

## 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  
}
```

```
➤(base) → terraform-multiple-tfvars cat main.tf
provider "aws" {
  region = var.region
}

resource "aws_instance" "example" {
  ami           = var.ami
  instance_type = var.instance_type
}
➤(base) → terraform-multiple-tfvars _
```

- Create a file named variables.tf:

**# variables.tf**

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

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

```
(base) → terraform-multiple-tfvars cat variables.tf
variable "ami" {
    type = string
}

variable "instance_type" {
    type = string
}

variable "region" {
    type = string
}
(base) → terraform-multiple-tfvars _
```

## 2. Create Multiple tfvars Files:

- Create a file named dev.tfvars:

# dev.tfvars

```
ami = "ami-0123456789abcdef0"
```

```
instance_type = "t2.micro"
```

```
(base) → terraform-multiple-tfvars cat dev.tfvars
ami = "ami-0447b33427d7585a6"
instance_type = "t2.micro"
(base) → terraform-multiple-tfvars _
```

- Create a file named prod.tfvars:

# prod.tfvars

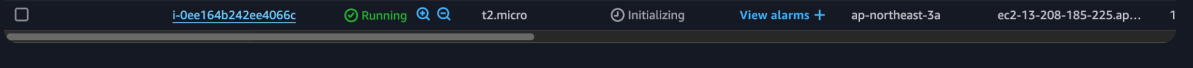
```
ami      = "ami-9876543210fedcbao"
instance_type = "t2.large"
instance_type = "t2.micro"
(base) → terraform-multiple-tfvars cat prod.tfvars
ami      = "ami-0447b33427d7585a6"
instance_type = "t2.large"
(base) → terraform-multiple-tfvars _
```

- 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
```



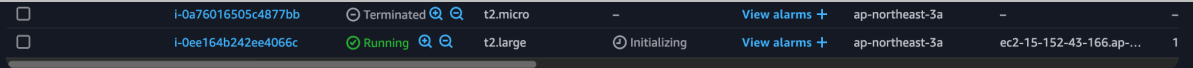
```
aws_instance.example: Creating...
aws_instance.example: Still creating... [10s elapsed]
aws_instance.example: Still creating... [20s elapsed]
aws_instance.example: Creation complete after 22s [id=i-0ee164b242ee4066c]

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-0ee164b242ee4066c]
aws_instance.example: Still modifying... [id=i-0ee164b242ee4066c, 10s elapsed]
aws_instance.example: Still modifying... [id=i-0ee164b242ee4066c, 20s elapsed]
aws_instance.example: Still modifying... [id=i-0ee164b242ee4066c, 30s elapsed]
aws_instance.example: Still modifying... [id=i-0ee164b242ee4066c, 40s elapsed]
aws_instance.example: Still modifying... [id=i-0ee164b242ee4066c, 50s elapsed]
aws_instance.example: Modifications complete after 53s [id=i-0ee164b242ee4066c]

Apply complete! Resources: 0 added, 1 changed, 0 destroyed.
(base) → 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
```

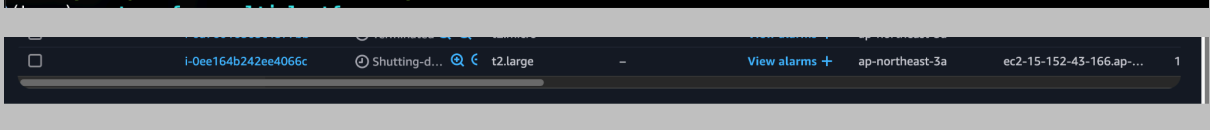
```
aws_instance.example: Destroying... [id=i-0ee164b242ee4066c]
aws_instance.example: Still destroying... [id=i-0ee164b242ee4066c, 10s elapsed]
aws_instance.example: Still destroying... [id=i-0ee164b242ee4066c, 20s elapsed]
aws_instance.example: Still destroying... [id=i-0ee164b242ee4066c, 30s elapsed]
aws_instance.example: Still destroying... [id=i-0ee164b242ee4066c, 40s elapsed]
aws_instance.example: Still destroying... [id=i-0ee164b242ee4066c, 50s elapsed]
aws_instance.example: Destruction complete after 58s

Destroy complete! Resources: 1 destroyed.
var.region
Enter a value: yes

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.
```



- 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.