



## 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
<b>9.5</b>

#### ĐÁNH GIÁ KHÁC:

Tổng thời gian thực hiện	14 ngày
Phân chia công việc	+ Lê Quốc Khánh: Thực hiện câu 1 và 2, viết báo cáo + Nguyễn Văn Anh Tuấn: Thực hiện câu 3, hỗ trợ viết báo cáo + Đỗ Thế Danh: Hỗ trợ thực hiện câu 3
Ý kiến (nếu có) + Khó khăn + Đề xuất, kiến nghị	Không có
Link Github	<a href="https://github.com/LQK164/NT548-DevOps-Exercises/tree/main/Lab2">https://github.com/LQK164/NT548-DevOps-Exercises/tree/main/Lab2</a>

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

## Contents

<b>A. BÁO CÁO CHI TIẾT.....</b>	2
1. Triển khai hạ tầng AWS sử dụng Terraform và tự động hóa quy trình với GitHub Actions .....	2
a. 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 .....	2
b. Tự động hóa quá trình triển khai với GitHub Actions.....	7
c. Tích hợp Checkov để kiểm tra tính tuân thủ và bảo mật của mã nguồn Terraform. ....	9
2. Triển khai hạ tầng AWS với CloudFormation và tự động hóa quy trình build và deploy với AWS CodePipeline .....	12
a. 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 .....	12
b. Sử dụng AWS CodeBuild, tích hợp cfn-lint và Taskcat để kiểm tra tính đúng đắn của mã CloudFormation.....	18
c. Sử dụng AWS CodePipeline để tự động hóa quy trình build và deploy từ mã nguồn trên CodeCommit.....	27
3. Sử dụng Jenkins để quản lý quy trình CI/CD cho ứng dụng microservices.....	36
a. Sử dụng Jenkins để tự động hóa quá trình build, test và deploy ứng dụng miroservices lên Docker. .....	36
b. Tích hợp SonarQube để kiểm tra chất lượng mã nguồn. ....	48

## A. BÁO CÁO CHI TIẾT

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

- a. 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"  
}
```

### Tạo VPC

```
# Tạo VPC  
resource "aws_vpc" "my_vpc" {  
    cidr_block = "10.0.0.0/16"  
  
    tags = {  
        Name = "Nhóm11-VPC"  
    }  
}
```

### Tạo NAT Gateway cho Private Subnet và Internet Gateway cho Public Subnet



```
# Tạo Internet Gateway cho Public Subnet
resource "aws_internet_gateway" "igw" {
    vpc_id = aws_vpc.my_vpc.id

    tags = {
        Name = "Nhóm11-IGW"
    }
}

# Tạo Elastic IP cho NAT Gateway
resource "aws_eip" "nat_eip" {
    domain = "vpc"

    tags = {
        Name = "Nhóm11-ENG"
    }
}

# Tạo NAT Gateway cho Private Subnet
resource "aws_nat_gateway" "nat_gw" {
    allocation_id = aws_eip.nat_eip.id
    subnet_id      = aws_subnet.private_subnet.id

    tags = {
        Name = "Nhóm11-NGW"
    }
}
```

#### Tạo 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 = "Nhóm11-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"
    }
}
```

#### Tạo 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
}
```

Tạo 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 = "Nhóm11-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 = "Nhóm11-PriEC2"
    }
}
```

Tạo các Security 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"
    }
}
```

### b. 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

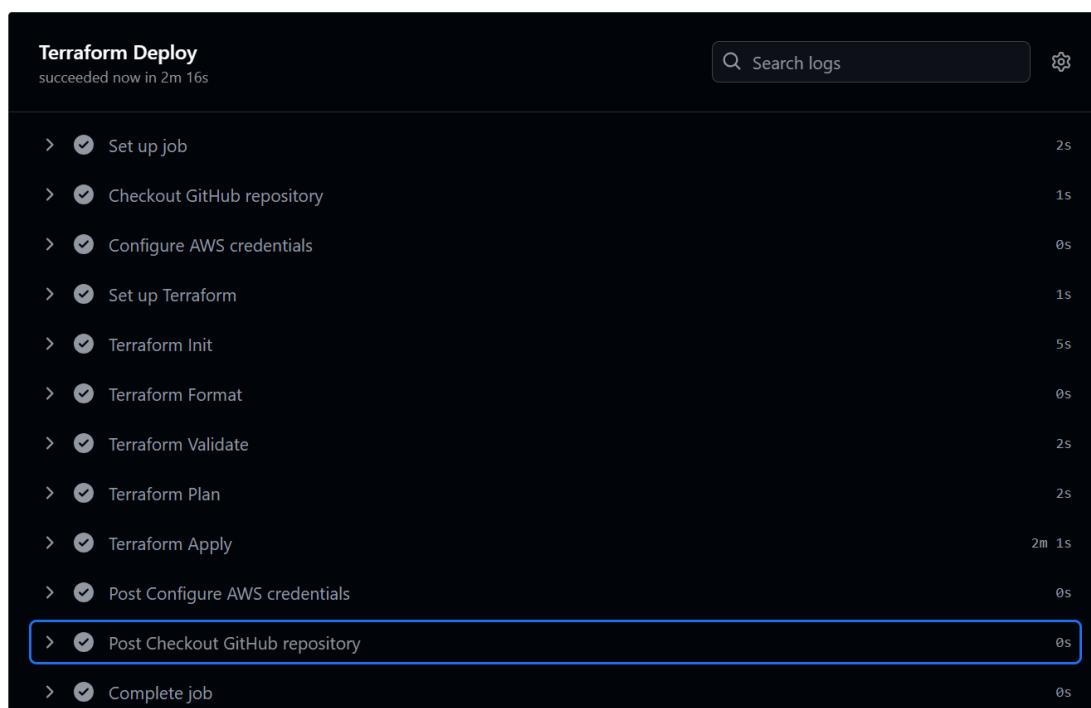
      - name: 'Terraform Validate'
        run: terraform validate
```

```
- 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.



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

c. **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

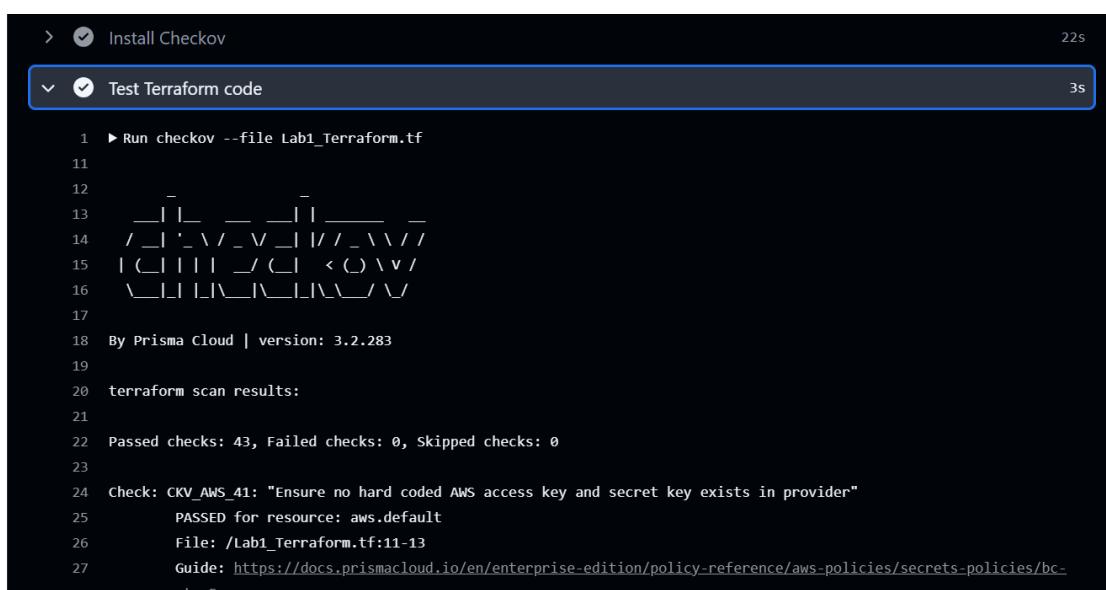
Code hiện tại đang cần cài tiến vì gặp phải một số lỗi: CKV\_AWS\_23, CKV2\_AWS\_41, CKV2\_AWS\_11, CKV2\_AWS\_12, CKV2\_AWS\_23, CKV\_AWS\_79 và CKV\_AWS\_135.

Hình 2. Chạy checkov kiểm tra code trước khi sửa

```
295 Check: CKV_AWS_23: "Ensure every security group and rule has a description"
296     FAILED for resource: aws_security_group.private_sg
297     File: /Lab1_Terraform.tf:161-178
298     Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/aws-networking-policies/networking\_31
299
300     161 | resource "aws_security_group" "private_sg" {
301     162 |     vpc_id = aws_vpc.my_vpc.id
302     163 |     ingress {
303     164 |         from_port    = 22
304     165 |         to_port      = 22
305     166 |         protocol     = "tcp"
306     167 |         security_groups = [aws_security_group.public_sg.id]
307     168 |     }
308     169 |     egress {
309     170 |         from_port    = 0
310     171 |         to_port      = 0
311     172 |         protocol     = "-1"
312     173 |         cidr_blocks = ["0.0.0.0/0"]
313     174 |     }
314     175 |     tags = {
315     176 |         Name = "Nhom11-PriSecGR"
316     177 |     }
```

Hình 3. 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.



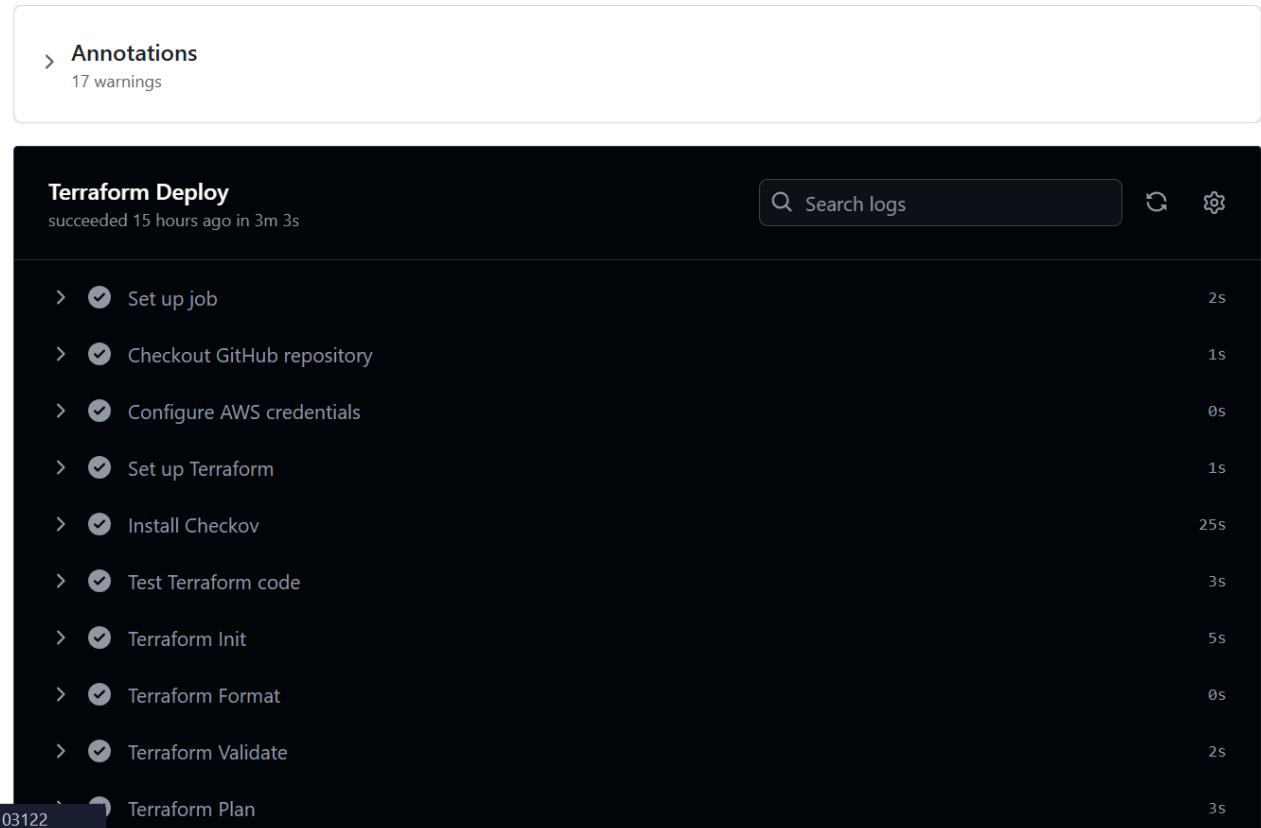
The screenshot shows a GitHub Actions workflow step titled 'Test Terraform code'. It includes a summary of the command run: 'Run checkov --file Lab1\_Terraform.tf'. The output shows the results of the scan, indicating 43 passed checks, 0 failed checks, and 0 skipped checks. A detailed breakdown of one of the passed checks is provided, linking to the AWS documentation for policy reference.

```
>  Install Checkov 22s

<input checked="" type="checkbox"/> Test Terraform code 3s
  1 ► Run checkov --file Lab1_Terraform.tf
 11
 12
 13   _|_|_ _ _ _|_|_____
 14   /_|'/_\/_\_|/_/_\_\_/
 15   |(_|||_|/_/(_|<_) \v/
 16   \_\_|_|_\|\_\_|_\|\_/\_/
 17
 18 By Prisma Cloud | version: 3.2.283
 19
 20 terraform scan results:
 21
 22 Passed checks: 43, Failed checks: 0, Skipped checks: 0
 23
 24 Check: CKV_AWS_41: "Ensure no hard coded AWS access key and secret key exists in provider"
 25     PASSED for resource: aws.default
 26     File: /Lab1_Terraform.tf:11-13
 27     Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/secrets-policies/bc-aws-secrets-5
```

Hình 4. Checkov scan code thành công

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



Hình 5. Github Actions sau khi tích hợp Checkov

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

### a. 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.

Khai báo các Parameter, đặt tên cho môi trường đồng thời gán địa chỉ mặc định cho VPC, Public Subnet và Private Subnet

Dịch vụ	Địa chỉ IP
VPC	10.0.0.0/16
Public Subnet	10.0.0.0/24
Private Subnet	10.0.1.0/24

Parameters:

EnvironmentName:

Description: An environment name that will be prefixed to resource names

Type: String

#AllowedPattern: '[A-Za-z0-9-]+'  
VpcCIDR:

Description: Please enter the IP range (CIDR notation) for this VPC

Type: String

Default: 10.0.0.0/16

PublicSubnetCIDR:

Description: Please enter the IP range (CIDR notation) for the public subnet in the Availability Zone

Type: String

Default: 10.0.0.0/24

PrivateSubnetCIDR:

Description: Please enter the IP range (CIDR notation) for the private subnet in the Availability Zone

Type: String

Default: 10.0.1.0/24

### Tạo VPC

#####Create VPC#####

MyVPC:

Type: AWS::EC2::VPC

Properties:

CidrBlock: !Ref VpcCIDR

EnableDnsSupport: true

EnableDnsHostnames: true

Tags:

- Key: Name

Value: !Ref EnvironmentName

### Tạo Private Subnet và Public Subnet

- **Public Subnet:** Một subnet trong VPC có thể truy cập Internet trực tiếp. Được sử dụng cho các instance cần truy cập Internet.
- **Private Subnet:** Một subnet trong VPC không thể truy cập Internet trực tiếp. Được sử dụng cho các instance không cần truy cập Internet.

```
#####Public Subnet#####
PublicSubnet:
  Type: AWS::EC2::Subnet
  Properties:
    AvailabilityZone:
      Fn::Select:
        - 0
        - Fn::GetAZs: ""
    VpcId: !Ref MyVPC
    CidrBlock: !Ref PublicSubnetCIDR
    MapPublicIpOnLaunch: true
  Tags:
    - Key: Name
      Value: !Join ['', [!Ref "AWS::StackName", "-Pub-SubNet"]]
```

```
#####Private Subnet#####
PrivateSubnet:
```

```
Type: AWS::EC2::Subnet
Properties:
  AvailabilityZone:
    Fn::Select:
      - 0
      - Fn::GetAZs: ""
  VpcId: !Ref MyVPC
  CidrBlock: !Ref PrivateSubnetCIDR
  MapPublicIpOnLaunch: false
  Tags:
    - Key: Name
      Value: !Join ['', [!Ref "AWS::StackName", "-Pri-SubNet"]]
```

Tạo Internet Gateway và gán cho VPC, từ đó cho phép các Instance trong VPC truy cập vào mạng Internet

```
#####Create Internet Gateway#####
InternetGateway:
  Type: AWS::EC2::InternetGateway
  DependsOn: MyVPC
  Properties:
    Tags:
      - Key: Name
        Value: !Join ['', [!Ref "AWS::StackName", "-IGW"]]
#####Attach Internet Gateway to VPC#####
AttachGateway:
  Type: AWS::EC2::VPCGatewayAttachment
  Properties:
    VpcId: !Ref MyVPC
    InternetGatewayId: !Ref InternetGateway
```

Tạo Nat Gateway và gán cho Private Subnet

- **Elastic IP:** Một địa chỉ IP tĩnh công cộng có thể được gán cho một NAT Gateway.
- **NAT Gateway:** Một thiết bị mạng logic trong AWS, cho phép các instance trong private subnet truy cập Internet.

```
ElasticIP:
  Type: AWS::EC2::EIP
  Properties:
    Domain: vpc
    Tags:
      - Key: Name
        Value: !Join ['', [!Ref "AWS::StackName", "-NGW-EIP"]]
NatGateway:
  Type: AWS::EC2::NatGateway
  Properties:
    SubnetId: !Ref PublicSubnet
    AllocationId: !GetAtt ElasticIP.AllocationId
    Tags:
      - Key: Name
        Value: !Join ['', [!Ref "AWS::StackName", "-NGW"]]
```

Tạo Default Security Groups cho VPC. Default Security Groups này đóng vai trò như tường lửa để kiểm soát các Instance ra vào trong VPC.

```
##### Default Security Group for VPC #####
DefaultSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Default security group for the VPC
    VpcId: !Ref MyVPC
    SecurityGroupIngress:
      - IpProtocol: -1
        FromPort: -1
        ToPort: -1
        CidrIp: 10.0.0.0/16
    SecurityGroupEgress:
      - IpProtocol: -1
        FromPort: -1
        ToPort: -1
        CidrIp: 0.0.0.0/0
    Tags:
      - Key: Name
        Value: !Join ['', [!Ref "AWS::StackName", "-Default-SG"]]
```

#### Tạo Security Groups cho EC2 của Private Subnet và Public Subnet

- Public EC2 Security Group: Chỉ cho phép kết nối SSH (port 22) từ một IP cụ thể (Ở đây là địa chỉ 123.250.165.100/32)
- Private EC2 Security Group: Cho phép kết nối từ Public EC2 instance thông qua port cần thiết (Ở đây là port 22).

```
#####Security Group for EC2 Public Instance#####
PublicSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Allow SSH access to Public EC2
    VpcId: !Ref MyVPC
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: 123.250.165.100/32 # User's IP
    Tags:
      - Key: Name
        Value: !Join ['', [!Ref "AWS::StackName", "-Pub-SecGr"]]
#####Security Group for EC2 Private Instance#####
PrivateSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Allow SSH from Public EC2
    VpcId: !Ref MyVPC
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        SourceSecurityGroupId: !Ref PublicSecurityGroup
    Tags:
      - Key: Name
        Value: !Join ['', [!Ref "AWS::StackName", "-Pri-SecGr"]]
```

Tạo EC2 cho Private Subnet và Public Subnet

- Public instance có thể truy cập từ Internet.
- Private instance chỉ có thể truy cập từ Public instance thông qua SSH hoặc các phương thức bảo mật khác

```
#####Create EC2 for Public Instance#####
myEC2InstancePublic:
  Type: AWS::EC2::Instance
  Properties:
    SubnetId: !Ref PublicSubnet
    InstanceType: t2.micro #Change another type if you want
    SecurityGroupIds:
      - !Ref PublicSecurityGroup
    ImageId: ami-0866a3c8686eaeeba #Insert your ID
    Tags:
      - Key: Name
        Value: !Join ['', [!Ref "AWS::StackName", "-Pub-EC2"]]

#####Create EC2 for Private Instance#####
myEC2InstancePrivate:
  Type: AWS::EC2::Instance
  Properties:
    SubnetId: !Ref PrivateSubnet
    InstanceType: t2.micro #Insert your type
    SecurityGroupIds:
      - !Ref PrivateSecurityGroup
    ImageId: ami-0866a3c8686eaeeba #Insert your ID
    Tags:
      - Key: Name
        Value: !Join ['', [!Ref "AWS::StackName", "-Pri-EC2"]]
```

Kết quả tạo CloudFormation thành công trên AWS Console

The screenshot shows two main sections of the AWS CloudFormation console:

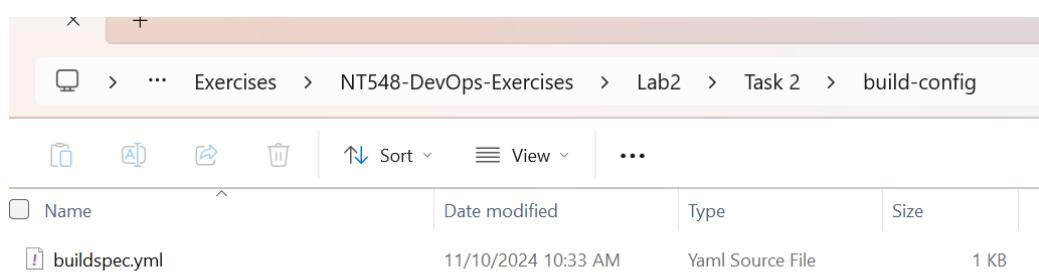
- Stacks (2)**: Shows a list of stacks. One stack, "Nhom11Stack", is selected and highlighted with a blue border. Its status is "CREATE\_COMPLETE".
- Events (62)**: Shows a table of events. The first three rows are listed below:
 

Timestamp	Logical ID	Status
2024-11-17 11:05:43 UTC+0700	Nhom11Stack	CREATE_COMPLETE
2024-11-17 11:05:42 UTC+0700	PrivateRoute	CREATE_COMPLETE
2024-11-17 11:05:42 UTC+0700	PrivateRoute	CREATE_IN_PROGRESS

Hình 6. CloudFormation trên AWS Console

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

Để chạy được AWS CodeBuild, phải có một file yaml để build đặt tên là "buildspec.yml".



Hình 7. File buildspec.yml

Cài đặt phiên bản runtime cần thiết, ở đây là phiên bản python3 trở lên

Phần install, sử dụng:

- pip install cfn-lint: Cài đặt công cụ cfn-lint để kiểm tra cú pháp và cấu trúc của template CloudFormation.
- pip install taskcat: Cài đặt công cụ taskcat để kiểm tra và triển khai template CloudFormation trên nhiều vùng.

Phần pre\_build:

- cfn-lint CloudFormation.yaml: Kiểm tra cú pháp và cấu trúc của file template CloudFormation.yaml.

Phần build:

- taskcat test run: Thực hiện việc kiểm tra và triển khai template CloudFormation bằng công cụ taskcat.

```
version: 0.2
phases:
  install:
    runtime-versions:
      python: 3.x
    commands:
      - pip install cfn-lint
      - pip install taskcat
  pre_build:
    commands:
      - cfn-lint CloudFormation.yaml
  build:
    commands:
      - taskcat test run
```

Xây dựng file taskcat để kiểm tra template Cloudformation:

Chọn region cần kiểm tra (ở đây là us-east-1)

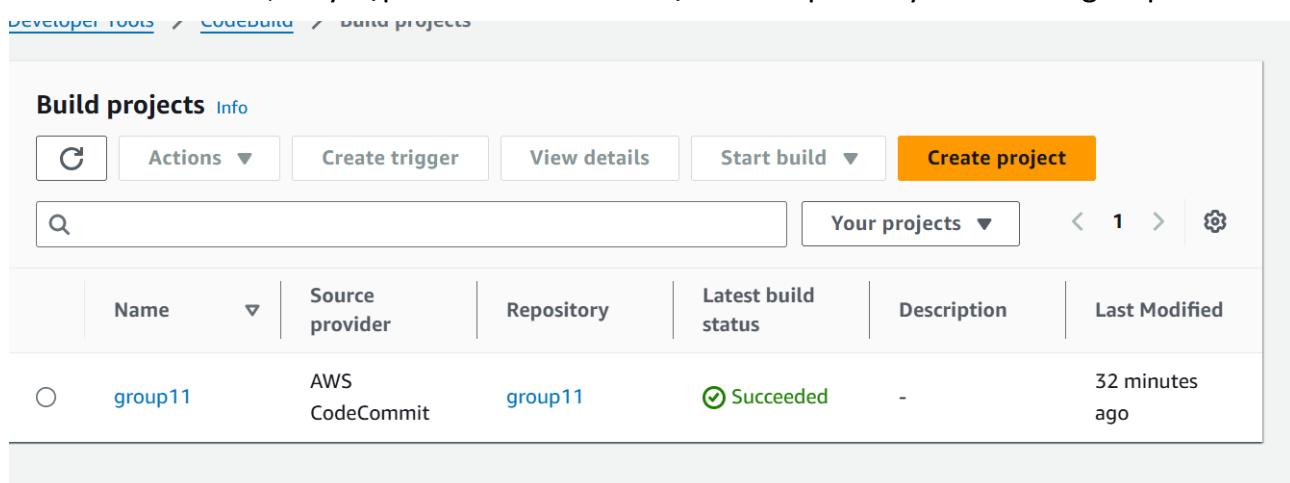
Phần test:

- **template:** Chỉ ra đường dẫn đến file template CloudFormation cần kiểm tra. Trong trường hợp này là "./CloudFormation.yml".
- **parameters:**

- + **EnvironmentName:** Đặt giá trị cho tham số "EnvironmentName" là "TestEnvironment".
- + **VpcCIDR, PublicSubnetCIDR, PrivateSubnetCIDR:** Cung cấp thông tin, địa chỉ IP cho VPC, Public Subnet và Private Subnet.

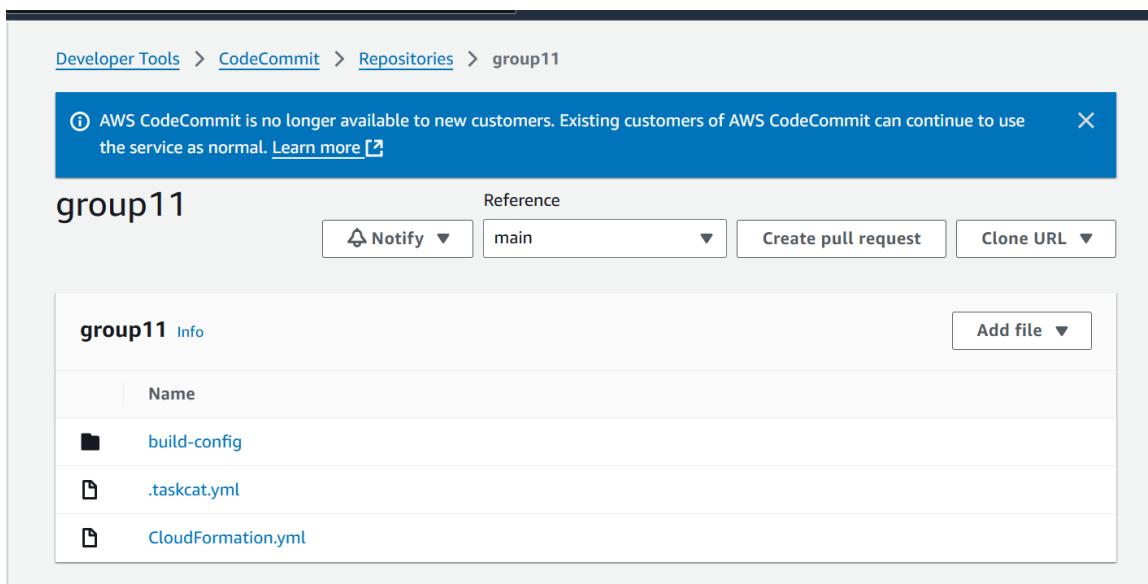
```
project:  
  name: lab2-cfn-check  
  regions:  
    - us-east-1 # Add any additional regions you want to test  
  
tests:  
  testcfn:  
    template: ./CloudFormation.yml  
    parameters:  
      EnvironmentName: TestEnvironment  
      VpcCIDR: 10.0.0.0/16  
      PublicSubnetCIDR: 10.0.0.0/24  
      PrivateSubnetCIDR: 10.0.1.0/24
```

Trên AWS Console, truy cập vào CodeCommit tạo một repository mới tên là group11.



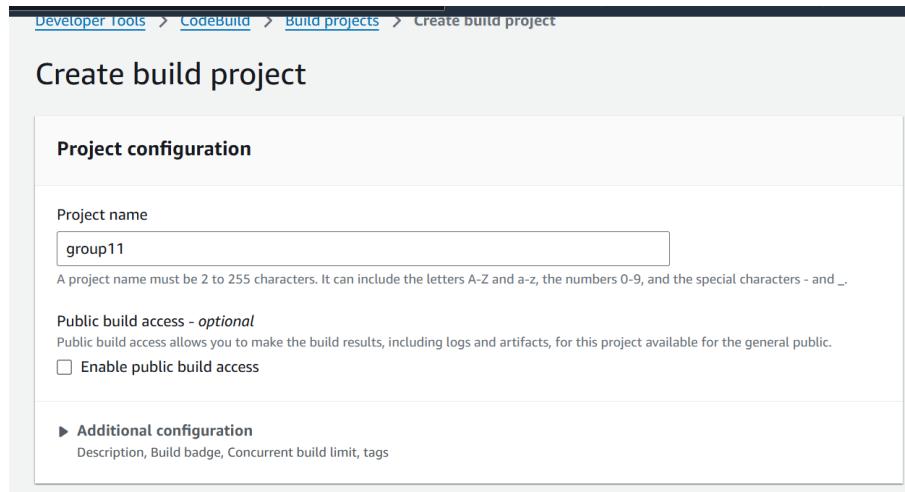
Hình 8. Repository group11

Thêm các file: template CloudFormation, taskcat.yml và một folder tên build-config chưa file buildspec.yml.



Hình 9. Các file trong repo group11

Truy cập vào AWS CodeBuild, chọn Create build project.



Hình 10. Project group11

Chọn Source là AWS Commit, và chọn đúng repo đã tạo ở phía trên

Source Add source

**Source 1 - Primary**

Source provider AWS CodeCommit

Repository group11

Reference type  
Choose the source version reference type that contains your source code.  
 Branch  
 Git tag  
 Commit ID

Branch  
Choose a branch that contains the code to build.  
main

Commit ID - optional  
Choose a commit ID. This can shorten the duration of your build.  
Search

Hình 11. Souce CodeBuild

Chọn OS muốn sử dụng (ở đây là Ubuntu)

**Environment**

Provisioning model [Info](#)  
 On-demand  
Automatically provision build infrastructure in response to new builds.

Reserved capacity  
Use a dedicated fleet of instances for builds. A fleet's compute and environment type will be used for the project.

Environment image  
 Managed image  
Use an image managed by AWS CodeBuild

Custom image  
Specify a Docker image

Compute  
 EC2  
Optimized for flexibility during action runs

Lambda  
Optimized for speed and minimizes the start up time of workflow actions

Operating system  
Ubuntu

Hình 12. Phần Environment

Hệ thống sẽ tự tạo một service role mới tên là codebuild-group11-service-role.

Always use the latest image for this runtime version ▾

Use GPU-enhanced compute

Service role

New service role  
Create a service role in your account

Existing service role  
Choose an existing service role from your account

Role name

codebuild-group11-service-role

Type your service role name

► Additional configuration  
Timeout, privileged, certificate, VPC, compute type, environment variables, file systems, auto-retry

Hình 13. Service role

Phần buildspec, chọn use a buildpec file và nhận đường dẫn tới file buildspec trong AWS CodeCommit.

Buildspec

Build specifications

Insert build commands  
Store build commands as build project configuration

Use a buildspec file  
Store build commands in a YAML-formatted buildspec file

Buildspec name - optional  
By default, CodeBuild looks for a file named buildspec.yml in the source code root directory. If your buildspec file uses a different name or location, enter its path from the source root here (for example, buildspec-two.yml or configuration/buildspec.yml).

build-config/buildspec.yml

Hình 14. Kết nối với file buildspec.yml

Hoàn thành các bước trên và chọn Create build project

## 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

CloudWatch

CloudWatch logs - *optional*  
Checking this option will upload build output logs to CloudWatch.

Group name - *optional*  
`aws/codebuild/group11`

The group name of the logs in CloudWatch Logs. The log group name will be /aws/codebuild/<project-name> by default.

Stream name prefix - *optional*

The prefix of the stream name of the CloudWatch Logs.

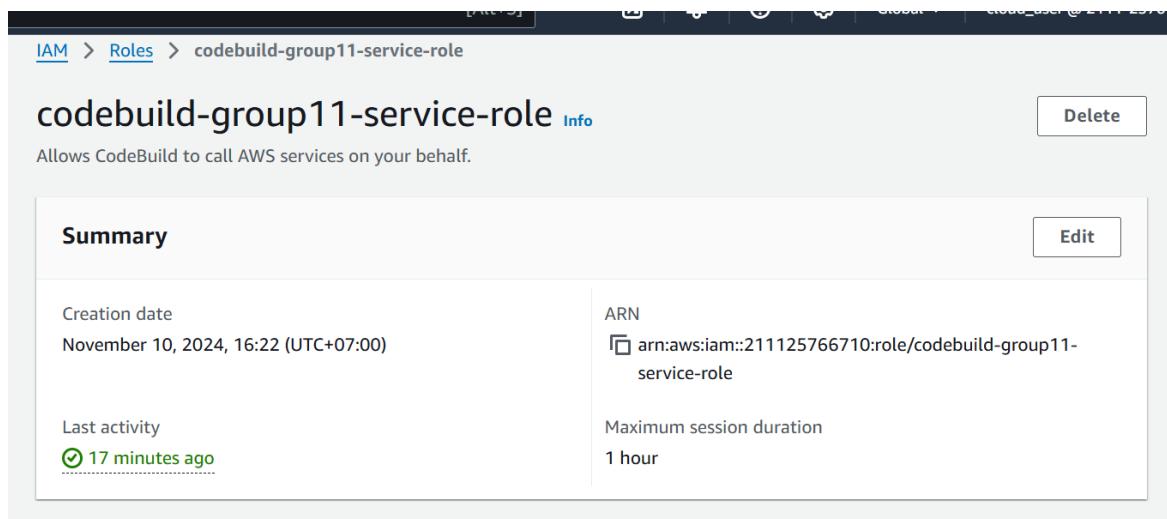
S3

S3 logs - *optional*  
Checking this option will upload build output logs to S3.

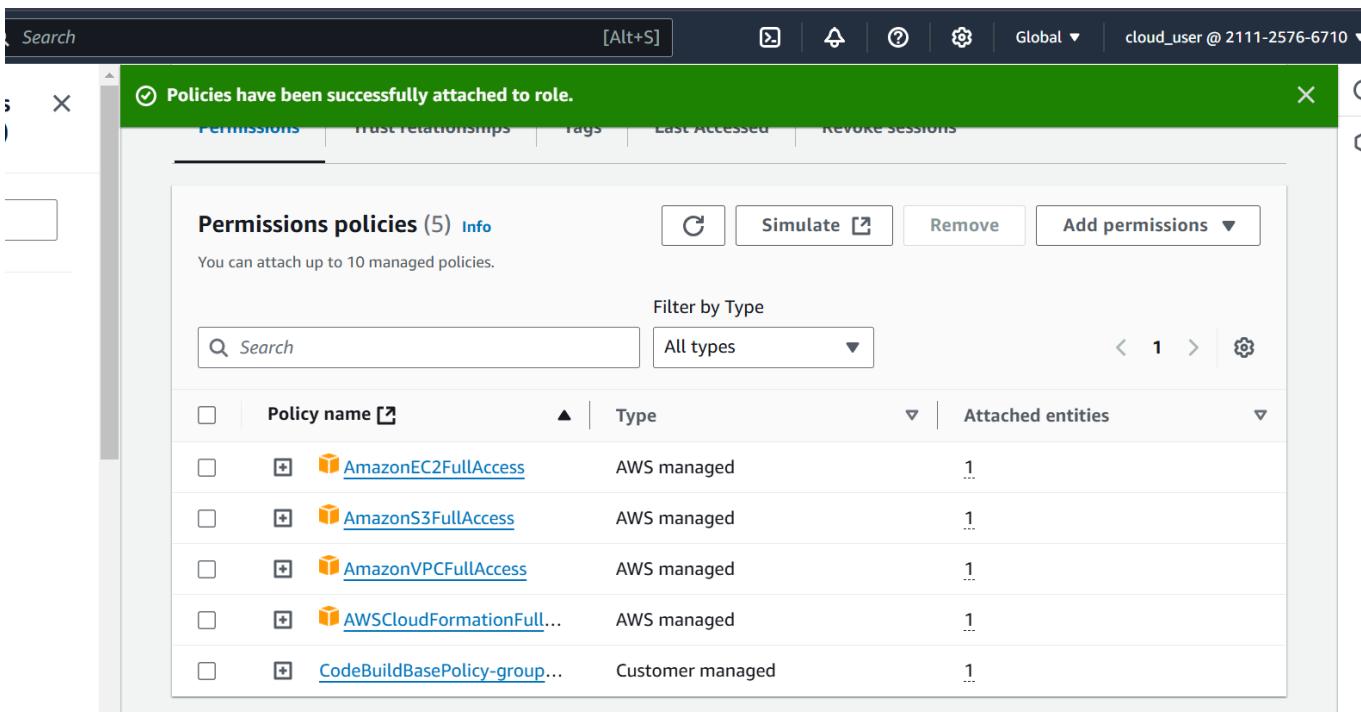
**Create build project**

Hình 15. Chọn Create build project

Sau khi tạo project thành công, truy cập vào IAM và chọn role, tìm role vừa tạo ở phần trên và attach các policy như hình 13 để thực hiện kiểm tra template của CloudFormation.



Hình 16. IAM role codebuild-group11-service-role



Hình 17. Permissions policies

Chạy cfn-lint thành công:

```
13:30:12.104917 Phase complete: INSTALL State: SUCCEEDED
13:30:12.104937 Phase context status code: Message:
13:30:12.144804 Entering phase PRE_BUILD
13:30:12.145801 Running command cfn-lint CloudFormation.yaml

13:30:14.190457 Phase complete: PRE_BUILD State: SUCCEEDED
13:30:14.190478 Phase context status code: Message:
13:30:14.228945 Entering phase BUILD
13:30:14.229879 Running command taskcat test run
```

Hình 18. Cfn-lint success

Quay lại AWS CodeBuild, truy cập vào project vào chọn start build. Hình 19 thể hiện Taskcat đã chạy thành công.

## 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

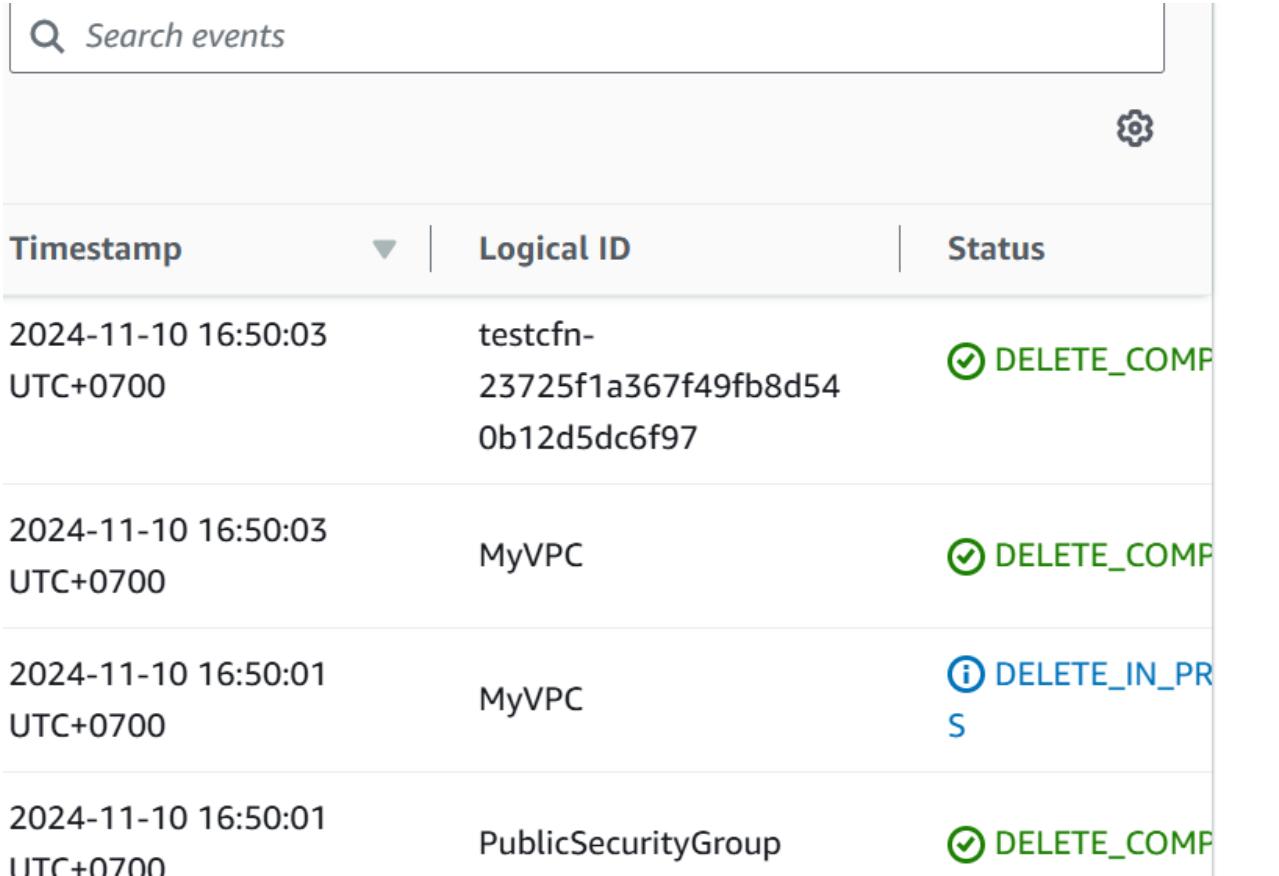
```

252 |   |   |   |   |   |   |   |
253 |   |   |   |   |   |   |   |
254 |   |   |   |   |   |   |   |
255 |   |   |   |   |   |   |   |
256
257
258
259 version 0.9.55
260 Not in terminal, reprint now using normal build-in print function.
261
262 [INFO ] : Linting passed for file: /codebuild/output/src4261778648/src/git-codecommit.us-east-
263     1.amazonaws.com/v1/repos/group11/CloudFormation.yml
264 [S3: --> ] s3://tcat-lab2-cfn-check-9k6hj9rh/lab2-cfn-check/CloudFormation.yml
265 [S3: --> ] s3://tcat-lab2-cfn-check-9k6hj9rh/lab2-cfn-check/build-config/buildspec.yml
266     ↳ stack ⚡ tCaT-lab2-cfn-check-testcfn-37378aeb43da4fdc9ce766644228cc63
267     | region: us-east-1
268     |   ↳ status: CREATE_IN_PROGRESS
269     | stack ⚡ tCaT-lab2-cfn-check-testcfn-37378aeb43da4fdc9ce766644228cc63
270     |   ↳ region: us-east-1
271     |   ↳ status: CREATE_IN_PROGRESS
272     | stack ⚡ tCaT-lab2-cfn-check-testcfn-37378aeb43da4fdc9ce766644228cc63
273     |   ↳ region: us-east-1
274     |   ↳ status: CREATE_IN_PROGRESS
275     | stack ⚡ tCaT-lab2-cfn-check-testcfn-37378aeb43da4fdc9ce766644228cc63
276     |   ↳ region: us-east-1
277     |   ↳ status: CREATE_IN_PROGRESS
278     | stack ⚡ tCaT-lab2-cfn-check-testcfn-37378aeb43da4fdc9ce766644228cc63
279     |   ↳ region: us-east-1
280     |   ↳ status: CREATE_IN_PROGRESS
281     | stack ⚡ tCaT-lab2-cfn-check-testcfn-37378aeb43da4fdc9ce766644228cc63

```

Hình 19. Taskcat chạy thành công

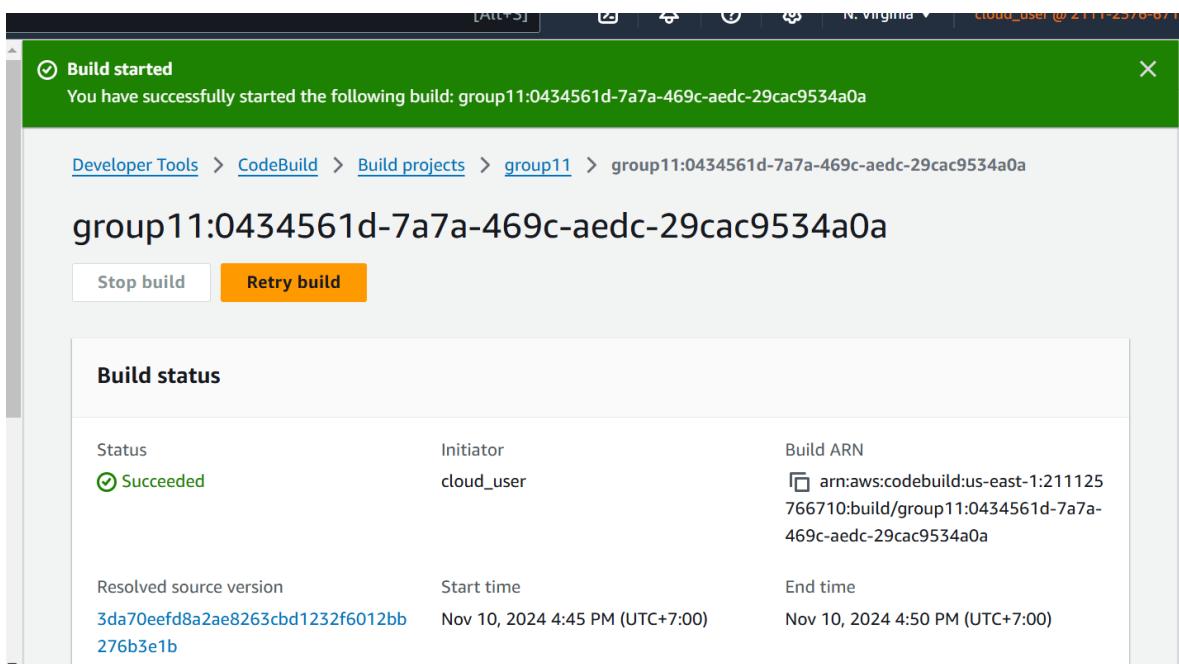
Kiểm tra bên CloudFormation thì Taskcat đã tạo template để kiểm tra xong và đã xóa template đó.



Search events			
Timestamp	Logical ID	Status	
2024-11-10 16:50:03 UTC+0700	testcfn- 23725f1a367f49fb8d54 0b12d5dc6f97	✓ DELETE_COMPLETE	
2024-11-10 16:50:03 UTC+0700	MyVPC	✓ DELETE_COMPLETE	
2024-11-10 16:50:01 UTC+0700	MyVPC	ℹ️ DELETE_IN_PROGRESS	
2024-11-10 16:50:01 UTC+0700	PublicSecurityGroup	✓ DELETE_COMPLETE	

Hình 20. Template CloudFormation

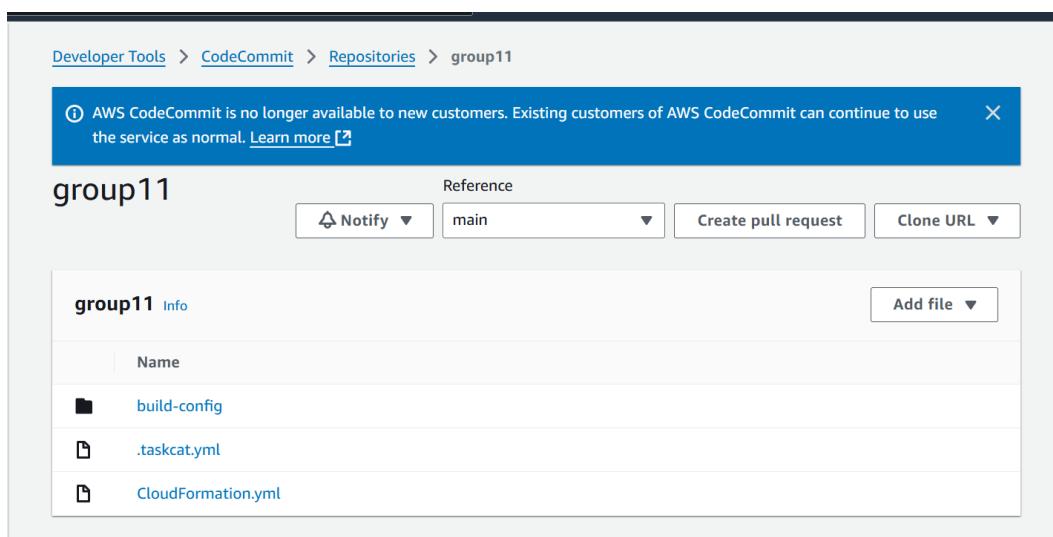
Project đã build thành công.



Hình 21. Build thành công

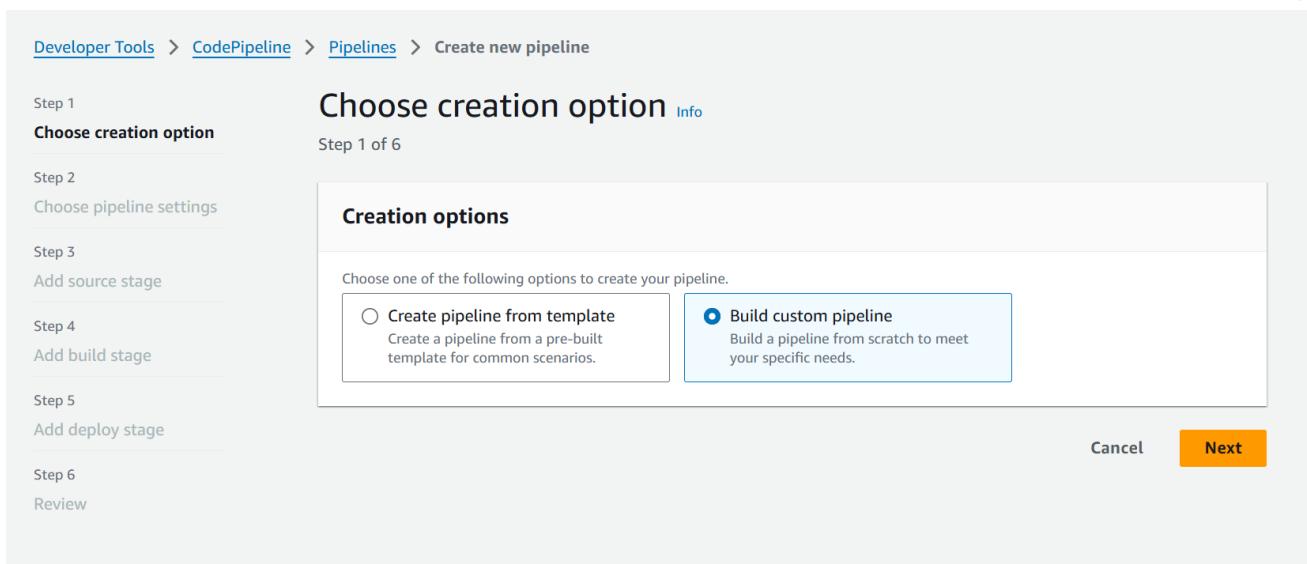
c. Sử dụng AWS CodePipeline để tự động hóa quy trình build và deploy từ mã nguồn trên CodeCommit.

Kiểm tra lại mã nguồn đã update trên AWS CodeCommit trên yêu cầu 2.



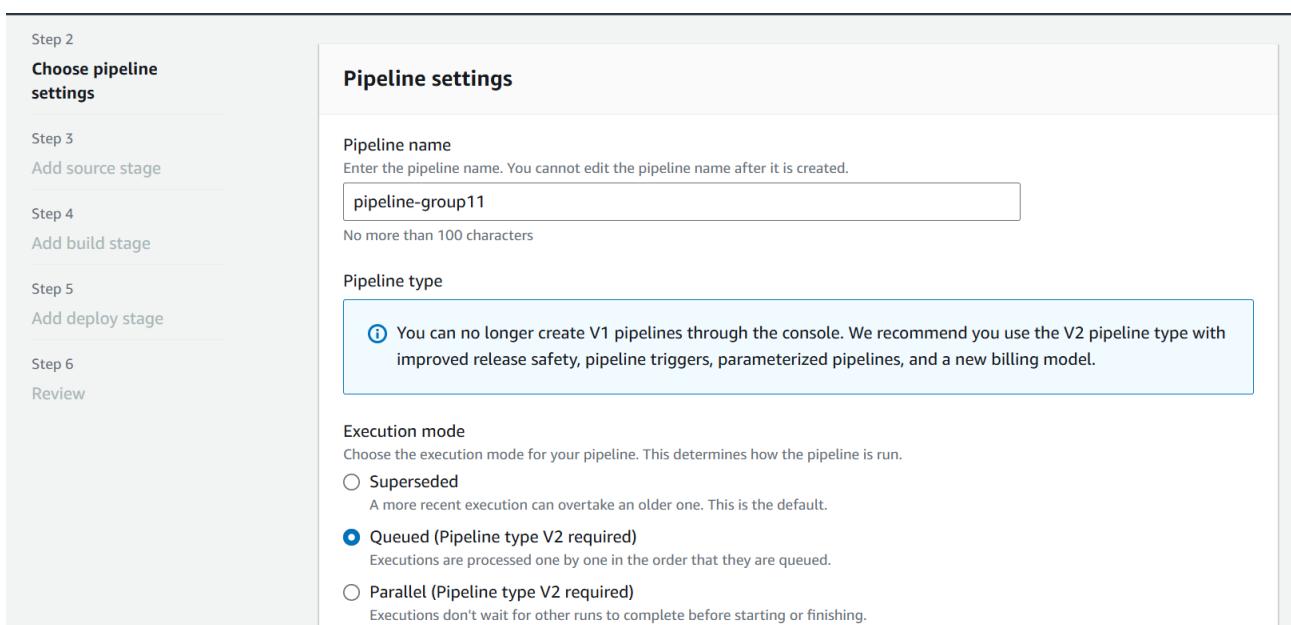
Hình 22. Repo đã được tạo trên yêu cầu 2.

Truy cập AWS CodePipeline chọn Create new pipeline. Sau đó chọn Build custom pipeline và chọn Next.



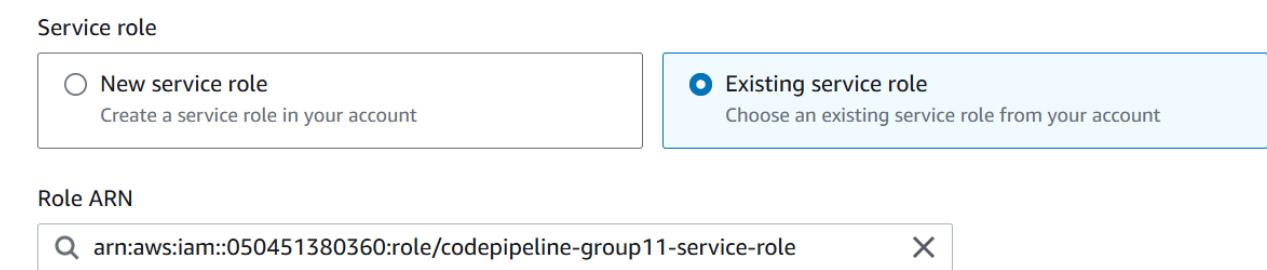
Hình 23. Chọn creation option

Đặt tên cho Pipeline là pipeline-group11



Hình 24. Pipeline-group11

Chọn Existing service role và nhập Role ARN



Hình 25. Role ARN

## 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

Trong role phía trên cấp các Policy phù hợp cho Pipeline

The screenshot shows the AWS IAM Policies page. On the left, there's a sidebar with navigation links like 'Search IAM', 'board', 'Access management', 'groups', 'AWS Lambda', 'AWS Lambda providers', 'AWS Lambda settings', and 'access management'. The main area has a heading 'You can attach up to 10 managed policies.' and a search bar. A table lists various AWS managed policies:

Policy name	Type	Attached entities
<a href="#">AdministratorAccess</a>	AWS managed - job function	3
<a href="#">AmazonEC2FullAccess</a>	AWS managed	2
<a href="#">AmazonS3FullAccess</a>	AWS managed	2
<a href="#">AmazonVPCFullAccess</a>	AWS managed	2
<a href="#">AWSCloudFormationFull...</a>	AWS managed	2
<a href="#">AWSCodeBuildAdminAcc...</a>	AWS managed	1
<a href="#">AWSCodeBuildDevelope...</a>	AWS managed	1
<a href="#">AWSCodeDeployFullAccess</a>	AWS managed	1

Hình 26. Codepipeline-group11-service-role

Ở phần Source, cho Source provider là AWS CodeCommit và nhập tên Repo đã tạo cùng nhánh thực thi yêu cầu

The screenshot shows the AWS CodePipeline Source stage configuration. On the left, a sidebar lists steps from 'Step 2' to 'Step 6'. The main area is titled 'Source' and contains the following fields:

- Source provider:** AWS CodeCommit
- Repository name:** group11
- Branch name:** main
- Change detection options:**
  - Amazon CloudWatch Events (recommended) - Use Amazon CloudWatch Events to automatically start your pipeline when a change occurs in the source code.
  - AWS CodePipeline - Use AWS CodePipeline to check periodically for changes

Hình 27. Source stage

Ở phần Build, chọn Other build providers và chọn AWS CodeBuild. Sau đó chọn Project là group11

Step 2  
Choose pipeline settings

Step 3  
Add source stage

Step 4  
Add build stage

Step 5  
Add deploy stage

Step 6  
Review

**Build - optional**

**Build provider**  
Choose the tool you want to use to run build commands and specify artifacts for your build action.

Commands       Other build providers

AWS CodeBuild

**Project name**  
Choose a build project that you have already created in the AWS CodeBuild console. Or create a build project in the AWS CodeBuild console and then return to this task.

group11      or

**Environment variables - optional**  
Choose the key, value, and type for your CodeBuild environment variables. In the value field, you can reference variables generated by CodePipeline. [Learn more](#)

**Build type**

Single build       Batch build

Hình 28. Build Stage

Phần Deploy chọn Deploy provider là AWS CloudFormation. Kiểm tra region (nếu cần thiết) và ở Action mode chọn Create or update a stack (hình 29). Cuối cùng chọn Stackname đã tạo bên CloudFormation (hình 30).

**Deploy - optional**

**Deploy provider**  
Choose how you deploy to instances. Choose the provider, and then provide the configuration details for that provider.

AWS CloudFormation

**Region**

US East (N. Virginia)

**Input artifacts**  
Choose an input artifact for this action. [Learn more](#)

BuildArtifact   
Defined by: Build

No more than 100 characters

**Action mode**  
When you update an existing stack, the update is permanent. When you use a change set, the result provides a diff of the updated stack and the original stack before you choose to execute the change.

Create or update a stack

Hình 29. Deploy stage (1)

Defined by: Build  
No more than 100 characters

Action mode  
When you update an existing stack, the update is permanent. When you use a change set, the result provides a diff of the update and the original stack before you choose to execute the change.

Create or update a stack

Stack name  
If you are updating an existing stack, choose the stack name.

Stack name: Nhom11Stack

Template  
Specify the template you uploaded to your source location.

Artifact name: BuildArtifact  
File name: cfn  
Template file path: BuildArtifact::cfn

Template configuration - optional  
Specify the configuration file you uploaded to your source location.

Use configuration file

Artifact name  
File name  
Template configuration file path

Hình 30. Deploy stage (2)

Chọn role đã thiết lập cho Cloudformation

Capabilities - optional  
Specify whether you want to allow AWS CloudFormation to create IAM resources on your behalf.

Role name  
arn:aws:iam::050451380360:role/Deploy\_cfn-group11

Output file name

File generated by this action

Hình 31. Role phần deploy

Quay lại phần IAM và chọn role Deploy\_cfn\_group11 và thiết lập các Policy cần thiết như hình 32.

Permissions policies (4) <a href="#">Info</a>			
<a href="#">G</a> <a href="#">Simulate</a> <a href="#">Remove</a> <a href="#">Add permissions</a> ▾			
You can attach up to 10 managed policies.			
Filter by Type			
<input type="checkbox"/> Policy name ▾	Type	Attached entities	▼
<input type="checkbox"/> <a href="#">AmazonEC2FullAccess</a>	AWS managed	3	...
<input type="checkbox"/> <a href="#">AmazonS3FullAccess</a>	AWS managed	3	...
<input type="checkbox"/> <a href="#">AmazonVPCFullAccess</a>	AWS managed	3	...
<input type="checkbox"/> <a href="#">AWSCloudFormationFull...</a>	AWS managed	3	...

Hình 32. Policies của role Deploy\_cfn\_group11

Sau đó review lại các bước trước khi tạo Pipeline lần lượt ở hình 33, hình 34, hình 35 và hình 36

**Step 1: Choose pipeline settings**

Pipeline settings
Pipeline name pipeline-group11
Pipeline type V2
Execution mode QUEUED
Artifact location A new Amazon S3 bucket will be created as the default artifact store for your pipeline
Service role name arn:aws:iam::992382613074:role/codepipeline-group11-service-role

Hình 33. Bước 1

Step 2: Add source stage

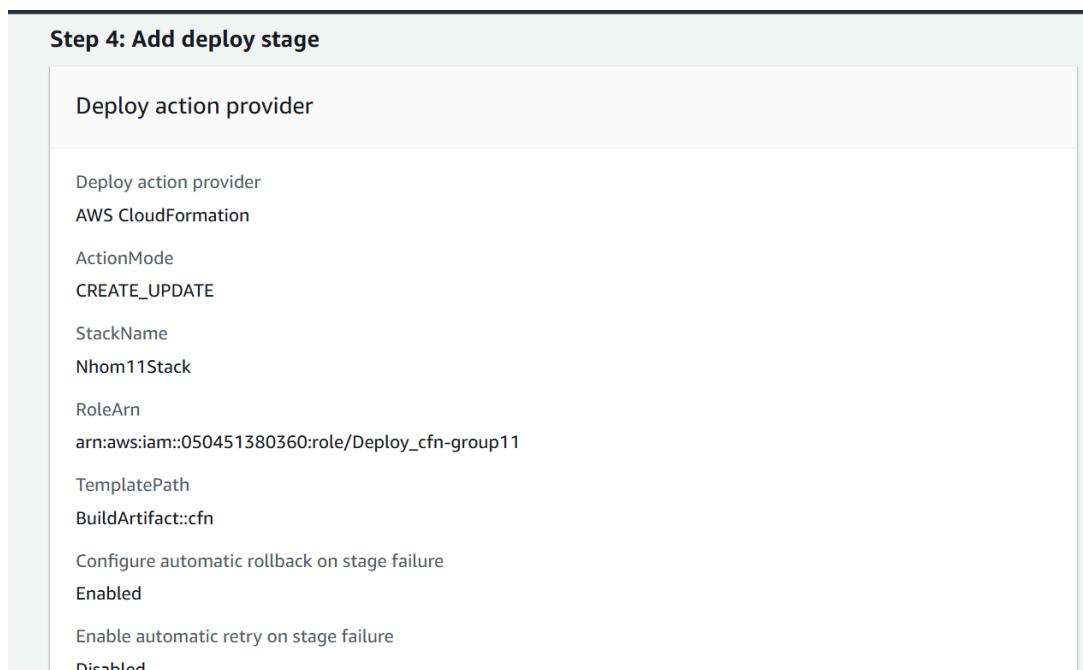
```
Source action provider  
  
Source action provider  
AWS CodeCommit  
  
RepositoryName  
group11  
  
Default branch  
main  
  
PollForSourceChanges  
false  
  
OutputArtifactFormat  
CODE_ZIP  
  
Enable automatic retry on stage failure  
Enabled
```

Hình 34. Bước 2

Step 3: Add build stage

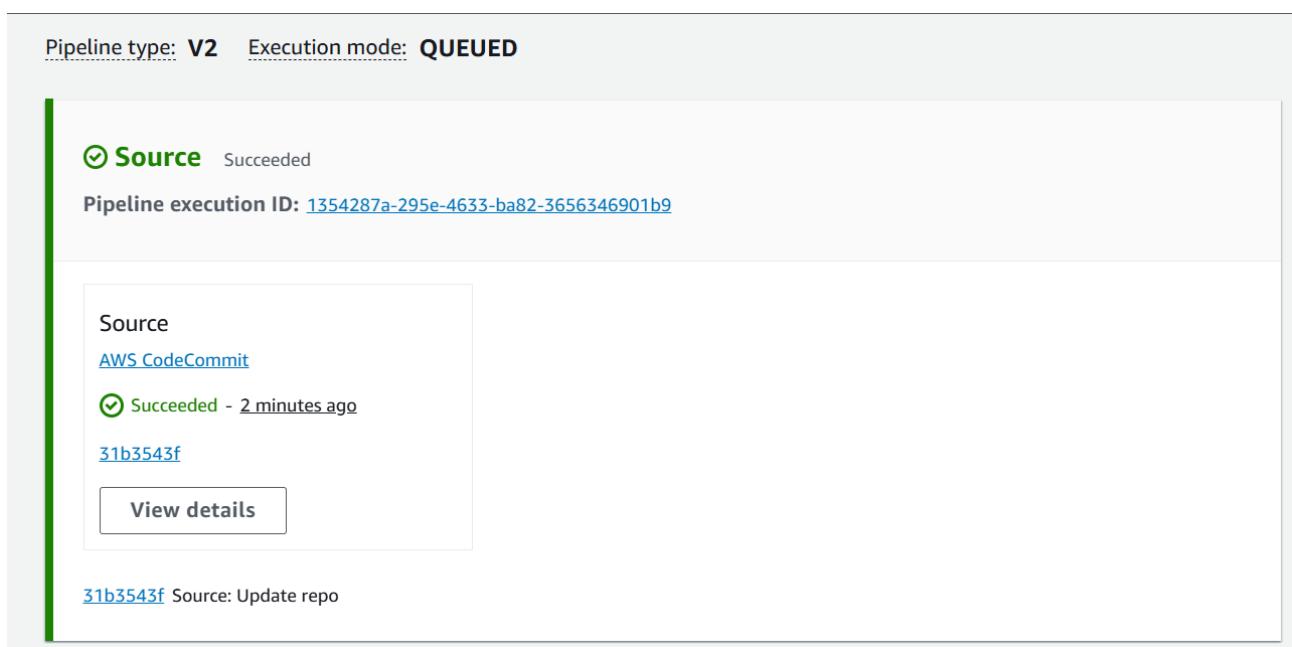
```
Build action provider  
  
Build action provider  
AWS CodeBuild  
  
ProjectName  
group11  
  
BatchEnabled  
false  
  
Commands  
-  
  
Enable automatic retry on stage failure  
Enabled
```

Hình 35. Bước 3

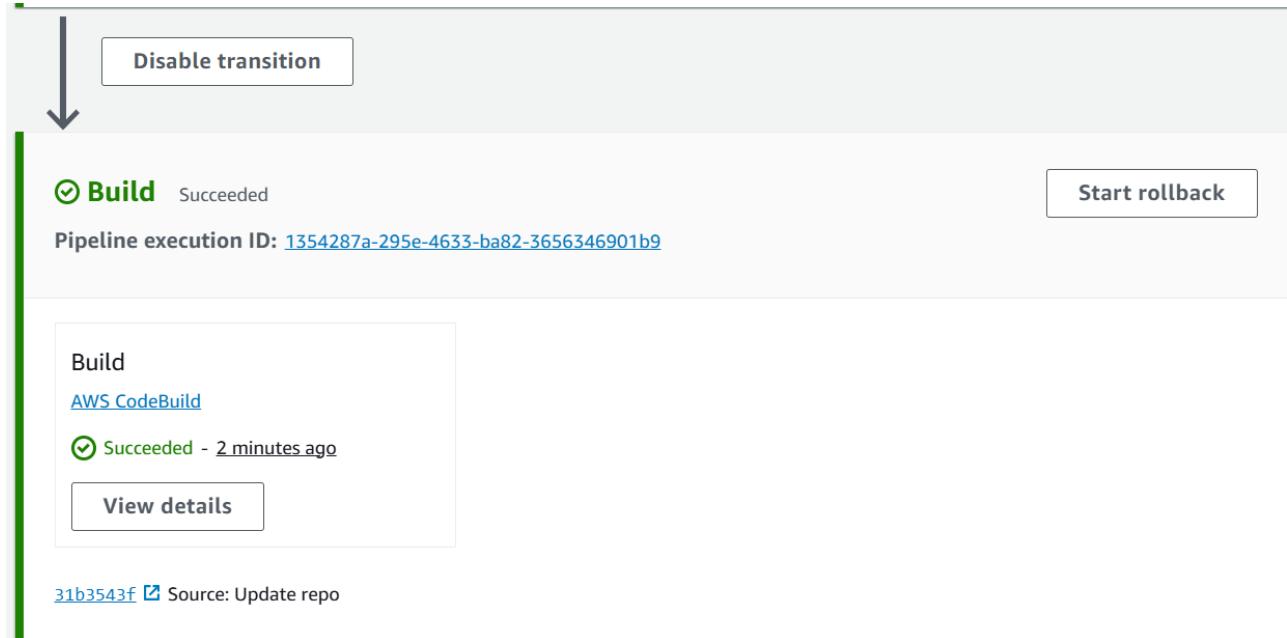


Hình 36. Bước 4

Ở hình 37, 38 và 39, có thể thấy được phần source, build và deploy lần lượt đều đã được triển khai thành công trên AWS CodePipeline.



Hình 37. Source Pipeline



Hình 38. Build pipeline



Hình 39. Deploy pipeline

Pipeline đã được triển khai thành công (hình 40)

Pipelines					Info		<a href="#">View history</a>	<a href="#">Release change</a>	<a href="#">Delete pipeline</a>	<a href="#" style="background-color: orange; color: white; border: 1px solid orange;">Create pipeline</a>
		Name	Latest execution status	Latest source revisions	Latest execution started	Most recent executions				
<input type="radio"/>	<a href="#">pipeline-group11</a> (Type: V2   Execution mode: QUEUED)		Succeeded	Source - <a href="#">31b3543f</a> : Update repo	10 minutes ago		<a href="#">View details</a>			

Hình 40. Pipeline-group11 được triển khai thành công

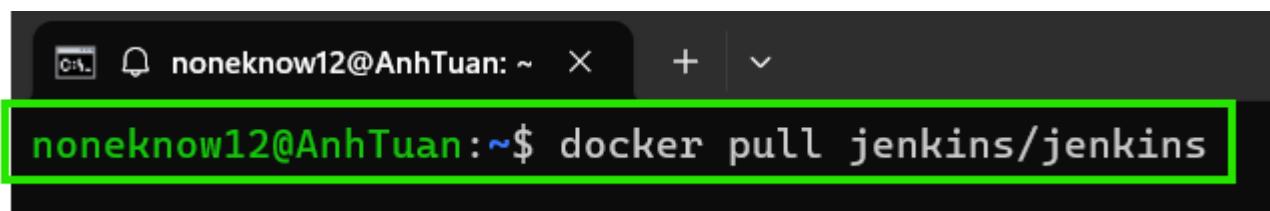
### 3. Sử dụng Jenkins để quản lý quy trình CI/CD cho ứng dụng microservices

- a. Sử dụng Jenkins để tự động hóa quá trình build, test và deploy ứng dụng miroservices lên Docker.

\* Cài đặt và cấu hình môi trường:

Jenkins: cài đặt sử dụng Docker Container.

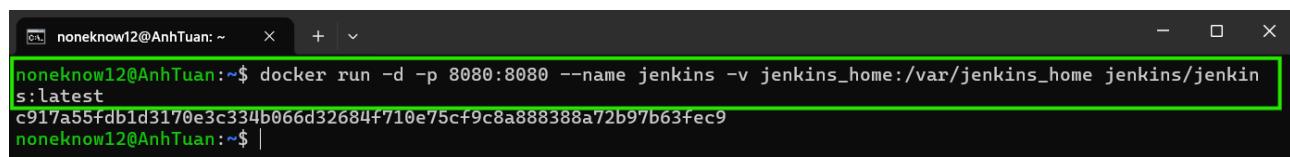
Đầu tiên pull images của jenkins từ Docker Hub:



```
noneknow12@AnhTuan:~$ docker pull jenkins/jenkins
```

Hình 41. Pull Jenkins từ Docker Hub

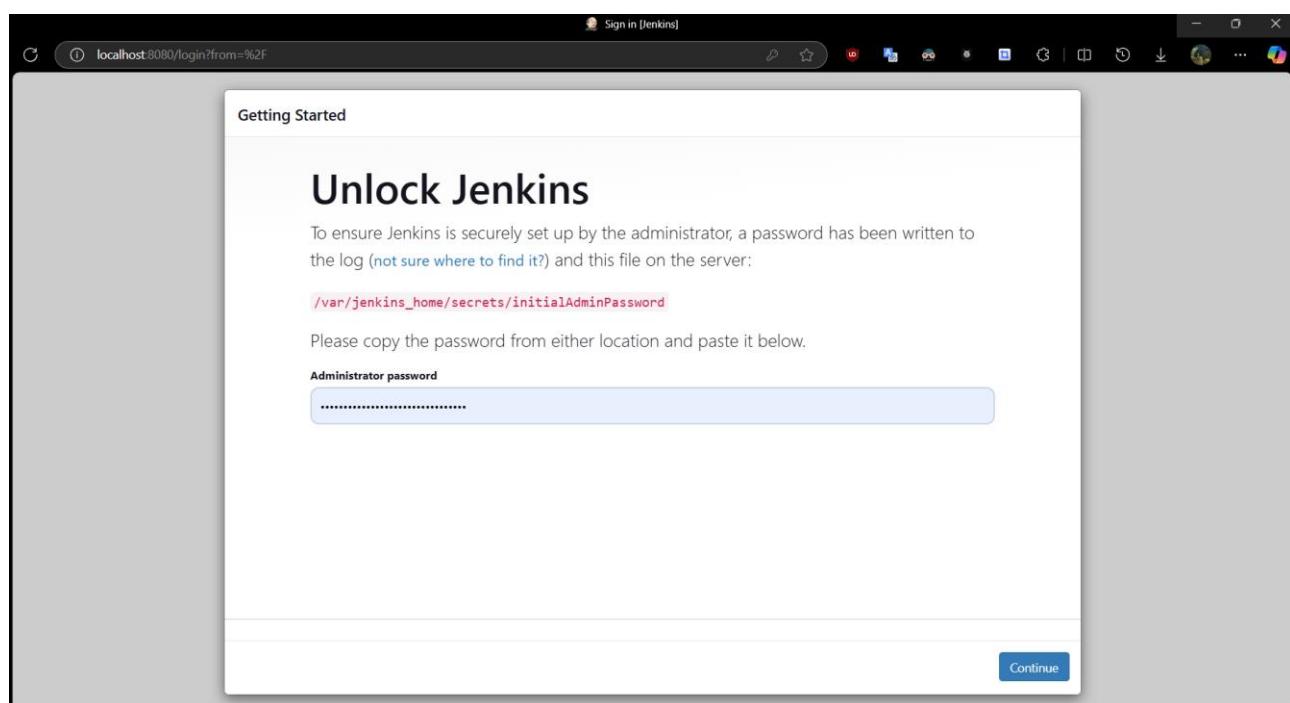
Chạy container từ images mới được pull về:



```
noneknow12@AnhTuan:~$ docker run -d -p 8080:8080 --name jenkins -v jenkins_home:/var/jenkins_home jenkins/jenkins:latest
c917a55fdb1d3170e3c334b066d32684f710e75cf9c8a888388a72b97b63fec9
noneknow12@AnhTuan:~$ |
```

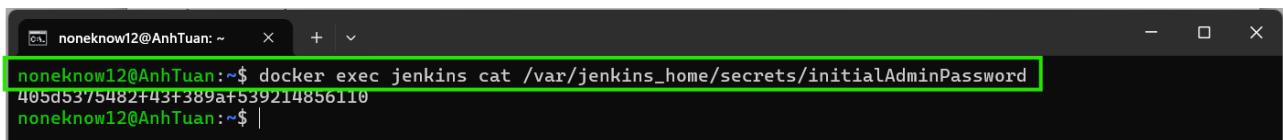
Hình 42. Chạy container

Truy cập vào localhost:8080 để kiểm tra:



Hình 43. Kiểm tra Jenkins đã install thành công hay chưa

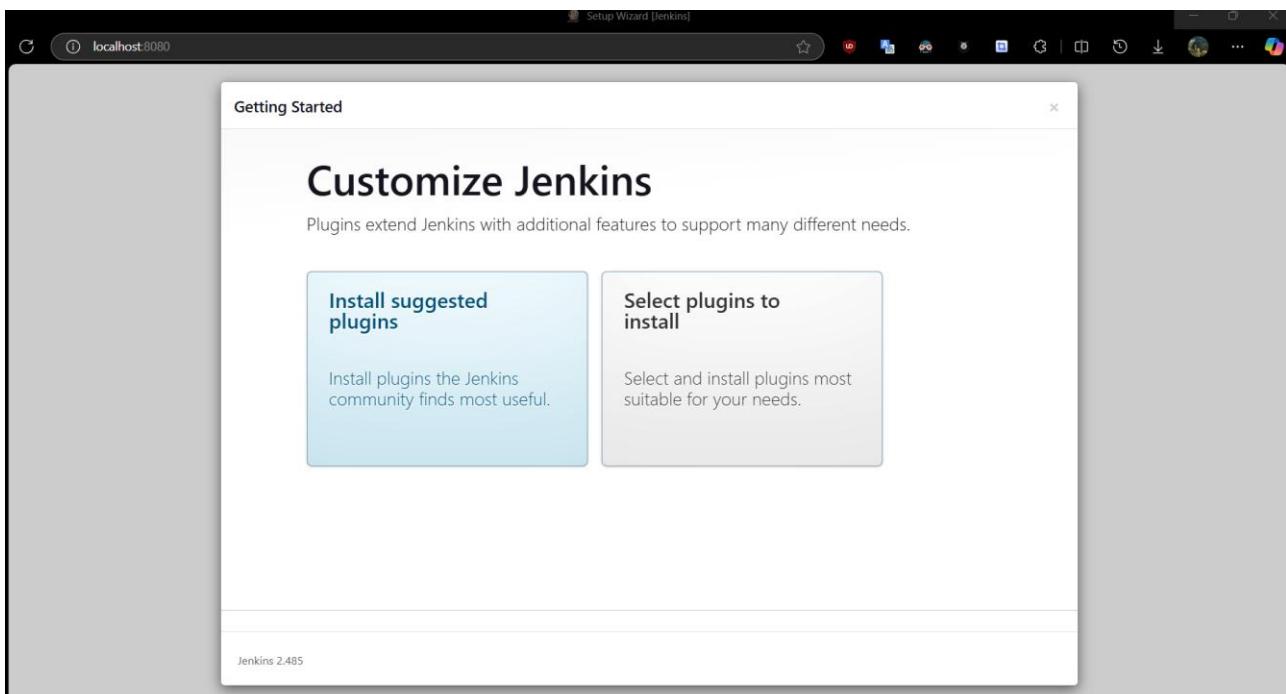
Lấy mật khẩu của Jenkins cung cấp ở trong container:



```
noneknow12@AnhTuan:~$ docker exec jenkins cat /var/jenkins_home/secrets/initialAdminPassword
405d5375482f43f389af539214856110
noneknow12@AnhTuan:~$ |
```

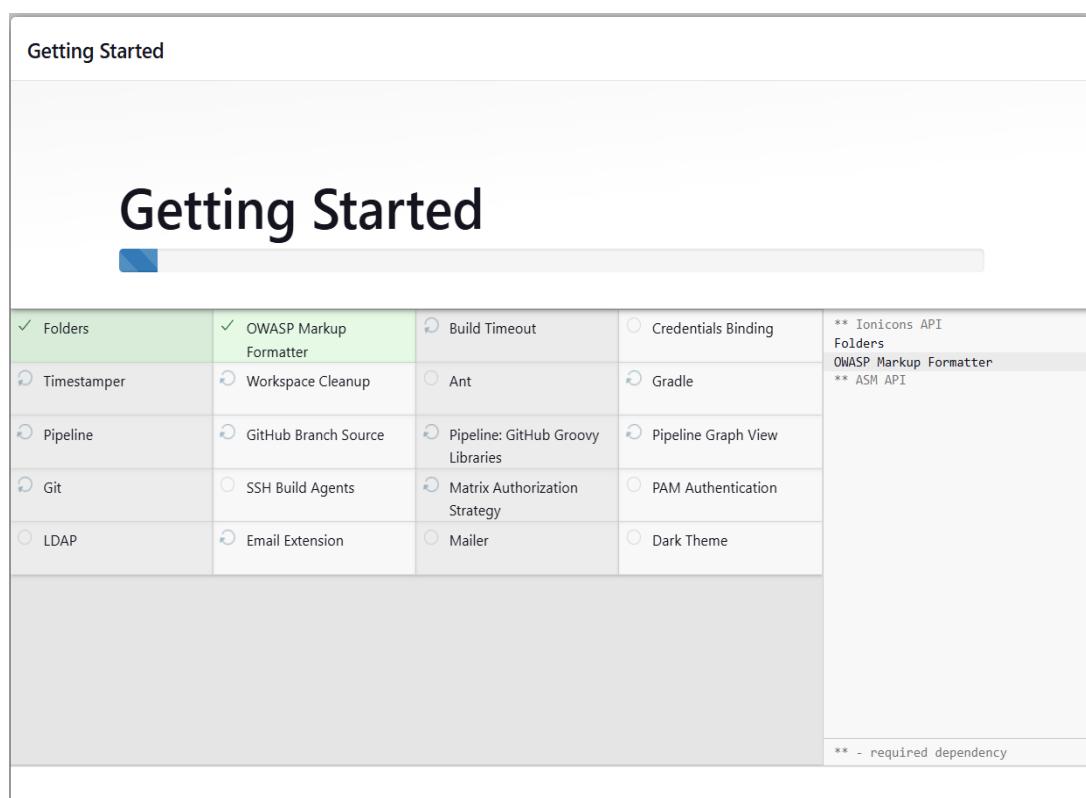
Hình 44. Lấy mật khẩu đăng nhập Jenkins

Chọn Install suggested plugins để cài các plugins khởi đầu:



Hình 45. Config Jenkins

Cài đặt các packet cần thiết:



Hình 46. Cài đặt packets

Tạo tài khoản admin

Getting Started

## Create First Admin User

Username  
admin

Password  
.....

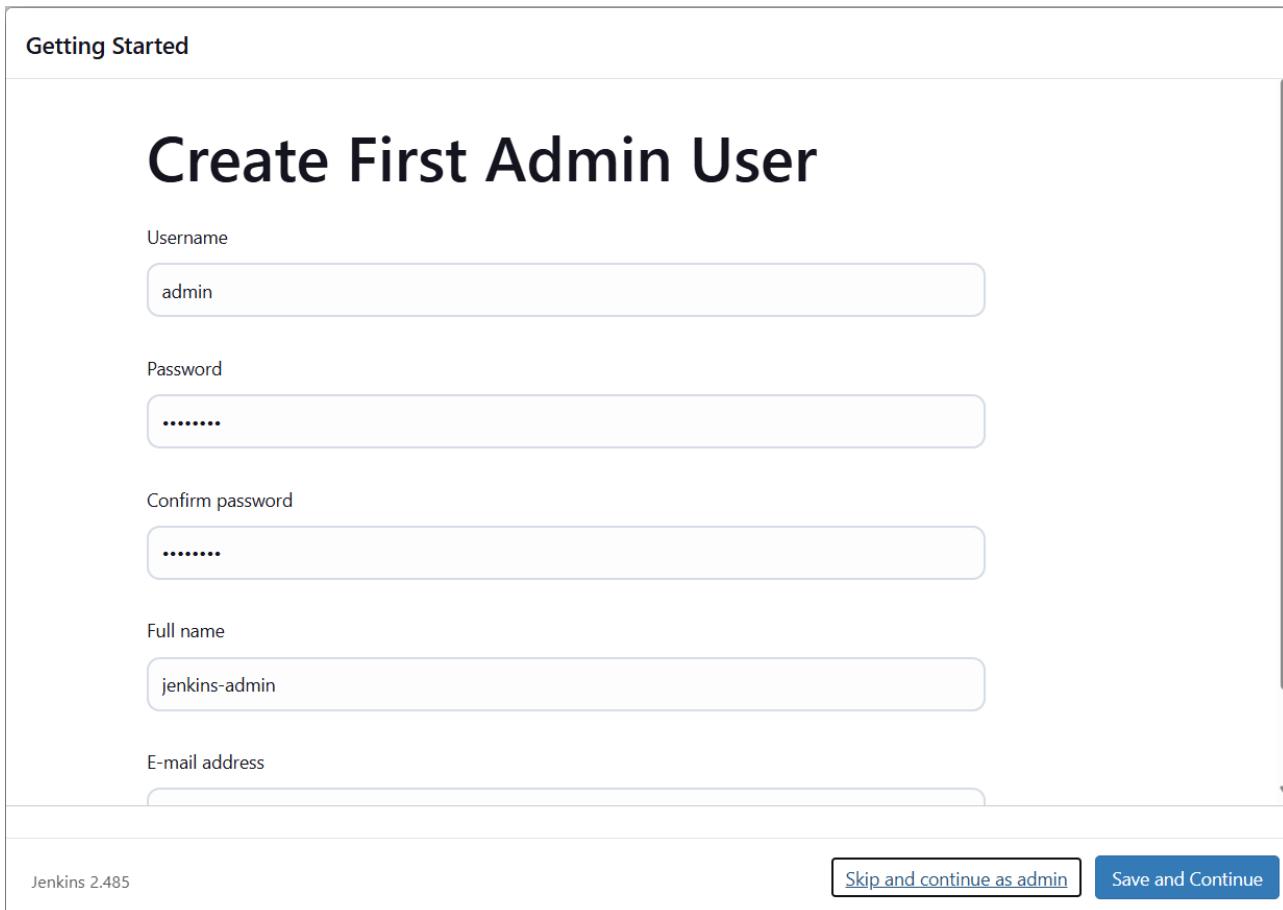
Confirm password  
.....

Full name  
jenkins-admin

E-mail address

Jenkins 2.485

[Skip and continue as admin](#) [Save and Continue](#)



Hình 47. Tạo tài khoản

Jenkins

Dashboard >

+ New Item

Build History

Manage Jenkins

My Views

Build Queue

No builds in the queue.

Build Executor Status

0/2

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job +

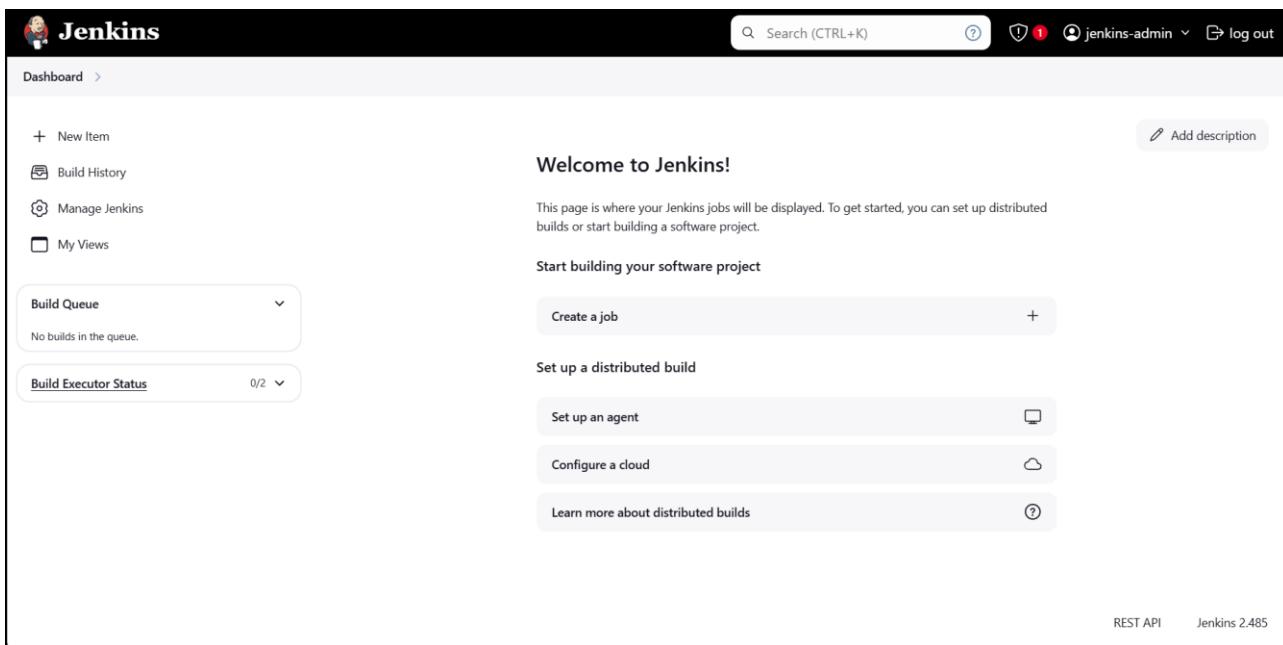
Set up a distributed build

Set up an agent

Configure a cloud

Learn more about distributed builds

REST API Jenkins 2.485



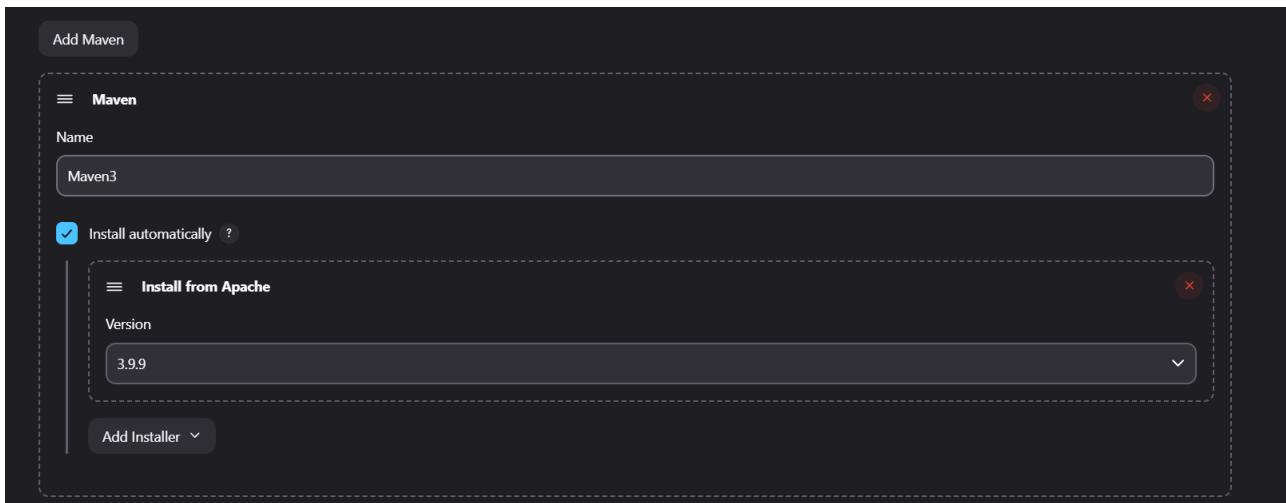
Hình 48. Trang chủ Jenkins

Tổng quan về microservice application:

Java Spring, JDK11, Spring Boot 2

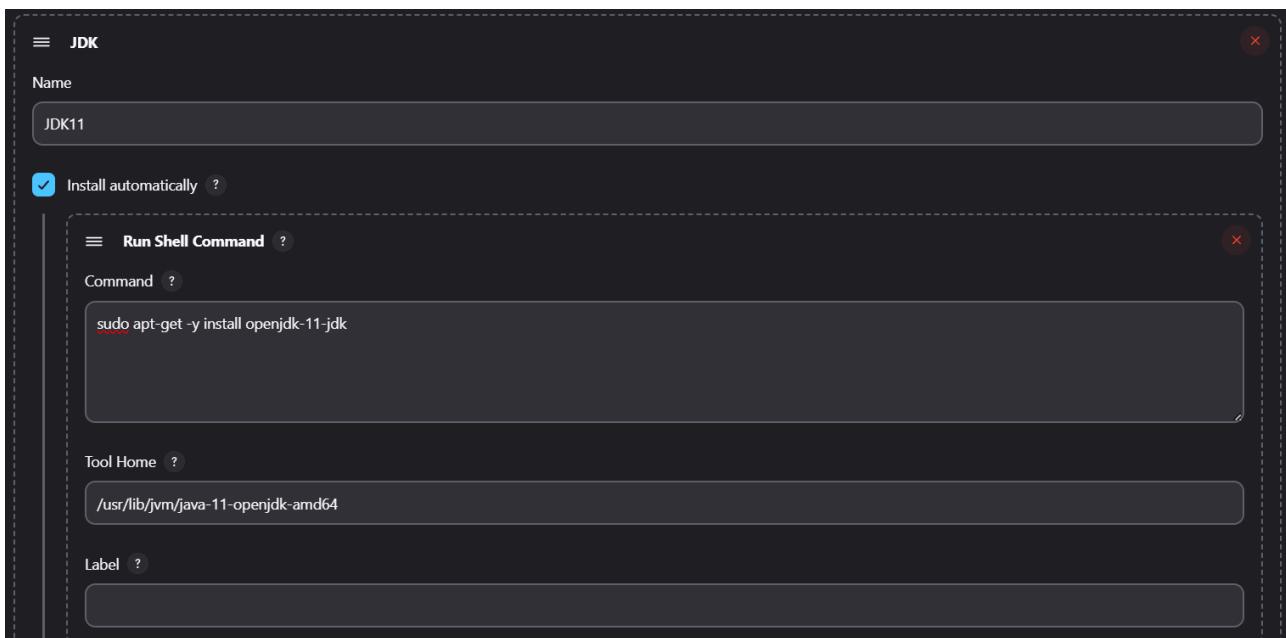
Cấu hình Tool cho jenkins:

Maven:



Hình 49. Cấu hình Maven

JDK11:



Hình 50. Cấu hình JDK11

Cài đặt các plugin cần thiết cho Jenkins:

## 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

40

The screenshot shows the Jenkins Plugins page. On the left, there's a sidebar with options: Dashboard, Manage Jenkins, Plugins, Updates, Available plugins (selected), Installed plugins, Advanced settings, and Download progress. The main area has a search bar at the top right. Below it, a table lists installed plugins:

Install	Name	Released
<input checked="" type="checkbox"/>	Nexus Artifact Uploader 2.14 Artifact Uploaders	1 yr 12 mo ago
<input checked="" type="checkbox"/>	SonarQube Scanner 2.17.2 External Site/Tool Integrations Build Reports	9 mo 1 day ago
<input checked="" type="checkbox"/>	Build Timestamp 1.0.4 Build Wrappers	9 days 11 hr ago
<input checked="" type="checkbox"/>	Pipeline Maven Integration 1457.vf7a_de13b_c064 pipeline Maven	1 mo 15 days ago
<input checked="" type="checkbox"/>	Pipeline Utility Steps 2.18.0 pipeline Build Tools Miscellaneous	1 mo 6 days ago
<input checked="" type="checkbox"/>	Docker Pipeline 580.vc0x340686b_54 pipeline DevOps Deployment docker	5 mo 29 days ago

Hình 51. Cài đặt plugins

### Các Plugins cài đặt thành công

Nexus Artifact Uploader	<span style="color: green;">✓ Success</span>
SonarQube Scanner	<span style="color: green;">✓ Success</span>
Build Timestamp	<span style="color: green;">✓ Success</span>
Pipeline Maven Plugin API	<span style="color: green;">✓ Success</span>
Config File Provider	<span style="color: green;">✓ Success</span>
Pipeline Maven Integration	<span style="color: green;">✓ Success</span>
Commons Compress API	<span style="color: green;">✓ Success</span>
Pipeline Utility Steps	<span style="color: green;">✓ Success</span>
Authentication Tokens API	<span style="color: green;">✓ Success</span>
Docker Commons	<span style="color: green;">✓ Success</span>
Docker Pipeline	<span style="color: green;">✓ Success</span>
SSH server	<span style="color: green;">✓ Success</span>
CloudBees Docker Build and Publish	<span style="color: green;">✓ Success</span>
Loading plugin extensions	<span style="color: green;">✓ Success</span>

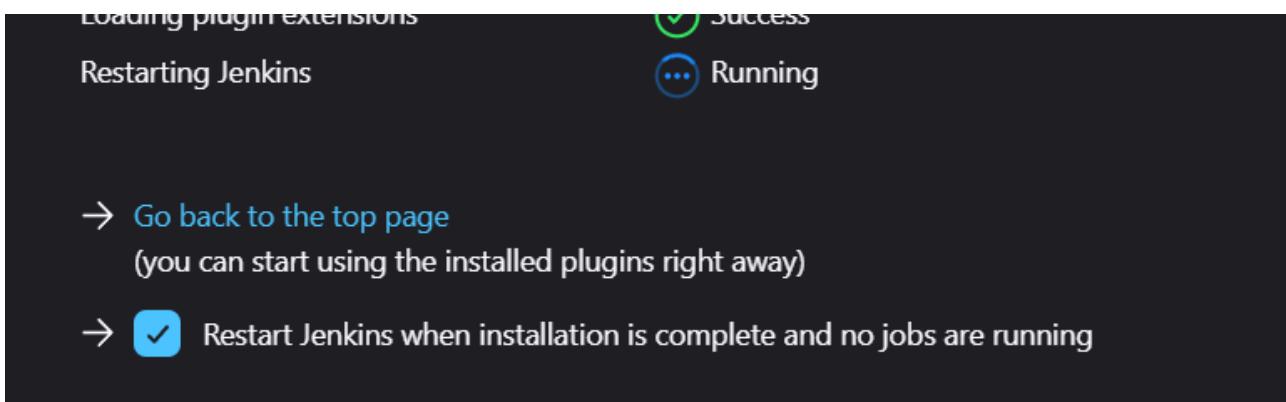
→ [Go back to the top page](#)

(you can start using the installed plugins right away)

→  Restart Jenkins when installation is complete and no jobs are running

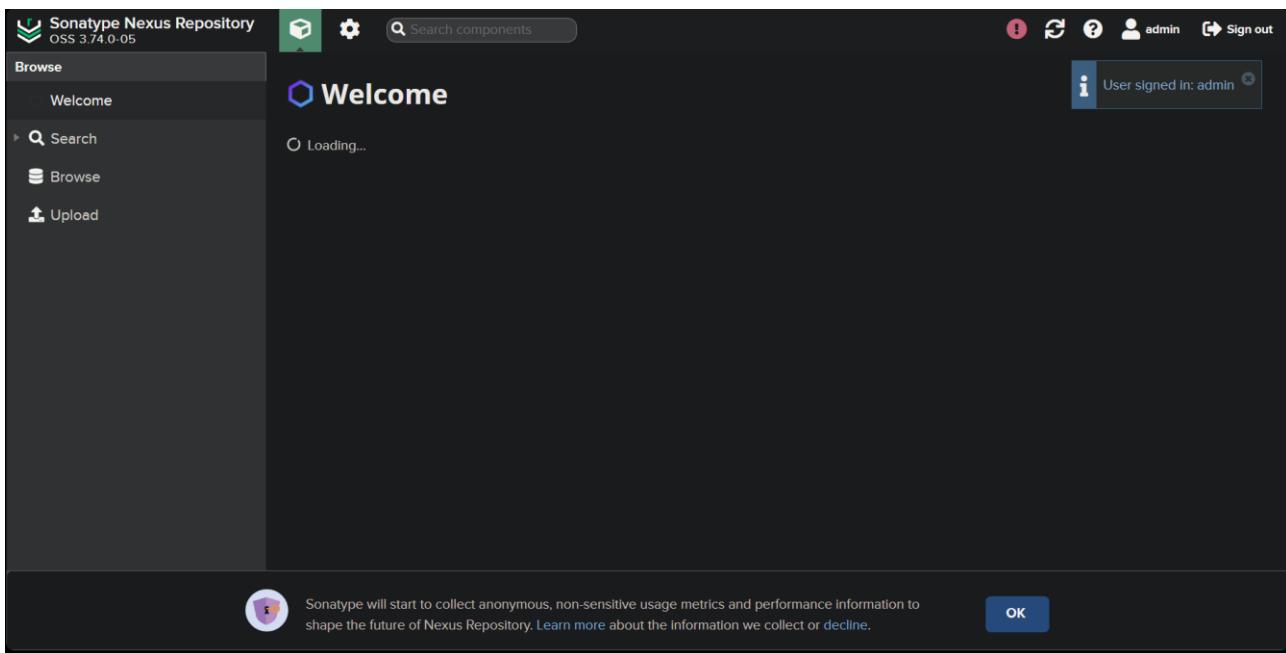
Hình 52. Plugins cài đặt thành công

Sau khi hoàn tất cài đặt plugins, restart Jenkins server



Hình 53. Jenkins restart

\* Cấu hình Nexus repo:



Hình 54. Cấu hình Nexus

Tạo Jenkins Credentials cho Nexus:

## 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

New credentials

Kind: Username with password

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: admin

Treat username as secret:

Password: .....

ID: nexuslogin

Description:

Create

Hình 55. Tạo Jenkins Credentials

Tạo private Docker repo:

Sonatype Nexus Repository OSS 3.74.0-05

Administration

Repository

Repositories

Name: nexus-docker-repo

Online:

Repository Connectors

Connectors allow Docker clients to connect directly to hosted registries, but are not always required. Consult our documentation for which connector is appropriate for your use case. For information on scaling the repositories see our scaling documentation.

HTTP: 8082

HTTPS:

Allow anonymous docker pull:

Docker Registry API Support

Enable Docker V1 API:

Hình 56. Private Docker repo (1)

Sonatype Nexus Repository OSS 3.74.0-05

Administration

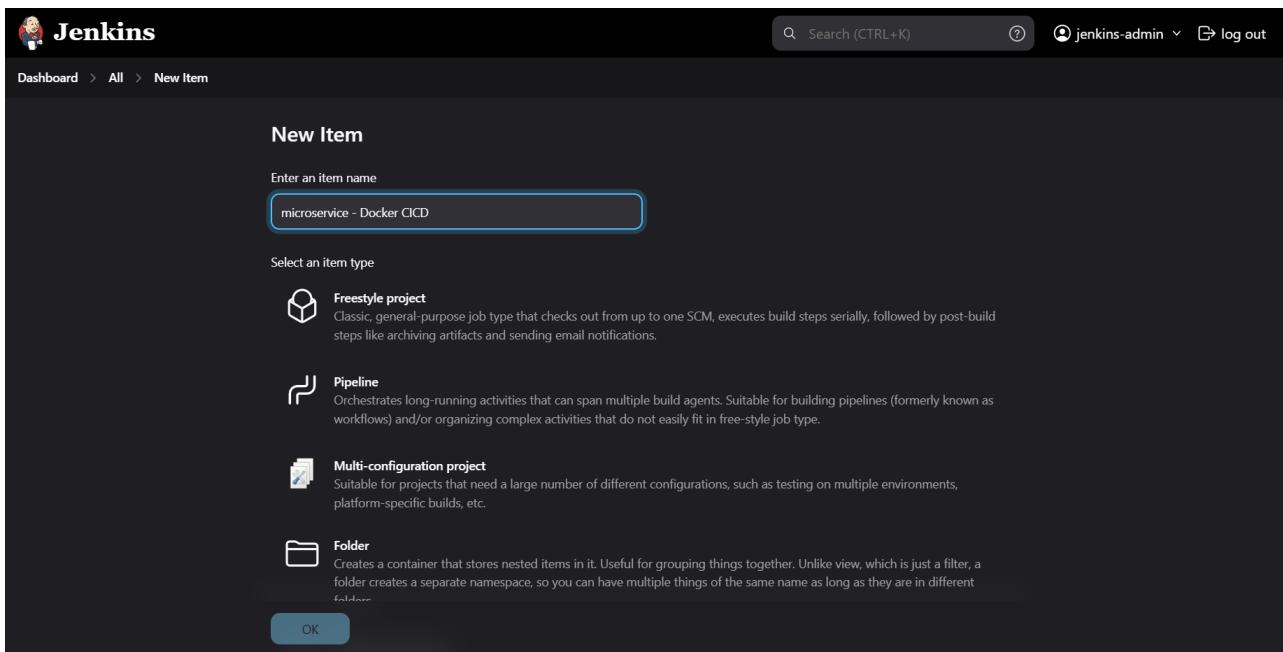
Repository

Repositories

Name	Type	Format	Blob Store	Status	URL	Health check	Firewall Re...
maven-central	proxy	maven2	default	Online - Ready to Co...	<input type="button" value="copy"/>	<input type="button" value="0"/>	<input type="button" value="0"/>
maven-public	group	maven2	default	Online	<input type="button" value="copy"/>	<input type="button" value="0"/>	<input type="button" value="0"/>
maven-releases	hosted	maven2	default	Online	<input type="button" value="copy"/>	<input type="button" value="0"/>	<input type="button" value="0"/>
maven-snapshots	hosted	maven2	default	Online	<input type="button" value="copy"/>	<input type="button" value="0"/>	<input type="button" value="0"/>
nexus-docker-repo	hosted	docker	default	Online	<input type="button" value="copy"/>	<input type="button" value="0"/>	<input type="button" value="0"/>
nuget-group	group	nuget	default	Online	<input type="button" value="copy"/>	<input type="button" value="0"/>	<input type="button" value="0"/>
nuget-hosted	hosted	nuget	default	Online	<input type="button" value="copy"/>	<input type="button" value="0"/>	<input type="button" value="0"/>
nuget.org-proxy	proxy	nuget	default	Online - Ready to Co...	<input type="button" value="copy"/>	<input type="button" value="0"/>	<input type="button" value="0"/>

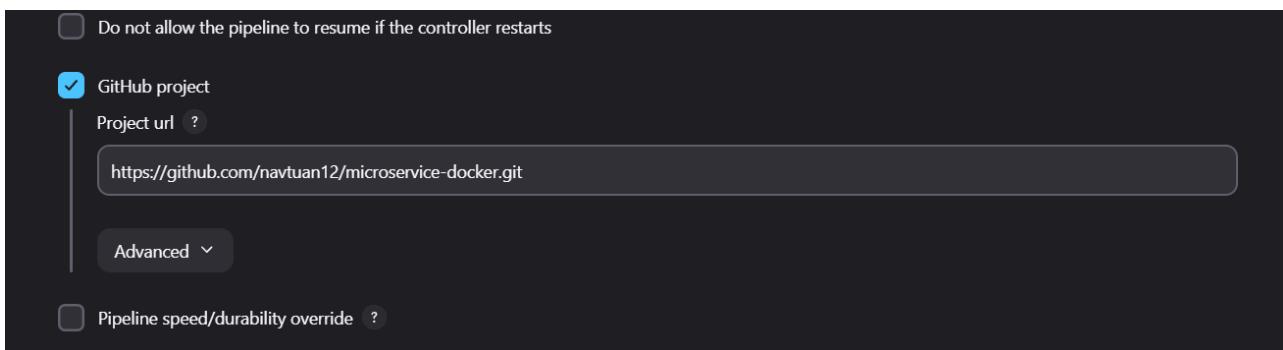
Hình 57. Private Nexus repo (2)

\* Tạo Job CICD:

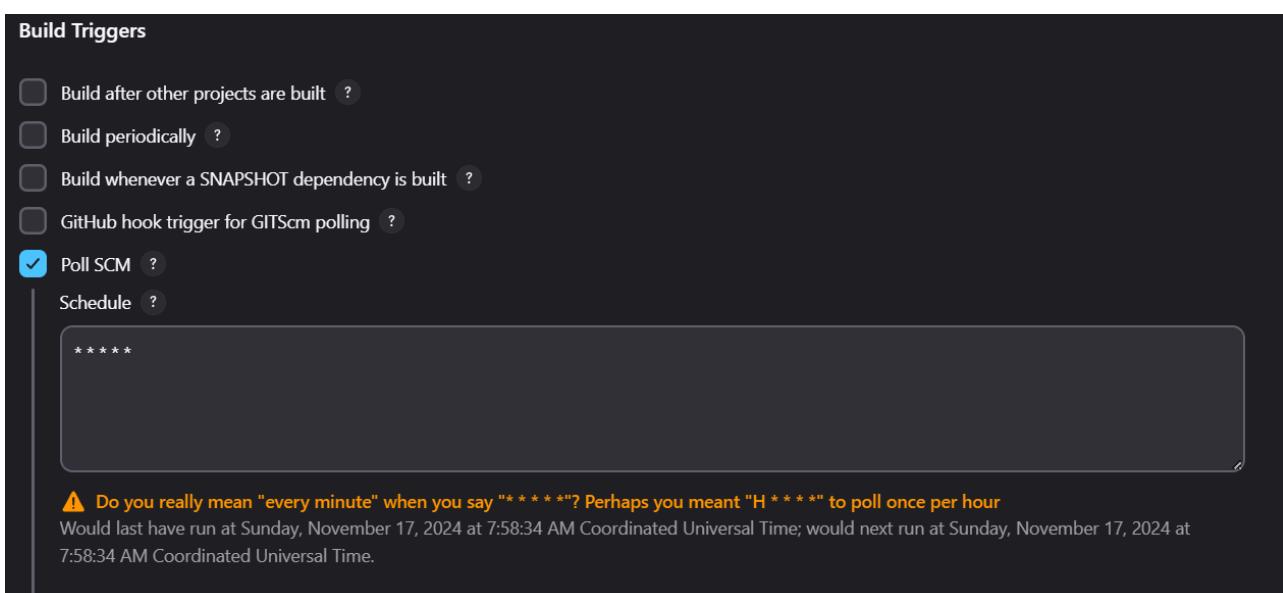


Hình 58. Tạo CI/CD

Add github repo và cấu hình poll scm để trigger khi có thay đổi:

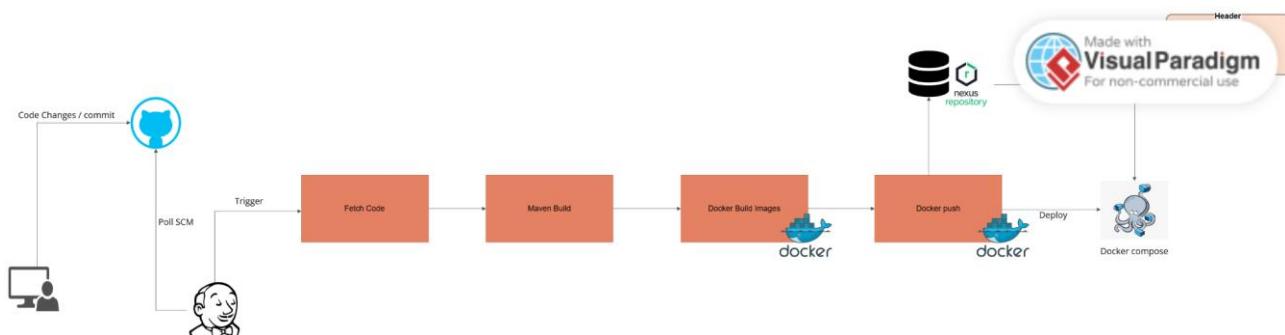


Hình 59. Add Github repo



Hình 60. Setup Schedule

\* Thiết kế pipeline:



Hình 61. Kiến trúc pipeline

```

pipeline {
    agent any

    environment {
        DOCKER_COMPOSE_FILE = 'docker-compose-local.yml'
    }
    stages {
        stage("Fetch code") {
            steps {
                git branch: 'main', url: 'https://github.com/navtuan12/microservice-docker.git'
            }
        }
        stage('Build with Maven') {
            steps {
                script {
                    // Define service directories based on your structure
                    def services = ['config-server', 'discovery-server', 'employee-service', 'organization-service']
                    services.each { service ->
                        dir("${service}") {
                            sh './mvnw clean package -DskipTests' // Adjust if tests should not be skipped
                        }
                    }
                }
            }
        }
        stage('Build Docker Images') {
            steps {
                script {
                    // Build the Docker images for each service
                    def services = ['config-server', 'discovery-server', 'employee-service', 'organization-service'] // Update with your service names
                    services.each { service ->
                        sh "docker build -t 172.26.195.46:8082/repository/nexus-docker-repo/${service}:${env.BUILD_ID} ./${service}"
                    }
                }
            }
        }
    }
}

```

Hình 62. Script run Jenkins

## 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

45

```
stage('Push Docker Images to Local Registry') {
    steps{
        script {
            withCredentials([usernamePassword(credentialsId: 'nexuslogin', passwordVariable: 'PSW', usernameVariable: 'USER')]){
                sh "echo ${PSW} | docker login -u ${USER} --password-stdin http://172.26.195.46:8082"
                def services = ['config-server', 'discovery-server', 'employee-service', 'organization-service'] // Update with your service name
                services.each { service ->
                    sh "docker push 172.26.195.46:8082/repository/nexus-docker-repo/${service}:${env.BUILD_ID}"
                }
            }
        }
    }
}

stage('Deploy with Docker Compose') {
    steps {
        script {
            try {
                // Stop and remove any existing containers
                sh 'docker compose -f docker-compose-local.yml down'
            } catch (Exception e) {
                echo 'No existing containers to stop.'
            }

            try {
                // Pull the latest images (will fail gracefully if images do not exist yet)
                sh 'docker compose -f docker-compose-local.yml pull'
            } catch (Exception e) {
                echo 'No images found in the registry, proceeding with local builds.'
            }
            sh 'docker compose -f docker-compose-local.yml up -d --build --force-recreate --remove-orphans'
        }
    }
}
```

Hình 63. Script run Jenkins

Thay đổi mã nguồn và commit lên github để trigger Jenkins build.

```
noneknow12@AnhTuan:~/microservice-docker$ git add .
noneknow12@AnhTuan:~/microservice-docker$ git commit -m "test: trigger jenkins build"
[main 6abb065] test: trigger jenkins build
 1 file changed, 1 insertion(+), 1 deletion(-)
noneknow12@AnhTuan:~/microservice-docker$ git push origin main
Enumerating objects: 21, done.
Counting objects: 100% (21/21), done.
Delta compression using up to 6 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (11/11), 754 bytes | 754.00 KiB/s, done.
Total 11 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/navtuan12/microservice-docker.git
  3338e13..6abb065  main -> main
noneknow12@AnhTuan:~/microservice-docker$ |
```

Hình 64. Commit lên github

Kiểm tra git polling log:

## 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

The screenshot shows the Jenkins interface with the path: Dashboard > microservice - Docker CICD > Git Polling Log. The main content area displays the 'Git Polling Log' with the following output:

```

Started on Nov 17, 2024, 5:50:22 PM
Using strategy: Default
[poll] Last Built Revision: Revision 3338e130ed0310387e7ea46e86415b0dbc691fc7 (refs/remotes/origin/main)
The recommended git tool is: NONE
No credentials specified
> git --version # timeout=10
> git --version # 'git version 2.34.1'
> git ls-remote -h -- https://github.com/navtuan12/microservice-docker.git # timeout=10
Found 1 remote heads on https://github.com/navtuan12/microservice-docker.git
[poll] Latest remote head revision on refs/heads/main is: 6abb065abfef8302e5c5982c3963d13e71779e5b
Done. Took 0.67 sec
Changes found
  
```

Below the log, there's a 'Build History' section showing a single build entry: '#11 (pending—In the quiet period. Expires in 3.4 sec)'. A tooltip indicates that the build is pending.

Hình 65. Git polling log

Dòng build lần thứ 11 kiểm tra Jenkins build

The screenshot shows the Jenkins build details for Build #11, which was triggered by an SCM change on Nov 17, 2024, at 5:50:30 PM. The 'Changes' section highlights the first commit:

- test: trigger jenkins build (commit: 6abb065) (details / githubweb)

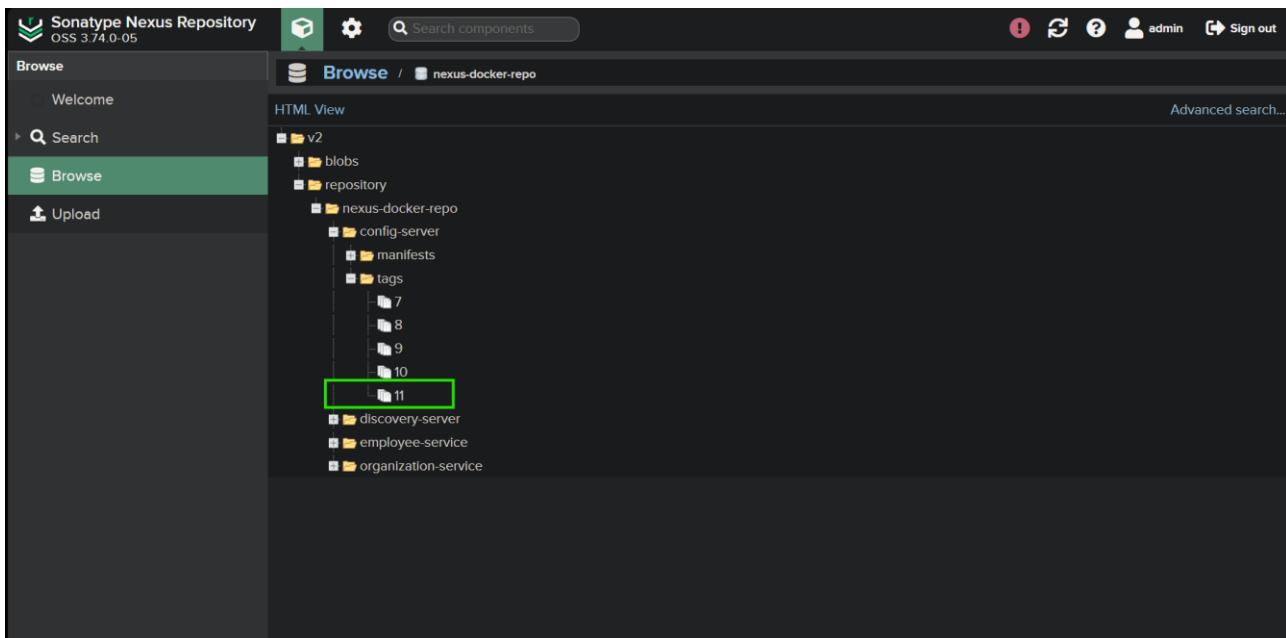
Other build information includes:

- Started by an SCM change**
- This run spent 6.7 sec waiting in the queue.
- Revision:** 6abb065abfef8302e5c5982c3963d13e71779e5b
- Repository:** <https://github.com/navtuan12/microservice-docker.git>
- refs/remotes/origin/main

Hình 66. Check Jenkins build

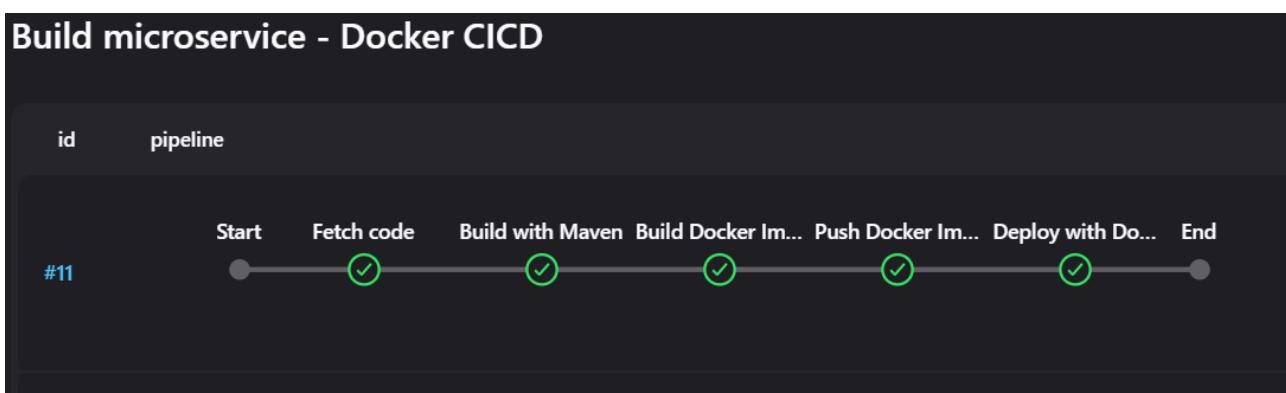
Kiểm tra lần build bên Nexus

## 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



Hình 67. Kiểm tra bên Nexus

Kết quả build, test và deploy thành công



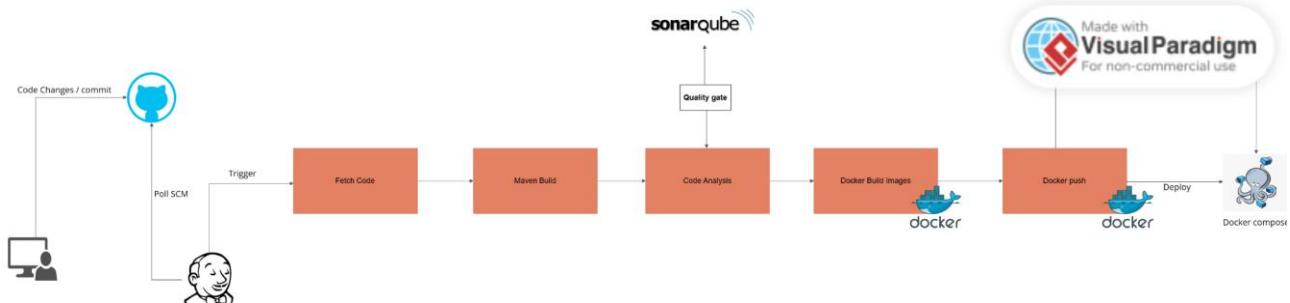
Hình 68. CI/CD chạy thành công

Tất cả container đều được update lên version mới nhất

```
noneknow12@AnhTuan:~/microservice-docker$ docker ps
CONTAINER ID   IMAGE                               COMMAND                  CREATED             STATUS              PORTS
75ccb57f5f3   172.26.195.46:8082/repository/nexus-docker-repo/organization-service:11   "/bin/sh -c 'java -j..."  33 seconds ago    Up 25 seconds   0.0.0.0:8
094->8080/tcp, [::]:8090->8080/tcp          microservice-dockercicd-organization-service-1
781c2523d9ee  172.26.195.46:8082/repository/nexus-docker-repo/employee-service:11   "/bin/sh -c 'java -j..."  34 seconds ago    Up 27 seconds   0.0.0.0:8
092->8080/tcp, [::]:8092->8080/tcp          microservice-dockercicd-employee-service-1
123aaadb8b8  172.26.195.46:8082/repository/nexus-docker-repo/discovery-server:11   "/bin/sh -c 'java -j..."  37 seconds ago    Up 28 seconds   0.0.0.0:8
090->8080/tcp, [::]:8090->8080/tcp          microservice-dockercicd-discovery-server-1
f67499eee24f  172.26.195.46:8082/repository/nexus-docker-repo/config-server:11   "/bin/sh -c 'java -j..."  39 seconds ago    Up 30 seconds   0.0.0.0:8
088->8080/tcp, [::]:8088->8080/tcp          microservice-dockercicd-config-server-1
b3at8d374800  sonatype/nexus3:latest           "/opt/sonatype/nexus..."  5 days ago        Up 4 hours     0.0.0.0:8
081-8082->8081-8882/tcp, :::8081-8082->8081-8082/tcp  nexus
6bde25d4b782  sonarqube:latest                "/opt/sonarqube/dock..."  7 days ago        Up 7 hours     0.0.0.0:9
noneknow12@AnhTuan:~/microservice-docker$ |
```

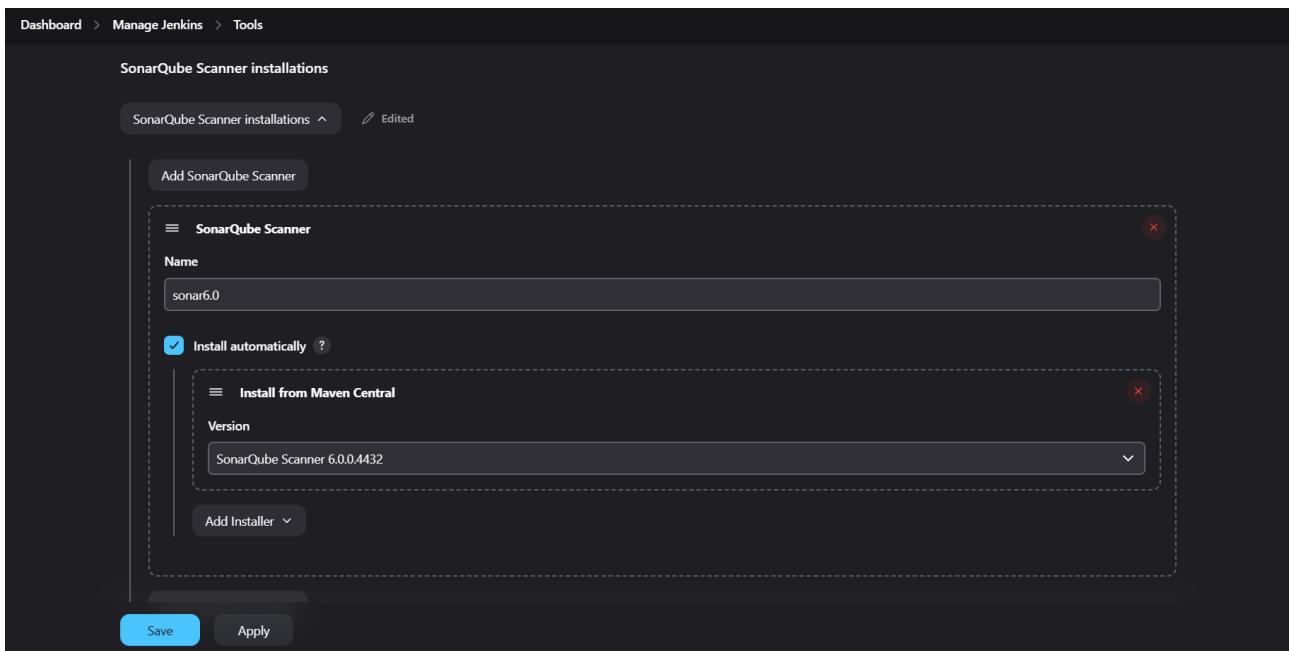
Hình 69. Kiểm tra trạng thái các container

b. Tích hợp SonarQube để kiểm tra chất lượng mã nguồn.



Hình 70. Kiến trúc pipeline tích hợp SonarQube

Add SonarQube Scanner trong Manager > Tools:



Hình 71. Add SonarQube trên Jenkins

Thêm SonarQube credentials:

Đầu tiên truy cập vào SonarQube để tạo secret key

## 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

49

The screenshot shows the SonarQube interface under the 'Administrator' tab. In the 'Security' section, there's a form for generating tokens. It has fields for 'Name' (jenkins), 'Type' (Global Analysis Token), and 'Expires in' (No expiration). A 'Generate' button is present. Below this, a table shows token details with one row labeled 'No tokens'. At the bottom, there's a 'Enter a new password' field and a note about required fields.

Hình 72. Lấy SonarQube credentials

Tạo secret key credentials trong jenkins và bỏ key vừa tạo vào:

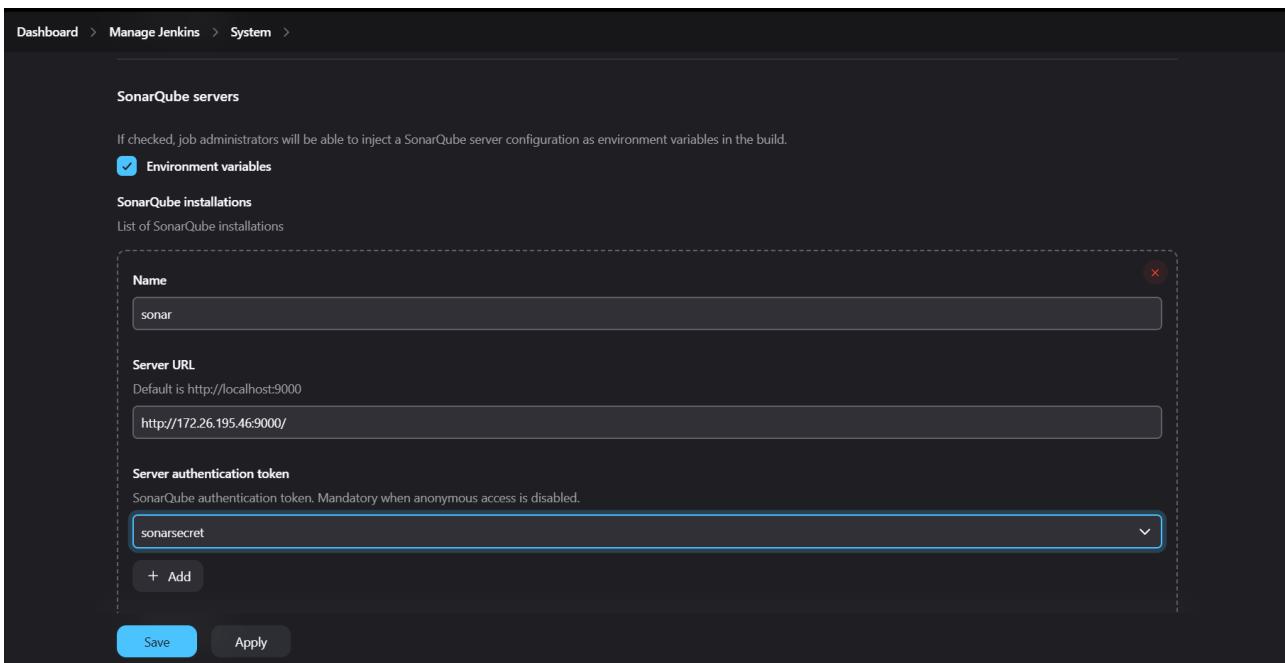
The screenshot shows the Jenkins 'New credentials' creation form. It includes fields for 'Kind' (Secret text), 'Scope' (Global), 'Secret' (containing a redacted value), 'ID' (sonarsecret), and 'Description' (sonarsecret). A 'Create' button is at the bottom.

Hình 73. Tạo key

Thêm sonar server trong Manage > System, sử dụng credentials key vừa tạo:

## 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

50



Hình 74. Sử dụng key vừa tạo

Thêm stage vào JenkinsFile:

```
stage("Sonar Analysis") {
    environment {
        scannerHome = tool 'sonar6.0'
    }

    steps {
        withSonarQubeEnv('sonar') {
            script {
                // Define services to analyze
                def services = ['config-server', 'discovery-server', 'employee-service', 'organization-service']

                // Loop through each service for SonarQube analysis
                services.each { service ->
                    echo "Running SonarQube analysis for ${service}"
                    sh """
                    ${scannerHome}/bin/sonar-scanner \
                    -Dsonar.projectKey=${service} \
                    -Dsonar.projectName=${service} \
                    -Dsonar.projectVersion=1.0 \
                    -Dsonar.sources=${service}/src/main/java \
                    -Dsonar.java.binaries=${service}/target/classes \
                    -Dsonar.java.coveragePlugin=jacoco \
                    -Dsonar.junit.reportPaths=${service}/target/surefire-reports \
                    -Dsonar.jacoco.reportPaths=${service}/target/jacoco.exec \
                    -Dsonar.java.checkstyle.reportPaths=${service}/target/checkstyle-result.xml
                    """
                }
            }
        }
    }
}
```

Hình 75. Thêm Stage

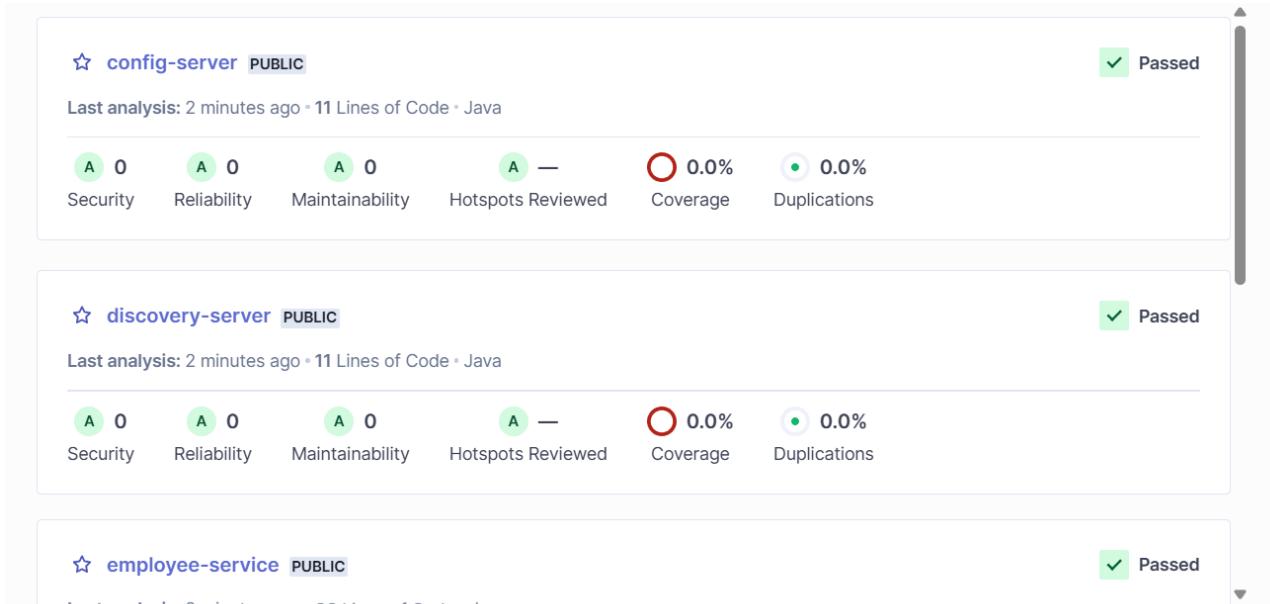
Kết quả:

Kiểm tra thành công tính đúng đắn của mã nguồn trên SonarQube. (Hình 76)

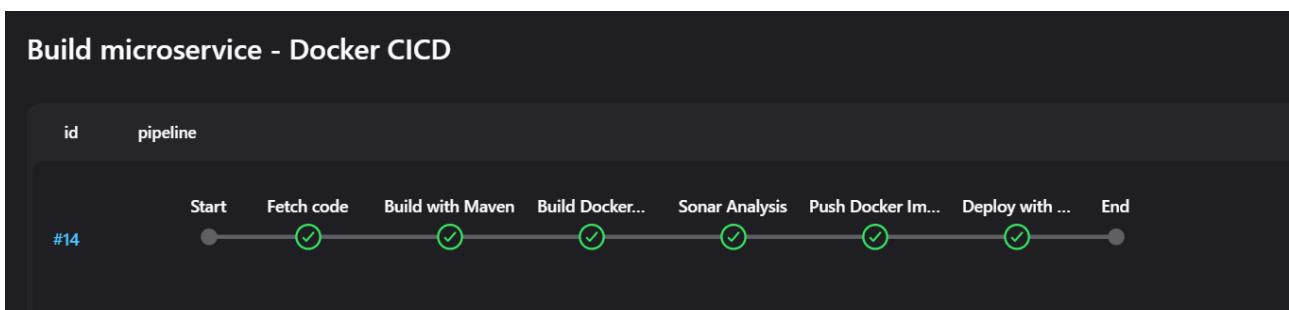
Chạy lại Pipeline vừa tích hợp trên SonarQube để kiểm tra. (Hình 77)

## 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

51



Hình 76. Kiểm tra trên SonarQube



Hình 77. Tích hợp thành công