QATIP Intermediate
Azure Lab05
Implementing Checks and Validations

Contents

Overview

This lab is designed to help you...

By the end of this exercise, you will:

- Apply **input validation rules** in Terraform configurations.
- Implement **precondition and postcondition checks** to enforce constraints.

- Introduce **Azure Policy restrictions** on **resource group names and VM names**.
- Implement **post-deployment validation checks** to ensure compliance.

Before you begin

Ensure you have completed Lab0 before attempting this lab.

In the IDE terminal pane, enter the following command...

```
cd c:\azure-tf-int\lab\05
```

This shifts your current working directory to labs\05. Ensure all commands are executed in this directory

Close any open files and use the Explorer pane to navigate to and open the labs\05 folder.

Reference Documentation

- Terraform Input Validation
- Terraform Preconditions and Postconditions
- Azure Resource Naming Rules
- Azure Policy

Scenario

You are working as a cloud engineer responsible for enforcing naming and policy constraints in Terraform deployments. You need to deploy Azure resources whilst enforcing naming and policy constraints using:

- **Preconditions** (e.g., names must follow conventions)
- **Postconditions** (e.g., verifying replication type & location)
- **Azure Policy Constraints** (e.g., VM names must be VM1–VM6)
- **Post-Deployment Terraform Data Checks**

Step 1: Review Terraform Files

main.tf (Terraform Configuration)

variables.tf (Input Variables)

terraform.tfvars (Default Values)

Step 2: Check Azure Policy Compliance

Before running the Terraform deployment, verify if any Azure policies affect the resources.

1. **List Azure Policies Assigned to Your Subscription**

   az policy assignment list --query "[].{Name:displayName, Scope:scope}" --output table

   This command shows all policies assigned at the subscription level.

2. **Check Policies Assigned to Your Resource Group**

   az policy assignment list --scope /subscriptions/YOUR_SUBSCRIPTION_ID/resourceGroups/RG1 --query "[].{Name:displayName, EnforcementMode:enforcementMode}" --output table

   This command checks which policies are assigned to the resource group you're deploying into.

3. **Check Policy Compliance for Your Deployed Resources** After deploying resources, verify if they comply with existing Azure Policies:

   az policy state list --query "[].{Policy:policyDefinitionName, Resource:resourceId, Compliance:complianceState}" --output table

   This helps determine whether resources are compliant or if Azure enforces restrictions beyond Terraform's validation checks.

4. **Expected Outcome:**

If policies enforce naming conventions, Terraform deployments violating them should fail.

If post-deployment compliance checks fail, Azure policies may need adjustments or Terraform configurations must be revised.

Step 3: Deploy the configuration

terraform init
terraform plan
terraform apply
terraform output

Expected Results (your IP address will differ):

```
admin_username = <sensitive>
nic_name = "lab-nic"
resource_group_name = "RG1"
route_table_name = "lab-route-table"
security_group_name = "lab_SecurityGroup"
storage_account_name = "stgvalidaccount"
storage_account_replication = "LRS"
subnet_name = "lab-subnet"
virtual_network_name = "lab-vnet"
vm_name = "VM1"
vm_public_ip = "52.174.6.235"
```

Step 4: Test Validation Failures

Test 1: Invalid Resource Group Name

Modify terraform.tfvars: **resource_group_name = "RG7"**

Run **terraform apply**

Expected error: Resource group name must be RG1 through RG6

Test 2: Incorrect Storage Account Replication

Modify main.tf: **account_replication_type = "GRS"**

Run: terraform apply

Expected error: Storage account replication type is incorrect. Expected 'LRS'.

Test 3: Invalid VM Name

Modify terraform.tfvars: **vm_name = "VM10"**

Run: **terraform apply**

Expected error: **VM name must be between VM1 and VM6**

Step 5: Try Yourself - Implementing VM Size Control

Objective:

Enhance the Terraform configuration to enforce constraints on VM sizes, ensuring only allowed VM sizes are used.

Requirements:

- Modify variables.tf to introduce a **validated** vm_size variable that restricts the VM sizes to only a predefined set (e.g., Standard_B2s, Standard_D2s_v3).
- Update **terraform.tfvars** to use an **allowed** VM size.
- Modify main.tf to reference the validated **vm_size** variable.
- Run terraform apply with both **valid** and **invalid** sizes to test validation.

Step 6: Lab Clean Up

- Remove all deployed resource using **terraform destroy**

Congratulations you have completed this lab