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** |
| **9.5** |

**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 | <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

Contents

[**A.** BÁO CÁO CHI TIẾT 2](#_Toc182766363)

[1. Triển khai hạ tầng AWS sử dụng Terraform và tự động hóa quy trình với GitHub Actions 2](#_Toc182766364)

[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](#_Toc182766365)

[b. Tự động hóa quá trình triển khai với GitHub Actions 6](#_Toc182766366)

[c. Tích hợp Checkov để kiểm tra tính tuân thủ và bảo mật của mã nguồn Terraform. 8](#_Toc182766367)

[2. Triển khai hạ tầng AWS với CloudFormation và tự động hóa quy trình build và deploy với AWS CodePipeline 11](#_Toc182766368)

[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. 11](#_Toc182766369)

[b. Sử dụng AWS CodeBuild, tích hợp cfn-lint và Taskcat để kiểm tra tính đúng đắn của mã CloudFormation. 16](#_Toc182766370)

[c. Sử dụng AWS CodePipeline để tự động hóa quy trình build và deploy từ mã nguồn trên CodeCommit. 24](#_Toc182766371)

[3. Sử dụng Jenkins để quản lý quy trình CI/CD cho ứng dụng microservices 32](#_Toc182766372)

[a. Sử dụng Jenkins để tự động hóa quá trình build, test và deploy ứng dụng miroservices lên Docker. 32](#_Toc182766373)

[b. Tích hợp SonarQube để kiểm tra chất lượng mã nguồn. 44](#_Toc182766374)

# 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"

}

Tạo VPC

# Tạo VPC

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

cidr\_block = "10.0.0.0/16"

tags = {

Name = "Nhom11-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 = "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"

}

}

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 = "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"

}

}

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 = "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"

}

}

Tạo 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

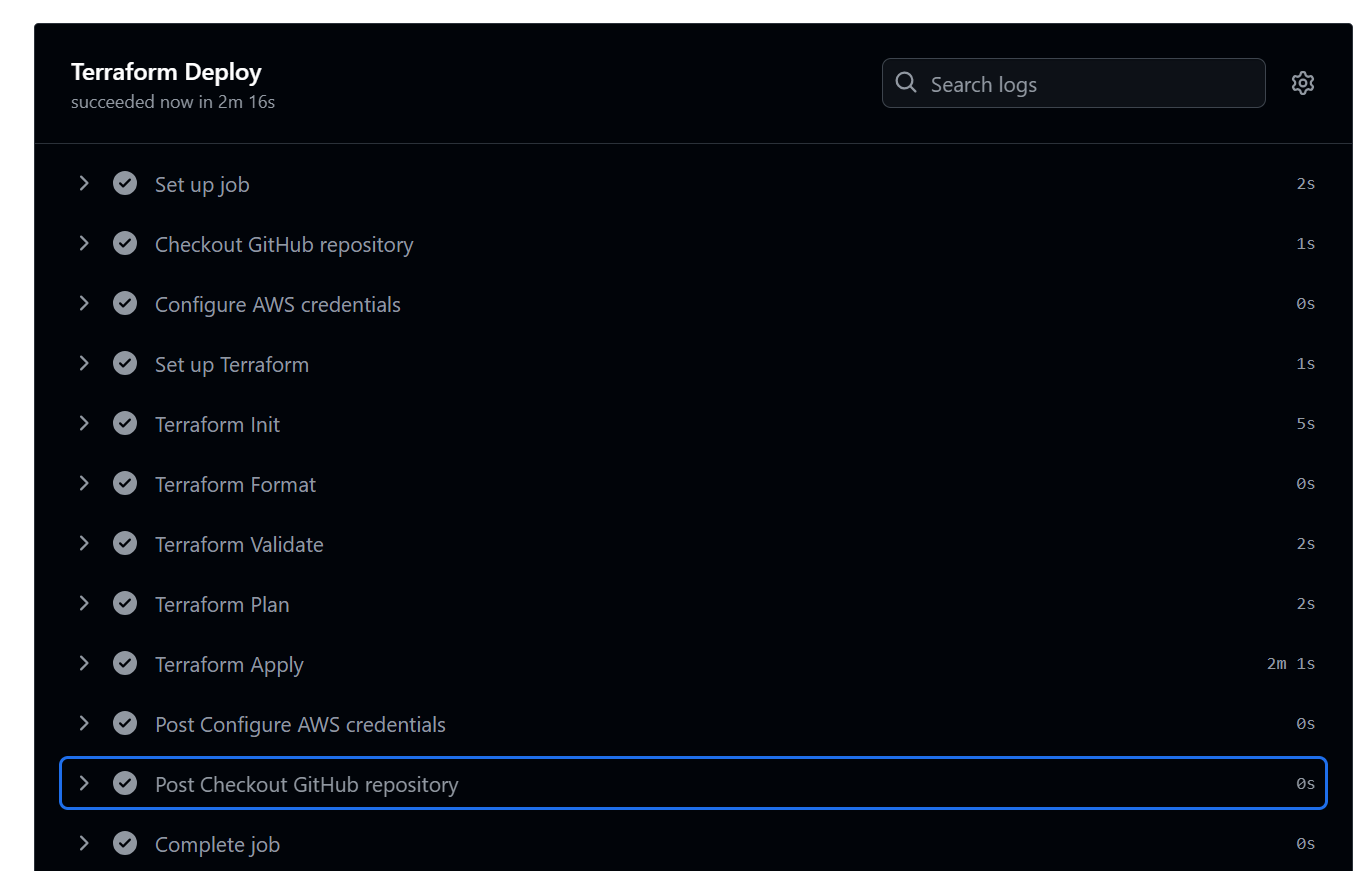
- name: 'Terraform Apply'

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: |

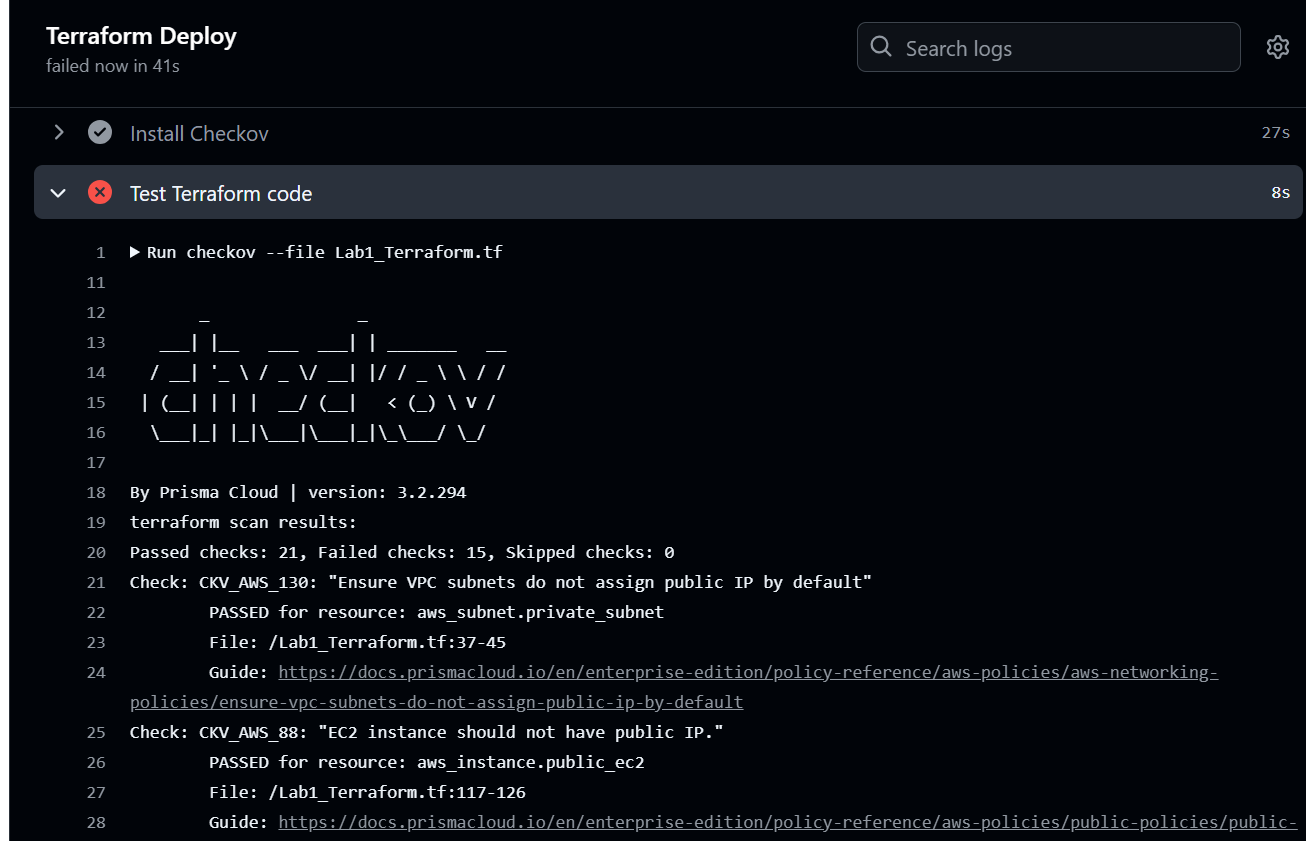
pip3 install checkov

checkov --version

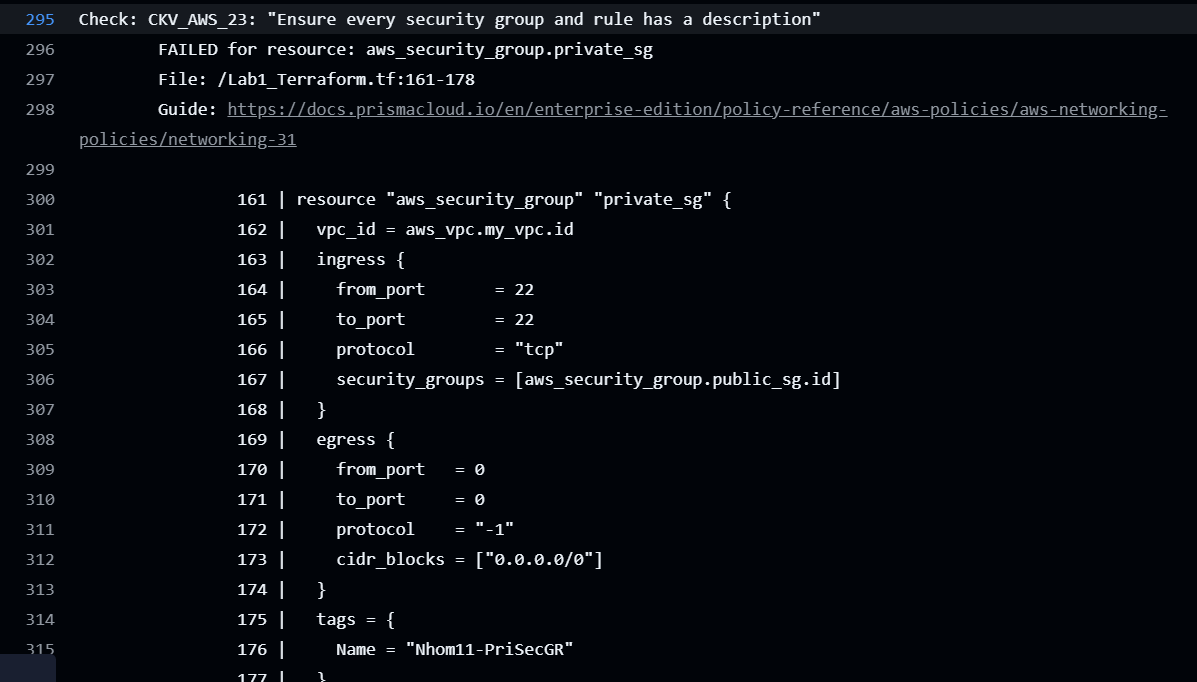
- name: "Test Terraform code"

run: checkov --file Lab1\_Terraform.tf

Code hiện tại đang cần cải tiến vì 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.



1. Chạy checkov kiểm tra code trước khi sửa



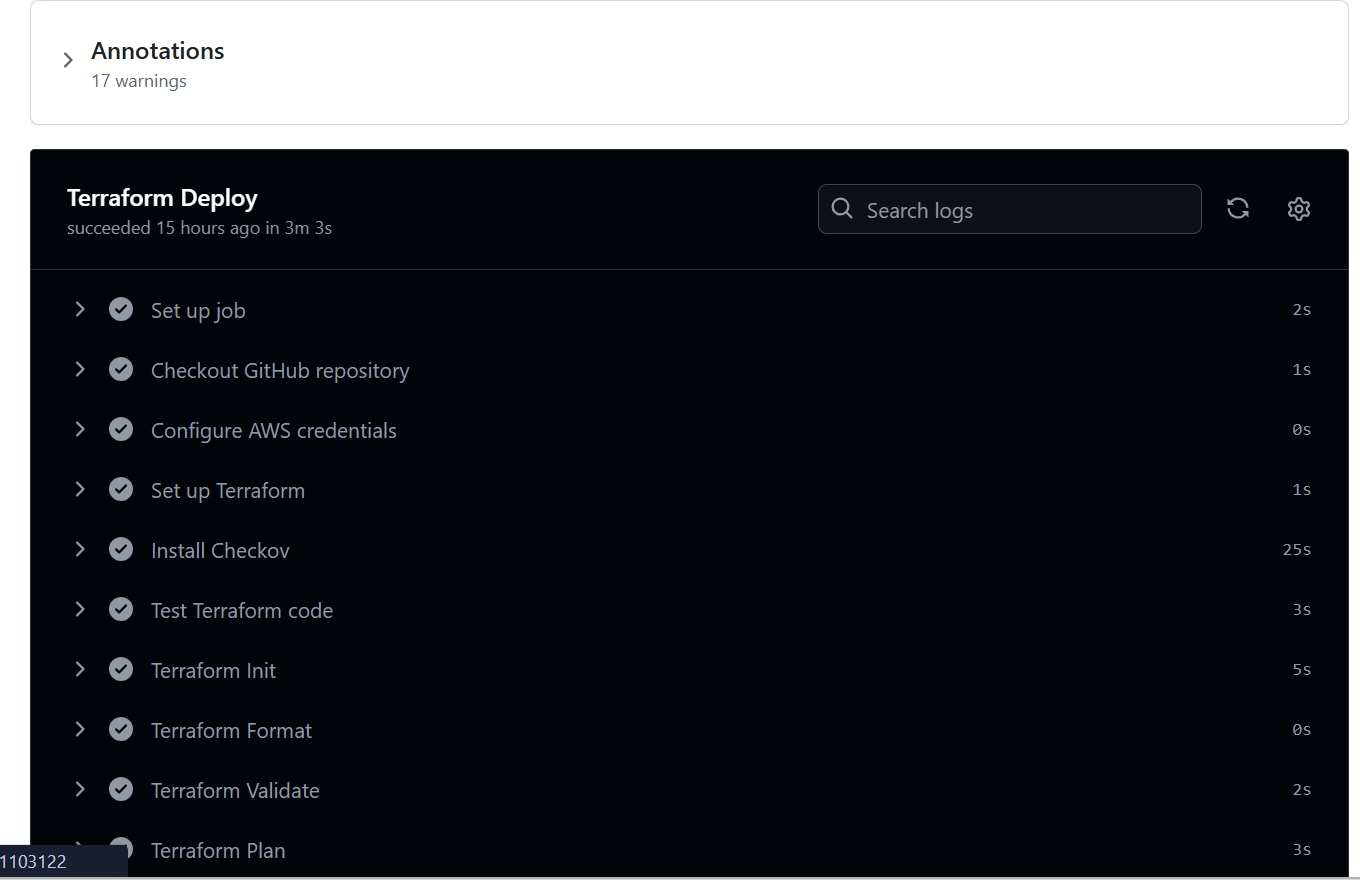
1. Ví dụ về lỗi báo khi sử dụng Checkov scan code

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.

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 thểtruy cập từ Internet.
* Private instance chỉ có thể truy cập từ Public instance thôngqua 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

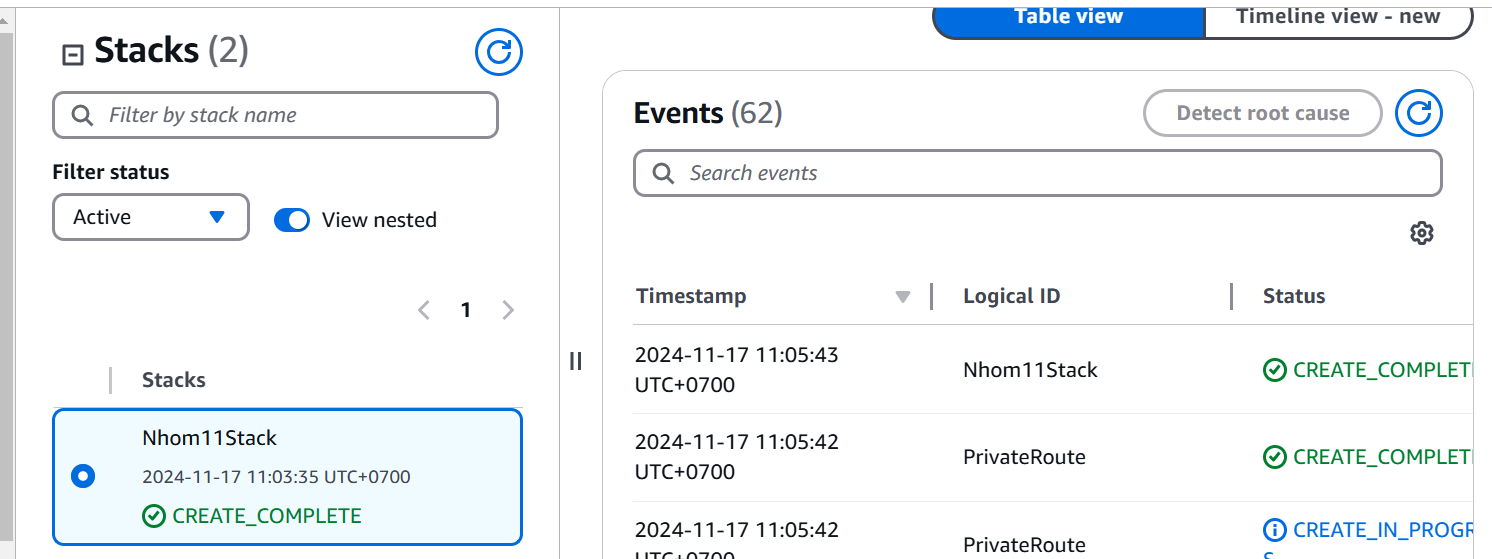
ImageId: ami-0866a3c8686eaeeba #Insert your ID

Tags:

- Key: Name

Value: !Join ['', [!Ref "AWS::StackName", "-Pri-EC2"]]

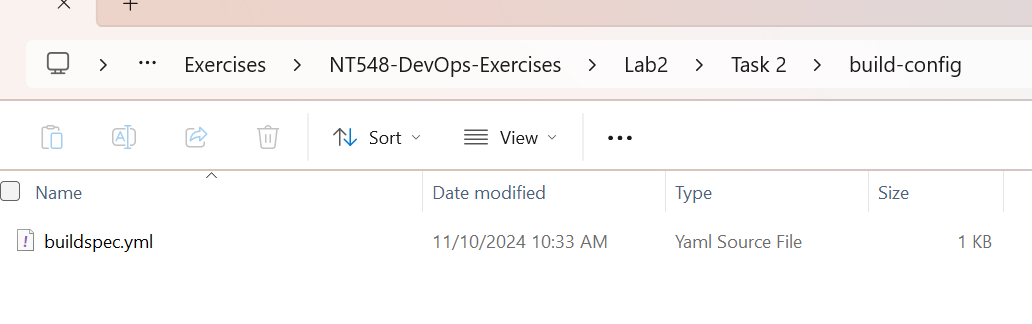
Kết quả tạo CloudFormation thành công trên AWS Console



1. CloudFormation trên AWS Console

### 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".



1. 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:

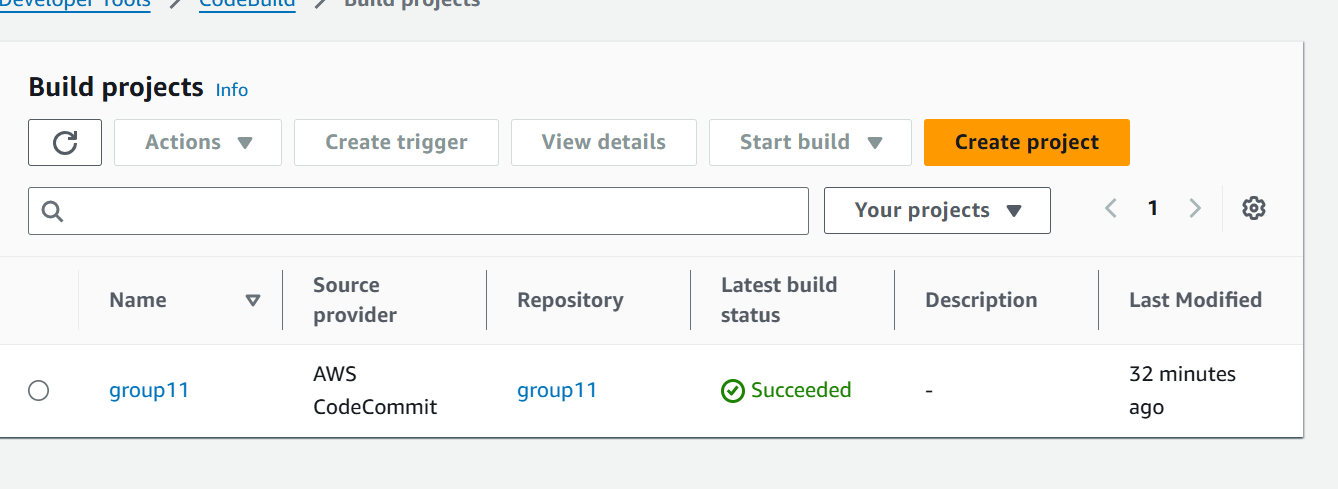
EnvironmentName: TestEnvironment

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 môt repository mới tên là group11.



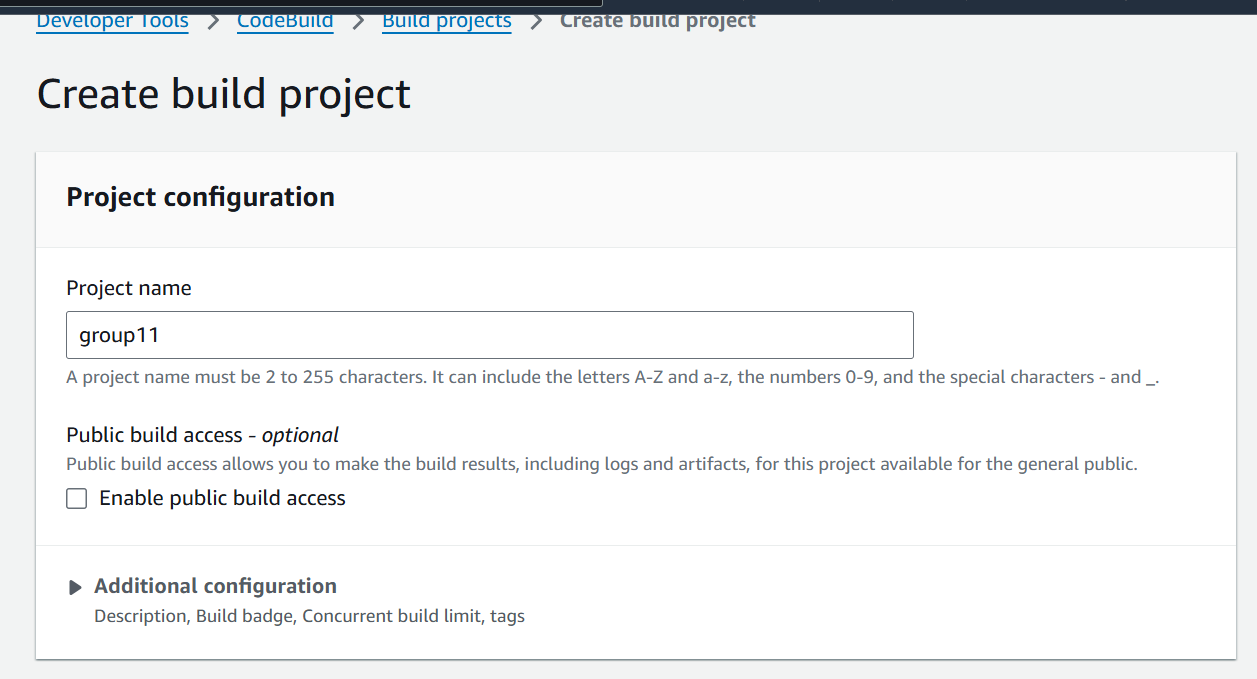
1. Repository group11

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



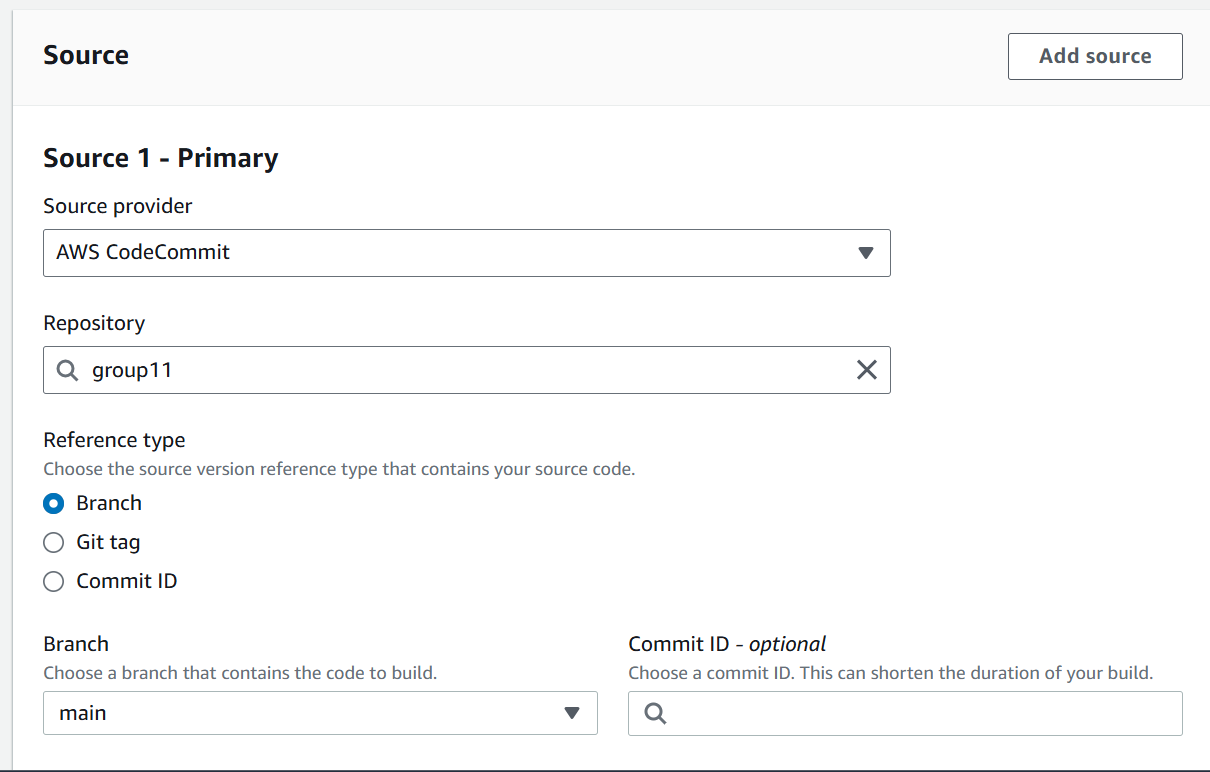
1. Các file trong repo group11

Truy cập vào AWS CodeBuild, chọn Create build project.



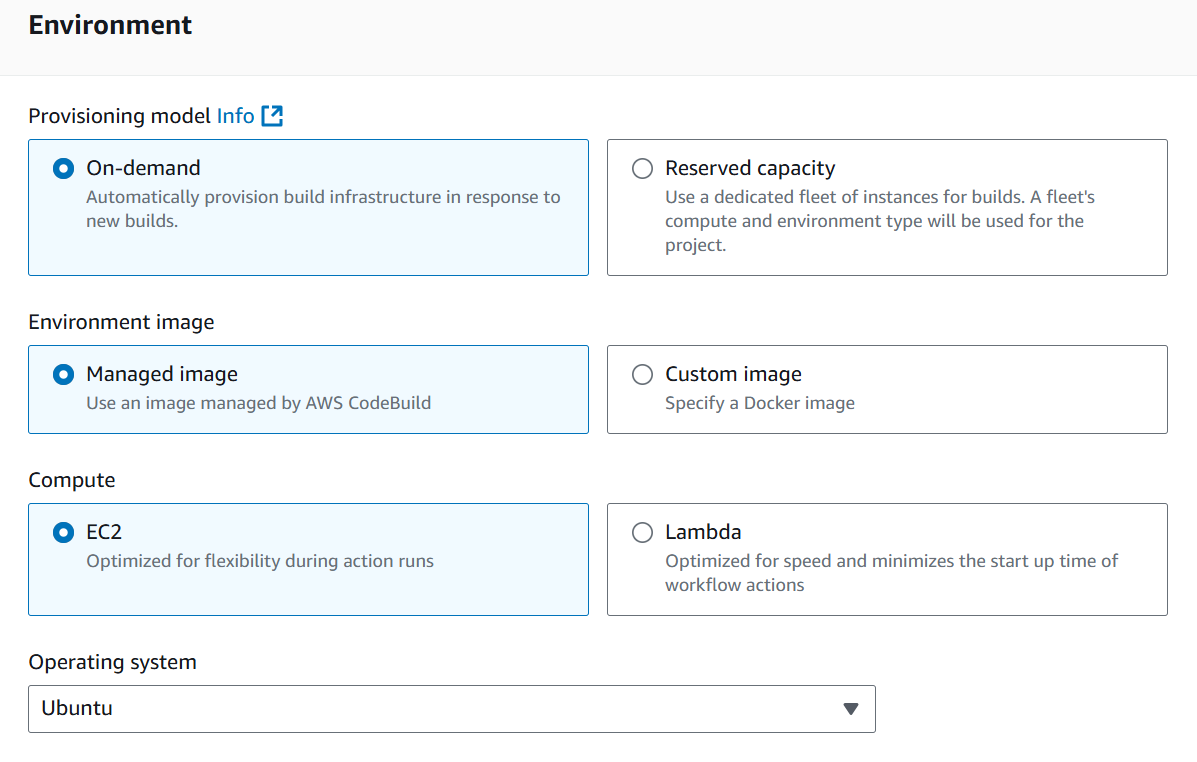
1. Project group11

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



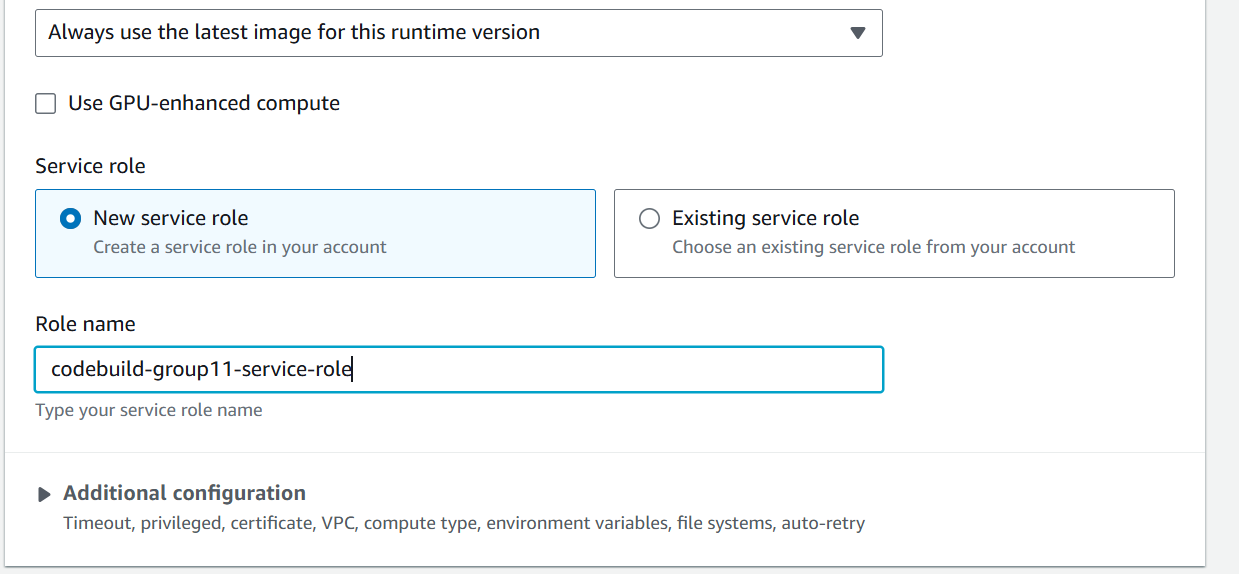
1. Souce CodeBuild

Chọn OS muốn sử dụng (ở đây là Ubuntu)



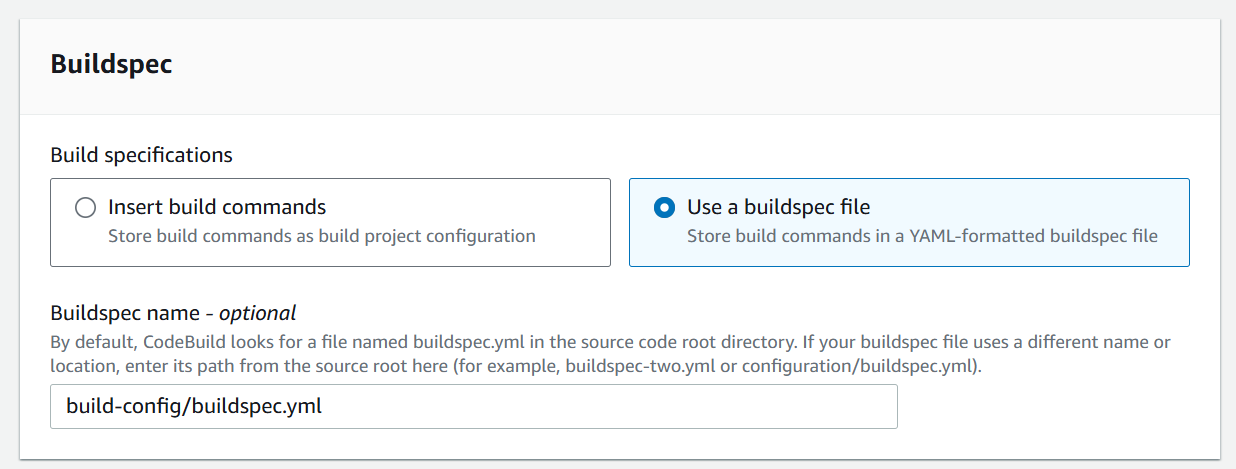
1. Phần Environment

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



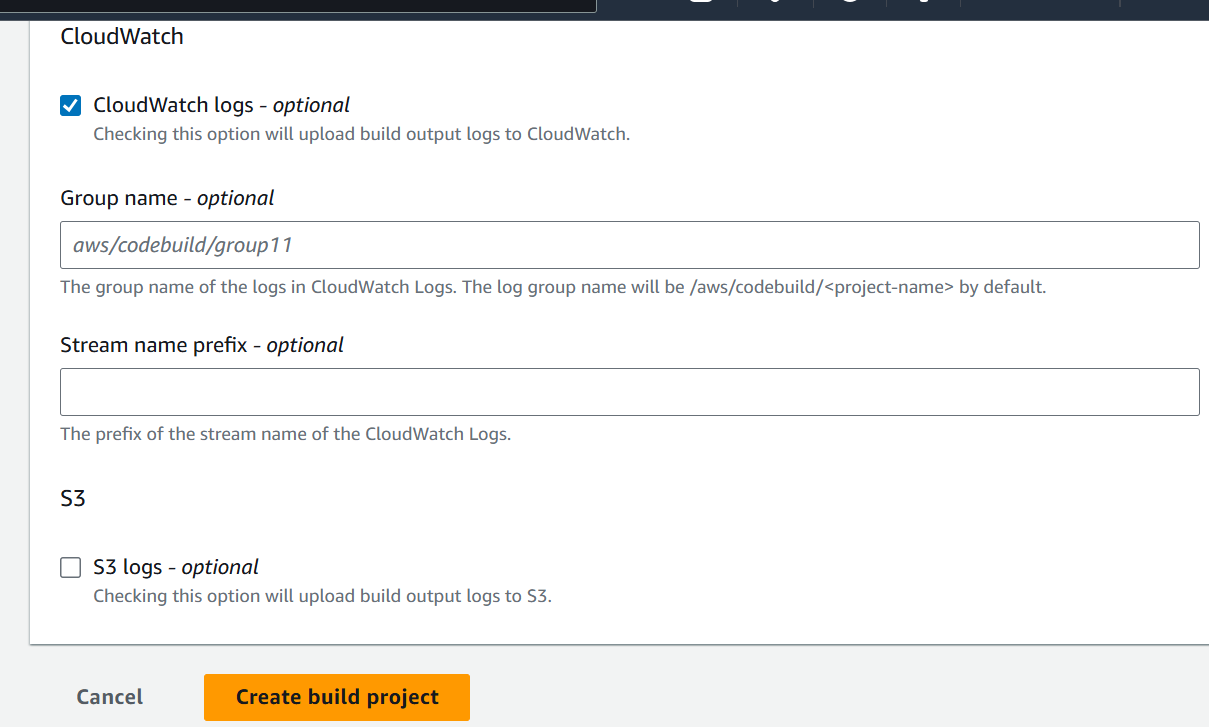
1. Service role

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



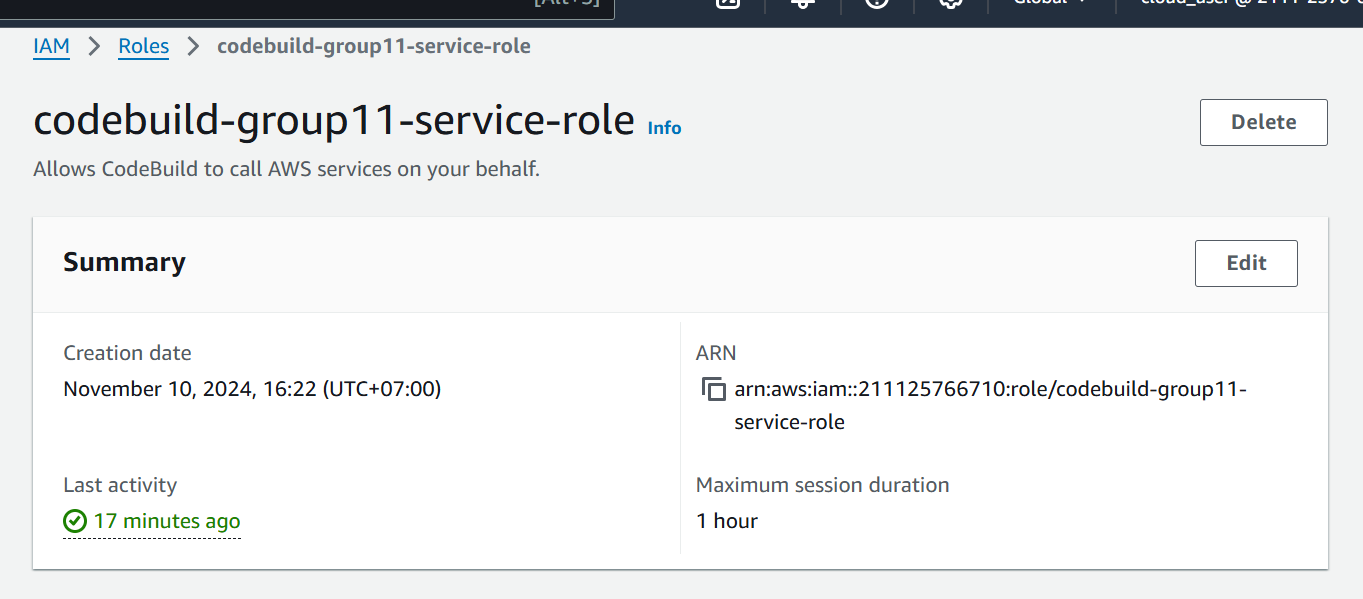
1. Kết nối với file buildspec.yml

Hoàn thành các bước trên và chọn Create build project



1. Chọn Create build project

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

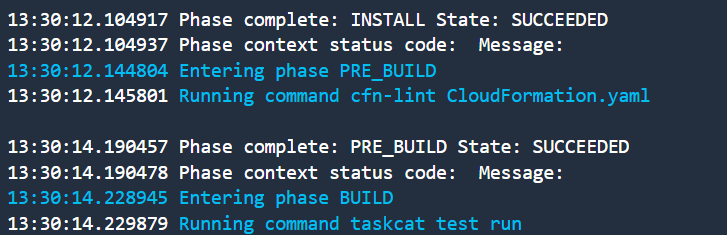


1. IAM role codebuild-group11-service-role



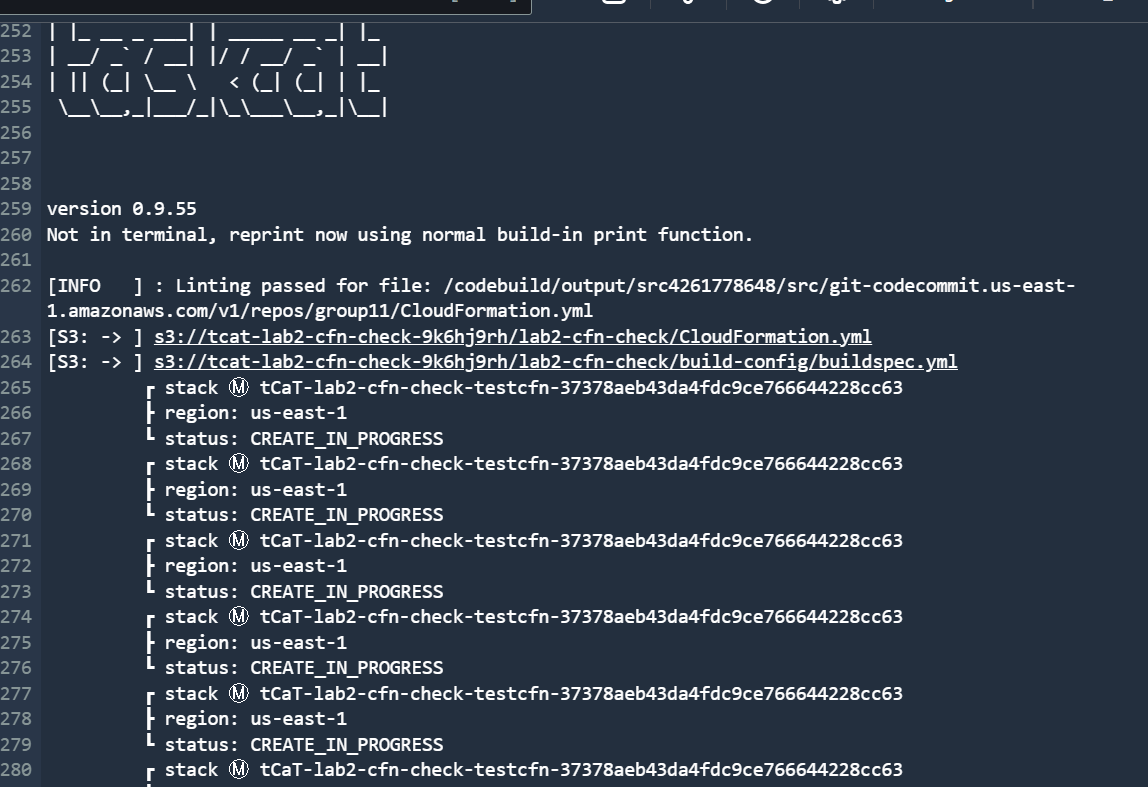
1. Permissions policies

Chạy cfn-lint thành công:



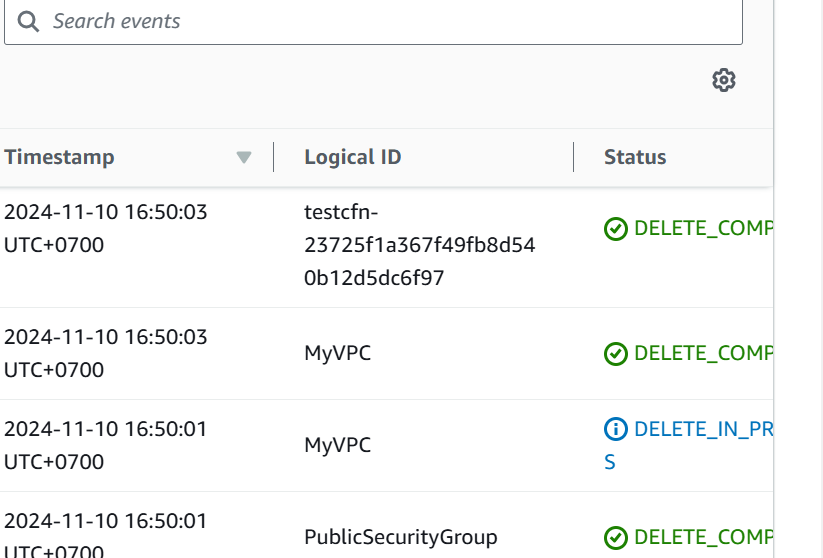
1. Cfn-lint success

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



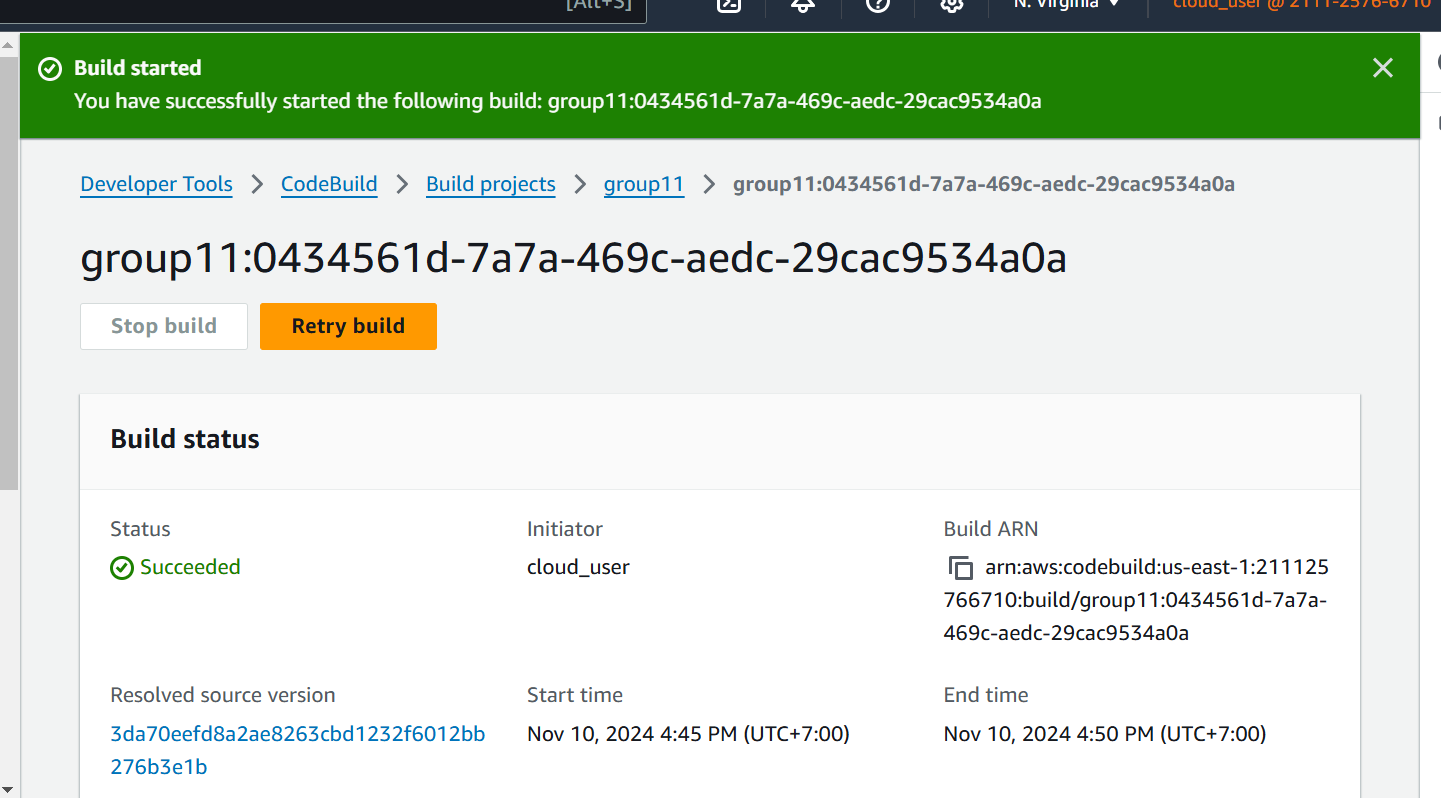
1. Taskcat chạy thành công

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



1. Template CloudFormation

Project đã build thành công.



1. Build thành công

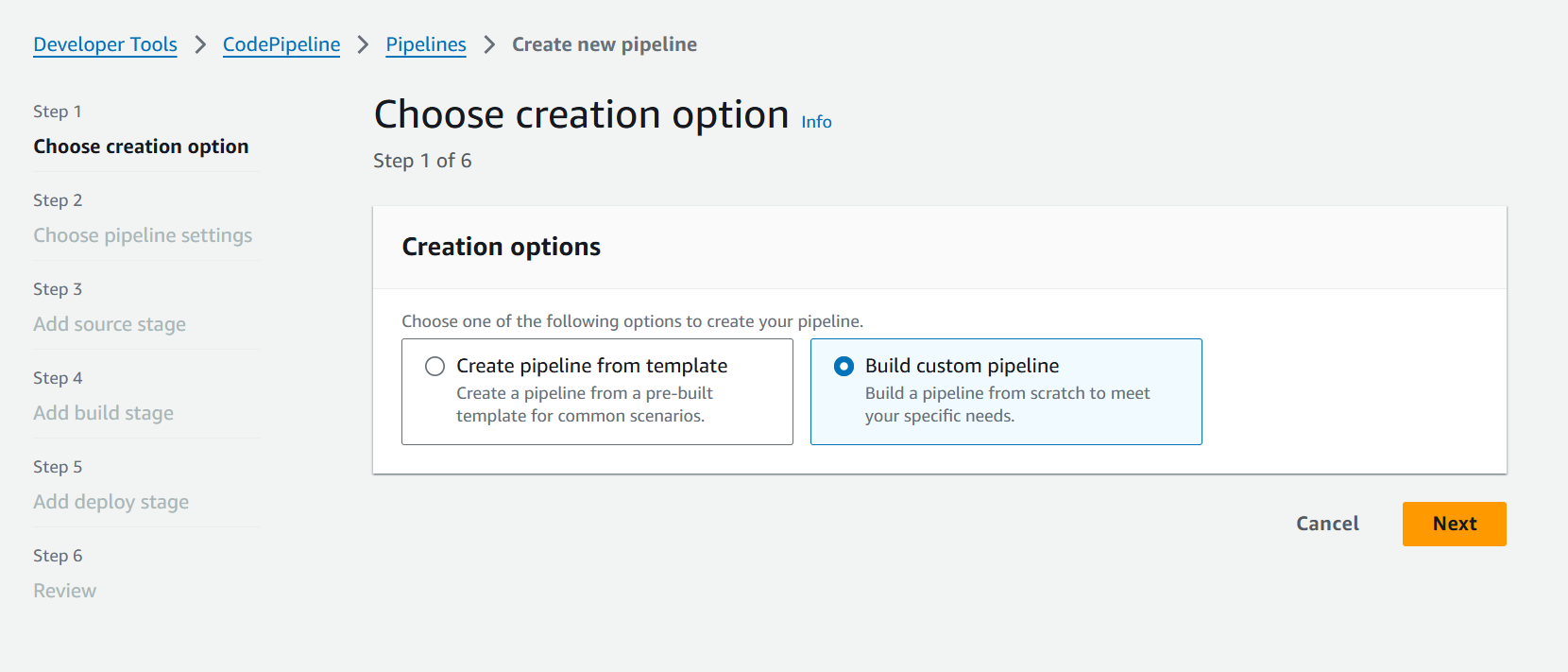
### Sử dụng AWS CodePipeline để tự động hóa quy trình build và deploy từ mã nguồn trên CodeCommit.

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



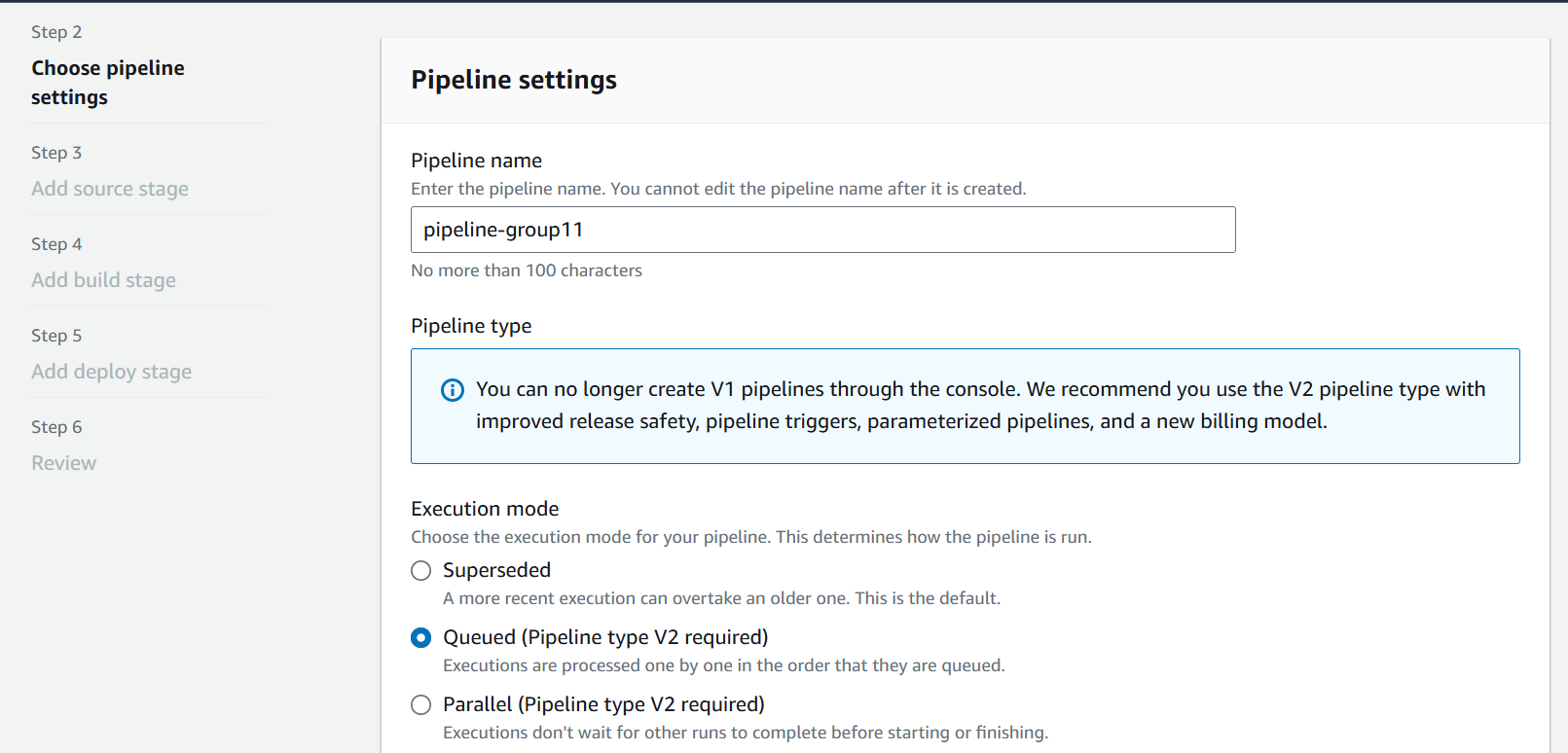
1. Repo đã được tạo trên yêu cầu 2.

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



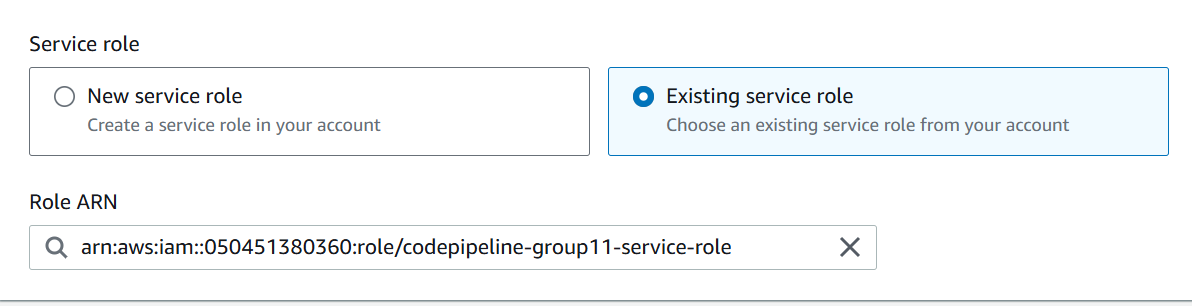
1. Chọn creation option

Đặt tên cho Pipeline là pipeline-group11



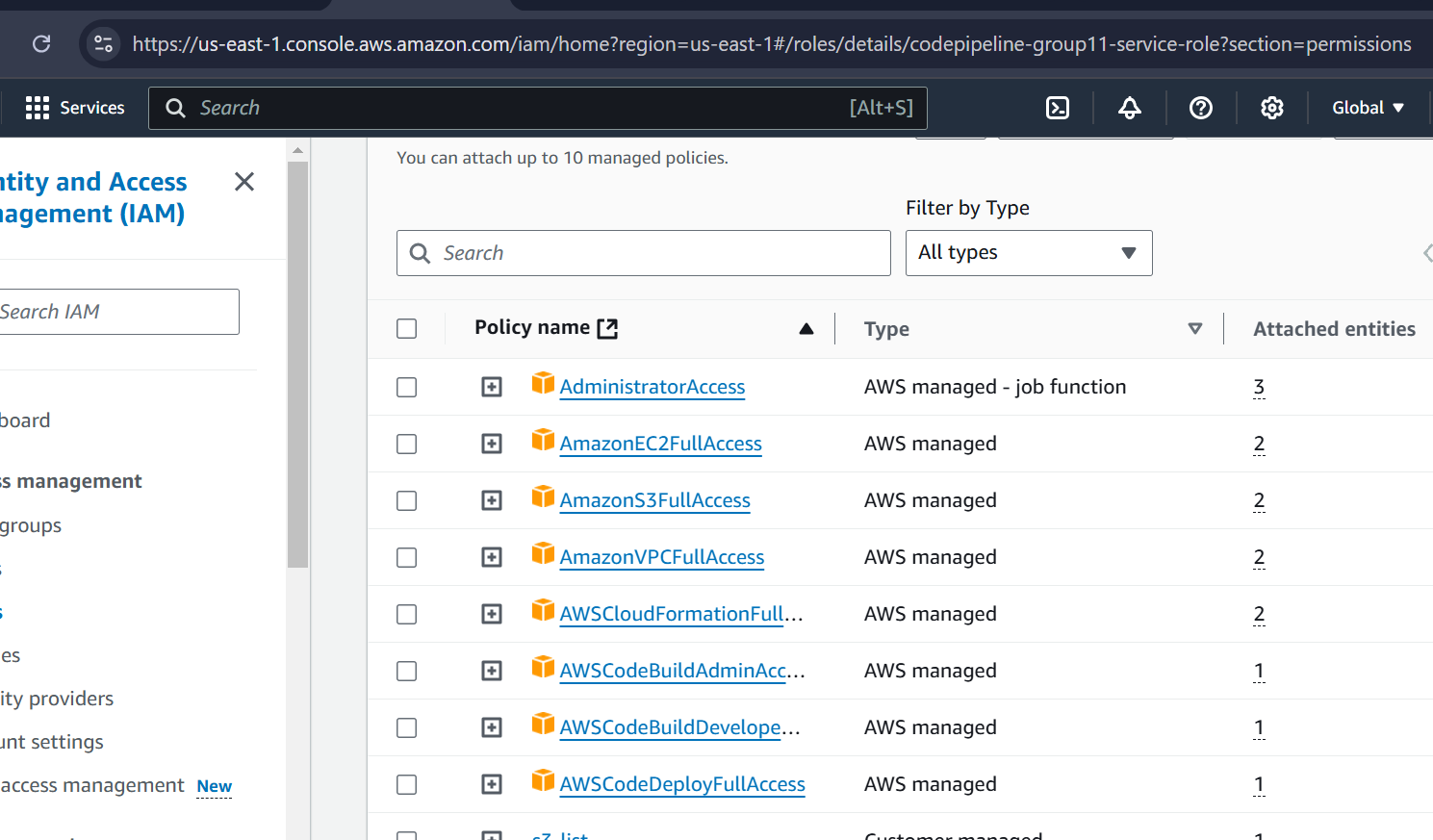
1. Pipeline-group11

Chọn Existing service role và nhập Role ARN



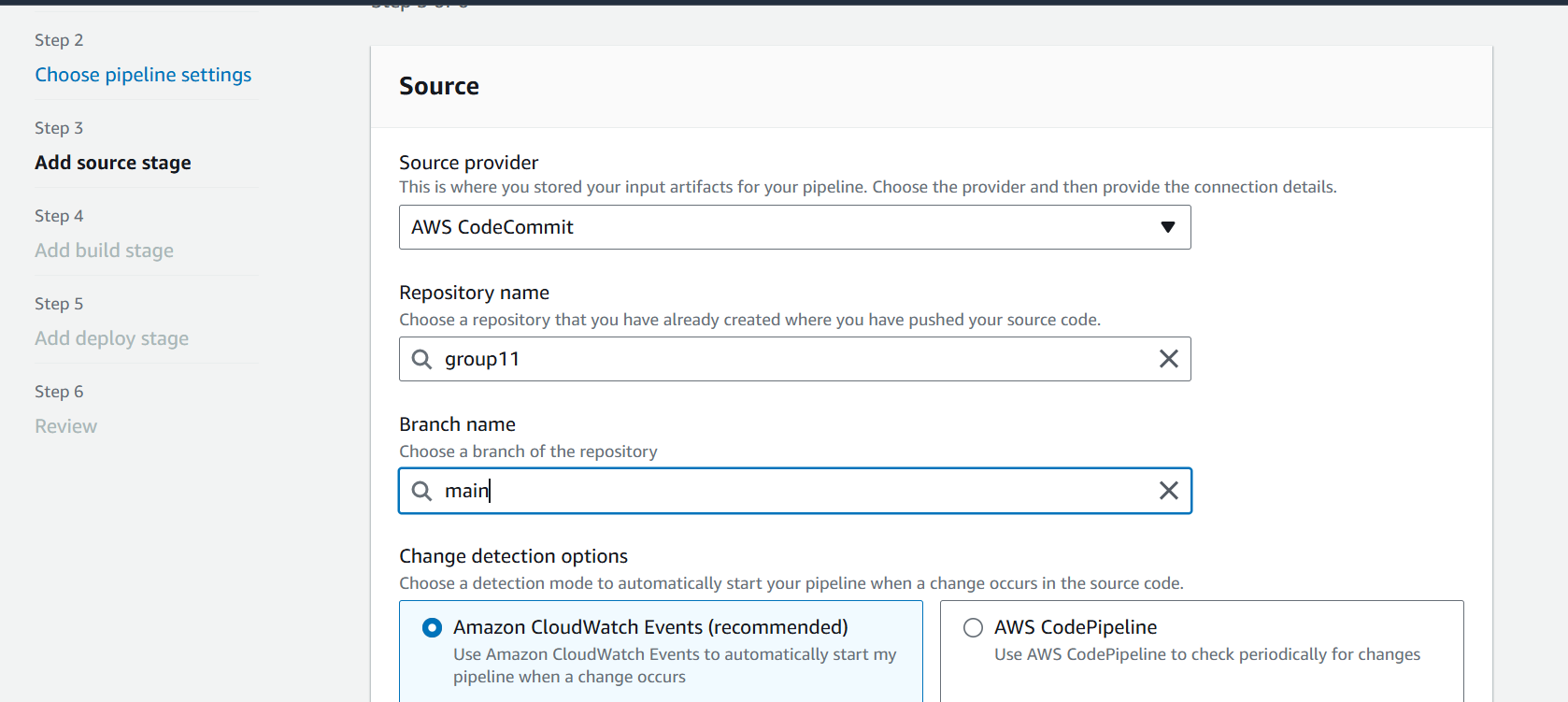
1. Role ARN

Trong role phía trên cấp các Policy phù hợp cho Pipeline

**

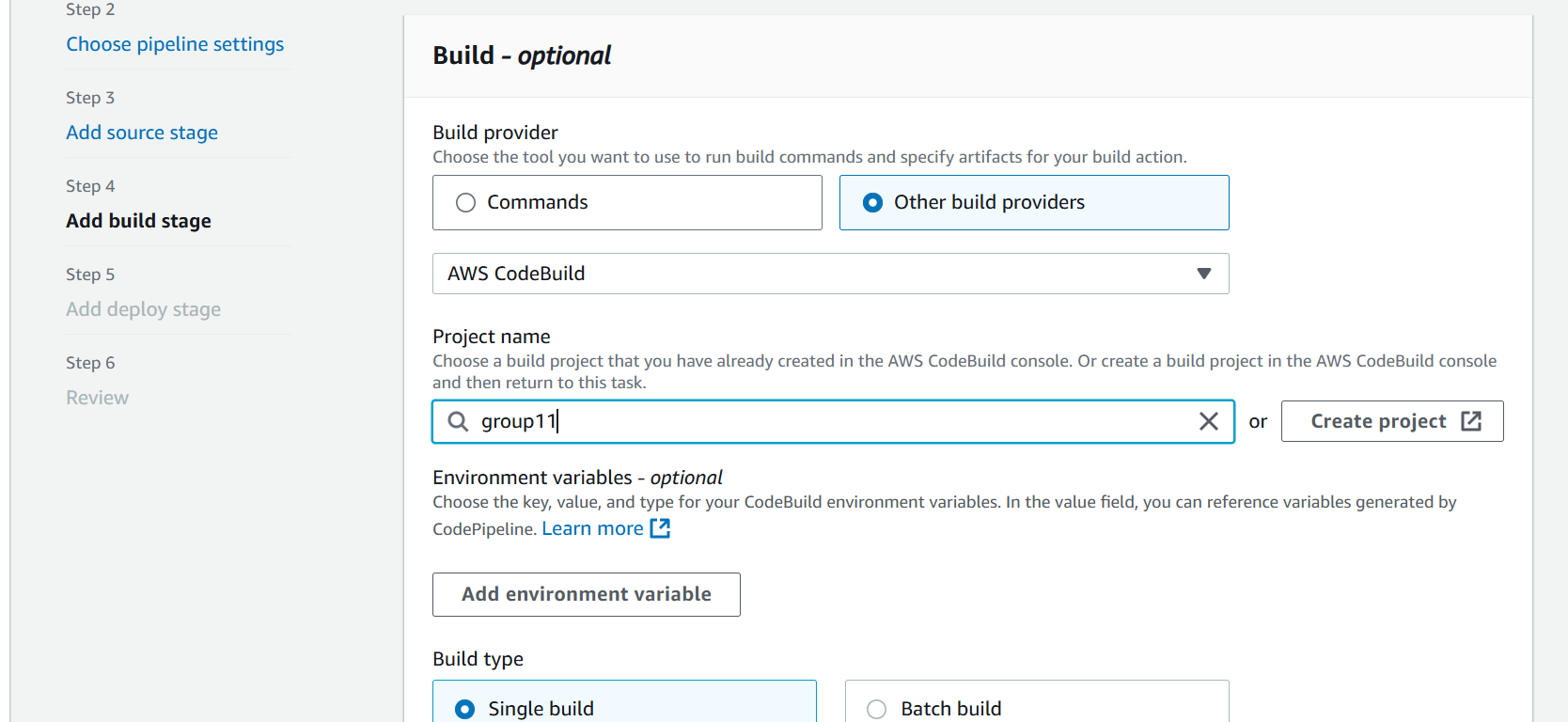
1. Codepipeline-group11-service-role

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



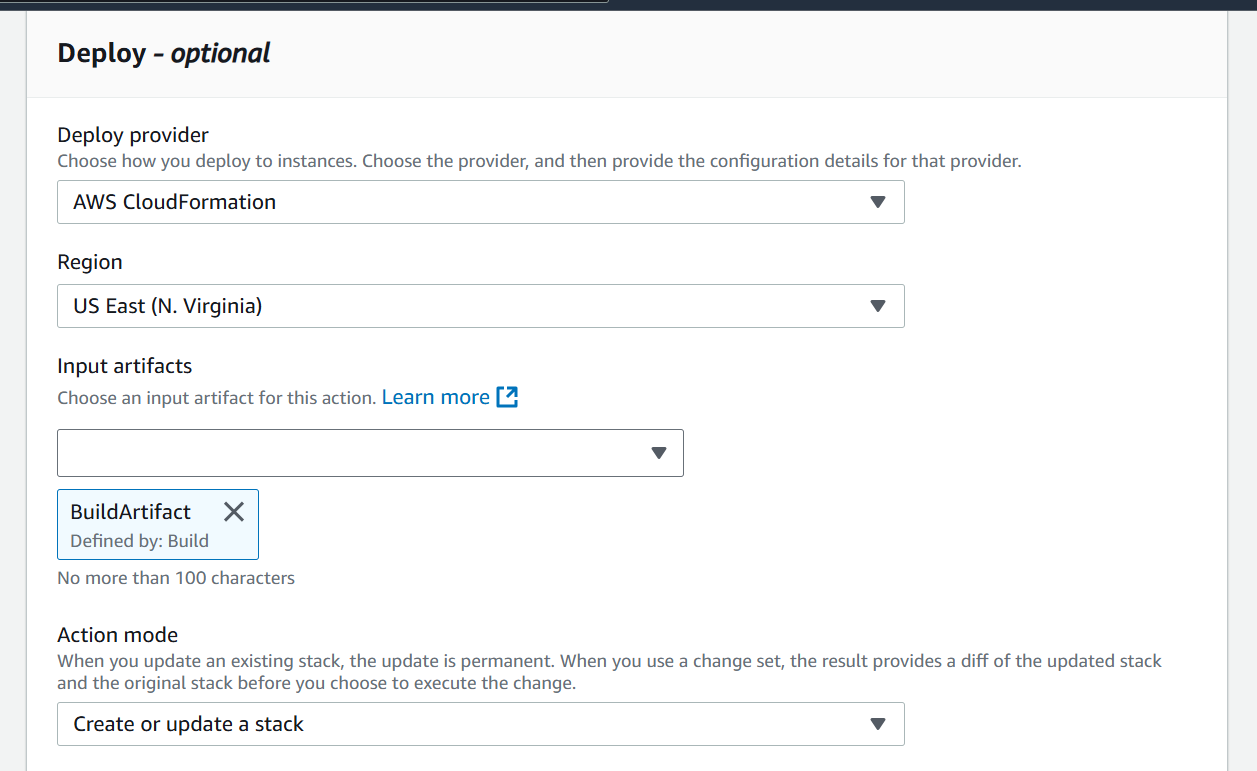
1. Source stage

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

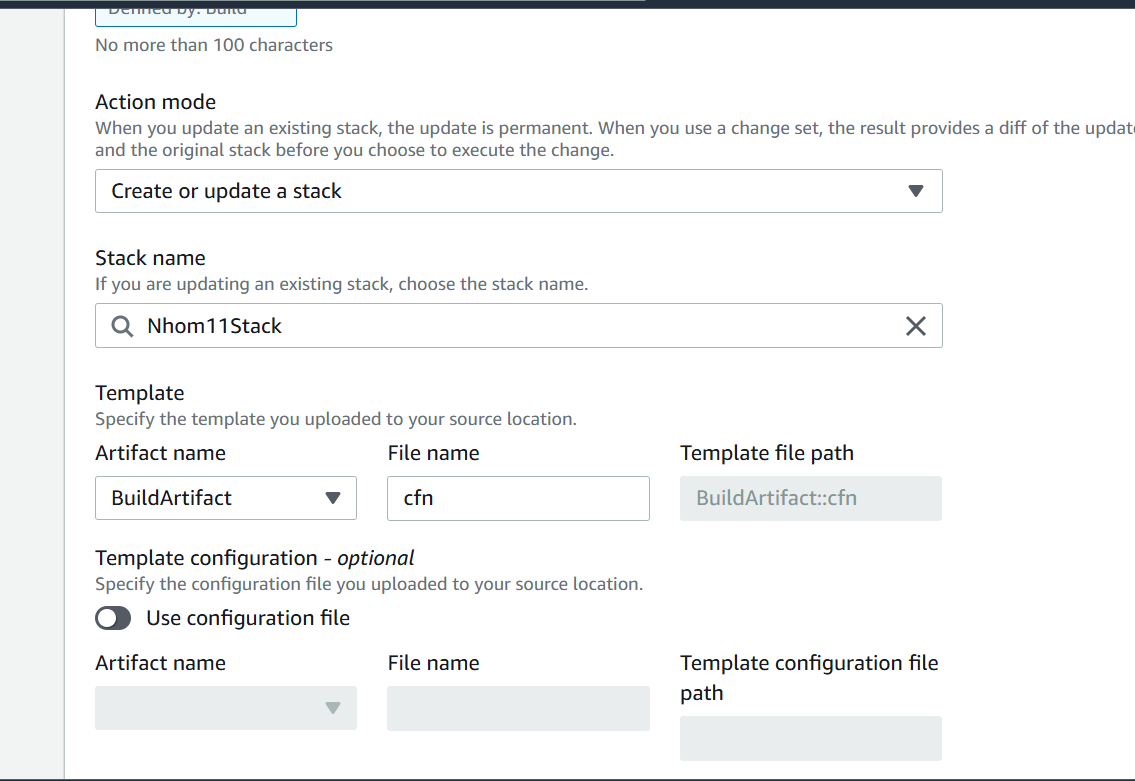


1. Build Stage

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).

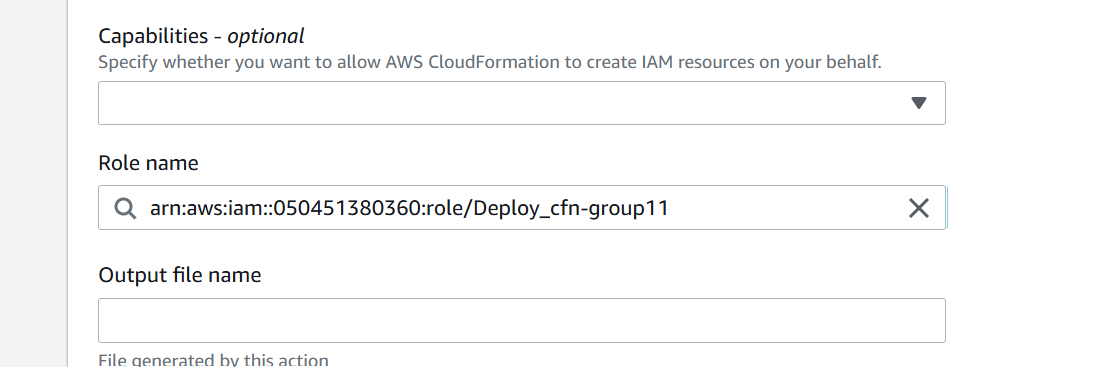


1. Deploy stage (1)



1. Deploy stage (2)

Chọn role đã thiết lập cho Cloudformation



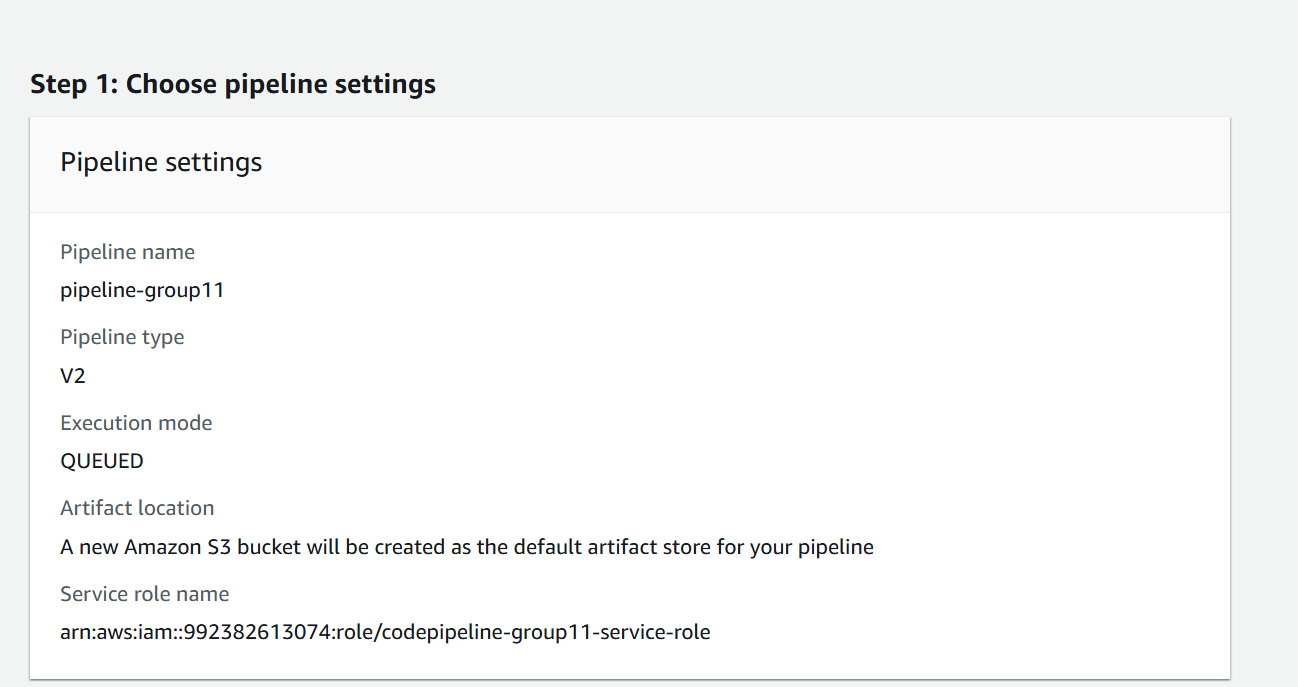
1. Role phần deploy

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

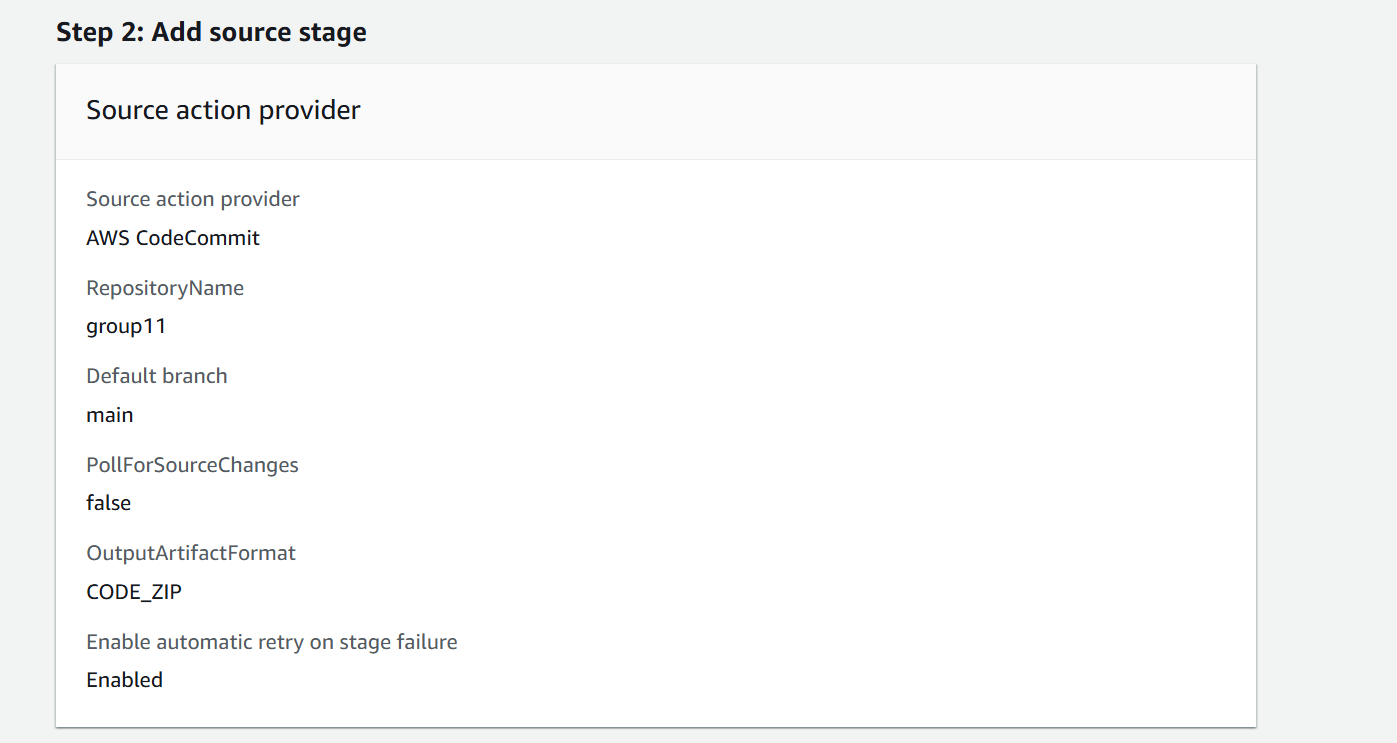


1. Policies của role Deploy\_cfn\_group11

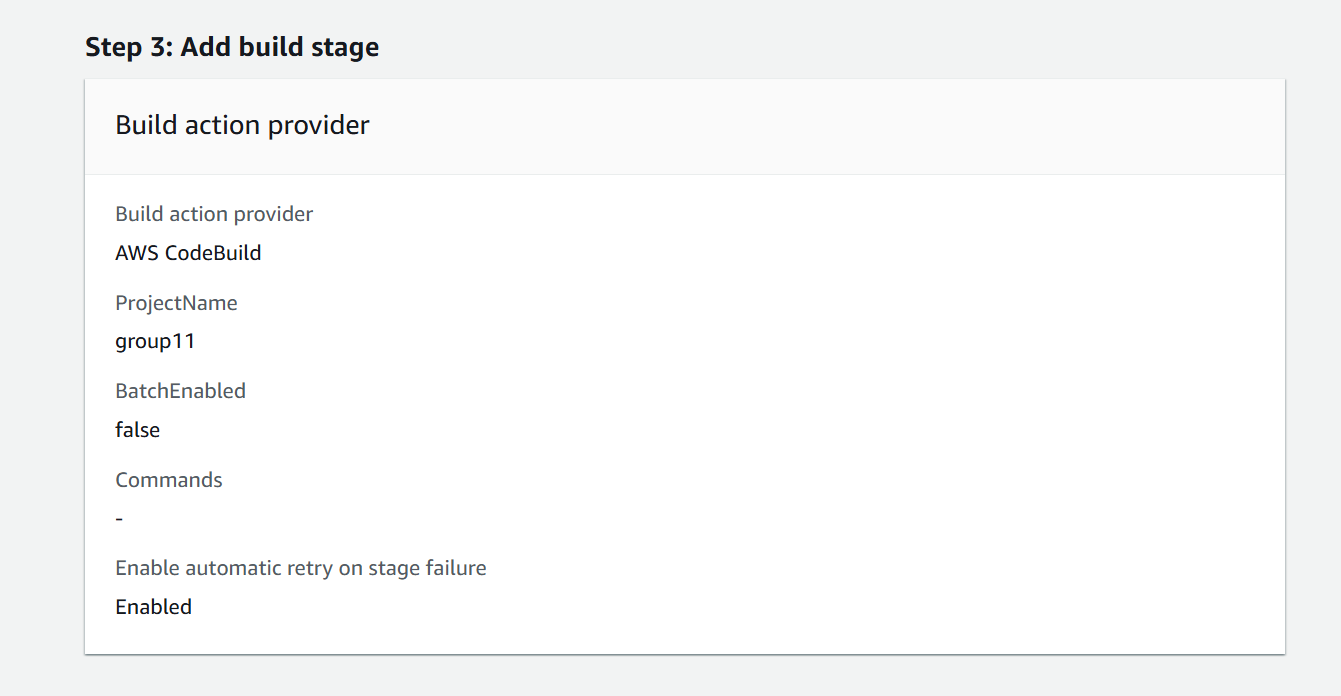
Sau đó review lại các bước trước khi tạo Pipeline lần lượt ở hình 33, hình 34, hình 35 và hình 36



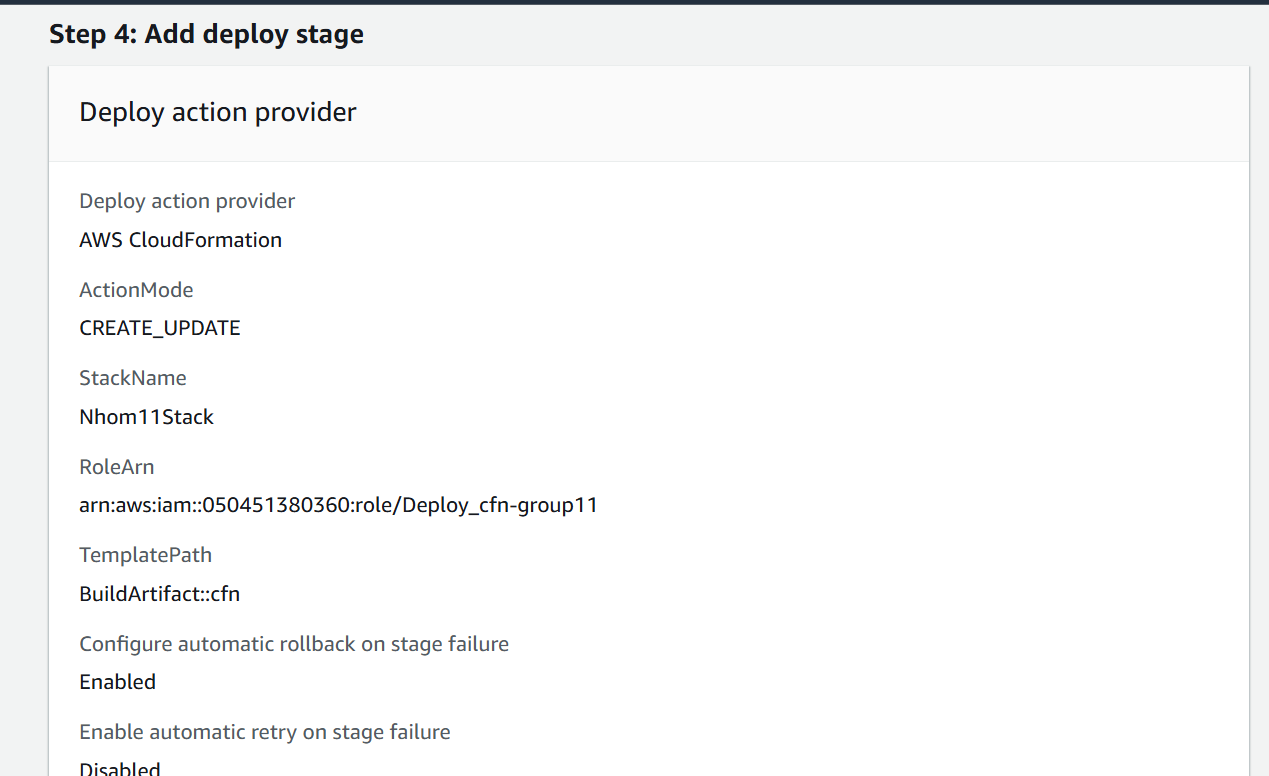
1. Bước 1



1. Bước 2

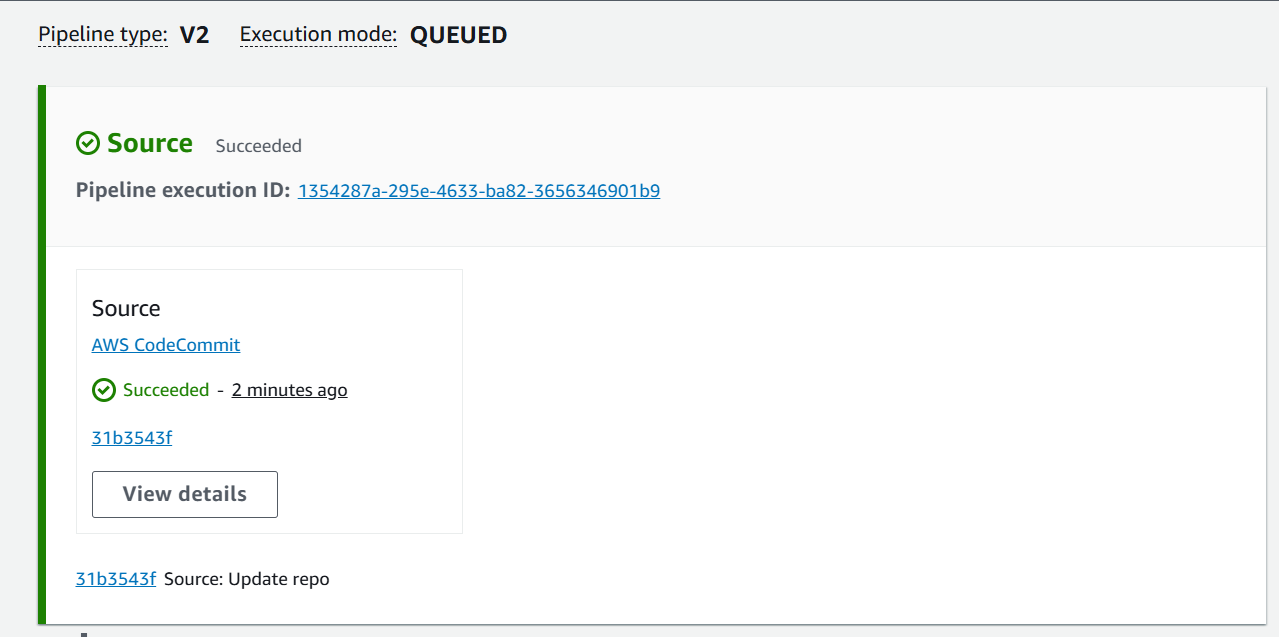


1. Bước 3

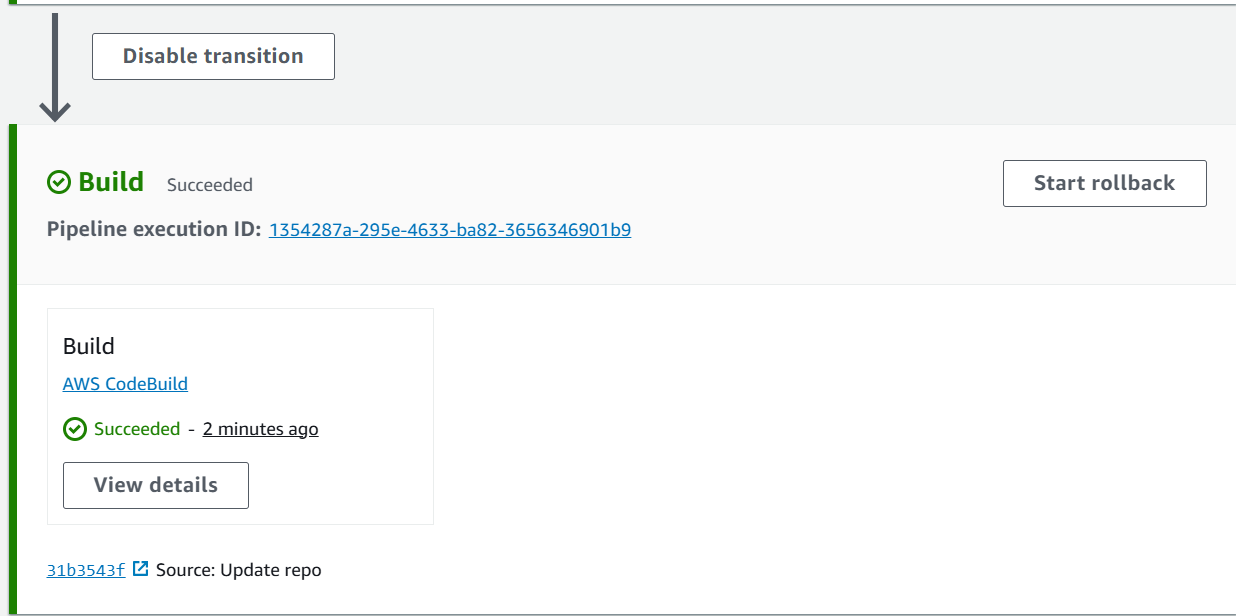


1. Bước 4

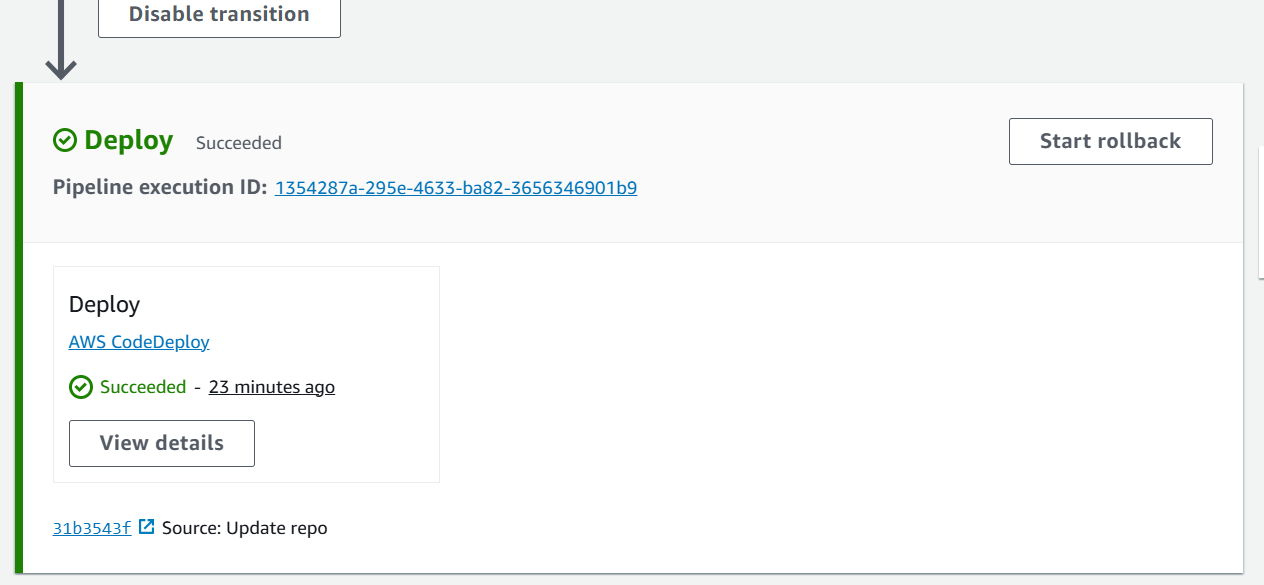
Ở hình 37, 38 và 39, có thể thấy được phần source, build và deploy lần lượt đều đã được triển khai thành công trên AWS CodePipeline.



1. Source Pipeline

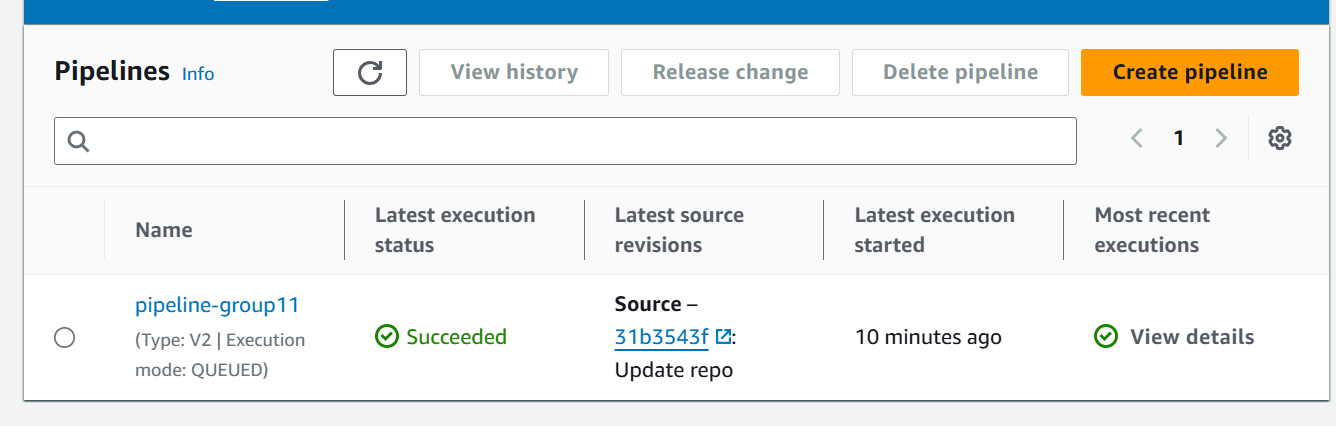


1. Build pipeline

****

1. Deploy pipeline

Pipeline đã được triển khai thành công (hình 40)



1. Pipeline-group11 được triển khai thành công

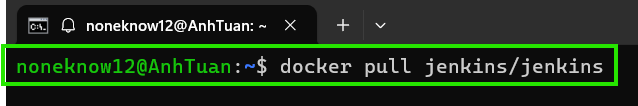
## Sử dụng Jenkins để quản lý quy trình CI/CD cho ứng dụng microservices

### 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:

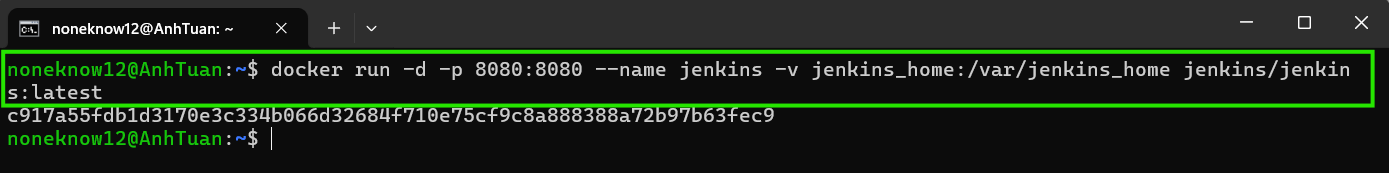
Jenkins: cài đặt sử dụng Docker Container.

Đầu tiên pull images của jenkins từ Docker Hub:



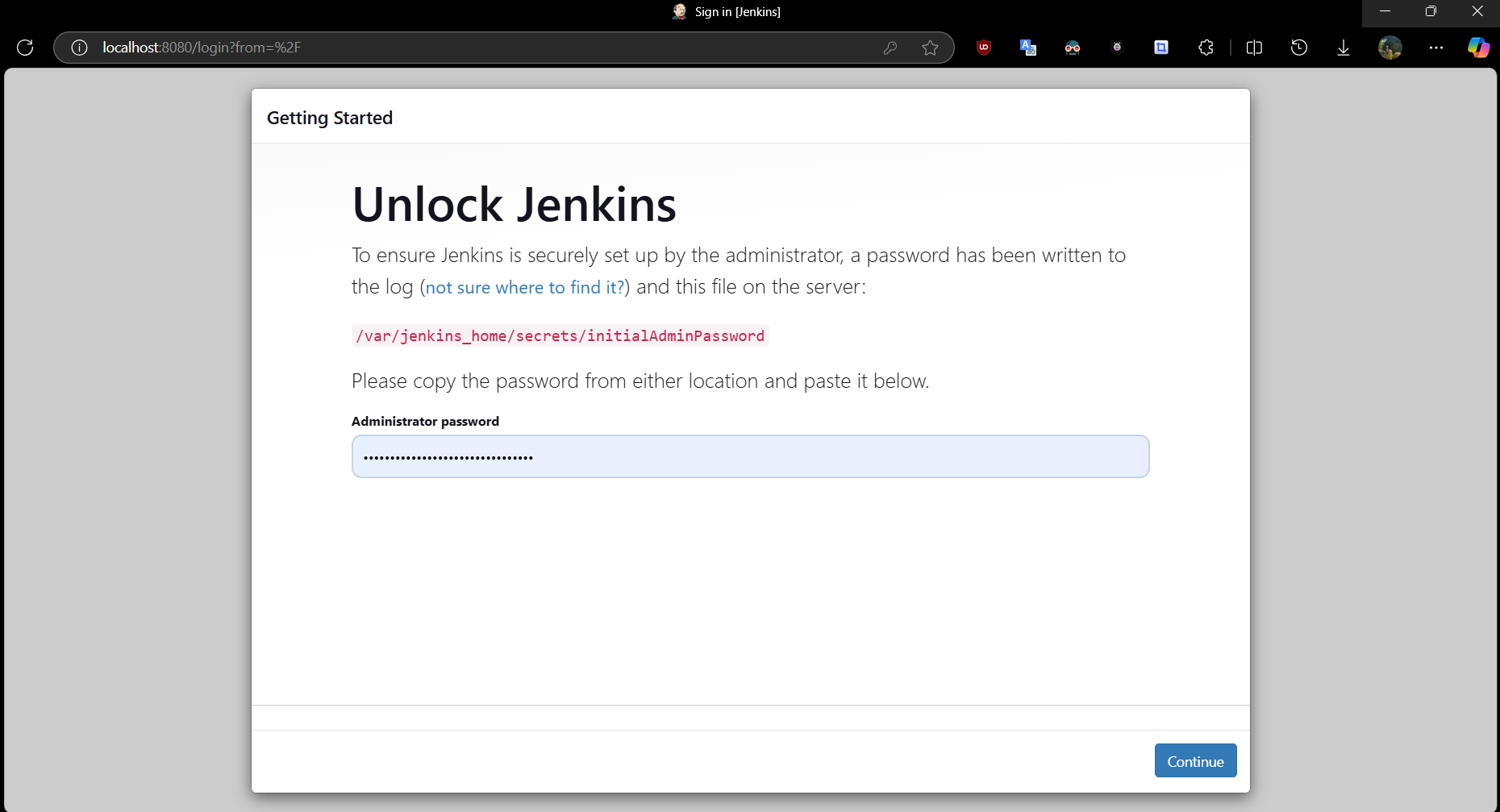
1. Pull Jenkins từ Docker Hub

Chạy container từ images mới được pull về:



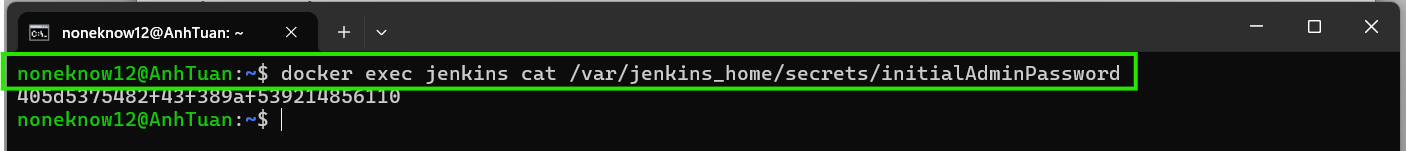
1. Chạy container

Truy cập vào localhost:8080 để kiểm tra:



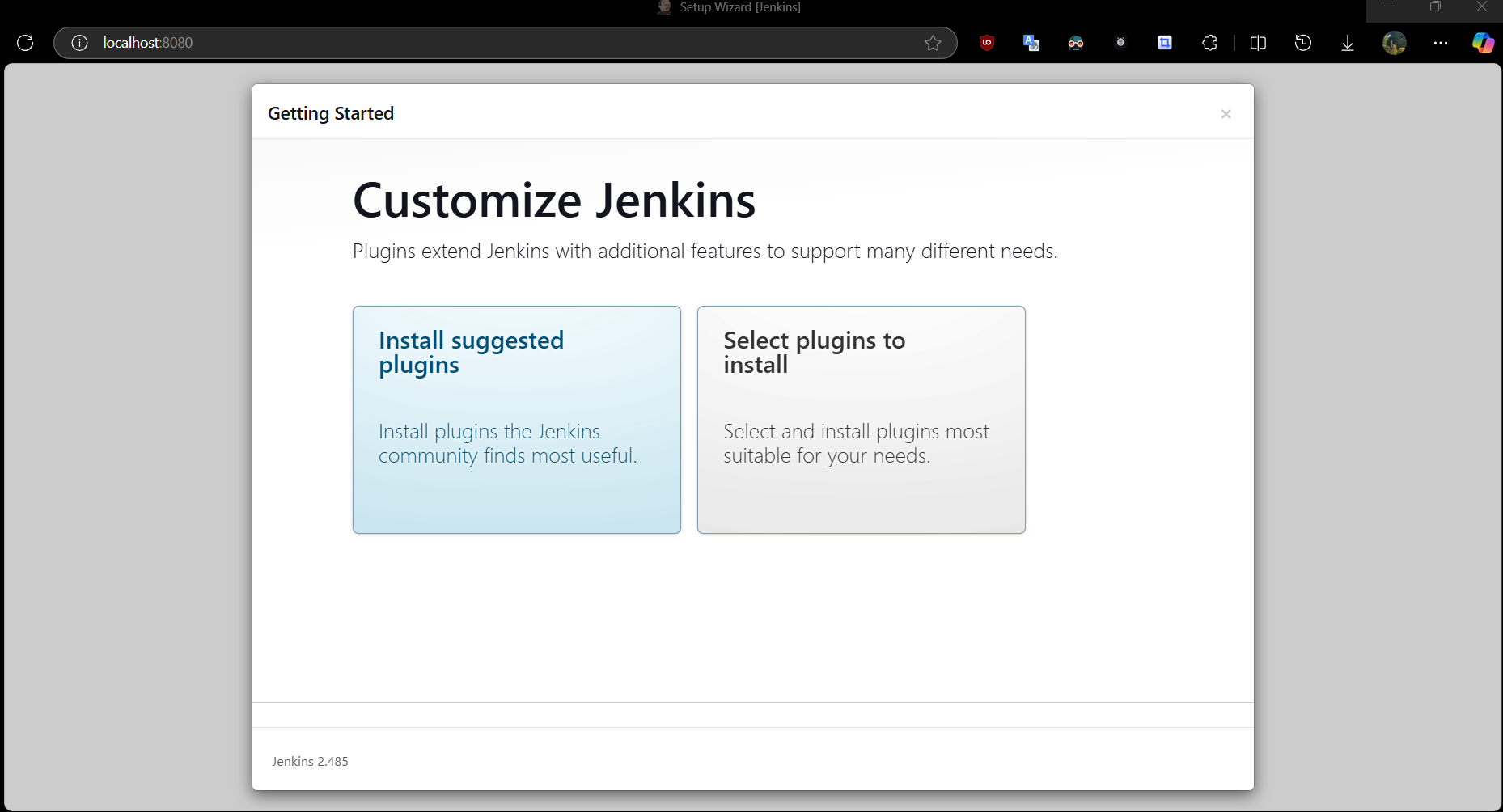
1. Kiểm tra Jenkins đã install thành công hay chưa

Lấy mật khẩu của Jenkins cung cấp ở trong container:



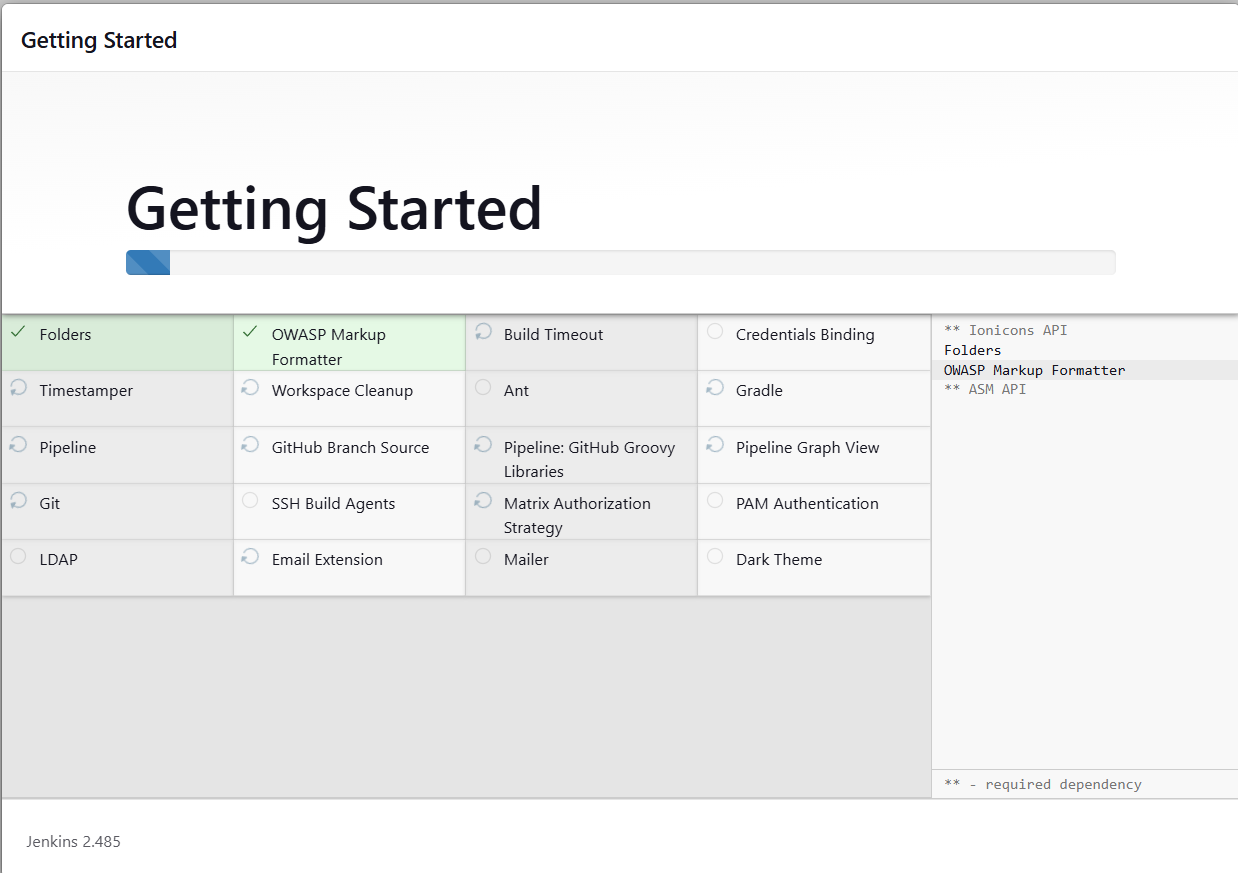
1. Lấy mật khẩu đăng nhập Jenkins

Chọn Install suggested plugins để cài các plugins khởi đầu:



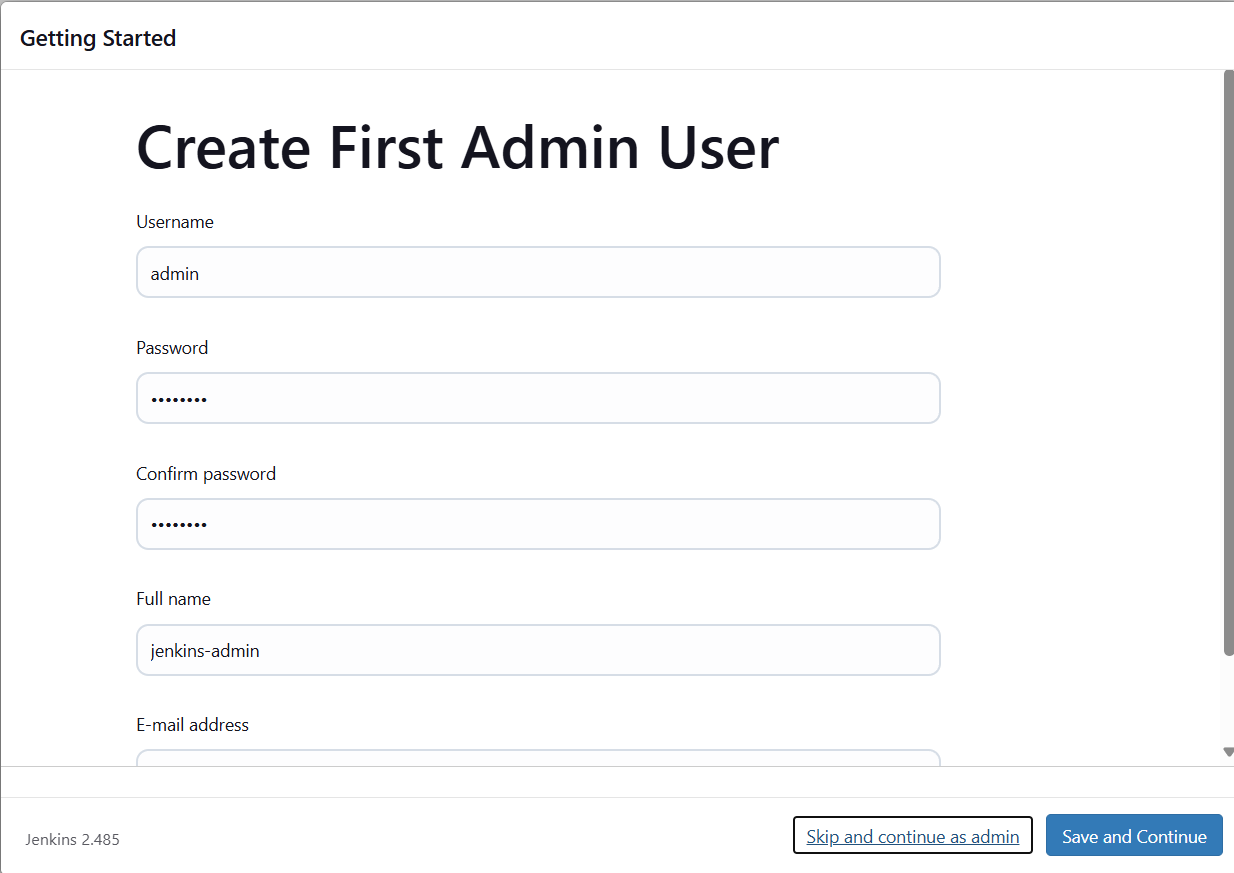
1. Config Jenkins

Cài đặt các packet cần thiết:

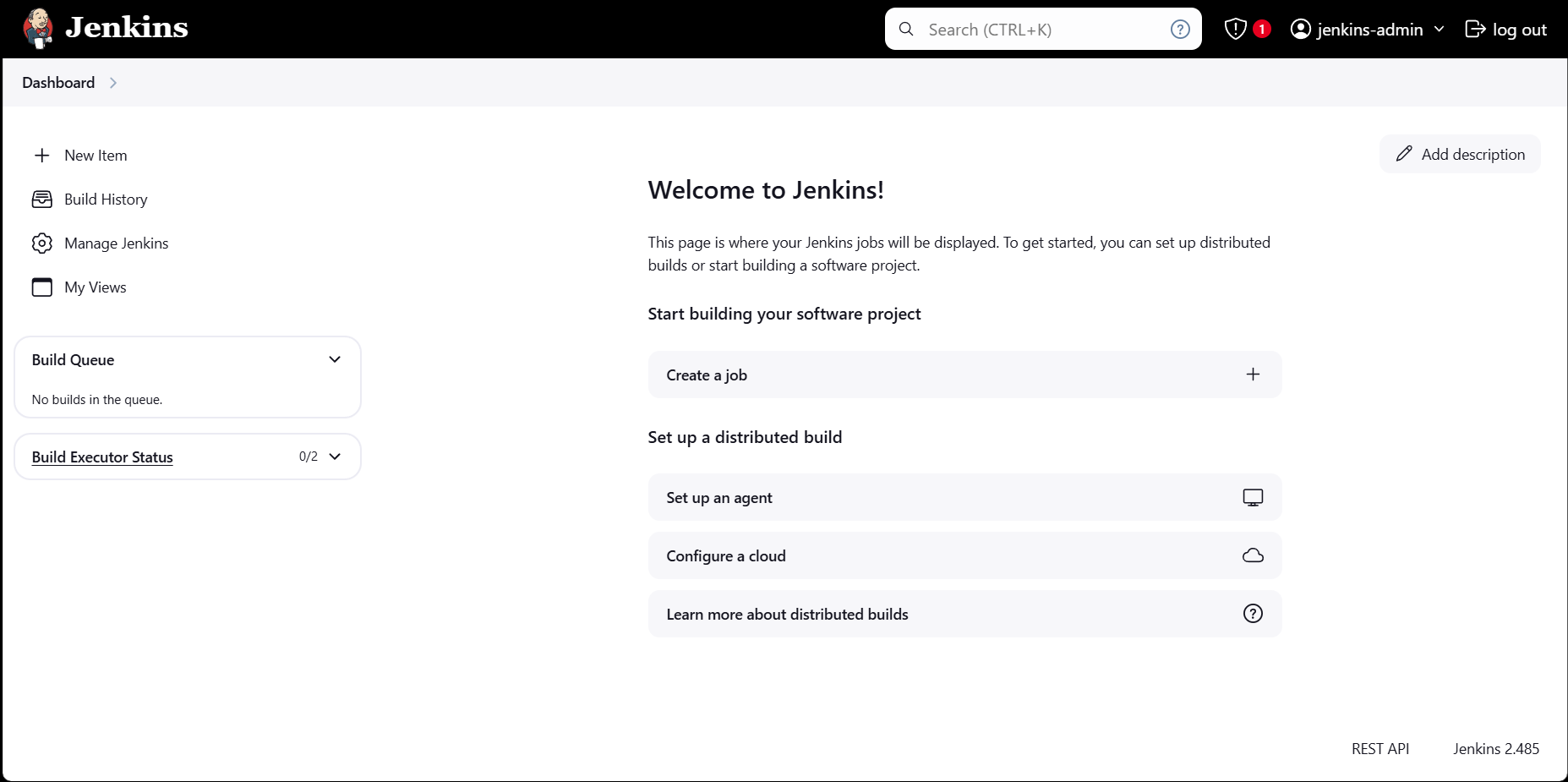


1. Cài đặt packets

Tạo tài khoản admin



1. Tạo tài khoản



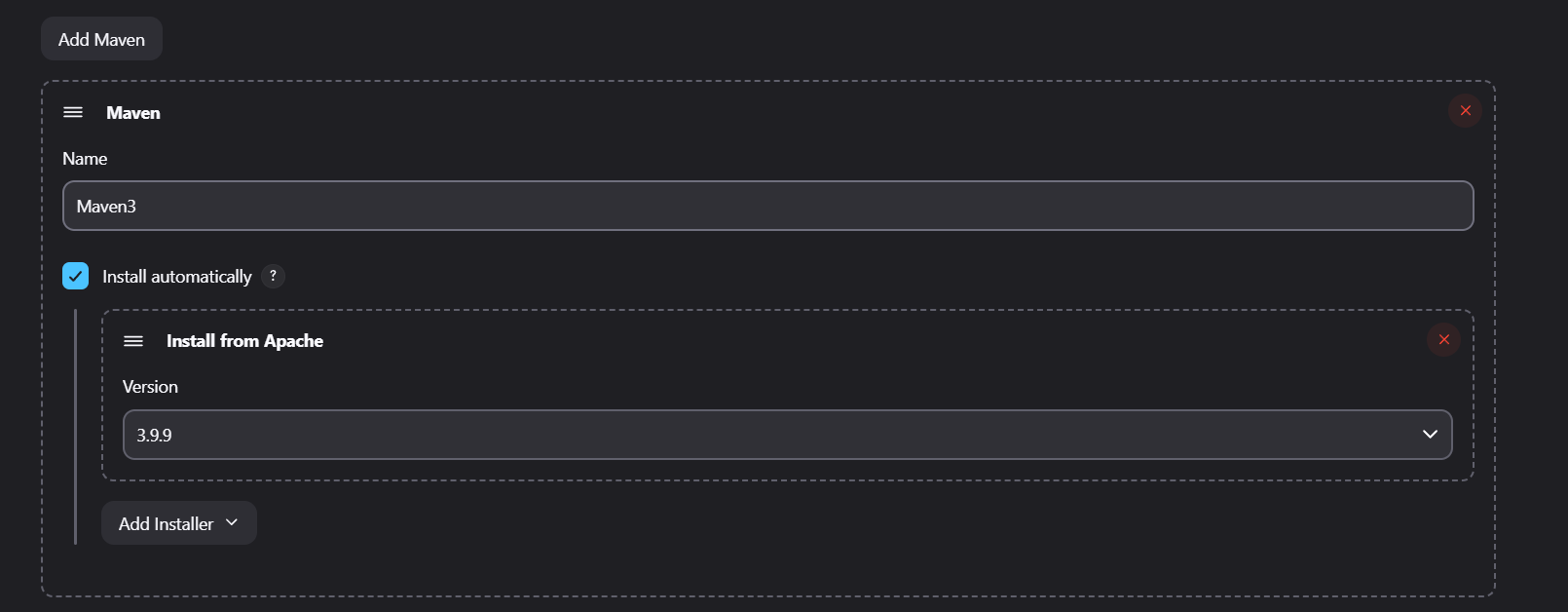
1. Trang chủ Jenkins

Tổng quan về microservice application:

Java Spring, JDK11, Spring Boot 2

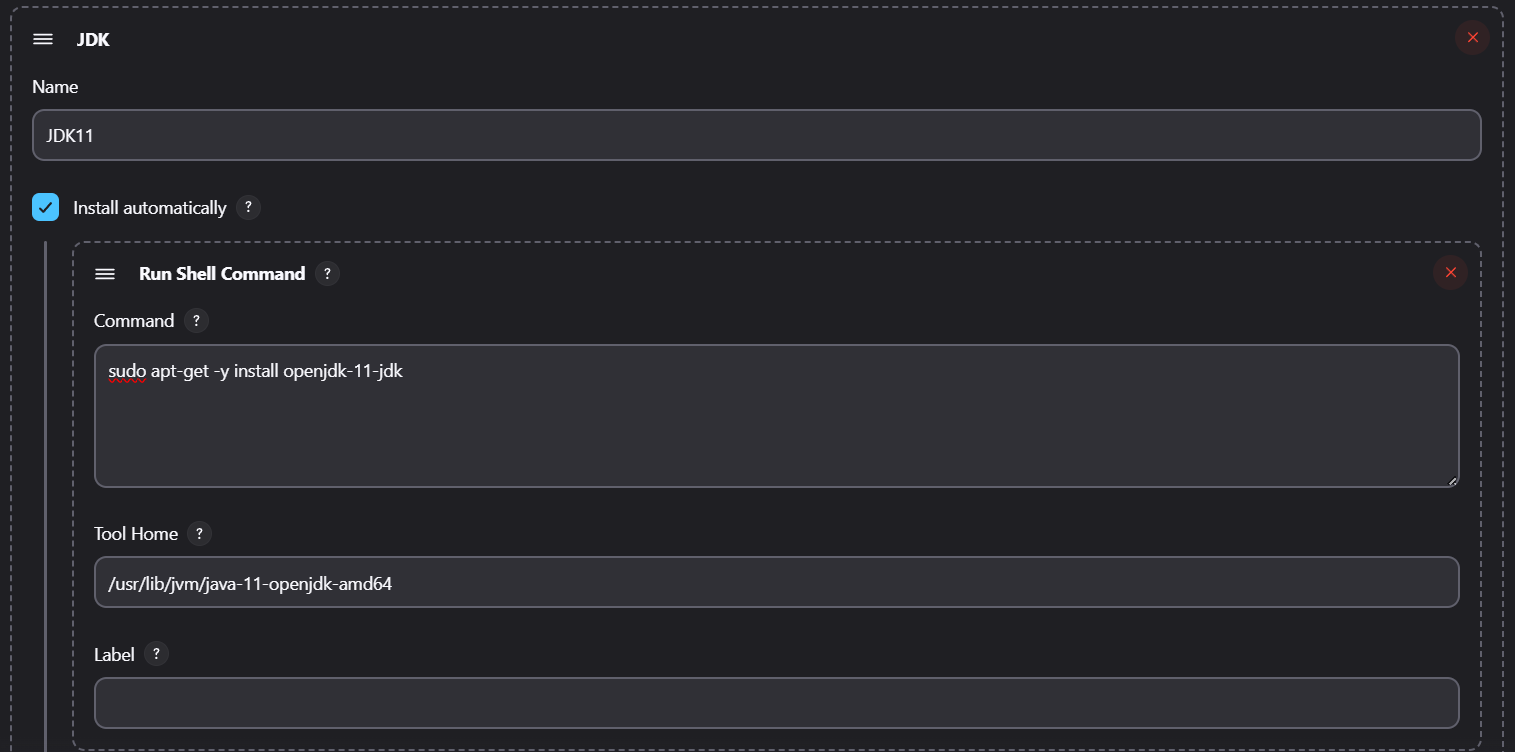
Cấu hình Tool cho jenkins:

Maven:



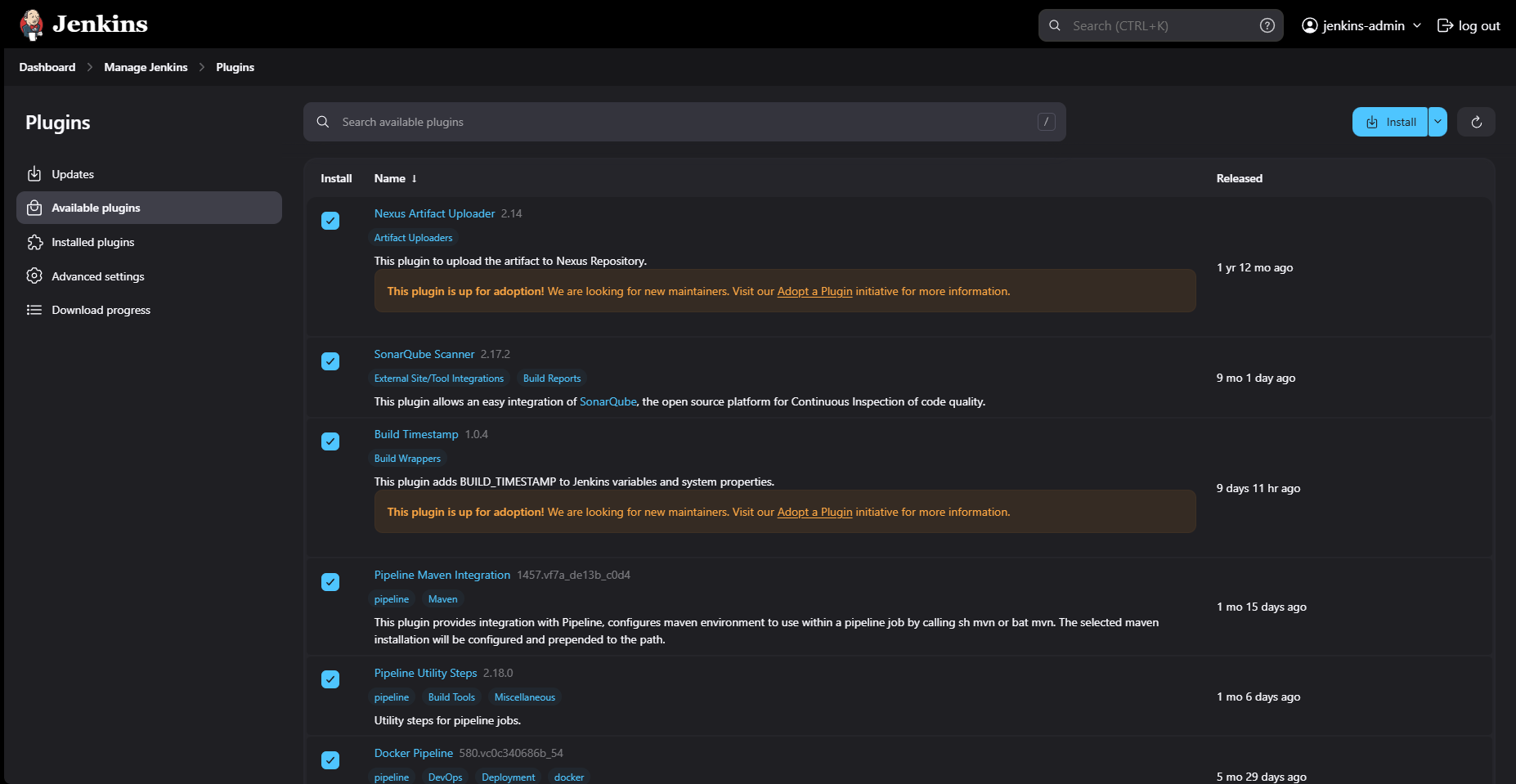
1. Cấu hình Maven

JDK11:



1. Cấu hình JDK11

Cài đặt các plugin cần thiết cho Jenkins:



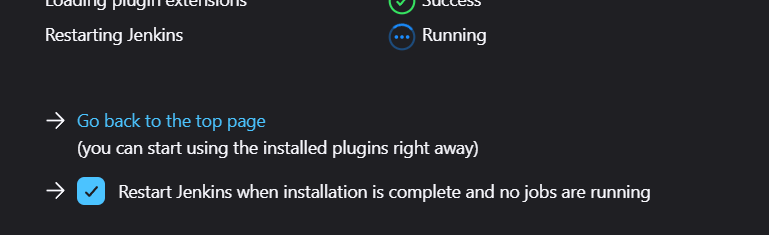
1. Cài đặt plugins

Các Plugins cài đặt thành công



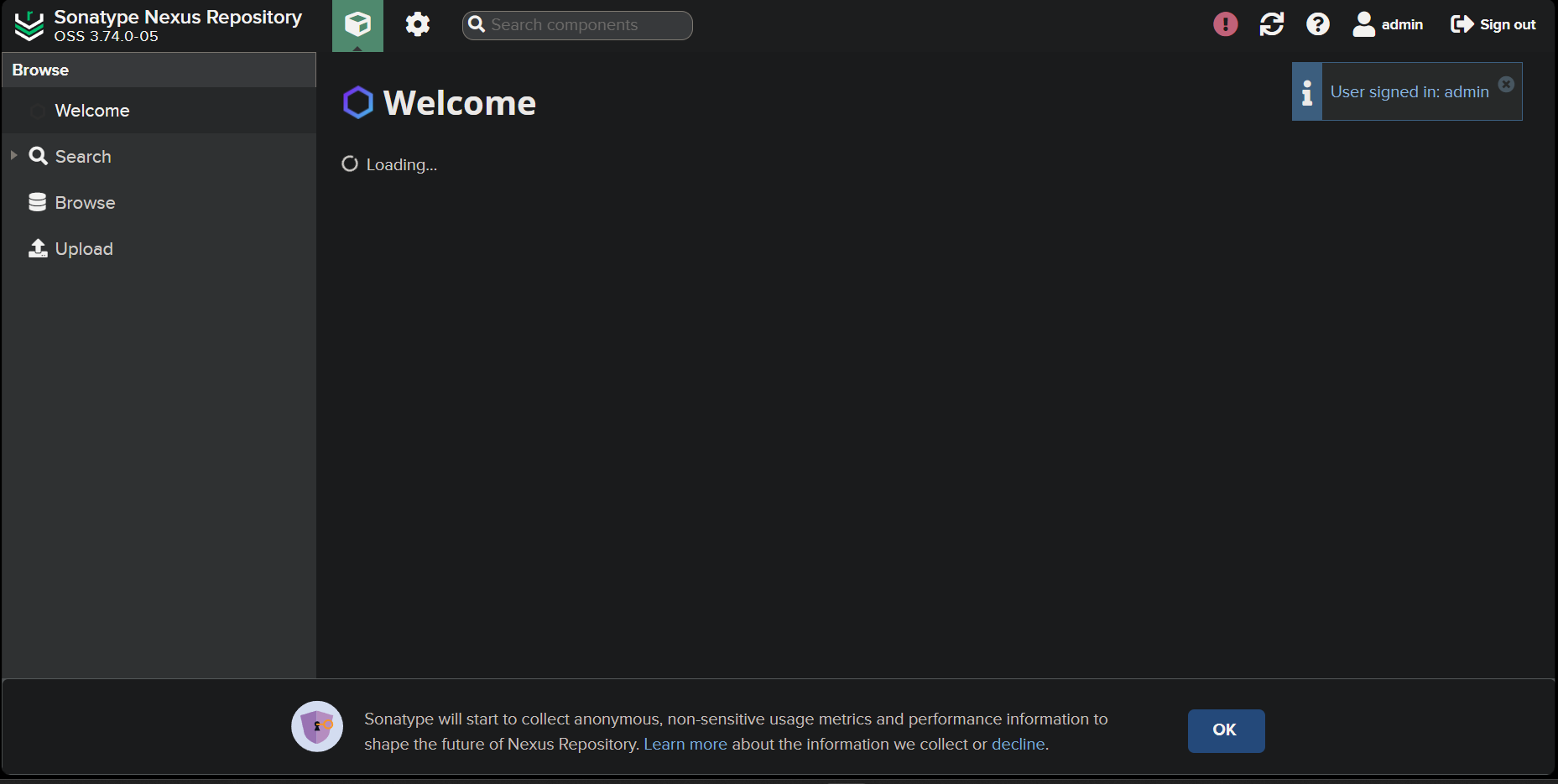
1. Plugins cài đặt thành công

Sau khi hoàn tất cài đặt plugins, restart Jenkins server



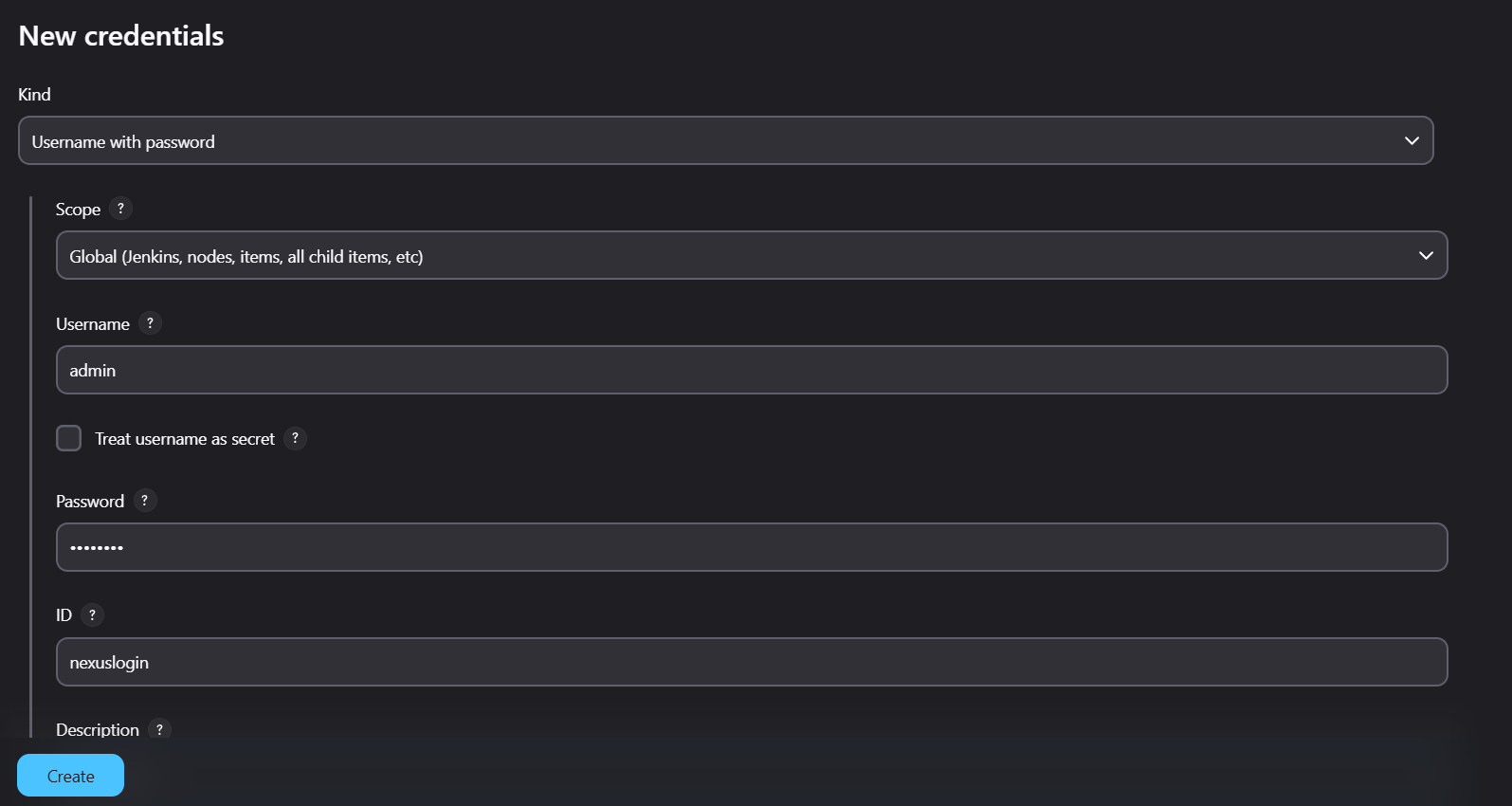
1. Jenkins restart

\* Cấu hình Nexus repo:



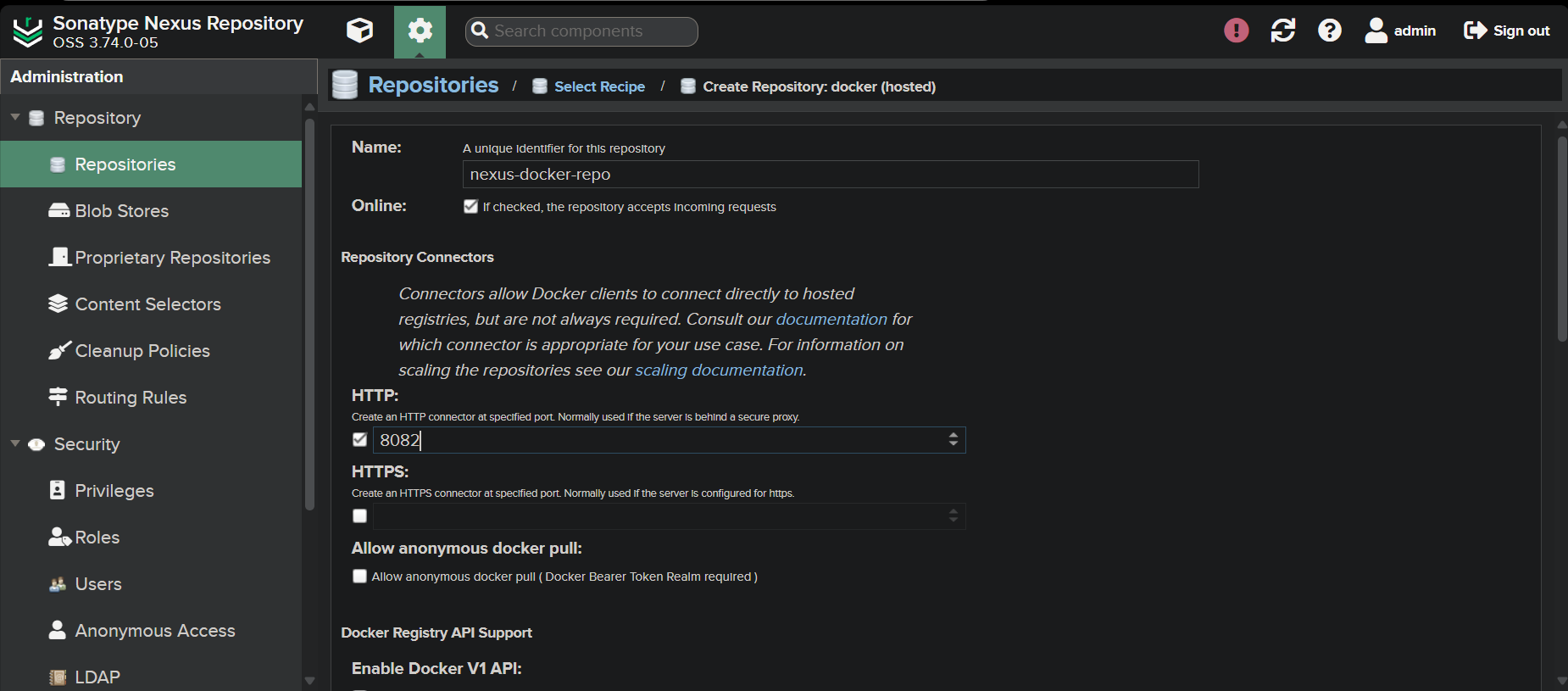
1. Cấu hình Nexus

Tạo Jenkins Credentials cho Nexus:

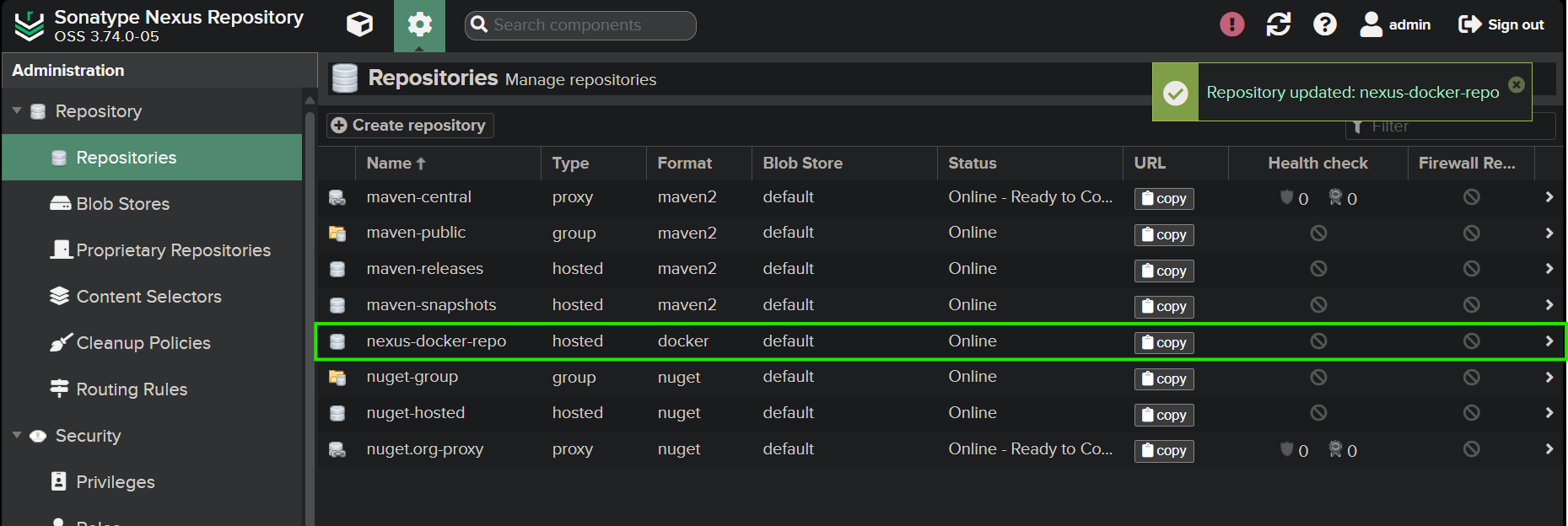


1. Tạo Jenkins Credentials

Tạo private Docker repo:

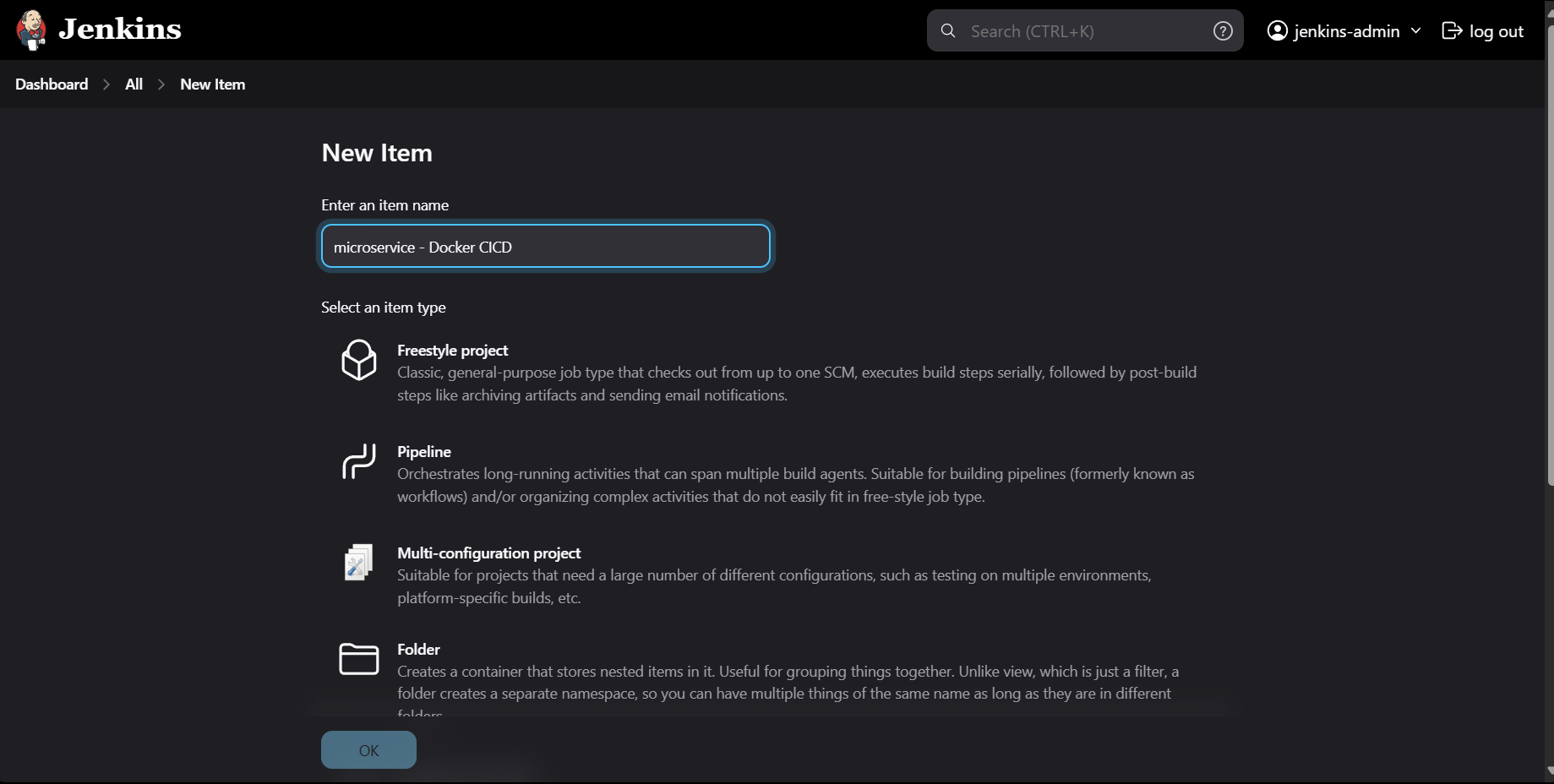


1. Private Docker repo (1)



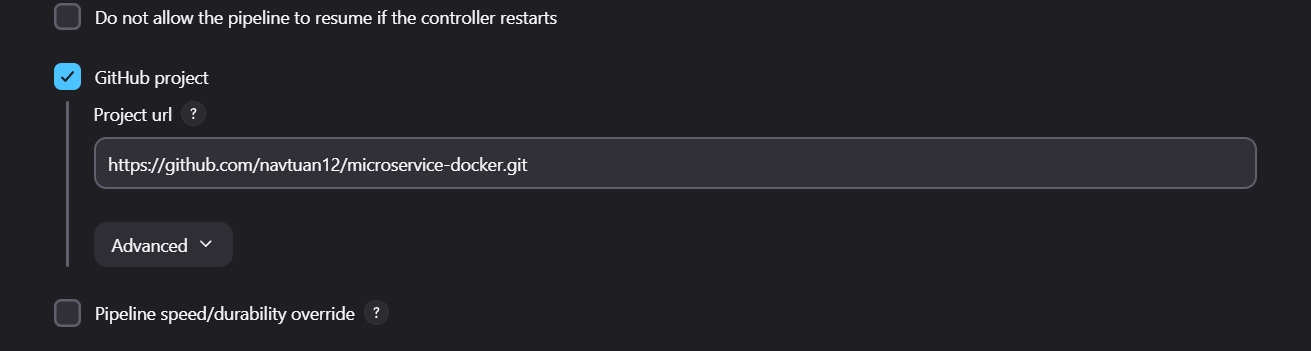
1. Private Nexus repo (2)

\* Tạo Job CICD:

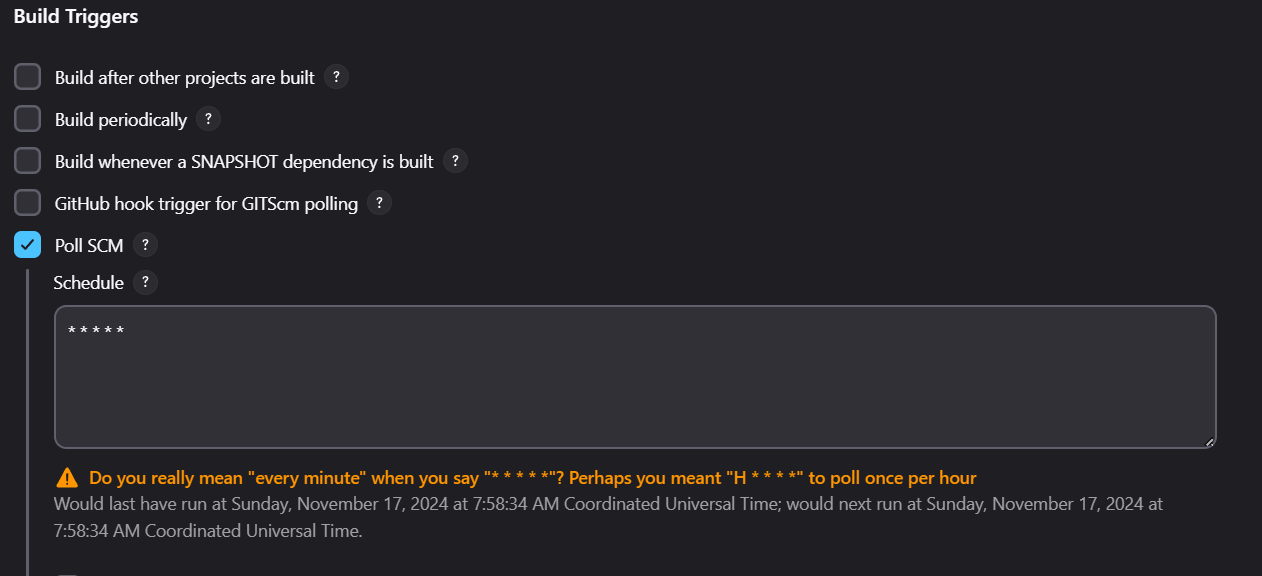


1. Tạo CI/CD

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

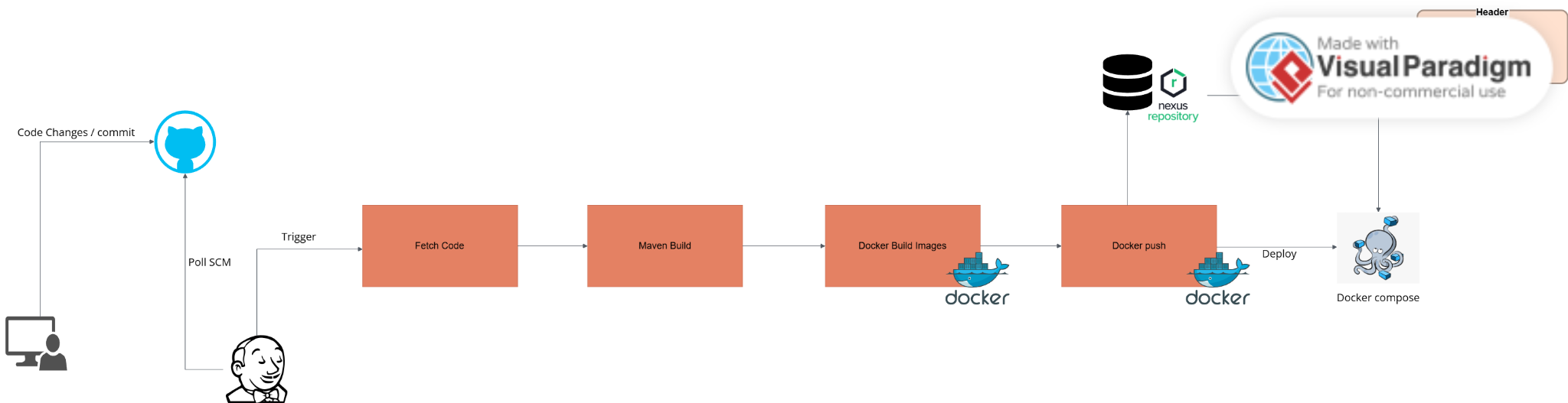


1. Add Github repo



1. Setup Schedule

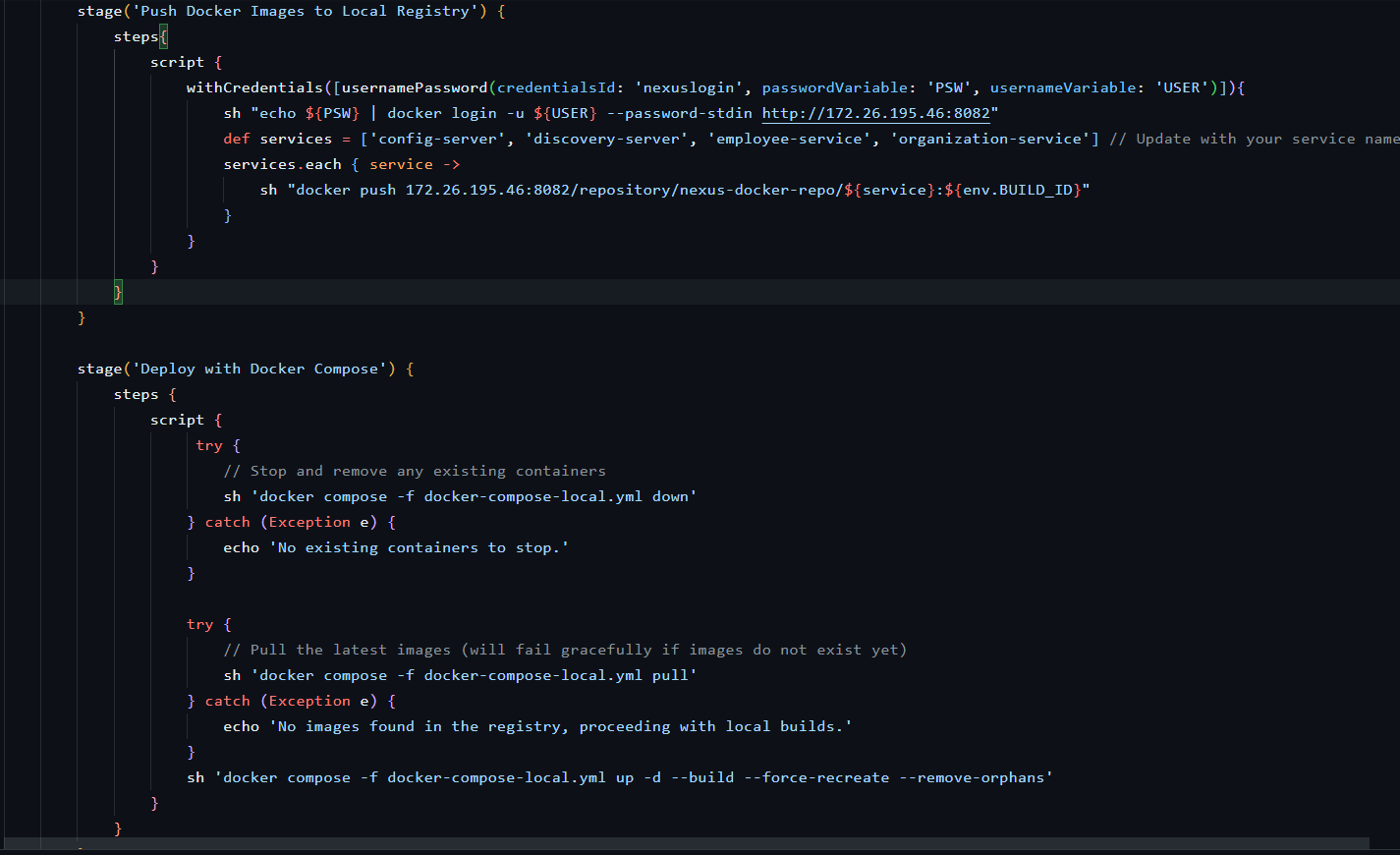
\* Thiết kế pipeline:



1. Kiến trúc pipeline

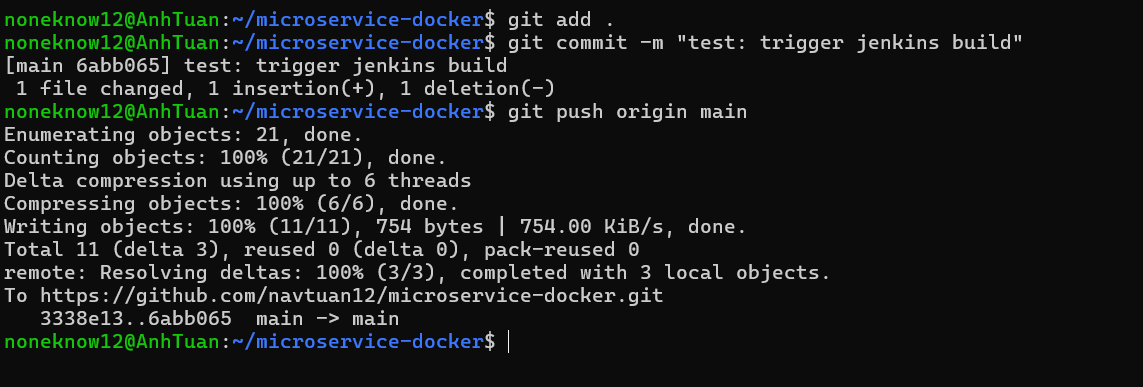


1. Script run Jenkins



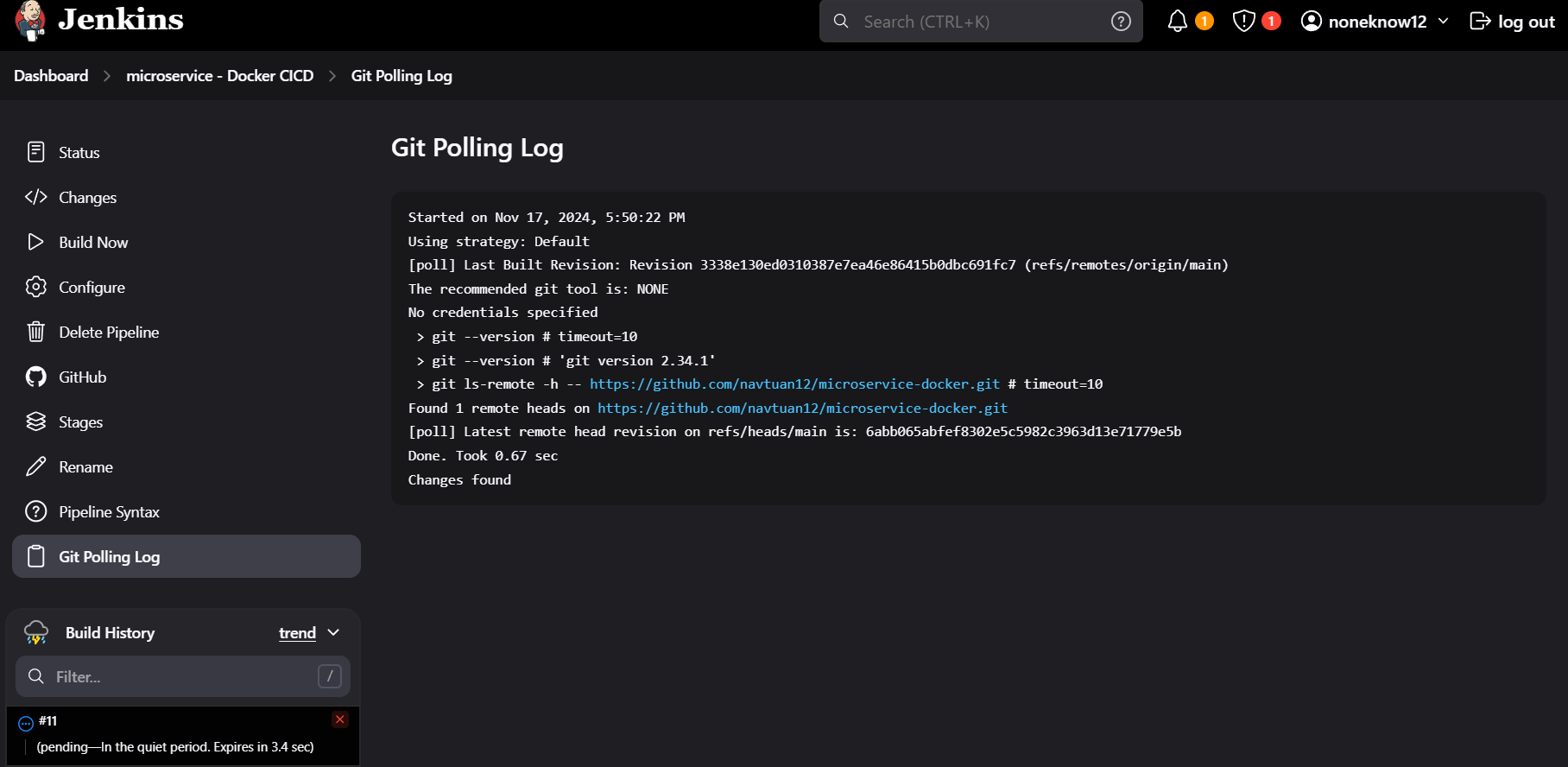
1. Script run Jenkins

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



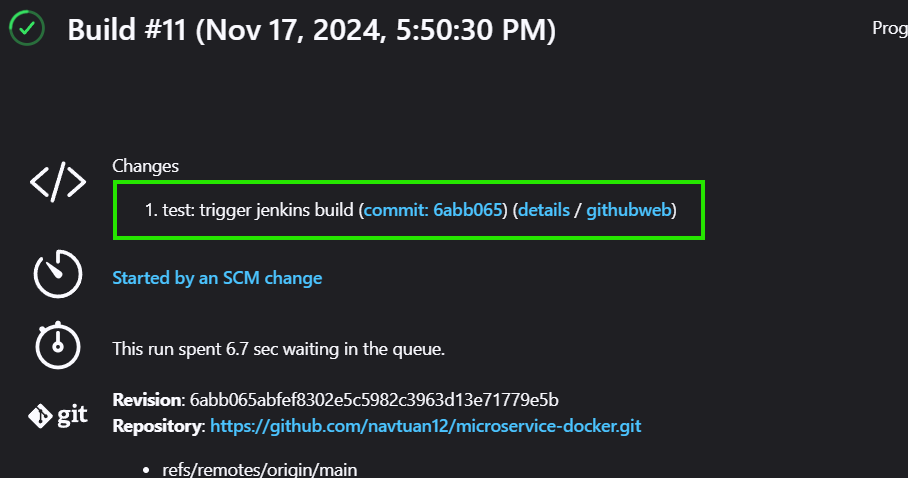
1. Commit lên github

Kiểm tra git polling log:



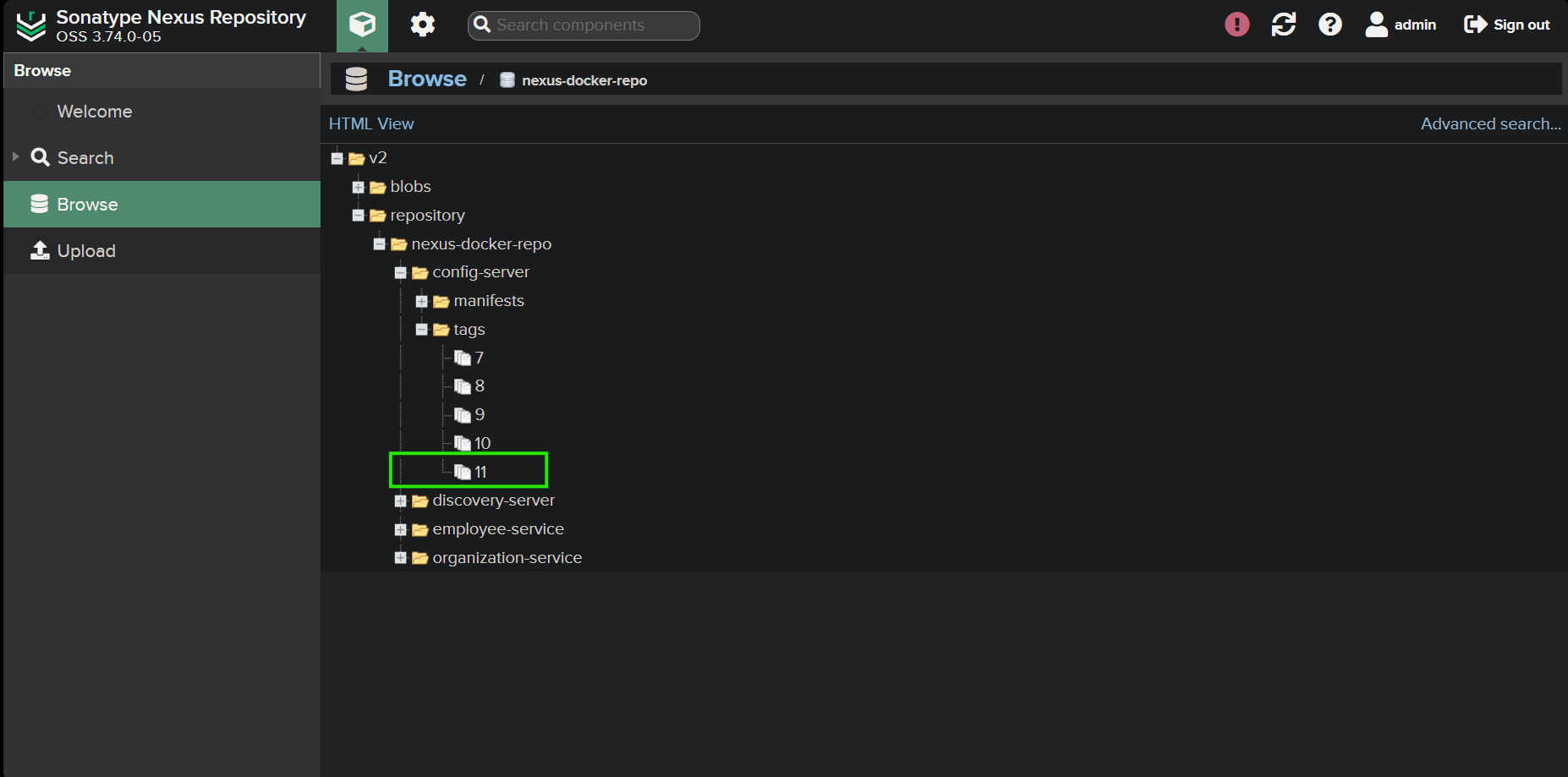
1. Git polling log

Dòng build lần thứ 11 kiểm tra Jenkins build



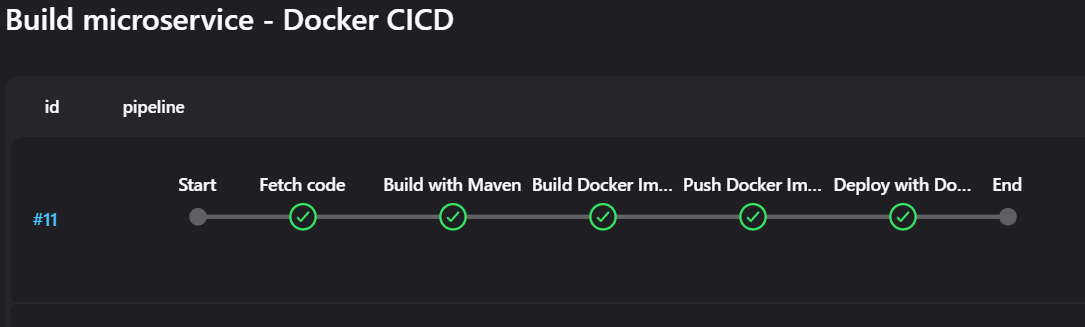
1. Check Jenkins build

Kiểm tra lần build bên Nexus



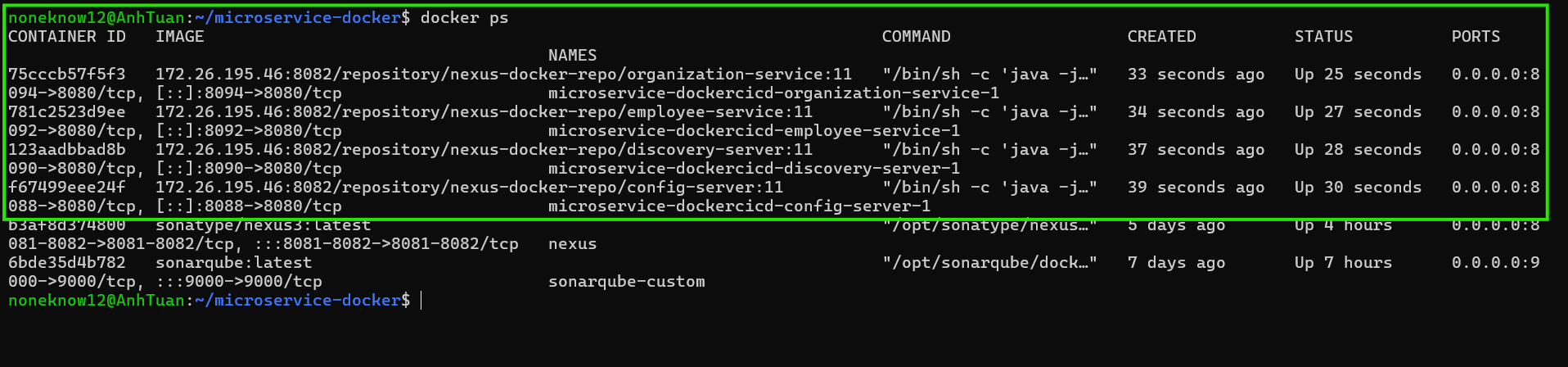
1. Kiểm tra bên Nexus

Kết quả buid, test và deploy thành công



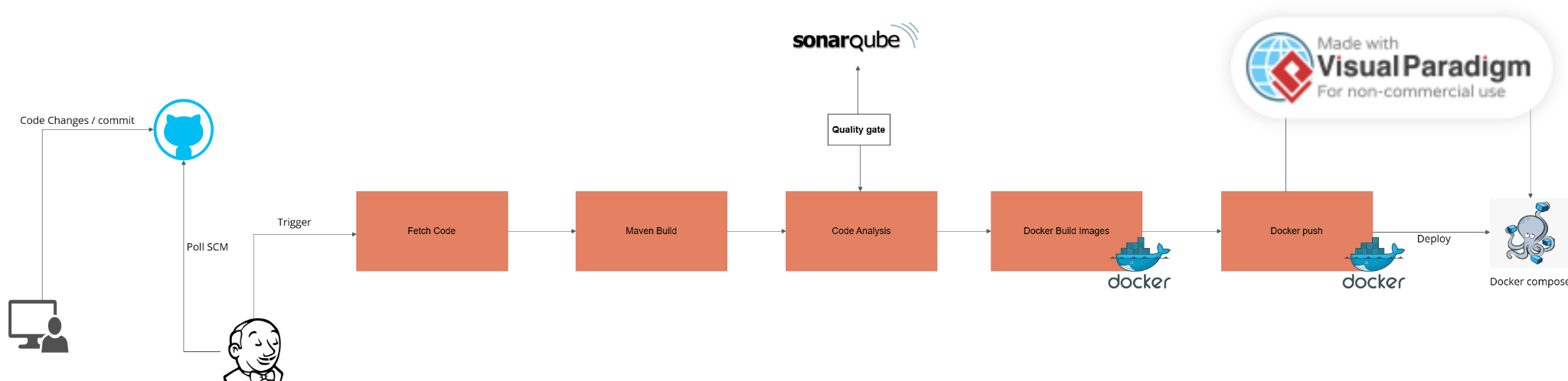
1. CI/CD chạy thành công

Tất cả container đều được update lên version mới nhất



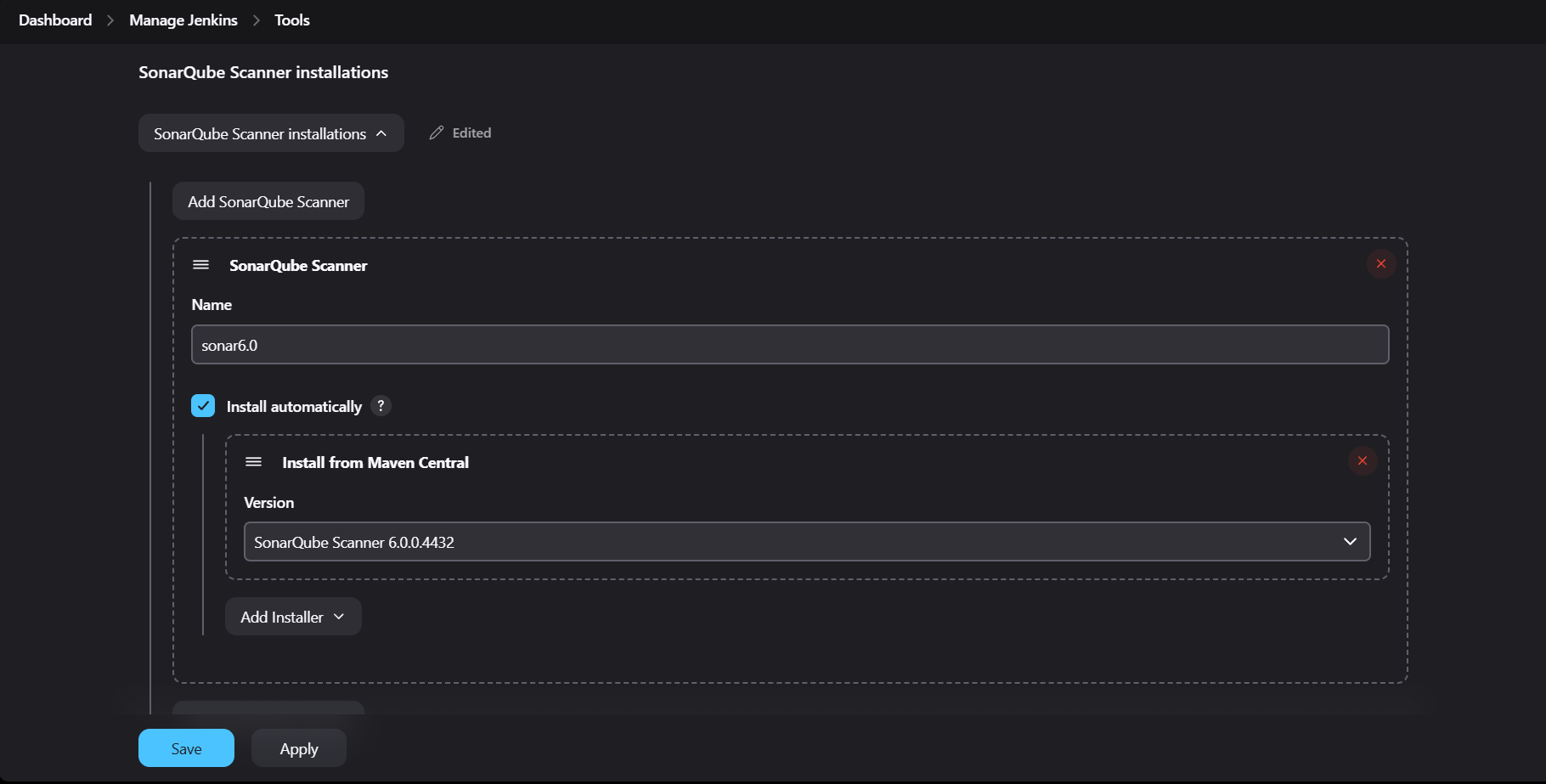
1. Kiểm tra trạng thái các container

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



1. Kiến trúc pipeline tích hợp SonarQube

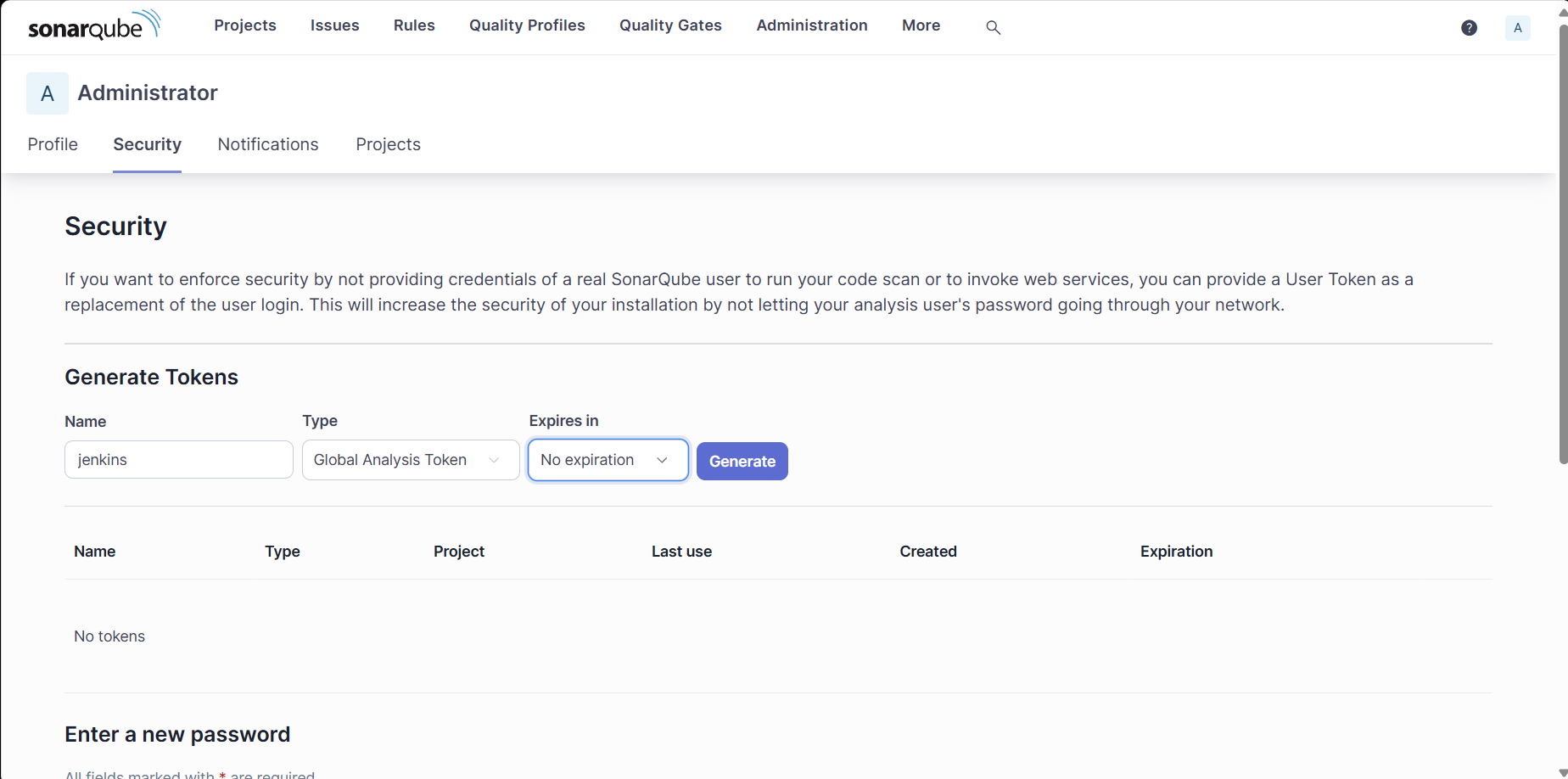
Add SonarQube Scanner trong Manager > Tools:



1. Add SonarQube trên Jenkins

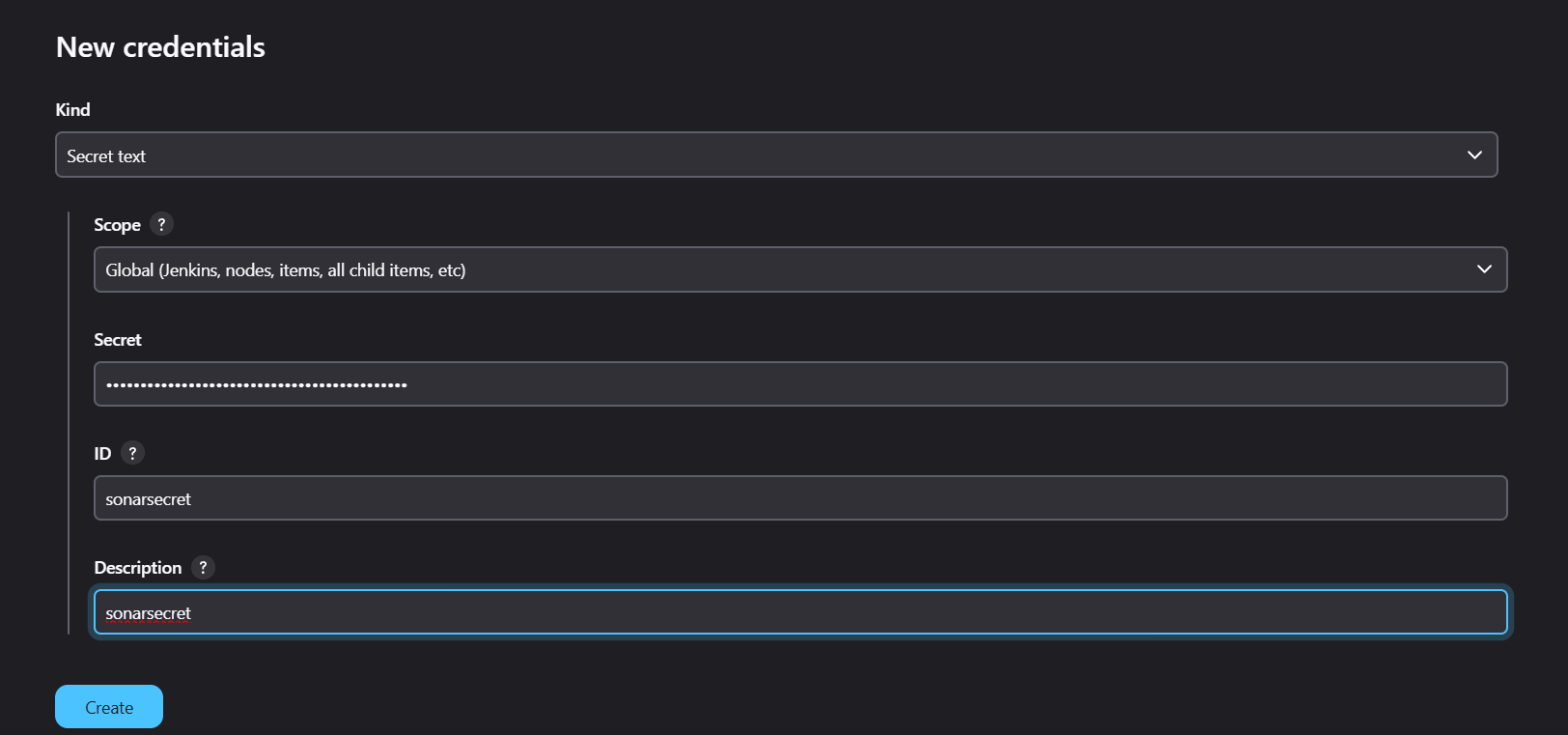
Thêm SonarQube credentials:

Đầu tiên truy cập vào SonarQube để tạo secret key



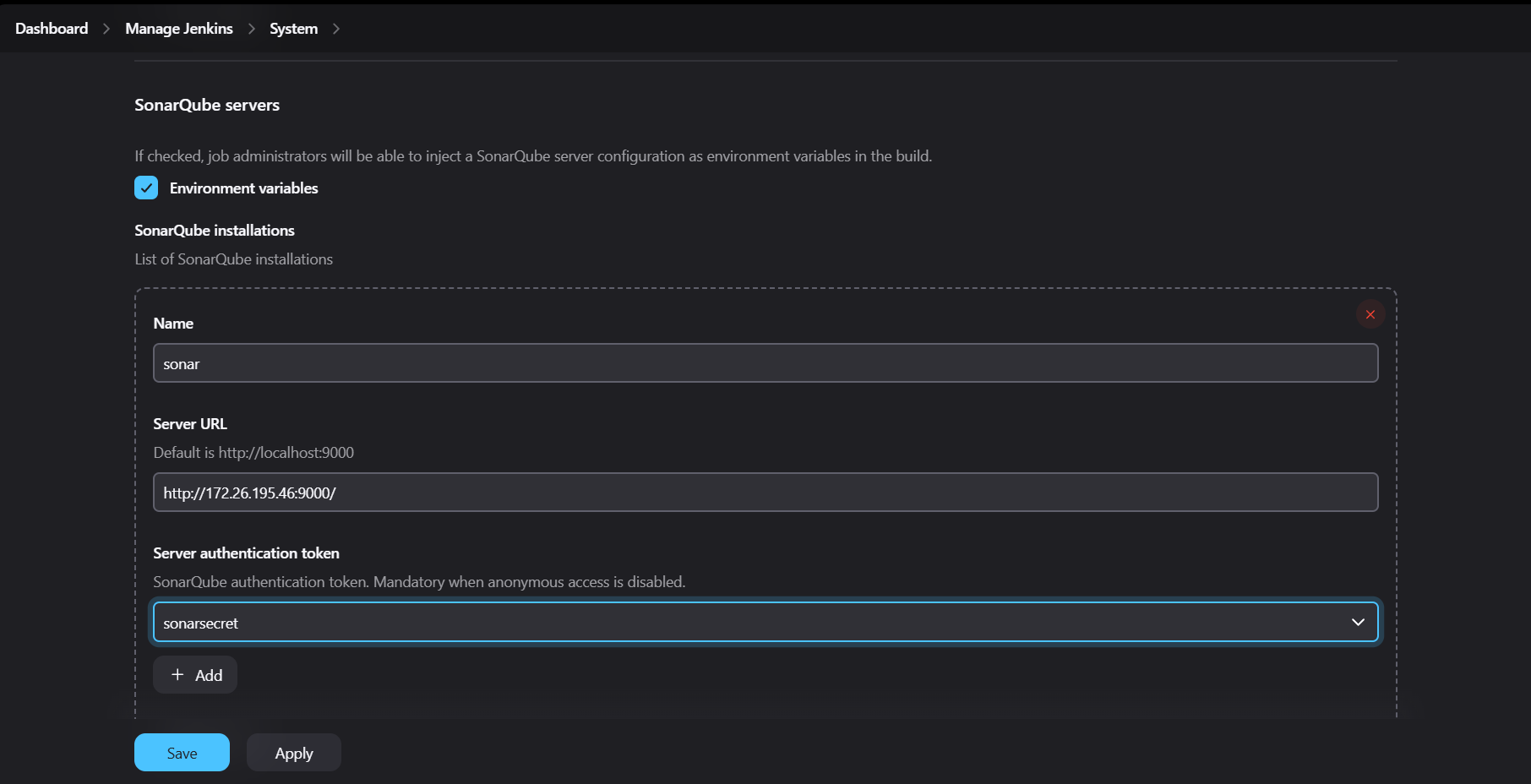
1. Lấy SonarQube credentials

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



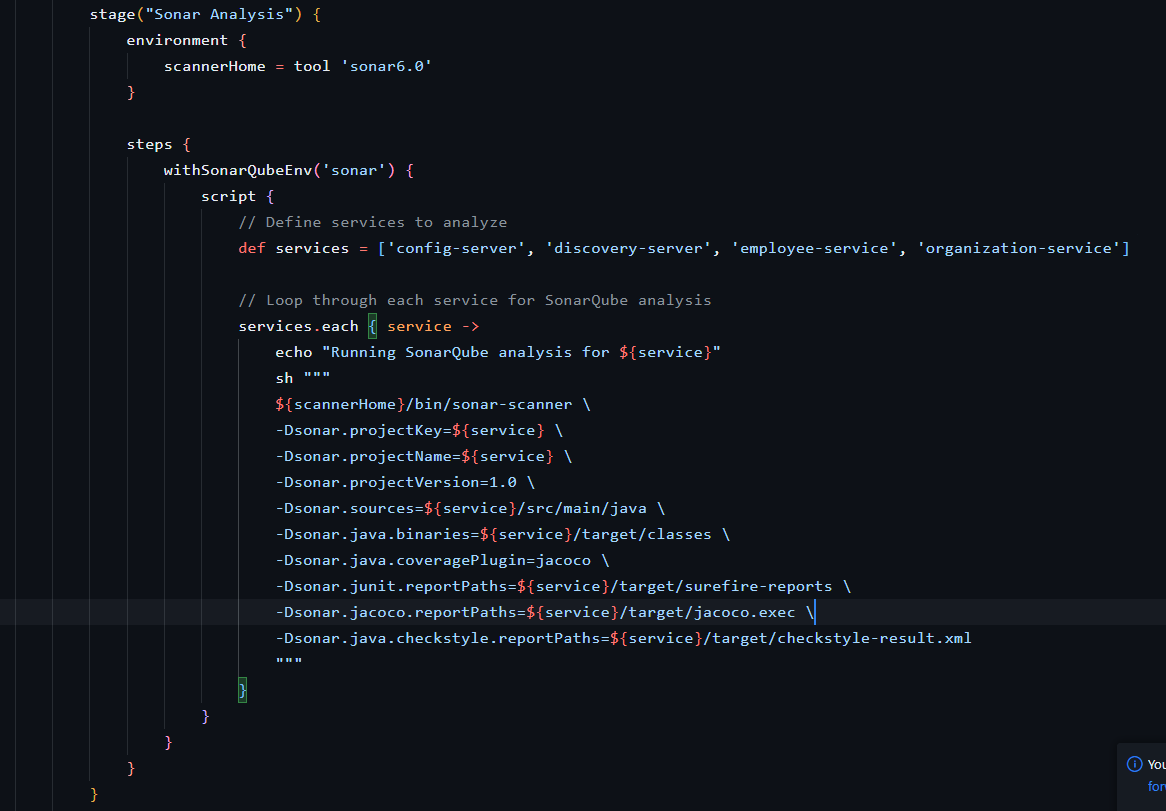
1. Tạo key

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



1. Sử dụng key vừa tạo

Thêm stage vào JenkinsFile:



1. Thêm Stage

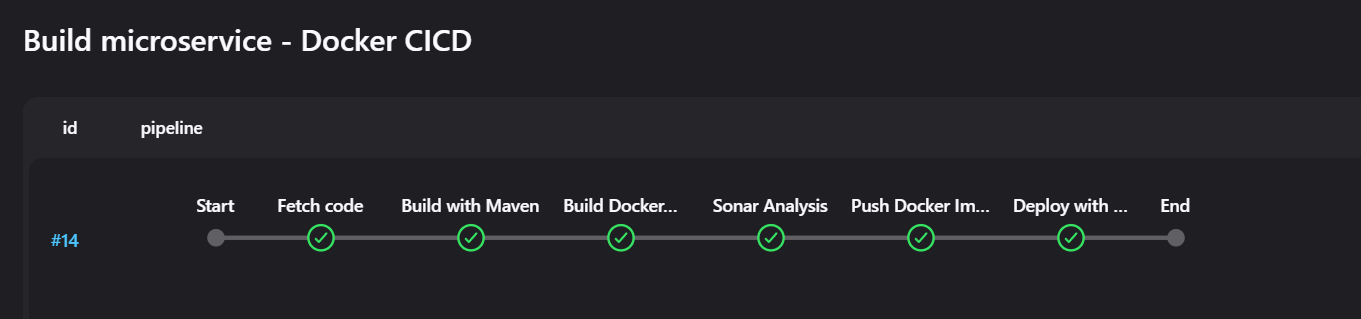
Kết quả:

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

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



1. Kiểm tra trên SonarQube



1. Tích hợp thành công