Lab 05

Instructions:

1. Paste all screenshots (highlighted in red) in a single Word document in the correct order
2. Name the document as **YourName-lab05**
3. Submit the document as an attachment in Bb under Labs
4. Use a WSL terminal for all activities

Lab submissions must be made by the due date (as indicated on the Critical Path). Each day thereafter will incur a **10%** deduction from the earned marks, up to a maximum of **3 days**. Submissions beyond this deadline will receive a grade of **Zero**.

Lab Objectives:

There is one section in this lab as described below:

Section 1: Use a provisioner to automate operating system configuration
Section 2: Configure Azure backend to store Terraform state information

---

WARNING

Code generated by ChatGPT or a similar generative AI tool, and copied and pasted without making the **right** modifications will result in a **ZERO** for that **entire section**.

Section 1

Objectives:

- Use a null_resource provisioner to make simple changes in guest Linux operating system
- Validate, deploy, expand, analyze, and destroy infrastructure

Part 1: Prepare for the lab:

1. Copy folder **lab04s1** as **lab05s1**
2. Change into **lab05s1**
3. Create an empty file called **linux_provisioner.tf**

Part 2: Update linux_provisioner.tf file:

4. Define a single null_resource resource block to use the **remote-exec** provisioner to display the hostname of all provisioned Linux virtual machines. This provisioner must run only after the Linux VMs have been deployed (depends_on). Use either count or for_each meta-argument.

SCREENSHOT of the linux_provisioner.tf file

Part 3: Validate configuration:

5. Validate the configuration to ensure there are no errors or typos in the file (**terraform validate**)
6. Fix any issues in the Terraform files if reported
7. Re-run the validation until no errors are reported (**terraform validate**)

Part 4: Run simulation:

8. Perform a dry run (**terraform plan**)
9. Review output and ensure all configuration is as per requirements. Observe the resources with +, -, or -/+ signs.
10. Fix any issues in the Terraform files if reported
11. Redo the dry run until no errors are reported (**terraform plan**)

Part 5: Deploy infrastructure:

12. Deploy the infrastructure and monitor progress (**terraform apply**)

SCREENSHOT of the output that shows the execution and results of the provisioners

Part 6: Get information from Terraform state:

13. View and analyze state information (**terraform state list | nl**) **SCREENSHOT**
14. Display the output values (**terraform output**) **SCREENSHOT**

Part 7: Destroy all resources and verify:

15. Destroy all the resources (**terraform destroy**)
16. Verify deletion (**terraform state list | nl**)
17. View the content of terraform.tfstate file (**tail -20 terraform.tfstate**)

========== End of Section 1 ==========

Section 2

Objectives:

- Configure Azure blob container to store Terraform state information for security and team collaboration
- Validate, deploy, expand, analyze, and destroy infrastructure

Part 1: Prepare for the lab:

1. Copy folder **lab05s1** as **lab05s2**
2. Change into **lab05s2**

Part 2: Configure backend:

3. Create a separate resource group called **tfstate<HumberID>RG** to store TF state information:

$ **az group create -l canadacentral -n tfstate<HumberID>RG**

4. Create a storage account called **tfstate<HumberID>sa** using standard LRS in the resource group:

$ **az storage account create -l canadacentral -n tfstate<HumberID>sa -g tfstate<HumberID>RG --sku Standard_LRS**

5. Obtain the storage account access key:

$ **az storage account keys list -g tfstate<HumberID>RG -n tfstate<HumberID>sa**

6. Create a blob container called **tfstatefiles** in the storage account:

$ **az storage container create -n tfstatefiles --account-name tfstate<HumberID>sa --account-key "<access-key-from-previous-step>"**

7. Log in to the Azure Portal. Navigate to the TF state resource group | Storage Account | Containers to verify the creation of the account and blob container. **SCREENSHOT**

Part 3: Configure access to the backend:

8. Create a file called **backend.tf** in the root module and define a backend block
SCREENSHOT of the backend.tf file
9. Open **~/.profile** file and append the following line:

export ARM_ACCESS_KEY=**"<access-key-from-previous-step>"**

10. Activate the new variable, and confirm:

$ **. ~/.profile ; env | grep ARM**

The output should display all five environment variables.

11. Execute **terraform init** and enter **yes** to move the local state information to the new remote backend (if prompted)

Note: Remove the **.terraform** directory from the root module if you see an authentication error and re-run **terraform init**

Part 4: Validate configuration:

12. Validate the configuration to ensure there are no errors or typos in the file (**terraform validate**)
13. Fix any issues in the Terraform files if reported
14. Re-run the validation until no errors are reported (**terraform validate**)

Part 5: Format configuration:

15. Format all Terraform configuration files (**terraform fmt -recursive**)

Part 6: Run simulation:

16. Perform a dry run (**terraform plan**)
17. Review output and ensure all configuration is as per requirements. Observe the resources with +, -, or -/+ signs.
18. Fix any issues in the Terraform files if reported
19. Redo the dry run until no errors are reported (**terraform plan**)

Part 7: Deploy infrastructure:

20. Deploy the infrastructure and monitor progress (**terraform apply**)

Part 8: Confirm backend update in Azure:

21. Log in to the Azure Portal. Navigate to the TF state resource group | Storage Account | Containers. Click on the key (file) and then on Edit to view the TF state information. **SCREENSHOT**

Part 9: Get information from Terraform state:

22. View and analyze state information (**terraform state list | nl**) **SCREENSHOT**
23. Display the output values (**terraform output**) **SCREENSHOT**

Part 10: Destroy all resources and verify:

24. Destroy all the resources (**terraform destroy**)

25. Verify deletion (**terraform state list | nl**)

Do Not Remove the Resource Group that contains the Backend

========== End of Section 2 ==========