School of Computer Science

UNIVERSITY OF PETROLEUM AND ENERGY STUDIES
DEHRADUN, UTTARAKHAND

SYSTEM PROVISIONING AND SYSTEM MONITORING

(2023-2024)

for

6th Semester

Submitted To:

Hitesh Kumar Sharma

Submitted By:

Anmol Ghai

B.tech CSE

500090959

R2142210118

B1- DevOps (Non-Hons)

Experiment 1:

Install Terraform on Windows

```
Command Prompt

Microsoft Windows [Version 10.0.23620.1000]
(c) Microsoft Corporation. All rights reserved.

C:\Users\anmol>terraform --version
Terraform v1.7.1
on windows_amd64

C:\Users\anmol>
```

Experiment 2:

Terraform AWS Provider and IAM User Setting

1. Create Terraform Configuration File

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

provider "aws" {
  region = "ap-south-1"
  access_key = "AKIAW3MEEF7ICUA7QDW2"
  secret_key = "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"
}
```

2. Initialize Terraform

```
D:\UPES Content\Third Year\Sixth Semester\SPCM\aws-terraform-demo>terraform init

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 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.

D:\UPES Content\Third Year\Sixth Semester\SPCM\aws-terraform-demo>
```

Experiment -3

Provisioning an EC2 Instance on AWS

Step 1: Create Terraform Configuration File for EC2 instance (instance.tf)

```
instance.tf
resource "aws_instance" "My-instance" {
instance_type = "t2.micro"
ami = "ami-0e670eb768a5fc3d4"
count = 1
tags = {
    Name = "UPES-EC2-Instnace"
}
}
```

Step 2: Initialize Terraform:

```
D:\UPES Content\Third Year\Sixth Semester\SPCM\aws-terraform-demo>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.
```

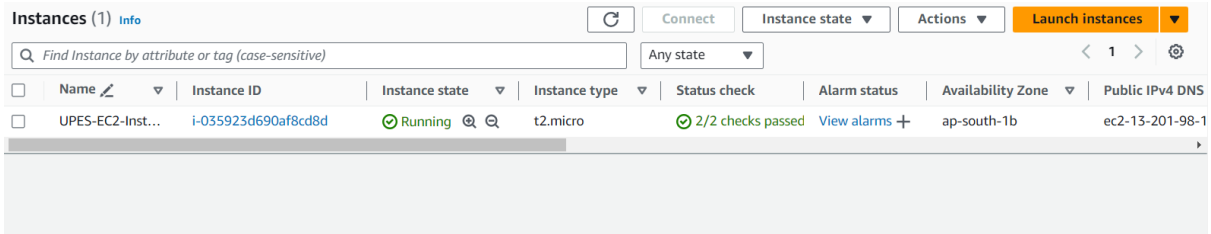
Step 3: Review Plan:

```
D:\UPES Content\Third Year\Sixth Semester\SPCM\aws-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[0] will be created
  + resource "aws_instance" "My-instance" {
      + ami                                  = "ami-0e670eb768a5fc3d4"
      + 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)
```

Step 4: Terraform Apply

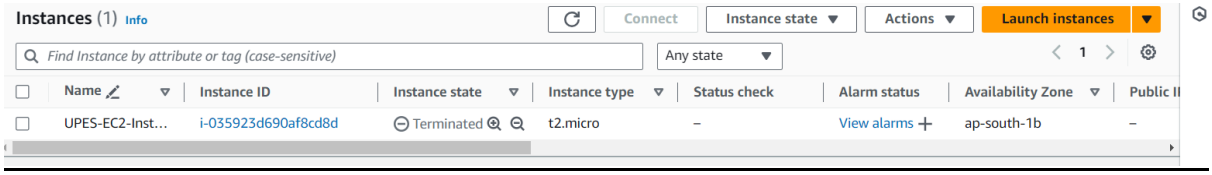
```
D:\UPES Content\Third Year\Sixth Semester\SPCM\aws-terraform-demo>terraform apply

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[0] will be created
  + resource "aws_instance" "My-instance" {
      + ami                                  = "ami-0e670eb768a5fc3d4"
      + 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"
      + 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)
```

```
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[0]: Creating...
aws_instance.My-instance[0]: Still creating... [10s elapsed]
aws_instance.My-instance[0]: Still creating... [20s elapsed]
aws_instance.My-instance[0]: Creation complete after 24s [id=i-035923d690af8cd8d]

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



Step 5: Cleanup Resources/Destroy instance

```
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[0]: Destroying... [id=i-035923d690af8cd8d]
aws_instance.My-instance[0]: Still destroying... [id=i-035923d690af8cd8d, 10s elapsed]
aws_instance.My-instance[0]: Still destroying... [id=i-035923d690af8cd8d, 20s elapsed]
aws_instance.My-instance[0]: Still destroying... [id=i-035923d690af8cd8d, 30s elapsed]
aws_instance.My-instance[0]: Destruction complete after 31s

Destroy complete! Resources: 1 destroyed.

D:\UPES Content\Third Year\Sixth Semester\SPCM\aws-terraform-demo>
```



Experiment 4:

TERRAFORM VARIABLES

Step 1: Create a Terraform Configuration File:

main.tf × variables.tf

terraform-variables > main.tf

```
provider "aws" {
 region = "us-west-2"
}
resource "aws_instance" "example" {
 ami = "ami-0e670eb768a5fc3d4"
 instance_type = "t2.micro"
}
```

Step 2: Define Variables:

main.tf variables.tf ×

terraform-variables > variables.tf

```
variable "region" {
  description = "AWS region"
  default = "us-west-2"
}
variable "ami" {
  description = "AMI ID"
  default = "ami-0e670eb768a5fc3d4"
}
variable "instance_type" {
  description = "EC2 Instance Type"
  default = "t2.micro"
}
```

Step 3: Use Variables in instance.tf file:

```
terraform-variables > main.tf
1  provider "aws" {
2    region = "var.region"
3  }
4  resource "aws_instance" "example" {
5    ami = "var.ami"
6    instance_type = "var.instance_type"
7  }
```

Step 4: Now do terraform initialize & RUN:

```
\UPES Content\Third Year\Sixth Semester\SPCM\aws-terraform-demo\terraform-variables>terraform init

itializing the backend...

itializing provider plugins...
Finding latest version of hashicorp/aws...
Installing hashicorp/aws v5.38.0...
Installed hashicorp/aws v5.38.0 (signed by HashiCorp)

rraform has created a lock file .terraform.lock.hcl to record the provider
lections it made above. Include this file in your version control repository
that Terraform can guarantee to make the same selections by default when
u run "terraform init" in the future.

rraform has been successfully initialized!

u may now begin working with Terraform. Try running "terraform plan" to see
y changes that are required for your infrastructure. All Terraform commands
ould now work.

you ever set or change modules or backend configuration for Terraform,
run this command to reinitialize your working directory. If you forget, other
mmands will detect it and remind you to do so if necessary.
```

Step 5: Now perform Terraform Destroy to clean up:

```
      - capacity_reservation_specification {
          - capacity_reservation_preference = "open" -> null
        }

      - cpu_options {
          - core_count       = 1 -> null
          - threads_per_core = 1 -> null
        }

      - credit_specification {
          - cpu_credits = "standard" -> null
        }

      - enclave_options {
          - enabled = false -> null
        }

      - maintenance_options {
          - auto_recovery = "default" -> null
        }

      - metadata_options {
          - http_endpoint               = "enabled" -> null
          - http_protocol_ipv6          = "disabled" -> null
          - http_put_response_hop_limit = 1 -> null
          - http_tokens                 = "optional" -> null
          - instance_metadata_tags      = "disabled" -> null
        }

      - 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                  = 100 -> null
          - tags                  = {} -> null
          - throughput            = 0 -> null
          - volume_id             = "vol-04d6479744095f96c" -> null
          - volume_size           = 8 -> null
          - volume_type           = "gp2" -> null
        }
    }

  # aws_instance.My-Instnace-02[0] will be destroyed
  - resource "aws_instance" "My-Instnace-02" {
      - ami                                  = "ami-03f48787554349 77f" -> null
      - arn                                  = "arn:aws:ec2:ap-south-1:637423348062:instance/i-0a2c04548c6185370" -> null
      - associate_public_ip_address          = true -> null
      - availability_zone                    = "ap-south-1a" -> null
      - cpu_core_count                       = 1 -> null
```