## Step-01: Introduction

- Understand Resource Syntax
- Understanding Terraform State File
  - terraform.tfstate
- Understanding Desired and Current States

## Step-02: Understand Resource Syntax

- Resource Block
- Resource Type
- Resource Local Name
- Resource Arguments
- Resource Meta-Arguments

## Step-03: c1-versions.tf

```
# Terraform Block
terraform {
  required_version = ">= 1.0.0"
  required_providers {
    azurerm = {
      source = "hashicorp/azurerm"
      version = ">= 2.0"
    }
  }
}

# Provider Block
provider "azurerm" {
 features {}
}
```

## Step-04: c2-resource-group.tf

```
# Resource-1: Azure Resource Group
resource "azurerm_resource_group" "myrg" {
  name = "myrg-1"
  location = "East US"
}
```

## Step-05: c3-virtual-network.tf

1. Resource-2: Create Virtual Network
2. Resource-3: Create Subnet
3. Resource-4: Create Public IP Address
4. Resource-5: Create Network Interface

```
# Resource-2: Create Virtual Network
resource "azurerm_virtual_network" "myvnet" {
  name                = "myvnet-1"
  address_space       = ["10.0.0.0/16"]
  location            = azurerm_resource_group.myrg.location
  resource_group_name = azurerm_resource_group.myrg.name
```

```
}

# Resource-3: Create Subnet
resource "azurerm_subnet" "mysubnet" {
  name                 = "mysubnet-1"
  resource_group_name  = azurerm_resource_group.myrg.name
  virtual_network_name = azurerm_virtual_network.myvnet.name
  address_prefixes     = ["10.0.2.0/24"]
}

# Resource-4: Create Public IP Address
resource "azurerm_public_ip" "mypublicip" {
  name                = "mypublicip-1"
  resource_group_name = azurerm_resource_group.myrg.name
  location            = azurerm_resource_group.myrg.location
  allocation_method   = "Static"
  tags = {
    environment = "Dev"
  }
}

# Resource-5: Create Network Interface
resource "azurerm_network_interface" "myvm1nic" {
  name                = "vm1-nic"
  location            = azurerm_resource_group.myrg.location
  resource_group_name = azurerm_resource_group.myrg.name

  ip_configuration {
    name                          = "internal"
    subnet_id                     = azurerm_subnet.mysubnet.id
    private_ip_address_allocation = "Dynamic"
    public_ip_address_id = azurerm_public_ip.mypublicip.id
  }
}
```

## Step-06: Understand Resource Behaviour

- We are going to understand resource behavior in combination with Terraform State

## Step-07: Resource: Create Resources

```
# Initialize Terraform
terraform init

Observation:
1) Successfully downloaded providers in .terraform folder
2) Created lock file named ".terraform.lock.hcl"

# Validate Terraform configuration files
terraform validate
Observation: No files changed / added in current working directory

# Format Terraform configuration files
terraform fmt
Observations: *.tf files will change to format them if any format changes exists

# Review the terraform plan
terraform plan
Observation-1: Nothing happens during the first run from terraform state perspective
Observation-2: From Resource Behaviour perspective you can see "+ create", we are creating

# Create Resources
terraform apply -auto-approve
Observation:
1) Creates terraform.tfstate file in local working directory
2) Creates actual resource in Azure Cloud
```

## Step-08: Understanding Terraform State File

- What is Terraform State ?

1. It is the primary core thing for terraform to function
2. In a short way, its the underlying database containing the resources information which are provisioning using Terraform
3. **Primary Purpose:** To store bindings between objects in a remote system and resource instances declared in your configuration.
4. When Terraform creates a remote object in response to a change of configuration, it will record the identity of that remote object against a particular resource instance, and then potentially update or delete that object in response to future configuration changes.
5. Terraform state file created when we first run the `terraform apply`
6. Terraform state file is created locally in working directory.
7. If required, we can confiure the `backend block` in `terraform block` which will allow us to store state file remotely. Storing remotely is recommended option.

## Step-09: Review terraform.tfstate file

- Terraform State files are JSON based
- Manual editing of Terraform state files is highly not recommended
- Review `terraform.tfstate` file step by step

## Step-10: Resource: Update In-Place: Make changes by adding new tag to Virtual Network Resource

- Add a new tag in `azurerm_virtual_network` resource

```
# Add this for Virtual Network Resource
    "Environment" = "Dev"
```

- **Review Terraform Plan**

```
# Review the terraform plan
terraform plan
Observation: You should see "~ update in-place"
"Plan: 0 to add, 1 to change, 0 to destroy."

# Create / Update Resources
terraform apply -auto-approve
Observation: "Apply complete! Resources: 0 added, 1 changed, 0 destroyed."
```

## Step-11: Resource: Destroy and Re-create Resources: Update Virtual Network Name

- This will destroy the Virtual Network, Subnet and Recreate them.

```
# Before
  name                = "vm1-nic"

# After
  name                = "vm1-nic1"
```

- Execute Terraform Commands

```
# Review the terraform plan
terraform plan
Observation:
1)   -/+ destroy and then create replacement
2) -/+ resource "azurerm_network_interface" "myvm1nic" {
3) -/+ resource "azurerm_network_interface" "myvm1nic" {
4) Plan: 2 to add, 0 to change, 2 to destroy.

# Create / Update Resources
terraform apply -auto-approve
Observation:
1. Apply complete! Resources: 2 added, 0 changed, 2 destroyed.
```

## Step-12: Resource: Destroy Resource

```
# Destroy Resource
terraform destroy

Observation:
1) - destroy
2) All 7 resources will be destroyed
3) Plan: 0 to add, 0 to change, 7 to destroy.
4) Destroy complete! Resources: 7 destroyed.
```

## Step-13: Understand Desired and Current States

- **Desired State:** Local Terraform Manifest (All *.tf files)
- **Current State:** Real Resources present in your cloud

## Step-14: Clean-Up

```
# Destroy Resource
terraform destroy -auto-approve

# Remove Terraform Files
rm -rf .terraform*
rm -rf terraform.tfstate*
```

## Step-15: Revert files to Demo State

```
# Change-1: Comment in azurerm_virtual_network
#"Environment" = "Dev"  # Uncomment during Step-10

# Change-2: Revert name back in azurerm_network_interface Resource
name                = "vm1-nic"
```

## References

- Terraform State
- Manipulating Terraform State