Gautham Srinivasan

IT Analyst - DTCC

18 Nov 2024

Setting Up AWS Infrastructure Using Terraform

Project Overview

This documentation outlines the steps to set up AWS infrastructure using Terraform, including creating an EC2 instance, configuring a VPC, and deploying a sample application.

Prerequisites

- AWS account with administrative access.
- Key pair and security group configured in AWS.
- Basic understanding of Terraform..

Procedures

Step 1. Create an EC2 Instance

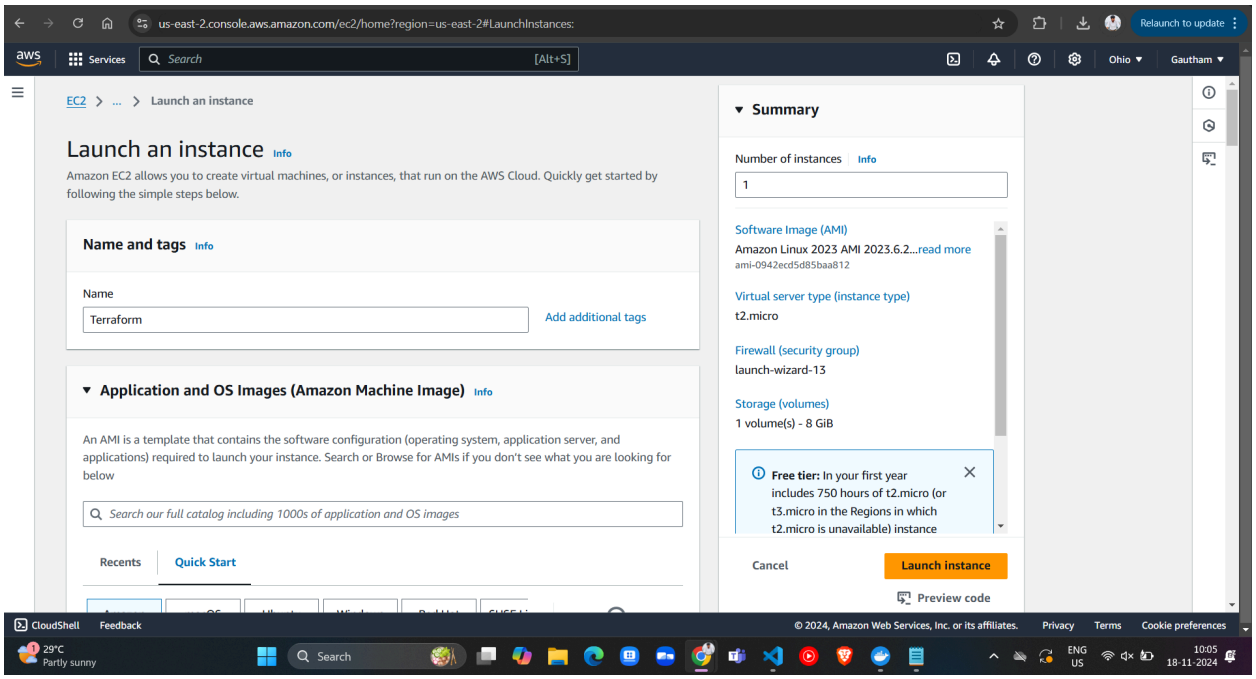
Navigate to the **EC2** page in the AWS Management Console.

Click on **Launch Instance**.

- Provide a name for your instance.
- Select **Amazon Linux 2023 AMI**.

Choose the **key pair** you created earlier.

Assign a **security group** that allows **All TCP traffic** and has an open **CIDR block**.



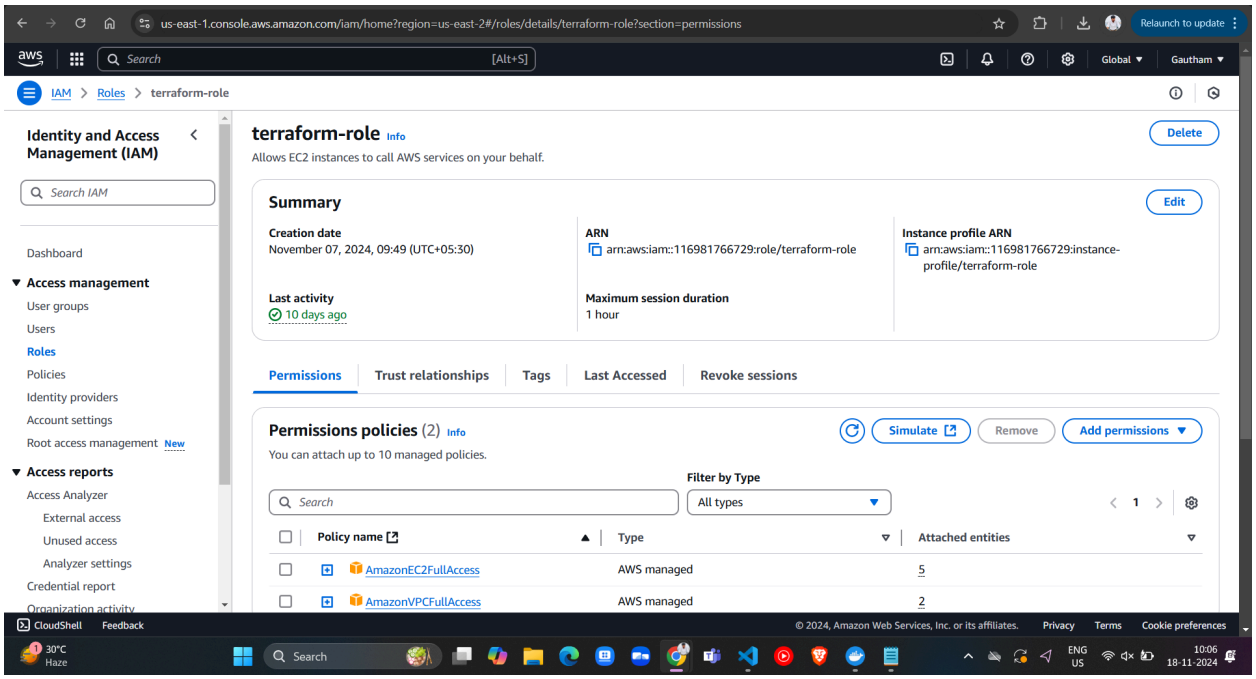
Step 2: Create a Role and Attach to the Instance

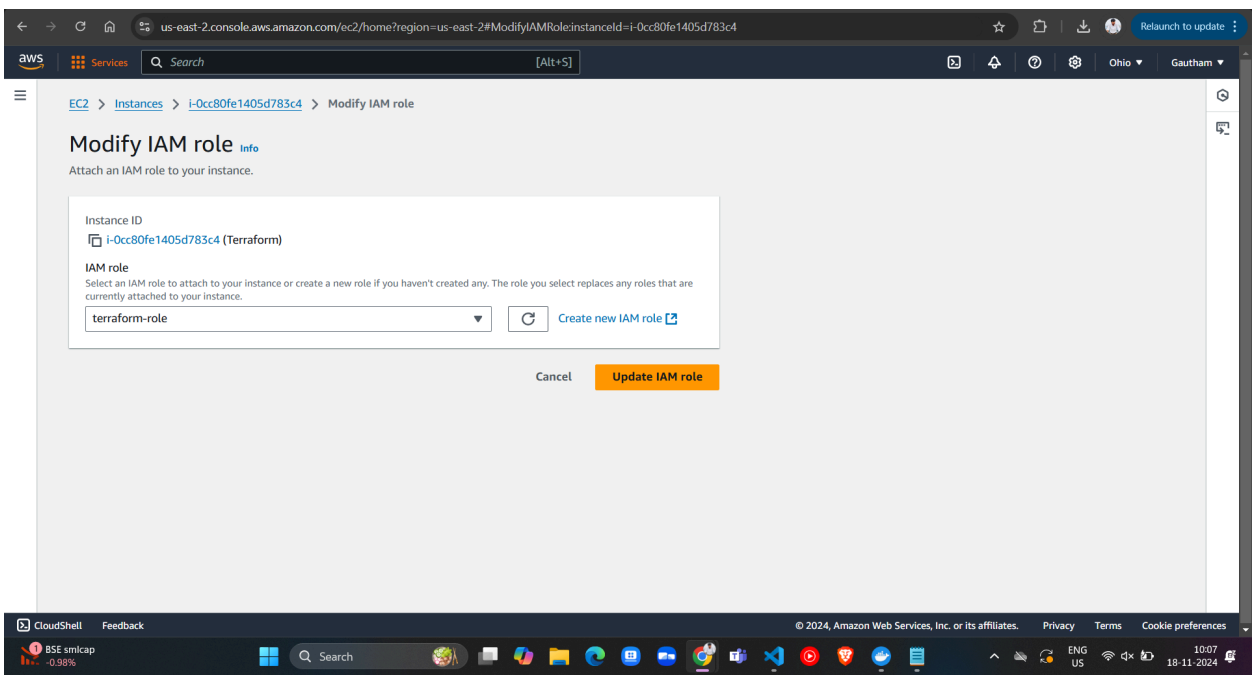
Navigate to the **IAM Roles** section in the AWS Console.

- Create a role with the following permissions:
  - `AmazonEC2FullAccess`
  - `AmazonVPCFullAccess`
- Name the role (e.g., `terraform-role`).

Go back to the EC2 page and select your instance.

- Navigate to **Actions** > **Security** > **Modify IAM Role**.
- Attach the created IAM role and click **Update**





Step 3: Install Terraform

Open the EC2 instance terminal.

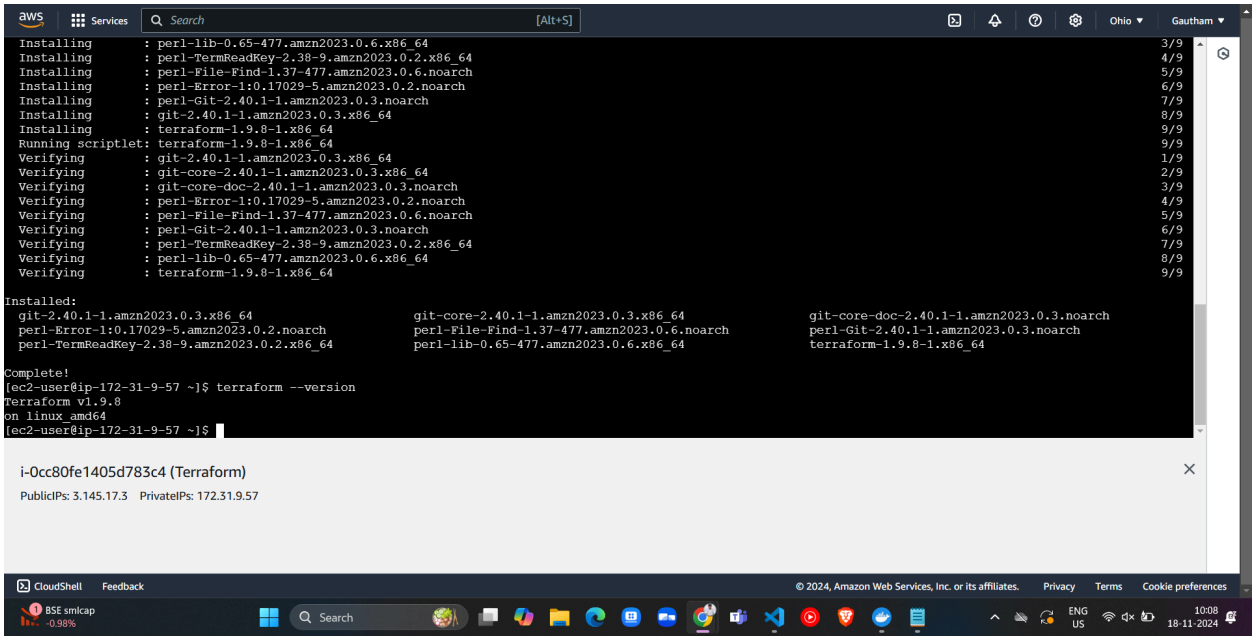
- Run the following commands to install Terraform:
- `sudo yum install -y yum-utils`

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

sudo yum -y install terraform
```

Verify the installation using:

```
terraform -version
```



Step 4: Write and Apply Terraform Code

- Write and Apply Terraform Code

```
terraform {

  required_providers {

    aws = {

      source = "hashicorp/aws"

      version = "5.74.0"

    }

  }

}
```

```
provider "aws" {
  region = "us-east-1" # Replace with your region
}

resource "aws_vpc" "vpc_gau" {
  cidr_block = "10.0.0.0/16"
  instance_tenancy = "default"
  tags = {
    Name = "vpc-gau"
  }
}

resource "aws_subnet" "pub_sub" {
  vpc_id = aws_vpc.vpc_gau.id
  cidr_block = "10.0.20.0/24"
  tags = {
    Name = "pub_sub"
  }
}

resource "aws_internet_gateway" "my_new_gw" {
  vpc_id = aws_vpc.vpc_gau.id
```

```
  tags = {
    Name = "my_new_gw"
  }
}

resource "aws_route_table" "pub_rt" {
  vpc_id = aws_vpc.vpc_gau.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.my_new_gw.id
  }
  tags = {
    Name = "pub-rt"
  }
}

resource "aws_route_table_association" "pub_assoc" {
  subnet_id = aws_subnet.pub_sub.id
  route_table_id = aws_route_table.pub_rt.id
}

resource "aws_security_group" "pub_sg" {
  name = "allow_all"
```

```
  vpc_id = aws_vpc.vpc_gau.id

  tags = {

    Name = "pub_sg"

  }

}



resource "aws_instance" "my_instance" {

  ami = "ami-06b21ccaeff8cd686"

  instance_type = "t2.micro"

  subnet_id = aws_subnet.pub_sub.id

  security_groups = [aws_security_group.pub_sg.id]

  associate_public_ip_address = true

  user_data = <<-EOF

  #!/bin/bash

  yum update -y

  yum install -y httpd

  echo "Hello, This is a test page" > /var/www/html/index.html

  systemctl enable httpd

  systemctl start httpd

  EOF

  tags = {

    Name = "my_instance"

  }
```

} Step 5: Initialize and Deploy Resources Using Terraform

Open the terminal in the folder containing your Terraform code.

- Run `terraform init` to initialize Terraform.

```
you run "terraform init" in the future.

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.
[root@ip-172-31-9-57 ~]# nano main.tf
[root@ip-172-31-9-57 ~]# terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.74.0

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.
[root@ip-172-31-9-57 ~]#
```

i-0cc80fe1405d783c4 (Terraform)
PublicIPs: 3.145.17.3 PrivateIPs: 172.31.9.57

- Use `terraform plan` to preview the resources that will be created.

```
    + from_port              = 443
    + id                     = (known after apply)
    + ip_protocol            = "tcp"
    + security_group_id      = (known after apply)
    + security_group_rule_id = (known after apply)
    + tags_all               = {}
    + to_port                = 443
  }

  # aws_vpc_security_group_ingress_rule.allow_ssh_ipv4 will be created
  + resource "aws_vpc_security_group_ingress_rule" "allow_ssh_ipv4" {
    + arn                    = (known after apply)
    + cidr_ipv4              = "10.0.0.0/16"
    + from_port              = 22
    + id                     = (known after apply)
    + ip_protocol            = "tcp"
    + security_group_id      = (known after apply)
    + security_group_rule_id = (known after apply)
    + tags_all               = {}
    + to_port                = 22
  }

Plan: 10 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.
[root@ip-172-31-9-57 ~]#
```

i-0cc80fe1405d783c4 (Terraform)
PublicIPs: 3.145.17.3 PrivateIPs: 172.31.9.57

- Run `terraform apply` to create the resources.

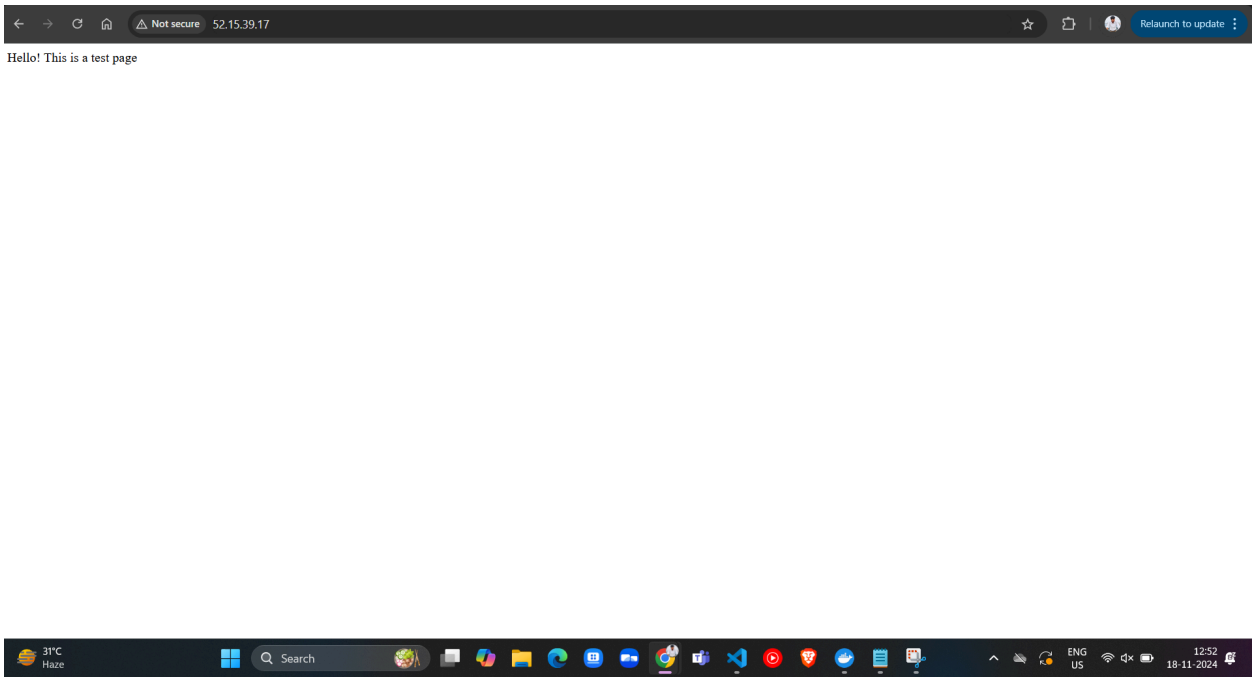
Step 6: Create a Role and Attach to the Instance

Open the **AWS Management Console**.

- Check if the EC2 instance, VPC, and other resources are created.

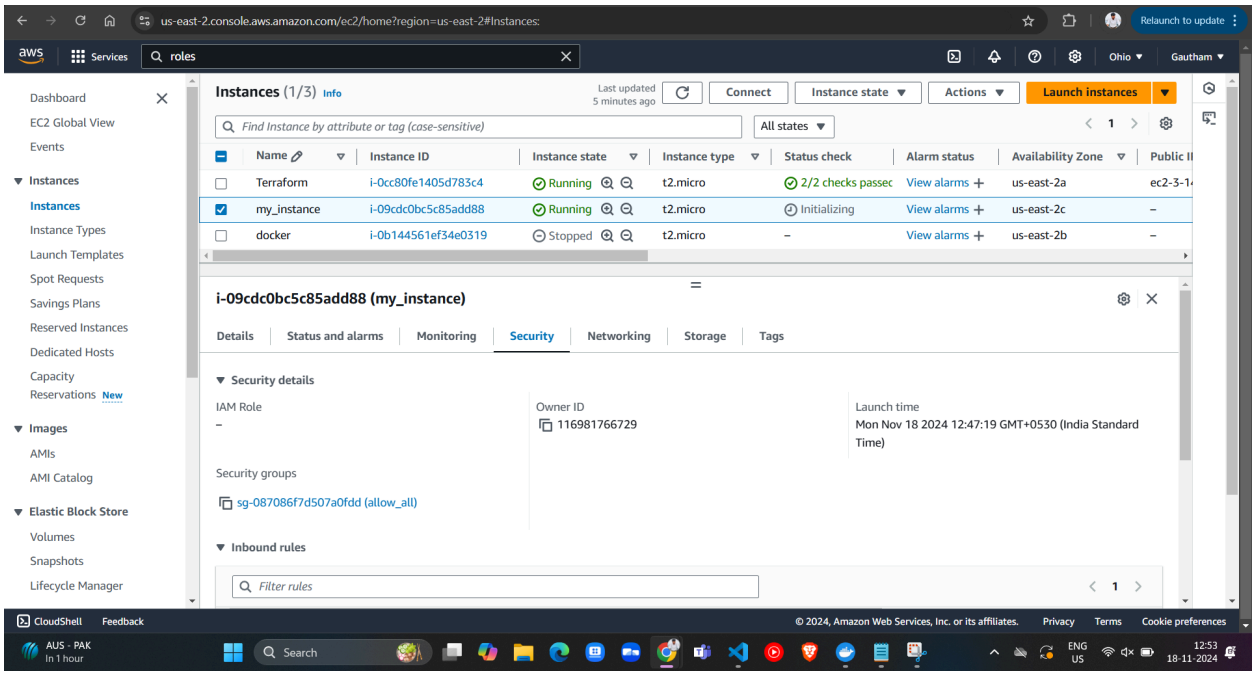
Access the public IP of the EC2 instance in your browser.

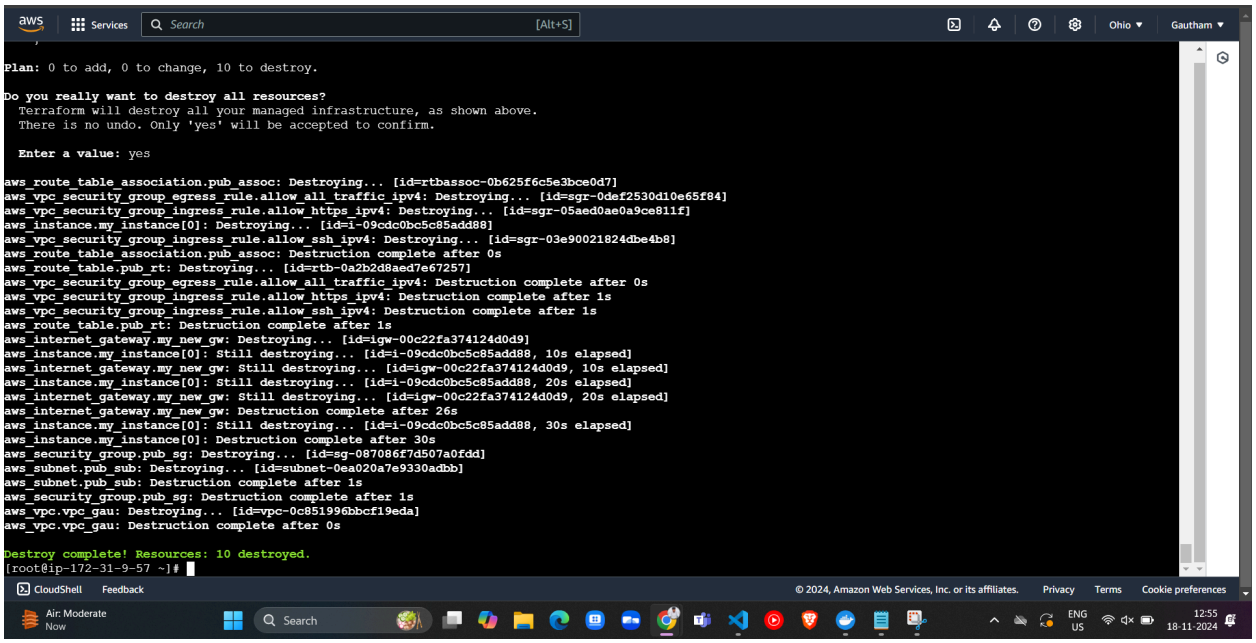
- The output should display:



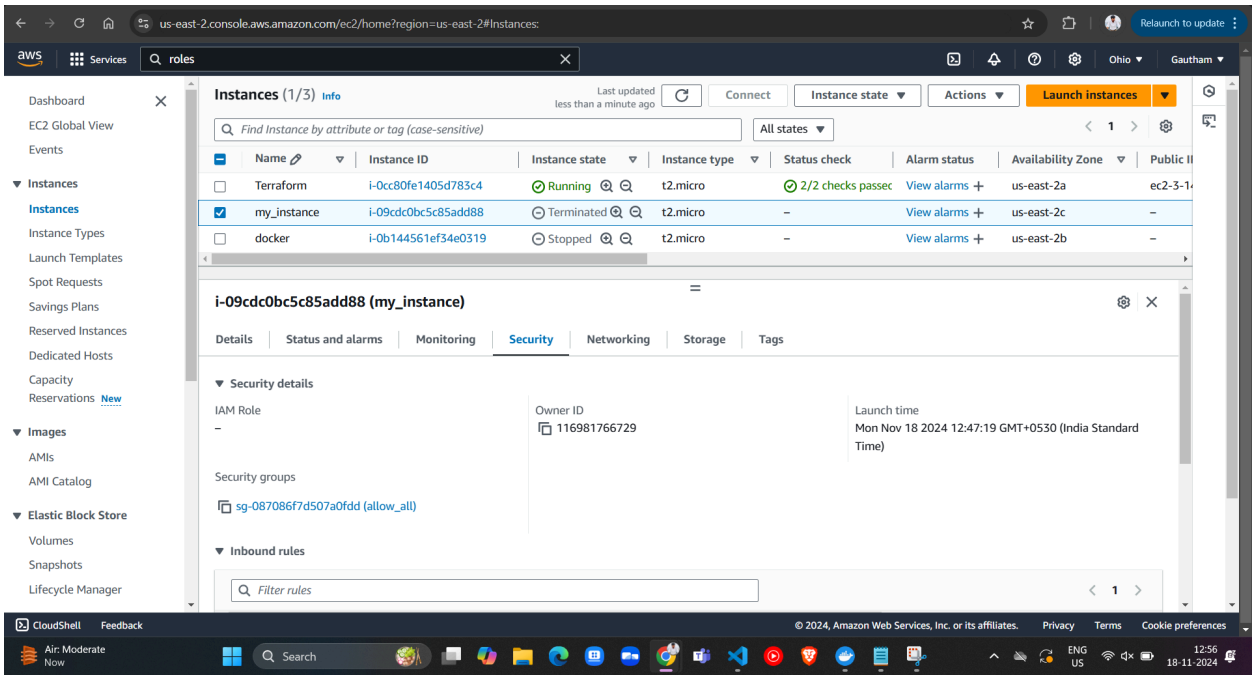
Step 6: Destroy Resources

- Run `terraform destroy` to remove all created resources.





- Go to the AWS Console and terminate the EC2 instance.



Conclusion

This documentation demonstrates setting up AWS infrastructure with Terraform for creating and managing a secure, scalable EC2 instance and associated resources. Following these steps ensures effective infrastructure as code practices.