Project

Automation with Terraform + Ansible

Marks: 21

Project Overview

This project provides learners with a real-world challenging scenario of codifying both infrastructure and configuration in the Azure cloud. The project challenges them to apply their Terraform, Ansible, and Microsoft Azure knowledge and skills gained from the course to build a highly available, scalable, and secure infrastructure in Azure and configure the guest operating systems. Learners may use any development platform to automate the deployment of resources and configuration of operating systems in Azure.

Technical Skills Assessed

Terraform skills assessed in the project 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

Ansible skills assessed in the project include:

- Code development of parameterized roles
- Decision making, iteration, and handlers
- File manipulation
- Call multiple roles from a playbook

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

- Identity and Access (Azure Active Directory)
- Networking (virtual networks, subnets, peering, etc.)
- Security (network security groups, etc.)
- Storage (storage accounts, Azure Files, blob containers, etc.)
- Compute (Windows/Linux virtual machines, managed disks)

- Network Traffic Management (load balancers)
- Monitoring (Azure 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

Project Requirements

Use a free-tier or pay-as-you-go Azure account to accomplish the project deliverables.

Recommendations

Implement the project in 1 Azure region that has availability zones (Australia Central, UK West, Canada Central, etc.). If any of these regions show VM sizes as unavailable, search for other regions that do not have that issue and use those instead.

Deliverables

1. Take screenshots or short videos, or a combination, where indicated, and add them to a Word document in the order in which they are requested along with an appropriate heading.
2. Attach the Word document under **Automation Project** in Blackboard and submit.

Warning

Copying and pasting the work of other students and submitting it as one's own is a serious academic misconduct. All involved parties will get a **ZERO**, no exceptions.

Feedback

We look forward to receiving your constructive feedback in terms of project flow, errors encountered, fixes you applied to make things functional, and so on. Please record your comments where they apply in your project document.

Rubric and Passing Marks

The following rubric will be used to grant marks to the project work. Each task under Project Details has maximum marks shown in parentheses. The Hands-On Implementation is worth **74** marks and the Presentation **10** marks.

Hands-On Implementation	**Max Marks**
Terraform Code from Assignment	8
Must be working	8
Parameterization	8
Heavy input parameterization	8
Medium input parameterization	6
Light input parameterization	3
Terraform/Ansible Integration with Provisioner	12
Provisioner executes Ansible playbook successfully	12
User Role	6
Log in with a new user account without entering password/passphrase	3
All user accounts and groups created successfully	3
Profile Role	6
The system-wide profile file updated successfully	6
Data Disk Role	10
Disks mounted to Linux VMs	10
Web Server Functionality	14
All web servers behind the load balancer work as expected	14
100% Non-Interactive and Flawless Provisioning	10
	10
Sub-Total	**74**
Presentation	
	10
Sub-Total	**10**
GRAND TOTAL	**84**
Note: The final marks will be divided by 4 as the maximum for the project are 21.	

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. Create all resources in a single resource group
6. Parametrize Terraform and Ansible configuration as much as possible
7. Store Terraform state information in Azure backend
8. Naming for root module files: providers.tf, backend.tf, main.tf, and outputs.tf
9. Naming for child module files: main.tf, variables.tf, outputs.tf, and provisioner.tf
10. Hardcode values in child modules that you do not expect to change often
11. Use your knowledge and comprehension to make configuration choice decisions where enough information is not provided
12. Shut down the VMs when not in use to save cost
13. Use the following tags for all your resources:

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

Project Details

Phase I - Development

Develop a Role for Disk Configuration:

Develop a parameterized Ansible role called **datadisk-*HumberID*** to partition the 10 GB data disks as follows:

a. 1 x 4 GB partition initialized with XFS and persistently mounted on **/part1**
b. 1 x 5 GB partition initialized with EXT4 and persistently mounted on **/part2**

Develop a Role for File Update:

Develop an Ansible role called **profile-*HumberID*** to append the following two lines to the **/etc/profile** file:

"#Test block added by Ansible……<Your username>"
export TMOUT=1500

Develop a Role for User/Group Creation:

Develop a parametrized Ansible role called **user-*HumberID*** to accomplish the following:

a. Add a group called **cloudadmins**
b. Add 3 user accounts called **user100**, **user200**, and **user300**
c. Add all three users to **cloudadmins** and **wheel** groups
d. Generate SSH keys for all three users but **without** a passphrase (the user module)
e. Distribute SSH keys for all three users (the authorized_key module)

Develop a Role to Configure a Load-Balanced Website:

Develop a parametrized Ansible role called **webserver-*HumberID*** to perform the following:

a. Install and configure Apache web server software
b. Create files on the automation server called **vm1.html**, **vm2.html**, and **vm3.html** containing the FQDN of the respective node
c. Copy the files as **index.html** to their respective node under the **/var/www/html** directory
d. Set permissions on the files to 0444
e. Start the Apache web server (via **handlers**)
f. Ensure that the Apache web service automatically starts on subsequent system reboots

Develop Playbook:

Develop a playbook called ***HumberID*-playbook.yml** and list all the roles in it. This playbook must run against all **Linux** inventory nodes.

Update the Terraform Code:

Make a copy of the Terraform assignment code. Update the null_resource provisioner to **automatically** execute the Ansible playbook ***HumberID*-playbook.yml** against all Linux inventory nodes.

Phase II – Pre-Provisioning Validation

1. Run **terraform init** to initialize the plugins and backend. **SCREENSHOT**
2. Run **terraform validate** to confirm there are no typos and syntax errors. **SCREENSHOT**
3. Run **terraform plan** and review the entire plan prior to deployment.

Phase III – Provisioning

1. The entire configuration as code (Ansible) developed above must be provisioned flawlessly and non-interactively with the infrastructure as code (Terraform) when **terraform apply --auto-approve** is issued.

VIDEO of the entire deployment process

Phase IV – Post-Provisioning Validation

1. Run the **terraform state list | nl** command. This should show **exactly 48 lines** in the output.

SCREENSHOT showing the entire output

2. Run the **terraform output** command.

SCREENSHOT showing the entire output

Phase V – Configuration Validation

Perform the following tests **ONLY** after the above has been completed **SUCCESSFULLY**

1. Log in as **user100** to one of the managed Linux nodes using the **ssh** command or **PuTTY**. You **must not** be prompted for password/passphrase. Run the command **tail -3 /etc/passwd**. Run the command **grep -E 'cloudadmins|wheel'**. Run the command **tail -3 /etc/profile**. Run the command **df -Th**.

VIDEO of the entire process

2. Enter the **FQDN** of the load balancer in a browser window using the HTTP protocol and click the **Refresh** button every 7 seconds for a few times. You should see the FQDN of a different managed node appearing on each click.

VIDEO of the entire process

Phase VI – Submission

1. Share the GitHub project repo with the instructor
2. Upload the videos and screenshots to a cloud drive and share the location with the instructor

End of Project

DESTROY THE INFRASTRUCTURE AFTER YOUR PROJECT HAS BEEN MARKED TO SAVE COST

DO NOT REMOVE THE TERRAFORM AND THE ANSIBLE CODE