BÁO CÁO THỰC HÀNH

**Bài thực hành số 02: Quản lý và triển khai hạ tầng AWS và ứng dụng microservices với Terraform,CloudFormation, GitHub Actions, AWS CodePipeline và Jenkins**

**Môn học:** Công nghệ DevOps và Ứng dụng

**Lớp:** NT548.P11.MMCL

**THÀNH VIÊN THỰC HIỆN (Nhóm 11):**

|  |  |  |
| --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** |
| 1 | Lê Quốc Khánh | 21520978 |
| 2 | Nguyễn Văn Anh Tuấn | 21522757 |
| 3 | Đỗ Thế Danh | 21520685 |

|  |
| --- |
| **Điểm tự đánh giá** |
| **9** |

**ĐÁNH GIÁ KHÁC:**

|  |  |
| --- | --- |
| Tổng thời gian thực hiện | 21 ngày |
| Phân chia công việc | * Lê Quốc Khánh: Thưc hiện câu 2, hỗ trợ câu 1, viết báo cáo * Nguyễn Văn Anh Tuấn: Thực câu 1, hỗ trợ câu 3, hỗ trợ viết báo cáo * Đỗ Thế Danh: Thực hiện câu 3 |
| Ý kiến *(nếu có)*  + Khó khăn  + Đề xuất, kiến nghị | Không có |
| Link Github | <https://github.com/LQK164/NT548-DevOps-Exercises/tree/main/Lab2> |

Phần bên dưới của báo cáo này là báo cáo chi tiết của nhóm thực hiện

# BÁO CÁO CHI TIẾT

## **Triển khai hạ tầng AWS sử dụng Terraform và tự động hóa quy trình với GitHub Actions**

### **Dùng Terraform để triển khai các dịch vụ AWS bao gồm: VPC, Route Tables, NAT** **Gateway, EC2, Security Groups) đã thực hiện ở bài tập 1**

Sử dụng lại phần code Terraform như bài lab 1

Khai báo Provider là AWS

terraform {

required\_providers {

aws = {

source = "hashicorp/aws"

version = ">= 5.6"

}

}

required\_version = ">= 0.13"

}

provider "aws" {

region = "us-east-1"

}

Khai báo VPC

# Tạo VPC

resource "aws\_vpc" "my\_vpc" {

cidr\_block = "10.0.0.0/16"

tags = {

Name = "Nhom11-VPC"

}

}

Khai báo NAT Gateway và Internet Gateway

# Tạo Internet Gateway cho Public Subnet

resource "aws\_internet\_gateway" "igw" {

vpc\_id = aws\_vpc.my\_vpc.id

tags = {

Name = "Nhom11-IGW"

}

}

# Tạo Elastic IP cho NAT Gateway

resource "aws\_eip" "nat\_eip" {

domain = "vpc"

tags = {

Name = "Nhom11-ENG"

}

}

# Tạo NAT Gateway cho Private Subnet

resource "aws\_nat\_gateway" "nat\_gw" {

allocation\_id = aws\_eip.nat\_eip.id

subnet\_id = aws\_subnet.public\_subnet.id

tags = {

Name = "Nhom11-NGW"

}

}

Khai báo Public Route Table và Private Route Table

# Tạo Public Route Table và định tuyến qua Internet Gateway

resource "aws\_route\_table" "public\_rt" {

vpc\_id = aws\_vpc.my\_vpc.id

route {

cidr\_block = "0.0.0.0/0"

gateway\_id = aws\_internet\_gateway.igw.id

}

tags = {

Name = "Nhom11-PubRT"

}

}

# Tạo Private Route Table và định tuyến qua NAT Gateway

resource "aws\_route\_table" "private\_rt" {

vpc\_id = aws\_vpc.my\_vpc.id

route {

cidr\_block = "0.0.0.0/0"

nat\_gateway\_id = aws\_nat\_gateway.nat\_gw.id

}

tags = {

Name = "Nhom11-PriRT"

}

}

Khái báo Private Subnet và Public Subnet

# Tạo Public Subnet

resource "aws\_subnet" "public\_subnet" {

vpc\_id = aws\_vpc.my\_vpc.id

cidr\_block = "10.0.1.0/24"

availability\_zone = "us-east-1a"

map\_public\_ip\_on\_launch = true

tags = {

Name = "Nhom11-PubSub"

}

}

# Tạo Private Subnet

resource "aws\_subnet" "private\_subnet" {

vpc\_id = aws\_vpc.my\_vpc.id

cidr\_block = "10.0.2.0/24"

availability\_zone = "us-east-1a"

tags = {

Name = "Nhom11-PriSub"

}

}

Gán Private Subnet và Public Subnet vào các Route Table

# Gán Public Subnet vào Public Route Table

resource "aws\_route\_table\_association" "public\_rt\_assoc" {

subnet\_id = aws\_subnet.public\_subnet.id

route\_table\_id = aws\_route\_table.public\_rt.id

}

# Gán Private Subnet vào Private Route Table

resource "aws\_route\_table\_association" "private\_rt\_assoc" {

subnet\_id = aws\_subnet.private\_subnet.id

route\_table\_id = aws\_route\_table.private\_rt.id

}

Khai báo các EC2 Instance cho Private Subnet và Public Subnet

# Tạo EC2 instance trong Public Subnet

resource "aws\_instance" "public\_ec2" {

ami = "ami-0866a3c8686eaeeba"

instance\_type = "t2.micro"

subnet\_id = aws\_subnet.public\_subnet.id

security\_groups = [aws\_security\_group.public\_sg.id]

tags = {

Name = "Nhom11-PubEC2"

}

}

# Tạo EC2 instance trong Private Subnet

resource "aws\_instance" "private\_ec2" {

ami = "ami-0866a3c8686eaeeba"

instance\_type = "t2.micro"

subnet\_id = aws\_subnet.private\_subnet.id

security\_groups = [aws\_security\_group.private\_sg.id]

tags = {

Name = "Nhom11-PriEC2"

}

}

Khai báo các Secuirty Groups cho Public và Private EC2

# Tạo Security Group cho Public EC2

resource "aws\_security\_group" "public\_sg" {

vpc\_id = aws\_vpc.my\_vpc.id

ingress {

from\_port = 22

to\_port = 22

protocol = "tcp"

cidr\_blocks = ["123.250.165.100/32"]

}

egress {

from\_port = 0

to\_port = 0

protocol = "-1"

cidr\_blocks = ["0.0.0.0/0"]

}

tags = {

Name = "Nhom11-PubSecGR"

}

}

# Tạo Security Group cho Private EC2

resource "aws\_security\_group" "private\_sg" {

vpc\_id = aws\_vpc.my\_vpc.id

ingress {

from\_port = 22

to\_port = 22

protocol = "tcp"

security\_groups = [aws\_security\_group.public\_sg.id]

}

egress {

from\_port = 0

to\_port = 0

protocol = "-1"

cidr\_blocks = ["0.0.0.0/0"]

}

tags = {

Name = "Nhom11-PriSecGR"

}

}

### **Tự động hóa quá trình triển khai với GitHub Actions**

name: Terraform CI/CD

on:

push:

branches:

- main

pull\_request:

branches:

- main

jobs:

terraform:

name: 'Terraform Deploy'

runs-on: ubuntu-latest

env:

AWS\_REGION: 'us-east-1'

TF\_VERSION: '1.9.6'

steps:

- name: 'Checkout GitHub repository'

uses: actions/checkout@v2

- name: 'Configure AWS credentials'

uses: aws-actions/configure-aws-credentials@v2

with:

aws-access-key-id: ${{ secrets.AWS\_ACCESS\_KEY\_ID }}

aws-secret-access-key: ${{ secrets.AWS\_SECRET\_ACCESS\_KEY }}

aws-region: ${{ env.AWS\_REGION }}

- name: 'Set up Terraform'

uses: hashicorp/setup-terraform@v2

with:

terraform\_version: ${{ env.TF\_VERSION }}

- name: 'Terraform Init'

run: terraform init

- name: 'Terraform Format'

run: terraform fmt

- name: 'Terraform Validate'

run: terraform validate

- name: 'Terraform Plan'

run: terraform plan -out=tfplan

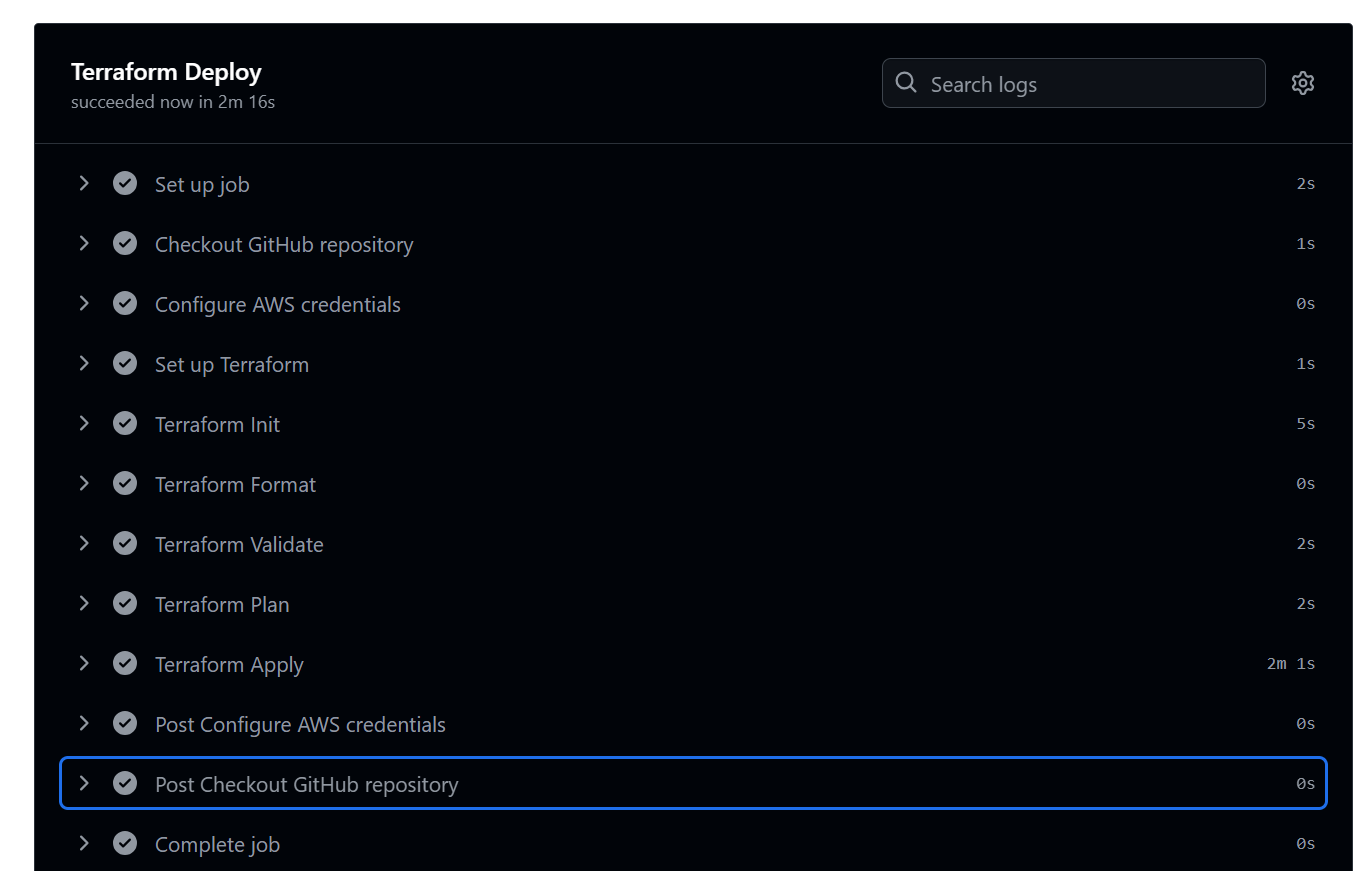
- name: 'Terraform Apply'

if: github.ref == 'refs/heads/main'

run: terraform apply -auto-approve tfplan

Giải thích Workflows:

* **Trigger:** Workflow được kích hoạt khi có push hoặc pull request đến nhánh main.
* **Checkout Repository:** Clone repository từ GitHub vào runner.
* **Configure AWS Credentials:** Cấu hình AWS credentials sử dụng secret được lưu trữ trên GitHub.
* **Set up Terraform:** Cài đặt phiên bản Terraform đã định cấu hình.
* **Terraform Init:** Khởi tạo working directory của Terraform.
* **Terraform Format:** Định dạng code Terraform theo chuẩn.
* **Terraform Validate:** Kiểm tra cú pháp của code Terraform.
* **Terraform Plan:** Tạo một execution plan, hiển thị các thay đổi sẽ được áp dụng.
* **Terraform Apply:** Áp dụng những bước cấu hình trước vào AWS.



1. Sử dụng Github Actions deploy Terraform lên AWS thành công

### **Tích hợp Checkov để kiểm tra tính tuân thủ và bảo mật của mã nguồn Terraform.**

Thêm phần tích hợp checkov và chạy để kiểm tra code

* **Install Checkov:** Tải Checkov và kiểm tra phiên bản hiện tại đang dùng.
* **Test Terraform code:** chạy checkov để kiểm tra file Terraform.

- name: 'Install Checkov'

run: |

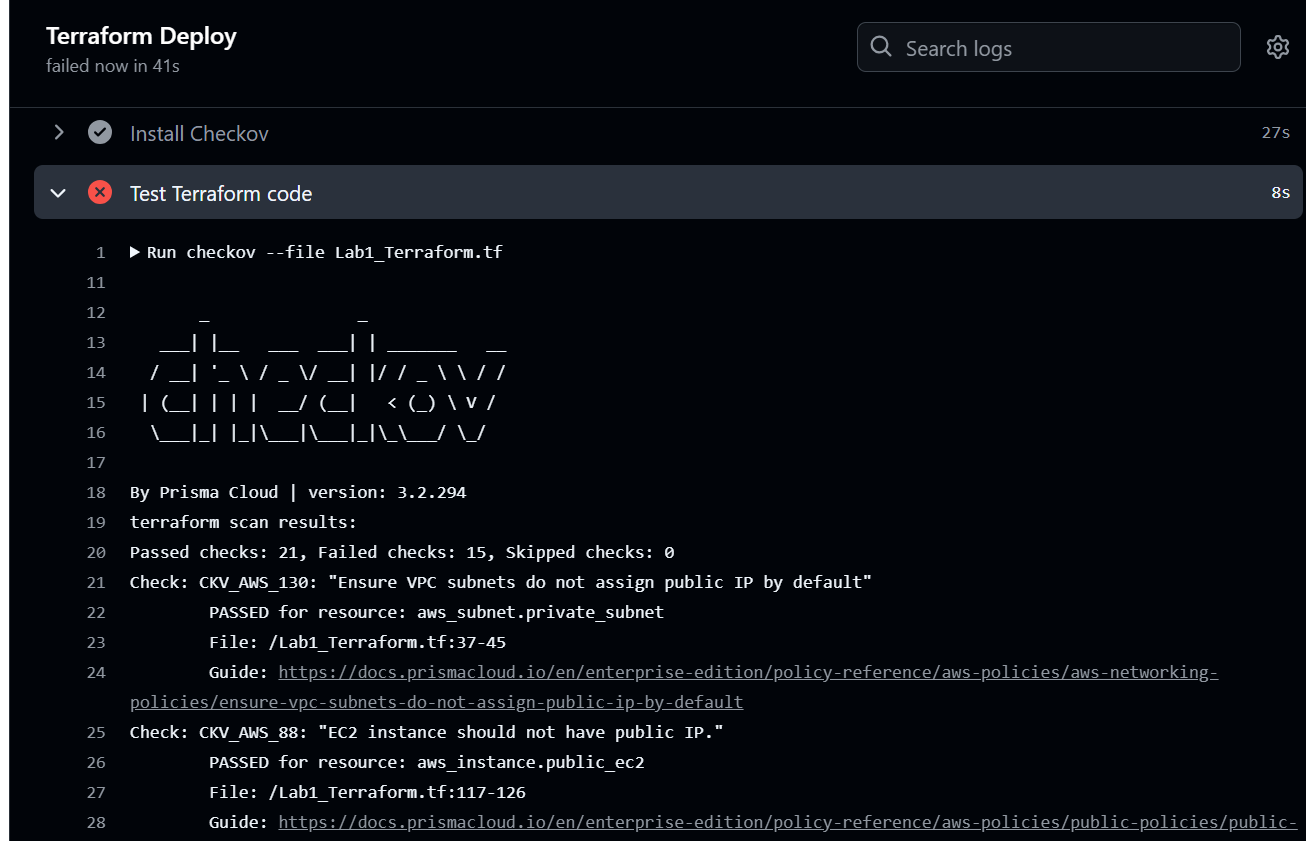
pip3 install checkov

checkov --version

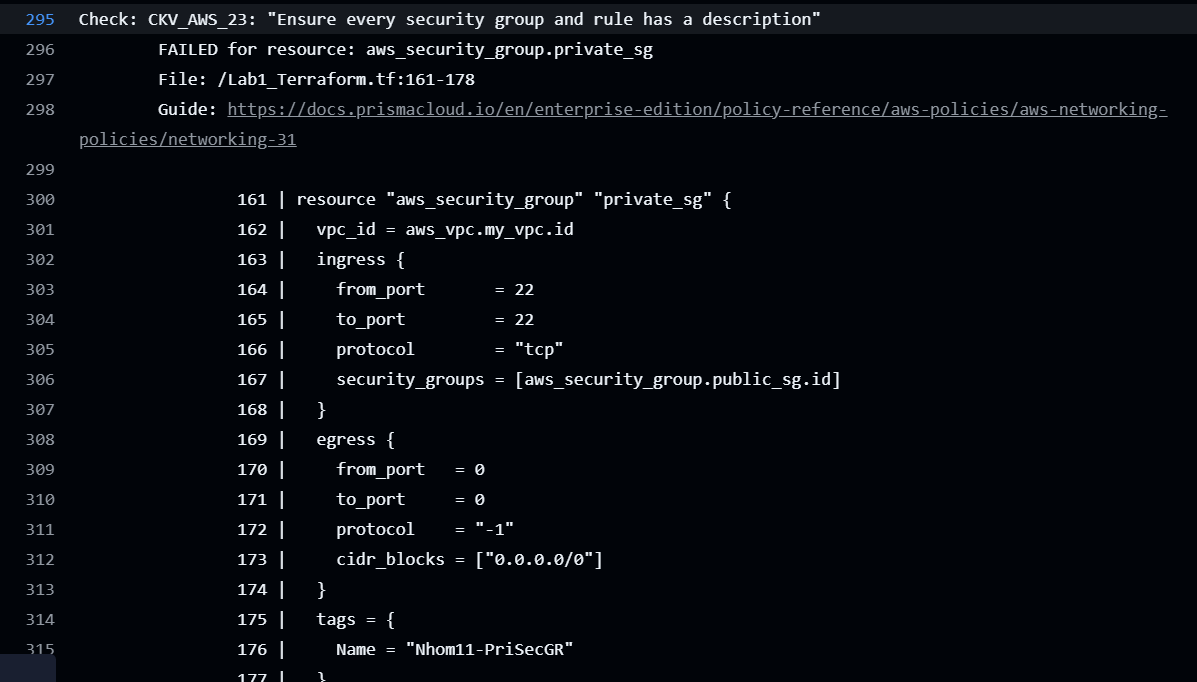
- name: "Test Terraform code"

run: checkov --file Lab1\_Terraform.tf

Code hiện tại đang cần cải tiến vì vướng một số lỗi: CKV\_AWS\_23, CKV2\_AWS\_41, CKV2\_AWS\_11, CKV2\_AWS\_12, CKV2\_AWS\_23, CKV\_AWS\_79 và CKV\_AWS\_135.



1. Chạy checkov kiểm tra code trước khi sửa



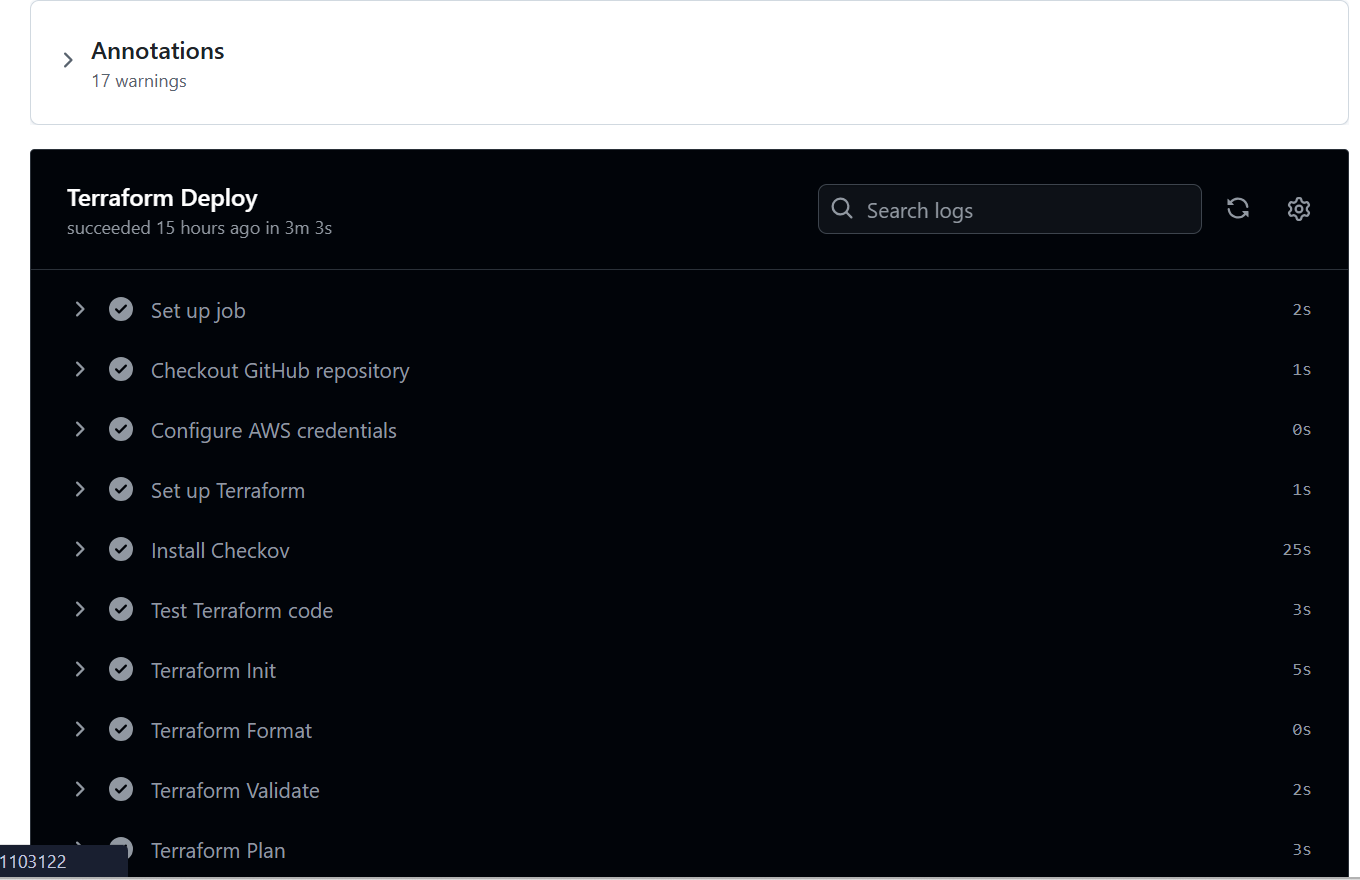
1. Ví dụ về lỗi báo khi sử dụng Checkov scan code

Người dùng có thể bấm vào đường link kèm theo trong từng lỗi để xem và sửa code. Sau đó, chạy lại để Checkov kiểm tra code.



1. Checkov scan code thành công

Sau đó các bước còn lại được thực hiện và thành công deploy code Terraform lên AWS.



1. Github Actions sau khi tích hợp Checkov

## **Triển khai hạ tầng AWS với CloudFormation và tự động hóa quy trình build và deploy** **với AWS CodePipeline**

### **Dùng CloudFormation để triển khai các dịch vụ AWS bao gồm: VPC, Route Tables, NAT, EC2, Security Groups đã thực hiện ở bài tập 1.**

### **Sử dụng AWS CodeBuild, tích hợp cfn-lint và Taskcat để kiểm tra tính đúng đắn của** **mã CloudFormation.**

### **Sử dụng AWS CodePipeline để tự động hóa quy trình build và deploy từ mã nguồn** **trên CodeCommit.**

## **Sử dụng Jenkins để quản lý quy trình CI/CD cho ứng dụng microservices**

### Mục 1

### Mục 2