

## 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:

```
C:\Users\OM VATS>mkdir terraform-multiple-tfvars  
C:\Users\OM VATS>cd terraform-multiple-tfvars
```

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

**# main.tf**

```
C:\Users\OM VATS\terraform-multiple-tfvars>notepad main.tf
```

- Create a file named variables.tf:

**# variables.tf**

```
C:\Users\OM VATS\terraform-multiple-tfvars>notepad variables.tf
```

```
variable "ami" {  
    type = string  
}  
  
variable "instance_type" {  
    type = string  
}  
  
variable "region" {  
    type = string  
}  
|
```

## 2. Create Multiple tfvars Files:

- Create a file named dev.tfvars:

**# dev.tfvars**

```
C:\Users\OM VATS\terraform-multiple-tfvars>notepad dev.tfvars
```

```
ami          = "ami-075d39ebbca89ed55"  
instance_type = "t2.micro"  
region       = "ap-south-1"  
|
```

- Create a file named prod.tfvars:

## # prod.tfvars

```
C:\Users\OM VATS\terraform-multiple-tfvars>notepad prod.tfvars
```

```
ami          = "ami-075d39ebbca89ed55"  
instance_type = "t2.large"  
region       = "ap-south-1"
```

- 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:

```
C:\Users\OM VATS\terraform-multiple-tfvars>terraform init  
Initializing the backend...  
Initializing provider plugins...  
  
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\OM VATS\terraform-multiple-tfvars>terraform apply -var-file=dev.tfvars  
  
No changes. Your infrastructure matches the configuration.
```

- Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.

## 4. Initialize and Apply for Prod Environment:

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

```
C:\Users\OM VATS\terraform-multiple-tfvars>terraform apply -var-file=prod.tfvars
No changes. Your infrastructure matches the configuration.
Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.
Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
C:\Users\OM VATS\terraform-multiple-tfvars>aws ec2 describe-instances --filters "Name=instance-state-name,Values=running"
{
  "Reservations": []
}
```

## 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.

```
Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
C:\Users\OM VATS\terraform-multiple-tfvars>aws ec2 describe-instances --filters "Name=instance-state-name,Values=running"
{
  "Reservations": []
}
```

## 6. Clean Up:

- After testing, you can clean up resources:

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

- 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.