

LAB ACTIVITY 1

- Shaurya Srinet (RA2111032010006)

CREATE AN EC2 INSTANCE USING TERRAFORM

Procedure:

1. Install Terraform: If you haven't already, download and install Terraform from the official website: <https://www.terraform.io/downloads.html>

2. Set up AWS Credentials: Ensure you have AWS access key ID and secret access key with the required permissions to create EC2 instances. You can set these up via AWS IAM.

3. Create a Terraform Configuration File: Open a text editor and create a new file named `main.tf`.

4. Write Terraform Configuration:

```
provider "aws" {  
  region = "us-east-1"    # Change this to your desired AWS region  
}  
  
resource "aws_instance" "example" {  
  ami           = "ami-07d9b9ddc6cd8dd30" # Change this to the desired AMI ID  
  instance_type = "t2.micro"                # Change this to your desired  
instance type  
  
  tags = {  
    Name = "New_Instance" # Specify the name for your EC2 instance  
  }  
}
```

Replace `"ami-07d9b9ddc6cd8dd30"` with your desired AMI ID.

5. Save the File: Save the `main.tf` file in a directory of your choice.

6. Initialize Terraform: Open a terminal or command prompt in the directory where your `main.tf` file is located and run: **terraform init**

7. Review and Apply Changes: Before applying the changes, review what Terraform plans to do by running: **terraform plan**

If everything looks good, apply the changes: **terraform apply**

Type `yes` when prompted to confirm.

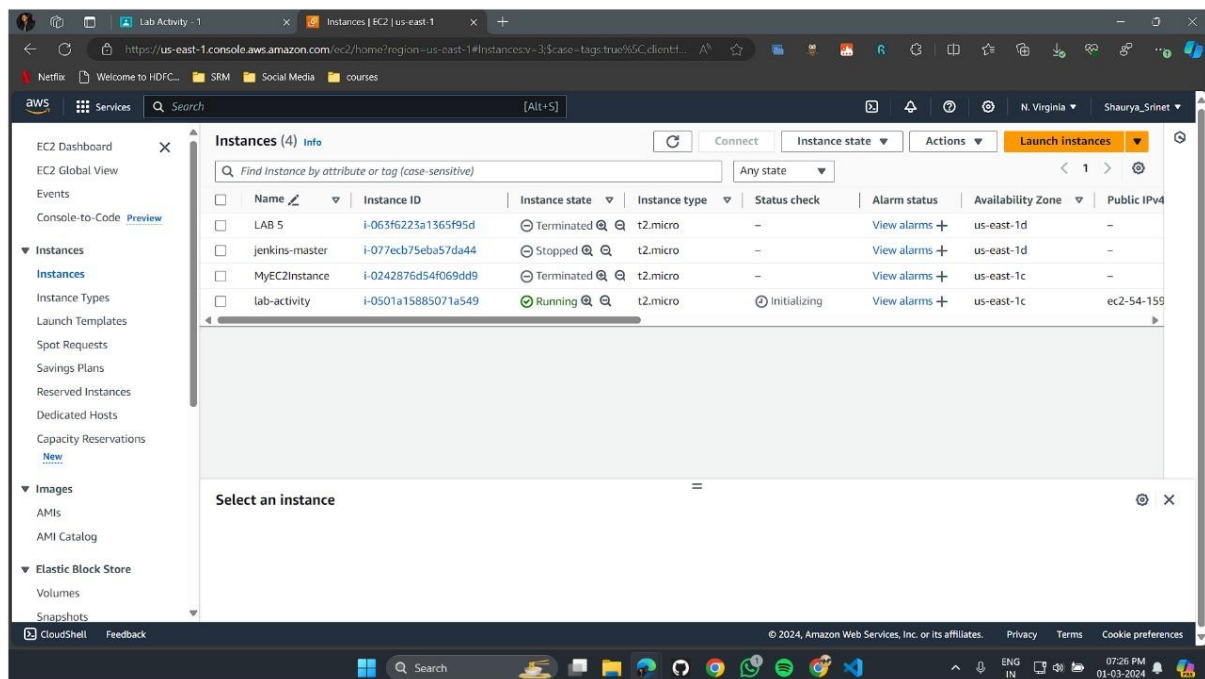
8. Verify: Once the apply command completes successfully, verify that your EC2 instance is created by logging into the AWS Management Console, navigating to the EC2 service, and checking the instances list.

9. Clean Up: To clean up resources created by Terraform, you can run: **terraform destroy**

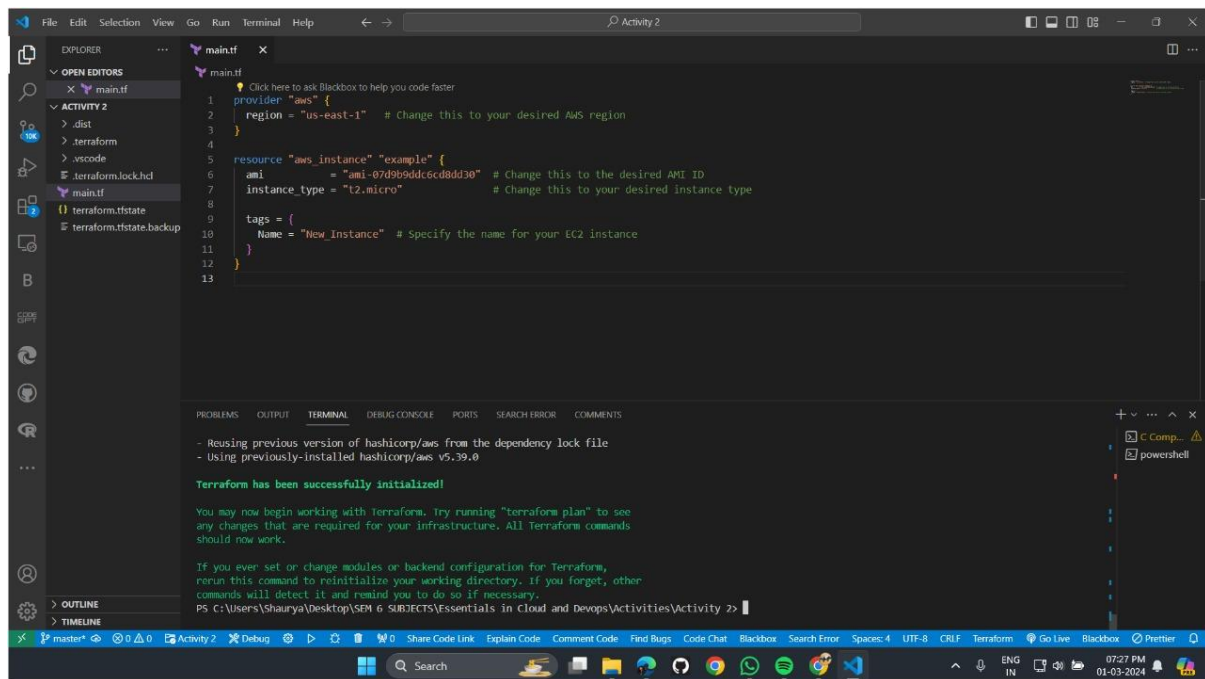
This command will remove all the resources defined in your Terraform configuration.

Output Screenshots:

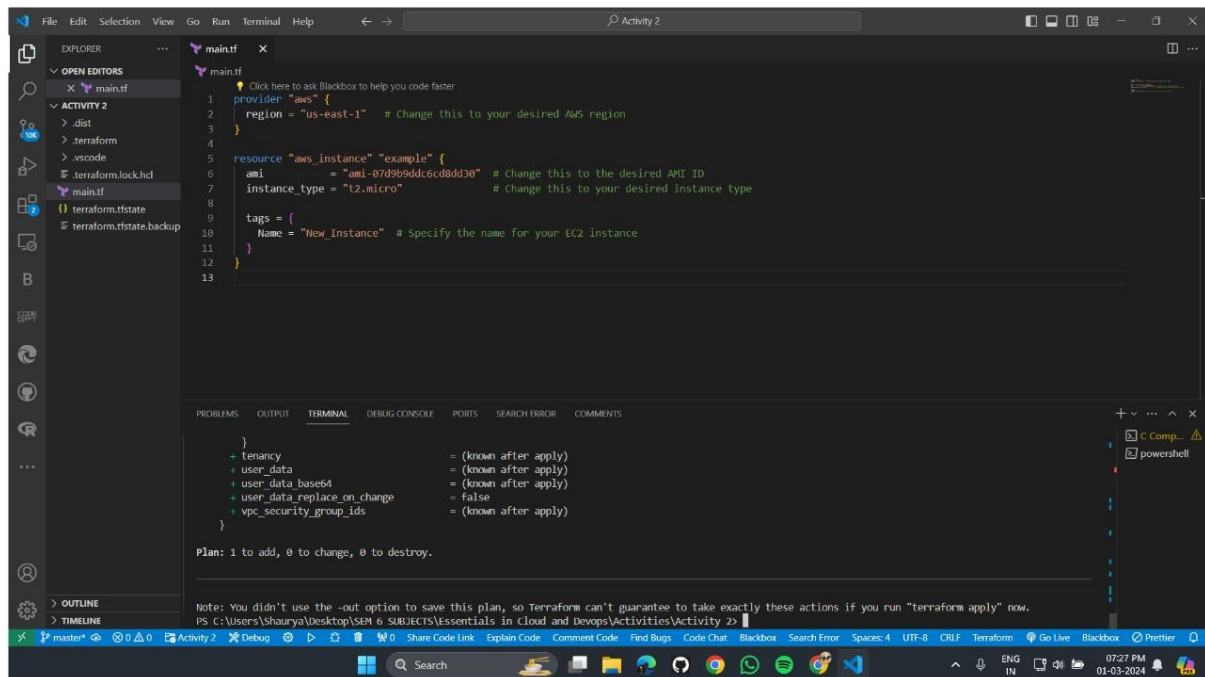
1. EC2 instance dashboard at the start:



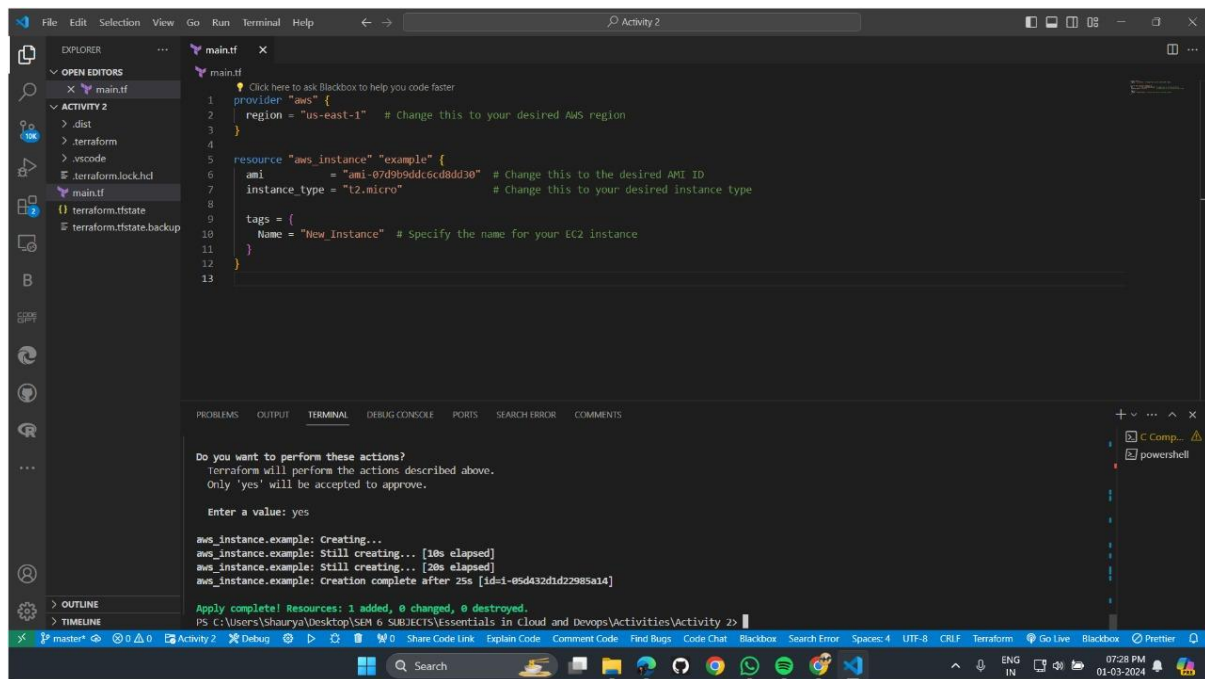
2. terraform init command:



3. terraform plan command:



4. terraform apply command:



The screenshot shows a Visual Studio Code editor with a Terraform configuration file named `main.tf`. The configuration defines an AWS provider and an `aws_instance` resource. The terminal window displays the output of the `terraform apply` command, showing the creation of the EC2 instance.

```
1 Click here to ask Blackbox to help you code faster
2 provider "aws" {
3   region = "us-east-1" # Change this to your desired AWS region
4 }
5
6 resource "aws_instance" "example" {
7   ami           = "ami-07d9b9ddc6cd8dd30" # Change this to the desired AMI ID
8   instance_type = "t2.micro" # Change this to your desired instance type
9
10  tags = {
11    Name = "New_Instance" # Specify the name for your EC2 Instance
12  }
13 }
```

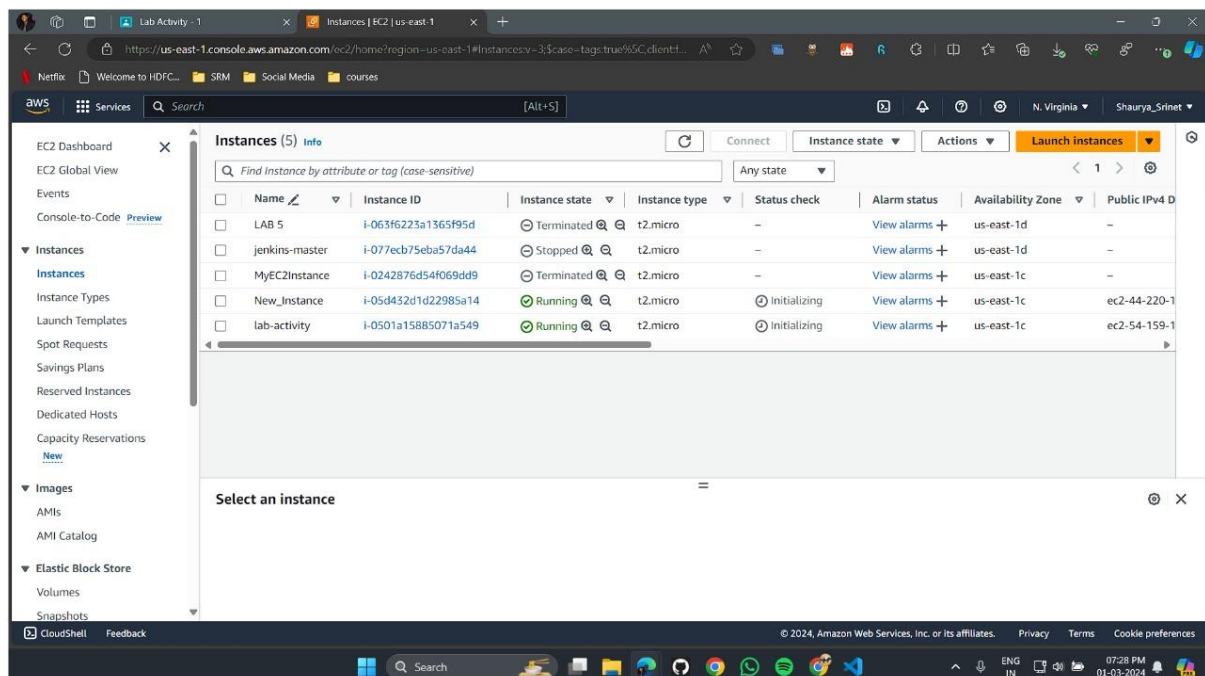
Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_instance.example: Creating...
aws_instance.example: Still creating... [10s elapsed]
aws_instance.example: Still creating... [20s elapsed]
aws_instance.example: Creation complete after 25s [id=i-05d432d1d22985a14]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

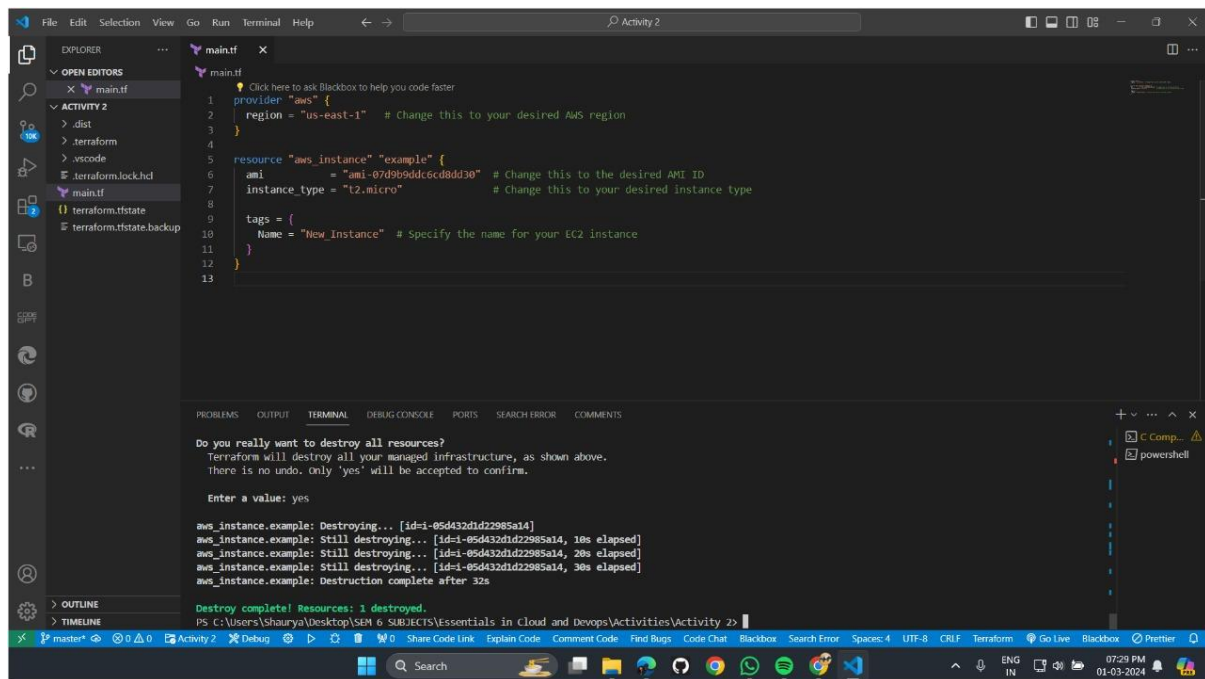
5. EC2 Dashboard after creating the instance:



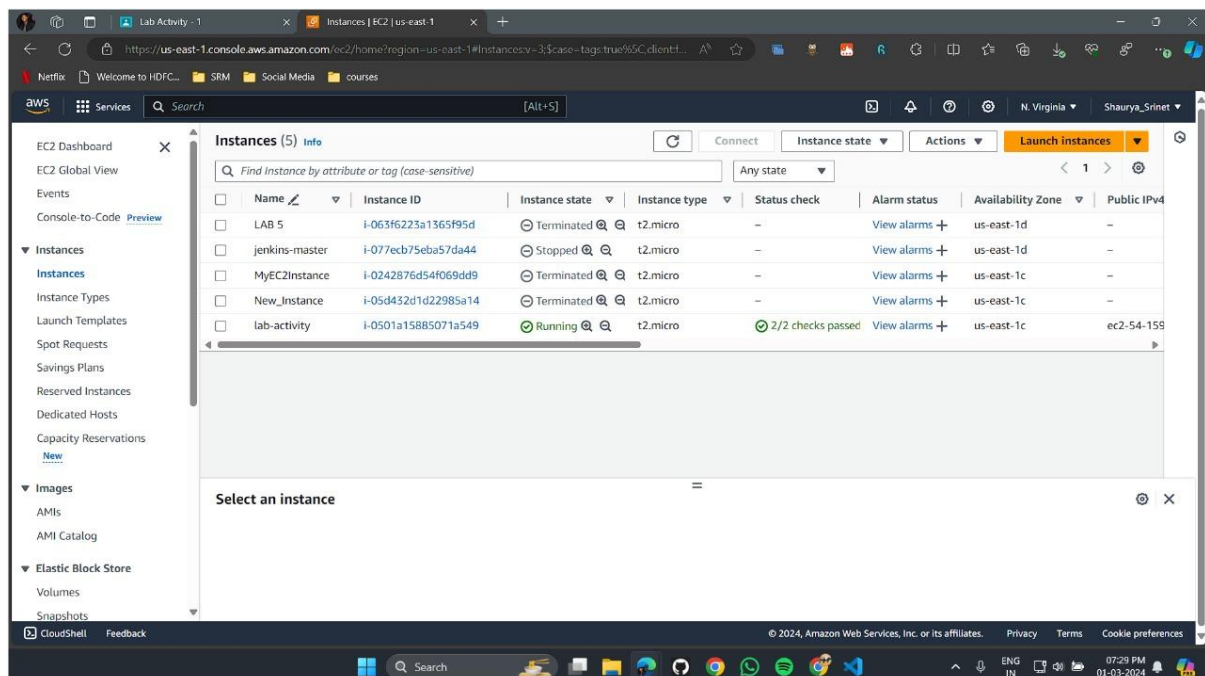
The screenshot shows the AWS Management Console for the `us-east-1` region. The `Instances` page is displayed, showing a list of EC2 instances. The instance `New_Instance` is in the `Running` state.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 D
LAB 5	i-063f6223a1365f95d	Terminated	t2.micro	-	View alarms	us-east-1d	-
jenkins-master	i-077ecb75eba57da44	Stopped	t2.micro	-	View alarms	us-east-1d	-
MyEC2Instance	i-0242876d54f069dd9	Terminated	t2.micro	-	View alarms	us-east-1c	-
New_Instance	i-05d432d1d22985a14	Running	t2.micro	Initializing	View alarms	us-east-1c	ec2-44-220-1
lab-activity	i-0501a15885071a549	Running	t2.micro	Initializing	View alarms	us-east-1c	ec2-54-159-1

6. terraform destroy command:



7. EC2 Dashboard after destroying the instance:



CREATE A VPN USING TERRAFORM

Procedure:

1. Install Terraform: If you haven't already, download and install Terraform from the official website: <https://www.terraform.io/downloads.html>

2. Set up AWS Credentials: Ensure you have AWS access key ID and secret access key with the required permissions to create VPN resources. You can set these up via AWS IAM.

3. Create a Terraform Configuration File: Open a text editor and create a new file named `main.tf`.

4. Write Terraform Configuration:

```
# Define AWS provider
provider "aws" {
  region = "us-east-1"    # Change this to your desired AWS region
}

# Create a Virtual Private Gateway
resource "aws_vpn_gateway" "example" {
  vpc_id = "vpc-0ba4add071b4c0bd9" # Replace this with your actual VPC ID

  tags = {
    Name = "MyVPNGateway" # Specify the name for your VPN gateway
  }
}

# Create a Customer Gateway
resource "aws_customer_gateway" "example" {
  bgp_asn      = 65000          # Replace this with your network's BGP
  ip_address   = "54.197.194.197" # Replace this with your network's
  public IP address
  type         = "ipsec.1"      # The type of VPN connection (ipsec.1 for
  IPsec)
}

# Create a VPN Connection
resource "aws_vpn_connection" "example" {
  customer_gateway_id = aws_customer_gateway.example.id
  vpn_gateway_id      = aws_vpn_gateway.example.id
}
```

```

    type          = "ipsec.1"      # The type of VPN connection (ipsec.1 for
IPSec)
    static_routes_only = false      # Whether the VPN connection uses static
routes only

    tags = {
      Name = "MyVPNConnection" # Specify the name for your VPN connection
    }
  }
}

```

Replace `"vpc-12345678"` with your VPC ID. Modify the `'bgp_asn'`, `'ip_address'`, and other parameters in the `'aws_customer_gateway'` resource according to your network setup.

5. Save the File: Save the `'main.tf'` file in a directory of your choice.

6. Initialize Terraform: Open a terminal or command prompt in the directory where your `'main.tf'` file is located and run: **terraform init**

7. Review and Apply Changes: Before applying the changes, review what Terraform plans to do by running: **terraform plan**

If everything looks good, apply the changes: **terraform apply**

Type **'yes'** when prompted to confirm.

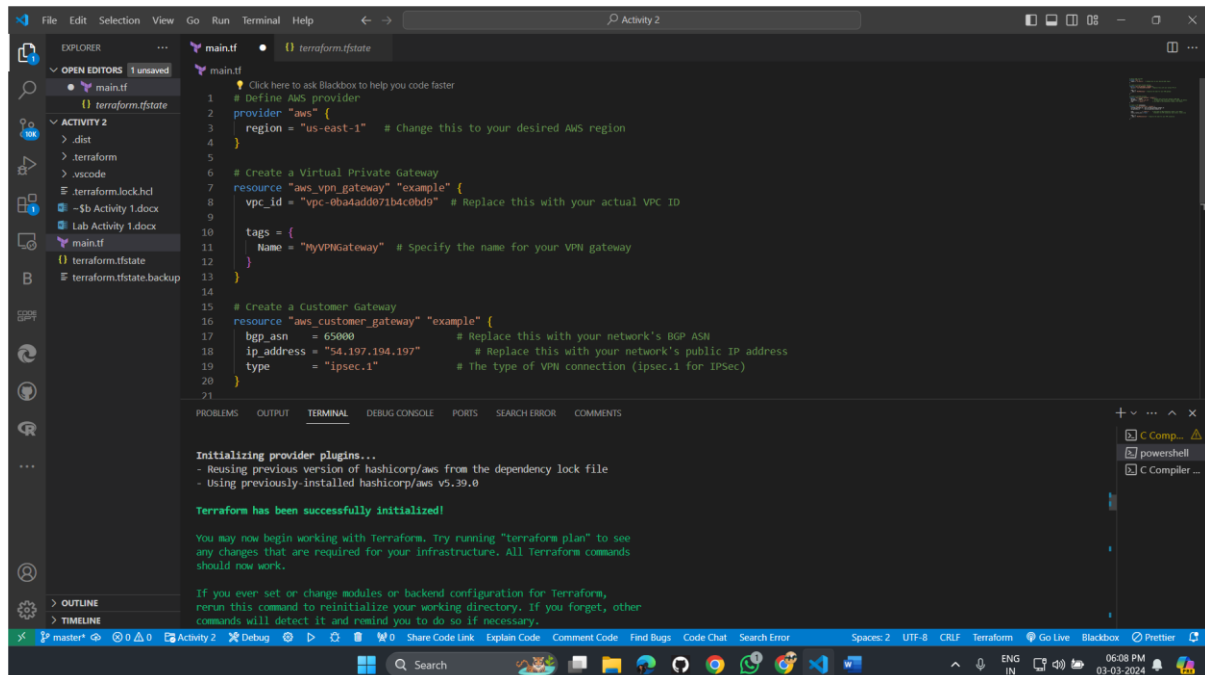
8. Verify: Once the apply command completes successfully, verify that your VPN resources are created by logging into the AWS Management Console, navigating to the VPC service, and checking the VPN connections and gateways.

9. Clean Up: To clean up resources created by Terraform, you can run: **terraform destroy**

This command will remove all the resources defined in your Terraform configuration.

Output Screenshots:

1. terraform init command:



```
main.tf
1 Click here to ask Blackbox to help you code faster
2 # Define AWS provider
3 provider "aws" {
4   region = "us-east-1" # Change this to your desired AWS region
5 }
6 # Create a Virtual Private Gateway
7 resource "aws_vpn_gateway" "example" {
8   vpc_id = "vpc-0ba4add071b4c0bd9" # Replace this with your actual VPC ID
9 }
10 tags = {
11   Name = "MyVPNGateway" # Specify the name for your VPN gateway
12 }
13 }
14 # Create a Customer Gateway
15 resource "aws_customer_gateway" "example" {
16   bgp_asn = 65000 # Replace this with your network's BGP ASN
17   ip_address = "54.197.194.197" # Replace this with your network's public IP address
18   type = "ipsec.1" # The type of VPN connection (ipsec.1 for IPsec)
19 }
20 }
21
```

Initializing provider plugins...

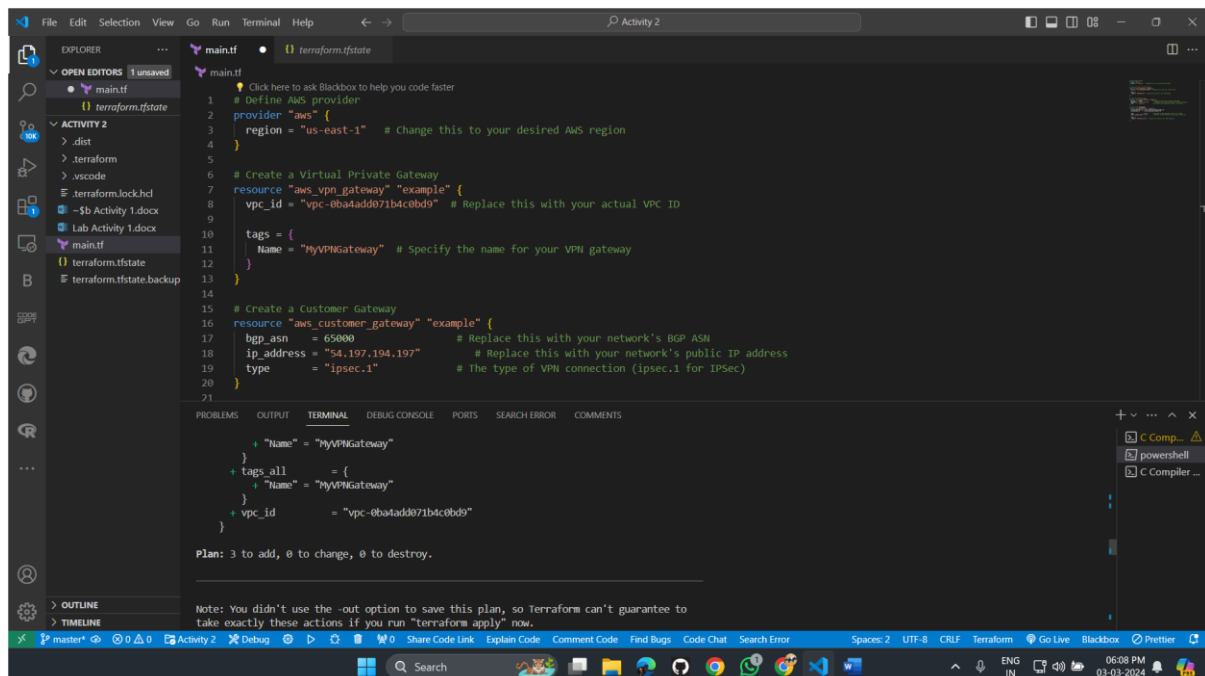
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.39.0

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.

2. terraform plan command:

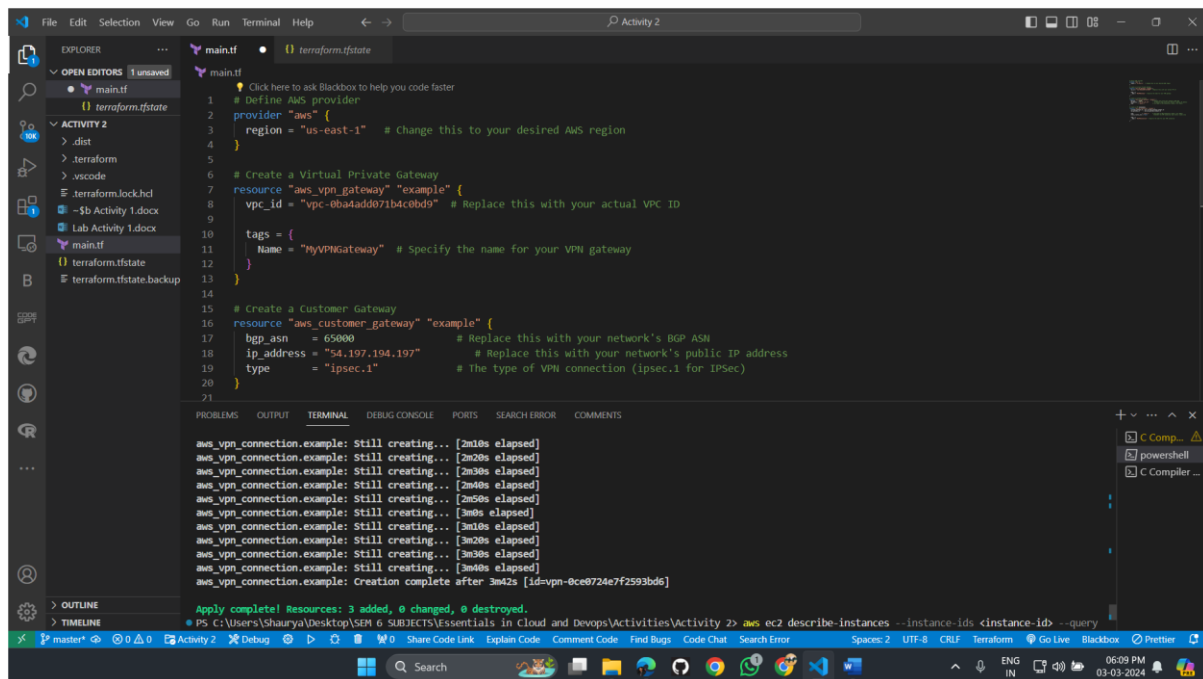


```
main.tf
1 Click here to ask Blackbox to help you code faster
2 # Define AWS provider
3 provider "aws" {
4   region = "us-east-1" # Change this to your desired AWS region
5 }
6 # Create a Virtual Private Gateway
7 resource "aws_vpn_gateway" "example" {
8   vpc_id = "vpc-0ba4add071b4c0bd9" # Replace this with your actual VPC ID
9 }
10 tags = {
11   Name = "MyVPNGateway" # Specify the name for your VPN gateway
12 }
13 }
14 # Create a Customer Gateway
15 resource "aws_customer_gateway" "example" {
16   bgp_asn = 65000 # Replace this with your network's BGP ASN
17   ip_address = "54.197.194.197" # Replace this with your network's public IP address
18   type = "ipsec.1" # The type of VPN connection (ipsec.1 for IPsec)
19 }
20 }
21
```

Plan: 3 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.

3. terraform apply command:

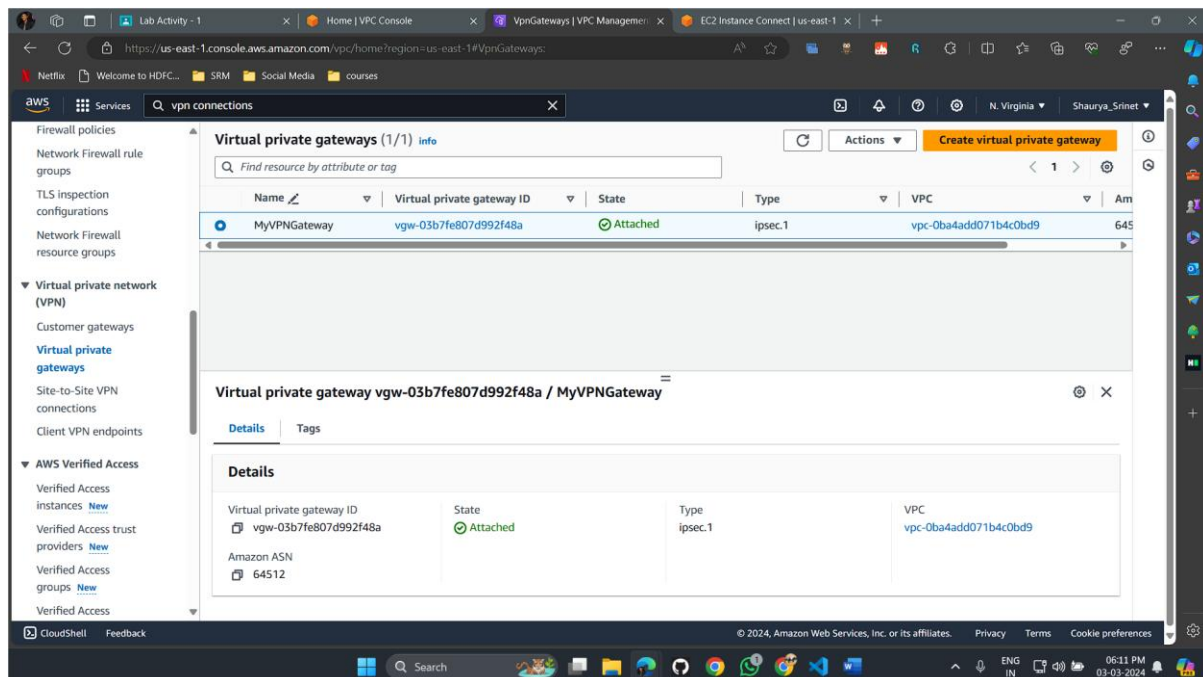


```
main.tf
1 # Click here to ask Blackbox to help you code faster
2 # Define AWS provider
3 provider "aws" {
4   region = "us-east-1" # Change this to your desired AWS region
5 }
6 # Create a Virtual Private Gateway
7 resource "aws_vpn_gateway" "example" {
8   vpc_id = "vpc-0ba4add071b4c0bd9" # Replace this with your actual VPC ID
9   tags = {
10     Name = "MyVPNGateway" # Specify the name for your VPN gateway
11   }
12 }
13
14 # Create a Customer Gateway
15 resource "aws_customer_gateway" "example" {
16   bgp_asn = 65000 # Replace this with your network's BGP ASN
17   ip_address = "54.197.194.197" # Replace this with your network's public IP address
18   type = "ipsec.1" # The type of VPN connection (ipsec.1 for IPsec)
19 }
20
21
```

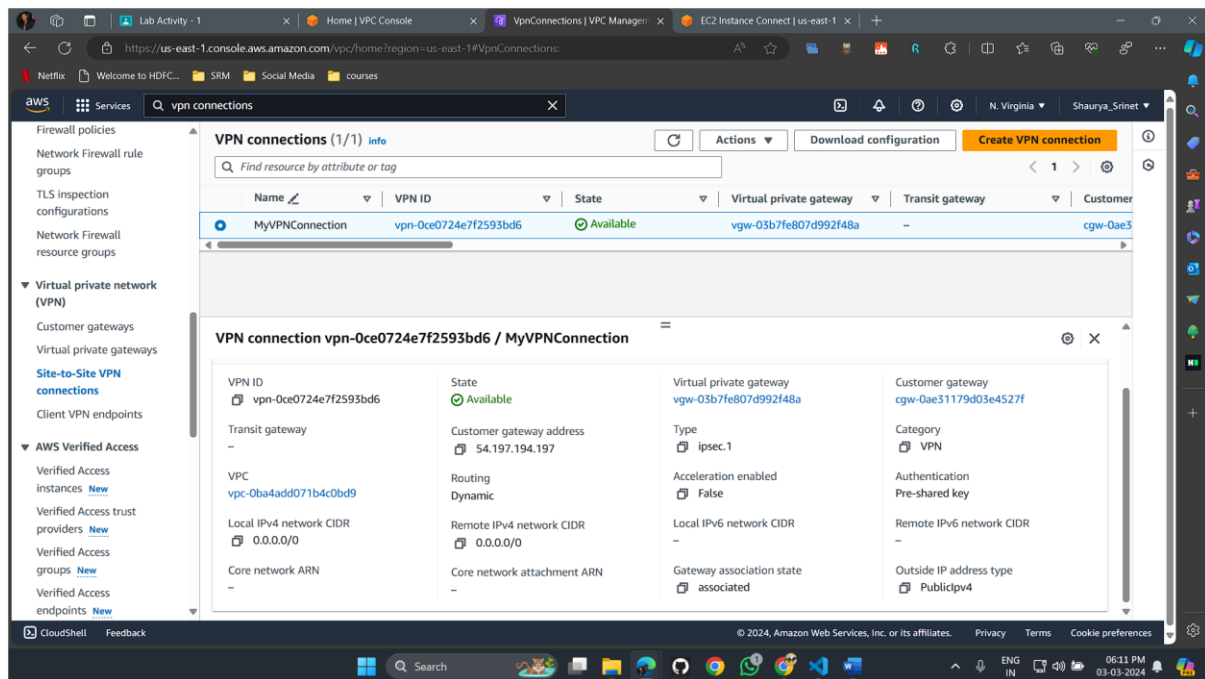
```
aws_vpn_gateway.example: Still creating... [2m10s elapsed]
aws_vpn_gateway.example: Still creating... [2m20s elapsed]
aws_vpn_gateway.example: Still creating... [2m30s elapsed]
aws_vpn_gateway.example: Still creating... [2m40s elapsed]
aws_vpn_gateway.example: Still creating... [2m50s elapsed]
aws_vpn_gateway.example: Still creating... [3m0s elapsed]
aws_vpn_gateway.example: Still creating... [3m10s elapsed]
aws_vpn_gateway.example: Still creating... [3m20s elapsed]
aws_vpn_gateway.example: Still creating... [3m30s elapsed]
aws_vpn_gateway.example: Still creating... [3m40s elapsed]
aws_vpn_gateway.example: Creation complete after 3m42s [id=vpn-0ce0724e7f2593bd5]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
PS C:\Users\Shaurya\Desktop\SEM 6 SUBJECTS\Essentials in Cloud and Devops\Activities\Activity 2> aws ec2 describe-instances --instance-ids <instance-id> --query
```

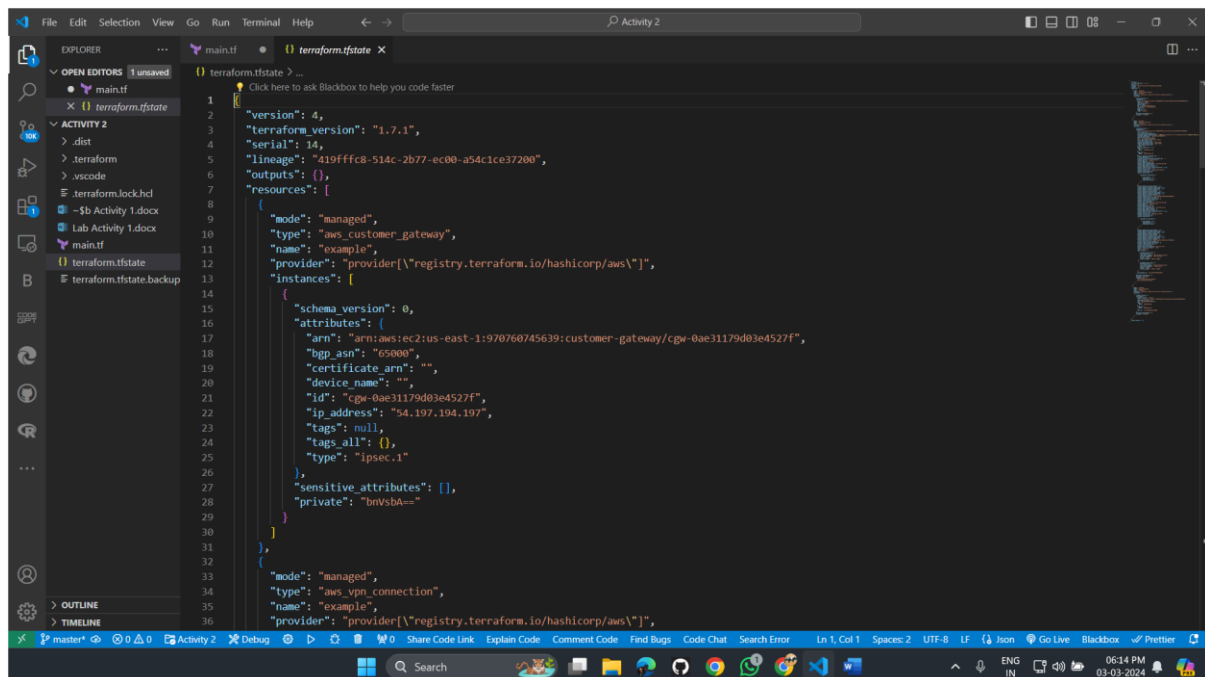
4. Virtual Private Gateways Dashboard after creating:



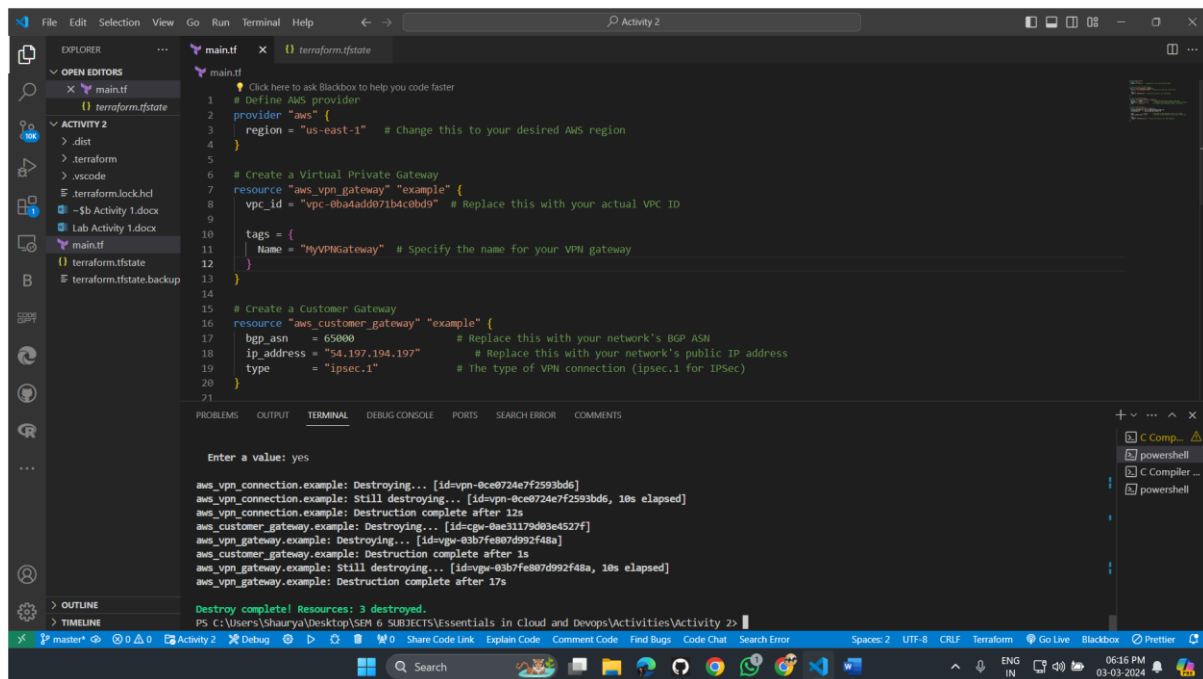
5. VPN Connections Dashboard after creating:



6. Terraform State File (terraform.tfstate):

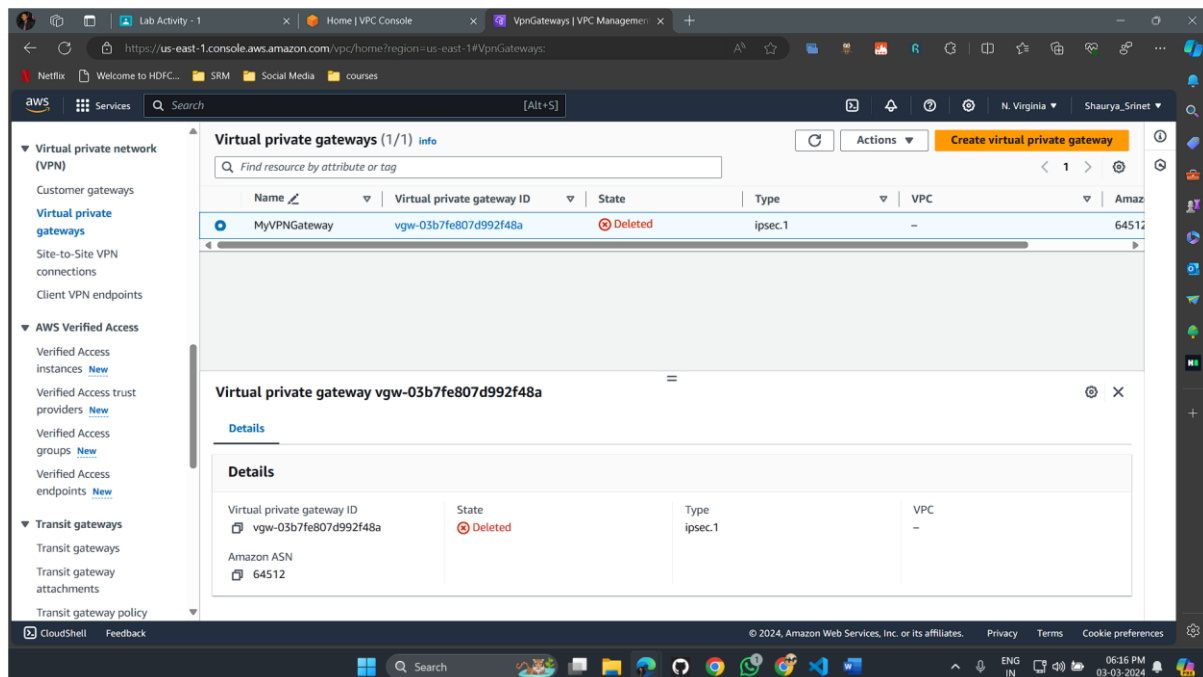


7. terraform destroy command:



```
main.tf x terraform.tfstate
main.tf
1 Click here to ask Blackbox to help you code faster
2 # Define AWS provider
3 provider "aws" {
4   region = "us-east-1" # Change this to your desired AWS region
5 }
6 # Create a Virtual Private Gateway
7 resource "aws_vpn_gateway" "example" {
8   vpc_id = "vpc-0ba4add071b4c0bd9" # Replace this with your actual VPC ID
9   tags = {
10     Name = "MyVPNGateway" # Specify the name for your VPN gateway
11   }
12 }
13
14 # Create a Customer Gateway
15 resource "aws_customer_gateway" "example" {
16   bgp_asn = 65000 # Replace this with your network's BGP ASN
17   ip_address = "54.197.194.197" # Replace this with your network's public IP address
18   type = "ipsec.1" # The type of VPN connection (ipsec.1 for IPsec)
19 }
20
21
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE PORTS SEARCH ERROR COMMENTS
Enter a value: yes
aws_vpn_connection.example: Destroying... [id=vpn-0ce0724e7f2593bd6]
aws_vpn_connection.example: Still destroying... [id=vpn-0ce0724e7f2593bd6, 10s elapsed]
aws_vpn_connection.example: Destruction complete after 12s
aws_customer_gateway.example: Destroying... [id=cgw-0a31175d03e4527f]
aws_vpn_gateway.example: Destroying... [id=vgw-03b7fe807d992f48a]
aws_customer_gateway.example: Destruction complete after 1s
aws_vpn_gateway.example: Still destroying... [id=vgw-03b7fe807d992f48a, 10s elapsed]
aws_vpn_gateway.example: Destruction complete after 17s
Destroy complete! Resources: 3 destroyed.
PS C:\Users\Shaurya\Desktop\SEM 6 SUBJECTS\Essentials in Cloud and Devops\Activities\Activity 2>
```

8. Virtual Private Gateways Dashboard after destroying:



9. VPN Connections Dashboard after destroying:

The screenshot displays the AWS VPN Connections dashboard in the AWS Management Console. The left-hand navigation pane includes sections for Virtual private network (VPN), AWS Verified Access, and Transit gateways. The main content area shows a table of VPN connections with one entry, 'MyVPNConnection', which is in a 'Deleted' state. Below the table, the details for the selected VPN connection 'vpn-0ce0724e7f2593bd6' are shown, including its ID, state, associated virtual private gateway, customer gateway, and various network settings.

VPN connections (1/1)

Name	VPN ID	State	Virtual private gateway	Transit gateway	Customer gateway
MyVPNConnection	vpn-0ce0724e7f2593bd6	Deleted	vgw-03b7fe807d992f48a	-	cgw-0ae31179d03e4527f

VPN connection vpn-0ce0724e7f2593bd6

Details

VPN ID	vpn-0ce0724e7f2593bd6	State	Deleted	Virtual private gateway	vgw-03b7fe807d992f48a	Customer gateway	cgw-0ae31179d03e4527f
Transit gateway	-	Customer gateway address	-	Type	ipsec.1	Category	VPN
VPC	-	Routing	Dynamic	Acceleration enabled	False	Authentication	Pre-shared key
Local IPv4 network CIDR	-	Remote IPv4 network CIDR	-	Local IPv6 network CIDR	-	Remote IPv6 network CIDR	-