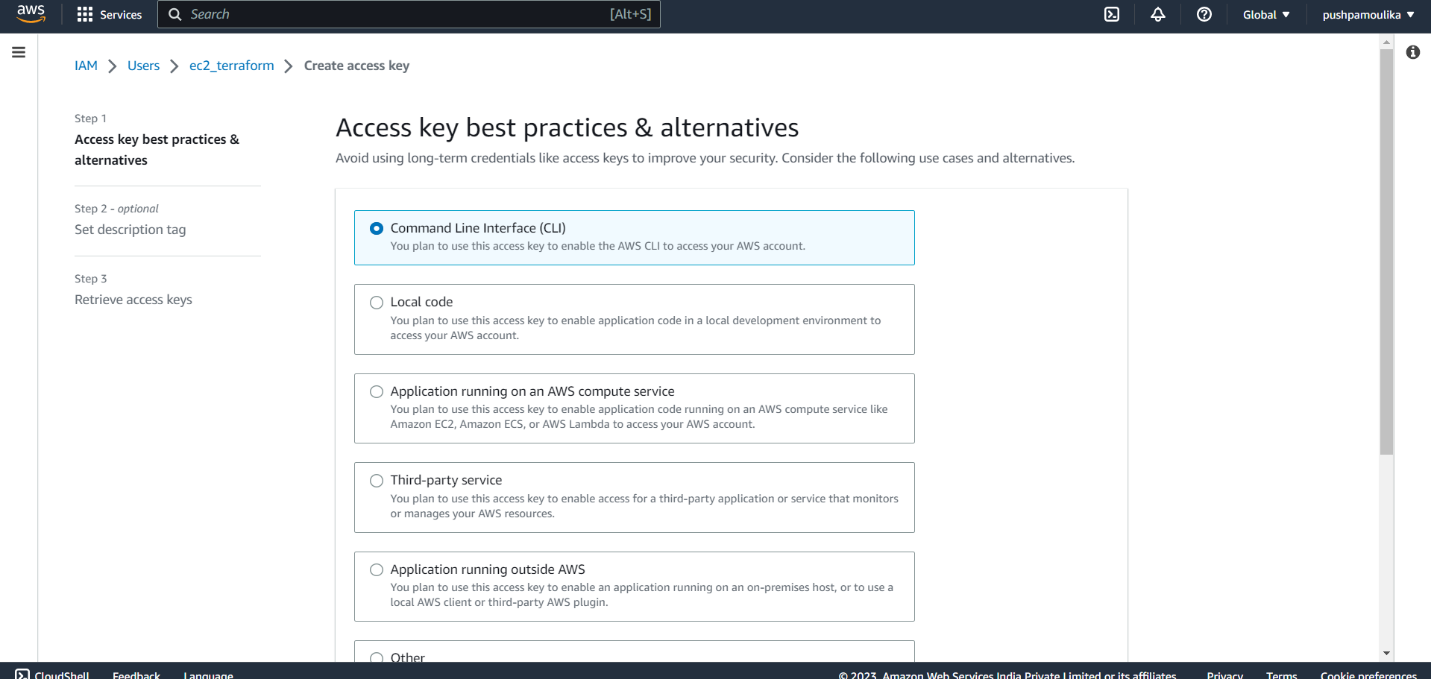


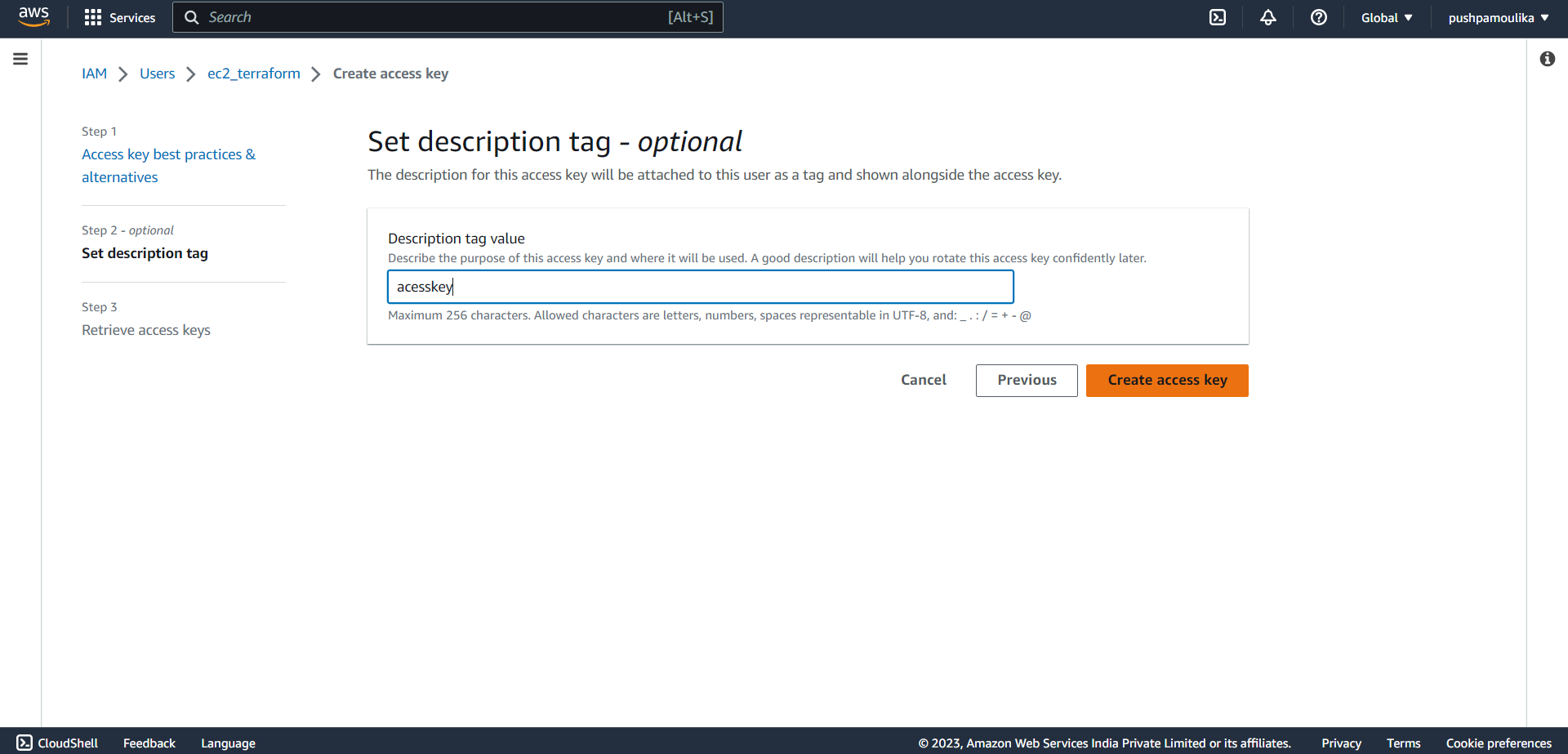


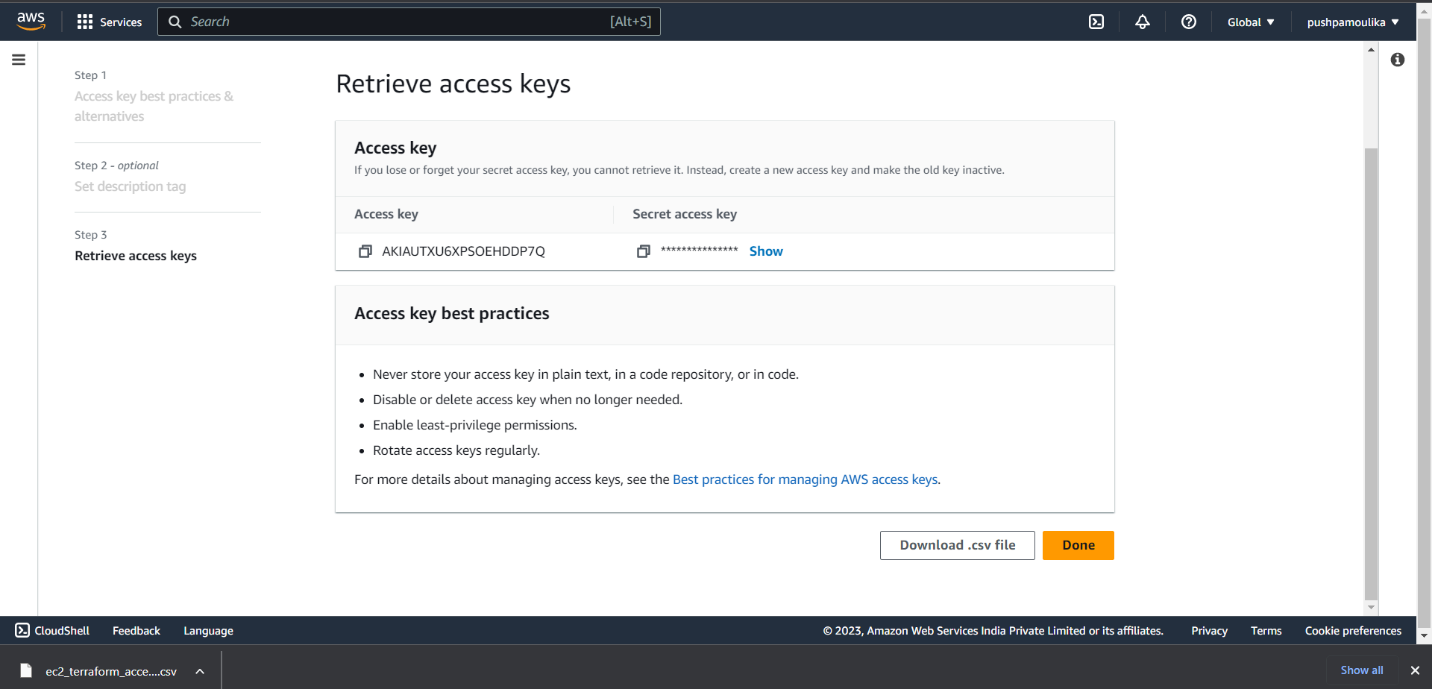






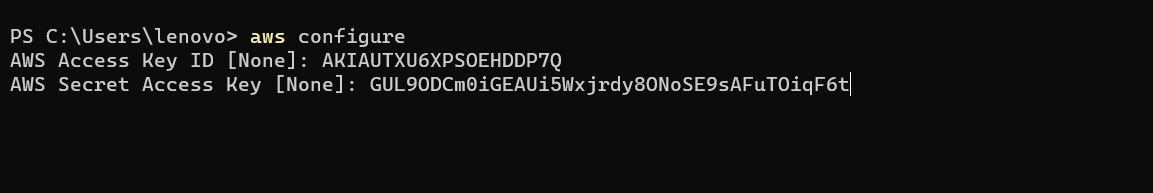


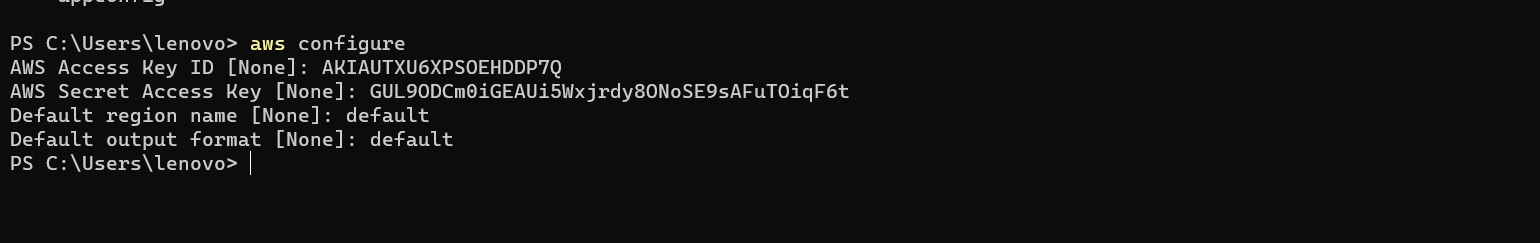




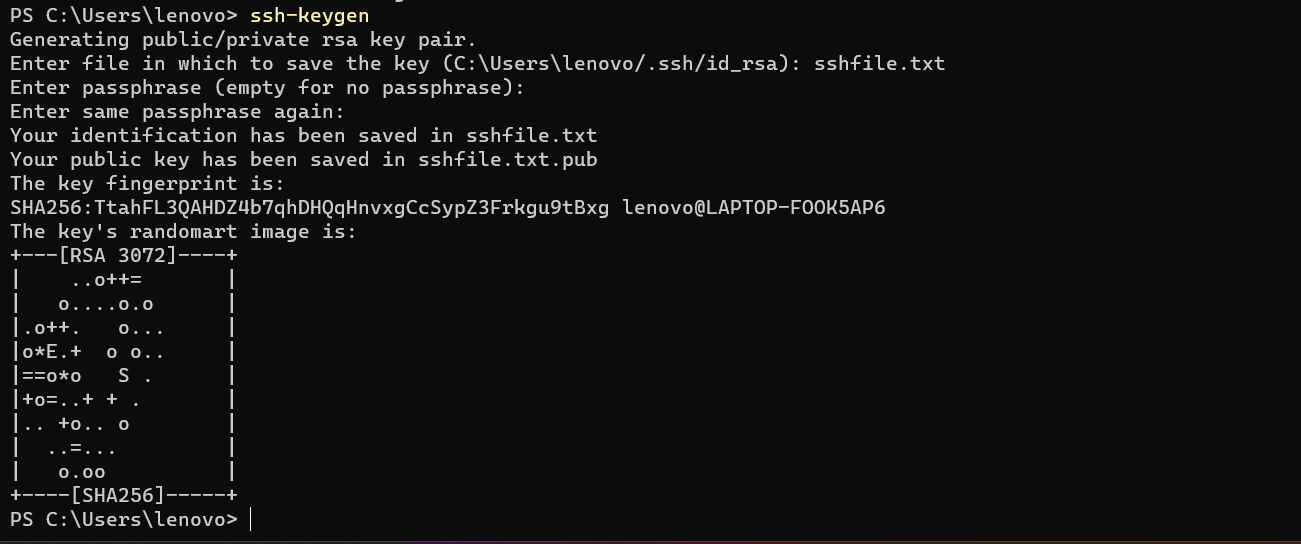
**STEP2**: Now open PowerShell with run as administrator and give the command aws configure.

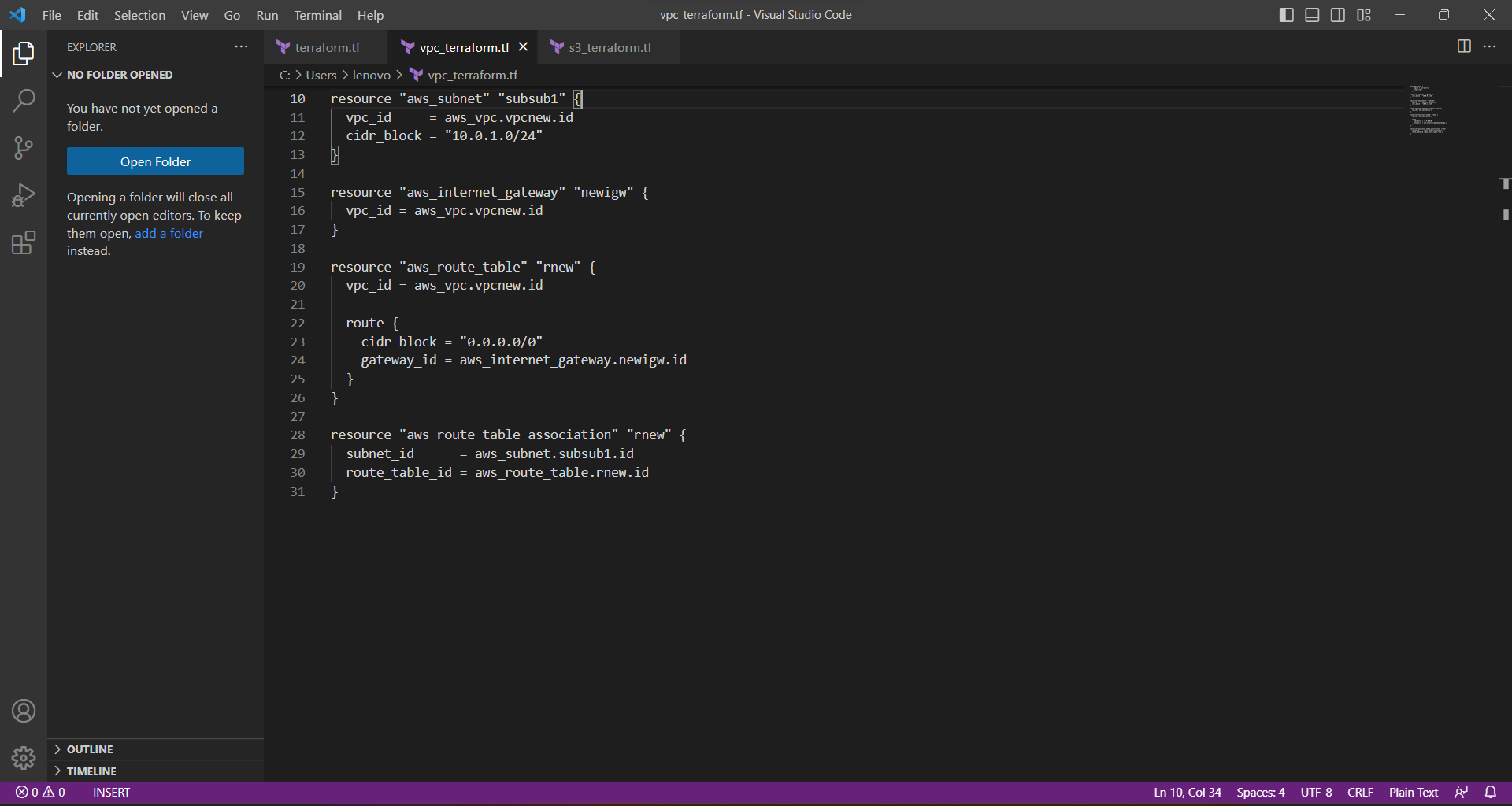
It asks for access key and secret key give them later it asks for region give default.



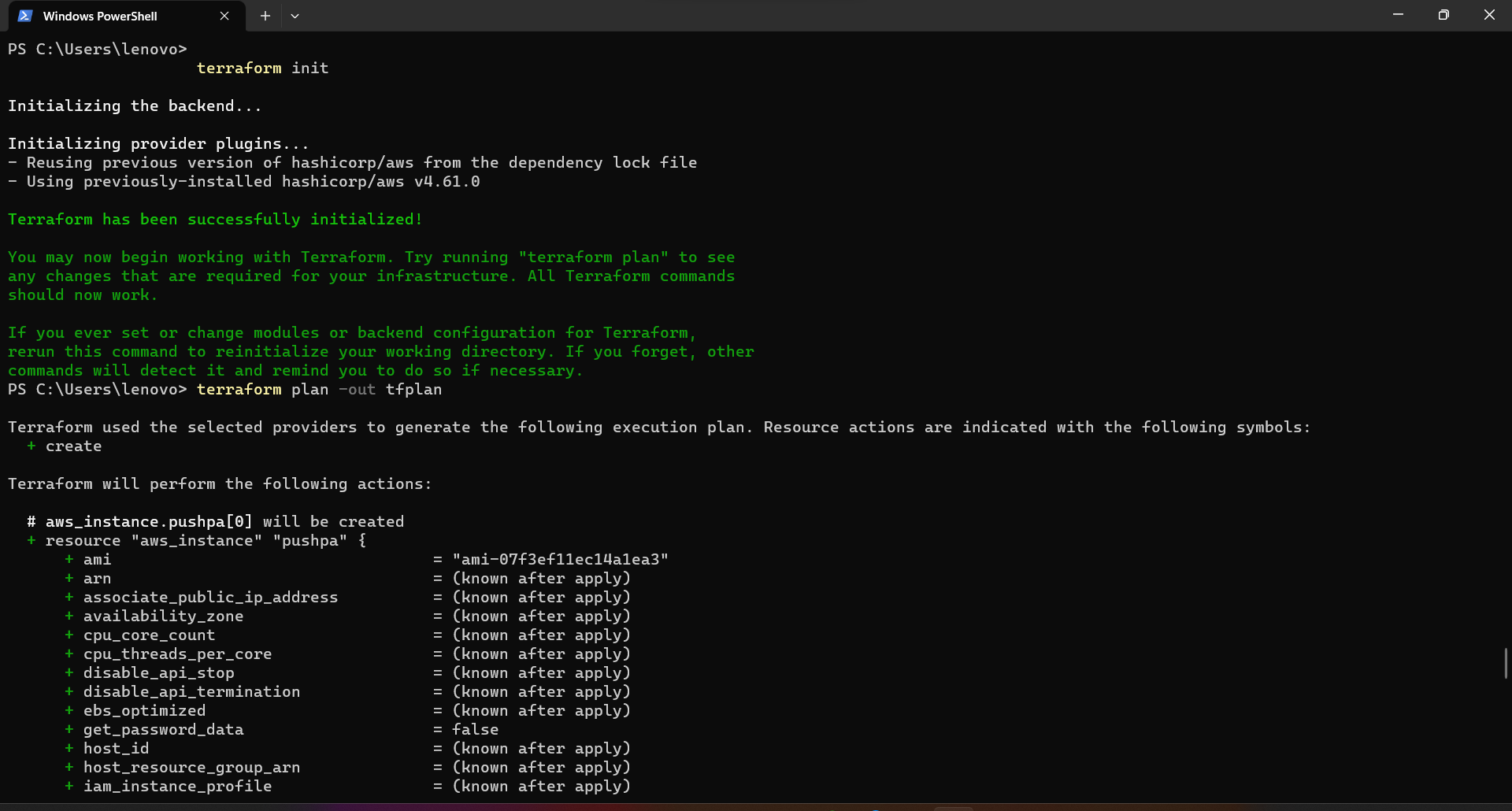


To generate the SSH key give the command ssh-keygen.

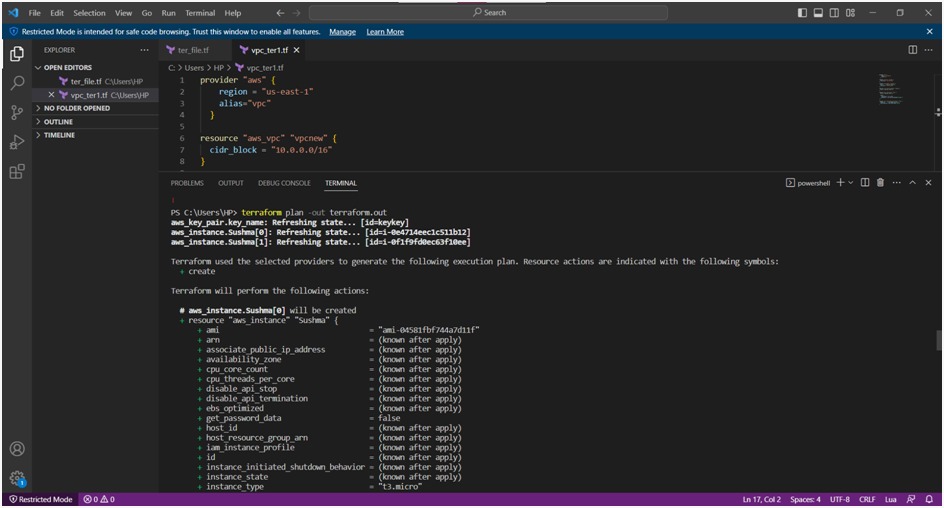


**STEP3:** Now create a folder in a desired path and open it in visual studio code and type the following code. 

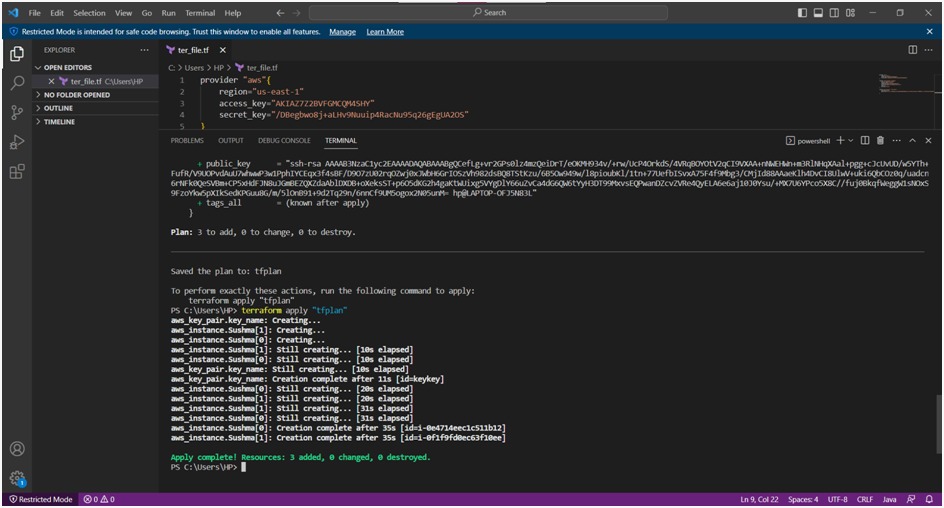
**STEP 4:** Now run the following command **“terraform init”** Terraform init command is used Initializes a Terraform working directory by downloading the necessary provider plugins and modules.



**STEP5:** Now after the terraform init command give the following command **“terraform plan”** terraform plan command is used generates an execution plan that shows what changes Terraform will make to your infrastructure. This is a useful command to run before applying changes to your infrastructure.



**STEP6:** Now after the terraform plan command give the following command **“terraform apply”** Applies the changes specified in the Terraform configuration to your infrastructure.



**REPOSITORY DETAILS**

**main.tf:** This file contains the main Terraform code that defines the VPC, subnets, route tables, and other resources.

**variables.tf**: This file defines the input variables that are used in the main.tf file. This allows you to customize the VPC configuration, such as the VPC CIDR block or the number of subnets.

**outputs.tf:** This file defines the output variables that are generated by the main.tf file, such as the VPC ID or the public and private subnets.

**terraform.tfvars**: This file contains the values for the input variables defined in variables.tf.

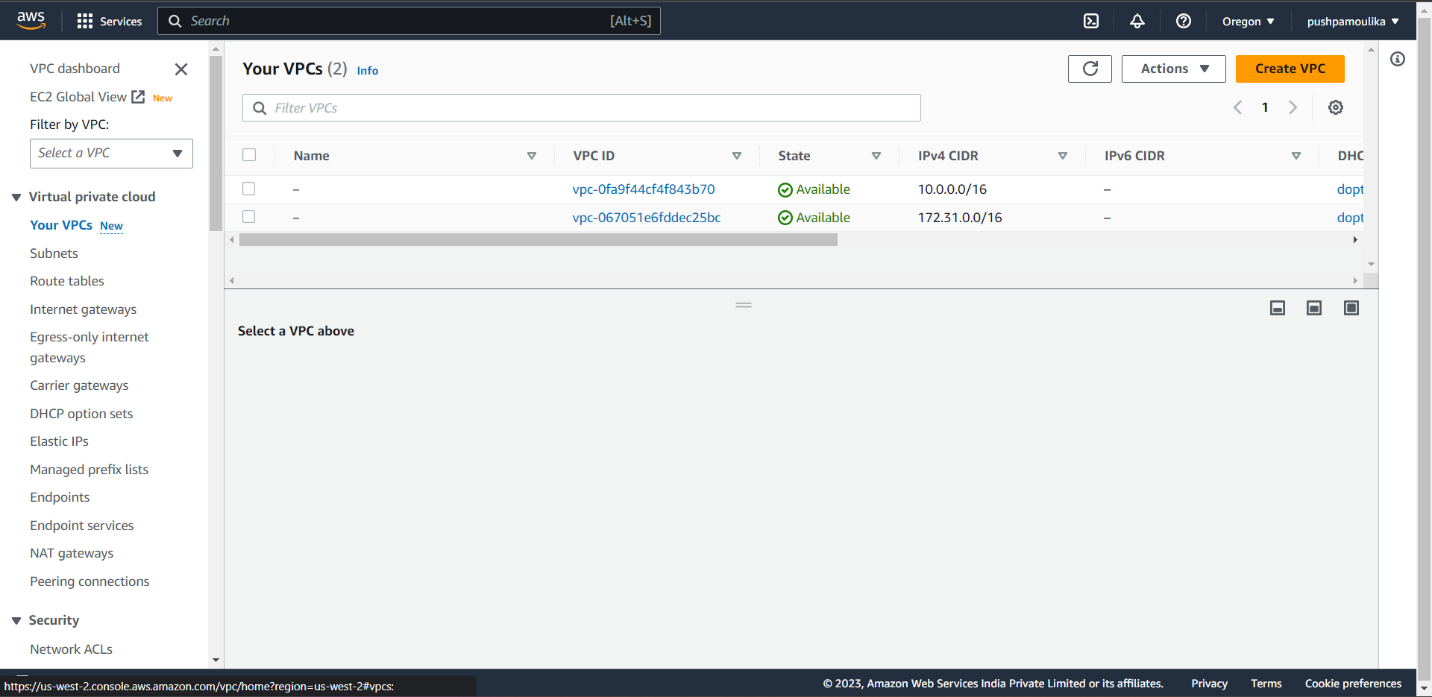
**provider.tf:** This file specifies the provider details, such as AWS credentials and region.

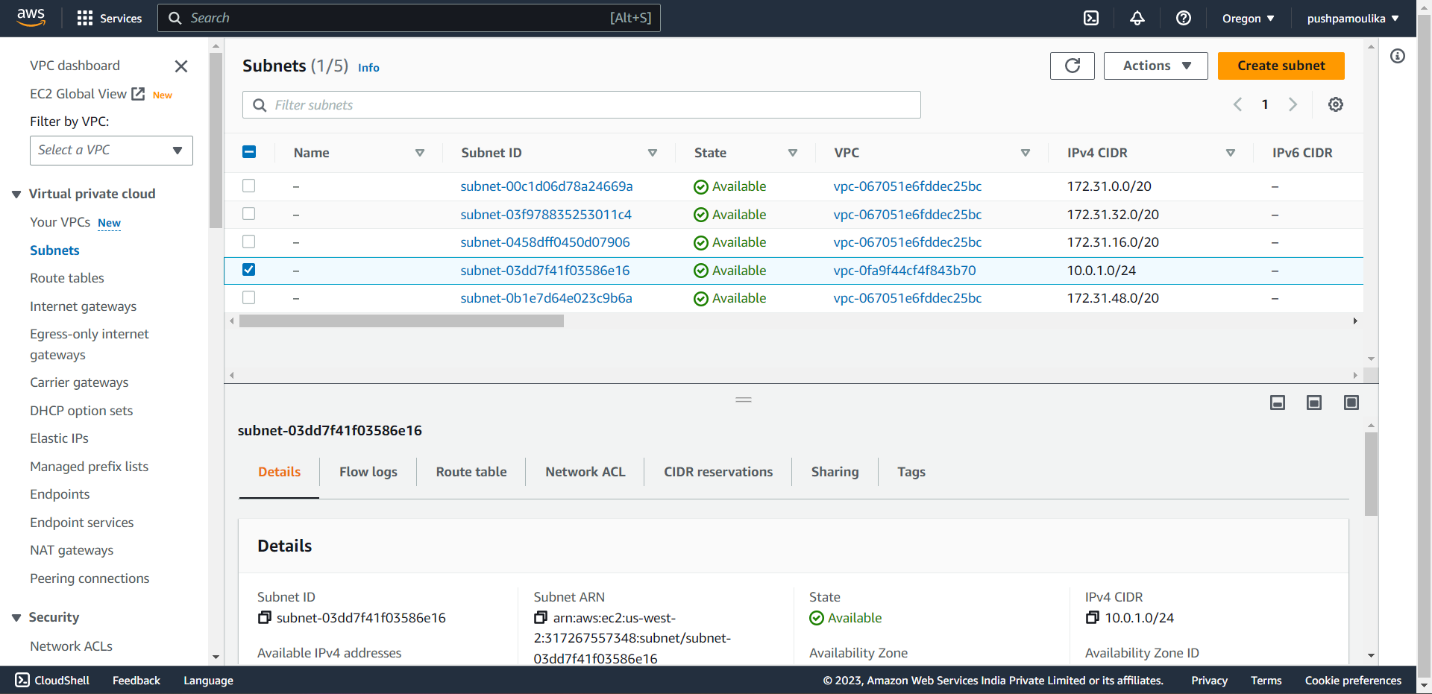
**modules/:** This directory contains any Terraform modules used in the project, such as a module for creating EC2 instances.

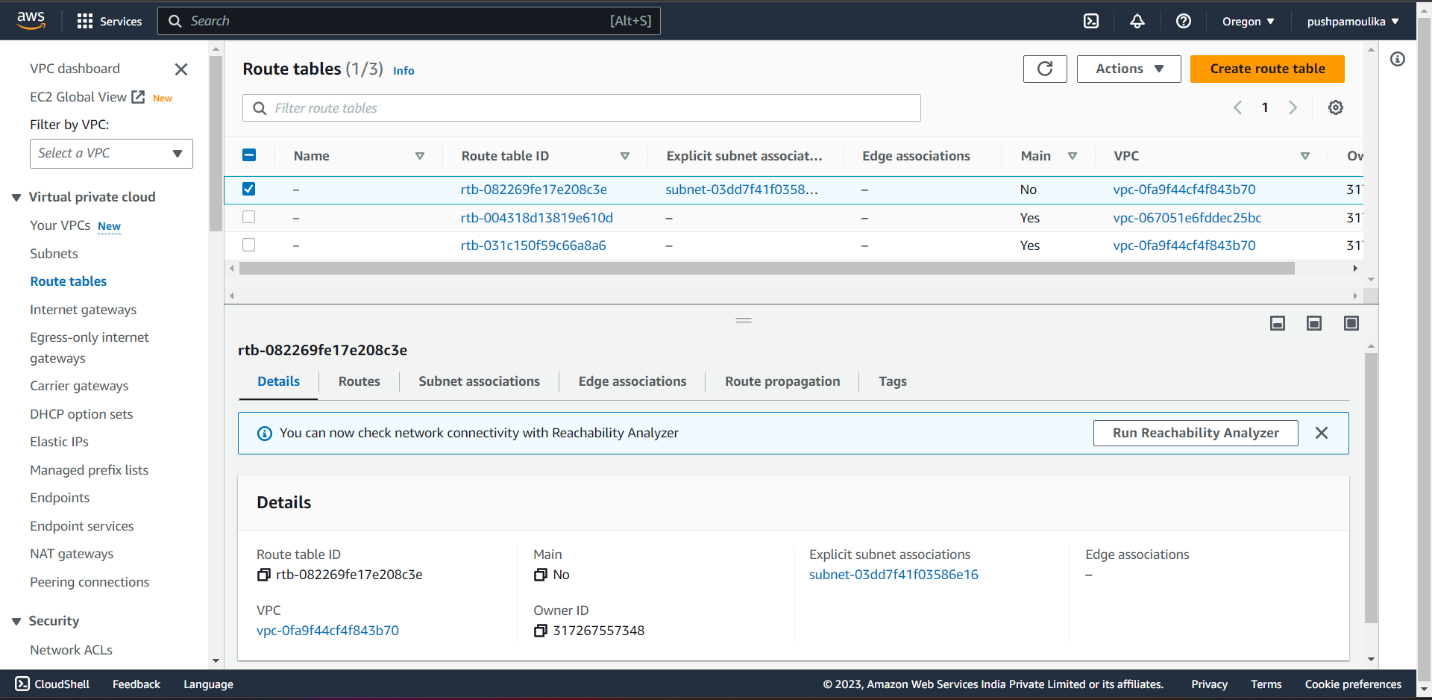
**main.tf:** This file contains the main Terraform code that defines the VPC, subnets, route tables, and other resources.

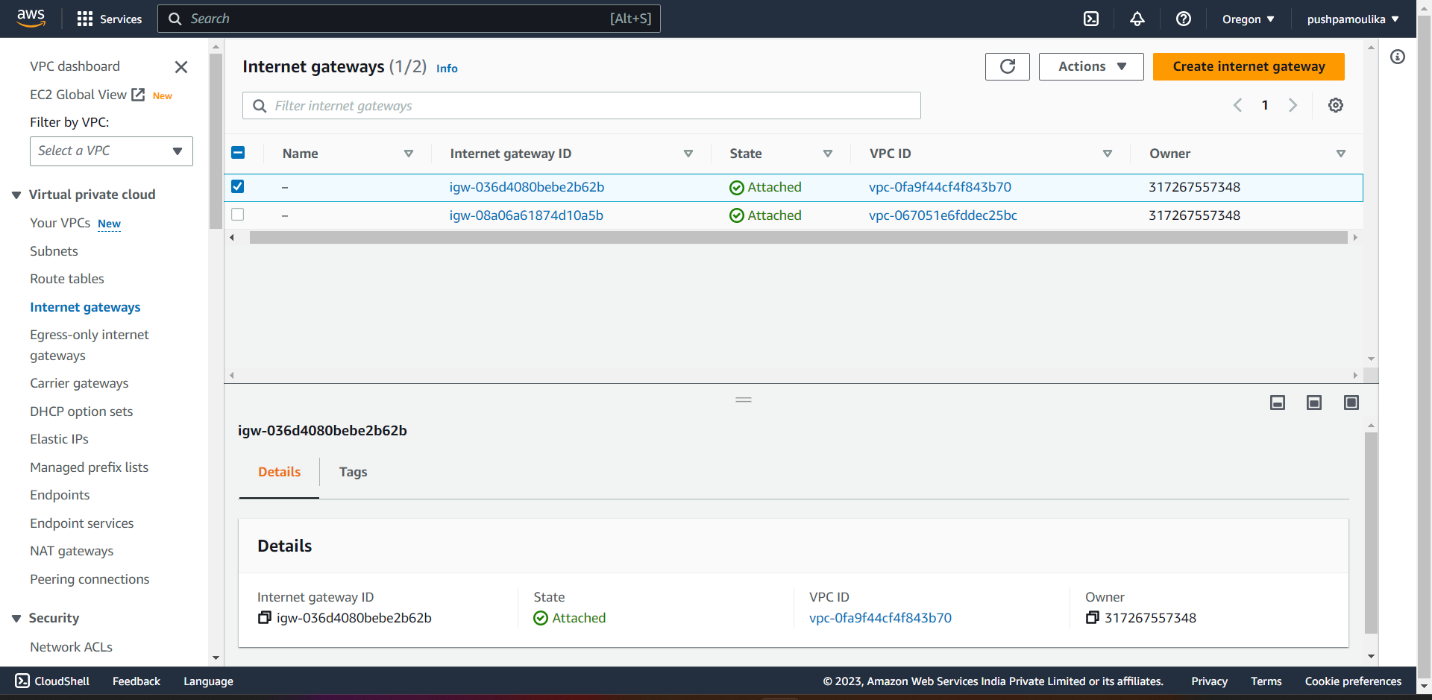
**OUTCOME**

The outcome of a VPC using Terraform project is a fully functional and customizable VPC infrastructure in Amazon Web Services (AWS). With the help of Terraform, you can quickly and easily create and manage VPC resources, including subnets, route tables, security groups, and EC2 instances. This allows you to set up a secure and isolated network environment that is tailored to the needs of your application. By using Terraform to define and manage your VPC infrastructure as code, you can ensure consistency and repeatability, as well as easy collaboration among team members. Moreover, Terraform enables you to easily update, modify, or delete VPC resources, and provides a clear view of the entire infrastructure state. Overall, the outcome of a VPC using Terraform project is a reliable, scalable, and maintainable VPC infrastructure that can support your application's growth and evolution.









At the end of the VPC using Terraform project, you should have a fully provisioned and functional VPC infrastructure in your chosen cloud provider (such as AWS, Azure, or Google Cloud Platform). This infrastructure should include all necessary resources such as VPC, subnets, routing tables, security groups, and any additional resources required for your application.

Once the infrastructure is set up, you can deploy your application and start using the VPC to host your application. You can also continue to use Terraform to manage and update the infrastructure over time, making it easier to scale and maintain your application.

The final outcome of the VPC using Terraform project is a reliable and flexible network infrastructure that enables your application to run efficiently and securely in the cloud.

**CONCLUSION**

In conclusion, using Terraform for creating and managing a Virtual Private Cloud (VPC) in Amazon Web Services (AWS) offers many benefits, such as improved infrastructure consistency, reduced human error, and increased automation. Terraform allows you to define your infrastructure as code, enabling you to rapidly and consistently create and manage your cloud infrastructure. With the help of Terraform, you can easily create and configure VPC resources, including subnets, route tables, security groups, and EC2 instances. Moreover, Terraform enables you to easily update, modify, or delete VPC resources, and provides a clear view of the entire infrastructure state. As such, Terraform is a powerful tool that can significantly enhance the management and provisioning of VPCs in AWS, making it an essential technology for cloud infrastructure management.