Lab Exercise 4–Provisioning an EC2 Instance on AWS

Name-Manish kumar

SAP ID- 500119723

Batch-2

Exercise Steps:

Step 1: Create a New Directory:

Create a new directory for your Terraform configuration:

“Terraform-Demo”

Step 2: Create Terraform Configuration File (main.tf):

Create a file named main.tf with the following content:

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.31.0"
    }
  }
}
provider "aws" {
  region     = "ap-south-1"
  access_key = "your IAM access key"
  secret_key = "your secret access key"
}
```

This script defines an AWS provider and provisions an EC2 instance.

Step 3: Initialize Terraform:

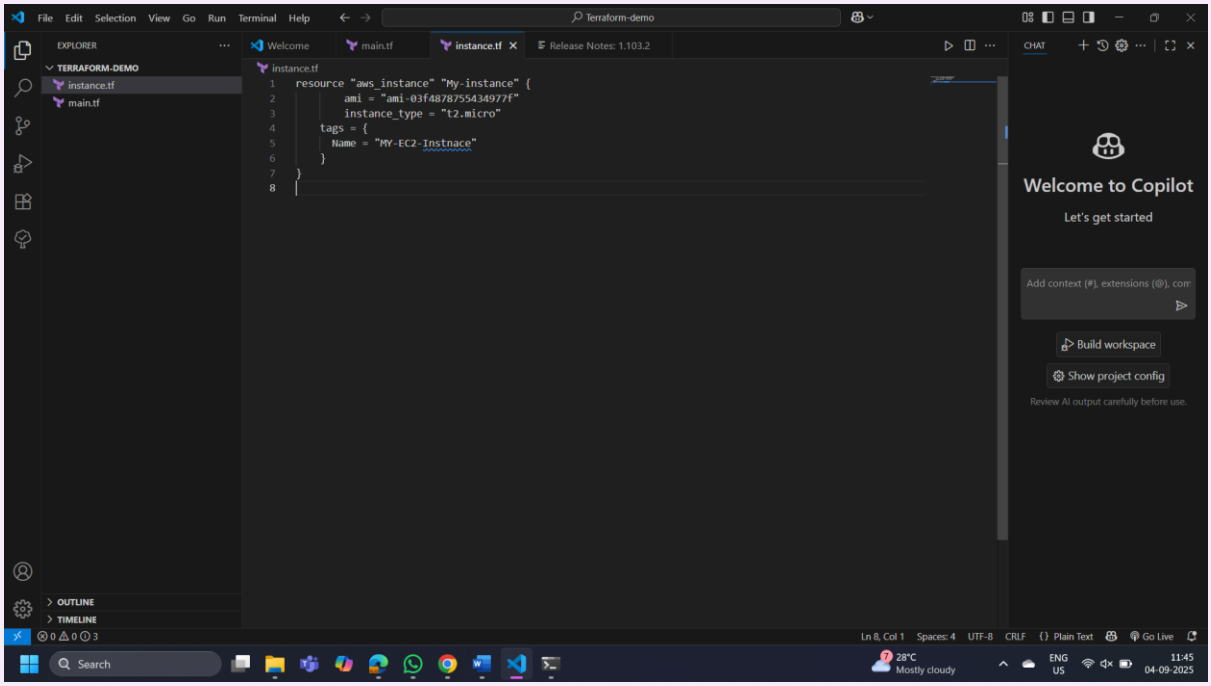
Run the following command to initialize your Terraform working directory:

```
terraform init
```

Step 4: Create Terraform Configuration File for EC2 instance (instance.tf):

Create a file named instnace.tf with the following content:

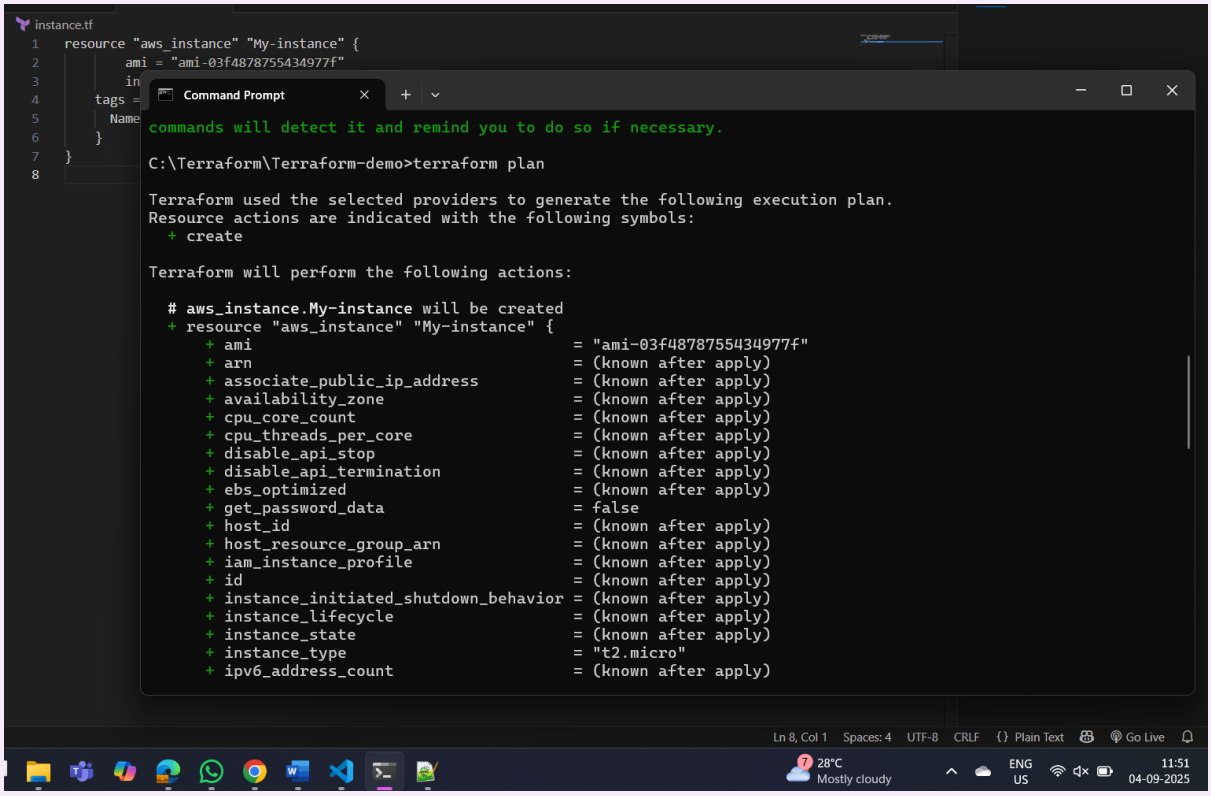
```
resource "aws_instance" "My-instance" {

      ami = "ami-03f4878755434977f"

      instance_type = "t2.micro"

   tags = {

     Name = "MY-EC2-Instnace"

   }

}
```



Step 5: Review Plan:

Run the following command to see what Terraform will do:

```
terraform plan
```



Review the plan to ensure it aligns with your expectations.

Step 6: Apply Changes:

Apply the changes to create the AWS resources:

```
terraform apply
```

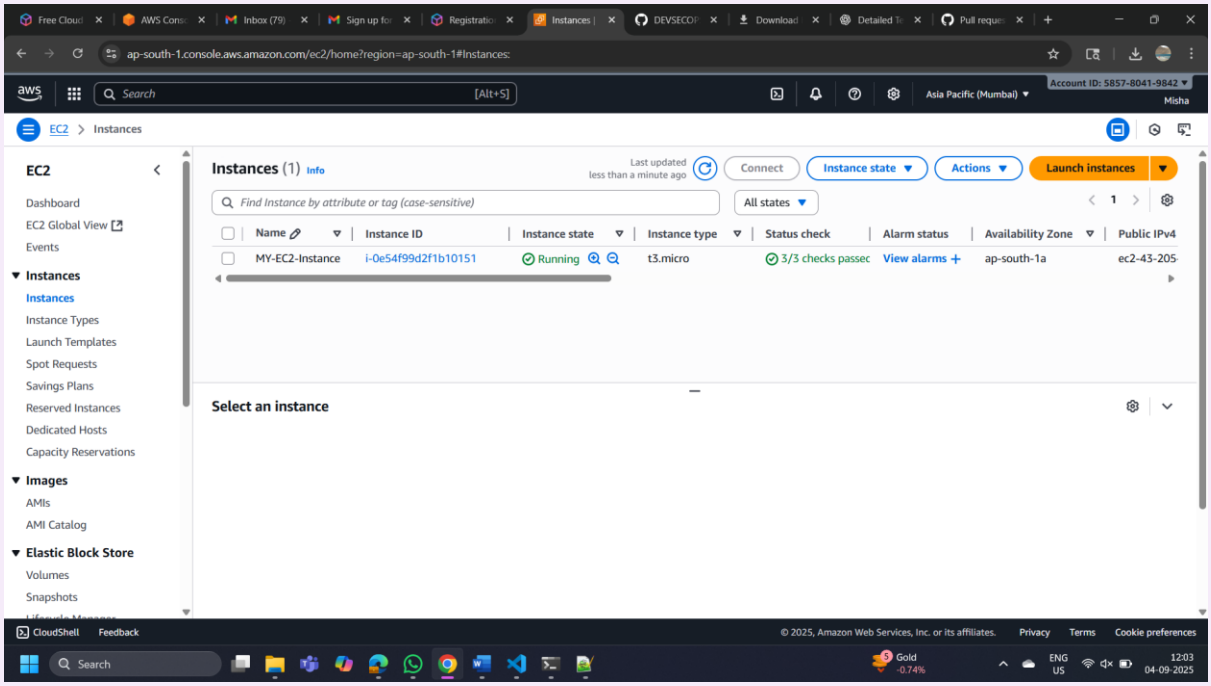
```
C:\Terraform\Terraform-demo>terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are
indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_instance.My-instance will be created
  + resource "aws_instance" "My-instance" {
      + ami                                  = "ami-03f4878755434977f"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t3.micro"
```

```
    }

Plan: 1 to add, 0 to change, 0 to destroy.

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.My-instance: Creating...
aws_instance.My-instance: Still creating... [00m10s elapsed]
aws_instance.My-instance: Creation complete after 15s [id=i-0e54f99d2f1b10151]

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

C:\Terraform\Terraform-demo>
```

Type yes when prompted.

Step 7: Verify Resources:

After the terraform apply command completes, log in to your AWS Management Console and navigate to the EC2 dashboard. Verify that the EC2 instance has been created.



Step 8: Cleanup Resources:

When you are done experimenting, run the following command to destroy the created resources:

```
terraform destroy
```

```
C:\Terraform\Terraform-demo>terraform destroy
aws_instance.My-instance: Refreshing state... [id=i-0e54f99d2f1b10151]

Terraform used the selected providers to generate the following execution plan. Resource actions are
indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  # aws_instance.My-instance will be destroyed
  - resource "aws_instance" "My-instance" {
      - ami                                  = "ami-03f4878755434977f" -> null
      - arn                                  = "arn:aws:ec2:ap-south-1:585780419842:instance/i-0e54f99d2f1b1
0151" -> null
      - associate_public_ip_address          = true -> null
      - availability_zone                    = "ap-south-1a" -> null
      - cpu_core_count                       = 1 -> null
      - cpu_threads_per_core                 = 2 -> null
      - disable_api_stop                     = false -> null
      - disable_api_termination              = false -> null
      - ebs_optimized                        = false -> null
      - get_password_data                    = false -> null
      - hibernation                          = false -> null
      - id                                   = "i-0e54f99d2f1b10151" -> null
      - instance_initiated_shutdown_behavior = "stop" -> null
      - instance_state                       = "running" -> null
      - instance_type                        = "t3.micro" -> null
      - ipv6_address_count                   = 0 -> null
      - ipv6_addresses                       = [] -> null
      - monitoring                           = false -> null
```

```
          - volume_id             = "vol-0ed9e2c8e4022f1f2" -> null
          - volume_size           = 8 -> null
          - volume_type           = "gp2" -> null
            # (1 unchanged attribute hidden)
        }
    }

Plan: 0 to add, 0 to change, 1 to destroy.

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

aws_instance.My-instance: Destroying... [id=i-0e54f99d2f1b10151]
aws_instance.My-instance: Still destroying... [id=i-0e54f99d2f1b10151, 00m10s elapsed]
aws_instance.My-instance: Still destroying... [id=i-0e54f99d2f1b10151, 00m20s elapsed]
aws_instance.My-instance: Still destroying... [id=i-0e54f99d2f1b10151, 00m30s elapsed]
aws_instance.My-instance: Still destroying... [id=i-0e54f99d2f1b10151, 00m40s elapsed]
aws_instance.My-instance: Destruction complete after 41s

Destroy complete! Resources: 1 destroyed.

C:\Terraform\Terraform-demo>
```

Type yes when prompted.

Notes:

Customize the instance.tf file to provision different AWS resources.

Explore the Terraform AWS provider documentation for additional AWS resources and configuration options.

Always be cautious when running terraform destroy to avoid accidental resource deletion.

This exercise provides a basic introduction to using Terraform with the AWS provider. Feel free to explore more complex Terraform configurations and resources based on your needs.