Lab 12: Terraform Modules – Using Existing Module to Create a Virtual Machine in Azure

Author: Dr. Sandeep Kumar Sharma
Level: Intermediate
Platform: Ubuntu Linux + Microsoft Azure
Prerequisite: Lab 1 to Lab 11

Learning Objective

Participants will learn:

- What a Terraform module is
- Why modules are used
- Difference between root module and child module
- What is a public module
- How to use an existing (ready-made) Terraform module
- How to create Azure VM using a module

Learning Outcome

After completing this lab, participants will:

- Understand modular Terraform design
- Use community modules
- Build infrastructure faster
- Follow real production Terraform practices

Concept Explanation

What is a Terraform Module?

A Terraform module is a **reusable block of Terraform code**.

Instead of writing everything again and again, we reuse code.

Module = Ready-made infrastructure template

Types of Modules

1. Root Module → Your main project folder
2. Child Module → Reusable component folder
3. Public Module → Modules available on Terraform Registry

Why Modules are Used

- Reusability
- Standardization
- Clean structure
- Scalability
- Team collaboration
- Faster deployments
- Production design

Using Existing Public Module

We will use a **Terraform Registry Azure VM module**.

Source:

```
Azure/compute/azurerm
```

This is an official community-supported module.

Hands-On Lab

Step 1: Create New Folder

```
mkdir terraform-azure-module-lab
cd terraform-azure-module-lab
```

Step 2: Create Main File

```
touch main.tf
nano main.tf
```

Step 3: Provider Configuration

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

Step 4: Use Existing VM Module

```
module "linux_vm" {
  source  = "Azure/compute/azurerm"
  version = "~> 5.0"

  resource_group_name = "rg-module-lab"
  location            = "East US"

  vm_os_simple        = "UbuntuServer"
  vm_size             = "Standard_B1s"

  admin_username = "azureuser"

  ssh_public_keys = [
    {
      username   = "azureuser"
      public_key = file("~/.ssh/id_rsa.pub")
    }
  ]

  vm_name = "Sandeep-module-vm"
}
```

Step 5: Initialize Terraform

```
terraform init
```

Terraform will: - Download module - Download providers

Step 6: Plan

```
terraform plan
```

Step 7: Apply

```
terraform apply
```

Type:

```
yes
```

Step 8: Verify in Azure

- Resource Group: `rg-module-lab`
- Virtual Machine: `Sandeep-module-vm`

Understanding the Flow

```
main.tf
   ↓
module block
   ↓
Terraform Registry
   ↓
Module code
```

```
   ↓
Azure Infrastructure
```

Key Understanding

- We did not write VM code
- We reused module code
- Terraform handled everything
- Module abstracted complexity

Real Production Use

In real companies:

- VM modules
- Network modules
- Security modules
- Database modules
- Monitoring modules

Everything is modular.

Cleanup

```
terraform destroy
```

Type:

```
yes
```