Azure Lounge – Hands On Terraform Basics

Create a 3 Tier Architecture on Azure using Terraform

Tash Tahir

May 2019

Table of Contents

Azure Lounge – Hands On

Create a 3 Tier Architecture on Azure using Terraform

This Hands on Lab takes you through the steps of creating a Terraform script for a classic 3 tier architecture. Over the course of the lab you will create the Terraform script which will contain your Infrastructure as Code (IaC), and you will deploy the defined infrastructure to Azure using Terraform's command line tools.

> *The final portions of this hands on lab involve deploying resources to an Azure Subscription. You will incur costs for the duration that these resources are deployed. The final section will include instructions on how to clean up the resources you have deployed for this hands on lab.*

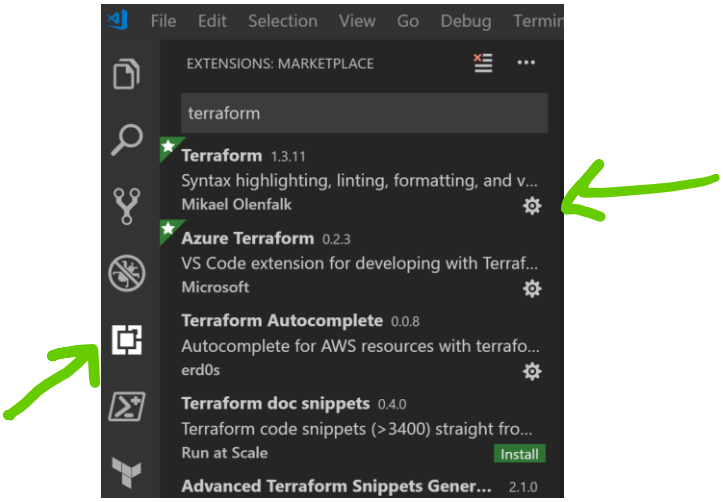
Prerequisites

Visual Studio Code

Install Visual Studio Code

Setup Terraform Extension (These steps help you visualize your code better):

Install Terraform Extension for Visual Studio Code



- Feel free to use Terraform documentation to help you write this code.

- Also, it is good practice to plan and apply after each step. This will help you troubleshoot your code if the need arises.
- You may also have to reload VS Code post Terraform extension installation.

Cloud Shell Basics

Access Cloud Shell

Basic Cloud Shell Commands (Bash):

- Touch NameOfFile.tf ← Creates File
- ls ← Lists files and directories in current path.
- Mkdir ← Make directory
- Cd ← Change directory, move from one directory to another

How to upload files and move them to current working directory:

Creating Your Terraform Script

Create Resource Group:

https://www.terraform.io/docs/providers/azurerm/r/resource_group.html

```
resource "azurerm_resource_group" "rsg" {
    name = "3TierArch"
    location = "East US 2"
    }
```

Create Virtual Network – VNet

https://www.terraform.io/docs/providers/azurerm/r/virtual_network.html

```
//Create Virtual Networks and subnet
    resource "azurerm_virtual_network" "TerraformVNET" {
    name = "HUBVNET"
    address_space = ["10.0.0.0/16"]
    location = "${azurerm_resource_group.rsg.location}"
    resource_group_name = "${azurerm_resource_group.rsg.name}"
    }
```

Create Subnets - Remember, we are creating 3 subnets for Web, App and Database Tier respectively. Hence, you will need to slice up your address space created in previous step accordingly.

https://www.terraform.io/docs/providers/azurerm/r/subnet.html

```
//subnet Web
resource "azurerm_subnet" "TerraformSubnetWeb" {
  name = "Subnet-Web"
  resource_group_name = "${azurerm_resource_group.rsg.name}"
  virtual_network_name ="${azurerm_virtual_network.TerraformVNET.name}"
  address_prefix = "10.0.1.0/24"
  network_security_group_id =
"${azurerm_network_security_group.TerraformrsgNSGWeb.id}"
  }

//subnet App
resource "azurerm_subnet" "TerraformSubnetApp" {
  name = "Subnet-App"
  resource_group_name = "${azurerm_resource_group.rsg.name}"
  virtual_network_name = "${azurerm_virtual_network.TerraformVNET.name}"
  address_prefix = "10.0.2.0/24"
  network_security_group_id =
"${azurerm_network_security_group.TerraformrsgNSGApp.id}"
  }

//subnet DB
resource "azurerm_subnet" "TerraformSubnetDB" {
  name = "Subnet-DB"
  resource_group_name = "${azurerm_resource_group.rsg.name}"
  virtual_network_name = "${azurerm_virtual_network.TerraformVNET.name}"
  address_prefix = "10.0.3.0/24"
  network_security_group_id =
"${azurerm_network_security_group.TerraformrsgNSGDB.id}"
  }
```

Create Network Security Groups – NSGs

1. For this exercise please keep your NSGs completely open. In the real-world, you would only allow very specific traffic through your NSG, inbound and outbound.
2. Also keep in mind you can attach an NSG to a NIC card or a Subnet resource. Hence, you will have to create them accordingly.

https://www.terraform.io/docs/providers/azurerm/r/network_security_group.html

```
//NSG Web
            resource "azurerm_network_security_group" "TerraformrsgNSGWeb" {
            name                = "NSG-Web"
            location            = "${azurerm_resource_group.rsg.location}"
            resource_group_name = "${azurerm_resource_group.rsg.name}"

                security_rule {
                    name                       = "Test-Inbound"
                    priority                   = 100
                    direction                  = "Inbound"
                    access                     = "Allow"
                    protocol                   = "*"
                    source_port_range          = "*"
                    destination_port_range     = "*"
                    source_address_prefix      = "*"
                    destination_address_prefix = "*"
                    }
                security_rule {
                    name                       = "Test-Outbound"
                    priority                   = 100
                    direction                  = "outbound"
                    access                     = "Allow"
                    protocol                   = "*"
                    source_port_range          = "*"
                    destination_port_range     = "*"
                    source_address_prefix      = "*"
                    destination_address_prefix = "*"
                    }
            }
            /*
            resource "azurerm_subnet_network_security_group_association"
"TerraformrsgNSGWeb-Assosiation" {
```

```
                subnet_id                 =
"${azurerm_subnet.TerraformSubnetWeb.id}"
                network_security_group_id =
"${azurerm_network_security_group.TerraformrsgNSGWeb.id}"
            }
            */
//NSG App
            resource "azurerm_network_security_group" "TerraformrsgNSGApp" {
            name                = "NSG-App"
            location            = "${azurerm_resource_group.rsg.location}"
            resource_group_name = "${azurerm_resource_group.rsg.name}"

                security_rule {
                    name                       = "Test-Inbound"
                    priority                   = 100
                    direction                  = "Inbound"
                    access                     = "Allow"
                    protocol                   = "*"
                    source_port_range          = "*"
                    destination_port_range     = "*"
                    source_address_prefix      = "*"
                    destination_address_prefix = "*"
                    }
                security_rule {
                    name                       = "Test-Outbound"
                    priority                   = 100
                    direction                  = "outbound"
                    access                     = "Allow"
                    protocol                   = "*"
                    source_port_range          = "*"
                    destination_port_range     = "*"
                    source_address_prefix      = "*"
                    destination_address_prefix = "*"
                    }
            }
            /*
            resource "azurerm_subnet_network_security_group_association"
"TerraformrsgNSGApp-Assosiation" {
                subnet_id                 =
"${azurerm_subnet.TerraformSubnetApp.id}"
                network_security_group_id =
"${azurerm_network_security_group.TerraformrsgNSGApp.id}"
            }
            */
```

```
//NSG DB
            resource "azurerm_network_security_group" "TerraformrsgNSGDB" {
            name                = "NSG-DB"
            location            = "${azurerm_resource_group.rsg.location}"
            resource_group_name = "${azurerm_resource_group.rsg.name}"

                security_rule {
                    name                       = "Test-Inbound"
                    priority                   = 100
                    direction                  = "Inbound"
                    access                     = "Allow"
                    protocol                   = "*"
                    source_port_range          = "*"
                    destination_port_range     = "*"
                    source_address_prefix      = "*"
                    destination_address_prefix = "*"
                    }
                security_rule {
                    name                       = "Test-Outbound"
                    priority                   = 100
                    direction                  = "outbound"
                    access                     = "Allow"
                    protocol                   = "*"
                    source_port_range          = "*"
                    destination_port_range     = "*"
                    source_address_prefix      = "*"
                    destination_address_prefix = "*"
                    }
            }
            /*
            resource "azurerm_subnet_network_security_group_association"
"TerraformrsgNSGDMZ-Assosiation" {
                subnet_id                 =
"${azurerm_subnet.TerraformSubnetDB.id}"
                network_security_group_id =
"${azurerm_network_security_group.TerraformrsgNSGDB.id}"
            }
            */
```

Create Load Balancers and Public IP

a. Please keep in mind that only your Web Load Balancer will have a Public IP.

b. Also, your load balancers will require a "rule" to allow RDP traffic.

https://www.terraform.io/docs/providers/azurerm/r/loadbalancer.html

https://www.terraform.io/docs/providers/azurerm/r/loadbalancer_rule.html

```
//LoadBalancer Web
resource "azurerm_public_ip" "LoadBalancerPublicIP" {
  name                = "PublicIPForLB"
  location            = "${azurerm_resource_group.rsg.location}"
  resource_group_name = "${azurerm_resource_group.rsg.name}"
  public_ip_address_allocation = "dynamic"
  //allocation_method   = "Static"
}

resource "azurerm_lb" "LoadBalancerWeb" {
  name                = "WebLoadBalancer"
  location            = "${azurerm_resource_group.rsg.location}"
  resource_group_name = "${azurerm_resource_group.rsg.name}"

  frontend_ip_configuration {
    name                 = "Web-PublicFacing"
    public_ip_address_id = "${azurerm_public_ip.LoadBalancerPublicIP.id}"
  }
}

resource "azurerm_lb_backend_address_pool" "Web-LBBackendPool" {
  resource_group_name = "${azurerm_resource_group.rsg.name}"
  loadbalancer_id     = "${azurerm_lb.LoadBalancerWeb.id}"
  name                = "Web-BackEndAddressPool"
}

resource "azurerm_lb_rule" "RDP-LBRule" {
  resource_group_name            = "${azurerm_resource_group.rsg.name}"
  loadbalancer_id                = "${azurerm_lb.LoadBalancerWeb.id}"
  name                           = "RDP-LBRule"
  protocol                       = "Tcp"
  frontend_port                  = 3389
  backend_port                   = 3389
  frontend_ip_configuration_name = "Web-PublicFacing"
}


// LoadBalancer - App
```

```
resource "azurerm_lb" "LoadBalancerApp" {
  name                = "AppLoadBalancer"
  location            = "${azurerm_resource_group.rsg.location}"
  resource_group_name = "${azurerm_resource_group.rsg.name}"

  frontend_ip_configuration {
    name                 = "App-SubnetFacing"
    subnet_id = "${azurerm_subnet.TerraformSubnetApp.id}"
  }
}
resource "azurerm_lb_backend_address_pool" "App-LBBackendPool" {
  resource_group_name = "${azurerm_resource_group.rsg.name}"
  loadbalancer_id     = "${azurerm_lb.LoadBalancerApp.id}"
  name                = "App-BackEndAddressPool"
}
```

Create Virtual Machine

a. For this exercise please create only one Web, App and DB VM.

b. Please note, you will need to create a NIC card as well which is considered a separate resource within Azure.

c. This code uses large VMs to help reduce config time. Please make sure you delete them as soon as you're finished with this exercise.

https://www.terraform.io/docs/providers/azurerm/r/virtual_machine.html

```
//Web NIC
resource "azurerm_network_interface" "Web-TerraformNIC" {
  name                = "WebNic"
  location            = "${azurerm_resource_group.rsg.location}"
  resource_group_name = "${azurerm_resource_group.rsg.name}"

  ip_configuration {
    name                          = "WebNIC-Config"
    subnet_id                     = "${azurerm_subnet.TerraformSubnetWeb.id}"
    private_ip_address_allocation = "dynamic"
    load_balancer_backend_address_pools_ids =
["${azurerm_lb_backend_address_pool.Web-LBBackendPool.id}"]
  }
}
/*
```

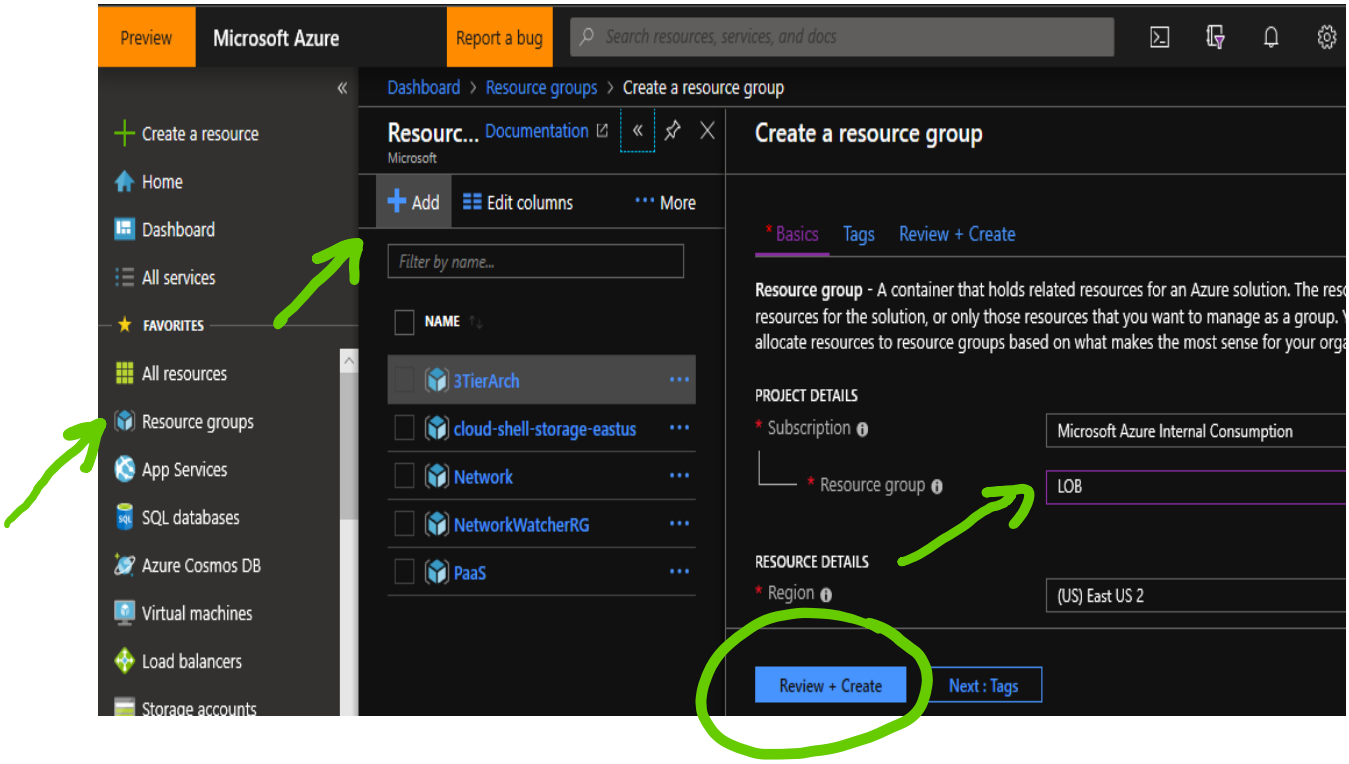
```
resource "azurerm_network_interface_backend_address_pool_association"
"WebLBBackEndNIC" {
  network_interface_id    = "${azurerm_network_interface.Web-TerraformNIC.id}"
  ip_configuration_name   = "WebNIC-Config"
  backend_address_pool_id = "${azurerm_lb_backend_address_pool.Web-
LBBackendPool.id}"
}
*/

//Web VM
resource "azurerm_virtual_machine" "TerraformVM" {
  name                  = "WebVM1"
  location              = "${azurerm_resource_group.rsg.location}"
  resource_group_name   = "${azurerm_resource_group.rsg.name}"
  network_interface_ids = ["${azurerm_network_interface.Web-TerraformNIC.id}"]
  //primary_network_interface_id = "${azurerm_network_interface.Web-
TerraformNIC.id}"
  vm_size               = "Standard_D64s_v3"

  // comment this line to delete the OS disk automatically when deleting the VM
  delete_os_disk_on_termination = true
  // Uncomment this line to delete the data disks automatically when deleting the
VM
  delete_data_disks_on_termination = true

  storage_image_reference {
    publisher = "MicrosoftWindowsServer"
    offer     = "WindowsServer"
    sku       = "2016-Datacenter"
    version   = "latest"
  }

  storage_os_disk {
    name              = "myosdiskWeb"
    caching           = "ReadWrite"
    create_option     = "FromImage"
    managed_disk_type = "Standard_LRS"
  }

  os_profile {
    computer_name  = "WebServer"
    admin_username = "testadmin"
    admin_password = "Password1234!"
  }
```

```
  os_profile_windows_config {
    enable_automatic_upgrades = false
    provision_vm_agent = true
  }
}

//App NIC
resource "azurerm_network_interface" "App-TerraformNIC" {
  name                = "AppNic"
  location            = "${azurerm_resource_group.rsg.location}"
  resource_group_name = "${azurerm_resource_group.rsg.name}"

  ip_configuration {
    name                          = "AppNIC-Config"
    subnet_id                     = "${azurerm_subnet.TerraformSubnetApp.id}"
    private_ip_address_allocation = "dynamic"
    load_balancer_backend_address_pools_ids =
["${azurerm_lb_backend_address_pool.App-LBBackendPool.id}"]
  }
}
/*
resource "azurerm_network_interface_backend_address_pool_association"
"AppLBBackEndNIC" {
  network_interface_id    = "${azurerm_network_interface.App-TerraformNIC.id}"
  ip_configuration_name   = "AppNIC-Config"
  backend_address_pool_id = "${azurerm_lb_backend_address_pool.App-
LBBackendPool.id}"
}
*/

//App VM
resource "azurerm_virtual_machine" "App-TerraformVM" {
  name                = "AppVM1"
  location            = "${azurerm_resource_group.rsg.location}"
  resource_group_name   = "${azurerm_resource_group.rsg.name}"
  network_interface_ids = ["${azurerm_network_interface.App-TerraformNIC.id}"]
  //primary_network_interface_id = "${azurerm_network_interface.Web-
TerraformNIC.id}"
  vm_size             = "Standard_D32s_v3"

  // comment this line to delete the OS disk automatically when deleting the VM
  delete_os_disk_on_termination = true
```

```
  // Uncomment this line to delete the data disks automatically when deleting the
VM
  delete_data_disks_on_termination = true

  storage_image_reference {
    publisher = "MicrosoftWindowsServer"
    offer     = "WindowsServer"
    sku       = "2016-Datacenter"
    version   = "latest"
  }

  storage_os_disk {
    name              = "myosdiskApp"
    caching           = "ReadWrite"
    create_option     = "FromImage"
    managed_disk_type = "Standard_LRS"
  }

  os_profile {
    computer_name  = "AppServer"
    admin_username = "testadmin"
    admin_password = "Password1234!"
  }

  os_profile_windows_config {
    enable_automatic_upgrades = false
    provision_vm_agent = true
  }
}
```

Final Steps

LOB with VNET Peering: Typically, when you walk into an existing Azure environment as a Line of Business. You will probably be given a Resource Group with enough rights to create your own resources within it. In most Hub and Spoke models, you will Peer with existing VNET (HUB). For this we will use the "Data Source" function.



1. Click "Review + Create." Then Click Create.

We will be using the Data source field to pull information regarding our Resource Groups and VNets to create the Peering. And subsequent resources.

2. In Cloud Shell, create a LOB Directory.
    a. Mkdir LOB
        i. cd LOB
    b. Upload the following file.

```
//Notice instead of using "Resource" we are using "data"
//This is pulling RG related information, which we will using going forward.
data "azurerm_resource_group" "LOBRG" {
  name = "LOB"
}

data "azurerm_resource_group" "HUBRG" {
  name = "3TierArch"
}

data "azurerm_virtual_network" "HubVnet" {
    name = "HUBVNET"
    resource_group_name = "3TierArch"
}

//VNET
```

```
//Notice how we call the data for location and RG Name.
resource "azurerm_virtual_network" "LOBVNET" {
  name                = "LOBVNET"
  location            = "${data.azurerm_resource_group.LOBRG.location}"
  resource_group_name = "${data.azurerm_resource_group.LOBRG.name}"
  address_space       = ["11.0.0.0/16"]

  subnet {
    name           = "LOBsubnet"
    address_prefix = "11.0.1.0/24"
    }
  }

//VNET Peering
resource "azurerm_virtual_network_peering" "HubtoSpoke" {
  name                      = "peerhubtospoke"
  resource_group_name       = "${data.azurerm_resource_group.HUBRG.name}"
  virtual_network_name      = "${data.azurerm_virtual_network.HubVnet.name}"
  remote_virtual_network_id = "${azurerm_virtual_network.LOBVNET.id}"
}

resource "azurerm_virtual_network_peering" "SpoketoHub" {
  name                      = "peerspoketohub"
  resource_group_name       = "${data.azurerm_resource_group.LOBRG.name}"
  virtual_network_name      = "${azurerm_virtual_network.LOBVNET.name}"
  remote_virtual_network_id = "${data.azurerm_virtual_network.HubVnet.id}"
}
```

a. Move the file to current working directory.
   i. mv /home/tash/LOB.tf .

3. You can view the contents of the file within Visual Studio Code for better visualization or you can type "cat LOB.tf" and have a look at the difference in code.

https://www.terraform.io/docs/providers/azurerm/d/resource_group.html

https://www.terraform.io/docs/providers/azurerm/d/virtual_network.html

In the interest of time, we are only creating the VNET and peering it back and forth. For your final step, feel free to create a Load Balancer, NSG and VM and see if you can RDP back and forth from one VNET to another across the Peering. Hint, you may need to create a Route table…