Terraform Modules Azure Lab 1.11

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.

Steps

1.0 Create Folder Structure

From the Cloud Shell, change directory into a folder specific to this challenge.

Create a directory lab1.11 and change directory into it.

```
mkdir lab1.11
```

```
cd lab1.11
```

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

```
lab1.11
├── environments
│   └── dev
│       └── main.tf
└── modules
    └── my_virtual_machine
        └── main.tf
```

1.2 Create the Module

Inside the `my_virtual_machine` module folder and add the following to main.tf

```
provider "azurerm" {
  features {}
}provider "azurerm" {
  features {}
}

terraform {
  required_version = "= 0.11.15" # check ???? is this ok ????
```

```
}


variable "vm_size" {}
variable "username" {}
variable "password" {}

variable "name" {
  default = "initials-xxx"
}

variable "location" {
  default = "uksouth"
}

variable "vmcount" {
  default = 0
  # default = 2
}

# Basic Resources
resource "azurerm_resource_group" "main" {
  name     = "${var.name}-rg"
  location = var.location
}

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

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

# VM Resources

resource "azurerm_public_ip" "main" {
  name                = "${var.name}-pubip${count.index}"
  location            = azurerm_resource_group.main.location
  resource_group_name = azurerm_resource_group.main.name
  allocation_method   = "Static"
  count               = var.vmcount
```

```
}

resource "azurerm_network_interface" "main" {
  name                = "${var.name}-nic${count.index}"
  location            = azurerm_resource_group.main.location
  resource_group_name = azurerm_resource_group.main.name
  count               = var.vmcount

  ip_configuration {
    name                          = "config1"
    subnet_id                     = azurerm_subnet.main.id
    public_ip_address_id          = element(azurerm_public_ip.main.*.id,
count.index)
    private_ip_address_allocation = "dynamic"
  }
}

resource "azurerm_virtual_machine" "main" {
  name                  = "${var.name}-vm${count.index}"
  location              = azurerm_resource_group.main.location
  resource_group_name   = azurerm_resource_group.main.name
  network_interface_ids = ["${element(azurerm_network_interface.main.*.id,
count.index)}"]
  vm_size               = var.vm_size
  count                 = var.vmcount

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

  storage_os_disk {
    name              = "${var.name}vm${count.index}-osdisk"
    caching           = "ReadWrite"
    create_option     = "FromImage"
    managed_disk_type = "Standard_LRS"
  }

  os_profile {
    computer_name = "${var.name}vm${count.index}"
    admin_username = var.username
    admin_password = var.password
  }

  os_profile_windows_config {}
}
```

```
## Outputs
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"
}
```

1.3 Create Variables

Extract name, vm size, username and password into variables without defaults.

This will result in them being required.

```
variable "name" {}
variable "vm_size" {}
variable "username" {}
variable "password" {}
```

Extra credit: How many other variables can you extract?

1.4 Create the Environment

Change your working directory to the `environments/dev` folder. Update main.tf to declare your module, it could look similar to this:

```
variable "username" {}

variable "password" {}

module "myawesomewindowsvm" {
  source = "../../modules/my_virtual_machine"
name   = "awesomeapp" }
```

Notice the relative module sourcing.

Terraform Init

Run `terraform init`.

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

```
Error: module "myawesomewindowsvm": missing required argument "name"
Error: module "myawesomewindowsvm": missing required argument "vm_size"
Error: module "myawesomewindowsvm": missing required argument "username"
Error: module "myawesomewindowsvm": missing required argument "password"
```
We have a problem! We didn't set required variables for our module.

Update the `main.tf` file:

```
module "myawesomewindowsvm" {
  source = "../../modules/my_virtual_machine"
name   = "awesomeapp"   vm_size  =
"Standard_A2_v2"   username =
"${var.username}"   password =
"${var.password}" }
```

Run `terraform init` again, this time there should not be any errors.

Terraform Plan

Run `terraform plan` and you should see your windows VM built from your module.
```
+
  module.myawesomewindowsvm.azurerm_resource_group.module
      id:                                 <computed>
      location:                           "uksouth"
      name:                               "awesomeapp-rg"
...
Plan: 6 to add, 0 to change, 0 to destroy.
```

2.0 Add Another Module

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

```
module "differentwindowsvm" {   source
= "../../modules/my_virtual_machine"   name
= "differentapp"   vm_size  =
"Standard_A2_v2"   username =
"${var.username}"   password =
"${var.password}" }
```

2.1 Scale a single module

Set the count of your first module to 2 and re-run a plan.

```
...
    vmcount = 2 ...
```

Run a plan and observe that your first module can scale independently of the second one.

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.module
      id:                                 <computed>
      location:                           "uksouth"
      name:                               "awesomeapp-rg"
...

  + module.differentlinuxvm.azurerm_resource_group.module
      id:                                 <computed>
      location:                           "uksouith"
      name:                               "differentapp-rg"
...

Plan: ?? to add, 0 to change, 0 to destroy.  How many resources will be created ?
```

2.2 More Variables

In your `environments/dev/main.tf` file we can see some duplication and secrets we do not want to store in configuration.
Add two variables to your **environment** **`main.tf`** file for username and password.
Create a new file and name it `terraform.tfvars` that will contain our secrets and automatically loaded when we run a `plan`.

```
username = "testadmin" password
= "Password1234!"
```

Terraform Plan

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

Advanced areas to explore

1. Use environment variables to load your secrets.
2. Add a reference to the Public Terraform Module for Azure Compute

Resources

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