# 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:

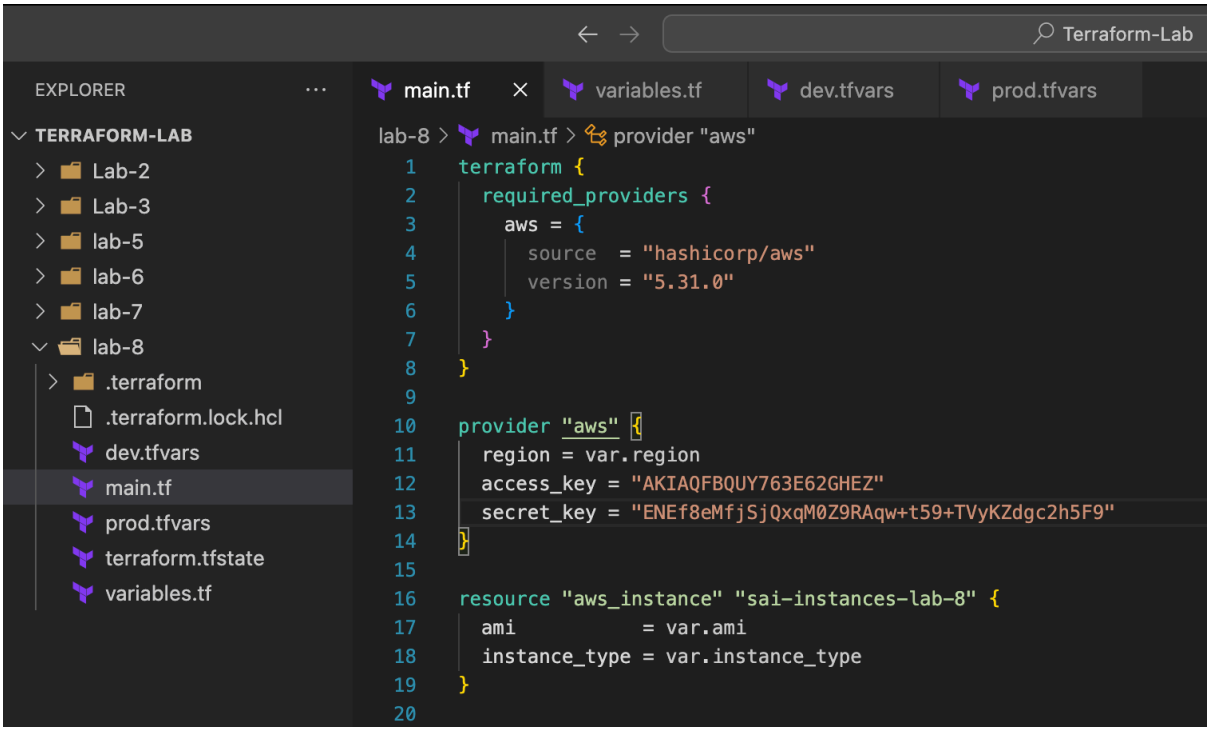
```
mkdir Terraform-Lab
cd Terraform-Lab
```

```
sai@Sais-Mac ~ % cd /Users/sai/Desktop/Terraform-Lab
sai@Sais-Mac Terraform-Lab %
```

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

**# main.tf**

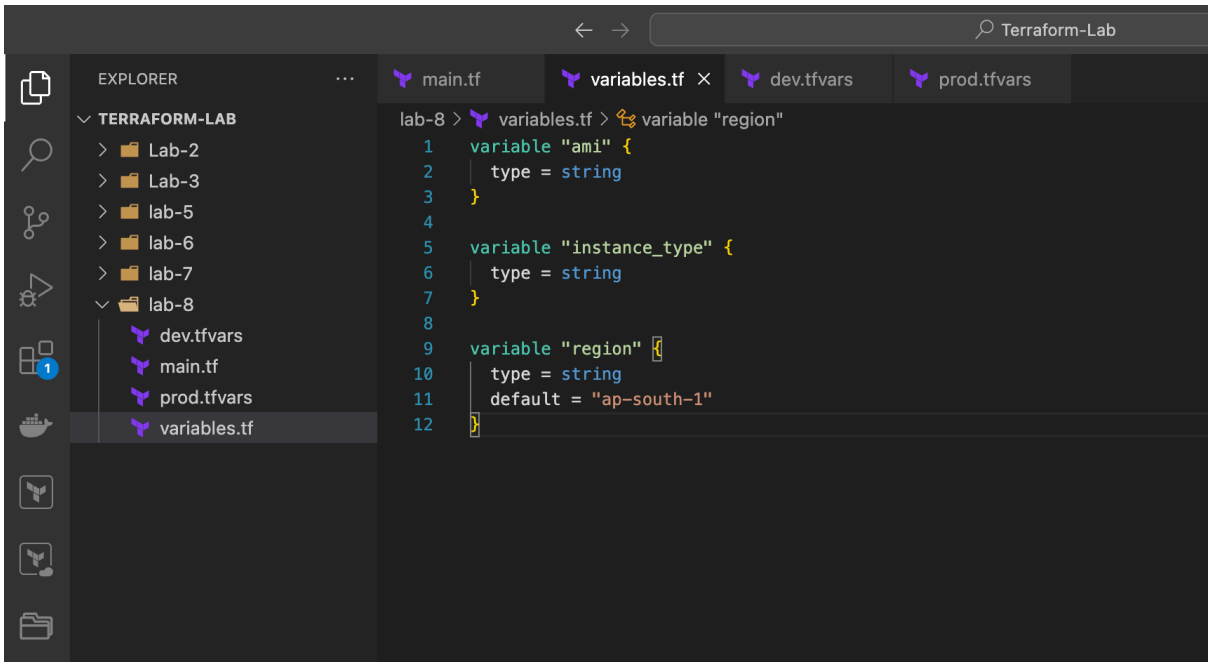
```
provider "aws" {
  region = var.region
}

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_type" {
   type = string
}

variable "region" {
   type = string
   default = “ap-south-1”
}
```



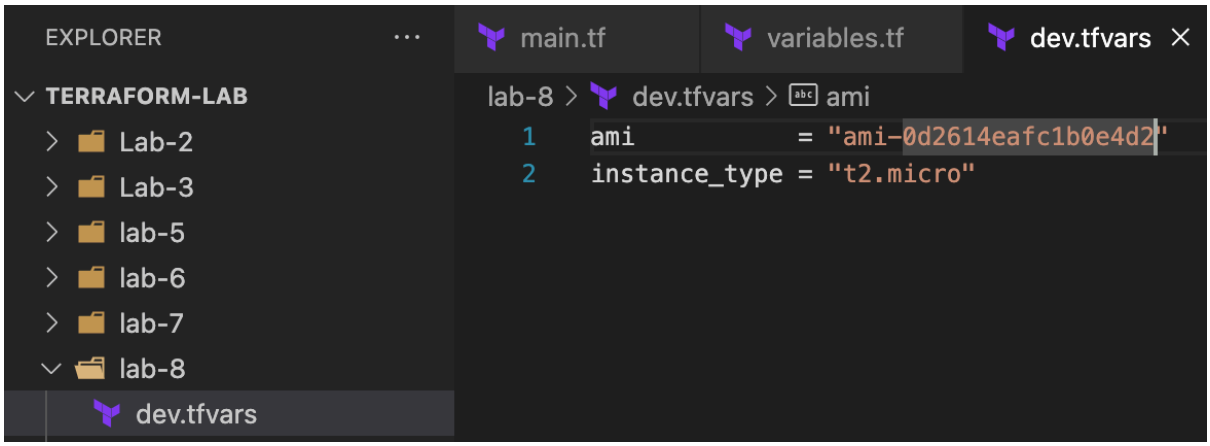

## 2. Create Multiple tfvars Files:

- Create a file named dev.tfvars:

**# dev.tfvars**

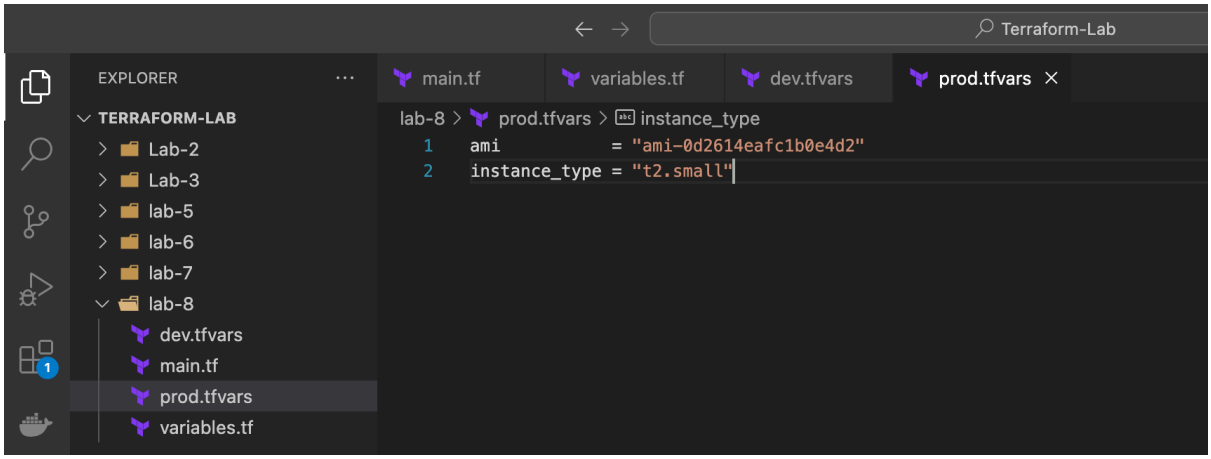
```
ami          = "ami-0123456789abcdef0"
instance_type = "t2.micro"
```

- Create a file named prod.tfvars:



**# prod.tfvars**

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




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

## 3. Initialize and Apply for Dev Environment:

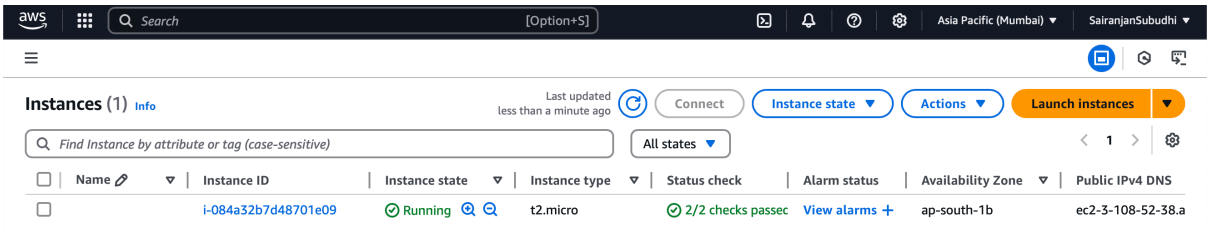
- Run the following Terraform commands to initialize and apply the configuration for the dev environment:

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

```
sai@Sais-Mac ~ % cd /Users/sai/Desktop/Terraform-Lab
sai@Sais-Mac Terraform-Lab % cd lab-8
sai@Sais-Mac lab-8 % terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.31.0"...
- Installing hashicorp/aws v5.31.0...
- Installed hashicorp/aws v5.31.0 (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.
sai@Sais-Mac lab-8 %
```

```
sai@Sais-Mac lab-8 % 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:
```

```
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.sai-instances-lab-8: Creating...
aws_instance.sai-instances-lab-8: Still creating... [10s elapsed]
aws_instance.sai-instances-lab-8: Creation complete after 13s [id=i-084a32b7d48701e09]

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



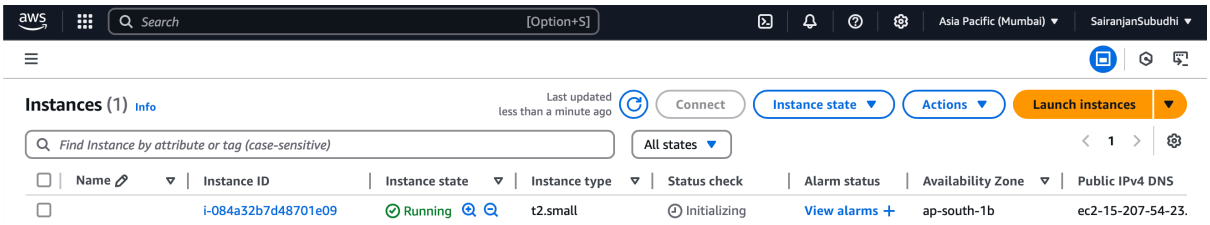

# 4. Initialize and Apply for Prod Environment:

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

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

```
sai@Sais-Mac lab-8 % terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.31.0

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.
```

```
sai@Sais-Mac lab-8 % terraform apply -var-file=prod.tfvars
aws_instance.sai-instances-lab-8: Refreshing state... [id=i-084a32b7d48701e09]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  ~ update in-place

Terraform will perform the following actions:

  # aws_instance.sai-instances-lab-8 will be updated in-place
  ~ resource "aws_instance" "sai-instances-lab-8" {
        id                                   = "i-084a32b7d48701e09"
      ~ instance_type                        = "t2.micro" -> "t2.small"
        tags                                 = {}
        # (38 unchanged attributes hidden)

        # (8 unchanged blocks hidden)
    }

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.sai-instances-lab-8: Modifying... [id=i-084a32b7d48701e09]
aws_instance.sai-instances-lab-8: Still modifying... [id=i-084a32b7d48701e09, 10s elapsed]
aws_instance.sai-instances-lab-8: Still modifying... [id=i-084a32b7d48701e09, 20s elapsed]
aws_instance.sai-instances-lab-8: Still modifying... [id=i-084a32b7d48701e09, 30s elapsed]
aws_instance.sai-instances-lab-8: Still modifying... [id=i-084a32b7d48701e09, 40s elapsed]
aws_instance.sai-instances-lab-8: Modifications complete after 43s [id=i-084a32b7d48701e09]

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:

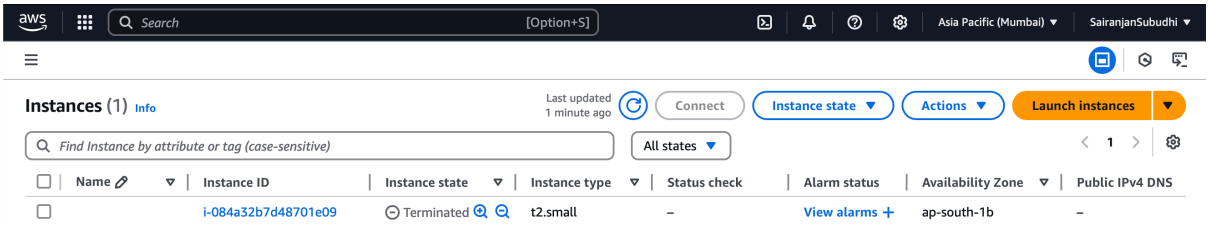
```
terraform destroy -var-file=dev.tfvars
terraform destroy -var-file=prod.tfvars
```

```
sai@Sais-Mac lab-8 % terraform destroy -var-file=dev.tfvars
aws_instance.sai-instances-lab-8: Refreshing state... [id=i-084a32b7d48701e09]

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:
```

- Confirm the destruction by typing yes.

```
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.sai-instances-lab-8: Destroying... [id=i-084a32b7d48701e09]
aws_instance.sai-instances-lab-8: Still destroying... [id=i-084a32b7d48701e09, 10s elapsed]
aws_instance.sai-instances-lab-8: Still destroying... [id=i-084a32b7d48701e09, 20s elapsed]
aws_instance.sai-instances-lab-8: Still destroying... [id=i-084a32b7d48701e09, 30s elapsed]
aws_instance.sai-instances-lab-8: Still destroying... [id=i-084a32b7d48701e09, 40s elapsed]
aws_instance.sai-instances-lab-8: Still destroying... [id=i-084a32b7d48701e09, 50s elapsed]
aws_instance.sai-instances-lab-8: Still destroying... [id=i-084a32b7d48701e09, 1m0s elapsed]
aws_instance.sai-instances-lab-8: Still destroying... [id=i-084a32b7d48701e09, 1m10s elapsed]
aws_instance.sai-instances-lab-8: Still destroying... [id=i-084a32b7d48701e09, 1m20s elapsed]
aws_instance.sai-instances-lab-8: Destruction complete after 1m21s

Destroy complete! Resources: 1 destroyed.
```

```
sai@Sais-Mac lab-8 % terraform destroy -var-file=prod.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.