Microsoft

Terraform with Azure Lab Guide

Harshal Dharia & Derrick Schwartz
10-4-2018

Table of Contents

Lab 1 – Setup Terraform

Important Resources:

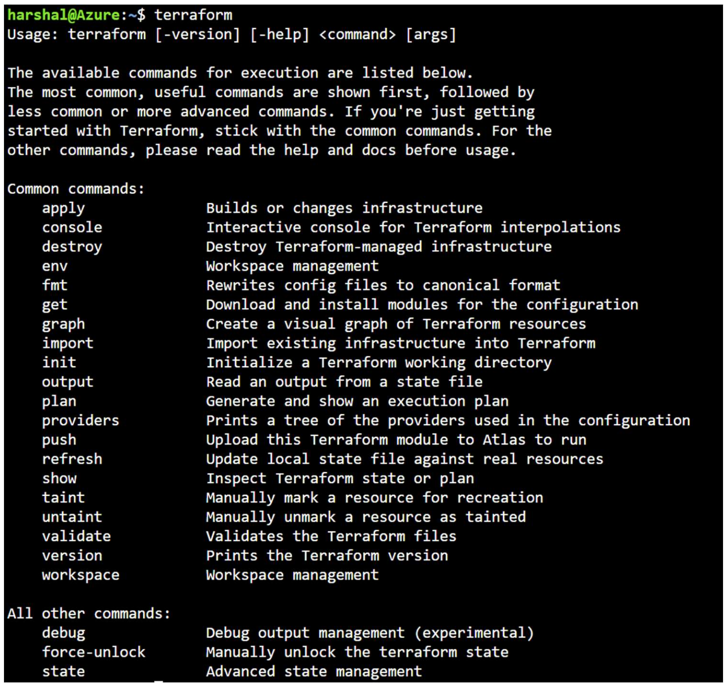
Installation - https://www.terraform.io/intro/getting-started/install.html

Installation & configuration - https://docs.microsoft.com/en-us/azure/virtual-machines/linux/terraform-install-configure

Terraform Documentation - https://www.terraform.io/docs/providers/azurerm/

Exercise 1 – Create Linux Virtual Machine and install terraform

- Provision a ubuntu VM
- Install Terraform
- Verify Terraform is installed



```
harshal@Azure:~$ terraform
Usage: terraform [-version] [-help] <command> [args]

The available commands for execution are listed below.
The most common, useful commands are shown first, followed by
less common or more advanced commands. If you're just getting
started with Terraform, stick with the common commands. For the
other commands, please read the help and docs before usage.

Common commands:
    apply              Builds or changes infrastructure
    console            Interactive console for Terraform interpolations
    destroy            Destroy Terraform-managed infrastructure
    env                Workspace management
    fmt                Rewrites config files to canonical format
    get                Download and install modules for the configuration
    graph              Create a visual graph of Terraform resources
    import             Import existing infrastructure into Terraform
    init               Initialize a Terraform working directory
    output             Read an output from a state file
    plan               Generate and show an execution plan
    providers          Prints a tree of the providers used in the configuration
    push               Upload this Terraform module to Atlas to run
    refresh            Update local state file against real resources
    show               Inspect Terraform state or plan
    taint              Manually mark a resource for recreation
    untaint            Manually unmark a resource as tainted
    validate           Validates the Terraform files
    version            Prints the Terraform version
    workspace          Workspace management

All other commands:
    debug              Debug output management (experimental)
    force-unlock       Manually unlock the terraform state
    state              Advanced state management
```

Exercise 2 – Launch Cloud Shell and launch terraform using the 'terraform' command

- Create Cloud Shell environment
    - Note: Go to Advanced setting and the cloud shell environment in the resource group assigned to the account.
- Verify Terraform is installed

Azure Cloud Shell – https://shell.azure.com

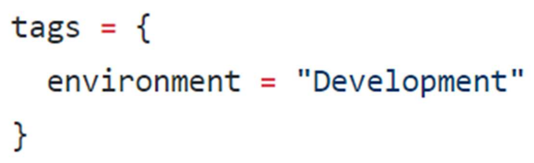
Note: All the exercises after this will be done in cloudshell.

Exercise 3 – Setup initial terraform project

- Clone - https://github.com/hdharia/Terraform-MSFT-Hackathon.git
- Navigate to Lab1
- Use Service Principal to authenticate into Terraform
- Update the variables file
- Execute – 'terraform init' to download the azure resource provider
- Execute – 'terraform plan' to see the changes
- Execute - 'terraform apply' to see the changes applied to Azure

Exercise 4 – Tag the Resource Group

- Modify the terraform script to add tagging to the resource group definition



```
tags = {
  environment = "Development"
}
```

- Apply the tag to Azure Resource by executing 'terraform apply'

Lab 2 – Provision IaaS infrastructure

Exercise 0 – Configure variables & resources

- Navigate to Lab2
- Update resource group to match based on your ID
- Update variables

Exercise 1 – Create Networking Infrastructure

During this exercise you will create the Virtual Network resource as well as specify what the subnet will be. This is the network your virtual machine will be connected to. For naming purposes and to ensure we do not have any duplicates you will use the format() function to inject a variable of studentID in to each objects name.

Format() leverages the sprintf syntax for formatting text strings. There are links below to help you gain a stronger understanding of this syntax. For this Lab you need to know that %s reprsents a string to insert in the name. The syntax we will be using is as follows

```
name                    = "${format("%s-vnet", var.StudentID)}"
```

What this tells terraform to insert the variable Student ID where the %s is located. So the final output or name of the resource would be StudentID-vnet. This can be confusing at first but will be used throughout the exercise in all of the naming constructs.

https://www.terraform.io/docs/configuration/interpolation.html#format-format-args-

https://golang.org/pkg/fmt/

- Update the Lab2.tf script to provision the virtual network, the following values will be needed:
    - Name – Use the format() structure to create your unique name
    - Address_Space – use the default 10.0.0.0/16
    - Location – reference the variable from the file
    - Resource_group_name – reference the variable from the file
- Update the Lab2.tf script to provision the subnet, the following values will be needed:
    - Name – Use the format() structure to create a unique name
    - Resource_group_name – reference the variable from the file
    - Virtual_network_name – callback to the resource you created above
    - Address_prefix – use default 10.0.1.0/24
- Run the Terraform plan command to check that your syntax is correct
- Run the Terraform apply command to deploy your virtual network

Azure Virtual Network - https://www.terraform.io/docs/providers/azurerm/d/virtual_network.html

Exercise 2 – Create Public IP

During this exercise you will create a public IP address for the virtual machine you will create. This allows us to remote desktop into the virtual machine after the Lab is complete to ensure it is running and was provisioned correctly.

- Update the Lab2.tf script to provision the public ip address the following values will be needed:
    - Name – use the format() structure to create a unique name
    - Location – reference the variable from the file
    - Resource_group_name – reference the variable from the file
    - Public_ip_address_allocation – this should be dynamic for this exercise
- Run the Terraform plan command to check that your syntax is correct
- Run the Terraform apply command to deploy your public IP address

Azure Public IP Address - https://www.terraform.io/docs/providers/azurerm/d/public_ip.html

Exercise 3 – Create Network Interface

Each Virtual machine requires a network interface. Each virtual NIC requires a Subnet on a virtual network to connect to. We will use the virtual network and subnet we previously created in the script as well as attach the public IP to this network card when we create it. This is done by specifying the previous objects in the ip_configuration argument of the network interface.

- Update the Lab2.tf script to provision the virtual network interface, the following values will be needed:
    - Name – use the format() structure to create a unique name
    - Location – reference the variable from the file
    - Resource_group_name – reference the variable from the file
- Leverage the following values for the IP configuration argument:

    - Name – use the format() structure to create a unique name
    - Subnet_id – this will be a call back to the id of the subnet created in Exercise 1
    - Private_ip_address_allocation – we will use dynamic for this exercise
    - Public_ip_address_id – this will be a call back to the id of public ip we created in Exercise 2
- Run the Terraform plan command to check that your syntax is correct
- Run the Terraform apply command to deploy your Network Interface

Azure Network Interface - https://www.terraform.io/docs/providers/azurerm/r/network_interface.html

Exercise 4 – Create Virtual Machine

In this exercise we will bring all of the things we have done before together to specify all the details necessary to create a virtual machine with Terraform. Because of the complexity of a virtual machine there are several arguments with information that must be provided. The Storage_Image_reference will specify which marketplace image to use for the OS, Storage_os_disk will create the managed disk we will use for our primary OS disk, os_profile provides the local information for the OS, and os_profile_windows_config sets any information specific to windows machines.

- Update the Lab2.tf script to provision the virtual machine, the following values will be needed:
    - Name – use the format() structure to create a unique name
    - Location – reference the variable from the file
    - Resource_group_name – reference the variable from the file
    - Network_interface_ids – this is a call back to the network interface we created in Exercise 3, pay close attention to the plural in ids this field supports multiple interfaces so it should be in a list format
    - Vm_size – we will be using Standard_DS2_V2 machines for this exercise
    - Delete_os_disk_on_termination – this sets the flag to delete the disk when the machine is destroyed we will use true for easier clean up
    - Delete_data_disks_on_termination – this sets the flag to delete the non os data disks when the machine is destoryed. We are not using any data disks but for completeness we will set for true for easiewr clean up if we add them later
- Leverage the following information for the Storage_image_reference argument:
    - Publisher – MicrosoftWindowsServer
    - Offer – WindowsServer
    - Sku – 2016-Datacenter
    - Version – latest

This information can be found by searching the web or using powershell to get lists of the publishers, offers, and skus. This document will walk you through getting those. https://docs.microsoft.com/en-us/azure/virtual-machines/windows/cli-ps-findimage

- Leverage the following information for the storage_os_disk argument:
    - Name – use the format() structure to create a unique name
    - Managed_disk_type - for this exercise we are using standard_LRS
    - Create_option – we are creating FromImage
    - Os_type – this will be a Windows type machine

- Levarage the following information for the os_profile argument:
    - Computer_name – this can be anything that meets windows naming standards
    - Admin_username – this can be anything you want
    - Admin_password – this can be anything you want but remember to login for verfication steps

admin_password must be between 6-72 characters long and must satisfy at least 3 of password complexity requirements from the following: 1. Contains an uppercase character 2. Contains a lowercase character 3. Contains a numeric digit 4. Contains a special character

- Leverage the following information for the os_profile_windows_config argument:
    - Provision_vm_agent – we want to set this to true so we will have an agent to monitor the system and allow us to connect with RDP
- Run the Terraform plan command to check that your syntax is correct
- Run the Terraform apply command to deploy your Virtual Machine

Azure Virtual Machine - https://www.terraform.io/docs/providers/azurerm/r/virtual_machine.html

Exercise 5 – Update the VM Size and Re-Provision

In this lab we will update the machine size to a new one and redeploy the solution. we will verify the machine size changed.

- In the Lab2.tf script Update the vm_size to Standard_DS3_V2
- Run the Terraform plan command and make note of what it says will happen
- Run the Terraform apply command

Notice the old machine is destroyed including any changes and a new machine is provisioned with the new specs.

- Verify that VM Size is updated by navigating to portal

Exercise 6 – Destroy the Virtual Machine

We will now clean up the lab environment from the previous exercises. This is easy by leveraging the destory command in Terraform, it will read the state and destroy anything it created.

- Run the Terraform destroy command to remove all of the resources we have created

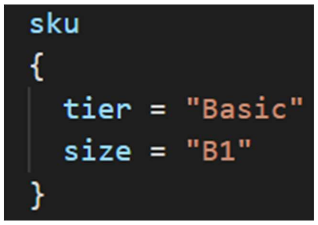
Lab 3 – Provision PaaS infrastructure

Exercise 0 – Configure variables & resources

- Navigate to Lab3 in the source code
- Update resource group to match based on your ID
- Update variables

The solution for lab3 is under the solution directory. Use this as a resource of last resort.

Exercise 1 – Provision App Service Plan



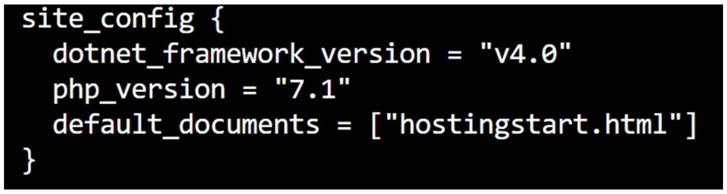
```
sku
{
  tier = "Basic"
  size = "B1"
}
```

- Add SKU to azurerm_app_service_plan within the lab3.tf file
- Provision App Service Plan using the 'terraform apply' command
- Update the SKU to Standard Tier and S1 Plan
- Apply the changes using the 'Terraform apply' command again.

Commented [JH1]: The snippet you provided for this is actually already in the script. May be more appropriate to say something like "view the sku and app service plan, then apply the config"... Then after that have the update the tier and plan and reapply.

Exercise 2 – Provision Azure Webapps

- Update azurerm_app_service to provision a default application for .net 4.0, php 7.1 and default_documents as 'hostingstart.html'



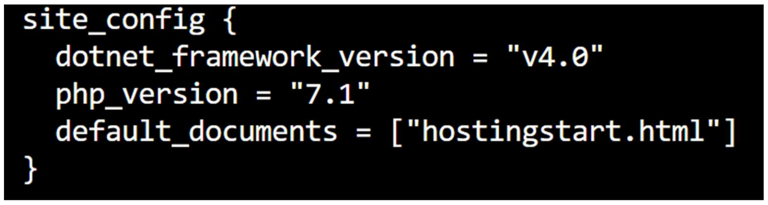
```
site_config {
  dotnet_framework_version = "v4.0"
  php_version = "7.1"
  default_documents = ["hostingstart.html"]
}
```

- Plan and apply the changes

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

Exercise 3 – Provision a Deployment Slot

- Update `azurerm_app_service_slot` to provision a deployment slot for the application defined in exercise 3 for .net 4.0, php 7.1 and default_documents as 'hostingstart.html'



```
site_config {
  dotnet_framework_version = "v4.0"
  php_version = "7.1"
  default_documents = ["hostingstart.html"]
}
```

- Plan and apply the changes

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

Exercise 4 – Provision a SQL DB

- Update azurerm_sql_server and azurerm_sql_database
- Uncomment the output to see the connection string for the database
- Ensure that SQL database is provision prior to webapps by adding a dependency
- Plan and apply the changes

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

Sample: https://github.com/terraform-providers/terraform-provider-azurerm/tree/master/examples/tf-sql-database

Outputs in Terraform: https://www.terraform.io/intro/getting-started/outputs.html

Dependency in Terraform: https://www.terraform.io/intro/getting-started/dependencies.html

Exercise 5 – Update Connection String

- Add connection string information from the database provisioned in exercise 4
- Apply the changes

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