System Provisioning & Configuration Management

Lab File

Submitted By

Keshav Bhardwaj

SAPID: 500094898

Enrollment No: R2142210413

Batch: 03

Semester VI

BTech CSE DevOps

Submitted To

Dr. Hitesh Kumar Sharma

SCHOOL OF COMPUTER SCIENCE

UNIVERSITY OF PETROLEUM & ENERGY STUDIES

Dehradun-248007

2023-24

Experiment 1
Install & Setup Terraform

1. Ensure that your system is up to date and you have installed the gnupg, software-properties-common, and curl packages installed.

```
→  ~  sudo apt-get update && sudo apt-get install -y gnupg software-properties-common
0% [Connecting to ppa.launchpadcontent.net] [Connecting to download.docker.com (108.158.245.1
```

2. Install the HashiCorp GPG key.

```
→  ~  wget -O- https://apt.releases.hashicorp.com/gpg | \
gpg --dearmor | \
sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg

--2024-01-17 10:08:29--  https://apt.releases.hashicorp.com/gpg
Resolving apt.releases.hashicorp.com (apt.releases.hashicorp.com)... 266
Connecting to apt.releases.hashicorp.com (apt.releases.hashicorp.com)|26
```

3. Verify the key's fingerprint

```
→  ~  gpg --no-default-keyring \
--keyring /usr/share/keyrings/hashicorp-archive-keyring.gpg \
--fingerprint
```

4. Add the official HashiCorp repository to your system.

```
→  ~  echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] \
https://apt.releases.hashicorp.com $(lsb_release -cs) main" | \
sudo tee /etc/apt/sources.list.d/hashicorp.list
```

5. Download the package information from HashiCorp.

```
→  ~  sudo apt update
Hit:1 https://download.docker.com/linux/ubuntu jammy InRelease
Hit:2 http://packages.microsoft.com/repos/code stable InRelease
Hit:3 https://apt.releases.hashicorp.com jammy InRelease
Ign:4 https://pkg.jenkins.io/debian binary/ InRelease
Hit:5 https://dl.google.com/linux/chrome/deb stable InRelease
Hit:6 https://pkg.jenkins.io/debian binary/ Release
```

6. Install Terraform from the new repository.

```
→  ~  sudo apt-get install terraform
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
terraform is already the newest version (1.6.6-1).
0 upgraded, 0 newly installed, 0 to remove and 12 not upgraded.
→  ~
```

7. Verify that the installation worked by opening a new terminal session

```
→  ~  terraform --version
Terraform v1.6.6
on linux_amd64
→  ~
```

Experiment 2

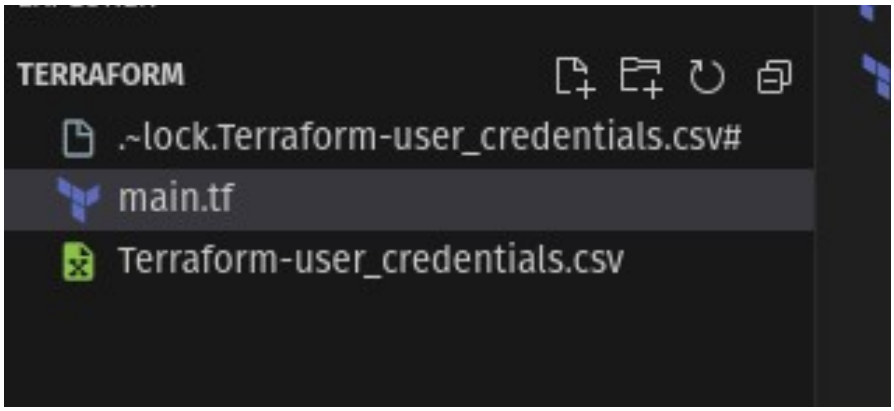
Terraform AWS Provider and

IAM User Settings

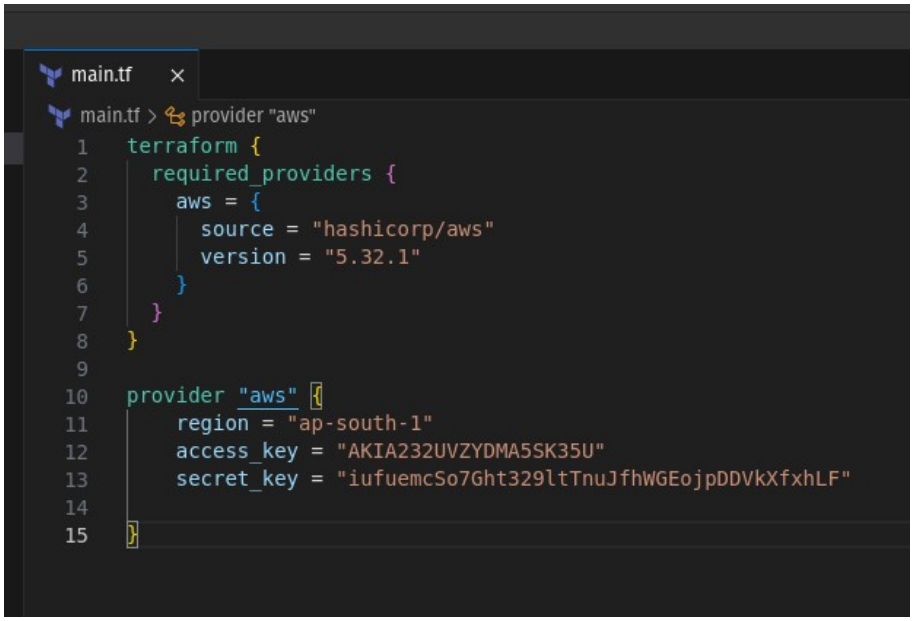
Prerequisites: Terraform Installed: Make sure you have Terraform installed on your machine. Follow the official installation guide if needed.

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.

Step 1. Create a directory named Terraform and make a main.tf file in it.



Step 2 . After creating the main.tf file , add the following content into it.



- This script defines an AWS provider and provisions an EC2 instance.

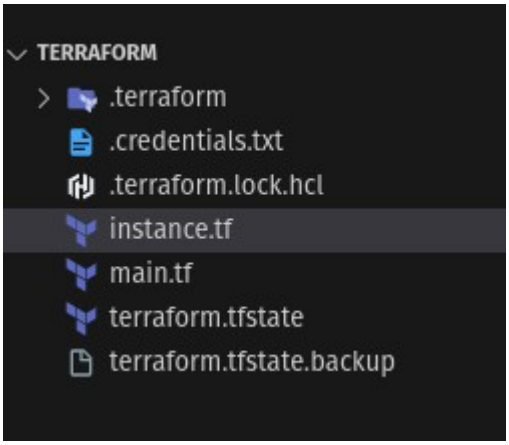
Step 3. Initialize Terraform

```
→  TERRAFORM  terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.32.1"...
- Installing hashicorp/aws v5.32.1...
- Installed hashicorp/aws v5.32.1 (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.
```

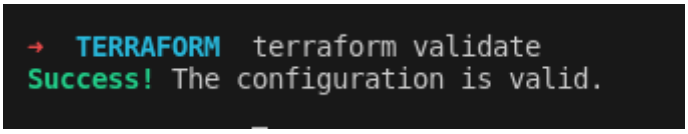
Experiment 3
Provisioning on EC2 Instance on AWS

Prerequisites:
Terraform Installed & AWS Credentials

Step 1. Create a terraform configuration file for EC2 Instance (instance.tf)



Step 2. Validate the configuration



Step 3. Review Plan

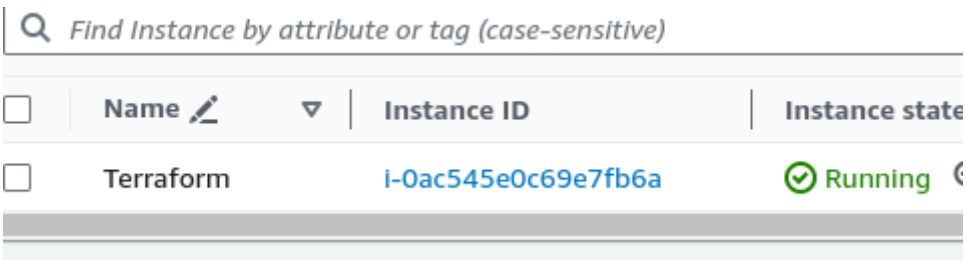
```
→  TERRAFORM  terraform plan

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.terraform will be created
  + resource "aws_instance" "terraform" {
      + 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)
```

Step 4. Terraform 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.terraform: Creating...
aws_instance.terraform: Still creating... [10s elapsed]
aws_instance.terraform: Still creating... [20s elapsed]
aws_instance.terraform: Still creating... [30s elapsed]
aws_instance.terraform: Creation complete after 34s [id=i-0ac545e0c69e7fb6a]
```

Step 5. Verifying resources:

Checking wether an EC2 instance is created in aws console or not.



Step 6. Cleaning Resources

```
→  TERRAFORM  terraform destroy
aws_instance.terraform: Refreshing state... [id=i-0ac545e0c69e7fb6a]

Terraform used the selected providers to generate the following execution pl
  - destroy

Terraform will perform the following actions:

  # aws_instance.terraform will be destroyed
  - resource "aws_instance" "terraform" {
      - ami                                  = "ami-03f4878755434977f" -> nu
      - arn                                  = "arn:aws:ec2:ap-south-1:74696
      - associate_public_ip_address          = true -> null
      - availability_zone                    = "ap-south-1a" -> null
      - cpu_core_count                       = 1 -> null
      - cpu_threads_per_core                 = 1 -> null
      - disable_api_stop                     = false -> null
      - disable_api_termination              = false -> null
```

Find Instance by attribute or tag (case-sensitive)

Name	Instance ID	Instance state
Terraform	i-0ac545e0c69e7fb6a	Shutting-d...

```
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

aws_instance.terraform: Destroying... [id=i-0ac545e0c69e7fb6a]
aws_instance.terraform: Still destroying... [id=i-0ac545e0c69e7fb6a, 10s elapsed]
aws_instance.terraform: Still destroying... [id=i-0ac545e0c69e7fb6a, 20s elapsed]
aws_instance.terraform: Still destroying... [id=i-0ac545e0c69e7fb6a, 30s elapsed]
aws_instance.terraform: Destruction complete after 32s

Destroy complete! Resources: 1 destroyed.
```

nstances (1) Info

Find Instance by attribute or tag (case-sensitive)

Name	Instance ID	Instance state
Terraform	i-0ac545e0c69e7fb6a	Terminated

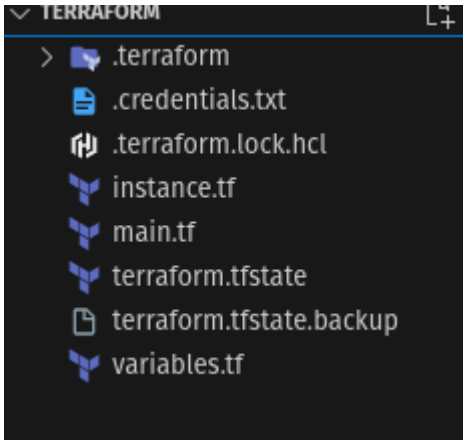
Terminated

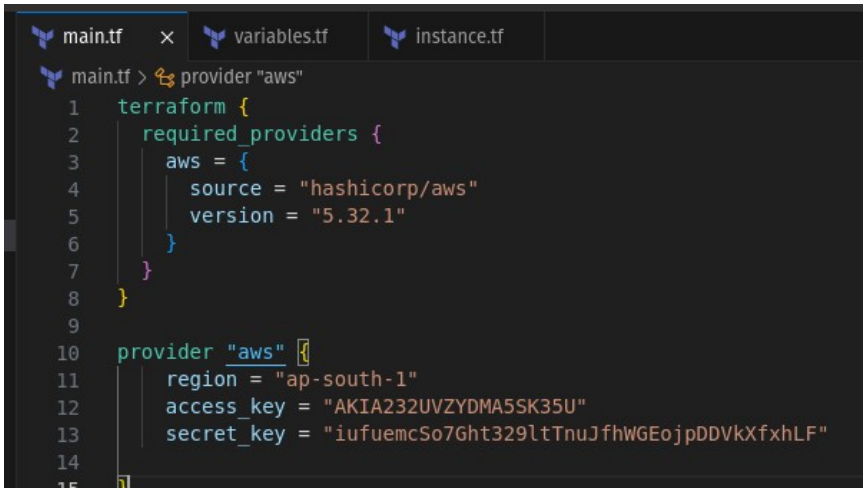
Instance is destroyed successfully

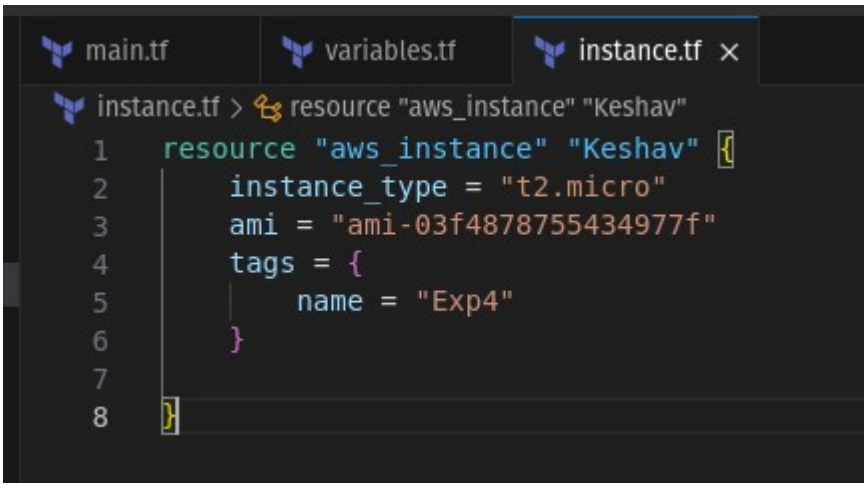
Experiment 4
Terraform Variables

Aim
Learn how to define and use variables in Terraform configuration

Step 1. Create a main file & terraform configuration file for EC2 Instance (instance.tf)







Step 2. Open a new file named variables.tf. Define variables for region, ami, secret_key, access_key and instance_type.

```
main.tf    variables.tf ×    instance.tf
variables.tf > variable "instance_type" > description
variable region {
    type = string
    default = "ap-south-1"
    description = "AWS Region"
}

variable ami {
    type = string
    default = "ami-03f4878755434977f"
    description = "AMI ID"
}

variable "instance_type" {
    type = string
    default = "t2.   string
    description = "Instance type"
}
```

Step 3. modify main.tf and instance.tf to use the variables.

```
main.tf    variables.tf    instance.tf ×
instance.tf > resource "aws_instance" "Keshav"
resource "aws_instance" "Keshav" {
    instance_type = var.instance_type
    ami = var.ami
    count = 1

    tags = {
        Name = "Exp4"
    }

}
```

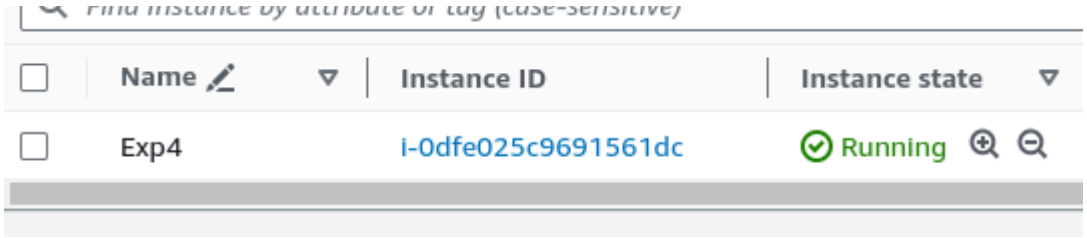
```
main.tf  ×   variables.tf   instance.tf
main.tf > provider "aws" > secret_key
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.32.1"
    }
  }
}

provider "aws" {
    region = var.region
    access_key = var.access_key
    secret_key = var.secret_key
}
```

Step 4. Run the following Terraform commands to initialize and apply the configuration.

```
      + 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" = "Exp4"
        }
      + tags_all                           = {
          + "Name" = "Exp4"
        }
      + 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)
    }

Plan: 1 to add, 0 to change, 0 to destroy.
```

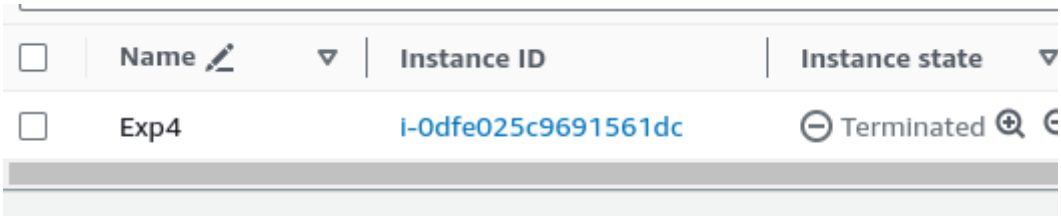
```
      + tenancy                            = (known after apply)
      + user_data                          = (known after apply)
      + user_data_base64                   = (known after apply)
      + user_data_replace_on_change        = false
      + vpc_security_group_ids             = (known after apply)
    }

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.Keshav[0]: Creating...
aws_instance.Keshav[0]: Still creating... [10s elapsed]
aws_instance.Keshav[0]: Still creating... [20s elapsed]
aws_instance.Keshav[0]: Still creating... [31s elapsed]
aws_instance.Keshav[0]: Creation complete after 36s [id=i-0dfe025c9691561dc]

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

Step 5 . Verifying Resources



Step 6. Cleanup Resources

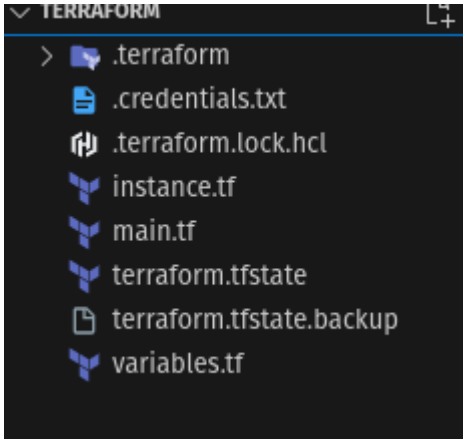
```
            - throughput            = 0 -> null
            - volume_id             = "vol-0c7d2df7d240732da" -> null
            - volume_size           = 8 -> null
            - volume_type           = "gp2" -> null
        }
    }

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.Keshav[0]: Destroying... [id=i-0dfe025c9691561dc]
aws_instance.Keshav[0]: Still destroying... [id=i-0dfe025c9691561dc, 10s elapsed]
aws_instance.Keshav[0]: Still destroying... [id=i-0dfe025c9691561dc, 20s elapsed]
aws_instance.Keshav[0]: Still destroying... [id=i-0dfe025c9691561dc, 30s elapsed]
aws_instance.Keshav[0]: Destruction complete after 31s
```

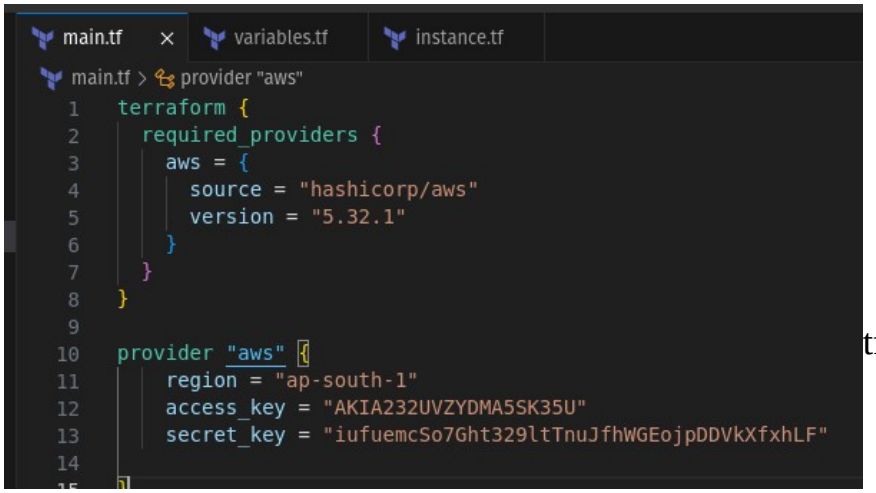


Experiment 5
Terraform Variables with Command Line Arguments

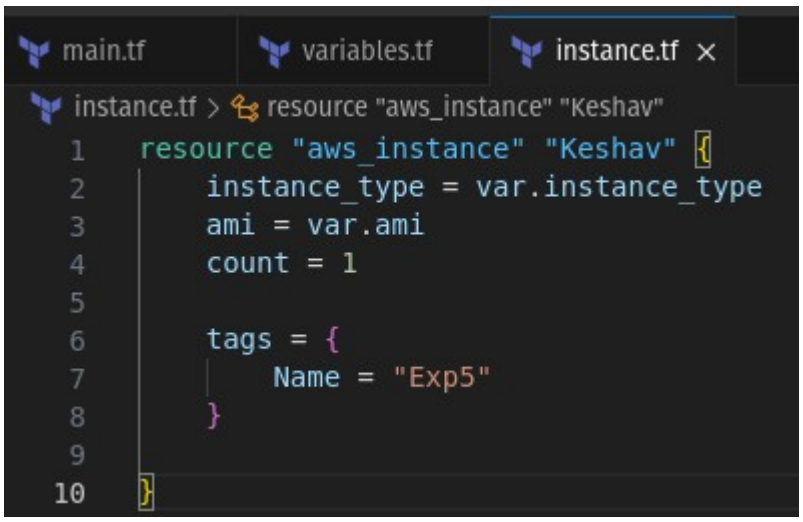
Aim
Learn how to pass values to Terraform variables using command line arguments.

Step 1. Create a main file & terraform configuration file for EC2 Instance (instance.tf) & add variables to them.





tfvars



Step 2. Open a new file named variables.tf. Define variables for region, ami, secret_key, access_key and instance_type.

```
main.tf    variables.tf ×    instance.tf
variables.tf > variable "instance_type" > description
variable region {
    type = string
    default = "ap-south-1"
    description = "AWS Region"
}

variable ami {
    type = string
    default = "ami-03f4878755434977f"
    description = "AMI ID"
}

variable "instance_type" {
    type = string
    default = "t2.   string
    description = "Instance type"
}
```

Step 3. Run the following Terraform commands to initialize and apply the configuration & pass variables as command line arguments.

```
→  TERRAFORM  terraform apply -var 'region=ap-south-1' -var 'ami=ami-03f4878755434977f' -var 'instance_type=t2.micro'

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:
```

Step 4. Verify Resources

Name	Instance ID	Instance state
Exp5	i-0bfb81933ff1181b5	Running
Exp4	i-0dfe025c9691561dc	Terminated

Step 5. Cleanup resources

```
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.Keshav[0]: Destroying... [id=i-0bfb81933ff1181b5]
aws_instance.Keshav[0]: Still destroying... [id=i-0bfb81933ff1181b5, 10s elapsed]
aws_instance.Keshav[0]: Still destroying... [id=i-0bfb81933ff1181b5, 20s elapsed]
aws_instance.Keshav[0]: Still destroying... [id=i-0bfb81933ff1181b5, 30s elapsed]
aws_instance.Keshav[0]: Destruction complete after 30s

Destroy complete! Resources: 1 destroyed.
→  TERRAFORM
```

Experiment 6
Terraform Multiple tfvars Files

Aim

Learn how to use multiple tfvars files in Terraform for different environments.

Steps

1. Create a main file, instance.tf file & variables.tf file for EC2 Instance.

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.32.1"
    }
  }
}

provider "aws" {
    region = "ap-south-1"
    access_key = "AKIA232UVZYDMA5SK35U"
    secret_key = "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"

}
```

```
resource "aws_instance" "Keshav" {
    instance_type = var.instance_type
    ami = var.ami
    count = 1

    tags = {
        Name = "Exp6"
    }

}
```

```
variable region {
    type = string
    default = "ap-south-1"
    description = "AWS Region"
}

variable ami {
    type = string
    default = "ami-03f4878755434977f"
    description = "AMI ID"
}

variable "instance_type" {
    type = string
    default = "t2.
    description = "Instance type"
}
```

2. Create two tfvars files for different environments

a. dev.tfvars

```
region= "ap-south-1"
ami= "ami-03f4878755434977f"
instance_type = "t2.micro"
```

b. prod.tfvars

```
region = "us-east-1"
ami = "ami-0c7217cdde317cfec"
instance_type = "t2.micro"
```

3. Initialize and provision resources in both environments

a. dev

```
→  TERRAFORM  terraform apply -var-file=dev.tfvars

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.Keshav[0] will be created
  + resource "aws_instance" "Keshav" {
      + ami                                  = "ami-03f4878755434977f
```

tfvars

```
aws_instance.Keshav[0]: Creating...
aws_instance.Keshav[0]: Still creating... [10s elapsed]
aws_instance.Keshav[0]: Still creating... [20s elapsed]
aws_instance.Keshav[0]: Still creating... [30s elapsed]
aws_instance.Keshav[0]: Creation complete after 33s [id=i-00897
3bd52]

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

tfvars

b. prod

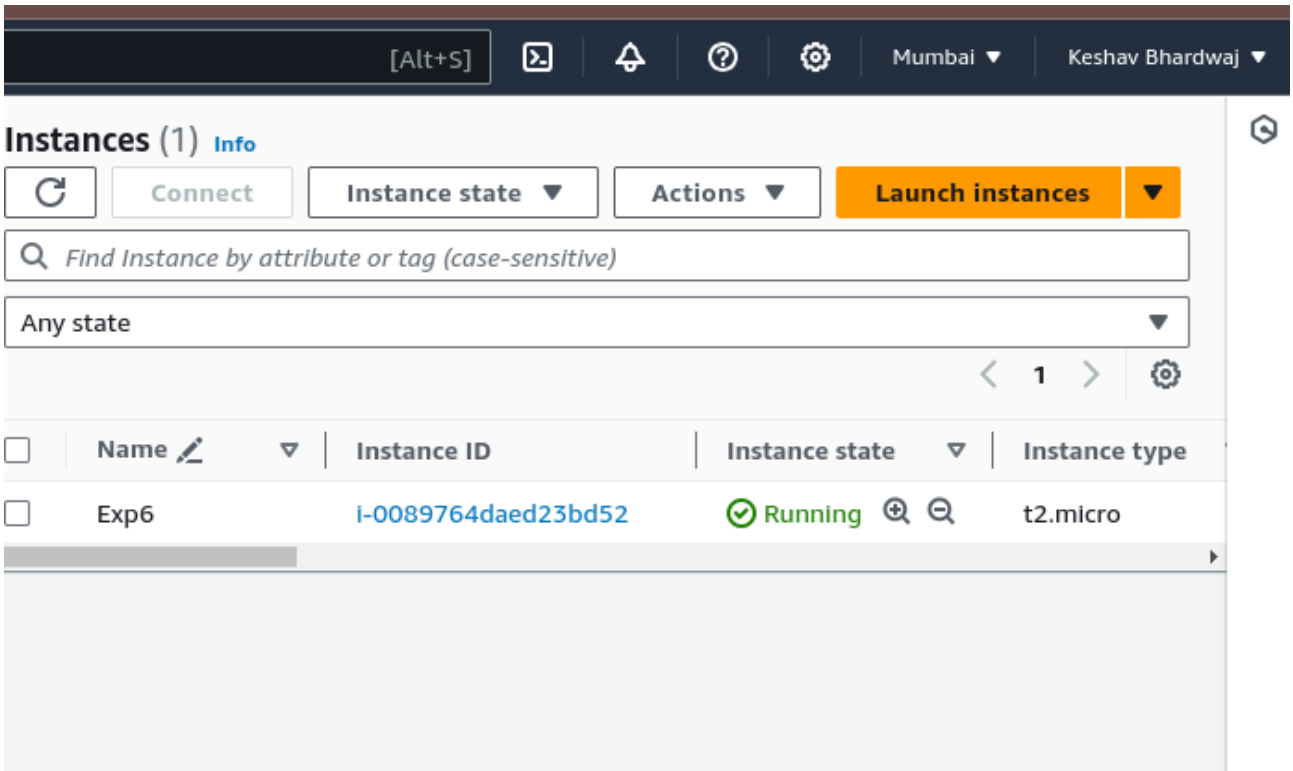
```
→  TERRAFORM  terraform apply -var-file=prod.tfvars
aws_instance.Keshav[0]: Refreshing state... [id=i-0089764daed23bd52]

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.Keshav[0] will be created
  + resource "aws_instance" "Keshav" {
```

```
  Enter a value: yes

aws_instance.Keshav[0]: Creating...
aws_instance.Keshav[0]: Still creating... [10s elapsed]
aws_instance.Keshav[0]: Still creating... [20s elapsed]
aws_instance.Keshav[0]: Creation complete after 27s [id=i-052052fbf949
d45ed]

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

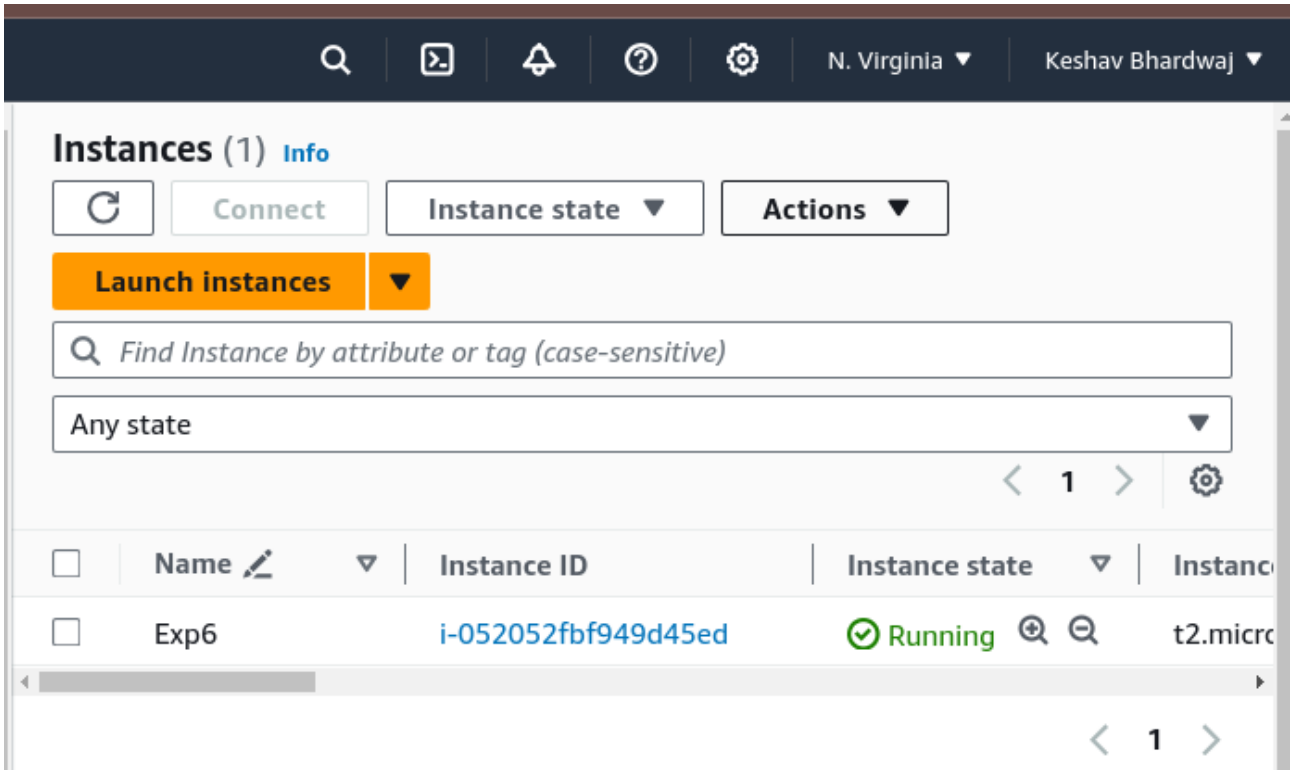
tfvars

4. Verify resources

a. dev



b. prod



5. Clean-up resources

```
  Enter a value: yes

aws_instance.Keshav[0]: Destroying... [id=i-0e46ec43416c47d4f]
aws_instance.Keshav[0]: Still destroying... [id=i-0e46ec43416c47d4f, 1
0s elapsed]
aws_instance.Keshav[0]: Still destroying... [id=i-0e46ec43416c47d4f, 2
0s elapsed]
aws_instance.Keshav[0]: Still destroying... [id=i-0e46ec43416c47d4f, 3
0s elapsed]
aws_instance.Keshav[0]: Destruction complete after 32s

Destroy complete! Resources: 1 destroyed.
```

Experiment 7

Creating Multiple IAM Users in Terraform

Aim

Learn how to use Terraform to create multiple IAM users with unique settings.

Steps

1. Create a main file & variables.tf file for EC2 Instance.

main.tf × variables.tf 1 terraform.tfstate.backup inst

main.tf > resource "aws_iam_user" "iam_users" > count

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.32.1"
    }
  }
}

provider "aws" {
    region = var.region
    access_key = var.access_key
    secret_key = var.secret_key
}

resource "aws_iam_user" "iam_users" {
  count= length(var.iam_users)
  name= var.iam_users[count.index]

  tags= {
    Name = "${var.iam_users[count.index]}-user"
  }

}
```

```
}

variable "iam_users" {
    type = list(string)
    default = [ "user1","user2","user3" ]

}

variable region {
    type = string
    default = "ap-south-1"
    description = "AWS Region"
}
```

2. Initialize & Apply the configuration

```
exp7  terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.32.1"...
- Installing hashicorp/aws v5.32.1...
- Installed hashicorp/aws v5.32.1 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record t
ovider
selections it made above. Include this file in your version contr
pository
so that Terraform can guarantee to make the same selections by de
 when
you run "terraform init" in the future.
```

```
exp7  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_iam_user.iam_users[0] will be created
  + resource "aws_iam_user" "iam_users" {
      + arn           = (known after apply)
      + force_destroy = false
      + id            = (known after apply)
      + name          = "user1"
      + path          = "/"
      + tags          = {
```

3. Verify the IAM users on AWS

- [ ] Terraform_user
- [ ] user1
- [ ] user2
- [ ] user3

4. Update the list of users to update the count of IAM Users on AWS

```
}

variable "iam_users" {
    type = list(string)
    default = [ "user1","keshav" ]

}
```

```
aws_iam_user.iam_users[1]: Modifying... [id=user2]
aws_iam_user.iam_users[2]: Destruction complete after 2s
aws_iam_user.iam_users[1]: Modifications complete after 2s [id=Ke

Apply complete! Resources: 0 added, 1 changed, 1 destroyed.
→  exp7
```

- [ ] Keshav
- [ ] Terraform_user
- [ ] user1

5. Clean up resources

```
aws_iam_user.iam_users[1]: Destroying... [id=Keshav]
aws_iam_user.iam_users[0]: Destroying... [id=user1]
aws_instance.Keshav[0]: Destroying... [id=i-0a39446cb5c1aad48]
aws_iam_user.iam_users[0]: Destruction complete after 2s
aws_iam_user.iam_users[1]: Destruction complete after 2s
aws_instance.Keshav[0]: Still destroying... [id=i-0a39446cb5c1aad48, 1
0s elapsed]
aws_instance.Keshav[0]: Still destroying... [id=i-0a39446cb5c1aad48, 2
0s elapsed]
aws_instance.Keshav[0]: Still destroying... [id=i-0a39446cb5c1aad48, 3
0s elapsed]
aws_instance.Keshav[0]: Destruction complete after 31s

Destroy complete! Resources: 3 destroyed.
```

Experiment 8

Creating a VPC in Terraform

Objective:

Learn how to use Terraform to create a basic Virtual Private Cloud (VPC) in AWS.

Steps:

1. Create Terraform Configuration Files:

- Create a file named main.tf

exp8 > main.tf > provider "aws" > region

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.32.1"
    }
  }
}

provider "aws" {
    region = var.region
    access_key = var.access_key
    secret_key = var.secret_key
}
```

exp8 > vpc.tf > resource "aws_subnet" "my_subnet" > tags > name

```
resource "aws_vpc" "my_vpc" {
cidr_block = "10.0.0.0/16"
enable_dns_support = true
enable_dns_hostnames = true
tags = {
    name = "my_vpc"
}
}
resource "aws_subnet" "my_subnet" {
    count = 2
    vpc_id = aws_vpc.my_vpc.id
    cidr_block = "10.0.${count.index}.0/24"
    availability_zone = "ap-south-1a"
    map_public_ip_on_launch = true
    tags = {
        name = "MySubnet-${count.index + 1}"
    }

}
```

2. Initialize and Apply:

```
exp8  terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.32.1"...
- Installing hashicorp/aws v5.32.1...
- Installed hashicorp/aws v5.32.1 (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.
exp8
```

```
exp8  terraform plan

Terraform used the selected providers to generate the following execution plan.
  + create

Terraform will perform the following actions:

  # aws_subnet.my_subnet[0] will be created
  + resource "aws_subnet" "my_subnet" {
      + arn                                            = (known after apply)
      + assign_ipv6_address_on_creation                = false
      + availability_zone                              = "ap-south-1a"
      + availability_zone_id                           = (known after apply)
      + cidr_block                                     = "10.0.0.0/24"
      + enable_dns64                                   = false
      + enable_resource_name_dns_a_record_on_launch    = false
      + enable_resource_name_dns_aaaa_record_on_launch = false
      + id                                             = (known after apply)
      + ipv6_cidr_block_association_id                 = (known after apply)
      + ipv6_native                                    = false
```

```
exp8  terraform apply

Terraform used the selected providers to generate the following execution p
  + create

Terraform will perform the following actions:

  # aws_subnet.my_subnet[0] will be created
  + resource "aws_subnet" "my_subnet" {
      + arn                                            = (known after apply
      + assign_ipv6_address_on_creation                = false
      + availability_zone                              = "ap-south-1a"
      + availability_zone_id                           = (known after apply
      + cidr_block                                     = "10.0.0.0/24"
      + enable_dns64                                   = false
      + enable_resource_name_dns_a_record_on_launch    = false
      + enable_resource_name_dns_aaaa_record_on_launch = false
```

3. Verifying Resources :

Subnets (2) Info

Find resources by attribute or tag

Name	Subnet ID	State	VPC
–	subnet-035c78e2c90d3d9b4	Available	vpc-07e0df8ce7ce8c4b3
–	subnet-0f7070aed309ac860	Available	vpc-07e0df8ce7ce8c4b3

Your VPCs (2) Info

Search

Name	VPC ID	State	IPv4 CIDR	IPv6 CID
–	vpc-06d95207b70e4bcb8	Available	172.31.0.0/16	–
–	vpc-07e0df8ce7ce8c4b3	Available	10.0.0.0/16	–

4. Clean up resources :

```
→  exp8  terraform destroy -auto-approve
aws_vpc.prod-vpc: Refreshing state... [id=vpc-07e0df8ce7ce8c4b3]
aws_subnet.prod-subnet[0]: Refreshing state... [id=subnet-035c78e2c90d3d9b4]
aws_subnet.prod-subnet[1]: Refreshing state... [id=subnet-0f7070aed309ac860]

Terraform used the selected providers to generate the following execution plan. R
  - destroy

Terraform will perform the following actions:

  # aws_subnet.prod-subnet[0] will be destroyed
  - resource "aws_subnet" "prod-subnet" {
      - arn                                            = "arn:aws:ec2:ap-south-1:
      - assign_ipv6_address_on_creation                = false -> null
      - availability_zone                              = "ap-south-1a" -> null
      - availability_zone_id                           = "aps1-az1" -> null
      - cidr_block                                     = "10.0.0.0/24" -> null
      - enable_dns64                                   = false -> null
```

```
      } -> null
    }

Plan: 0 to add, 0 to change, 3 to destroy.
aws_subnet.prod-subnet[0]: Destroying... [id=subnet-035c78e2c90d3d9b4]
aws_subnet.prod-subnet[1]: Destroying... [id=subnet-0f7070aed309ac860]
aws_subnet.prod-subnet[0]: Destruction complete after 0s
aws_subnet.prod-subnet[1]: Destruction complete after 0s
aws_vpc.prod-vpc: Destroying... [id=vpc-07e0df8ce7ce8c4b3]
aws_vpc.prod-vpc: Destruction complete after 1s

Destroy complete! Resources: 3 destroyed.
```

Experiment 9

Creating Multiple EC2 Instances with for each in Terraform

Objective:

Learn how to use for each in Terraform to create multiple AWS EC2 instances with specific settings for each instance.

Steps:

1. Create a Terraform Directory with main.tf and ec2.tf file

```
exp9 > main.tf > ...
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.32.1"
    }
  }
}
```

```
exp9 > ec2.tf > ...
provider "aws" {
  region = "us-east-1"
}

variable "Instances" {
  description = "Map of ec2 instances with settings"
  default = {
    "instance1" = {
      ami = "ami-0c55b159cbfafe1f0"
      instance_type = "t2.micro"
    },
    "instance2" = {
      ami = "ami-0e670eb768a5fc3d4"
      instance_type = "t2.micro"
    },
    "instance3" = {
      ami = "ami-0918bbd8513a9aa3b"
      instance_type = "t2.micro"
    }
  }
}

resource "aws_instance" "EXP-9" {
  for_each = var.Instances

  ami           = var.Instances[each.key].ami
  instance_type = var.Instances[each.key].instance_type

  tags = {
    Name = "Keshav-Instance-${each.key}"
  }
}
```

2. Initialize, validate, plan and apply the terraform repository

```
- Installed hashicorp/aws v5.32.1 (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.
→  exp9
```

```
→  exp9  terraform validate
Success! The configuration is valid.

→  exp9
```

```
→  exp9  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.EXP-9["instance1"] will be created
  + resource "aws_instance" "EXP-9" {
      + ami                                  = "ami-0c55b159cbfafe1f0"
      + 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)
```

```
      + tags                                 = {
          + "Name" = "EC2-Instance-instance3"
        }
      + tags_all                             = {
          + "Name" = "EC2-Instance-instance3"
        }
      + 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)
    }

Plan: 3 to add, 0 to change, 0 to destroy.
```

Apply :

```
→  exp9  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.EXP-9["instance1"] will be created
  + resource "aws_instance" "EXP-9" {
      + ami                                  = "ami-0c55b159cbfafe1f0"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
```

```
    }

  Plan: 3 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.EXP-9["instance1"]: Creating...
  aws_instance.EXP-9["instance2"]: Creating...
  aws_instance.EXP-9["instance3"]: Creating...
```

3. Clearing up resources :

```
→  exp8  terraform destroy -auto-approve
aws_vpc.prod-vpc: Refreshing state... [id=vpc-07e0df8ce7ce8c4b3]
aws_subnet.prod-subnet[0]: Refreshing state... [id=subnet-035c78e2c90d3d9b4]
aws_subnet.prod-subnet[1]: Refreshing state... [id=subnet-0f7070aed309ac860]

Terraform used the selected providers to generate the following execution plan. R
  - destroy

Terraform will perform the following actions:
```

```
    }

  Plan: 0 to add, 0 to change, 3 to destroy.
  aws_subnet.prod-subnet[0]: Destroying... [id=subnet-035c78e2c90d3d9b4]
  aws_subnet.prod-subnet[1]: Destroying... [id=subnet-0f7070aed309ac860]
  aws_subnet.prod-subnet[0]: Destruction complete after 0s
  aws_subnet.prod-subnet[1]: Destruction complete after 0s
  aws_vpc.prod-vpc: Destroying... [id=vpc-07e0df8ce7ce8c4b3]
  aws_vpc.prod-vpc: Destruction complete after 1s

  Destroy complete! Resources: 3 destroyed.
```

Experiment – 10

Creating an AWS RDS Instance in Terraform

Objective:

Learn how to use Terraform to create an AWS RDS instance.

Steps:

1. Create a Terraform Directory with configuration files:

rds.tf 2 main.tf

exp10 > main.tf > ...

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.32.1"
    }
  }
}
```

rds.tf 2 main.tf exp10 main.tf ./

exp10 > rds.tf > resource "aws_db_instance" "My-RDS"

```
provider "aws" {
region = "ap-south-1"
access_key = "AKIA232UVZYDMA5SK35U"
secret_key = "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"

}

resource "aws_db_instance" "My-RDS" {
  allocated_storage = 10
  identifier = "keshavdb"
  engine = "mysql"
  engine_version = "5.7"
  instance_class = "db.t2.micro"
  username = "keshavdb"
  password = "Keshav111"
  parameter_group_name = "default.mysql5.7"
  skip_final_snapshot = true
  publicly_accessible = true
  tags = {
    Name = "Myrdsdb"
  }
```

2. Initialize terraform repository :

```
Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.32.1"...
- Installing hashicorp/aws v5.32.1...
- Installed hashicorp/aws v5.32.1 (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.
→  exp10
```

3. Validate

```
→  exp10  terraform validate
Success! The configuration is valid.
```

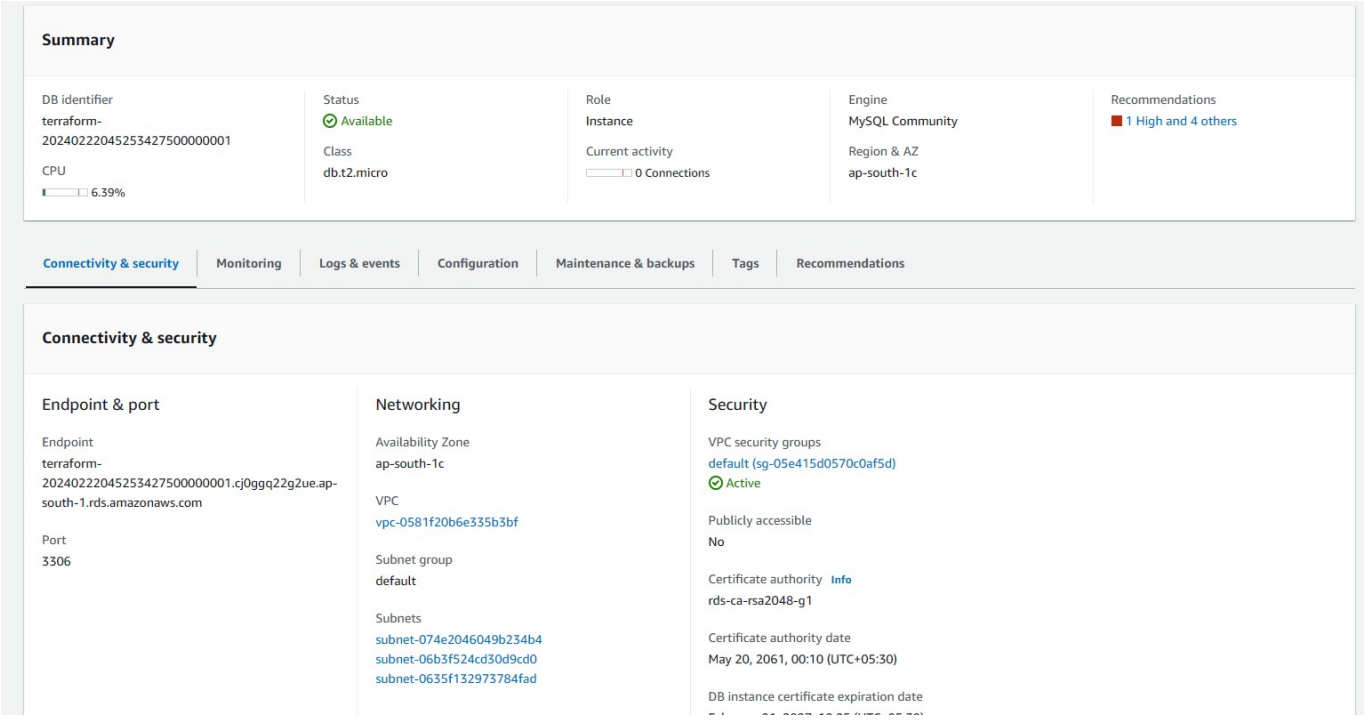
4. Plan

```
      + "Name" = "Myrdsdb"
    }
  + timezone                              = (known after apply)
  + username                              = "keshavdb"
  + vpc_security_group_ids                = (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
→  exp10
```

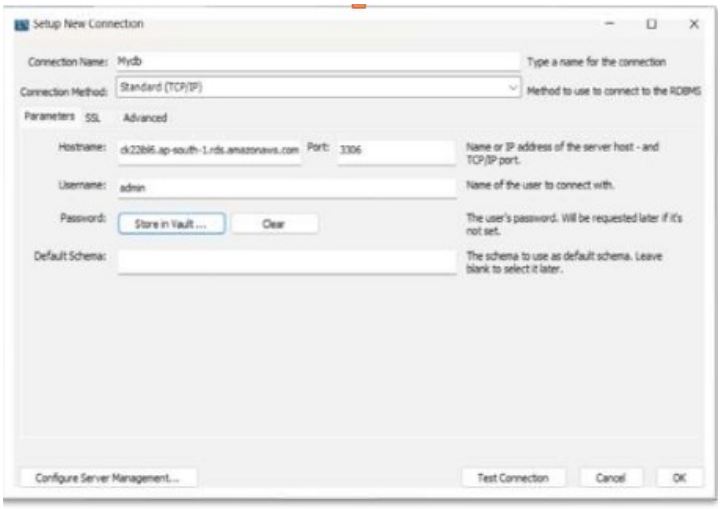
5. Apply

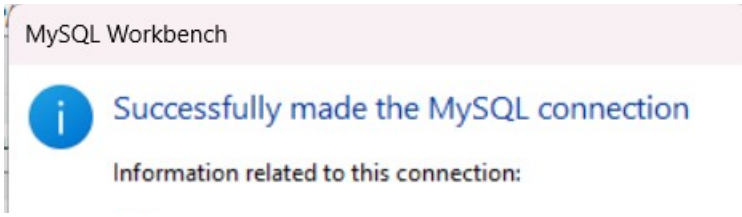
```
⊗ → exp10  terraform apply

Terraform used the selected providers to generate the following execution plan
  + create

Terraform will perform the following actions:

  # aws_db_instance.My-RDS will be created
  + resource "aws_db_instance" "My-RDS" {
      + address                               = (known after apply)
      + allocated_storage                     = 10
      + apply_immediately                     = false
      + arn                                   = (known after apply)
      + auto_minor_version_upgrade            = true
      + availability_zone                     = (known after apply)
```

6. Verifying resources :



7. Connect with MySQL Workbench with proper Configuration and save it.





8. Clean Up resources :

```
aws_db_instance.My-RDS: Still destroying... [id=db-H6GZ523XUALX23EB5TPPSTJCI4, 20s elapsed]
aws_db_instance.My-RDS: Still destroying... [id=db-H6GZ523XUALX23EB5TPPSTJCI4, 30s elapsed]
aws_db_instance.My-RDS: Still destroying... [id=db-H6GZ523XUALX23EB5TPPSTJCI4, 40s elapsed]
aws_db_instance.My-RDS: Still destroying... [id=db-H6GZ523XUALX23EB5TPPSTJCI4, 50s elapsed]
aws_db_instance.My-RDS: Still destroying... [id=db-H6GZ523XUALX23EB5TPPSTJCI4, 1m0s elapsed]
aws_db_instance.My-RDS: Still destroying... [id=db-H6GZ523XUALX23EB5TPPSTJCI4, 1m10s elapsed]
aws_db_instance.My-RDS: Still destroying... [id=db-H6GZ523XUALX23EB5TPPSTJCI4, 1m20s elapsed]
aws_db_instance.My-RDS: Still destroying... [id=db-H6GZ523XUALX23EB5TPPSTJCI4, 1m30s elapsed]
aws_db_instance.My-RDS: Still destroying... [id=db-H6GZ523XUALX23EB5TPPSTJCI4, 1m40s elapsed]
aws_db_instance.My-RDS: Still destroying... [id=db-H6GZ523XUALX23EB5TPPSTJCI4, 1m50s elapsed]
aws_db_instance.My-RDS: Still destroying... [id=db-H6GZ523XUALX23EB5TPPSTJCI4, 2m0s elapsed]
aws_db_instance.My-RDS: Still destroying... [id=db-H6GZ523XUALX23EB5TPPSTJCI4, 2m10s elapsed]
aws_db_instance.My-RDS: Still destroying... [id=db-H6GZ523XUALX23EB5TPPSTJCI4, 2m20s elapsed]
aws_db_instance.My-RDS: Still destroying... [id=db-H6GZ523XUALX23EB5TPPSTJCI4, 2m30s elapsed]
aws_db_instance.My-RDS: Still destroying... [id=db-H6GZ523XUALX23EB5TPPSTJCI4, 2m40s elapsed]
aws_db_instance.My-RDS: Still destroying... [id=db-H6GZ523XUALX23EB5TPPSTJCI4, 2m50s elapsed]
aws_db_instance.My-RDS: Still destroying... [id=db-H6GZ523XUALX23EB5TPPSTJCI4, 3m0s elapsed]
aws_db_instance.My-RDS: Still destroying... [id=db-H6GZ523XUALX23EB5TPPSTJCI4, 3m10s elapsed]
aws_db_instance.My-RDS: Still destroying... [id=db-H6GZ523XUALX23EB5TPPSTJCI4, 3m20s elapsed]
aws_db_instance.My-RDS: Still destroying... [id=db-H6GZ523XUALX23EB5TPPSTJCI4, 3m30s elapsed]
aws_db_instance.My-RDS: Still destroying... [id=db-H6GZ523XUALX23EB5TPPSTJCI4, 3m40s elapsed]
aws_db_instance.My-RDS: Still destroying... [id=db-H6GZ523XUALX23EB5TPPSTJCI4, 3m50s elapsed]
aws_db_instance.My-RDS: Still destroying... [id=db-H6GZ523XUALX23EB5TPPSTJCI4, 4m0s elapsed]
aws_db_instance.My-RDS: Still destroying... [id=db-H6GZ523XUALX23EB5TPPSTJCI4, 4m10s elapsed]
aws_db_instance.My-RDS: Destruction complete after 4m20s

Destroy complete! Resources: 1 destroyed.
```