Lab Exercise 17 – Terraform Multiple tfvars Files

Objective:

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

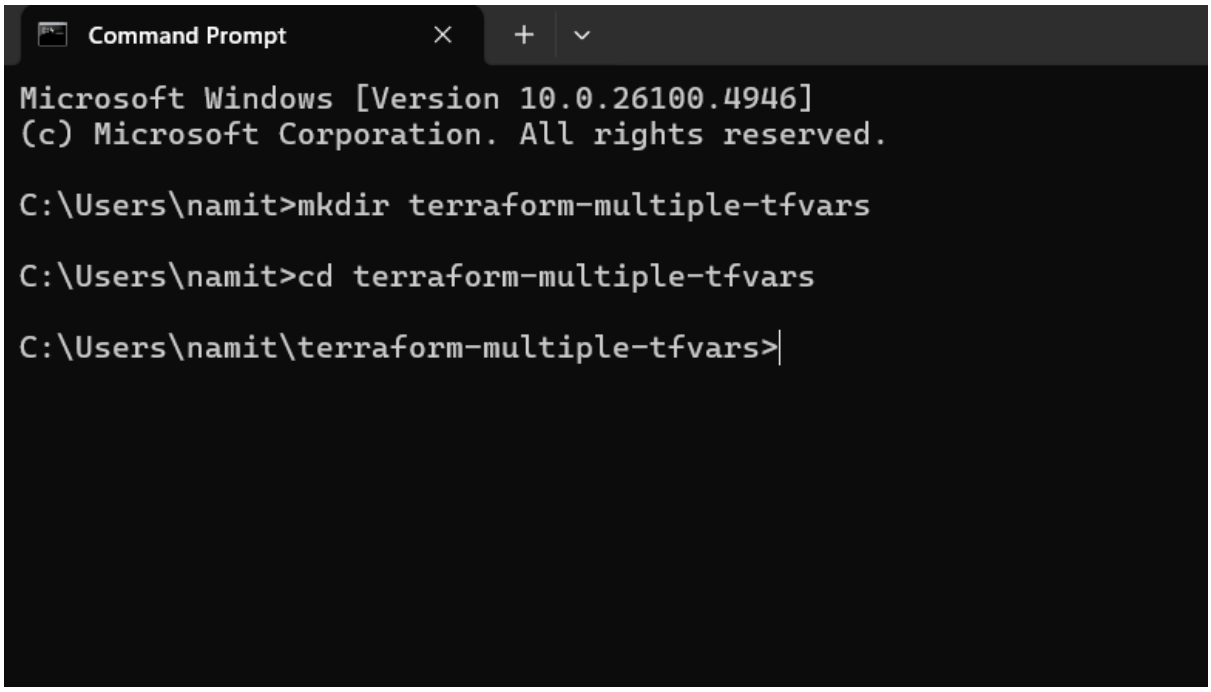
Prerequisites:

- Terraform installed on your machine.
- Basic knowledge of Terraform configuration and variables.

Steps:

1. Create a Terraform Directory:

```
mkdir terraform-multiple-tfvars
cd terraform-multiple-tfvars
```



- Create Terraform Configuration Files:
- Create a file named main.tf:

 # main.tf

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

resource "aws_instance" "example" {
  ami          = var.ami
  instance_type = var.instance_type
}
```

File Edit View

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

provider "aws" {
  region = var.region
}

resource "aws_instance" "example" {
  ami           = var.ami
  instance_type = var.instance_type

  tags = {
    Name = "Terraform-Example"
  }
}
```

- Create a file named variables.tf:

 # variables.tf

```
variable "ami" {
  type = string
}

variable "instance_ty" {
  type = string
}
```

File Edit View

```
variable "ami" {
  type = string
  description = "AMI ID to use for the instance"
}

variable "instance_type" {
  type = string
  description = "EC2 instance type"
}

variable "region" {
  type = string
  description = "AWS region to launch resources in"
}

variable "instance_count" {
  type    = number
  default = 1
  description = "How many EC2 instances to launch"
}
```

2. Create Multiple tfvars Files:

- Create a file named dev.tfvars:

 # dev.tfvars

```
ami         = "ami-0123456789abcdef0"
instance_type = "t2.micro"
```

```
ami           = "ami-0b982602dbb32c5bd"
instance_type = "t3.micro"
```

- Create a file named prod.tfvars:

 # prod.tfvars

```
ami         = "ami-9876543210fedcba0"
instance_type = "t2.large"
```

```
File    Edit    View

ami           = "ami-02d26659fd82cf299"
instance_type = "t3.micro"
```

- In these files, provide values for the variables based on the environments.

3. Initialize and Apply for Dev Environment:

- Run the following Terraform commands to initialize and apply the configuration for the dev environment:

```
terraform init
terraform apply -var-file=dev.tfvars
```

```
C:\Users\namit\terraform-multiple-tfvars>terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.13.0...
- Installed hashicorp/aws v6.13.0 (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.
```

```
C:\Users\namit\terraform-multiple-tfvars>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.example will be created
  + resource "aws_instance" "example" {
      + ami                                  = "ami-0b982602dbb32c5bd"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (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)
      + force_destroy                        = false
      + 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                        = "t3.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_group_id                   = (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)
```

4. Initialize and Apply for Prod Environment:

- Run the following Terraform commands to initialize and apply the configuration for the prod environment:

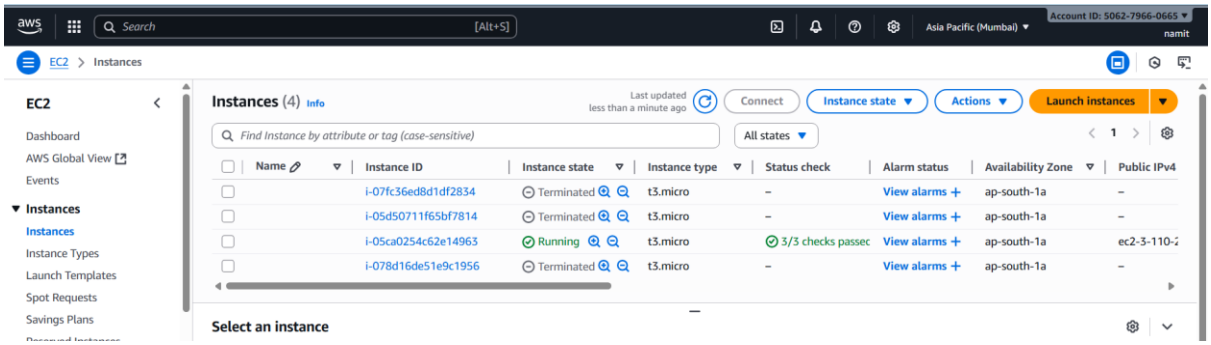
```
terraform init
terraform apply -var-file=prod.tfvars
```

```
Command Prompt

C:\Users\namit\terraform-multiple-tfvars>terraform apply -var-file="prod.tfvars"
aws_instance.example: Refreshing state... [id=i-05d50711f65bf7814]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
-/+ destroy and then create replacement

Terraform will perform the following actions:

  # aws_instance.example must be replaced
-/+ resource "aws_instance" "example" {
      ~ ami                                  = "ami-0b982602dbb32c5bd" -> "ami-02d26659fd82cf299" # forces replacement
      ~ arn                                  = "arn:aws:ec2:ap-south-1:506279660665:instance/i-05d50711f65bf7814" -> (known after apply)
      ~ associate_public_ip_address          = true -> (known after apply)
      ~ availability_zone                    = "ap-south-1a" -> (known after apply)
      ~ disable_api_stop                     = false -> (known after apply)
      ~ disable_api_termination              = false -> (known after apply)
      ~ ebs_optimized                        = false -> (known after apply)
      + enable_primary_ipv6                  = (known after apply)
      - hibernation                          = false -> null
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      ~ id                                   = "i-05d50711f65bf7814" -> (known after apply)
      ~ instance_initiated_shutdown_behavior = "stop" -> (known after apply)
      + instance_lifecycle                   = (known after apply)
      ~ instance_state                       = "running" -> (known after apply)
      ~ ipv6_address_count                   = 0 -> (known after apply)
      ~ ipv6_addresses                       = [] -> (known after apply)
      + key_name                             = (known after apply)
      ~ monitoring                           = false -> (known after apply)
      + outpost_arn                          = (known after apply)
      + password_data                        = (known after apply)
      + placement_group                      = (known after apply)
      + placement_group_id                   = (known after apply)
      ~ placement_partition_number           = 0 -> (known after apply)
      ~ primary_network_interface_id         = "eni-0bafa43d15056726f" -> (known after apply)
      ~ private_dns                          = "ip-172-31-40-155.ap-south-1.compute.internal" -> (known after apply)
      ~ private_ip                           = "172.31.40.155" -> (known after apply)
      ~ public_dns                           = "ec2-13-201-86-123.ap-south-1.compute.amazonaws.com" -> (known after apply)
      ~ public_ip                            = "13.201.86.123" -> (known after apply)
```

```
Command Prompt

      ~ root_block_device (known after apply)
      - root_block_device {
          - delete_on_termination = true -> null
          - device_name           = "/dev/xvda" -> null
          - encrypted             = false -> null
          - iops                  = 3000 -> null
          - tags                  = {} -> null
          - tags_all              = {} -> null
          - throughput            = 125 -> null
          - volume_id             = "vol-012f0e3bb97e32514" -> null
          - volume_size           = 8 -> null
          - volume_type           = "gp3" -> null
            # (1 unchanged attribute hidden)
        }
    }

Plan: 1 to add, 0 to change, 1 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.example: Destroying... [id=i-05d50711f65bf7814]
aws_instance.example: Still destroying... [id=i-05d50711f65bf7814, 00m10s elapsed]
aws_instance.example: Still destroying... [id=i-05d50711f65bf7814, 00m20s elapsed]
aws_instance.example: Still destroying... [id=i-05d50711f65bf7814, 00m30s elapsed]
aws_instance.example: Still destroying... [id=i-05d50711f65bf7814, 00m40s elapsed]
aws_instance.example: Still destroying... [id=i-05d50711f65bf7814, 00m50s elapsed]
aws_instance.example: Still destroying... [id=i-05d50711f65bf7814, 01m00s elapsed]
aws_instance.example: Still destroying... [id=i-05d50711f65bf7814, 01m10s elapsed]
aws_instance.example: Destruction complete after 1m11s
aws_instance.example: Creating...
aws_instance.example: Still creating... [00m10s elapsed]
aws_instance.example: Creation complete after 12s [id=i-05ca0254c62e14963]

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

C:\Users\namit\terraform-multiple-tfvars>
```

5. Test and Verify:

- Observe how different tfvars files are used to set variable values for different environments during the apply process.
- Access the AWS Management Console or use the AWS CLI to verify the creation of resources in the specified regions and instance types.



6. Clean Up:

- After testing, you can clean up resources:

terraform destroy -var-file=dev.tfvars
terraform destroy -var-file=prod.tfvars

- Confirm the destruction by typing yes.

```
Command Prompt
          - network_interface_id  = "eni-001290212ef9e0d9a" -> 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                  = 3000 -> null
          - tags                  = {} -> null
          - tags_all              = {} -> null
          - throughput            = 125 -> null
          - volume_id             = "vol-081d5923aed00391a" -> 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.example: Destroying... [id=i-05ca0254c62e14963]
aws_instance.example: Still destroying... [id=i-05ca0254c62e14963, 00m10s elapsed]
aws_instance.example: Still destroying... [id=i-05ca0254c62e14963, 00m20s elapsed]
aws_instance.example: Still destroying... [id=i-05ca0254c62e14963, 00m30s elapsed]
aws_instance.example: Destruction complete after 30s

Destroy complete! Resources: 1 destroyed.
```

```
C:\Users\namit\terraform-multiple-tfvars>terraform destroy -var-file=prod.tfvars

No changes. No objects need to be destroyed.

Either you have not created any objects yet or the existing objects were already deleted outside of Terraform.

Destroy complete! Resources: 0 destroyed.

C:\Users\namit\terraform-multiple-tfvars>
```

7. Conclusion:

This lab exercise demonstrates how to use multiple tfvars files in Terraform to manage variable values for different environments. It allows you to maintain separate configuration files for different environments, making it easier to manage and maintain your infrastructure code. Experiment with different values in the dev.tfvars and prod.tfvars files to observe how they impact the infrastructure provisioning process for each environment.