

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
}
```



```
lab-8 > main.tf > provider "aws"
1  provider "aws" {
2    region = var.region
3  }
4  resource "aws_instance" "example" {
5    ami          = var.ami
6    instance_type = var.instance_type
7  }
```

- Create a file named variables.tf:

variables.tf

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

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

```
lab-8 >  variable.tf >  variable "instance_type"  
1   variable "ami" {  
2   |   type = string  
3   |   }  
4   variable "instance_type" {  
5   |   type = string  
6   |   }  
7   variable "region" {  
8   |   type = string  
9   |   }
```



2. Create Multiple tfvars Files:

- Create a file named dev.tfvars:

dev.tfvars

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

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

```
lab-8 >  dev.tfvars >  instance_type
```

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

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

```
3 region = "ap-south-1"
```

- Create a file named prod.tfvars:

prod.tfvars

```
ami = "ami-9876543210fedcba0"
```

```
instance_type = "t2.large"
```

```
lab-8 >  prod.tfvars > ...
```

```
1 ami = "ami-9876543210fedcba0"
```

```
2 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

```
[dhruvchaubey@Dhruvs-MacBook-Pro 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 v5.84.0

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.
[dhruvchaubey@Dhruvs-MacBook-Pro lab-8 % terraform validate
Success! The configuration is valid.
```

terraform apply -var-file=dev.tfvars

```
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: Still creating... [20s elapsed]
aws_instance.example: Creation complete after 20s [id=i-08efc8c1fc70aa797]

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

dhruvchaubey@Dhruvs-MacBook-Pro lab-8 % terraform apply -var-file=prod.tfvars
var.region
  Enter a value: ap-south-1

aws_instance.example: Refreshing state... [id=i-08efc8c1fc70aa797]

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.
```

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.

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
<input type="checkbox"/>		i-08efc8c1fc70aa797	Running	t2.micro	2/2 checks passed View alarms +		ap-south-1b	ec2-13-126-55-62.ap-s...

6. Clean Up:

- After testing, you can clean up resources:

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

Enter a value: yes
aws_instance.example: Destroying... [id=i-08efc8c1fc70aa797]
aws_instance.example: Still destroying... [id=i-08efc8c1fc70aa797, 10s elapsed]
aws_instance.example: Still destroying... [id=i-08efc8c1fc70aa797, 20s elapsed]
aws_instance.example: Still destroying... [id=i-08efc8c1fc70aa797, 30s elapsed]
aws_instance.example: Still destroying... [id=i-08efc8c1fc70aa797, 40s elapsed]
aws_instance.example: Still destroying... [id=i-08efc8c1fc70aa797, 50s elapsed]
aws_instance.example: Still destroying... [id=i-08efc8c1fc70aa797, 1m0s elapsed]
aws_instance.example: Still destroying... [id=i-08efc8c1fc70aa797, 1m10s elapsed]
aws_instance.example: Destruction complete after 1m12s

Destroy complete! Resources: 1 destroyed.

terraform destroy -var-file=prod.tfvars

Destroy complete! Resources: 1 destroyed.
[dhruvchaubey@Dhruvs-MacBook-Pro lab-8 % terraform destroy -var-file=prod.tfvars
var.region
Enter a value: ap-south-1

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.