Lab 6 – Assigning Variable Values Using tfvars

Creater : Sandeep Kumar Sharma

Learning Objectives

- Understand why we need external files to supply variable values.
- Learn what a `.tfvars` file is in Terraform.
- Learn how to create and use a `terraform.tfvars` file.
- Learn how to pass variables during `plan` and `apply` without editing the main Terraform files.
- Deploy AWS resources using variable values from external files.

Learning Outcome

By the end of this lab, the learner will be able to:

- Explain the purpose of tfvars files.
- Create and use `terraform.tfvars`.
- Override default variable values.
- Run Terraform with clean, production-friendly variable management.

Concept Explanation (Natural Style)

In the previous lab (Lab 5), we learned what variables are and how to declare them.

Now imagine a real company situation.

You have one Terraform code, but you need to deploy in:

- Dev
- QA
- Stage
- Production

Each environment has different values:

- different instance sizes
- different region
- different naming convention

You cannot change the variable values inside the code every time — this is risky and unprofessional.

That's why Terraform gives us **tfvars**.

👉What is a tfvars file?

A tfvars file is simply a file where you put the values of variables.

Terraform automatically reads this file and injects the values.

It helps you:

- Keep your code clean
- Change environment values easily
- Avoid modifying main code
- Reuse same Terraform code for multiple environments

Think of it like giving ingredients for cooking in a separate bowl instead of mixing everything into the main recipe.

😠Part 1: Project Setup

Step 1: Create the Lab Folder

```
mkdir terraform-lab6-tfvars
cd terraform-lab6-tfvars
```

Step 2: Create Files

```
touch main.tf
touch variables.tf
touch terraform.tfvars
```

😠Part 2: Write Terraform Code

main.tf

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

resource "aws_instance" "lab6_ec2" {
```

```
  ami           = "ami-052c08d70def0ac62"  # Amazon Linux 2 Mumbai
  instance_type = var.instance_type

  tags = {
    Name = var.ec2_name
  }
}
```

This uses three variables.

---

variables.tf

```
variable "aws_region" {
  description = "AWS region"
  type        = string
}

variable "instance_type" {
  description = "EC2 size"
  type        = string
}

variable "ec2_name" {
  description = "Name tag for EC2 instance"
  type        = string
}
```

We didn't give default values because we want them to be supplied through the tfvars file.

😠Part 3: Create the tfvars File

Now open **terraform.tfvars** and add the variable values:

terraform.tfvars

```
aws_region    = "ap-south-1"
instance_type = "t2.micro"
ec2_name      = "Terraform-Lab6-EC2"
```

This file will feed values into your Terraform configuration.

😠Part 4: Run Terraform With tfvars

Terraform will automatically detect `terraform.tfvars`.

Step 1: Initialize

```
terraform init
```

Step 2: Plan

```
terraform plan
```

Terraform will show the values it picked from tfvars.

Step 3: Apply

```
terraform apply
```

Type **yes**.

AWS EC2 instance will be created with:

- region from tfvars
- instance type from tfvars
- name tag from tfvars

😠Part 5: Passing tfvars Manually (Optional)

You can also pass a tfvars file manually:

```
terraform apply -var-file="terraform.tfvars"
```

Or use a different file:

```
touch prod.tfvars
```

Add production values:

```
aws_region   = "us-east-1"
instance_type = "t3.medium"
ec2_name      = "Prod-Server"
```

Run:

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

This is how companies manage multiple environments.

😠Part 6: Destroy the Infrastructure (Optional)

```
terraform destroy
```

Type **yes**.

Summary

In this lab, you learned:

- What tfvars files are.
- Why tfvars is used to separate values from code.
- How to create and use `terraform.tfvars`.
- How to manage multiple environments using tfvars.
- How to apply and override values during run time.

End of Lab 6