Assignment
Automation with Terraform

Assignment Overview

This assignment provides learners with a real-world challenging scenario of codifying infrastructure and provisioning it in the Azure cloud. The assignment challenges them to apply their Terraform and Microsoft Azure knowledge and skills gained from the course to build a highly available, scalable, and secure infrastructure using a variety of Azure services, resources, and features across multiple availability zones. Learners are expected to use a development platform of their choice—Visual Studio Code, WSL (Windows Subsystem for Linux), GitBash, or a VM installed locally on their own laptop computers—to complete the assignment. They need to frequently access the Azure portal.

Technical Skills Assessed

Terraform skills assessed in the assignment include:

- Code development for a variety of Azure resources
- Management of single and multiple instances of infrastructure resources
- Parameterization via input variables
- Parameterization via locals block
- Parameterization via output values
- Modular architecture
- The use of the null provisioner
- Configuration and use of remote backend

Azure services, resources, and features assessed in the assignment include:

- Identity and Access (Microsoft Entra ID)
- Networking (virtual networks, subnets, etc.)
- Security (network security groups, etc.)
- Storage (storage accounts, Azure Files, blob containers, etc.)
- Compute (Windows and Linux virtual machines, managed disks)
- Network Traffic Management (load balancers)
- Monitoring (Microsoft Monitor)
- Cost Management (cost analysis)
- Azure Interfaces (Portal, AZ CLI)

Soft Skills Assessed

- Critical thinking
- Researching
- Troubleshooting
- Problem-solving
- Decision-making
- Self-judgement
- Organization
- Stress management
- Independent and collaborative working

Assignment Requirements

Use a free-tier or pay-as-you-go (preferred) Azure account to accomplish the assignment deliverables.

Instructions

1. Create a repo in GitHub to store code
2. You should be able to run the code from your automation VM
3. Select 1 CPU VM size (B1ms), LRS storage SKU, and cheapest DB options
4. Use logical names for all your resources. Prepend all resource names with the last 4 digits of your Humber ID to ensure uniqueness.
5. Parametrize Terraform configuration as much as possible
6. Store Terraform state information in Azure backend
7. Naming for root module files: providers.tf, backend.tf, main.tf, and outputs.tf
8. Naming for child module files: main.tf, variables.tf, outputs.tf, and provisioner.tf
9. Hardcode values in child modules that you do not expect to change often
10. Use your knowledge and comprehension to make configuration choice decisions where enough information is not provided
11. Shut down the VMs when not in use to save cost
12. Use the following tags for all your resources:

```
Assignment      = "CCGC 5502 Automation Assignment"
Name            = "firstname.lastname"
ExpirationDate  = "2024-12-31"
Environment     = "Learning"
```

Assignment Details

Phase I - Development

Develop a Child Module for Resource Group:

Develop a Terraform child module called **rgroup-*HumberID*** to provision 1 resource group called ***HumberID*-RG**. This module should return the name of the resource group to the root module.

Develop a Child Module for Networking:

Develop a Terraform child module called **network-*HumberID*** to provision 1 virtual network called ***HumberID*-VNET** along with 1 subnet called ***HumberID*-SUBNET** in ***HumberID*-RG** resource group. Add a network security group with 4 inbound access rules to allow traffic over ports 22, 3389, 5985, and 80. This network security group must be associated with the subnet. This module should return the names of the virtual network and the subnet to the root module.

Develop a Child Module for Common Services:

Develop a Terraform child module called **common-*HumberID*** to provision 1 log analytics workspace, 1 recovery services vault, and 1 standard storage account with LRS redundancy. This storage account **MUST** be different from the one with Terraform backend located. This module should return the names of the three resources to the root module.

Develop a Child Module for Linux Virtual Machines:

Develop a Terraform child module called **vmlinux-*HumberID*** to provision 3 CentOS 8.2 Linux VMs with public IP addresses created in 1 availability set (use for_each to ensure the code is scalable). Each VM must use the storage account created above for VM boot diagnostics. Each VM must have a unique DNS label assigned. The VMs must have 2 extensions installed: (1) Network Watcher extension [publisher: Microsoft.Azure.NetworkWatcher; name: NetworkWatcherAgentLinux; version 1.0] and (2) Azure Monitor extension [publisher: Microsoft.Azure.Monitor; name: AzureMonitorLinuxAgent; version 1.0]. Use the remote-exec null provisioner to display the hostnames of all 3 VMs. This module must return the hostnames, domain names, private IP addresses, and public IP addresses of the VMs to the root module.

Develop a Child Module for Windows Virtual Machines:

Develop a Terraform child module called **vmwindows-*HumberID*** to provision 1 Windows Server 2016 VM with public IP address created in 1 availability set (use count to ensure the code is scalable). The VM must have Antimalware extension installed. The VM must use the storage account created above for VM boot diagnostics. The VM must have a unique DNS label assigned. This module must return the hostname, domain name, private IP address, and public IP address of the VM to the root module.

Develop a Child Module for Data Disks:

Develop a Terraform child module called **datadisk-*HumberID*** to provision 4 x 10GB disks and attach them to the 4 VMs.

Develop a Child Module for Load Balancer:

Develop a Terraform child module called **loadbalancer-*HumberID*** to provision 1 public-facing basic load balancer with all 3 Linux VMs behind it. This module should return the name of the load balancer to the root module.

Develop a Child Module for Database:

Develop a Terraform child module called **database-*HumberID*** to provision 1 Azure DB for PostgreSQL Single Server instance. This module should return the name of the DB instance to the root module.

Develop the Root Module:

Develop a Terraform root module called **assignment1-*HumberID*** and define all child modules in it. This module should print on the screen the outputs received from child modules on a successful deployment.