**Lab 11: Deploying a Linux VM on Azure using Terraform**

**Lab Objective**

By the end of this lab, you will:

* Understand Terraform configuration for Azure resources.
* Deploy a Linux Virtual Machine (VM) with networking in Azure.
* Learn Terraform workflow: init, plan, apply.
* Gain hands-on experience with core Azure IaC concepts.

**Pre-requisites**

Before starting, ensure you have:

1. **Azure Account**
   * Active Azure subscription.
   * Sufficient privileges (Contributor role).
2. **Terraform Installed**
   * Terraform v1.0 or later.
   * Verify installation:
   * terraform -v
3. **Azure CLI Installed**
   * Verify installation:
   * az --version
4. **Code Editor**  
   VS Code with Terraform extension recommended.

**Lab Setup**

**Step 1 — Login to Azure**

az login

* This opens a browser for authentication.
* Confirm the active subscription:

az account show

**Step 2 — Directory Structure**

Create a folder for Terraform configuration:

mkdir terraform-vm-lab

cd terraform-vm-lab

**Step 3 — Create Terraform Configuration File**

Create vm.tf with the following content:

provider "azurerm" {

features {}

}

resource "azurerm\_resource\_group" "rg" {

name = "raman-tf-resources"

location = "Australia East"

}

resource "azurerm\_public\_ip" "example" {

name = "acceptanceTestPublicIp1"

resource\_group\_name = azurerm\_resource\_group.rg.name

location = azurerm\_resource\_group.rg.location

allocation\_method = "Static"

tags = {

environment = "Production"

}

}

resource "azurerm\_virtual\_network" "network" {

name = "raman-network"

address\_space = ["10.0.0.0/16"]

location = azurerm\_resource\_group.rg.location

resource\_group\_name = azurerm\_resource\_group.rg.name

}

resource "azurerm\_subnet" "subnet" {

name = "raman-subnet"

resource\_group\_name = azurerm\_resource\_group.rg.name

virtual\_network\_name = azurerm\_virtual\_network.network.name

address\_prefixes = ["10.0.2.0/24"]

}

resource "azurerm\_network\_interface" "nic" {

name = "raman-nic"

location = azurerm\_resource\_group.rg.location

resource\_group\_name = azurerm\_resource\_group.rg.name

ip\_configuration {

name = "internal"

subnet\_id = azurerm\_subnet.subnet.id

public\_ip\_address\_id = azurerm\_public\_ip.example.id

private\_ip\_address\_allocation = "Dynamic"

}

}

resource "azurerm\_linux\_virtual\_machine" "vm" {

name = "raman-machine"

resource\_group\_name = azurerm\_resource\_group.rg.name

location = azurerm\_resource\_group.rg.location

size = "Standard\_F2"

disable\_password\_authentication = false

admin\_username = "adminuser"

admin\_password = "Rmankhn@2023"

network\_interface\_ids = [

azurerm\_network\_interface.nic.id,

]

os\_disk {

caching = "ReadWrite"

storage\_account\_type = "Standard\_LRS"

}

source\_image\_reference {

publisher = "Canonical"

offer = "UbuntuServer"

sku = "16.04-LTS"

version = "latest"

}

}

**Step 4 — Understanding the Configuration**

**Key Blocks**

1. **Provider**
   * Configures Terraform to use Azure.
   * features {} is mandatory even if empty.
2. **Resource Group**
   * Logical container for all Azure resources.
   * Location defines region of deployment.
3. **Public IP**
   * Assigns a static IP for external VM access.
4. **Virtual Network**
   * Private network for resources.
   * Address space 10.0.0.0/16 allows multiple subnets.
5. **Subnet**
   * A segment of the VNet.
   * Address prefix 10.0.2.0/24 for VM NIC allocation.
6. **Network Interface**
   * Attaches VM to network and public IP.
   * Dynamic private IP allocation.
7. **Linux VM**
   * VM size: Standard\_F2 (2 vCPU, 4 GB RAM).
   * Password authentication enabled.
   * OS: Ubuntu 16.04-LTS.

**Dependency Graph**

Terraform creates resources in dependency order:

Resource Group → Public IP → Virtual Network → Subnet → NIC → VM

**Step 5 — Terraform Workflow**

**1. Initialize Terraform**

terraform init

* Downloads Azure provider plugins.
* Prepares environment.

**Output Example:**

Terraform has been successfully initialized!

**2. Plan Infrastructure**

terraform plan

* Shows the execution plan (what Terraform will create/change).
* Validates configuration.

**Example Output:**

Plan: 6 to add, 0 to change, 0 to destroy.

**3. Apply Configuration**

terraform apply

* Deploys resources.
* Terraform will prompt:

Do you want to perform these actions? (yes/no):

Type yes.

**4. Verify Deployment**

Check resources in Azure Portal or CLI.

Example CLI commands:

az group show --name raman-tf-resources

az vm show --name raman-machine --resource-group raman-tf-resources

az network public-ip show --name acceptanceTestPublicIp1 --resource-group raman-tf-resources

**Step 6 — Cleaning Up**

Destroy resources to avoid costs:

terraform destroy

Confirm with yes.

**Expected Outcomes**

After completion, learners should:

* Have a running Ubuntu VM in Azure.
* Understand how Terraform manages Azure resources.
* Know how to write and modify Terraform configurations.
* Understand the Terraform lifecycle (init, plan, apply, destroy).

**Lab 12: Associating a Network Security Group (NSG) to a Subnet using Terraform**

**Lab Objective**

By the end of this lab, you will:

* Understand what a Network Security Group (NSG) is.
* Create an NSG in Azure using Terraform.
* Define security rules in the NSG.
* Associate an NSG to a subnet to control traffic.
* Learn Terraform workflow for security group deployment.

**Pre-requisites**

Before starting:

* Azure subscription with **Contributor** role or higher.
* Terraform installed (v1.0+).
* Basic Azure networking knowledge.
* Existing Terraform setup with:
  + Resource Group.
  + Virtual Network (VNet).
  + Subnet.
* Terraform project directory ready (e.g., from your VM creation lab).

**Step 1 — Understand the Configuration**

Let’s break down your provided Terraform blocks.

**a) Network Security Group (NSG)**

resource "azurerm\_network\_security\_group" "example" {

name = "acceptanceTestSecurityGroup1"

location = azurerm\_resource\_group.example.location

resource\_group\_name = azurerm\_resource\_group.example.name

security\_rule {

name = "test123"

priority = 100

direction = "Inbound"

access = "Allow"

protocol = "Tcp"

source\_port\_range = "\*"

destination\_port\_range = "\*"

source\_address\_prefix = "\*"

destination\_address\_prefix = "\*"

}

tags = {

environment = "Production"

}

}

**Concepts:**

* **NSG:** A security container to control inbound and outbound traffic to Azure resources.
* **location:** Must match the resource group’s location.
* **security\_rule block:** Defines traffic rules.
  + name: Unique name for the rule.
  + priority: Determines rule order (lower numbers are processed first).
  + direction: Inbound/Outbound traffic direction.
  + access: Allow/Deny traffic.
  + protocol: TCP, UDP, or \* for all.
  + source\_port\_range / destination\_port\_range: Which ports are allowed/denied.
  + source\_address\_prefix / destination\_address\_prefix: Source and destination IP ranges.
* **tags:** Optional metadata for organization.

**b) NSG Association to a Subnet**

resource "azurerm\_subnet\_network\_security\_group\_association" "associate" {

subnet\_id = azurerm\_subnet.subnet.id

network\_security\_group\_id = azurerm\_network\_security\_group.sg.id

}

**Concepts:**

* Associates NSG with a subnet.
* **subnet\_id:** The subnet to secure.
* **network\_security\_group\_id:** The NSG to apply.
* Association enforces the NSG’s rules on all resources in the subnet.

💡 This means all VM NICs inside this subnet will inherit NSG rules unless overridden at NIC level.

**Step 2 — Add Configuration to Terraform Project**

In your Terraform project directory:

1. Create a new file:  
   nsg.tf
2. Add the above NSG and association definitions.

**Note:**  
Replace incorrect references like azurerm\_resource\_group.example.name with your actual resource group reference, e.g.:

resource\_group\_name = azurerm\_resource\_group.rg.name

location = azurerm\_resource\_group.rg.location

Updated example:

resource "azurerm\_network\_security\_group" "sg" {

name = "acceptanceTestSecurityGroup1"

location = azurerm\_resource\_group.rg.location

resource\_group\_name = azurerm\_resource\_group.rg.name

security\_rule {

name = "AllowAllInbound"

priority = 100

direction = "Inbound"

access = "Allow"

protocol = "\*"

source\_port\_range = "\*"

destination\_port\_range = "\*"

source\_address\_prefix = "\*"

destination\_address\_prefix = "\*"

}

tags = {

environment = "Production"

}

}

resource "azurerm\_subnet\_network\_security\_group\_association" "associate" {

subnet\_id = azurerm\_subnet.subnet.id

network\_security\_group\_id = azurerm\_network\_security\_group.sg.id

}

**Step 3 — Terraform Workflow**

**1. Initialize Terraform**

terraform init

* Downloads Azure provider plugins.

**Expected output:**

Terraform has been successfully initialized!

**2. Plan Deployment**

terraform plan

* Shows what Terraform will create and changes required.

**Example output:**

Plan: 2 to add, 0 to change, 0 to destroy.

**3. Apply Configuration**

terraform apply

* Confirms changes and provisions NSG and association.
* Type yes when prompted.

**Expected output:**  
Terraform will show the resources being created and their IDs.

**Step 4 — Verification**

**Check NSG**

az network nsg show \

--name acceptanceTestSecurityGroup1 \

--resource-group raman-tf-resources

**Check NSG association**

az network vnet subnet show \

--name raman-subnet \

--resource-group raman-tf-resources \

--vnet-name raman-network

Look for "networkSecurityGroup": {...} in the JSON output.

**Step 5 — Testing the Rule**

* Deploy a VM inside this subnet (already done in previous lab).
* Try connecting to the VM according to NSG rules.
* Since the NSG rule allows all inbound TCP traffic, your VM should be accessible on all ports.

**Step 6 — Clean-up**

terraform destroy

* Removes NSG and subnet association.
* Confirms with yes.

**Conceptual Diagram**

Here’s how this setup works:

[ NSG ] --> [ Subnet ]

|

[ VM NIC ]

|

[ VM ]

* NSG applies to subnet → all resources inside inherit the rule.
* You could also apply NSG to individual NICs for finer control.

**Step 7 — Best Practices**

* Use **variables** for NSG name, location, and rules instead of hardcoding.
* Create multiple security rules for finer traffic control.
* Use descriptive names and tags for resource organization.
* Manage NSG rules separately for better scalability.

**Example Variables for NSG**

variables.tf

variable "nsg\_name" {

default = "acceptanceTestSecurityGroup1"

}

variable "nsg\_location" {

default = "Australia East"

}

**Example Outputs**

outputs.tf

output "nsg\_id" {

value = azurerm\_network\_security\_group.sg.id

}

**Lab 13: Deploying a Linux VM with NSG and Output Values using Terraform**

**Lab Objective**

By the end of this lab, learners will:

* Understand Terraform resource blocks for NSG, subnet, VM, NIC, and public IP.
* Deploy a Linux VM with an associated Network Security Group (NSG).
* Retrieve output values from Terraform.
* Learn Terraform lifecycle: init, plan, apply.
* Validate the deployment in Azure.

**Pre-requisites**

Before starting:

* Azure subscription (Contributor role).
* Terraform v1.0+ installed.
* Azure CLI installed.
* Basic Azure networking knowledge.
* Existing SSH keypair (optional for SSH login).
* A working directory for Terraform files.

**Step 1 — Understand the Terraform Configuration**

Let’s break your vm.tf file into conceptual blocks.

**a) Provider Block**

provider "azurerm" {

features {}

}

* Tells Terraform to use Azure Resource Manager (AzureRM) as the provider.
* features {} is required for Azure provider configuration.

**b) Resource Group**

resource "azurerm\_resource\_group" "rg" {

name = "raman-tf-resources"

location = "Australia East"

}

* Creates an Azure resource group to contain related resources.
* location determines the Azure region for resource deployment.

**c) Virtual Network**

resource "azurerm\_virtual\_network" "network" {

name = "raman-network"

address\_space = ["10.0.0.0/16"]

location = azurerm\_resource\_group.rg.location

resource\_group\_name = azurerm\_resource\_group.rg.name

}

* Creates a Virtual Network (VNet) in Azure.
* address\_space defines network range.

**d) Subnet**

resource "azurerm\_subnet" "subnet" {

name = "raman-subnet"

resource\_group\_name = azurerm\_resource\_group.rg.name

virtual\_network\_name = azurerm\_virtual\_network.network.name

address\_prefixes = ["10.0.2.0/24"]

}

* Creates a subnet inside the VNet.
* address\_prefixes defines subnet IP range.

**e) Public IP**

resource "azurerm\_public\_ip" "pubIP" {

name = "PublicIp"

resource\_group\_name = azurerm\_resource\_group.rg.name

location = azurerm\_resource\_group.rg.location

allocation\_method = "Static"

tags = {

environment = "Production"

}

}

* Creates a static public IP for external access to VM.
* Tags help organize resources.

**f) Network Interface (NIC)**

resource "azurerm\_network\_interface" "nic" {

name = "raman-nic"

location = azurerm\_resource\_group.rg.location

resource\_group\_name = azurerm\_resource\_group.rg.name

ip\_configuration {

name = "internal"

subnet\_id = azurerm\_subnet.subnet.id

private\_ip\_address\_allocation = "Dynamic"

public\_ip\_address\_id = azurerm\_public\_ip.pubIP.id

}

}

* Creates NIC for VM with subnet and public IP binding.
* Private IP is assigned dynamically from subnet.

**g) Linux Virtual Machine**

resource "azurerm\_linux\_virtual\_machine" "vm" {

name = "raman-machine"

resource\_group\_name = azurerm\_resource\_group.rg.name

location = azurerm\_resource\_group.rg.location

size = "Standard\_F2"

disable\_password\_authentication = false

admin\_username = "adminuser"

admin\_password = "Rmankhn@2023"

network\_interface\_ids = [

azurerm\_network\_interface.nic.id,

]

os\_disk {

caching = "ReadWrite"

storage\_account\_type = "Standard\_LRS"

}

source\_image\_reference {

publisher = "Canonical"

offer = "UbuntuServer"

sku = "16.04-LTS"

version = "latest"

}

}

* Deploys an Ubuntu VM.
* Uses NIC created earlier.
* Allows password authentication.
* Configures OS disk storage.

**h) Network Security Group (NSG)**

resource "azurerm\_network\_security\_group" "sg" {

name = "raman-sg"

location = azurerm\_resource\_group.rg.location

resource\_group\_name = azurerm\_resource\_group.rg.name

security\_rule {

name = "test123"

priority = 100

direction = "Inbound"

access = "Allow"

protocol = "Tcp"

source\_port\_range = "80"

destination\_port\_range = "80"

source\_address\_prefix = "\*"

destination\_address\_prefix = "\*"

}

tags = {

environment = "Production"

}

}

* Creates NSG to allow inbound TCP traffic on port 80 (HTTP).
* Rules control network traffic at subnet level.

**i) NSG Association**

resource "azurerm\_subnet\_network\_security\_group\_association" "associate" {

subnet\_id = azurerm\_subnet.subnet.id

network\_security\_group\_id = azurerm\_network\_security\_group.sg.id

}

* Associates NSG with subnet.
* Applies NSG rules to all VMs in subnet.

**j) Outputs**

output "raman-subnetCIDR" {

value = [azurerm\_subnet.subnet.address\_prefix]

}

output "raman-rgName" {

value = [azurerm\_resource\_group.rg.name]

}

output "raman-pubIp" {

value = [azurerm\_public\_ip.pubIP.ip\_address]

}

output "raman-pvtIP" {

value = [azurerm\_linux\_virtual\_machine.vm.private\_ip\_address]

}

output "raman-sgRule-details" {

value = [azurerm\_network\_security\_group.sg.security\_rule]

}

* Outputs display key resource values after deployment.
* Examples:
  + Subnet CIDR range.
  + Resource group name.
  + VM public IP.
  + VM private IP.
  + NSG security rule details.

**Step 2 — Lab Execution**

**1. Initialize Terraform**

terraform init

* Downloads required provider plugins.

**2. Review Execution Plan**

terraform plan

* Validates configuration.
* Shows resources to be created.

**3. Apply Configuration**

terraform apply

* Deploys all resources.
* Confirm with yes when prompted.

**4. Review Outputs**

After apply, Terraform will display outputs:

Outputs:

raman-subnetCIDR = ["10.0.2.0/24"]

raman-rgName = ["raman-tf-resources"]

raman-pubIp = ["<public-ip-address>"]

raman-pvtIP = ["10.0.2.4"]

raman-sgRule-details = [

{

name = "test123"

priority = 100

direction = "Inbound"

access = "Allow"

protocol = "Tcp"

source\_port\_range = "80"

destination\_port\_range = "80"

source\_address\_prefix = "\*"

destination\_address\_prefix = "\*"

}

]

**Step 3 — Verification in Azure**

**Verify Resources via Azure CLI**

az group show --name raman-tf-resources

az network nsg show --name raman-sg --resource-group raman-tf-resources

az network public-ip show --name PublicIp --resource-group raman-tf-resources

az vm show --name raman-machine --resource-group raman-tf-resources

**Verify NSG Association**

az network vnet subnet show \

--name raman-subnet \

--resource-group raman-tf-resources \

--vnet-name raman-network

Look for "networkSecurityGroup" in JSON output.

**Step 4 — Clean Up**

terraform destroy

* Removes all resources.
* Confirm with yes.

**Step 5 — Best Practices**

* Use variables instead of hardcoding values.
* Store sensitive data in Azure Key Vault or Terraform variables.
* Use output values for automation and integration.
* Use modules for reusable Terraform configurations.
* Enable remote state storage for team collaboration.

**Step 6 — Lab Diagram**

Your setup looks like this conceptually:

Azure Resource Group

├── Virtual Network (10.0.0.0/16)

│ ├── Subnet (10.0.2.0/24) ← NSG applied

│ │ └── VM NIC (Public IP + Private IP)

│ └── NSG (Inbound TCP on port 80)

└── Linux VM

**Lab 14: Terraform Variables for Azure Infrastructure Deployment**

**Lab Objective**

By the end of this lab, you will:

* Understand Terraform variables and their usage.
* Learn how to parameterize Terraform configurations for reusability.
* Deploy a Linux VM in Azure using variables.
* Create outputs to retrieve deployed resource information.
* Understand the Terraform workflow: init, plan, apply.

**Pre-requisites**

Before starting this lab, ensure:

* Azure subscription with Contributor access.
* Terraform v1.0+ installed.
* Azure CLI installed.
* Access to an SSH client for VM connection.
* Basic Azure networking knowledge.

**Step 1 — Understanding Terraform Variables**

In Terraform, **variables** allow you to parameterize your configuration so it can be reused without hardcoding values.

**Benefits of using variables:**

* Reusability: Same code works for multiple environments.
* Maintainability: Change the variable instead of changing multiple places in code.
* Flexibility: Different values for dev, test, and prod.

**Your variables.tf**

variable "location" {

default = "West US"

}

variable "rgname" {

default = "raman-rg"

}

variable "saname" {

default = "ramanstorage"

}

**Explanation:**

* location: Azure region where resources will be deployed.
* rgname: Name of Azure Resource Group.
* saname: Storage account name (not used in vm.tf here but defined for expansion).

**Step 2 — Understanding vm.tf**

**Key Changes Due to Variables:**

In vm.tf, references to hardcoded resource group name and location are replaced with variables:

name = var.rgname

location = var.location

This means the actual values are provided by:

* **Defaults** in variable.tf.
* OR **terraform.tfvars** file.
* OR CLI -var argument.
* OR environment variables.

**Breakdown of vm.tf**

**1. Provider**

provider "azurerm" {

features {}

}

Sets Azure provider for Terraform.

**2. Resource Group**

resource "azurerm\_resource\_group" "rg" {

name = var.rgname

location = var.location

}

Creates a resource group with variable values for name and location.

**3. Virtual Network**

resource "azurerm\_virtual\_network" "network" {

name = "raman-network"

address\_space = ["10.0.0.0/16"]

location = azurerm\_resource\_group.rg.location

resource\_group\_name = azurerm\_resource\_group.rg.name

}

Creates a VNet inside the resource group.

**4. Subnet**

resource "azurerm\_subnet" "subnet" {

name = "raman-subnet"

resource\_group\_name = azurerm\_resource\_group.rg.name

virtual\_network\_name = azurerm\_virtual\_network.network.name

address\_prefixes = ["10.0.2.0/24"]

}

Creates a subnet inside the VNet.

**5. Public IP**

resource "azurerm\_public\_ip" "pubIP" {

name = "PublicIp"

resource\_group\_name = azurerm\_resource\_group.rg.name

location = azurerm\_resource\_group.rg.location

allocation\_method = "Static"

tags = {

environment = "Production"

}

}

Creates a static public IP for VM access.

**6. Network Interface**

resource "azurerm\_network\_interface" "nic" {

name = "raman-nic"

location = azurerm\_resource\_group.rg.location

resource\_group\_name = azurerm\_resource\_group.rg.name

ip\_configuration {

name = "internal"

subnet\_id = azurerm\_subnet.subnet.id

private\_ip\_address\_allocation = "Dynamic"

public\_ip\_address\_id = azurerm\_public\_ip.pubIP.id

}

}

Creates NIC attached to public IP and subnet.

**7. Linux VM**

resource "azurerm\_linux\_virtual\_machine" "vm" {

name = "raman-machine"

resource\_group\_name = azurerm\_resource\_group.rg.name

location = azurerm\_resource\_group.rg.location

size = "Standard\_F2"

disable\_password\_authentication = false

admin\_username = "adminuser"

admin\_password = "Rmankhn@2023"

network\_interface\_ids = [azurerm\_network\_interface.nic.id]

os\_disk {

caching = "ReadWrite"

storage\_account\_type = "Standard\_LRS"

}

source\_image\_reference {

publisher = "Canonical"

offer = "UbuntuServer"

sku = "16.04-LTS"

version = "latest"

}

}

Deploys Ubuntu VM with variable-driven resource group and location.

**8. NSG and Association**

resource "azurerm\_network\_security\_group" "sg" {

name = "raman-sg"

location = azurerm\_resource\_group.rg.location

resource\_group\_name = azurerm\_resource\_group.rg.name

security\_rule {

name = "test123"

priority = 100

direction = "Inbound"

access = "Allow"

protocol = "Tcp"

source\_port\_range = "80"

destination\_port\_range = "80"

source\_address\_prefix = "\*"

destination\_address\_prefix = "\*"

}

tags = {

environment = "Production"

}

}

resource "azurerm\_subnet\_network\_security\_group\_association" "associate" {

subnet\_id = azurerm\_subnet.subnet.id

network\_security\_group\_id = azurerm\_network\_security\_group.sg.id

}

Creates an NSG allowing inbound HTTP traffic on port 80 and associates it to subnet.

**9. Output Values**

output "raman-subnetCIDR" {

value = [azurerm\_subnet.subnet.address\_prefix]

}

output "raman-rgName" {

value = [azurerm\_resource\_group.rg.name]

}

output "raman-pubIp" {

value = [azurerm\_public\_ip.pubIP.ip\_address]

}

output "raman-pvtIP" {

value = [azurerm\_linux\_virtual\_machine.vm.private\_ip\_address]

}

output "raman-sgRule-details" {

value = [azurerm\_network\_security\_group.sg.security\_rule]

}

Outputs critical information for use in automation or post-deployment validation.

**Step 3 — Execution Workflow**

**1. Initialize Terraform**

terraform init

Initializes the project, downloads providers.

**2. Review Plan**

terraform plan

Shows resources to be created and confirms variable values.

**3. Apply Deployment**

terraform apply

Deploys resources. Confirm with yes.

**4. Review Outputs**

Terraform will display:

raman-subnetCIDR = ["10.0.2.0/24"]

raman-rgName = ["raman-rg"]

raman-pubIp = ["<public-ip-address>"]

raman-pvtIP = ["10.0.2.x"]

raman-sgRule-details = [

{

name = "test123"

...

}

]

**Step 4 — Verification**

**Verify Resource Group**

az group show --name raman-rg

**Verify VM**

az vm show --name raman-machine --resource-group raman-rg

**Verify NSG**

az network nsg show --name raman-sg --resource-group raman-rg

**Check NSG Association**

az network vnet subnet show --name raman-subnet --resource-group raman-rg --vnet-name raman-network

**Step 5 — Clean Up**

terraform destroy

Confirms with yes. Cleans all resources.

**Step 6 — Best Practices**

* Use terraform.tfvars or environment variables for different environments.
* Use descriptive variable names.
* Avoid hardcoding sensitive values like passwords — use Azure Key Vault or terraform.tfvars secured files.
* Modularize Terraform code for reusability.
* Enable Terraform state locking when collaborating.

**Step 7 — Diagram of Deployment**

Resource Group: var.rgname (location: var.location)

├── Virtual Network (10.0.0.0/16)

│ └── Subnet (10.0.2.0/24)

│ ├── Network Interface (NIC) → Public IP

│ └── NSG (Inbound TCP 80)

│ └── NSG Association to Subnet

└── Linux VM (Ubuntu 16.04-LTS)

**Lab 15 — Using Terraform in Azure DevOps**

**Step 1 — Prepare Terraform Code**

**File: variables.tf**

variable "rg\_name" {

  description = "Name of the existing Resource Group"

  type        = string

  default     = "raman"

}

variable "location" {

  description = "Location of the Resource Group"

  type        = string

  default     = "Australia East"

}

variable "vm\_admin\_username" {

  description = "Admin username for the Linux VM"

  type        = string

  default     = "adminuser"

}

variable "vm\_admin\_password" {

  description = "Admin password for the Linux VM"

  type        = string

  default     = "Rmankhn@2023"

}

variable "vm\_size" {

  description = "Size of the VM"

  type        = string

  default     = "Standard\_F2"

}

**File: vm.tf**

provider "azurerm" {

  features {}

}

# Reference existing Resource Group

data "azurerm\_resource\_group" "existing" {

  name = var.rg\_name

}

resource "azurerm\_virtual\_network" "network" {

  name                 = "raman-network"

  address\_space        = ["10.0.0.0/16"]

  location             = data.azurerm\_resource\_group.existing.location

  resource\_group\_name  = data.azurerm\_resource\_group.existing.name

}

resource "azurerm\_subnet" "subnet" {

  name                  = "raman-subnet"

  resource\_group\_name   = data.azurerm\_resource\_group.existing.name

  virtual\_network\_name  = azurerm\_virtual\_network.network.name

  address\_prefixes      = ["10.0.2.0/24"]

}

resource "azurerm\_public\_ip" "pubIP" {

  name                 = "PublicIp"

  resource\_group\_name = data.azurerm\_resource\_group.existing.name

  location             = data.azurerm\_resource\_group.existing.location

  allocation\_method    = "Static"

  tags = {

    environment = "Production"

  }

}

resource "azurerm\_network\_interface" "nic" {

  name                  = "raman-nic"

  location              = data.azurerm\_resource\_group.existing.location

  resource\_group\_name  = data.azurerm\_resource\_group.existing.name

  ip\_configuration {

    name                          = "internal"

    subnet\_id                     = azurerm\_subnet.subnet.id

    private\_ip\_address\_allocation = "Dynamic"

    public\_ip\_address\_id          = azurerm\_public\_ip.pubIP.id

  }

}

resource "azurerm\_linux\_virtual\_machine" "vm" {

  name                  = "raman-machine"

  resource\_group\_name   = data.azurerm\_resource\_group.existing.name

  location              = data.azurerm\_resource\_group.existing.location

  size                  = var.vm\_size

  admin\_username        = var.vm\_admin\_username

  admin\_password        = var.vm\_admin\_password

  disable\_password\_authentication = false

  network\_interface\_ids = [azurerm\_network\_interface.nic.id]

  os\_disk {

    caching              = "ReadWrite"

    storage\_account\_type = "Standard\_LRS"

  }

  source\_image\_reference {

    publisher = "Canonical"

    offer     = "UbuntuServer"

    sku       = "16.04-LTS"

    version   = "latest"

  }

}

resource "azurerm\_network\_security\_group" "sg" {

  name                 = "raman-sg"

  location             = data.azurerm\_resource\_group.existing.location

  resource\_group\_name = data.azurerm\_resource\_group.existing.name

  security\_rule {

    name                       = "test123"

    priority                   = 100

    direction                  = "Inbound"

    access                     = "Allow"

    protocol                   = "Tcp"

    source\_port\_range          = "80"

    destination\_port\_range     = "80"

    source\_address\_prefix      = "\*"

    destination\_address\_prefix = "\*"

  }

  tags = {

    environment = "Production"

  }

}

resource "azurerm\_subnet\_network\_security\_group\_association" "associate" {

  subnet\_id                  = azurerm\_subnet.subnet.id

  network\_security\_group\_id  = azurerm\_network\_security\_group.sg.id

}

# OUTPUT VALUES — FIXED

output "raman-subnetCIDR" {

  value = azurerm\_subnet.subnet.address\_prefixes[0]  # fixed property

}

output "raman-rgName" {

  value = data.azurerm\_resource\_group.existing.name

}

output "raman-pubIp" {

  value = azurerm\_public\_ip.pubIP.ip\_address

}

output "raman-pvtIP" {

  value = azurerm\_linux\_virtual\_machine.vm.private\_ip\_address

}

output "raman-sgRule-details" {

  value = azurerm\_network\_security\_group.sg.security\_rule

}

**Step 2 — Azure DevOps Pipeline YAML**

**File: azure-pipelines.yml**

trigger:

- main

pool:

  vmImage: 'ubuntu-latest'

variables:

  TF\_VERSION: '1.5.7'

  LOCATION: 'Australia East'

  RG\_NAME: 'raman'

  VM\_ADMIN\_USERNAME: 'adminuser'

  VM\_ADMIN\_PASSWORD: 'Rmankhn@2023'

  VM\_SIZE: 'Standard\_F2'

steps:

# Install Terraform

- script: |

    echo "Installing Terraform $(TF\_VERSION)..."

    wget https://releases.hashicorp.com/terraform/$(TF\_VERSION)/terraform\_$(TF\_VERSION)\_linux\_amd64.zip

    unzip terraform\_$(TF\_VERSION)\_linux\_amd64.zip

    sudo mv terraform /usr/local/bin/

    terraform -version

  displayName: 'Install Terraform'

# Azure Login

- task: AzureCLI@2

  inputs:

    #azureSubscription: 'Terraform-Azure-Connection'

    azureSubscription: 'tf'

    scriptType: 'bash'

    scriptLocation: 'inlineScript'

    inlineScript: |

      echo "Azure CLI login successful."

# Terraform Init

- script: terraform init

  displayName: 'Terraform Init'

# Terraform Plan

- script: terraform plan \

    -var="location=$(LOCATION)" \

    -var="rg\_name=$(RG\_NAME)" \

    -var="vm\_admin\_username=$(VM\_ADMIN\_USERNAME)" \

    -var="vm\_admin\_password=$(VM\_ADMIN\_PASSWORD)" \

    -var="vm\_size=$(VM\_SIZE)"

  displayName: 'Terraform Plan'

# Terraform Apply

- script: terraform apply -auto-approve \

    -var="location=$(LOCATION)" \

    -var="rg\_name=$(RG\_NAME)" \

    -var="vm\_admin\_username=$(VM\_ADMIN\_USERNAME)" \

    -var="vm\_admin\_password=$(VM\_ADMIN\_PASSWORD)" \

    -var="vm\_size=$(VM\_SIZE)"

  displayName: 'Terraform Apply'

**Step 3 — Set Up Azure DevOps Service Connection**

1. Go to Azure DevOps → **Project Settings** → **Service connections**.
2. Create a **New Service Connection** → Azure Resource Manager → Service Principal.
3. Select your Azure subscription and name it **Terraform-Azure-Connection** (this name is used in pipeline YAML).
4. Save.

**Step 4 — Run Pipeline**

1. Commit all files (vm.tf, variables.tf, azure-pipelines.yml) to your repo.

git add .

git commit -m "Add Terraform pipeline for existing resource group"

git push origin main

1. Go to Azure DevOps → **Pipelines** → Run pipeline.
2. Select branch → Run.
3. Watch Terraform provisioning logs.

**Step 5 — Verify Outputs**

After pipeline completes:

* Go to Azure Portal → Resource Group raman
* Verify:
  + Virtual Network
  + Subnet
  + Network Interface
  + Public IP
  + Linux VM
  + Network Security Group

**Step 6 — Destroy Infrastructure (Optional)**

Add a pipeline step for destroy:

- script: terraform destroy -auto-approve -var="location=$(LOCATION)" -var="rg\_name=$(RG\_NAME)"

displayName: 'Terraform Destroy'