Lab Exercise 8– 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
}
```

 Create a file named variables.tf:

 # variables.tf

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

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

2. Create Multiple tfvars Files:

 Create a file named dev.tfvars:

 # dev.tfvars

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

 Create a file named prod.tfvars:

 # prod.tfvars

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

 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

```
PS E:\collagefiles\sem 6\system provisioning lab> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.30.0

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.
PS E:\collagefiles\sem 6\system provisioning lab>
```

terraform apply -var-file=dev.tfvars

```
      + 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

aws_instance.example: Creating...
aws_instance.example: Still creating... [10s elapsed]
aws_instance.example: Creation complete after 19s [id=i-0105d98c298cf4f60]

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

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

```
Plan: 0 to add, 1 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.example: Modifying... [id=i-0105d98c298cf4f60]
aws_instance.example: Still modifying... [id=i-0105d98c298cf4f60, 10s elapsed]
aws_instance.example: Still modifying... [id=i-0105d98c298cf4f60, 20s elapsed]
aws_instance.example: Still modifying... [id=i-0105d98c298cf4f60, 30s elapsed]
aws_instance.example: Still modifying... [id=i-0105d98c298cf4f60, 40s elapsed]
aws_instance.example: Still modifying... [id=i-0105d98c298cf4f60, 50s elapsed]
aws_instance.example: Still modifying... [id=i-0105d98c298cf4f60, 1m0s elapsed]
aws_instance.example: Still modifying... [id=i-0105d98c298cf4f60, 1m10s elapsed]
aws_instance.example: Still modifying... [id=i-0105d98c298cf4f60, 1m20s elapsed]
aws_instance.example: Still modifying... [id=i-0105d98c298cf4f60, 1m30s elapsed]
aws_instance.example: Modifications complete after 1m33s [id=i-0105d98c298cf4f60]

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

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

```
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-0105d98c298cf4f60]
aws_instance.example: Still destroying... [id=i-0105d98c298cf4f60, 10s elapsed]
aws_instance.example: Still destroying... [id=i-0105d98c298cf4f60, 21s elapsed]
aws_instance.example: Still destroying... [id=i-0105d98c298cf4f60, 31s elapsed]
aws_instance.example: Still destroying... [id=i-0105d98c298cf4f60, 41s elapsed]
aws_instance.example: Still destroying... [id=i-0105d98c298cf4f60, 51s elapsed]
aws_instance.example: Still destroying... [id=i-0105d98c298cf4f60, 1m1s elapsed]
aws_instance.example: Still destroying... [id=i-0105d98c298cf4f60, 1m11s elapsed]
aws_instance.example: Still destroying... [id=i-0105d98c298cf4f60, 1m21s elapsed]
aws_instance.example: Destruction complete after 1m25s

Destroy complete! Resources: 1 destroyed.
```

terraform destroy -var-file=prod.tfvars

```
PS E:\collagefiles\sem 6\system provisioning lab> 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.
PS E:\collagefiles\sem 6\system provisioning lab>
```

- Confirm the destruction by typing yes.

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.