



System Provisioning and Configuration Management LAB

SUBMITTED TO

Dr. Hitesh Kumar Sharma

SUBMITTED BY

Siddharth Agarwal

500107594

R2142220663

Btech CSE DevOps B1

Lab Exercise 8– Terraform Multiple tfvars Files

Objective:

Learn how to use multiple tfvars files in Terraform for different environments.

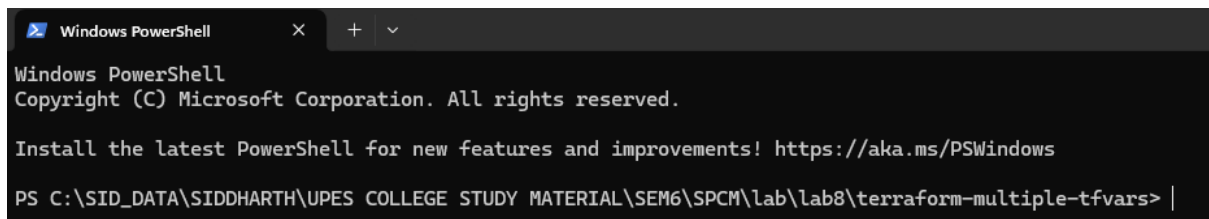
Prerequisites:

- 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
```



A screenshot of a Windows PowerShell terminal window. The title bar says 'Windows PowerShell'. The window content shows the following text: 'Windows PowerShell', 'Copyright (C) Microsoft Corporation. All rights reserved.', 'Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows', and the command prompt 'PS C:\SID_DATA\SIDDHARTH\UPES COLLEGE STUDY MATERIAL\SEM6\SPCM\lab\lab8\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  
}
```

```
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  
}
```

```
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

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

```
instance_type = "t2.micro"

ami          = "ami-09a9858973b288bdd"
instance_type = "t3.micro"
```

- Create a file named prod.tfvars:

prod.tfvars

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

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:

terraform init

```
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.
```

terraform apply -var-file=dev.tfvars

```
Plan: 1 to add, 0 to change, 0 to destroy.

Warning: Value for undeclared variable

The root module does not declare a variable named "instance_type" but a value was found in
file "dev.tfvars". If you meant to use this value, add a "variable" block to the
configuration.

To silence these warnings, use TF_VAR_... environment variables to provide certain "global"
settings to all configurations in your organization. To reduce the verbosity of these
warnings, use the -compact-warnings option.

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: Creation complete after 14s [id=i-0d236974aec0dd7f8]

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: 1 to add, 0 to change, 0 to destroy.
```

```
Warning: Value for undeclared variable
```

```
The root module does not declare a variable named "instance_type" but a value was found in file "dev.tfvars". If you meant to use this value, add a "variable" block to the configuration.
```

```
To silence these warnings, use TF_VAR_... environment variables to provide certain "global" settings to all configurations in your organization. To reduce the verbosity of these warnings, use the -compact-warnings option.
```

```
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: Creation complete after 14s [id=i-0d236974aec0dd7f8]
```

```
Apply complete! Resources: 1 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.

6. Clean Up:

- After testing, you can clean up resources:

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

```
Plan: 0 to add, 0 to change, 1 to destroy.
aws_instance.example: Destroying... [id=i-09bcb2489c9b019bf]
aws_instance.example: Still destroying... [id=i-09bcb2489c9b019bf, 10s elapsed]
aws_instance.example: Still destroying... [id=i-09bcb2489c9b019bf, 20s elapsed]
aws_instance.example: Still destroying... [id=i-09bcb2489c9b019bf, 30s elapsed]
aws_instance.example: Still destroying... [id=i-09bcb2489c9b019bf, 40s elapsed]
aws_instance.example: Still destroying... [id=i-09bcb2489c9b019bf, 50s elapsed]
aws_instance.example: Still destroying... [id=i-09bcb2489c9b019bf, 1m0s elapsed]
aws_instance.example: Still destroying... [id=i-09bcb2489c9b019bf, 1m10s elapsed]
aws_instance.example: Destruction complete after 1m11s

Destroy complete! Resources: 1 destroyed.
```

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

```
Plan: 0 to add, 0 to change, 1 to destroy.
aws_instance.example: Destroying... [id=i-09bcb2489c9b019bf]
aws_instance.example: Still destroying... [id=i-09bcb2489c9b019bf, 10s elapsed]
aws_instance.example: Still destroying... [id=i-09bcb2489c9b019bf, 20s elapsed]
aws_instance.example: Still destroying... [id=i-09bcb2489c9b019bf, 30s elapsed]
aws_instance.example: Still destroying... [id=i-09bcb2489c9b019bf, 40s elapsed]
aws_instance.example: Still destroying... [id=i-09bcb2489c9b019bf, 50s elapsed]
aws_instance.example: Still destroying... [id=i-09bcb2489c9b019bf, 1m0s elapsed]
aws_instance.example: Still destroying... [id=i-09bcb2489c9b019bf, 1m10s elapsed]
aws_instance.example: Destruction complete after 1m11s

Destroy complete! Resources: 1 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.