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:

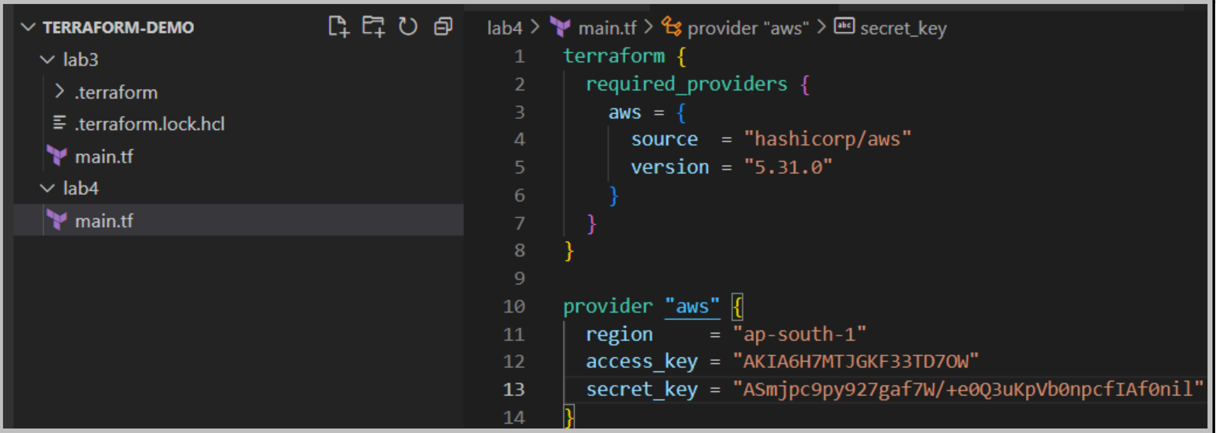
“Terraform-Demo”

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"

}
```



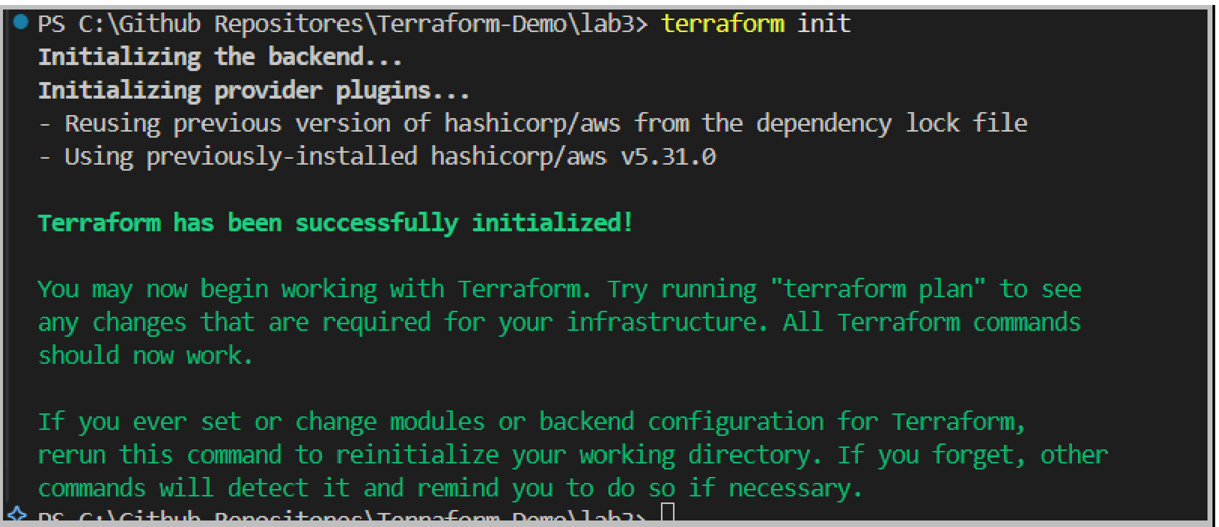
```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "5.31.0"
    }
  }
}

provider "aws" {
  region     = "ap-south-1"
  access_key = "AKIA6H7MTJGKF33TD7OW"
  secret_key = "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"
}
```

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
```

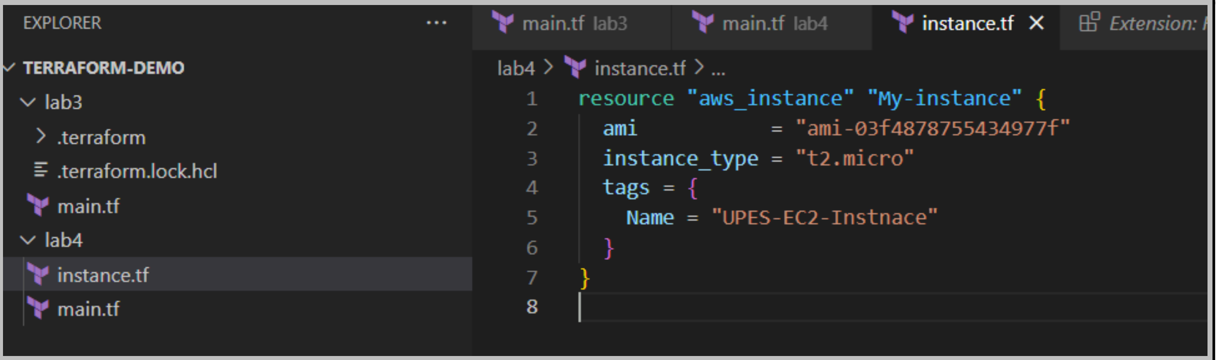


```
PS C:\Github Repositores\Terraform-Demo\lab3> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.31.0

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.
PS C:\Github Repositores\Terraform-Demo\lab3>
```

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"

    }

}
```

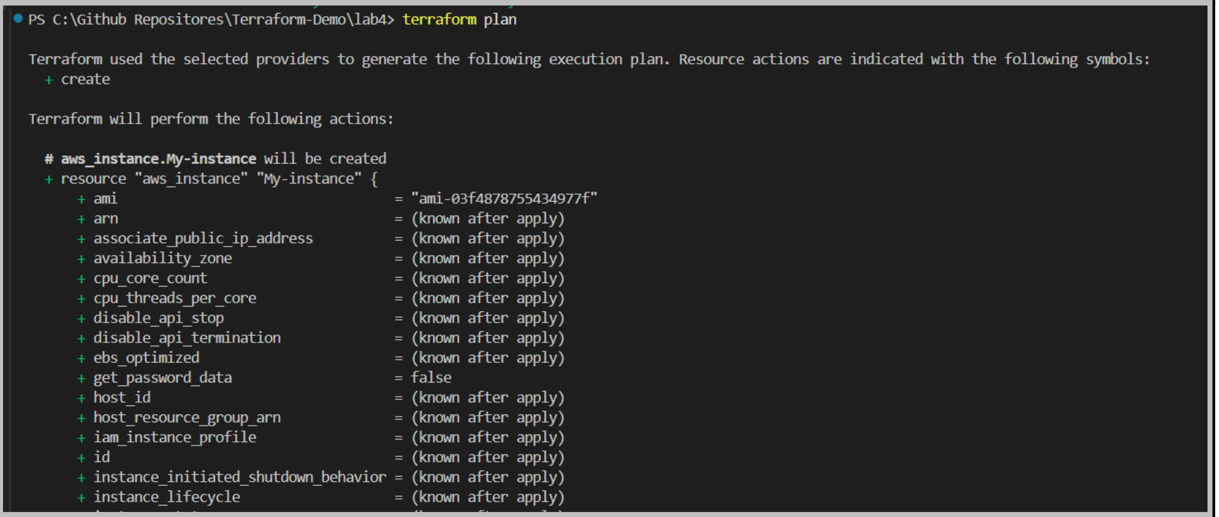


```
resource "aws_instance" "My-instance" {
  ami           = "ami-03f4878755434977f"
  instance_type = "t2.micro"
  tags = {
    Name = "UPES-EC2-Instnace"
  }
}
```

Step 5: Review Plan:

Run the following command to see what Terraform will do:

```
terraform plan
```



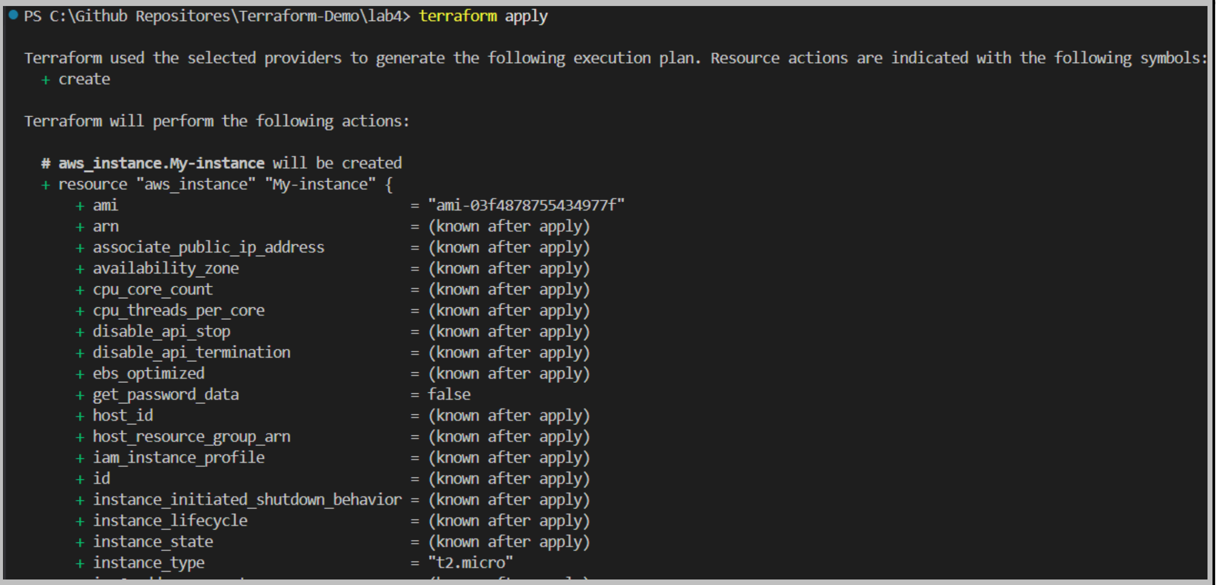
```
PS C:\Github Repositores\Terraform-Demo\lab4> terraform plan

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-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)
```

Review the plan to ensure it aligns with your expectations.

Step 6: Apply Changes:

Apply the changes to create the AWS resources:

```
terraform apply
```

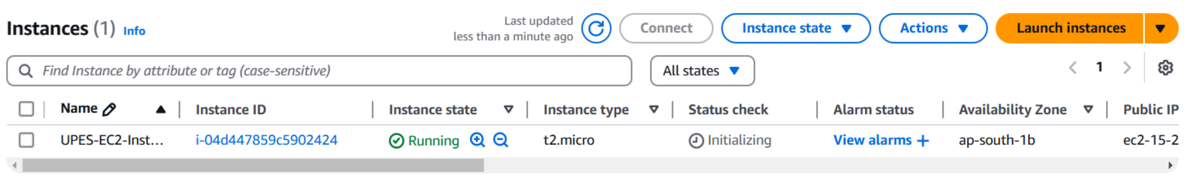


```
PS C:\Github Repositores\Terraform-Demo\lab4> terraform apply

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                        = "t2.micro"
```

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.

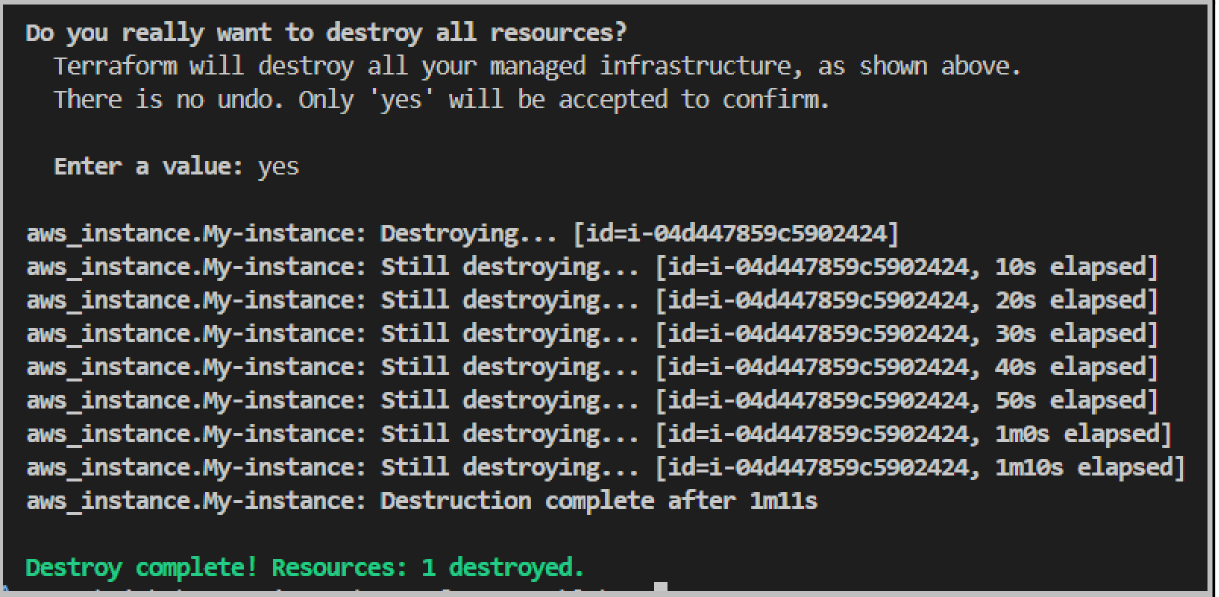

Instances (1) Info
Last updated less than a minute ago
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
1

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
UPES-EC2-Inst...	i-04d447859c5902424	Running	t2.micro	Initializing	View alarms +	ap-south-1b	ec2-15-2

Step 8: Cleanup Resources:

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

```
terraform 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-04d447859c5902424]
aws_instance.My-instance: Still destroying... [id=i-04d447859c5902424, 10s elapsed]
aws_instance.My-instance: Still destroying... [id=i-04d447859c5902424, 20s elapsed]
aws_instance.My-instance: Still destroying... [id=i-04d447859c5902424, 30s elapsed]
aws_instance.My-instance: Still destroying... [id=i-04d447859c5902424, 40s elapsed]
aws_instance.My-instance: Still destroying... [id=i-04d447859c5902424, 50s elapsed]
aws_instance.My-instance: Still destroying... [id=i-04d447859c5902424, 1m0s elapsed]
aws_instance.My-instance: Still destroying... [id=i-04d447859c5902424, 1m10s elapsed]
aws_instance.My-instance: Destruction complete after 1m11s

Destroy complete! Resources: 1 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.