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:

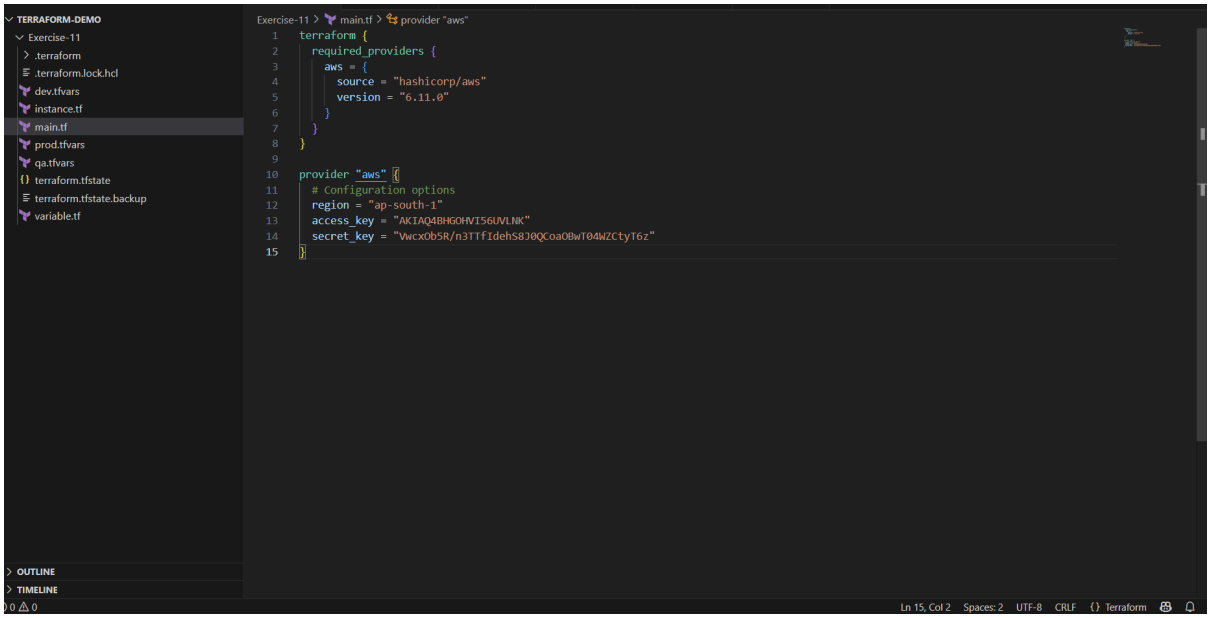
1. Create a Terraform Directory:

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

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

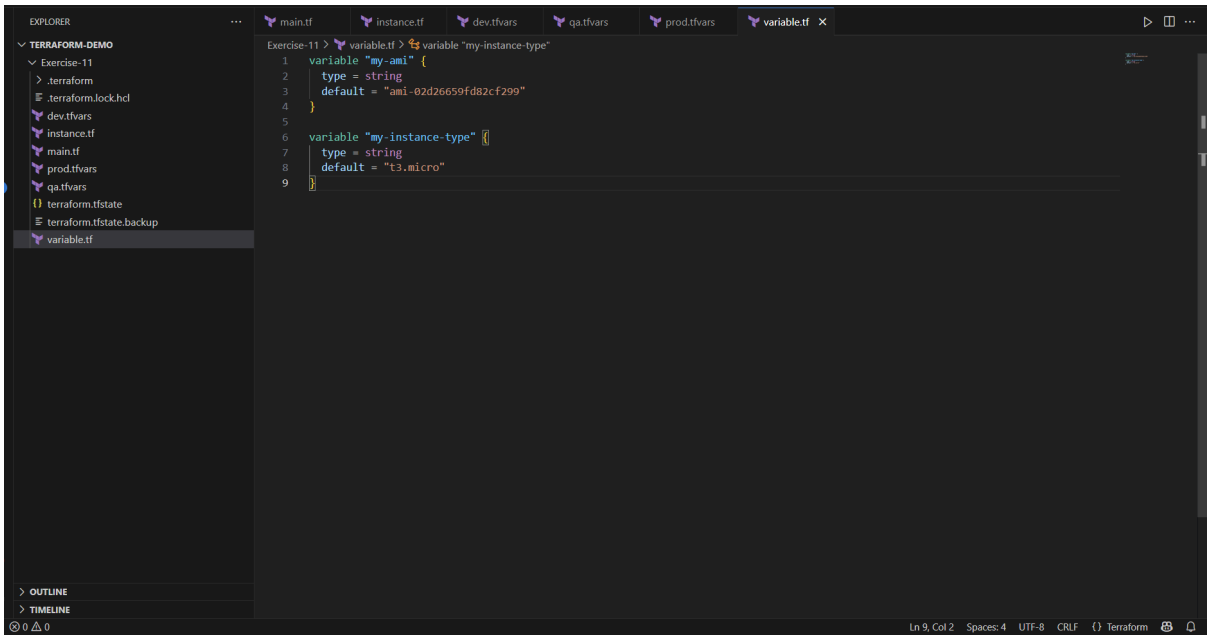
 # main.tf

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

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



- Create a file named variables.tf:

 # variables.tf




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

variable "instance_ty" {
```

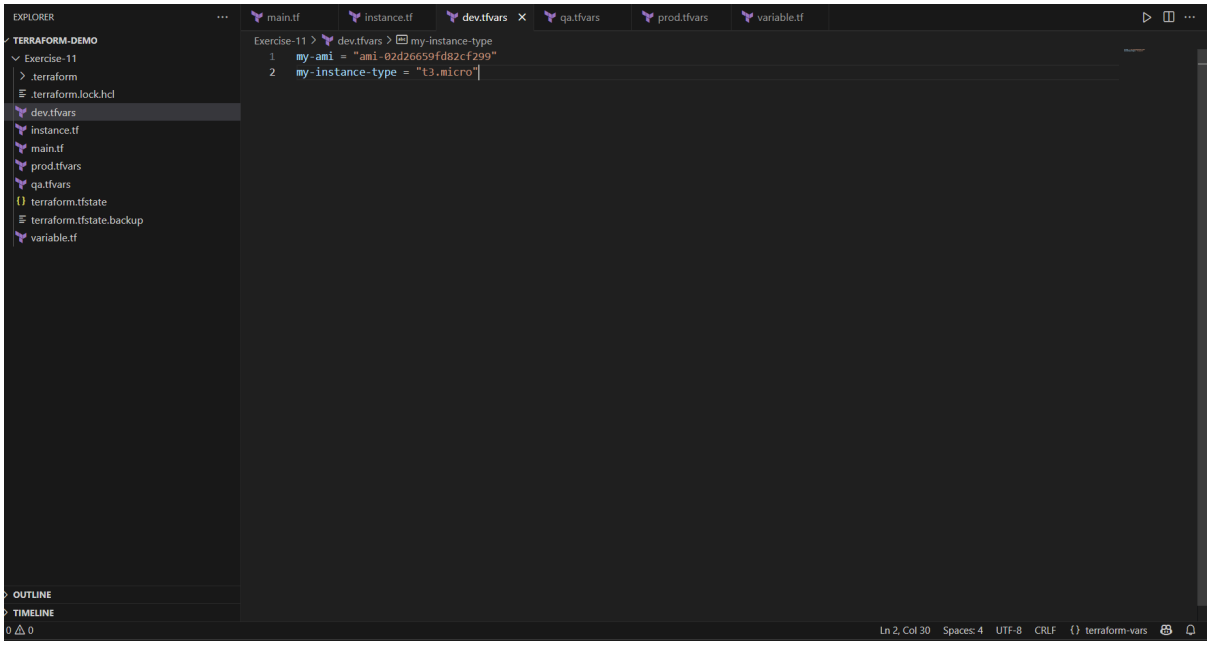
```
  type = string
}
```

2. Create Multiple tfvars Files:

- Create a file named dev.tfvars:

 # dev.tfvars

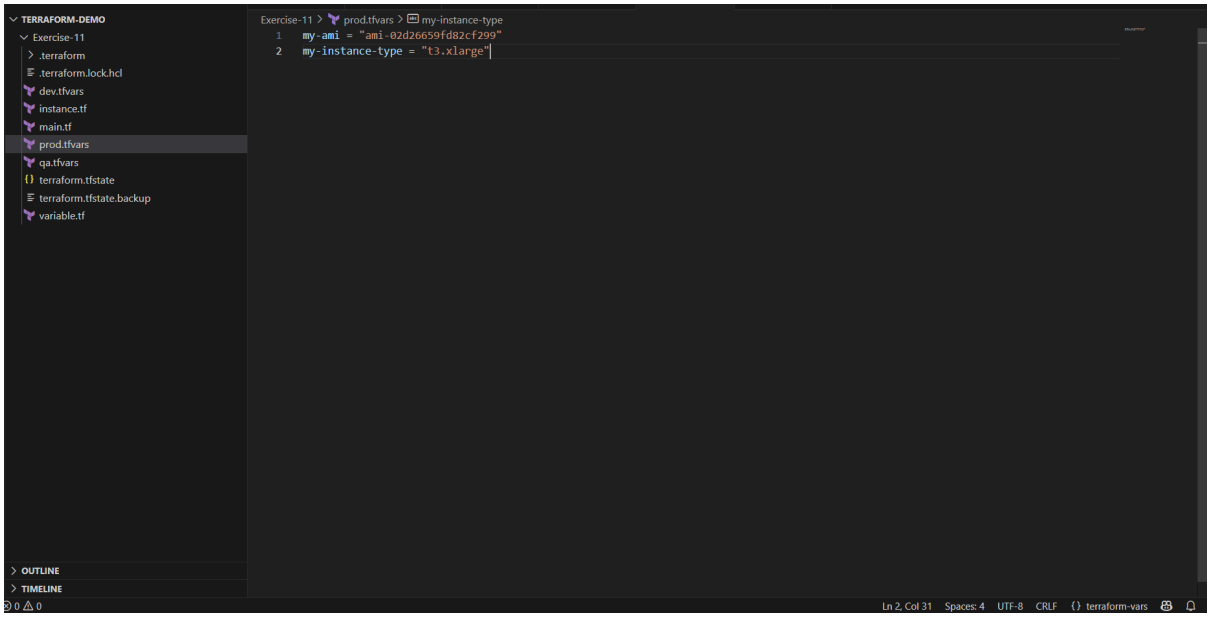
```
ami           = "ami-0123456789abcdef0"
instance_type = "t2.micro"
```



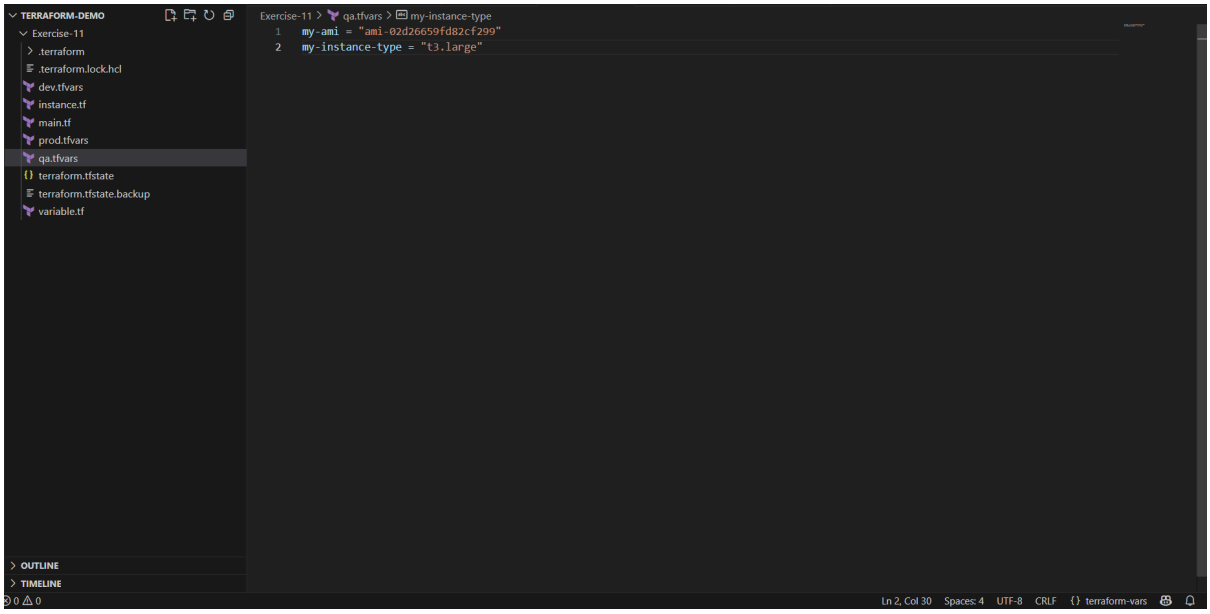
- Create a file named prod.tfvars:

 # 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:

```
C:\Me\DevSecOps\Terraform-demo\Exercise-11>terraform plan -var-file=prod.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.web1 will be created
  + resource "aws_instance" "web1" {
      + ami                                  = "ami-02d26659fd82cf299"
      + 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.xlarge"
      + 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_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)
```

terraform init

terraform apply -var-file=dev.tfvars

4. Initialize and Apply for Prod Environment:

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

```
PS C:\Me\DevSecOps\Terraform-demo\Exercise-11> terraform validate
Success! The configuration is valid.

PS C:\Me\DevSecOps\Terraform-demo\Exercise-11>
```

```
    + primary_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.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.

C:\Me\DevSecOps\Terraform-demo\Exercise-11>
```

terraform init

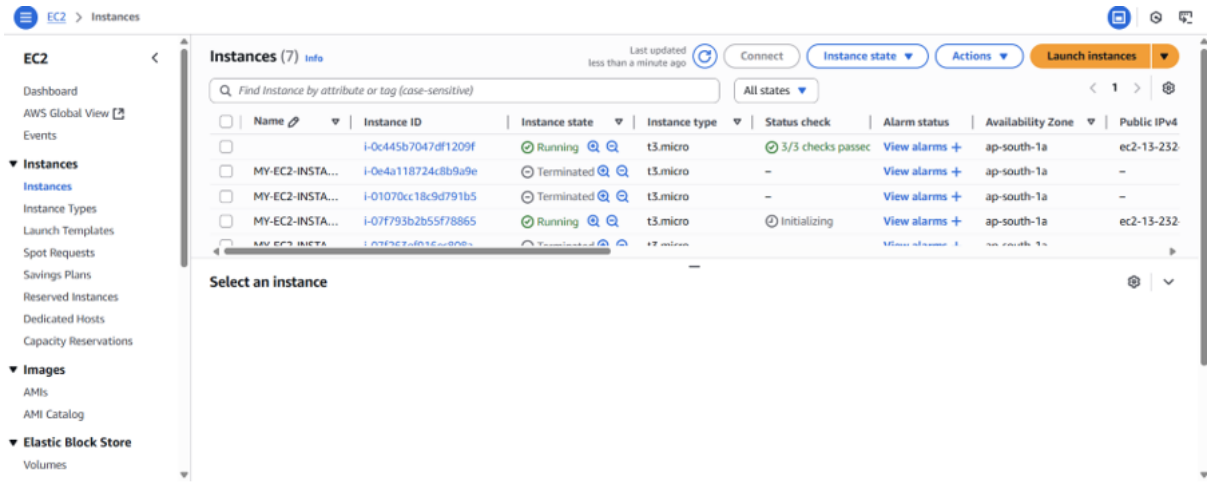
terraform apply -var-file=prod.tfvars

```
C:\Me\DevSecOps\Terraform-demo\Exercise-11>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.web1 will be created
  + resource "aws_instance" "web1" {
      + ami                                  = "ami-02d26659fd82cf299"
      + 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_partition_number           = (known after apply)
      + primary_network_interface_id         = (known after apply)
      + private_dns                          = (known after apply)
      + private_ip                           = (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.web1: Creating...
aws_instance.web1: Still creating... [00m10s elapsed]
aws_instance.web1: Creation complete after 15s [id=i-07f793b2b55f78865]

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

C:\Me\DevSecOps\Terraform-demo\Exercise-11>
```

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.

```
C:\Me\DevSecOps\Terraform-demo\Exercise-11>terraform destroy -var-file=dev.tfvars
aws_instance.web1: Refreshing state... [id=i-07f793b2b55f78865]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  # aws_instance.web1 will be destroyed
  - resource "aws_instance" "web1" {
      - ami                                  = "ami-02d26659fd82cf299" -> null
      - arn                                  = "arn:aws:ec2:ap-south-1:060211753450:instance/i-07f793b2b55f78865" -> null
      - associate_public_ip_address          = true -> null
      - availability_zone                    = "ap-south-1a" -> null
      - disable_api_stop                     = false -> null
      - disable_api_termination              = false -> null
      - ebs_optimized                        = false -> null
      - force_destroy                        = false -> null
      - get_password_data                    = false -> null
      - hibernation                          = false -> null
      - id                                   = "i-07f793b2b55f78865" -> null
      - instance_initiated_shutdown_behavior = "stop" -> null
      - instance_state                       = "running" -> null
      - instance_type                        = "t3.micro" -> null
      - ipv6_address_count                   = 0 -> null
      - ipv6_addresses                       = [] -> null
      - monitoring                           = false -> null
      - placement_partition_number           = 0 -> null
      - primary_network_interface_id         = "eni-0d11f6a8b28bcb2d4" -> null
      - private_dns                          = "ip-172-31-37-238.ap-south-1.compute.internal" -> null
      - private_ip                           = "172.31.37.238" -> null
      - public_dns                           = "ec2-13-232-4-255.ap-south-1.compute.amazonaws.com" -> 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.web1: Destroying... [id=i-07f793b2b55f78865]
aws_instance.web1: Still destroying... [id=i-07f793b2b55f78865, 00m10s elapsed]
aws_instance.web1: Still destroying... [id=i-07f793b2b55f78865, 00m20s elapsed]
aws_instance.web1: Still destroying... [id=i-07f793b2b55f78865, 00m30s elapsed]
aws_instance.web1: Still destroying... [id=i-07f793b2b55f78865, 00m40s elapsed]
aws_instance.web1: Still destroying... [id=i-07f793b2b55f78865, 00m50s elapsed]
aws_instance.web1: Still destroying... [id=i-07f793b2b55f78865, 01m00s elapsed]
aws_instance.web1: Destruction complete after 1m2s

Destroy complete! Resources: 1 destroyed.

C:\Me\DevSecOps\Terraform-demo\Exercise-11>
```

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.