|  |  |
| --- | --- |
|  | Azure Lounge – Hands On Terraform Basics |
|  | Create a 3 Tier Architecture on Azure using Terraform |
|  | Tash Tahir  May 2019 |

Table of Contents

[Prerequisites 2](#_Toc9944628)

[Visual Studio Code 2](#_Toc9944629)

[Update when things change 3](#_Toc9944630)

[Dive deeper than Heading 1 4](#_Toc9944631)

[Customize your TOC 4](#_Toc9944632)

[Remove a TOC 5](#_Toc9944633)

[Explore more 6](#_Toc9944634)

[Change text formatting of the TOC entries 6](#_Toc9944635)

[Change the number of TOC levels 7](#_Toc9944636)

[Get help in Word 8](#_Toc9944637)

[Let us know what you think 8](#_Toc9944638)

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.

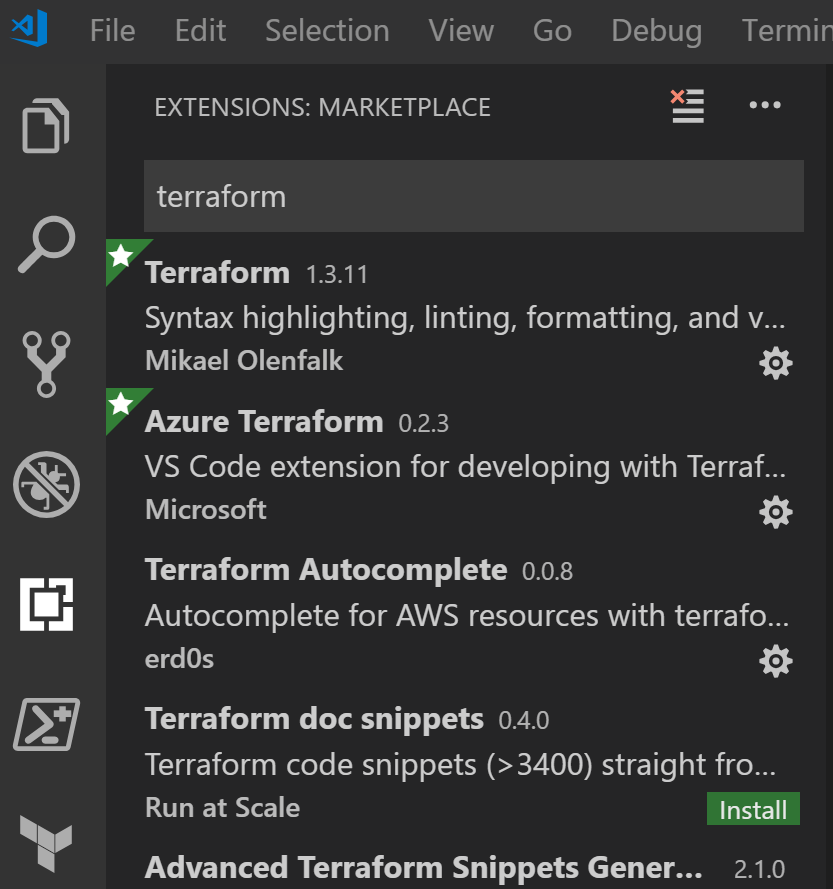
# Prerequisites

## 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

* 1. Please keep in mind that only your Web Load Balancer will have a Public IP.
  2. 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

1. For this exercise please create only one Web, App and DB VM.
2. Please note, you will need to create a NIC card as well which is considered a separate resource within Azure.
3. 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

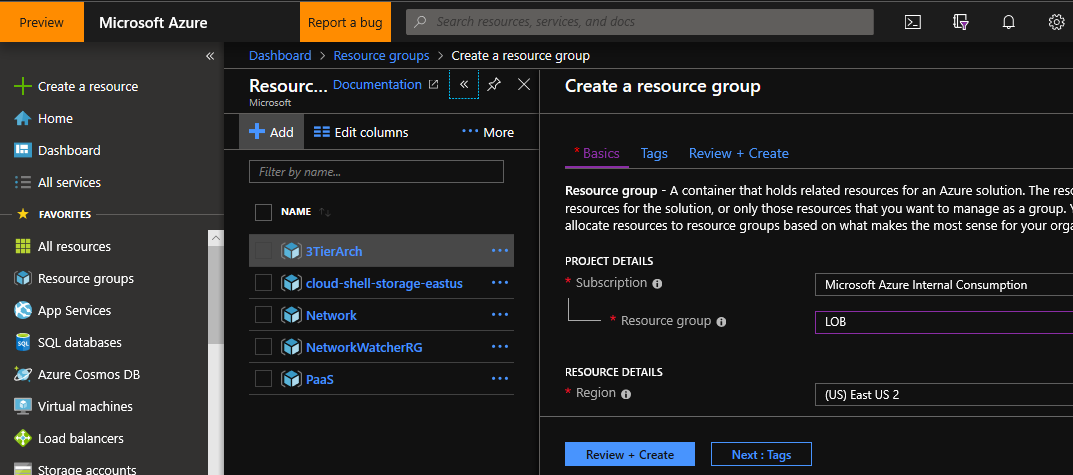
provision\_vm\_agent = true

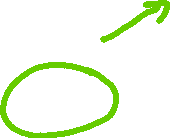
}

}

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.

1. In Cloud Shell, create a LOB Directory.
   1. Mkdir LOB
      1. cd LOB
   2. 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}"

}

1. Move the file to current working directory.
   * 1. mv /home/tash/LOB.tf .
2. 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…