Lab 15: Terraform Remote Backend – Azure Storage State Management

Author: Dr. Sandeep Kumar Sharma
Level: Intermediate to Advanced
Platform: Ubuntu Linux + Microsoft Azure
Prerequisite: Lab 1 to Lab 14

Learning Objective

Participants will learn:

- What a remote backend is
- Why remote state is required
- Problems with local state
- What state locking is
- How Terraform stores state remotely
- How teams collaborate using remote backend
- How to configure Azure Storage as Terraform backend

Learning Outcome

After completing this lab, participants will:

- Understand production Terraform architecture
- Use remote backend
- Manage shared state safely
- Enable team collaboration
- Prevent state corruption

Concept Explanation

What is Terraform Remote Backend?

A remote backend is a **central location** where Terraform stores the state file instead of local system.

Instead of:

```
local terraform.tfstate
```

Terraform stores state in:

```
Azure Storage Account
```

Why Remote Backend is Needed

Local state problems:

- No collaboration
- State overwrite risk
- No locking
- No backup
- No security
- No versioning
- No team access

Benefits of Remote Backend

- Centralized state
- Team collaboration
- State locking
- Versioning
- Backup
- Security
- Reliability
- Enterprise readiness

Architecture

```
Terraform CLI
     ↓
Azure Storage Account
     ↓
Blob Container
```

```
     ↓
terraform.tfstate
```

Hands-On Lab

Step 1: Create Resource Group for Backend

```
az group create --name rg-terraform-backend --location eastus
```

Step 2: Create Storage Account

```
az storage account create
  --name tfstatebackend12345
  --resource-group rg-terraform-backend
  --location eastus
  --sku Standard_LRS
```

Step 3: Create Blob Container

```
az storage container create
  --name tfstate
  --account-name tfstatebackend12345
```

Terraform Configuration

Step 4: Create Project Folder

```
mkdir terraform-remote-backend-lab
cd terraform-remote-backend-lab
```

Step 5: Create main.tf

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

Add:

```
terraform {
  backend "azurerm" {
    resource_group_name  = "rg-terraform-backend"
    storage_account_name = "tfstatebackend12345"
    container_name       = "tfstate"
    key                  = "lab15.tfstate"
  }
}

provider "azurerm" {
  features {}
}

resource "azurerm_resource_group" "rg" {
  name     = "rg-remote-backend-demo"
  location = "East US"
}
```

Step 6: Initialize Terraform

```
terraform init
```

Terraform will: - Configure backend - Migrate state - Connect to Azure Storage

---

Step 7: Plan

```
terraform plan
```

Step 8: Apply

```
terraform apply
```

Type:

```
yes
```

Verification

Azure Portal:

- Storage Account → Containers → tfstate
- File: `lab15.tfstate`

State Locking

Azure backend supports:

- Automatic state locking
- Concurrent operation protection
- Safe team operations

Team Collaboration Model

```
Engineer A  → Terraform Apply
Engineer B  → Terraform Plan
Engineer C  → Terraform Apply

Single state file
Central backend
No conflicts
```

Cleanup

```
terraform destroy
```

Type:

```
yes
```

(Optional) Delete backend resources:

```
az group delete --name rg-terraform-backend
```