Lab 9: Terraform Variables with Separate Files (variables.tf + terraform.tfvars + main.tf)

Author: Dr. Sandeep Kumar Sharma
Level: Beginner
Platform: Ubuntu Linux + Microsoft Azure
Prerequisite: Lab 1 to Lab 8

Concept Explanation (Important)

What is a Variable in Terraform?

A **variable** in Terraform is a named value holder.

It is used to store data that can be reused in multiple places.

Simple Meaning:

A variable is a **container for a value** that can change without changing the code.

Why Variables are Used

Variables are used to:

- Avoid hardcoding values
- Make Terraform code reusable
- Make infrastructure flexible
- Support multiple environments (dev, test, prod)
- Centralize configuration
- Improve maintainability

What is variables.tf?

`variables.tf` is a file where **variables are declared**.

Purpose of variables.tf:

- Define variable names
- Define variable types

- Define descriptions
- Set default values (optional)

Example Meaning:

`variables.tf` = **Definition file**
(Only structure of variables, not real environment values)

What is terraform.tfvars?

`terraform.tfvars` is a file where **actual values of variables are defined**.

Purpose of terraform.tfvars:

- Store real values of variables
- Separate code from configuration
- Change environment values without touching code
- Support environment-based deployments

Example Meaning:

`terraform.tfvars` = **Value file**
(Actual data for variables)

Difference Between variables.tf and terraform.tfvars

File	Purpose
variables.tf	Declare variables (structure)
terraform.tfvars	Define variable values (data)

Why terraform.tfvars is Important (Use Case)

1. Environment Separation

- dev.tfvars
- test.tfvars
- prod.tfvars

Same code, different values.

---

2. Configuration Management

Values change, code stays same.

---

3. Security

Sensitive values can be stored separately.

---

4. Reusability

One Terraform codebase → Multiple deployments.

---

5. Automation

Used in CI/CD pipelines easily.

Hands-On Lab

Objective

Create a Linux Virtual Machine using:

- `variables.tf`
- `terraform.tfvars`
- `main.tf`

Step 1: Go to Terraform Directory

```
cd terraform-azure-lab
```

Step 2: Create Files

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

```
touch main.tf
```

Step 3: variables.tf

```
variable "rg_name" {
  description = "Resource Group name"
  type        = string
}

variable "location" {
  description = "Azure region"
  type        = string
}

variable "vnet_name" {
  description = "Virtual Network name"
  type        = string
}

variable "subnet_name" {
  description = "Subnet name"
  type        = string
}

variable "nsg_name" {
  description = "Network Security Group name"
  type        = string
}

variable "public_ip_name" {
  description = "Public IP name"
  type        = string
}

variable "nic_name" {
  description = "NIC name"
  type        = string
}

variable "vm_name" {
  description = "Virtual Machine name"
  type        = string
}

variable "admin_username" {
  description = "VM admin username"
  type        = string
}

variable "vm_size" {
  description = "VM size"
  type        = string
}
```

Step 4: terraform.tfvars

```
rg_name         = "rg-terraform-lab"
location        = "East US"

vnet_name       = "vnet-tfvars-lab"
subnet_name     = "subnet-tfvars-lab"
nsg_name        = "nsg-tfvars-lab"

public_ip_name = "pip-tfvars-lab"
nic_name        = "nic-tfvars-lab"

vm_name         = "Sandeep-machine-tfvars"
admin_username = "azureuser"
vm_size         = "Standard_B1s"
```

Step 5: main.tf

```
terraform {
  required_providers {
    azurerm = {
      source  = "hashicorp/azurerm"
      version = "~> 3.0"
    }
  }
}
```

```
provider "azurerm" {
  features {}
}

resource "azurerm_resource_group" "rg" {
  name     = var.rg_name
  location = var.location
}

resource "azurerm_virtual_network" "vnet" {
  name                = var.vnet_name
  address_space       = ["10.20.0.0/16"]
  location            = var.location
  resource_group_name = var.rg_name
}

resource "azurerm_subnet" "subnet" {
  name                 = var.subnet_name
  resource_group_name  = var.rg_name
  virtual_network_name = azurerm_virtual_network.vnet.name
  address_prefixes     = ["10.20.1.0/24"]
}

resource "azurerm_network_security_group" "nsg" {
  name                = var.nsg_name
  location            = var.location
  resource_group_name = var.rg_name
}

resource "azurerm_public_ip" "pip" {
  name                = var.public_ip_name
  location            = var.location
  resource_group_name = var.rg_name
  allocation_method   = "Static"
}

resource "azurerm_network_interface" "nic" {
  name                = var.nic_name
  location            = var.location
  resource_group_name = var.rg_name

  ip_configuration {
    name                          = "internal"
    subnet_id                     = azurerm_subnet.subnet.id
    private_ip_address_allocation = "Dynamic"
    public_ip_address_id          = azurerm_public_ip.pip.id
  }
}

resource "azurerm_linux_virtual_machine" "vm" {
  name                = var.vm_name
  resource_group_name = var.rg_name
  location            = var.location
  size                = var.vm_size
  admin_username      = var.admin_username

  network_interface_ids = [
    azurerm_network_interface.nic.id
  ]

  admin_ssh_key {
    username   = var.admin_username
    public_key = file("~/.ssh/id_rsa.pub")
  }

  os_disk {
    caching              = "ReadWrite"
    storage_account_type = "Standard_LRS"
  }

  source_image_reference {
    publisher = "Canonical"
    offer     = "0001-com-ubuntu-server-focal"
    sku       = "20_04-lts"
    version   = "latest"
  }
}
```

Step 6: Initialize Terraform

```
terraform init
```

Step 7: Plan

```
terraform plan
```

Step 8: Apply

```
terraform apply
```

Type:

```
yes
```

Step 9: Verify VM

VM Name:

```
Sandeep-machine-tfvars
```

Step 10: Cleanup

```
terraform destroy
```

Type:

```
yes
```