Name : Ankur sharma

Sap id : 500110541

Enroll no. R2142221433

Batch : B2 (DevOps)

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:

```
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

```
PS D:\terraform lab> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.30.0

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.
```

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 = "UPES-EC2-Instnace"
   }
}
```

Step 5: Review Plan:

Run the following command to see what Terraform will do:

```
terraform plan
```

```
PS D:\terraform lab> terraform plan

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are
needed.
```

Review the plan to ensure it aligns with your expectations.

Step 6: Apply Changes:

Apply the changes to create the AWS resources:

```
terraform apply
```

```
PS D:\terraform lab> terraform apply

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are
needed.

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

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

```
PS D:\terraform lab> terraform destroy

No changes. No objects need to be destroyed.

Either you have not created any objects yet or the existing objects were already deleted outside of Terraform.

Destroy complete! Resources: 0 destroyed.
```

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.