Terraform Modules

Expected Outcome

In this challenge, you will create a module to contain a scalable virtual machine deployment, then create an environment where you will call the module.

How to

Connect with Service Principal

Should you have more than one Subscription, you can specify the Subscription to use via the following command:

```
$ az account set --subscription="SUBSCRIPTION_ID"
```

We can now create the Service Principal which will have permissions to manage resources in the specified Subscription using the following command:

```
$ az ad sp create-for-rbac --role="Contributor" --
scopes="/subscriptions/SUBSCRIPTION_ID"
```

This command will output 5 values:

```
{
  "appId": "00000000-0000-0000-0000-000000000000", #client_id
  "displayName": "azure-cli-2017-06-05-10-41-15",
  "name": "http://azure-cli-2017-06-05-10-41-15",
  "password": "0000-0000-0000-0000-000000000000",  # client_secret
  "tenant": "00000000-0000-0000-0000-000000000000" # tenant_id
}
```

Replace the provider as follows:

```
provider "azurerm" {
  # Whilst version is optional, we /strongly recommend/ using it to pin the version
of the Provider being used
  version = "=1.38.0"

  subscription_id = "00000000-0000-0000-0000-000000000000"
  client_id       = "00000000-0000-0000-0000-000000000000"
  client_secret   = "${var.client_secret}"
  tenant_id       = "00000000-0000-0000-0000-000000000000"
}
```

When storing the credentials as Environment Variables, for example:

```
$ export ARM_CLIENT_ID="00000000-0000-0000-0000-000000000000"
$ export ARM_CLIENT_SECRET="00000000-0000-0000-0000-000000000000"
$ export ARM_SUBSCRIPTION_ID="00000000-0000-0000-0000-000000000000"
$ export ARM_TENANT_ID="00000000-0000-0000-0000-000000000000"
```

Create Folder Structure

Change directory into a folder specific to this challenge.

For example: `cd ~/TerraformWorkshop/201-vm-module/`.

In order to organize your code, create the following folder structure with `main.tf` files.

```
├ main.tf
└── modules
    └── my_linux_vm
        └── main.tf
```

Create the Module

Inside the `my_linux_vm` module folder there should be a `main.tf` file with the following contents:

> Note: This is very similar to the original VM lab.

```
variable "prefix" {}
variable "location" {}
variable "username" {}
variable "vm_size" {}

resource "random_password" "password" {
  length           = 16
  special          = true
  override_special = "!"
}

resource "azurerm_resource_group" "main" {
  location = var.location
  name     = "${var.prefix}-my-rg"
}

resource "azurerm_virtual_network" "main" {
  resource_group_name = azurerm_resource_group.main.name
  location            = azurerm_resource_group.main.location
  name                = "${var.prefix}-my-vnet"
  address_space       = ["10.0.0.0/16"]
}

resource "azurerm_subnet" "main" {
  resource_group_name  = azurerm_resource_group.main.name
  virtual_network_name = azurerm_virtual_network.main.name
  name                 = "${var.prefix}-my-subnet"
  address_prefix       = "10.0.1.0/24"
}

resource "azurerm_network_interface" "main" {
  name                = "${var.prefix}-my-nic"
  location            = azurerm_resource_group.main.location
  resource_group_name = azurerm_resource_group.main.name

  ip_configuration {
    name                          = "config1"
    subnet_id                     = azurerm_subnet.main.id
    private_ip_address_allocation = "dynamic"
    public_ip_address_id          = azurerm_public_ip.main.id
  }
}
```

```
resource "azurerm_virtual_machine" "main" {
  name                  = "${var.prefix}-my-vm"
  location              = azurerm_resource_group.main.location
  resource_group_name   = azurerm_resource_group.main.name
  network_interface_ids = [azurerm_network_interface.main.id]
  vm_size               = "${var.vm_size}"

  storage_image_reference {
    publisher = "Canonical"
    offer     = "UbuntuServer"
    sku       = "18.04-LTS"
    version   = "latest"
  }

  storage_os_disk {
    name              = "${var.prefix}myvm-osdisk"
    caching           = "ReadWrite"
    create_option     = "FromImage"
    managed_disk_type = "Standard_LRS"
  }

  os_profile_linux_config {
    disable_password_authentication = false
  }

  os_profile {
    computer_name  = "${var.prefix}myvm"
    admin_username = var.username
    admin_password = random_password.password.result
  }
}

resource "azurerm_public_ip" "main" {
  name                = "${var.prefix}-my-pubip"
  location            = azurerm_resource_group.main.location
  resource_group_name = azurerm_resource_group.main.name
  allocation_method   = "Static"
}

output "vm-password" {
  value       = random_password.password.result
  description = "Dynamically generated password to access the VM."
}
output "private-ip" {
  value       = azurerm_network_interface.main.private_ip_address
  description = "Private IP Address"
}

output "public-ip" {
  value       = azurerm_public_ip.main.ip_address
  description = "Public IP Address"
}
```

Create Variables in Root

In your root directory, there should be a `main.tf` file.

Create "prefix", "location", and "username" variables without defaults.

This will result in them being required.

```
variable "prefix" {}
variable "location" {}
variable "username" {}
```

> Extra credit: How many other variables can you extract?

Pass in Variables

Create a file called 'terraform.tfvars' and add the following variables:

```
prefix   = ""
location = ""
username = ""
```

Create the Module declaration in Root

Update main.tf to declare your module, it could look similar to this:

```
module "myawesomelinuxvm-a" {
  source    = "./modules/my_linux_vm"
}
```

> Notice the relative module sourcing.

Terraform Init

Run `terraform init`.

```
Initializing modules...
- module.myawesomewindowsvm
  Getting source "./modules/my_linux_vm"
```

Terraform Plan

Run `terraform plan`.

```


Error: Missing required argument

  on main.tf line 1, in module "myawesomelinuxvm-a":
   1: module "myawesomelinuxvm-a" {

The argument "prefix" is required, but no definition was found.


Error: Missing required argument

  on main.tf line 1, in module "myawesomelinuxvm-a":
   1: module "myawesomelinuxvm-a" {

The argument "location" is required, but no definition was found.


Error: Missing required argument

  on main.tf line 1, in module "myawesomelinuxvm-a":
   1: module "myawesomelinuxvm-a" {
```

```
The argument "username" is required, but no definition was found.


Error: Missing required argument

  on main.tf line 1, in module "myawesomelinuxvm-a":
   1: module "myawesomelinuxvm-a" {

The argument "vm_size" is required, but no definition was found.
```

We have a problem! We didn't set required variables for our module.

Update the `main.tf` file:

```
module "myawesomelinuxvm-a" {
  source   = "./modules/my_linux_vm"
  prefix   = "${var.prefix}a"
  location = var.location
  username = var.username
  vm_size  = "Standard_A2_v2"
}
```

Run `terraform plan` again, this time there should not be any errors and you should see your VM built from your module.

```
  + module.myawesomelinuxvm-a.azurerm_resource_group.module
      id:                                    <computed>
      location:                              "centralus"

...

Plan: 7 to add, 0 to change, 0 to destroy.
```

Add Another Module

Add another `module` block describing another set of Virtual Machines:

```
module "myawesomelinuxvm-b" {
  source   = "./modules/my_linux_vm"
  prefix   = "${var.prefix}b"
  location = var.location
  username = var.username
  vm_size  = "Standard_A2_v2"
}
```

Terraform Plan

Since we added another module call, we must run `terraform init` again before running `terraform plan`.

We should see twice as much infrastructure in our plan.

```
  # module.myawesomewindowsvm.azurerm_resource_group.main will be created
  + resource "azurerm_resource_group" "main" {
      + id       = (known after apply)
      + location = "centralus"
    }
```

```
...

  # module.differentwindowsvm.azurerm_resource_group.main will be created
  + resource "azurerm_resource_group" "main" {
      + id       = (known after apply)
      + location = "centralus"
    }

...

Plan: 14 to add, 0 to change, 0 to destroy.
```

More Variables

In your `main.tf` file at the root we can see some duplication.

Extract "vm_size" into a local variable to your environment `main.tf` and reference in each module.

```
locals {
  vm_size = "Standard_A2_v2"
}
```

Now reference them in the module blocks:

```
module "myawesomelinuxvm-a" {
  ...
  vm_size = local.vm_size
}

module "myawesomelinuxvm-b" {
  ...
  vm_size = local.vm_size
}
```

Terraform Plan

Run `terraform plan` and verify that your plan succeeds and looks the same.

> Note: Feel free to apply this infrastructure to validate the workflow. Be sure to destroy when you are done.

Advanced areas to explore

1. Extend module outputs to root level outputs.
2. Extract the Resource Group, Virtual Network, and Subnet into a "Networking" module and the Network Interface and Virtual Machine into a "VM" module and reference them with module declarations.
3. Update the VM module to use SSH instead of password authentication.
4. Add a reference to the Public Terraform Module for Azure Compute

Resources

- Using Terraform Modules
- Source Terraform Modiules
- Public Module Registry