**目标**

1. 使用 Terraform 创建一个资源组和 NSG，配置防火墙规则。
2. 使用 Conftest 强制执行安全策略（限制开放防火墙规则和强制标签）。
3. 集成到 GitHub Actions 流水线，确保自动化测试。
4. 在本地 macOS 环境（VS Code）中运行和测试。
5. 提供完整的 Terraform 和 Conftest 配置文件，满足演示要求。

**步骤指导**

**1. 确认环境设置**

您已经在 macOS 上通过 az login 登录到 Azure，并选择了 **Azure for Students** 订阅。确保以下工具已安装（根据 README）：

* **Terraform**：版本 1.6.6（运行 terraform -v 确认）。
* **Conftest**：版本 0.45.0（运行 conftest --version 确认）。
* **Azure CLI**：已安装并登录（运行 az account show 确认当前订阅）。
* **Git**：用于版本控制和推送到 GitHub。

运行以下命令确认订阅：

bash

CollapseWrapRun

Copy

az account show

**预期输出**（部分）：

json

CollapseWrap

Copy

{

"name": "Azure for Students",

"id": "127e7a44-d802-42e4-b654-a434382666ac",

"tenantId": "<tenant-id>",

...

}

如果输出不正确，设置订阅：

bash

CollapseWrapRun

Copy

az account set --subscription "127e7a44-d802-42e4-b654-a434382666ac"

**2. 创建 Terraform 配置**

在您的项目目录（例如 OPA 目录）中，创建或更新 main.tf 文件，定义资源组和 NSG。以下是一个示例配置，包含一个资源组和一个 NSG，带有防火墙规则：

main.tf：

terraform {

required\_providers {

azurerm = {

source = "hashicorp/azurerm"

version = "~> 3.0"

}

}

}

provider "azurerm" {

features {}

subscription\_id = "127e7a44-d802-42e4-b654-a434382666ac" # Azure for Students 订阅 ID

}

resource "azurerm\_resource\_group" "example" {

name = "example-rg"

location = "East US"

tags = {

Environment = "Production"

Owner = "shaoxianduan"

}

}

resource "azurerm\_network\_security\_group" "example" {

name = "example-nsg"

location = azurerm\_resource\_group.example.location

resource\_group\_name = azurerm\_resource\_group.example.name

security\_rule {

name = "allow-ssh"

priority = 100

direction = "Inbound"

access = "Allow"

protocol = "Tcp"

source\_port\_range = "\*"

destination\_port\_range = "22"

source\_address\_prefix = "0.0.0.0/0" # 开放规则，用于测试策略失败

destination\_address\_prefix = "\*"

}

tags = {

Environment = "Production"

Owner = "shaoxianduan"

}

}

**说明**：

* **订阅 ID**：在 provider "azurerm" 中明确指定了您的 **Azure for Students** 订阅 ID。
* **资源组**：创建名为 example-rg 的资源组，位于 East US。
* **NSG**：创建名为 example-nsg 的网络安全组，包含一个允许 SSH（端口 22）的入站规则，source\_address\_prefix = "0.0.0.0/0" 是一个不安全的配置，稍后会通过 Conftest 策略检测到。
* **标签**：为资源组和 NSG 添加了 Environment 和 Owner 标签，满足演示要求的标签强制。

**3. 初始化 Terraform**

在包含 main.tf 的目录中，运行以下命令以初始化 Terraform 并下载 Azure 提供者插件：

bash

CollapseWrapRun

Copy

terraform init

**预期输出**（部分）：

text

CollapseWrap

Copy

Initializing provider plugins...

- Finding hashicorp/azurerm versions matching "~> 3.0"...

- Installing hashicorp/azurerm v3.x.x...

- Installed hashicorp/azurerm v3.x.x...

Terraform has been successfully initialized!

**4. 生成 Terraform 计划**

生成 Terraform 计划并转换为 JSON 格式，用于 Conftest 测试：

terraform plan -out=tfplan

terraform show -json tfplan > tfplan.json

terraform apply "tfplan"

**5. 创建 Conftest 策略**

在项目目录中，创建 policies 目录（如果不存在）：

bash

CollapseWrapRun

Copy

mkdir -p policies

在 policies 目录中，创建 policy.rego 文件，包含以下策略：

* **限制开放防火墙规则**：禁止 NSG 的入站规则使用 0.0.0.0/0，除非是特定端口（如 443）。
* **强制标签**：确保资源包含 Environment 和 Owner 标签。

policy.rego：

package main

# 策略 1：禁止过于开放的防火墙规则

deny[msg] {

resource := input.resource\_changes[\_]

resource.type == "azurerm\_network\_security\_group"

rule := resource.change.after.security\_rule[\_]

rule.direction == "Inbound"

rule.source\_address\_prefix == "0.0.0.0/0"

rule.destination\_port\_range != "443"

msg := sprintf("NSG '%s' contains an overly permissive inbound rule allowing '0.0.0.0/0' on port '%s'.", [resource.change.after.name, rule.destination\_port\_range])

}

# 策略 2：强制 Environment 标签

deny[msg] {

resource := input.resource\_changes[\_]

allowed\_types := ["azurerm\_resource\_group", "azurerm\_network\_security\_group"]

resource.type == allowed\_types[\_]

not resource.change.after.tags["Environment"]

msg := sprintf("Resource '%s' is missing required tag 'Environment'.", [resource.change.after.name])

}

# 策略 3：强制 Owner 标签

deny[msg] {

resource := input.resource\_changes[\_]

allowed\_types := ["azurerm\_resource\_group", "azurerm\_network\_security\_group"]

resource.type == allowed\_types[\_]

not resource.change.after.tags["Owner"]

msg := sprintf("Resource '%s' is missing required tag 'Owner'.", [resource.change.after.name])

}

**说明**：

* **防火墙规则策略**：检测 NSG 的入站规则，如果 source\_address\_prefix 是 0.0.0.0/0 且端口不是 443（HTTPS），则拒绝。
* **标签策略**：确保 azurerm\_resource\_group 和 azurerm\_network\_security\_group 包含 Environment 和 Owner 标签。
* 这些策略满足演示要求的“限制开放防火墙规则”和“标签强制”。

**6. 本地运行 Conftest 测试**

测试 Terraform 计划是否符合策略：

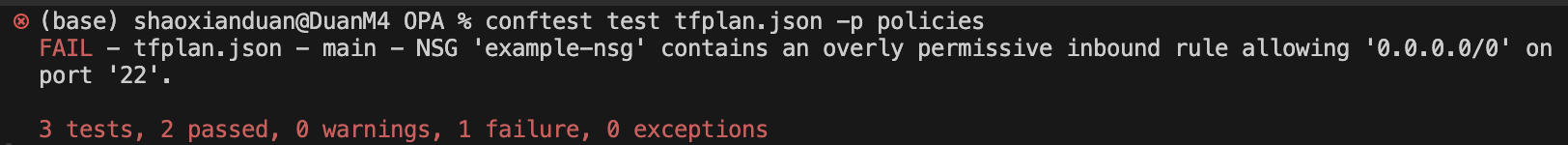
conftest test tfplan.json -p policies

**预期输出**： 由于 main.tf 中的 NSG 规则允许 0.0.0.0/0 访问端口 22（SSH），测试将失败：

(base) shaoxianduan@DuanM4 OPA % conftest test tfplan.json -p policies

FAIL - tfplan.json - main - NSG 'example-nsg' contains an overly permissive inbound rule allowing '0.0.0.0/0' on port '22'.

3 tests, 2 passed, 0 warnings, 1 failure, 0 exceptions

****

**7. 修改 Terraform 配置以通过策略**

为了通过策略，更新 main.tf，将 NSG 规则的 source\_address\_prefix 限制为特定 IP 范围，并确保端口为允许的端口（例如 443）。以下是更新后的 main.tf：

main.tf：

terraform {

required\_providers {

azurerm = {

source = "hashicorp/azurerm"

version = "~> 3.0"

}

}

}

provider "azurerm" {

features {}

subscription\_id = "127e7a44-d802-42e4-b654-a434382666ac" # Azure for Students 订阅 ID

}

resource "azurerm\_resource\_group" "example" {

name = "example-rg"

location = "East US"

tags = {

Environment = "Production"

Owner = "shaoxianduan"

}

}

resource "azurerm\_network\_security\_group" "example" {

name = "example-nsg"

location = azurerm\_resource\_group.example.location

resource\_group\_name = azurerm\_resource\_group.example.name

security\_rule {

name = "allow-https"

priority = 100

direction = "Inbound"

access = "Allow"

protocol = "Tcp"

source\_port\_range = "\*"

destination\_port\_range = "443" # 更改为允许的端口

source\_address\_prefix = "10.0.0.0/16" # 限制源 IP 范围

destination\_address\_prefix = "\*"

}

tags = {

Environment = "Production"

Owner = "shaoxianduan"

}

}

重新生成计划并测试：

bash

CollapseWrapRun

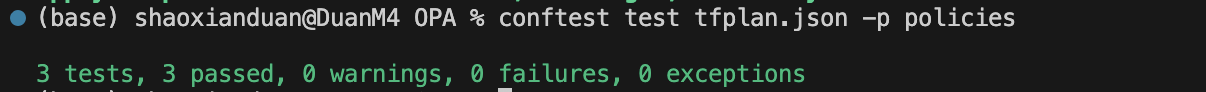
Copy

terraform plan -out=tfplan

terraform show -json tfplan > tfplan.json

conftest test tfplan.json -p policies

**预期输出**：

 **8. 应用 Terraform 配置（可选）**

如果您想在 Azure 中实际创建资源组和 NSG，运行以下命令：

bash

CollapseWrapRun

Copy

terraform apply tfplan

**注意**：

* **Azure for Students** 订阅可能有资源限制（例如 CPU 配额或区域限制）。如果遇到错误（如配额限制），尝试更改 location（例如 West US）或联系订阅管理员。
* 确保您有权限在订阅中创建资源。如果权限不足，可能需要联系 Algonquin College 的 IT 管理员。

**9. 集成到 GitHub Actions**

更新 GitHub Actions 工作流（.github/workflows/terraform-conftest.yml），以支持 Azure 身份验证和策略测试。以下是工作流文件：

terraform-conftest.yml：

name: Terraform and Conftest CI

on:

push:

branches:

- main

pull\_request:

branches:

- main

jobs:

terraform-conftest:

name: Terraform Plan and Conftest Test

runs-on: ubuntu-latest

steps:

# 检出代码

- name: Checkout code

uses: actions/checkout@v3

# 安装 Terraform

- name: Setup Terraform

uses: hashicorp/setup-terraform@v2

with:

terraform\_version: 1.6.6

# 安装 Conftest

- name: Install Conftest

run: |

curl -LO https://github.com/open-policy-agent/conftest/releases/download/v0.45.0/conftest\_0.45.0\_Linux\_x86\_64.tar.gz

tar -xzf conftest\_0.45.0\_Linux\_x86\_64.tar.gz

sudo mv conftest /usr/local/bin/

conftest --version

# 安装 Azure CLI

- name: Install Azure CLI

run: |

curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash

# Azure 登录

- name: Azure Login

uses: azure/login@v1

with:

creds: ${{ secrets.AZURE\_CREDENTIALS }}

# 初始化 Terraform

- name: Terraform Init

run: terraform init

# 生成 Terraform 计划

- name: Terraform Plan

id: plan

run: terraform plan -out=tfplan -no-color

# 转换为 JSON 格式

- name: Convert Terraform Plan to JSON

run: terraform show -json tfplan > tfplan.json

# 运行 Conftest 测试

- name: Conftest Test

run: conftest test tfplan.json -p policies

**设置 Azure 凭据**：

1. 创建服务主体（如果尚未创建）：

bash

CollapseWrapRun

Copy

az ad sp create-for-rbac --name "TerraformSP" --role Contributor --scopes /subscriptions/127e7a44-d802-42e4-b654-a434382666ac

保存输出的 appId、password、tenant 和 subscriptionId。

1. 在 GitHub 仓库的 **Settings > Secrets and variables > Actions** 中，添加一个名为 AZURE\_CREDENTIALS 的秘密，内容为：

json

CollapseWrap

Copy

{

"clientId": "<appId>",

"clientSecret": "<password>",

"subscriptionId": "127e7a44-d802-42e4-b654-a434382666ac",

"tenantId": "<tenant>"

}

**10. 推送到 GitHub 并验证**

将代码推送到 GitHub 仓库：

bash

CollapseWrapRun

Copy

git add .

git commit -m "Add Azure resource group, NSG, and Conftest policies"

git push origin main

访问 GitHub 仓库（例如 https://github.com/shaoxian423/8922Terraform），转到 **Actions** 选项卡，检查工作流运行情况。如果策略通过，工作流将显示 **PASS**；否则，将显示失败原因。

**满足演示要求**

* **强制安全策略**：
  + policy.rego 限制了 0.0.0.0/0 的入站规则，仅允许端口 443，防止不安全的防火墙配置。
  + 标签策略确保资源包含 Environment 和 Owner 标签，符合合规性要求。
* **CI/CD 集成**：
  + GitHub Actions 工作流集成了 Terraform 计划生成、Conftest 测试和 Azure 身份验证，自动化了策略检查。
* **示例**：
  + NSG 示例展示了如何检测和修复过于开放的防火墙规则。
  + 标签强制示例确保资源合规性。

**故障排除**

1. **Azure 权限问题**：
   * 如果 terraform apply 或 GitHub Actions 报告权限错误，检查服务主体的角色（需要 Contributor 或类似权限）。联系 Algonquin College 的管理员以获取帮助。
2. **配额限制**：
   * **Azure for Students** 订阅可能限制资源创建。运行 az vm list-skus --location eastus --query "[].name" -o tsv 检查可用区域和资源类型。如果受限，尝试其他区域（如 westus）。
3. **Conftest 失败**：
   * 如果策略测试失败，检查 tfplan.json 和 policy.rego 的内容。确保规则逻辑正确（例如，检查 source\_address\_prefix 或 tags 的路径）。

**下一步（可选）**

* **扩展策略**：添加更多安全策略，例如：
  + 限制特定协议（仅允许 TCP）。
  + 要求 NSG 规则使用特定 IP 范围（例如公司 VPN）。
* **多环境支持**：使用 Terraform 变量支持不同环境（开发、测试、生产）。
* **监控和清理**：在测试完成后，运行 terraform destroy 删除资源，避免产生费用。

如果您遇到任何问题（例如错误信息）或需要进一步定制（例如添加更多防火墙规则），请提供详细信息，我会继续协助！