Lab Exercise 3–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:

```
mkdir Lab3
cd Lab3
```

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"
```

```
}
~/Documents/Semester 6/System Provision /Lab Work /MyLab ❯ mkdir Lab3          02:41:34 pm
❯
~/Documents/Semester 6/System Provision /Lab Work /MyLab ❯ cd Lab3             02:41:38 pm
❯
~/Documents/Semester 6/System Provision /Lab Work /MyLab/Lab3 ❯ touch main.tf 02:41:42 pm
❯
~/Documents/Semester 6/System Provision /Lab Work /MyLab/Lab3 ❯ code .         02:41:50 pm
❯
~/Documents/Semester 6/System Provision /Lab Work /MyLab/Lab3 ❯ cat main.tf    02:41:58 pm
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.83.0"
    }
  }
}

provider "aws" {
region = "ap-south-1"
access_key = "AKIA4ZZIDPTHGCZXIMPI"
secret_key = "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"
}
❯
~/Documents/Semester 6/System Provision /Lab Work /MyLab/Lab3 ❯ _              02:42:56 pm
```

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

```
~/Documents/Semester 6/System Provision /Lab Work /MyLab/Lab3 ❯ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.83.0"...
- Installing hashicorp/aws v5.83.0...
- Installed hashicorp/aws v5.83.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

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" {

  instance_type = "t2.micro"

  ami = "ami-03f4878755434977f"

  count = 1

  tags = {

    Name = "UPES-EC2-Instnace"

  }
```

```
}
```

```
~/Documents/Semester 6/System Provision /Lab Work /MyLab/Lab3 20s ❯ cat instanceBhavesh.tf

resource "aws_instance" "DevOpsBhavesh" {
  instance_type = "t2.micro"
  ami = ""
  count = 2
  tags = {
    Name = "UPES-EC2-Instance"
  }
}
```

Step 5: Review Plan:

Run the following command to see what Terraform will do:

terraform plan

```
~/Documents/Semester 6/System Provision /Lab Work /MyLab/Lab3 ❯ 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.DevOpsBhavesh[0] will be created
  + resource "aws_instance" "DevOpsBhavesh" {
      + ami                                  = (known after apply)
      + 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)
      + enable_primary_ipv6                  = (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"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = (known after apply)
      + monitoring                           = (known after apply)
      + outpost_arn                          = (known after apply)
      + password_data                        = (known after apply)
      + placement_group                      = (known after apply)
      + placement_partition_number           = (known after apply)
      + primary_network_interface_id         = (known after apply)
      + private_dns                          = (known after apply)
      + private_ip                           = (known after apply)
      + public_dns                           = (known after apply)
      + public_ip                            = (known after apply)
      + secondary_private_ips                = (known after apply)
      + security_groups                      = (known after apply)
      + source_dest_check                    = true
      + spot_instance_request_id             = (known after apply)
      + subnet_id                            = (known after apply)
      + tags                                 = {
          + "Name" = "UPES-EC2-Instance"
        }
      + tags_all                             = {
          + "Name" = "UPES-EC2-Instance"
        }
      + tenancy                              = (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:

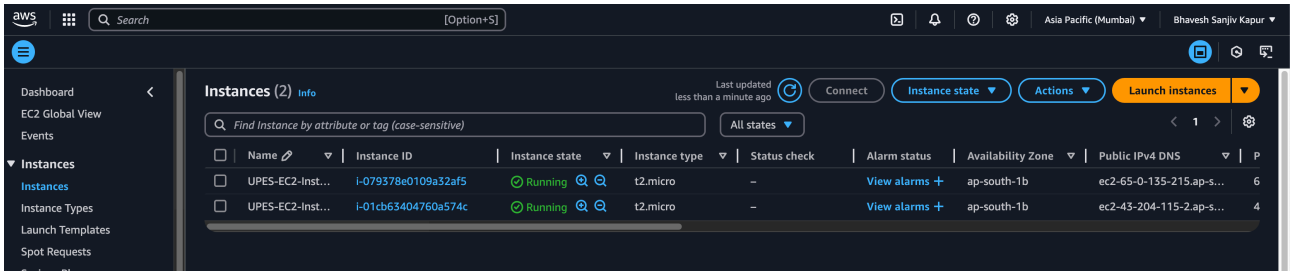
terraform apply

```
Plan: 2 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.DevOpsBhavesh[0]: Creating...
aws_instance.DevOpsBhavesh[1]: Creating...
aws_instance.DevOpsBhavesh[1]: Still creating... [10s elapsed]
aws_instance.DevOpsBhavesh[0]: Still creating... [10s elapsed]
aws_instance.DevOpsBhavesh[1]: Creation complete after 19s [id=i-079378e0109a32af5]
aws_instance.DevOpsBhavesh[0]: Creation complete after 19s [id=i-01cb63404760a574c]

Apply complete! Resources: 2 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.

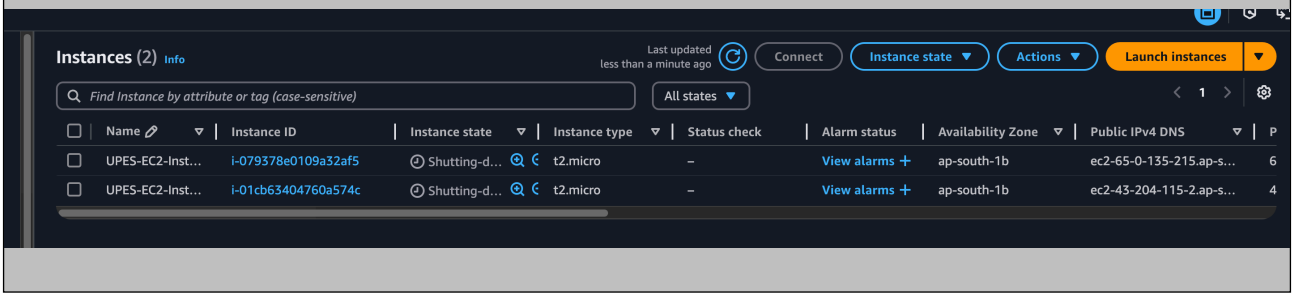


Step 8: Cleanup Resources:

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

terraform destroy

```
Plan: 0 to add, 0 to change, 2 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.DevOpsBhavesh[1]: Destroying... [id=i-079378e0109a32af5]
aws_instance.DevOpsBhavesh[0]: Destroying... [id=i-01cb63404760a574c]
aws_instance.DevOpsBhavesh[1]: Still destroying... [id=i-079378e0109a32af5, 10s elapsed]
aws_instance.DevOpsBhavesh[0]: Still destroying... [id=i-01cb63404760a574c, 10s elapsed]
aws_instance.DevOpsBhavesh[1]: Still destroying... [id=i-079378e0109a32af5, 20s elapsed]
aws_instance.DevOpsBhavesh[0]: Still destroying... [id=i-01cb63404760a574c, 20s elapsed]
aws_instance.DevOpsBhavesh[0]: Still destroying... [id=i-01cb63404760a574c, 30s elapsed]
aws_instance.DevOpsBhavesh[1]: Still destroying... [id=i-079378e0109a32af5, 30s elapsed]
aws_instance.DevOpsBhavesh[1]: Still destroying... [id=i-079378e0109a32af5, 40s elapsed]
aws_instance.DevOpsBhavesh[0]: Still destroying... [id=i-01cb63404760a574c, 40s elapsed]
aws_instance.DevOpsBhavesh[1]: Still destroying... [id=i-079378e0109a32af5, 50s elapsed]
aws_instance.DevOpsBhavesh[0]: Still destroying... [id=i-01cb63404760a574c, 50s elapsed]
aws_instance.DevOpsBhavesh[1]: Still destroying... [id=i-079378e0109a32af5, 1m0s elapsed]
aws_instance.DevOpsBhavesh[0]: Still destroying... [id=i-01cb63404760a574c, 1m0s elapsed]
aws_instance.DevOpsBhavesh[0]: Destruction complete after 1m6s
aws_instance.DevOpsBhavesh[1]: Still destroying... [id=i-079378e0109a32af5, 1m10s elapsed]
aws_instance.DevOpsBhavesh[1]: Destruction complete after 1m16s

Destroy complete! Resources: 2 destroyed.

~/Documents/Semester 6/System Provision /Lab Work /MyLab/Lab3 1m 47s ❯ _
```



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.