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:

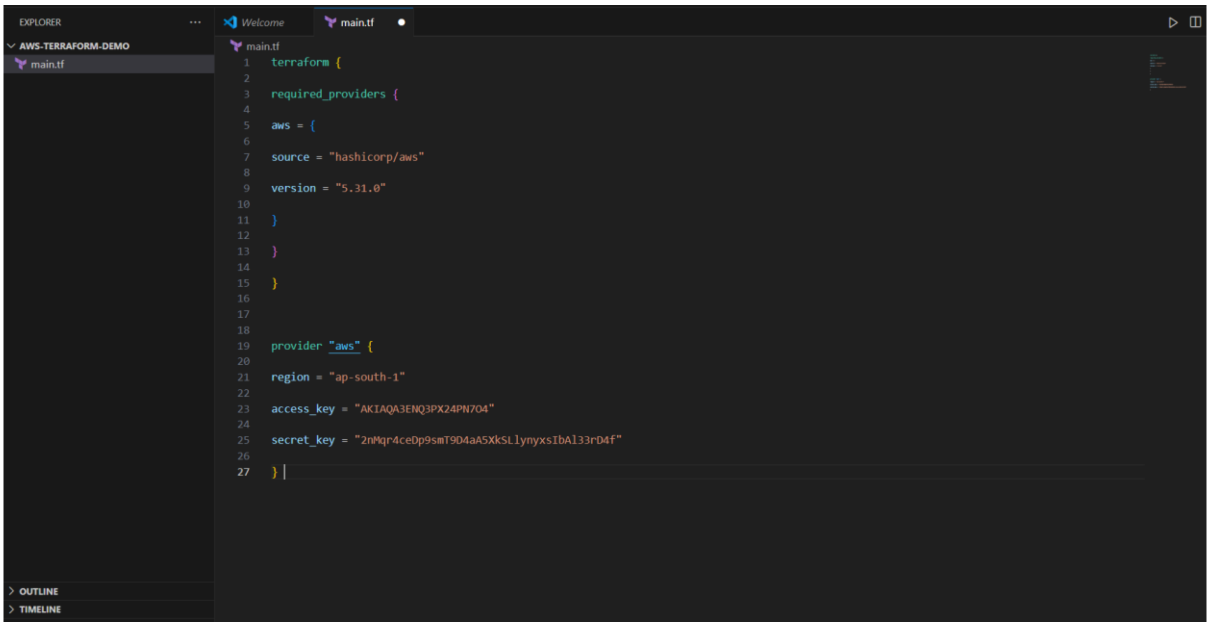
Step 1: Create a New Directory:

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 = "AKIAQA3ENQ3PX24PN7O4"

secret_key = "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"

}
```

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

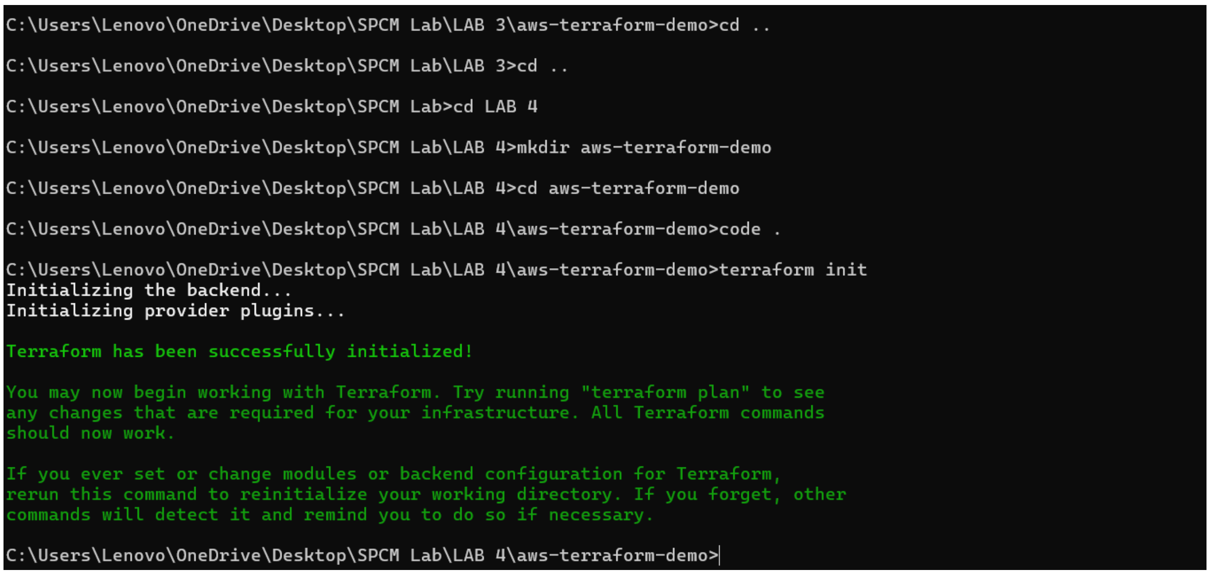
```
provider "aws" {
  region     = "ap-south-1"
  access_key = "your IAM access key"
  secret_key = "your secret access key"
}
```

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
```

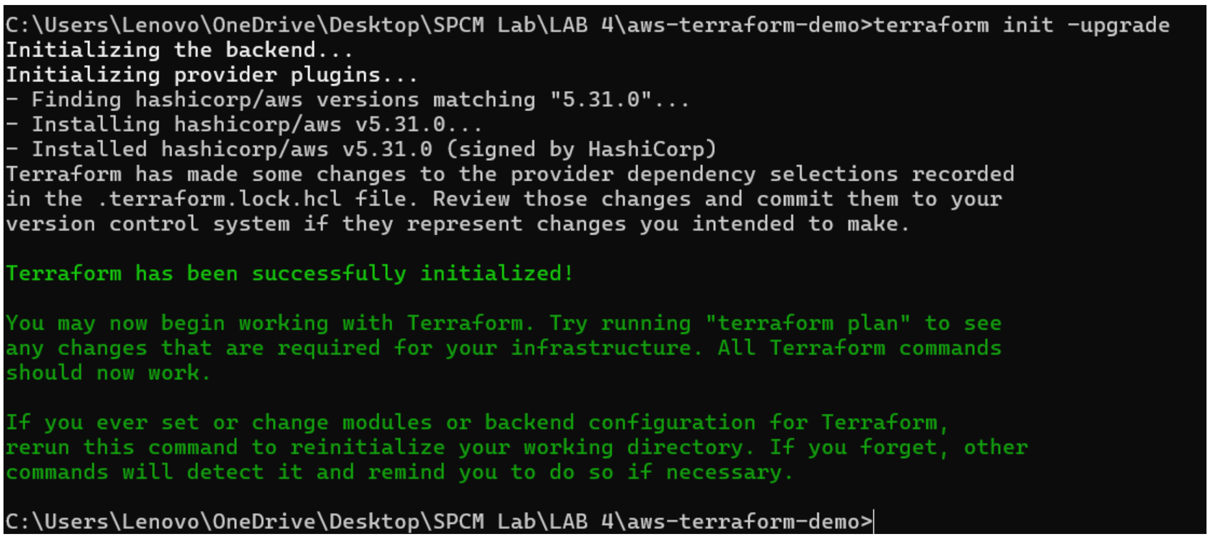


```
C:\Users\Lenovo\OneDrive\Desktop\SPCM Lab\LAB 3\aws-terraform-demo>cd ..

C:\Users\Lenovo\OneDrive\Desktop\SPCM Lab\LAB 3>cd ..

C:\Users\Lenovo\OneDrive\Desktop\SPCM Lab>cd LAB 4

C:\Users\Lenovo\OneDrive\Desktop\SPCM Lab\LAB 4>mkdir aws-terraform-demo

C:\Users\Lenovo\OneDrive\Desktop\SPCM Lab\LAB 4>cd aws-terraform-demo

C:\Users\Lenovo\OneDrive\Desktop\SPCM Lab\LAB 4\aws-terraform-demo>code .

C:\Users\Lenovo\OneDrive\Desktop\SPCM Lab\LAB 4\aws-terraform-demo>terraform init
Initializing the backend...
Initializing provider plugins...

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.

C:\Users\Lenovo\OneDrive\Desktop\SPCM Lab\LAB 4\aws-terraform-demo>
```

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"

   }

}
```

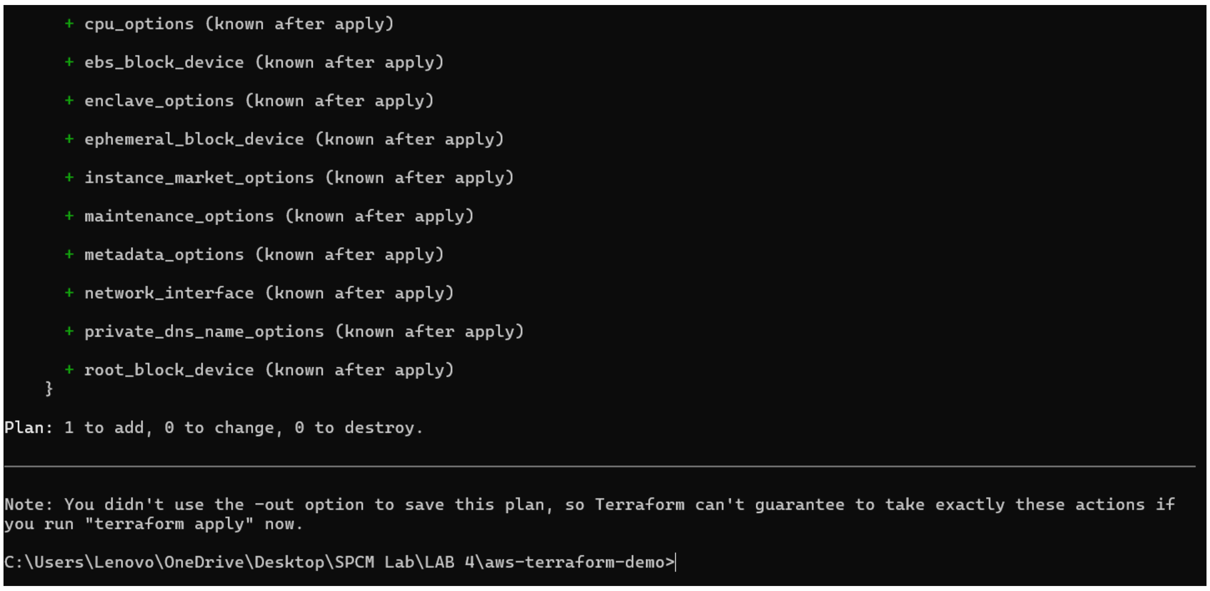


```
C:\Users\Lenovo\OneDrive\Desktop\SPCM Lab\LAB 4\aws-terraform-demo>terraform init -upgrade
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.31.0"...
- Installing hashicorp/aws v5.31.0...
- Installed hashicorp/aws v5.31.0 (signed by HashiCorp)
Terraform has made some changes to the provider dependency selections recorded
in the .terraform.lock.hcl file. Review those changes and commit them to your
version control system if they represent changes you intended to make.

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.

C:\Users\Lenovo\OneDrive\Desktop\SPCM Lab\LAB 4\aws-terraform-demo>
```

Step 5: Review Plan:

Run the following command to see what Terraform will do:

```
terraform plan
```

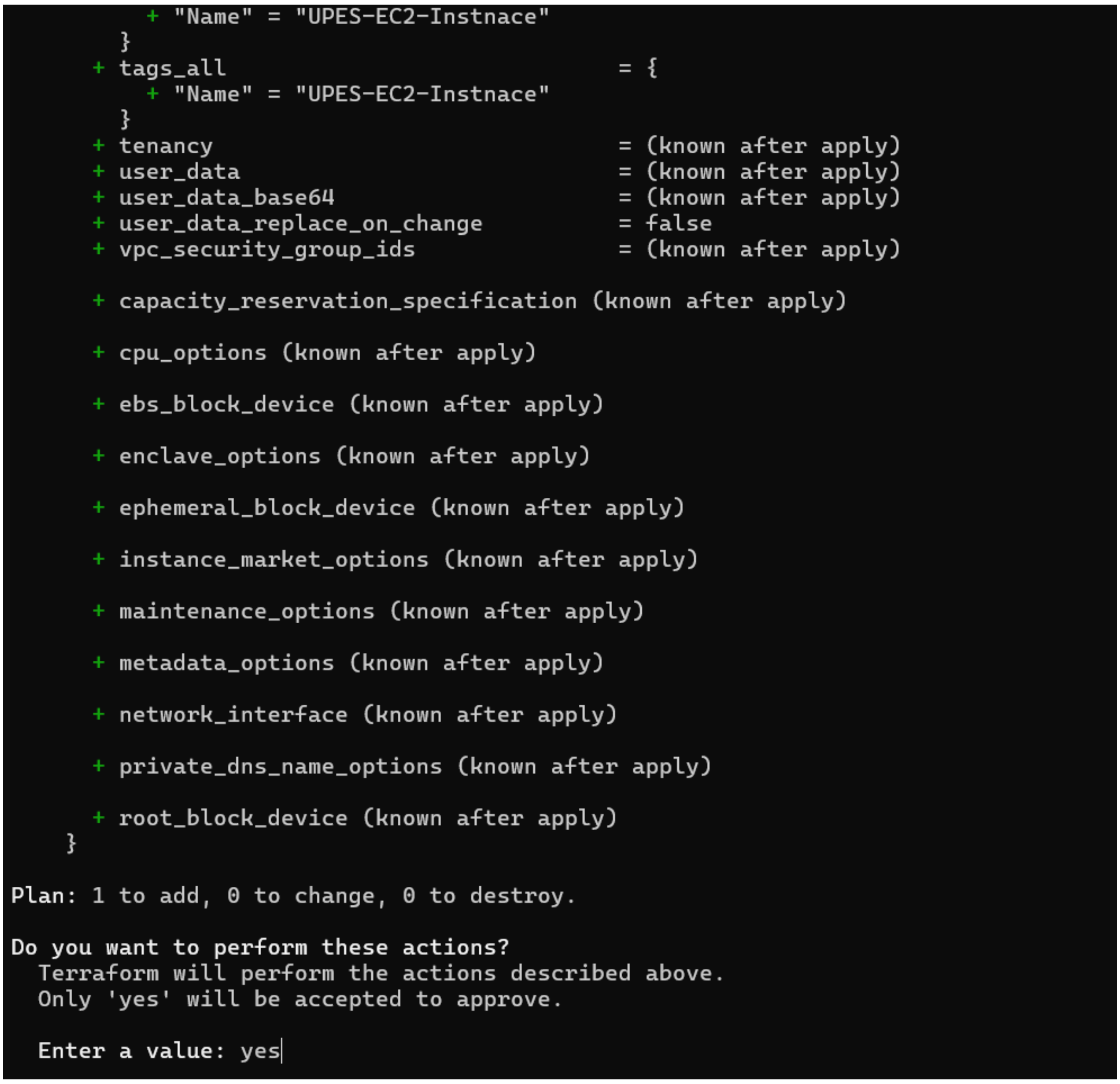


```
      + cpu_options (known after apply)

      + ebs_block_device (known after apply)

      + enclave_options (known after apply)

      + ephemeral_block_device (known after apply)

      + instance_market_options (known after apply)

      + maintenance_options (known after apply)

      + metadata_options (known after apply)

      + network_interface (known after apply)

      + private_dns_name_options (known after apply)

      + root_block_device (known after apply)
    }

Plan: 1 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.

C:\Users\Lenovo\OneDrive\Desktop\SPCM Lab\LAB 4\aws-terraform-demo>
```

Review the plan to ensure it aligns with your expectations.

Step 6: Apply Changes:

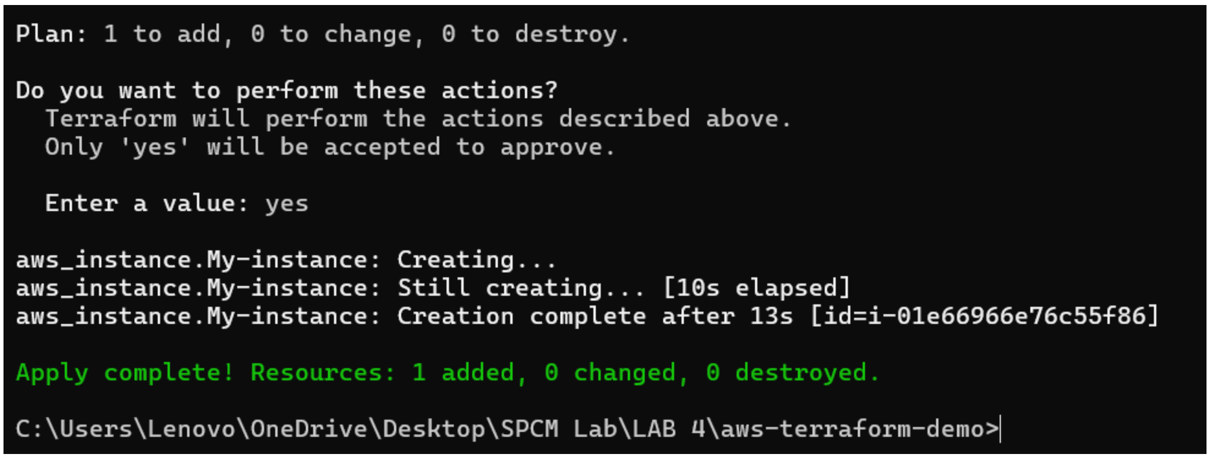
Apply the changes to create the AWS resources:



```
          + "Name" = "UPES-EC2-Instnace"
        }
      + tags_all                             = {
          + "Name" = "UPES-EC2-Instnace"
        }
      + tenancy                              = (known after apply)
      + user_data                            = (known after apply)
      + user_data_base64                     = (known after apply)
      + user_data_replace_on_change          = false
      + vpc_security_group_ids               = (known after apply)

      + capacity_reservation_specification (known after apply)

      + cpu_options (known after apply)

      + ebs_block_device (known after apply)

      + enclave_options (known after apply)

      + ephemeral_block_device (known after apply)

      + instance_market_options (known after apply)

      + maintenance_options (known after apply)

      + metadata_options (known after apply)

      + network_interface (known after apply)

      + private_dns_name_options (known after apply)

      + root_block_device (known after apply)
    }

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
```

```
terraform apply
```

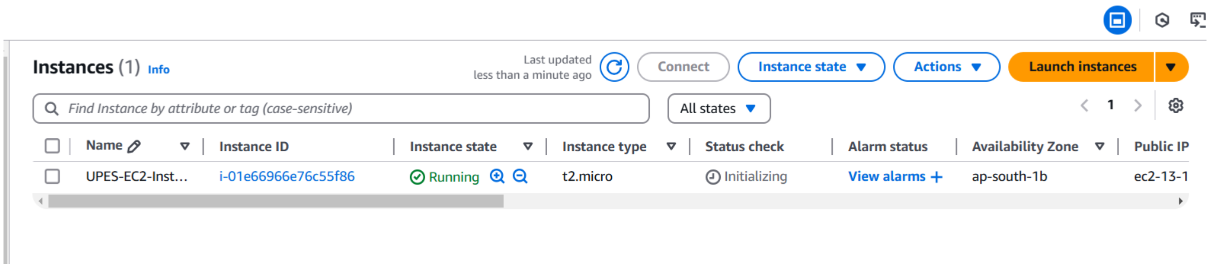
Type yes when prompted.



```
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... [10s elapsed]
aws_instance.My-instance: Creation complete after 13s [id=i-01e66966e76c55f86]

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

C:\Users\Lenovo\OneDrive\Desktop\SPCM Lab\LAB 4\aws-terraform-demo>
```

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.

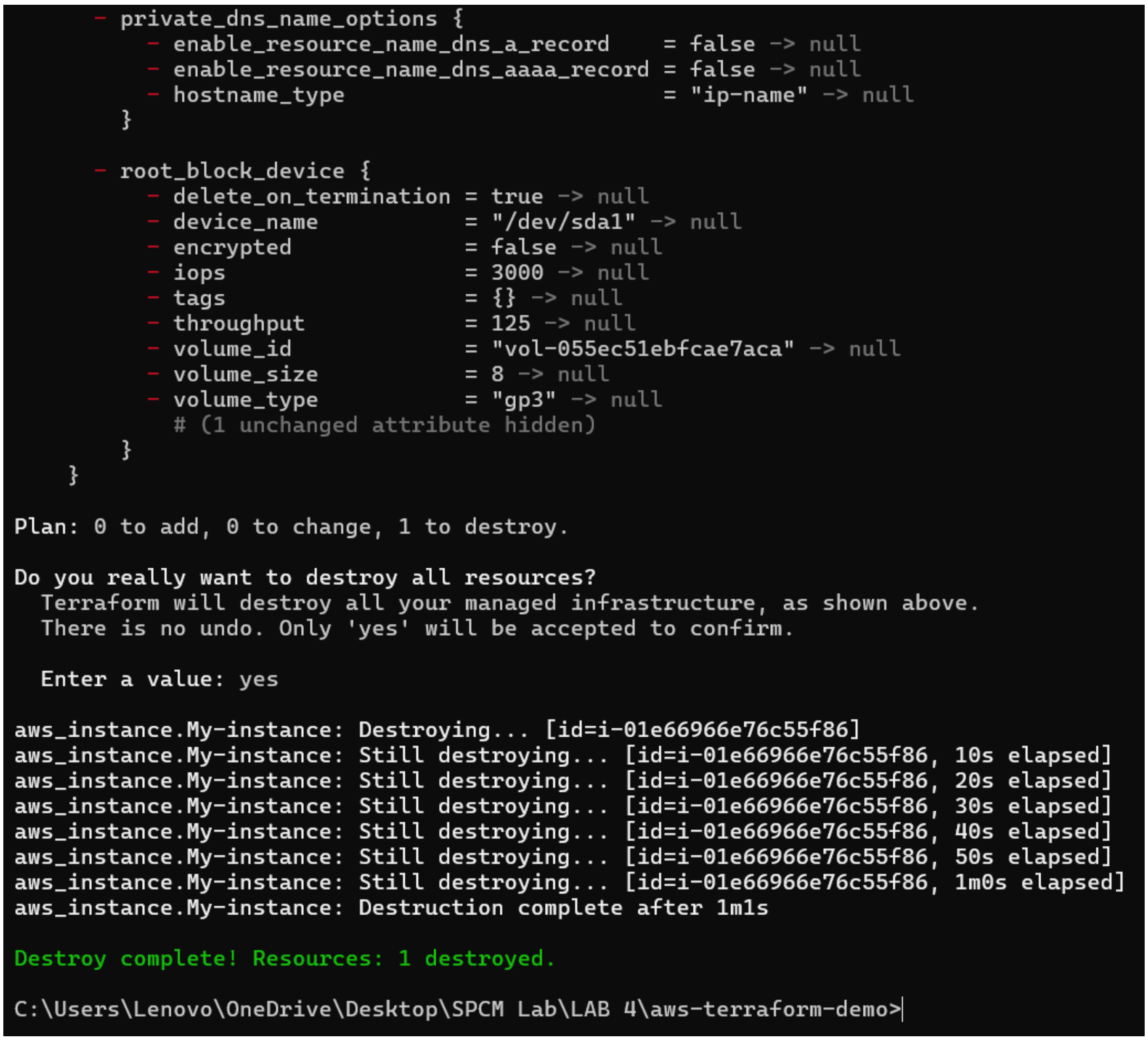

Instances (1) Info
Last updated less than a minute ago
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
1

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
UPES-EC2-Inst...	i-01e66966e76c55f86	Running	t2.micro	Initializing	View alarms +	ap-south-1b	ec2-13-1

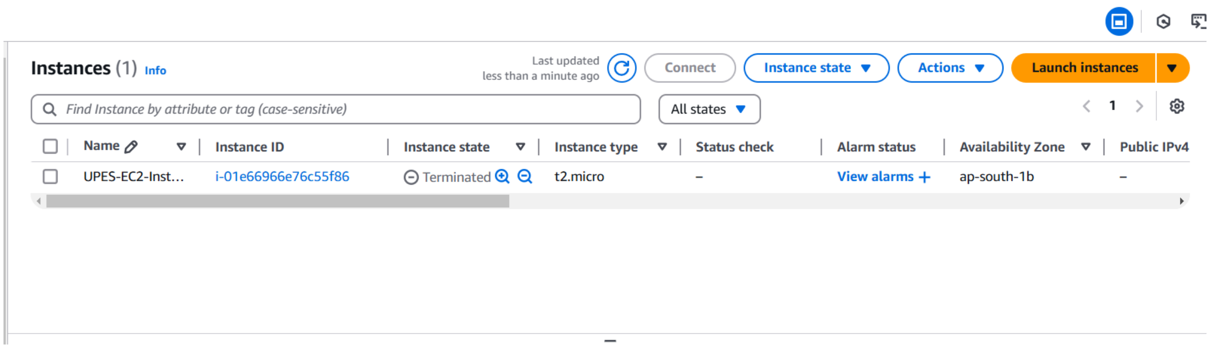
Step 8: Cleanup Resources:

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

```
terraform destroy
```



```
      - private_dns_name_options {
          - enable_resource_name_dns_a_record    = false -> null
          - enable_resource_name_dns_aaaa_record = false -> null
          - hostname_type                        = "ip-name" -> null
        }

      - root_block_device {
          - delete_on_termination = true -> null
          - device_name           = "/dev/sda1" -> null
          - encrypted             = false -> null
          - iops                  = 3000 -> null
          - tags                  = {} -> null
          - throughput            = 125 -> null
          - volume_id             = "vol-055ec51ebfcae7aca" -> null
          - volume_size           = 8 -> null
          - volume_type           = "gp3" -> 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-01e66966e76c55f86]
aws_instance.My-instance: Still destroying... [id=i-01e66966e76c55f86, 10s elapsed]
aws_instance.My-instance: Still destroying... [id=i-01e66966e76c55f86, 20s elapsed]
aws_instance.My-instance: Still destroying... [id=i-01e66966e76c55f86, 30s elapsed]
aws_instance.My-instance: Still destroying... [id=i-01e66966e76c55f86, 40s elapsed]
aws_instance.My-instance: Still destroying... [id=i-01e66966e76c55f86, 50s elapsed]
aws_instance.My-instance: Still destroying... [id=i-01e66966e76c55f86, 1m0s elapsed]
aws_instance.My-instance: Destruction complete after 1m1s

Destroy complete! Resources: 1 destroyed.

C:\Users\Lenovo\OneDrive\Desktop\SPCM Lab\LAB 4\aws-terraform-demo>
```



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 IPv4
UPES-EC2-Inst...	i-01e66966e76c55f86	Terminated	t2.micro	–	View alarms +	ap-south-1b	–

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.