Lab Exercise 4–Provisioning an EC2 Instance on AWS

Prerequisites: Terraform Installed: Make sure you have Terraform installed on your machine. Follow the official installation guide if needed.

AWS Credentials: Ensure you have AWS credentials (Access Key ID and Secret Access Key) configured. You can set them up using the AWS CLI or by setting environment variables.

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:

main.tf

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "5.31.0"
    }
  }
}

provider "aws" {
  region = "us-east-1"
}

```

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:

instance.tf

```
resource "aws_instance" "My-instance" {
  ami           = "ami-09ec59ede75ed2db7" # Use an appropriate AMI ID for your region
  instance_type = "t3.micro"

  tags = {
    Name = "UPES-EC2-Instance"
  }
}

```

Step 5: Review Plan:

Run the following command to see what Terraform will do:

terraform plan

```
iamyo@Acernitro MINGW64 /c/terraform/terraform-demo
$ terraform plan

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-09ec59ede75ed2db7"
      + 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)
```

Review the plan to ensure it aligns with your expectations.

Step 6: Apply Changes:

Apply the changes to create the AWS resources:

terraform apply

```
jamyo@Acernitro MINGW64 /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-09ec59ede75ed2db7"
      + 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)
```

Type yes when prompted.

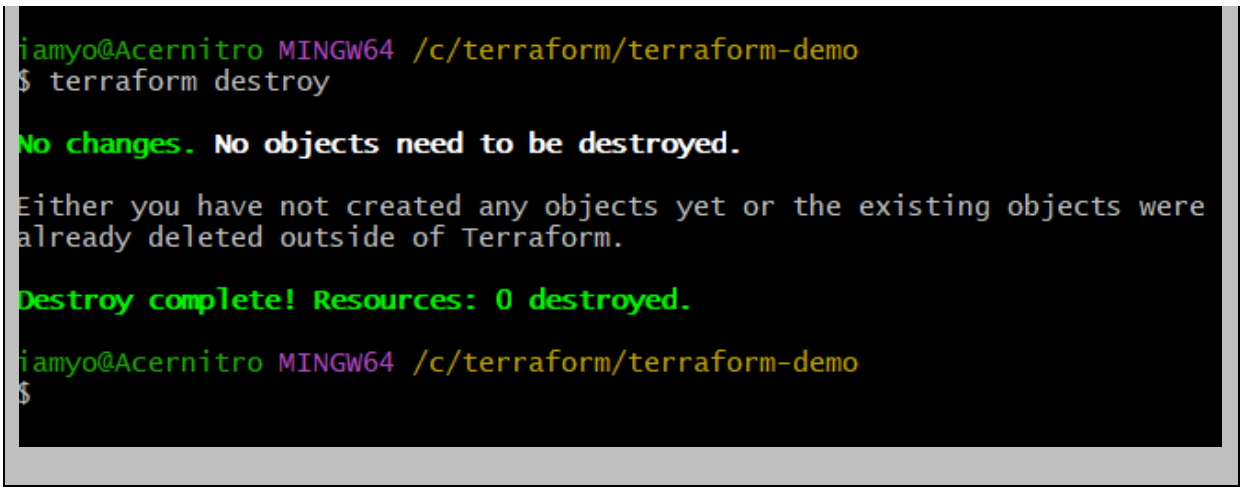
Step 7: Verify Resources:

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
```



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.