TRIỂN KHAI HỆ THỐNG

3. TỰ ĐỘNG HÓA HẠ TẦNG VỚI AWS CLOUDFORMATION

3.1.1. Giới thiệu về AWS CloudFormation

AWS CloudFormation là một dịch vụ Infrastructure as Code (IaC), cho phép tự động hóa việc tạo, quản lý và triển khai tài nguyên AWS thông qua các tệp mẫu (template) viết bằng JSON hoặc YAML. Với CloudFormation, người dùng có thể dễ dàng định nghĩa toàn bộ cơ sở hạ tầng trong một tệp duy nhất, đảm bảo tính nhất quán, khả năng tái sử dụng và tự động hóa cao.

Dịch vụ này hỗ trợ quản lý các tài nguyên AWS như EC2, S3, VPC, RDS, API Gateway và nhiều dịch vụ khác. CloudFormation không chỉ giúp giảm thiểu lỗi cấu hình thủ công mà còn tối ưu hóa quy trình triển khai, phù hợp với các hệ thống phức tạp hoặc môi trường yêu cầu tính linh hoạt cao. Đây là công cụ lý tưởng để triển khai hệ thống nhanh chóng và nhất quán, đặc biệt trong các ứng dụng đám mây.

Lý do sử dụng AWS CloudFormation trong triển khai

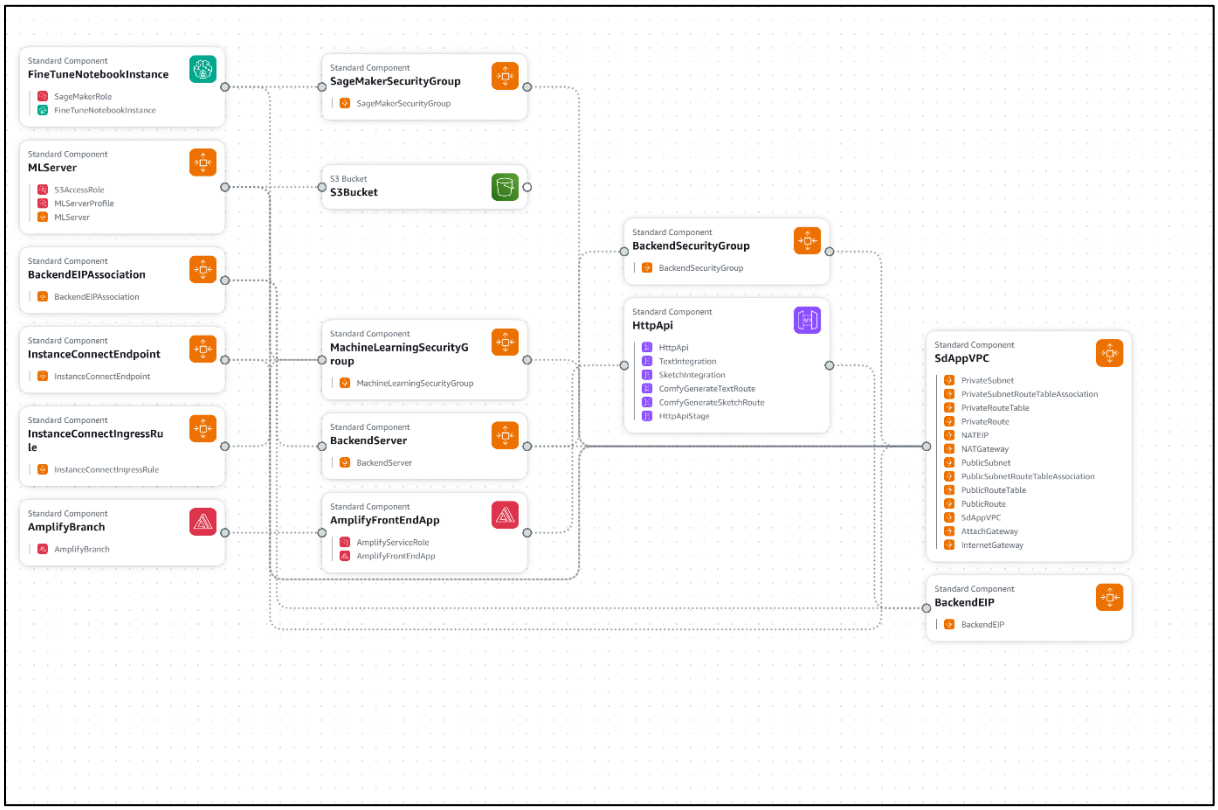
- Tự động hóa: Tất cả tài nguyên được triển khai tự động dựa trên định nghĩa trong template, giúp tiết kiệm thời gian và công sức.
- Tính nhất quán: Đảm bảo rằng cấu hình hạ tầng giống hệt nhau giữa các môi trường (dev, staging, production).
- Dễ dàng quản lý: Mọi cấu hình được quản lý dưới dạng mã (Infrastructure as Code), dễ dàng bảo trì, cập nhật và mở rộng.

3.1.2. Thiết kế cơ sở hạ tầng ứng dụng với AWS CloudFormation Template

Hệ thống web app sử dụng AWS CloudFormation, thông qua việc định nghĩa các tài nguyên cần thiết cho ứng dụng trong một file template YAML. Mục tiêu là tự động hóa việc tạo dựng các tài nguyên AWS, đảm bảo tính nhất quán và dễ dàng tái sử dụng trong các môi trường khác nhau (như phát triển, thử nghiệm và sản xuất). Việc triển khai hạ tầng bằng CloudFormation giúp giảm thiểu các lỗi cấu hình thủ công và tăng hiệu quả vận hành hệ thống.

Template YAML của AWS CloudFormation bao gồm các thành phần chính sau:

- Resources: Định nghĩa các tài nguyên AWS cần triển khai, như EC2, S3, VPC, API Gateway, v.v.
- Parameters: Các giá trị tùy chỉnh có thể thay đổi khi triển khai, ví dụ như loại EC2, kích thước ổ đĩa, v.v.
- Outputs: Các thông tin quan trọng sau khi triển khai, như ARN của các tài nguyên hoặc địa chỉ IP của các EC2 instances.



Hình 3.1: Sơ đồ các tài nguyên AWS được tạo với AWS CloudFormation

3.1.2.1. Cấu hình mạng (Networking Configuration)

a) VPC (Virtual Private Cloud)

VPC (Virtual Private Cloud) là một mạng ảo riêng biệt trong AWS, cung cấp môi trường mạng cô lập để triển khai tài nguyên đám mây.

- *SdAppVPC*: Tạo Virtual Private Cloud (VPC) với CIDR (Classless Inter-Domain Routing) 172.16.0.0/16, là mạng riêng để chứa các tài nguyên.

```
SdAppVPC:
  Type: AWS::EC2::VPC
  Properties:
    CidrBlock: 172.16.0.0/16
    EnableDnsSupport: true
    EnableDnsHostnames: true
```

b) Subnet

Subnet là ột phân đoạn mạng con trong VPC, được sử dụng để nhóm các tài nguyên theo khu vực và mục đích, với khả năng kiểm soát lưu lượng truy cập chi tiết.

- PublicSubnet: Subnet công khai cho các tài nguyên cần truy cập từ internet (CIDR 172.16.0.0/24).
- PrivateSubnet: Subnet riêng cho tài nguyên không cần trực tiếp truy cập từ internet (CIDR 172.16.1.0/24).

```
PublicSubnet:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref SdAppVPC
    CidrBlock: 172.16.0.0/24
    MapPublicIpOnLaunch: true
PrivateSubnet:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref SdAppVPC
    CidrBlock: 172.16.1.0/24
    MapPublicIpOnLaunch: false
```

c) Internet Gateway

Internet Gateway là thành phần cho phép tài nguyên trong VPC giao tiếp với Internet công cộng, đồng thời nhận kết nối từ Internet đến VPC.

- InternetGateway: Tài nguyên để kết nối VPC với internet.
- AttachGateway: Đính Internet Gateway vào VPC.

```
InternetGateway:
  Type: AWS::EC2::InternetGateway
AttachGateway:
  Type: AWS::EC2::VPCGatewayAttachment
  Properties:
    VpcId: !Ref SdAppVPC
    InternetGatewayId: !Ref InternetGateway
```

d) NAT Gateway

NAT Gateway là Gateway đóng vai trò làm trung gian để tài nguyên trong subnet riêng tư truy cập Internet mà vẫn giữ được tính bảo mật.

- NATGateway: Cho phép các tài nguyên trong private subnet kết nối với internet để tải dữ liệu nhưng không bị truy cập từ ngoài.
- NATEIP: Elastic IP gắn với NAT Gateway.

```
NATGateway:
  Type: AWS::EC2::NatGateway
  Properties:
    AllocationId: !GetAtt NATEIP.AllocationId
    SubnetId: !Ref PublicSubnet
NATEIP:
  Type: AWS::EC2::EIP
  Properties:
    Domain: vpc
```

e) Route Tables và Routes

Route Tables là bảng định tuyến xác định cách lưu lượng mạng được điều hướng trong VPC, kết nối giữa các subnet và gateway. Routes là các quy tắc trong bảng định tuyến, chỉ rõ điểm đích và cổng giao tiếp cho các gói tin đi qua mạng.

- PublicRouteTable: Bảng route cho public subnet, có route tới internet thông qua Internet Gateway.
- PrivateRouteTable: Bảng route cho private subnet, có route tới internet thông qua NAT Gateway.

```
PublicRouteTable:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref SdAppVPC

PublicRoute:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PublicRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    GatewayId: !Ref InternetGateway

PrivateRouteTable:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref SdAppVPC

PrivateRoute:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PrivateRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId: !Ref NATGateway
```

f) Subnet-Route Table Association

Subnet-Route Table Association liên kết giữa một subnet và bảng định tuyến, xác định cách mạng con này giao tiếp với các thành phần mạng khác.

- *Gắn từng subnet với bảng route tương ứng.*

```
PublicSubnetRouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    SubnetId: !Ref PublicSubnet
```

```
    RouteTableId: !Ref PublicRouteTable

  PrivateSubnetRouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PrivateSubnet
      RouteTableId: !Ref PrivateRouteTable
```

g) Security Groups và Instance Connect Endpoint

Security Group (SG) là lớp bảo mật để kiểm soát luồng dữ liệu ra/vào các tài nguyên trong AWS.

SageMakerSecurityGroup:

- Ingress (Inbound): Cho phép tất cả loại kết nối (IpProtocol: -1) trong nội bộ VPC (172.16.0.0/16).
- Egress (Outbound): Cho phép tất cả loại kết nối ra ngoài Internet (0.0.0.0/0).

BackendSecurityGroup:

- Ingress: Cho phép SSH (port 22), HTTP (port 80), và HTTPS (port 443) từ mọi nguồn (0.0.0.0/0).
- Egress: Cho phép mọi loại kết nối ra ngoài Internet (0.0.0.0/0).

MachineLearningSecurityGroup:

- Ingress: Cho phép truy cập từ 172.16.0.0/24 (nội bộ) qua các cổng: 8188–8189 (Custom ứng dụng ML), 80 (HTTP), 443 (HTTPS)
- Egress: Cho phép mọi loại kết nối ra ngoài Internet (0.0.0.0/0).

Instance Connect Endpoint của AWS là một giải pháp bảo mật và tiện lợi, cho phép người dùng kết nối đến các phiên bản Amazon EC2 trong mạng riêng mà không cần mở cổng SSH hoặc sử dụng public IP, đồng thời giảm thiểu rủi ro bảo mật liên quan đến quản lý truy cập từ xa.

- InstanceConnectEndpoint: sử dụng MachineLearningSecurityGroup để quản lý bảo mật và triển khai trong PrivateSubnet, cho phép kết nối an toàn đến các EC2 instance mà không cần mở cổng SSH.

```
SageMakerSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Allow SageMaker access
    VpcId: !Ref SdAppVPC
    SecurityGroupIngress:
      - IpProtocol: '-1'
        CidrIp: 172.16.0.0/16
    SecurityGroupEgress:
      - IpProtocol: '-1'
        CidrIp: 0.0.0.0/0

 BackendSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Security Group for Backend Server
    VpcId: !Ref SdAppVPC
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: 0.0.0.0/0 # SSH Access (có thể hạn chế IP)
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
        CidrIp: 0.0.0.0/0 # SSH Access (có thể hạn chế IP)
      - IpProtocol: tcp
        FromPort: 443
        ToPort: 443
        CidrIp: 0.0.0.0/0 # SSH Access (có thể hạn chế IP)
      - IpProtocol: tcp
```

```
      FromPort: 443
      ToPort: 443
      CidrIp: 0.0.0.0/0 # SSH Access (có thể hạn chế IP)
    SecurityGroupEgress:
    - IpProtocol: '-1'
      CidrIp: 0.0.0.0/0 # Outbound traffic

MachineLearningSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Security Group for ML Server
    VpcId: !Ref SdAppVPC
    SecurityGroupIngress:
    - IpProtocol: tcp
      FromPort: 8188
      ToPort: 8189
      CidrIp: 172.16.0.0/24
    - IpProtocol: tcp
      FromPort: 80
      ToPort: 80
      CidrIp: 172.16.0.0/24
    - IpProtocol: tcp
      FromPort: 443
      ToPort: 443
      CidrIp: 172.16.0.0/24
    - IpProtocol: tcp
      FromPort: 443
      ToPort: 443
      CidrIp: 172.16.0.0/24
    SecurityGroupEgress:
    - IpProtocol: '-1'
```

```
      CidrIp: 0.0.0.0/0


InstanceConnectEndpoint:
  Type: "AWS::EC2::InstanceConnectEndpoint"
  Properties:
    SecurityGroupIds:
      - !Ref MachineLearningSecurityGroup
    SubnetId: !Ref PrivateSubnet
```

3.1.2.2. Kho lưu trữ mô hình (Amazon S3)

Dịch vụ AWS S3 cung cấp giải pháp lưu trữ đối tượng linh hoạt, bền vững và bảo mật, hỗ trợ quản lý dữ liệu hiệu quả cho nhiều ứng dụng khác nhau trên nền tảng đám mây AWS.

- BucketName: Tên của bucket là my-basic-s3-bucket. Bạn có thể thay đổi để phù hợp với yêu cầu của mình. Tên này phải là duy nhất trên toàn cầu trong AWS.
- AccessControl: Đặt quyền truy cập là Private để chỉ chủ sở hữu bucket mới có quyền truy cập.
- VersioningConfiguration: Kích hoạt tính năng Versioning để lưu trữ nhiều phiên bản của cùng một tệp.

```
S3Bucket:
  Type: AWS::S3::Bucket
  Properties:
    BucketName: sd-app-s3
    AccessControl: Private
    VersioningConfiguration:
      Status: Enabled
```

3.1.2.3. Quản lý quyền truy cập (IAM Roles and Policies)

a) S3AccessRole

IAM role cho phép các dịch vụ như EC2 và SageMaker truy cập vào S3 để thực hiện các thao tác như tải lên và tải xuống dữ liệu từ S3 bucket (sd-app-s3).

```
S3AccessRole:
```

```
Type: AWS::IAM::Role
Properties:
 AssumeRolePolicyDocument:
  Version: '2012-10-17'
  Statement:
   - Effect: Allow
     Principal:
      Service:
       - ec2.amazonaws.com
       - sagemaker.amazonaws.com
     Action: sts:AssumeRole
 Path: /
 Policies:
  - PolicyName: MyS3AccessPolicy
    PolicyDocument:
     Version: '2012-10-17'
     Statement:
      - Effect: Allow
        Action:
         - s3:GetObject
         - s3:PutObject
         - s3:ListBucket
        Resource:
         - !Sub arn:aws:s3:::sd-app-s3
         - !Sub arn:aws:s3:::sd-app-s3/*
```

b) SageMakerRole

IAM role dành cho SageMaker với quyền truy cập đầy đủ vào dịch vụ SageMaker thông qua managed policy AmazonSageMakerFullAccess.

```
SageMakerRole:
 Type: AWS::IAM::Role
 Properties:
```

```
  AssumeRolePolicyDocument:
    Version: '2012-10-17'
    Statement:
      - Effect: Allow
        Principal:
          Service:
            - sagemaker.amazonaws.com
        Action: sts:AssumeRole
  ManagedPolicyArns:
    - arn:aws:iam::aws:policy/AmazonSageMakerFullAccess
  Path: /
```

c) AmplifyServiceRole

IAM role cho phép AWS Amplify truy cập vào các tài nguyên AWS, bao gồm quyền truy cập quản trị đầy đủ cho Amplify và các tài nguyên AWS.

```
AmplifyServiceRole:
  Type: AWS::IAM::Role
  Properties:
    RoleName: AmplifyServiceRole
    AssumeRolePolicyDocument:
      Version: "2012-10-17"
      Statement:
        - Effect: "Allow"
          Principal:
            Service: "amplify.amazonaws.com"
          Action: "sts:AssumeRole"
    ManagedPolicyArns:
      - arn:aws:iam::aws:policy/AdministratorAccess-Amplify
      - arn:aws:iam::aws:policy/AdministratorAccess
    Path: "/"
```

d) MLServerProfile

Instance profile cho EC2 instance (MLServer), gán IAM role S3AccessRole để EC2 instance có quyền truy cập vào S3.

```
MLServerProfile:
  Type: AWS::IAM::InstanceProfile
  Properties:
    Roles:
      - !Ref S3AccessRole
```

3.1.2.4. *Máy chủ EC2 (EC2 Instances)* và SageMaker Notebook

a) BackendServer (Máy chủ Backend)

Máy chủ BackendServer:

- *Sử dụng một* Amazon Linux 2023 AMI (*ami-0c80e2b6ccb9ad6d1)*
- *Loại instance là t2.medium với 2 vCPU và 4 GB RAM.*
- *Subnet là PublicSubnet, và security group là BackendSecurityGroup.*
- *Máy chủ và một key pair sd-app-key-pair cho SSH access.*
- *Được gắn BackendEIP thông qua BackendEIPAssociation để cho phép truy cập từ bên ngoài, đảm bảo IP của máy chủ không thay đổi khi instance EC2 bị dừng hoặc khởi động lại*

```
BackendServer:
  Type: AWS::EC2::Instance
  Properties:
    InstanceType: t2.medium
    SubnetId: !Ref PublicSubnet
    SecurityGroupIds:
      - !Ref BackendSecurityGroup
    ImageId: ami-0c80e2b6ccb9ad6d1
    KeyName: sd-app-key-pair

BackendEIP:
    Type: AWS::EC2::EIP
```

```
  Properties:
    Domain: vpc
BackendEIPAssociation:
  Type: AWS::EC2::EIPAssociation
  Properties:
    AllocationId: !GetAtt BackendEIP.AllocationId
    InstanceId: !Ref BackendServer
```

b) MLServer (Máy chủ Machine Learning)

Máy chủ MLServer (chạy Comfy UI):

- *Sử dụng một Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.5 (Ubuntu 22.04)* AMI (*ami-0c80e2b6ccb9ad6d1)*
- *Sử dụng loại instance g5.xlarge với 4 vCPU, 16 GB RAM và hỗ trợ GPU.*
- *Được triển khai trong PrivateSubnet, với security group là MachineLearningSecurityGroup.*
- *Sử dụng instance profile là MLServerProfile để gán các IAM roles.*

```
MLServer:
  Type: AWS::EC2::Instance
  Properties:
    InstanceType: g5.xlarge
    SubnetId: !Ref PrivateSubnet
    SecurityGroupIds:
      - !Ref MachineLearningSecurityGroup
    ImageId: ami-0c9465cbafe2032b2
    KeyName: sd-app-key-pair
    IamInstanceProfile: !Ref MLServerProfile
```

c) FineTuneNotebookInstance (Máy chủ SageMaker Notebook)

FineTuneNotebookInstance là một SageMaker Notebook Instance với loại ml.g5.xlarge, gán IAM role SageMakerRole, lưu trữ 100GB và sử dụng subnet PrivateSubnet.

```
FineTuneNotebookInstance:
```

```
Type: AWS::SageMaker::NotebookInstance
Properties:
  InstanceType: ml.g5.xlarge
  RoleArn: !GetAtt SageMakerRole.Arn
  VolumeSizeInGB: 100
  SubnetId: !Ref PrivateSubnet
  SecurityGroupIds:
    - !Ref SageMakerSecurityGroup
```

3.1.2.5. *Amplify và Frontend Application*

a) Chuẩn bị Repository và AccessToken với Github

Trước khi kết nối với AWS Amplify, cần chuẩn bị một repository GitHub chứa mã nguồn ứng dụng frontend và tạo AccessToken cho phép AWS Amplify truy cập vào repository. Repository này sẽ được AWS Amplify sử dụng để tự động build và deploy ứng dụng lên môi trường AWS.

Các bước tạo Repository Github:

- *Đăng nhập vào tài khoản Github.*
- *Chọn New Repository từ trang chính của Github.*
- *Điền thông tin cần thiết như tên repository, mô tả và chọn chế độ public hoặc private.*

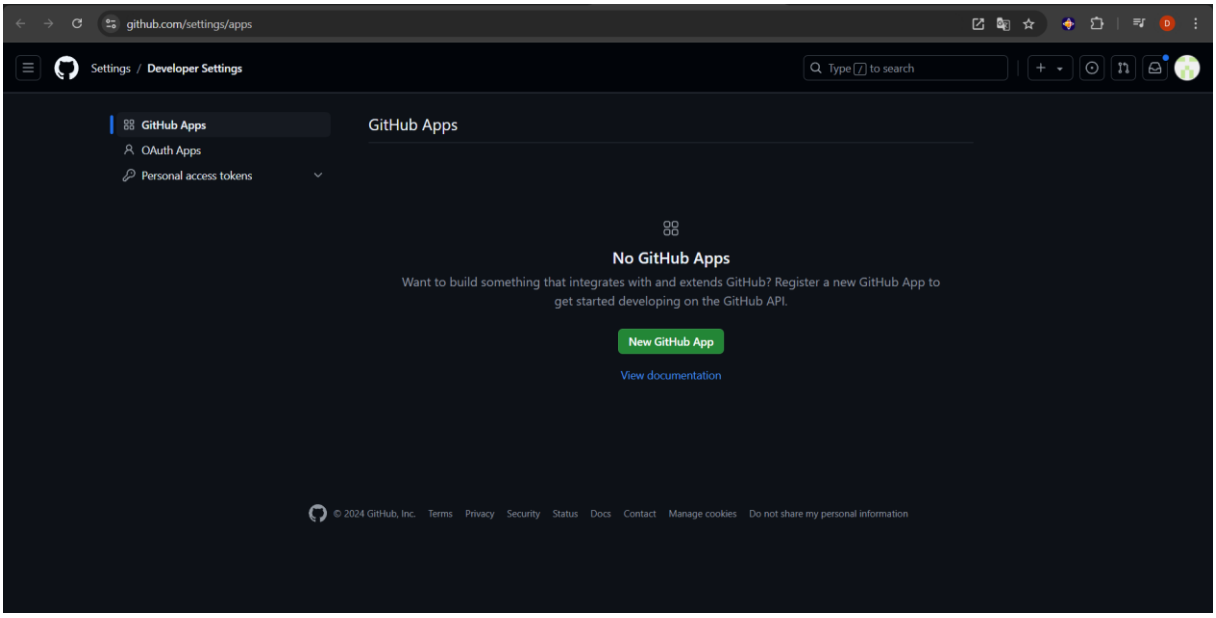
Sau khi tạo xong, sẽ có một URL của repository để sử dụng trong quá trình tích hợp với AWS Amplify. Sau đó, tải mã nguồn của ứng dụng lên repositor vừa tạo.

Repository Github chữa mã nguồn của ứng dụng*:*

https://github.com/hiep20012003/generate-image-app

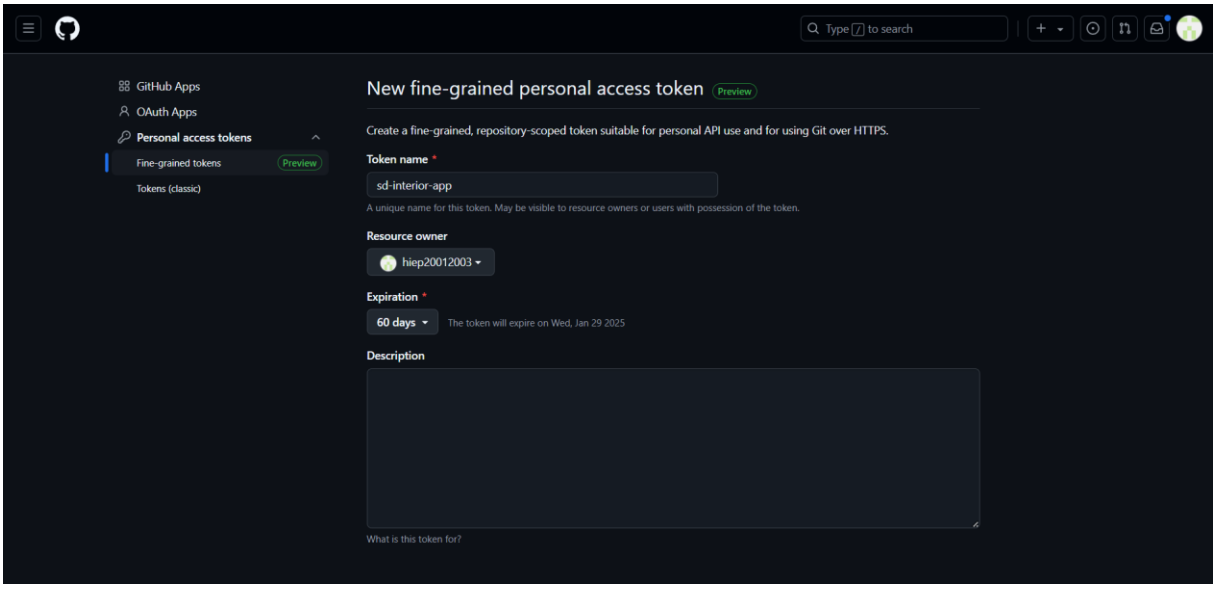
Các bước tạo AccessToken:

- *Bước 1: Truy cập vào đường dẫn https://github.com/settings/apps và đăng nhập vào tài khoản Github.*



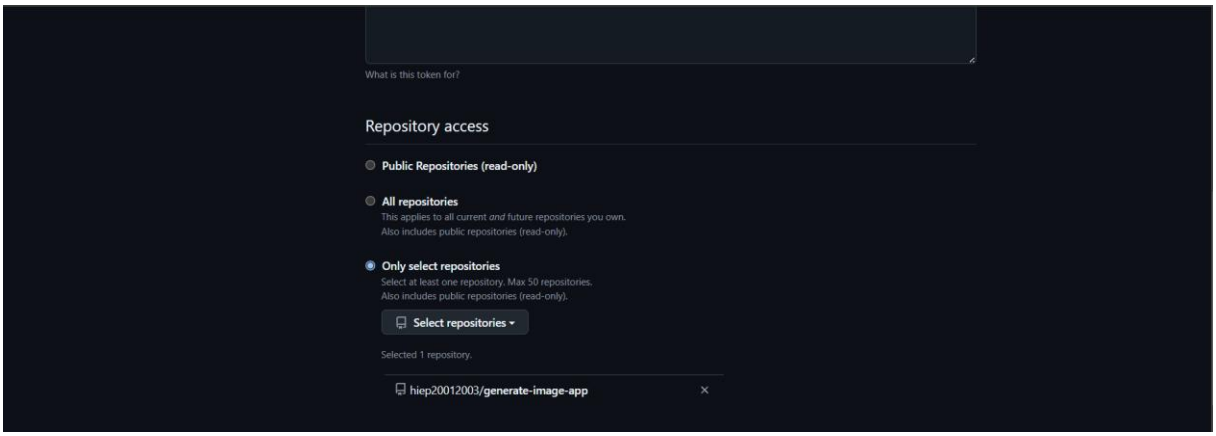
Hình 3.2: Github Developer Settings

- *Bước 2: Truy cập Personal access tokens → Fine-grained tokens → Generate new token. Nhập Token name, Resource owner, Expiration*



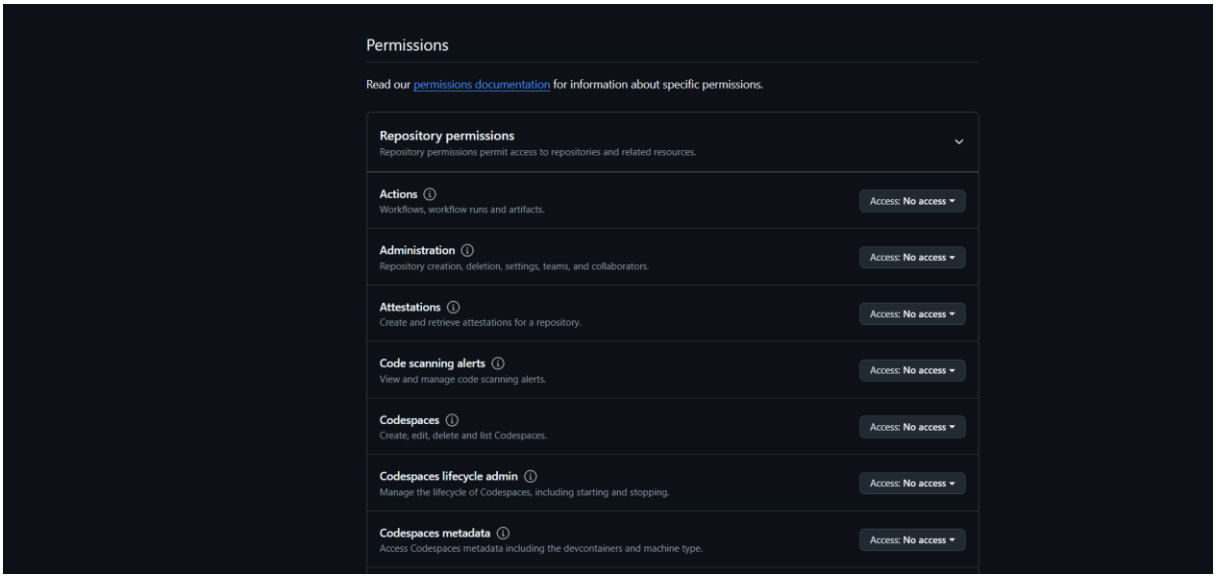
Hình 3.3: Tạo *Fine-grained tokens trong Github*

- *Bước 3: Cấu hình Repository access, chọn Repository chứa mã nguồn ứng dụng*



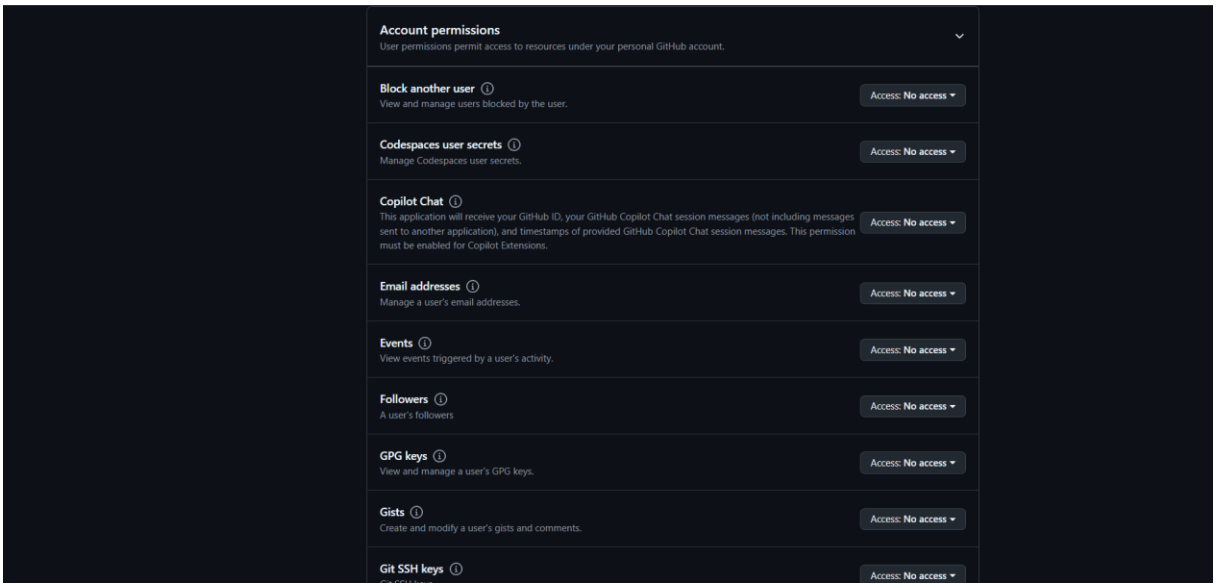
Hình 3.4: ***Cấu hình Repository access cho Fine-grained tokens***

- *Bước 4: Cấu hình Permissions*
  - *Repository Permissions: Cho phép các quyền* Contents, Deployments, Pull requests, Workflows, Metadata, Webhooks



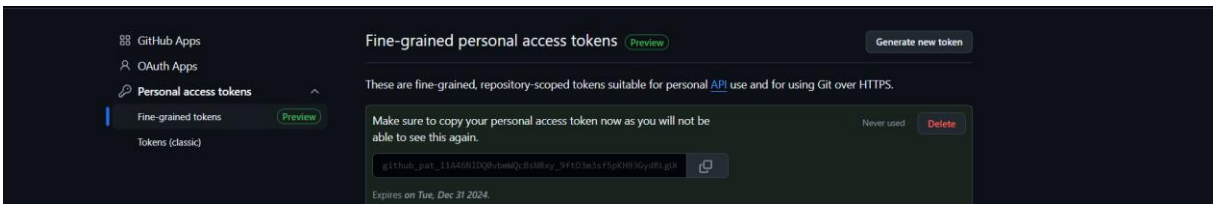
Hình 3.5: ***Cấu hình Repository Permissions cho Fine-grained tokens***

  - *Account Permissions: Cho phép quyền Email addresses*



Hình 3.6: ***Cấu hình Account Permissions cho Fine-grained tokens***

- *Bước 5: Nhấn Generate token. Sau đó copy access token và lưu lại (không thể lấy lại nên cần lưu trữ access token cẩn thận)*



Hình 3.7: ***Fine-grained tokens***

b) Cấu hình AWS Amplify với Github Repository

Các cấu hình quan trọng:

- *Name: Tên ứng dụng.*
- *Repository: URL kho mã nguồn GitHub.*
- *AccessToken: Token truy cập kho GitHub.*
- *IAMServiceRole: Role IAM để Amplify có quyền truy cập dịch vụ AWS.*
- *CustomRules: Quy tắc chuyển hướng cho các ứng dụng SPA (đưa mọi yêu cầu không phải tài nguyên tĩnh đến index.html).*
- *EnvironmentVariables: Biến môi trường cho ứng dụng, như API URL.*
- *BuildSpec: Quy trình build, bao gồm cài đặt, xây dựng, và triển khai ứng dụng frontend.*

```
AmplifyFrontEndApp:
  Type: AWS::Amplify::App
  Properties:
    Name: SdFrontEndApp
    Platform: WEB
    Repository: "https://github.com/hiep20012003/generate-image-app" # URL của repository
    AccessToken: "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"  # Token truy cập
Github
    IAMServiceRole: !GetAtt AmplifyServiceRole.Arn
    CustomRules:
      - Source: "</^[^.]+$|\\.(?!(css|gif|ico|jpg|js|png|txt|svg|woff|ttf)$)([^.]+$)/>"
        Target: "/index.html"
        Status: "200"
    EnvironmentVariables:
      - Name: REACT_APP_API_URL
        Value: !Sub "https://${HttpApi}.execute-api.${AWS::Region}.amazonaws.com/prod"
    BuildSpec: |
      version: 1
      applications:
        - appRoot: client
          frontend:
            phases:
              preBuild:
                commands:
                  - npm install
              build:
                commands:
                  - npm run build
              postBuild:
                commands:
                  - echo "Deployment completed"
```

```
    artifacts:
      baseDirectory: build
      files:
        - '**/*'
    cache:
      paths:
        - node_modules/**/*
```

3.1.2.6. API Gateway và các Routes

- *HttpApi: Tạo một API Gateway với giao thức HTTP, cấu hình CORS để hỗ trợ các yêu cầu từ mọi nguồn, phục vụ cho môi trường triển khai prod.*
- *TextIntegration: Tích hợp API Gateway với dịch vụ backend thông qua HTTP Proxy, cho phép chuyển tiếp yêu cầu đến endpoint /comfy/generate/text trên backend.*
- *SketchIntegration: Tương tự TextIntegration, cấu hình tích hợp chuyển tiếp yêu cầu đến endpoint /comfy/generate/sketch, đảm bảo tính linh hoạt cho các chức năng backend.*
- *ComfyGenerateTextRoute: Định nghĩa route POST /comfy/generate/text, kết nối với TextIntegration để xử lý yêu cầu tạo text thông qua backend.*
- *ComfyGenerateSketchRoute: Định nghĩa route POST /comfy/generate/sketch, cho phép API Gateway định tuyến yêu cầu đến SketchIntegration để tạo sketch.*
- *HttpApiStage: Tạo stage triển khai API Gateway với tên prod, kích hoạt tính năng tự động triển khai giúp giảm thiểu thời gian cập nhật các thay đổi.*

```
# API Gateway HTTP
HttpApi:
  Type: AWS::ApiGatewayV2::Api
  Properties:
    Name: "SdAppApi"                # Tên API Gateway
    ProtocolType: "HTTP"            # Loại giao thức HTTP
    CorsConfiguration:              # Cấu hình CORS
      AllowOrigins:
```

```
      - "*"
    AllowMethods:
      - "GET"
      - "POST"
    AllowHeaders:
      - "*"
  Tags:
    Environment: "prod"

# Integration cho route /comfy/generate/text
TextIntegration:
  Type: AWS::ApiGatewayV2::Integration
  Properties:
    ApiId: !Ref HttpApi
    IntegrationType: "HTTP_PROXY"       # Tích hợp dạng HTTP Proxy
    IntegrationMethod: "ANY"
    IntegrationUri: !Sub "http://${BackendEIP.PublicIp}:80/comfy/generate/text" # Backend URL
    PayloadFormatVersion: "1.0"
    TimeoutInMillis: 29000

# Integration cho route /comfy/generate/sketch
SketchIntegration:
  Type: AWS::ApiGatewayV2::Integration
  Properties:
    ApiId: !Ref HttpApi
    IntegrationType: "HTTP_PROXY"
    IntegrationMethod: "ANY"
    IntegrationUri: !Sub "http://${BackendEIP.PublicIp}:80/comfy/generate/sketch" # Backend
URL
    PayloadFormatVersion: "1.0"
    TimeoutInMillis: 29000
```

```
# Route cho Text Generation
ComfyGenerateTextRoute:
  Type: AWS::ApiGatewayV2::Route
  Properties:
    ApiId: !Ref HttpApi
    RouteKey: "POST /comfy/generate/text"   # Định nghĩa route key
    Target: !Sub "integrations/${TextIntegration}"

# Route cho Sketch Generation
ComfyGenerateSketchRoute:
  Type: AWS::ApiGatewayV2::Route
  Properties:
    ApiId: !Ref HttpApi
    RouteKey: "POST /comfy/generate/sketch"
    Target: !Sub "integrations/${SketchIntegration}"

# Stage cho API Gateway
HttpApiStage:
  Type: AWS::ApiGatewayV2::Stage
  Properties:
    ApiId: !Ref HttpApi
    StageName: "prod"                  # Tên stage là 'prod'
    AutoDeploy: true                   # Tự động triển khai
```

3.1.3. Triển khai cơ sở hạ tầng ứng dụng bằng AWS CloudFormation

3.1.3.1. File AWS CloudFormation Template

```
AWSTemplateFormatVersion: '2010-09-09'
Description: CloudFormation template
Resources:
  SdAppVPC:
```

```
  Type: AWS::EC2::VPC
  Properties:
    CidrBlock: 172.16.0.0/16
    EnableDnsSupport: true
    EnableDnsHostnames: true
PublicSubnet:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref SdAppVPC
    CidrBlock: 172.16.0.0/24
    MapPublicIpOnLaunch: true
    AvailabilityZone: !Select
      - 0
      - !GetAZs ''
PrivateSubnet:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref SdAppVPC
    CidrBlock: 172.16.1.0/24
    MapPublicIpOnLaunch: false
    AvailabilityZone: !Select
      - 0
      - !GetAZs ''
InternetGateway:
  Type: AWS::EC2::InternetGateway
AttachGateway:
  Type: AWS::EC2::VPCGatewayAttachment
  Properties:
    VpcId: !Ref SdAppVPC
    InternetGatewayId: !Ref InternetGateway
PublicRouteTable:
```

```
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref SdAppVPC
PublicRoute:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PublicRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    GatewayId: !Ref InternetGateway
PublicSubnetRouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    SubnetId: !Ref PublicSubnet
    RouteTableId: !Ref PublicRouteTable
NATGateway:
  Type: AWS::EC2::NatGateway
  Properties:
    AllocationId: !GetAtt NATEIP.AllocationId
    SubnetId: !Ref PublicSubnet
NATEIP:
  Type: AWS::EC2::EIP
  Properties:
    Domain: vpc
PrivateRouteTable:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref SdAppVPC
PrivateRoute:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PrivateRouteTable
```

```
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId: !Ref NATGateway
PrivateSubnetRouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    SubnetId: !Ref PrivateSubnet
    RouteTableId: !Ref PrivateRouteTable
S3AccessRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
       - Effect: Allow
         Principal:
           Service:
             - ec2.amazonaws.com
             - sagemaker.amazonaws.com
         Action: sts:AssumeRole
    Path: /
    Policies:
     - PolicyName: MyS3AccessPolicy
       PolicyDocument:
         Version: '2012-10-17'
         Statement:
          - Effect: Allow
            Action:
              - s3:GetObject
              - s3:PutObject
              - s3:ListBucket
            Resource:
```

```
            - !Sub arn:aws:s3:::sd-app-s3
            - !Sub arn:aws:s3:::sd-app-s3/*
SageMakerRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Principal:
            Service:
              - sagemaker.amazonaws.com
          Action: sts:AssumeRole
    ManagedPolicyArns:
      - arn:aws:iam::aws:policy/AmazonSageMakerFullAccess
    Path: /
SageMakerSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Allow SageMaker access
    VpcId: !Ref SdAppVPC
    SecurityGroupIngress:
      - IpProtocol: '-1'
        CidrIp: 172.16.0.0/16
    SecurityGroupEgress:
      - IpProtocol: '-1'
        CidrIp: 0.0.0.0/0
BackendSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Security Group for Backend Server
```

```
    VpcId: !Ref SdAppVPC
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 443
        ToPort: 443
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 443
        ToPort: 443
        CidrIp: 0.0.0.0/0
    SecurityGroupEgress:
      - IpProtocol: '-1'
        CidrIp: 0.0.0.0/0
MachineLearningSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Security Group for ML Server
    VpcId: !Ref SdAppVPC
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 8188
        ToPort: 8189
        CidrIp: 172.16.0.0/24
```

```
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
        CidrIp: 172.16.0.0/24
      - IpProtocol: tcp
        FromPort: 443
        ToPort: 443
        CidrIp: 172.16.0.0/24
      - IpProtocol: tcp
        FromPort: 443
        ToPort: 443
        CidrIp: 172.16.0.0/24
    SecurityGroupEgress:
      - IpProtocol: '-1'
        CidrIp: 0.0.0.0/0
BackendEIP:
  Type: AWS::EC2::EIP
  Properties:
    Domain: vpc
BackendEIPAssociation:
  Type: AWS::EC2::EIPAssociation
  Properties:
    AllocationId: !GetAtt BackendEIP.AllocationId
    InstanceId: !Ref BackendServer
FineTuneNotebookInstance:
  Type: AWS::SageMaker::NotebookInstance
  Properties:
    InstanceType: ml.g5.xlarge
    RoleArn: !GetAtt SageMakerRole.Arn
    VolumeSizeInGB: 100
    SubnetId: !Ref PrivateSubnet
```

```
    SecurityGroupIds:
      - !Ref SageMakerSecurityGroup
BackendServer:
  Type: AWS::EC2::Instance
  Properties:
    InstanceType: t2.medium
    SubnetId: !Ref PublicSubnet
    SecurityGroupIds:
      - !Ref BackendSecurityGroup
    ImageId: ami-0c80e2b6ccb9ad6d1
    KeyName: sd-app-key-pair
MLServerProfile:
  Type: AWS::IAM::InstanceProfile
  Properties:
    Roles:
      - !Ref S3AccessRole
MLServer:
  Type: AWS::EC2::Instance
  Properties:
    InstanceType: g5.xlarge
    SubnetId: !Ref PrivateSubnet
    SecurityGroupIds:
      - !Ref MachineLearningSecurityGroup
    ImageId: ami-0c9465cbafe2032b2
    KeyName: sd-app-key-pair
    IamInstanceProfile: !Ref MLServerProfile
InstanceConnectEndpoint:
  Type: "AWS::EC2::InstanceConnectEndpoint"
  Properties:
    SecurityGroupIds:
      - !Ref MachineLearningSecurityGroup
```

```
    SubnetId: !Ref PrivateSubnet
  InstanceConnectIngressRule:
    Type: "AWS::EC2::SecurityGroupIngress"
    Properties:
      GroupId: !Ref MachineLearningSecurityGroup
      IpProtocol: "tcp"
      FromPort: 22
      ToPort: 22
      SourceSecurityGroupId: !Ref MachineLearningSecurityGroup
  AmplifyServiceRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: AmplifyServiceRole
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: "Allow"
            Principal:
              Service: "amplify.amazonaws.com"
            Action: "sts:AssumeRole"
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/AdministratorAccess-Amplify
        - arn:aws:iam::aws:policy/AdministratorAccess
      Path: "/"
  AmplifyFrontEndApp:
    Type: AWS::Amplify::App
    Properties:
      Name: SdFrontEndApp
      Platform: WEB
      Repository: "https://github.com/hiep20012003/generate-image-app"
```

```
    AccessToken:
"github_pat_11A46NIDQ0qx4rw2MDkO3d_cgCaM8szQuuN5Rshuhe8tYUIZ1ABkgUSIWheP
57gNEP76Y3K6KNZbH8R0Rp"
    IAMServiceRole: !GetAtt AmplifyServiceRole.Arn
    CustomRules:
      - Source: "</^[^.]+$|\\.(?!(css|gif|ico|jpg|js|png|txt|svg|woff|ttf)$)([^.]+$)/>"
        Target: "/index.html"
        Status: "200"
    EnvironmentVariables:
      - Name: REACT_APP_API_URL
        Value: !Sub "https://${HttpApi}.execute-api.${AWS::Region}.amazonaws.com/prod"
    BuildSpec: |
      version: 1
      applications:
        - appRoot: client
          frontend:
            phases:
              preBuild:
                commands:
                  - npm install
              build:
                commands:
                  - npm run build
              postBuild:
                commands:
                  - echo "Deployment completed"
            artifacts:
              baseDirectory: build
              files:
                - '**/*'
            cache:
```

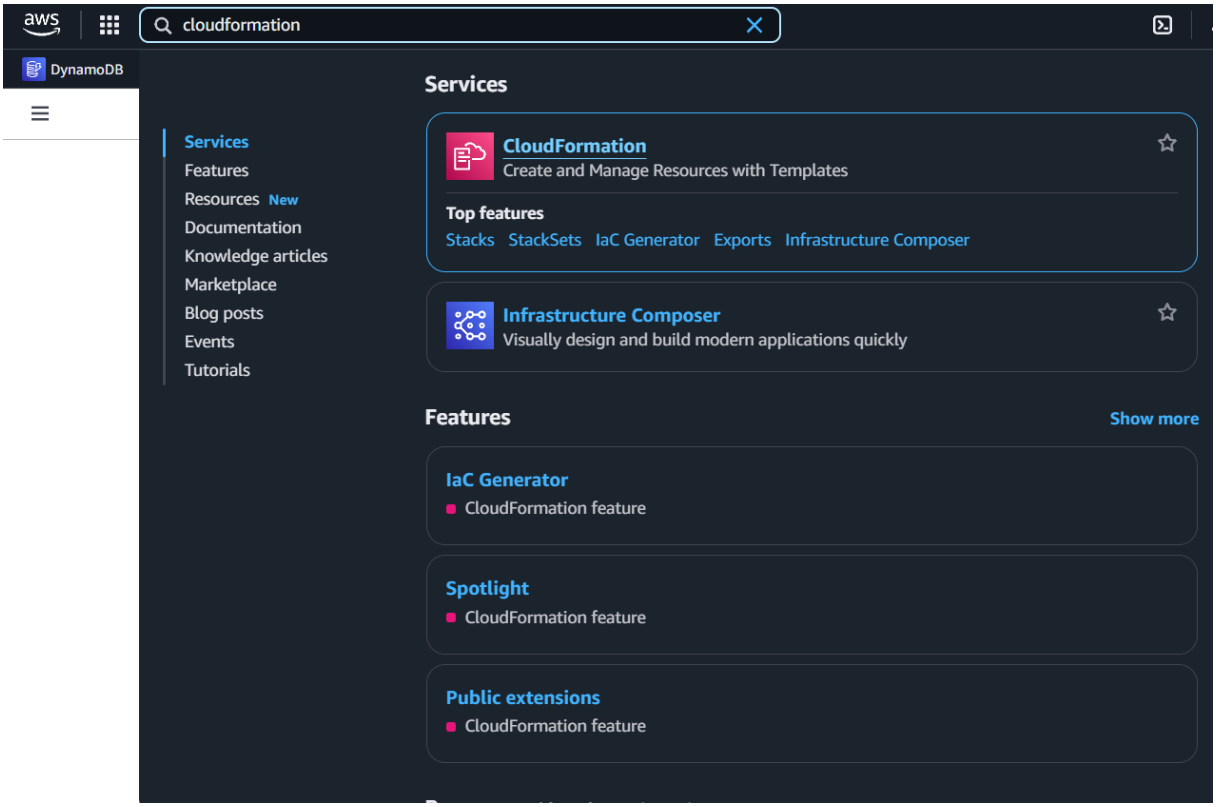
```
        paths:
          - node_modules/**/*
AmplifyBranch:
  Type: AWS::Amplify::Branch
  Properties:
    AppId: !GetAtt AmplifyFrontEndApp.AppId
    BranchName: "main"
    Stage: "PRODUCTION"
    EnableAutoBuild: true
HttpApi:
  Type: AWS::ApiGatewayV2::Api
  Properties:
    Name: "SdAppApi"
    ProtocolType: "HTTP"
    CorsConfiguration:
      AllowOrigins:
        - "*"
      AllowMethods:
        - "GET"
        - "POST"
      AllowHeaders:
        - "*"
    Tags:
      Environment: "prod"
TextIntegration:
  Type: AWS::ApiGatewayV2::Integration
  Properties:
    ApiId: !Ref HttpApi
    IntegrationType: "HTTP_PROXY"
    IntegrationMethod: "ANY"
    IntegrationUri: !Sub "http://${BackendEIP.PublicIp}:80/comfy/generate/text"
```

```
      PayloadFormatVersion: "1.0"
      TimeoutInMillis: 29000
  SketchIntegration:
    Type: AWS::ApiGatewayV2::Integration
    Properties:
      ApiId: !Ref HttpApi
      IntegrationType: "HTTP_PROXY"
      IntegrationMethod: "ANY"
      IntegrationUri: !Sub "http://${BackendEIP.PublicIp}:80/comfy/generate/sketch"
      PayloadFormatVersion: "1.0"
      TimeoutInMillis: 29000
  ComfyGenerateTextRoute:
    Type: AWS::ApiGatewayV2::Route
    Properties:
      ApiId: !Ref HttpApi
      RouteKey: "POST /comfy/generate/text"
      Target: !Sub "integrations/${TextIntegration}"
  ComfyGenerateSketchRoute:
    Type: AWS::ApiGatewayV2::Route
    Properties:
      ApiId: !Ref HttpApi
      RouteKey: "POST /comfy/generate/sketch"
      Target: !Sub "integrations/${SketchIntegration}"
  HttpApiStage:
    Type: AWS::ApiGatewayV2::Stage
    Properties:
      ApiId: !Ref HttpApi
      StageName: "prod"
      AutoDeploy: true
Outputs:
  AmplifyAppId:
```

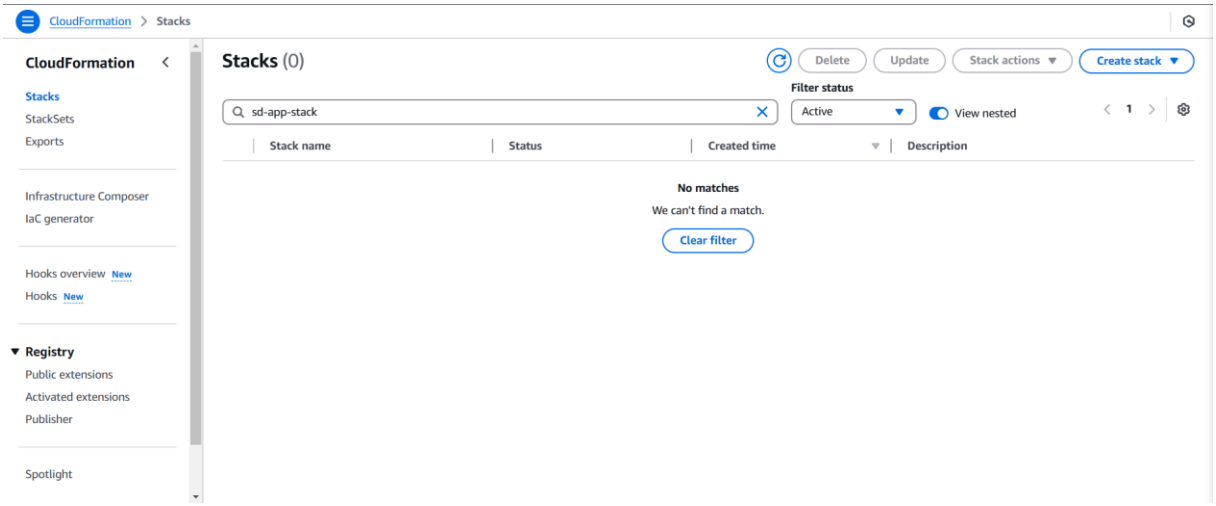
```
  Description: Amplify Frontend App ID
  Value: !GetAtt AmplifyFrontEndApp.AppId
AmplifyAppURL:
  Description: URL for the Amplify Frontend
  Value: !Sub "https://${AmplifyFrontEndApp}.amplifyapp.com"
```

3.1.3.2. Các bước triển khai với AWS CloudFormation

- Bước 1: Truy cập vào AWS Console Home và Đăng nhập vào tài khoản AWS: *https://console.aws.amazon.com/?nc2=h_m_mc*
- Bước 2: Truy cập vào dịch vụ AWS CloudFormation

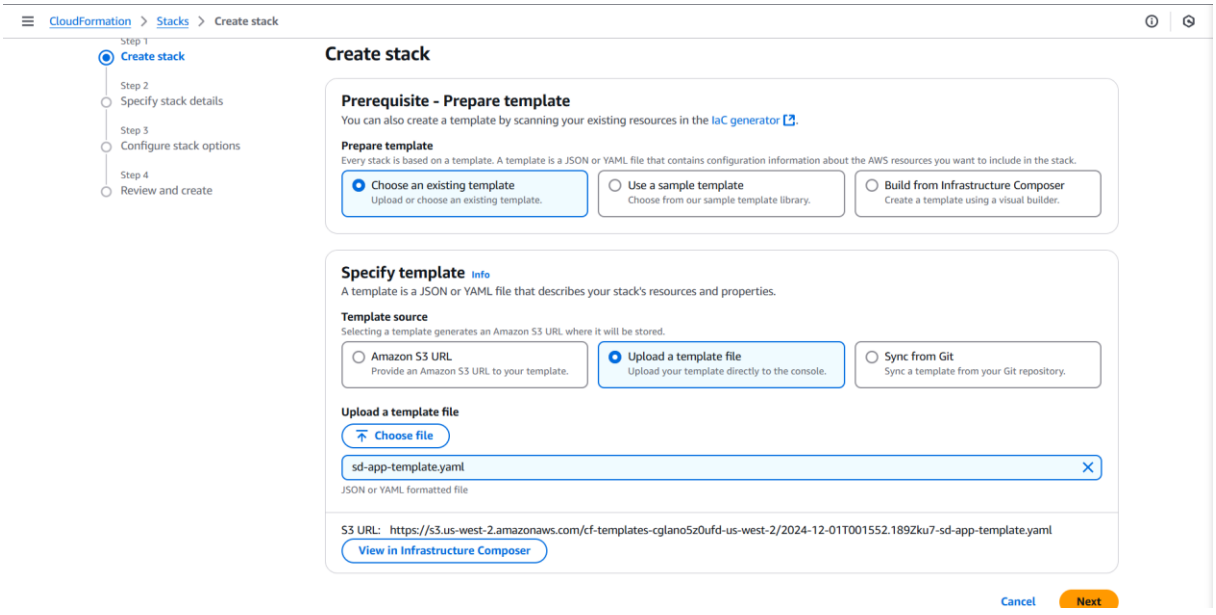


Hình 3.8: Tìm kiếm dịch vụ CloudFormation



Hình 3.9: Giao diện dịch vụ CloudFormation

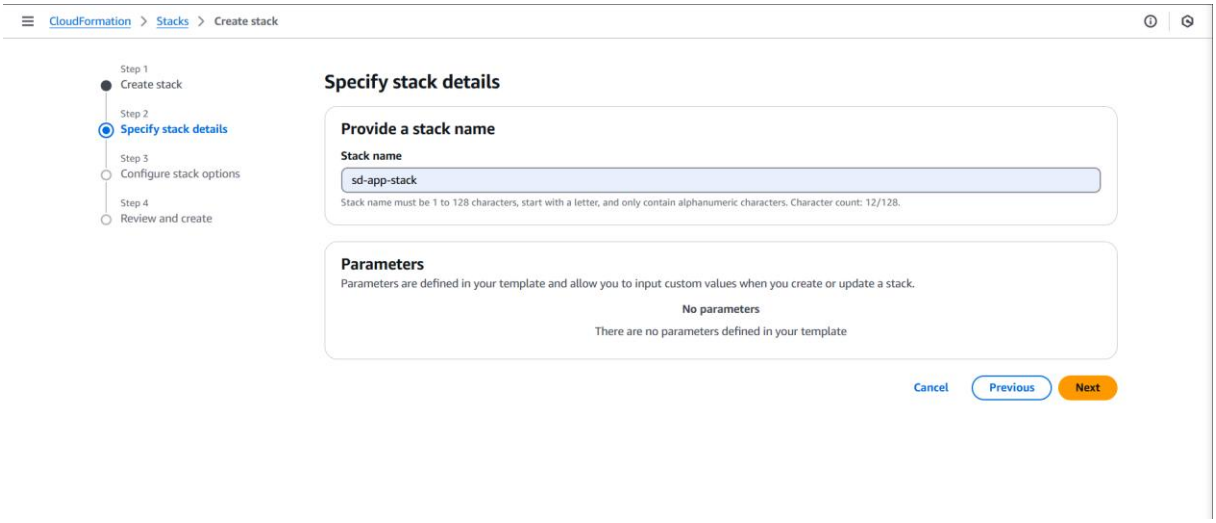
- Bước 3: Nhấn vào Create stack → With new resources (standard)
- Bước 4: Step 1 – Create Stack
  - Prepare template: chọn *Choose an existing template*
  - Specify template – template source: chọn *Upload a template file*
  - Sau đó upload file template đã tạo: sd-app-template.yaml
  - Nhấn vào Next



Hình 3.10: CloudFormation Step 1 – Create stack

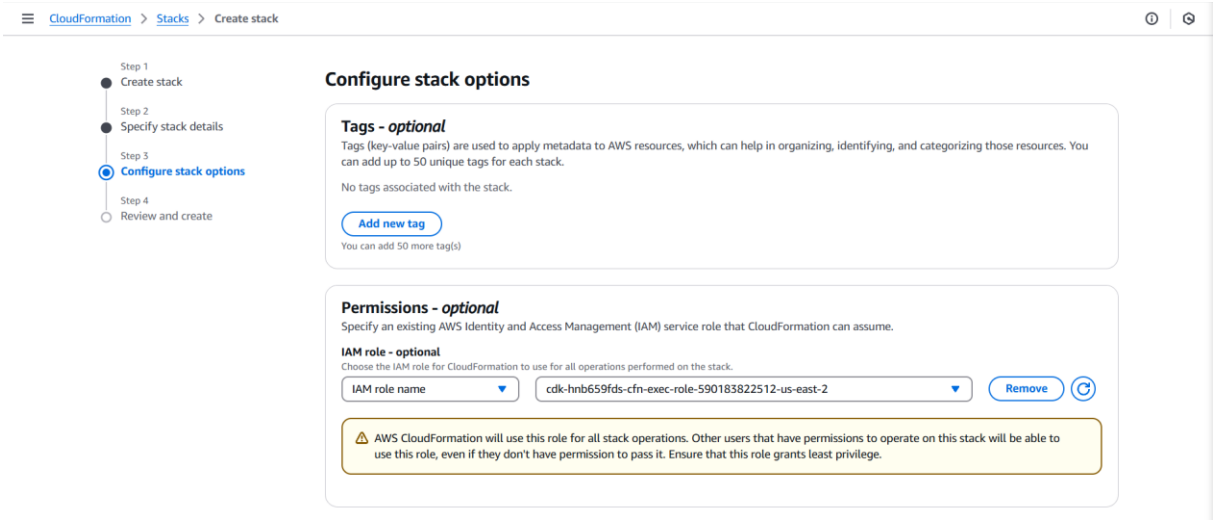
- Bước 5: Step 2 - Specify stack details
  - Stack name : sd-app-stack

- Nhấn vào Next



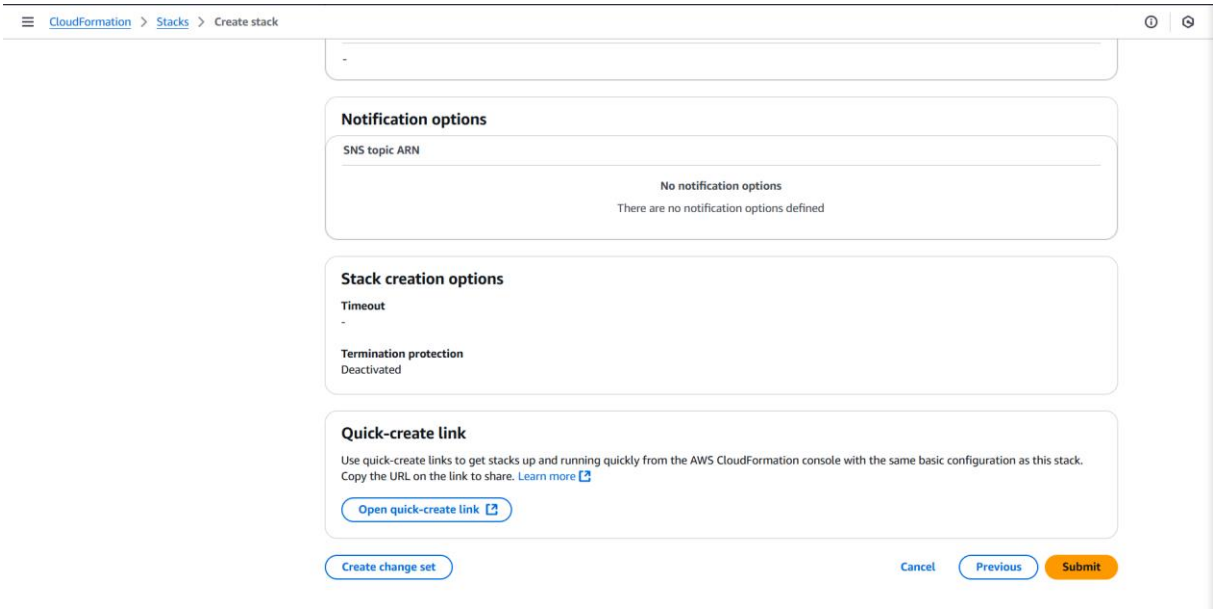
Hình 3.11: CloudFormation Step 2 – Specify stack details

- Bước 6: Step 3 - Configure stack options
  - Permissions - IAM role: chọn Role có quyền tạo CloudFormation stack và các tài nguyên khác.
  - Nhấn vào Next

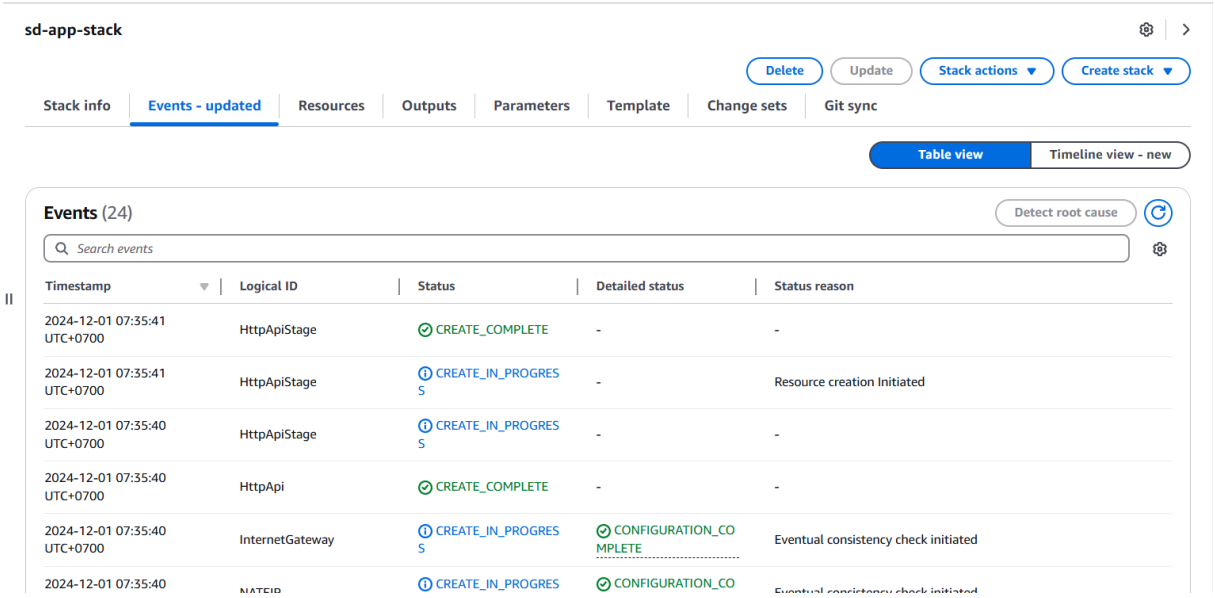


Hình 3.12: CloudFormation Step 3 – Configure stack options

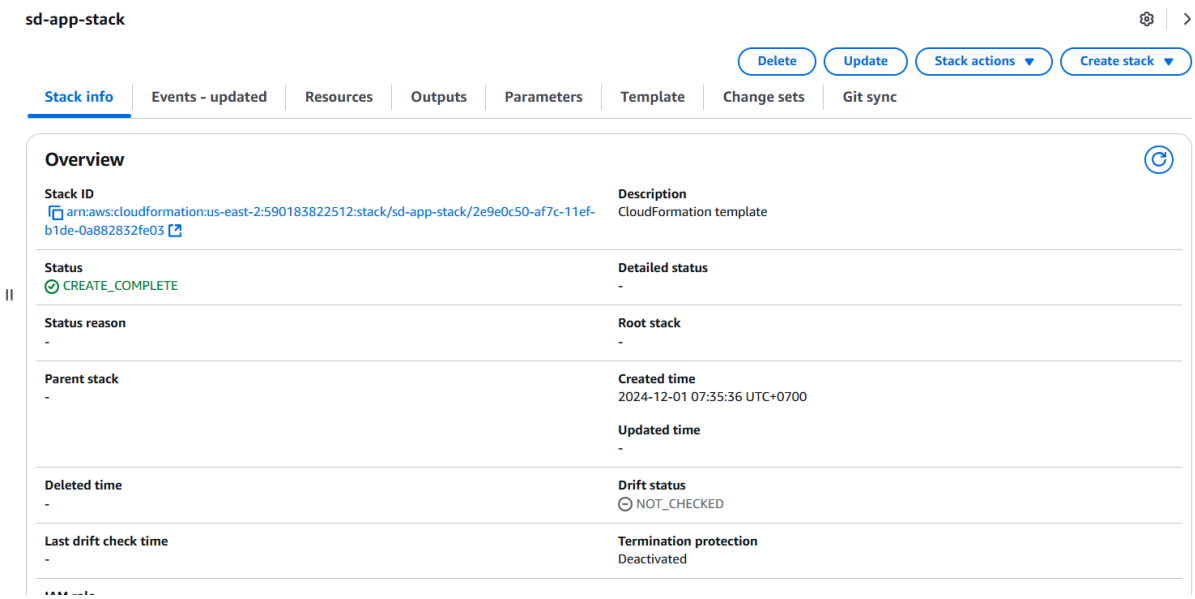
- Bước 7: Step 4 - Review and create
  - Xem lại cấu hình stack
  - Nhấn vào Submit



Hình 3.13: CloudFormation Step 4 – Review and create



Hình 3.14: CloudFormation đang tạo các tài nguyên AWS



Hình 3.15: CloudFormation *tạo stack thành công*

3.2. TINH CHỈNH MÔ HÌNH AI STABLE DIFFUSION VỚI LORA

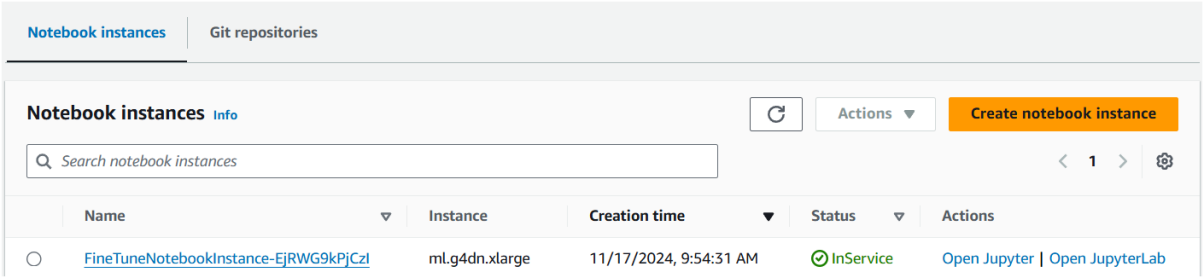
3.2.1. Tổng quan về tập dữ liệu nội thất

3.2.2. Tinh chỉnh mô hình với Kohya trên SageMaker Notebook

Bước 1: Truy cập dịch vụ AWS SageMaker → Notebooks

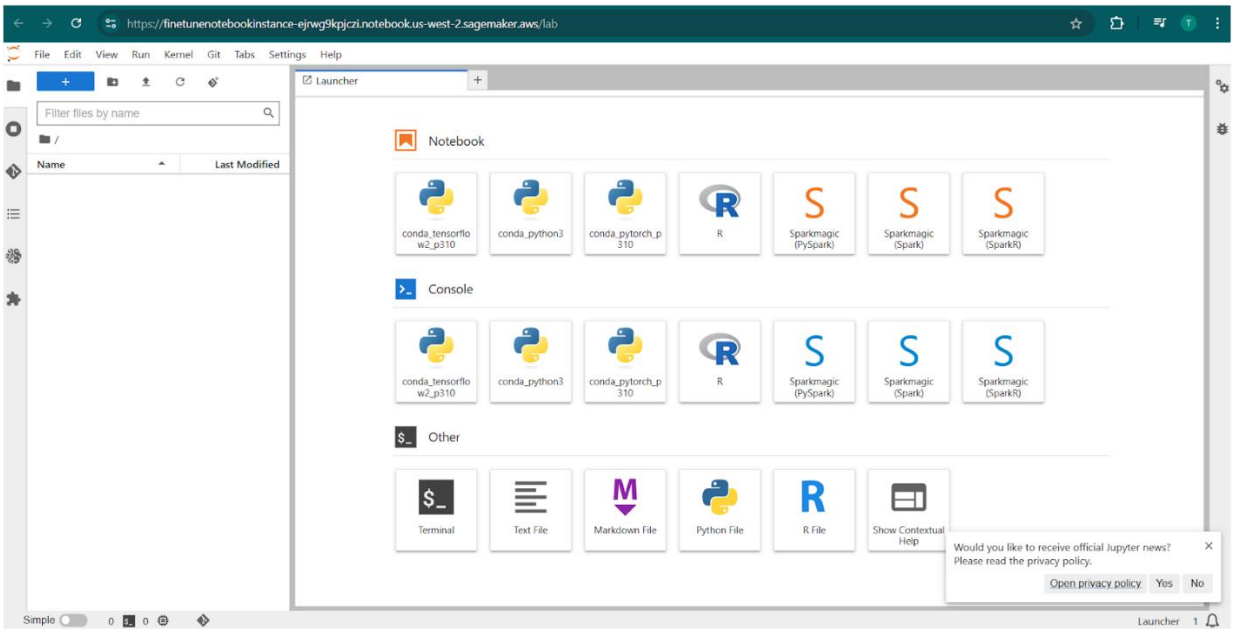
- Nhấn vào FineTuneNotebookInstance đã tạo với CloudFormation

Bước 2: Click vào open JupyterLab



Hình 3.16: Giao diện quản lý Notebook Instance trên AWS Sagemaker

Ta sẽ thấy giao diện như thế này. Sau đó mở terminal



Hình 3.17: Giao diện JupyterLab trong AWS SageMaker Notebook Instance

Bước 3: Triển khai Kohya để thực hiện training model

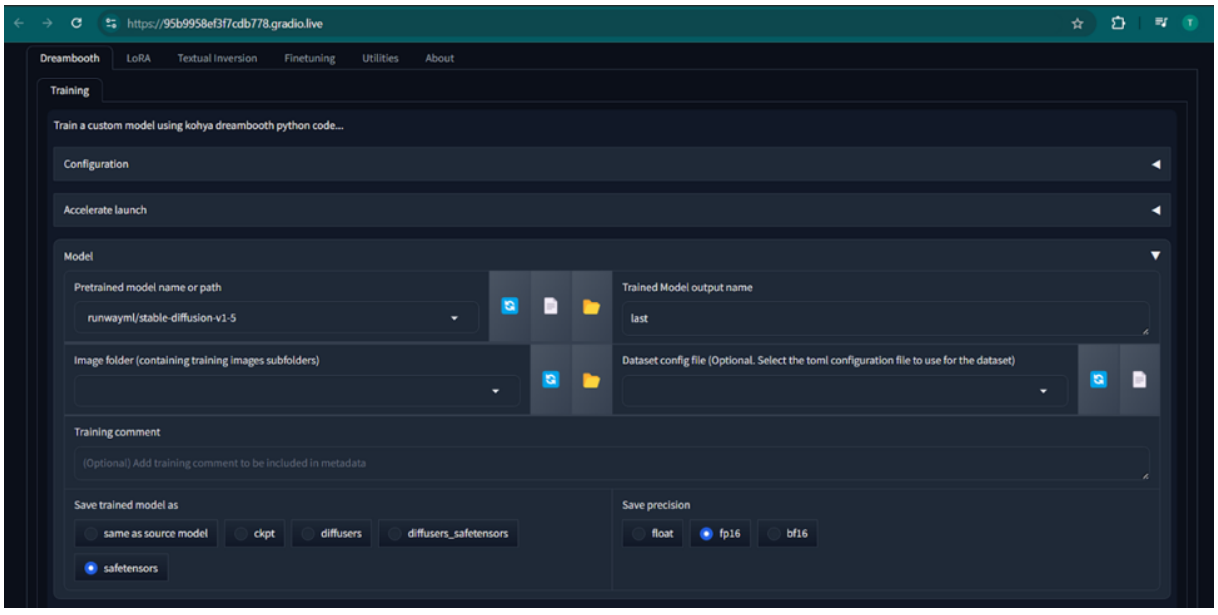
```
# Install a separate conda installation via Miniconda
WORKING_DIR=/home/ec2-user/SageMaker
mkdir -p "$WORKING_DIR"
wget https://repo.anaconda.com/miniconda/Miniconda3-py310_23.5.2-0-Linux-x86_64.sh -O
"$WORKING_DIR/miniconda.sh"
bash "$WORKING_DIR/miniconda.sh" -b -u -p "$WORKING_DIR/miniconda"
rm -rf "$WORKING_DIR/miniconda.sh"

# Create a custom conda environment
source "$WORKING_DIR/miniconda/bin/activate"
KERNEL_NAME="kohya"
PYTHON="3.10.12"  # Updated to Python 3.10.12
conda create --yes --name "$KERNEL_NAME" python="$PYTHON"
conda activate "$KERNEL_NAME"
pip install --quiet ipykernel
```

```
# Clone and set up kohya_ss project
cd SageMaker
git clone https://github.com/bmaltais/kohya_ss
cd kohya_ss
python3 -m venv venv
source venv/bin/activate
./setup.sh -n
source /home/ec2-user/SageMaker/miniconda/bin/activate
conda activate kohya
bash /home/ec2-user/SageMaker/kohya_ss/gui.sh --share
```

Bước 4: Khi chạy code triển khai trên sẽ xuất ra được link public của Kohya

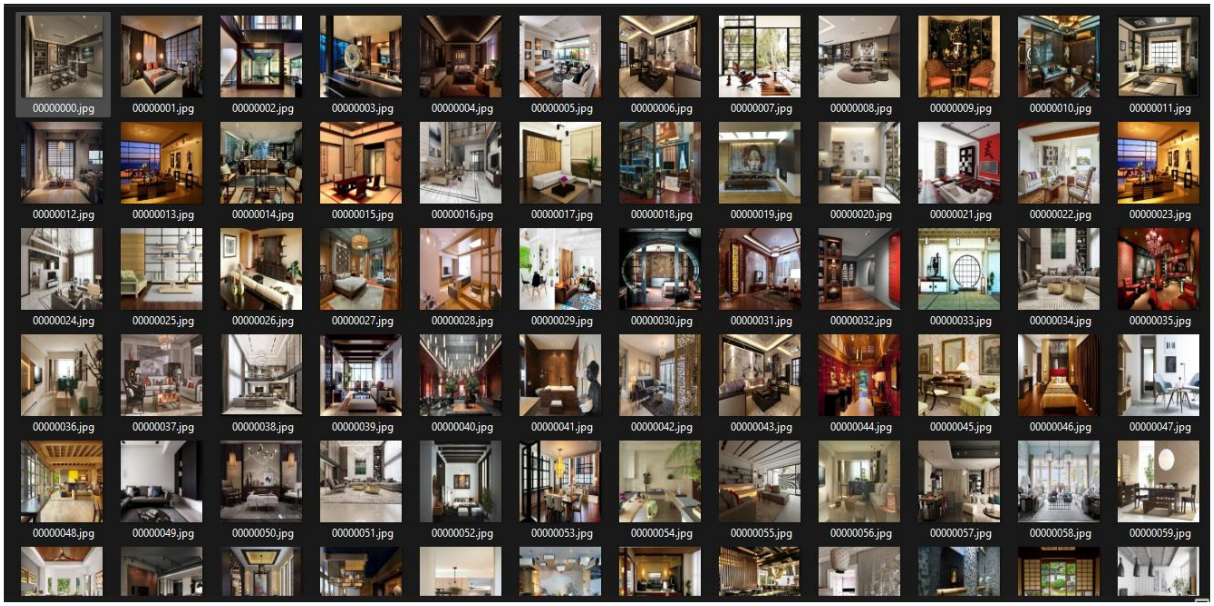
```
Running on local URL:  http://127.0.0.1:7860
Running on public URL: https://95b9958ef3f7cdb778.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from Terminal to deploy to Spaces (https://huggingface.co/spaces)
```

Hình 3.18: Link truy cập vào Kohya



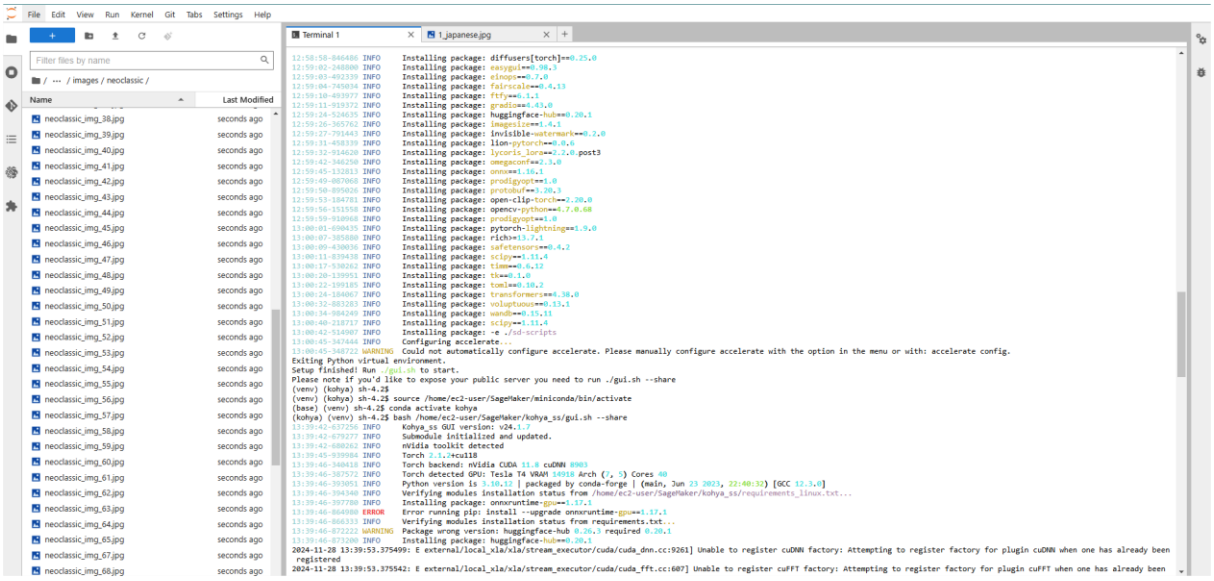
Hình 3.19: Giao diện Kohya

Bước 5: Chuẩn bị dữ liệu hình ảnh (size ảnh 512x512) để nạp vào Jupyter



Hình 3.20: Tệp dữ liệu hình ảnh

- Trong phần File Explore vào /kohyass/dataset/images để nạp hình ảnh vào

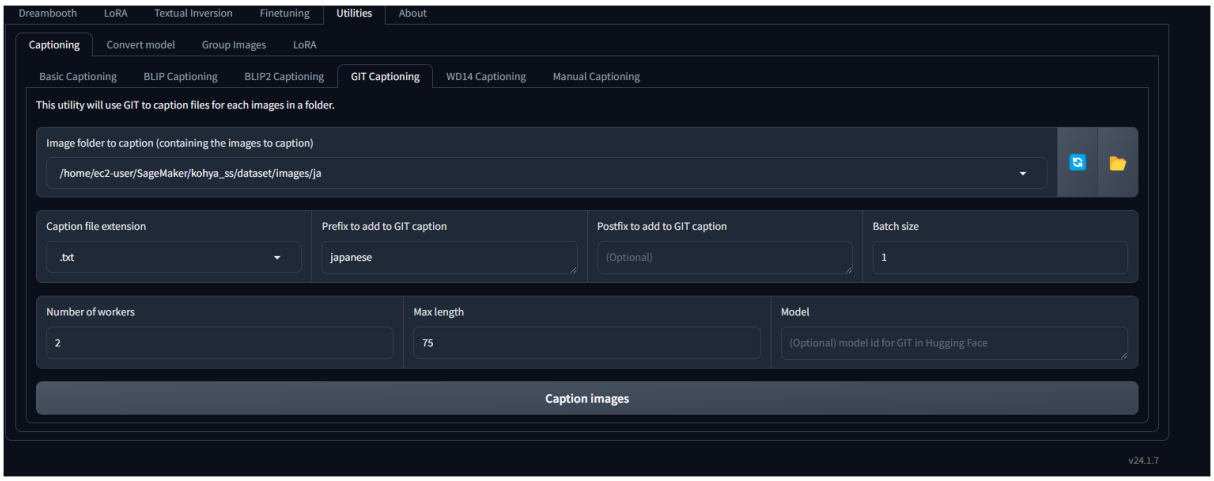


Hình 3.21: Nạp hình ảnh vào kohya để huấn luyện

- Bắt đầu caption data: vào lại giao diện Kohya

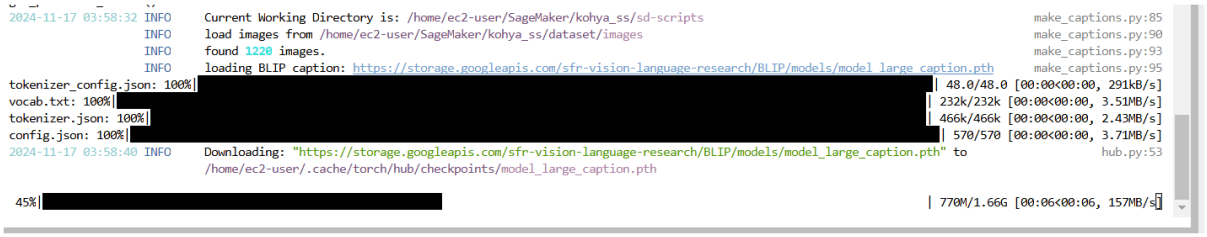
  Chọn mục Ultilities -> BLIP Catptioning. Sau đó chọn Folder lúc nãy đã lưu hình ảnh /kohya_ss/dataset/images.

- Click vào Caption image. Hệ thống sẽ tự động gắn nhãn cho các hình ảnh đã truyền vào trong thư mục dataset/images



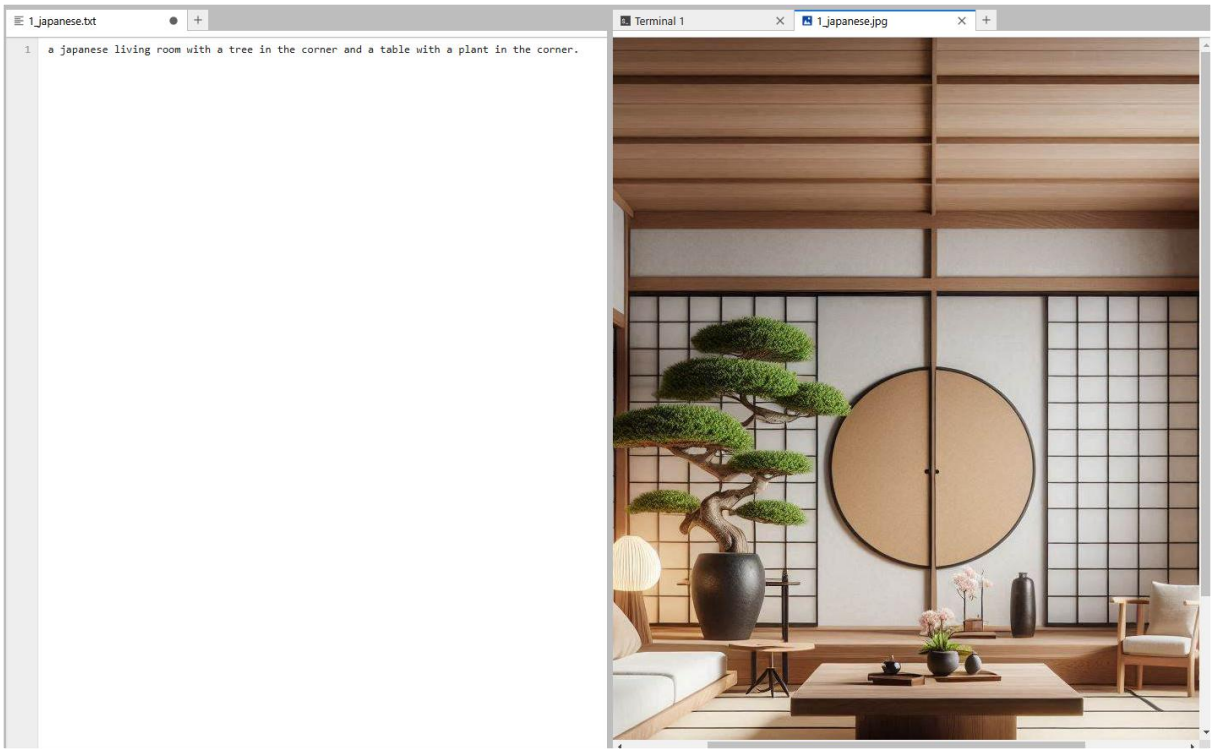
Hình 3.22: Giao diện gán nhãn hình ảnh bằng BLIP

- Hệ thống đang caption hình ảnh



Hình 3.23: Quá trình tải và xử lý mô hình BLIP để gán nhãn hình ảnh

- Ví dụ về hình ảnh đã Caption

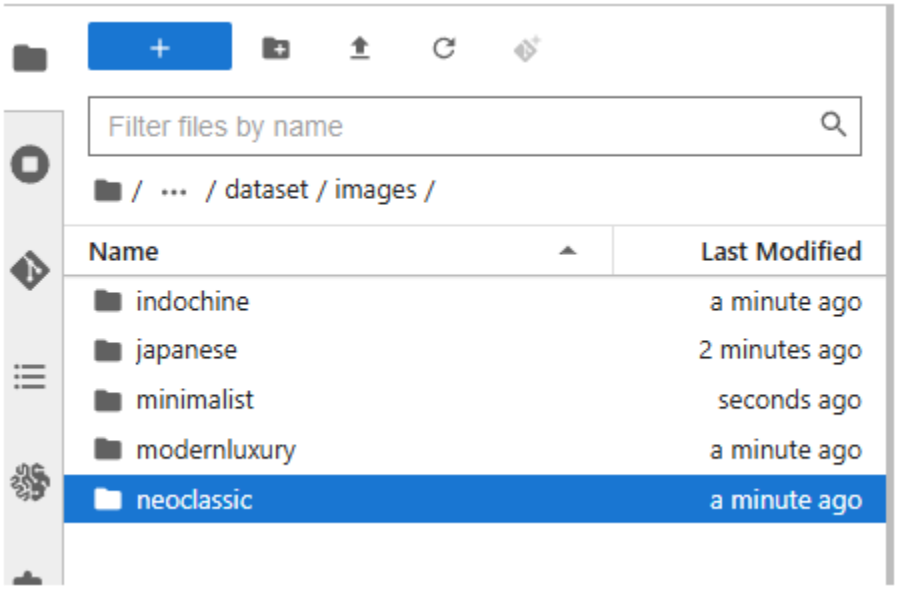


Hình 3.24: Kết quả hình ảnh sau khi đã caption

Bước 6: Training Lora

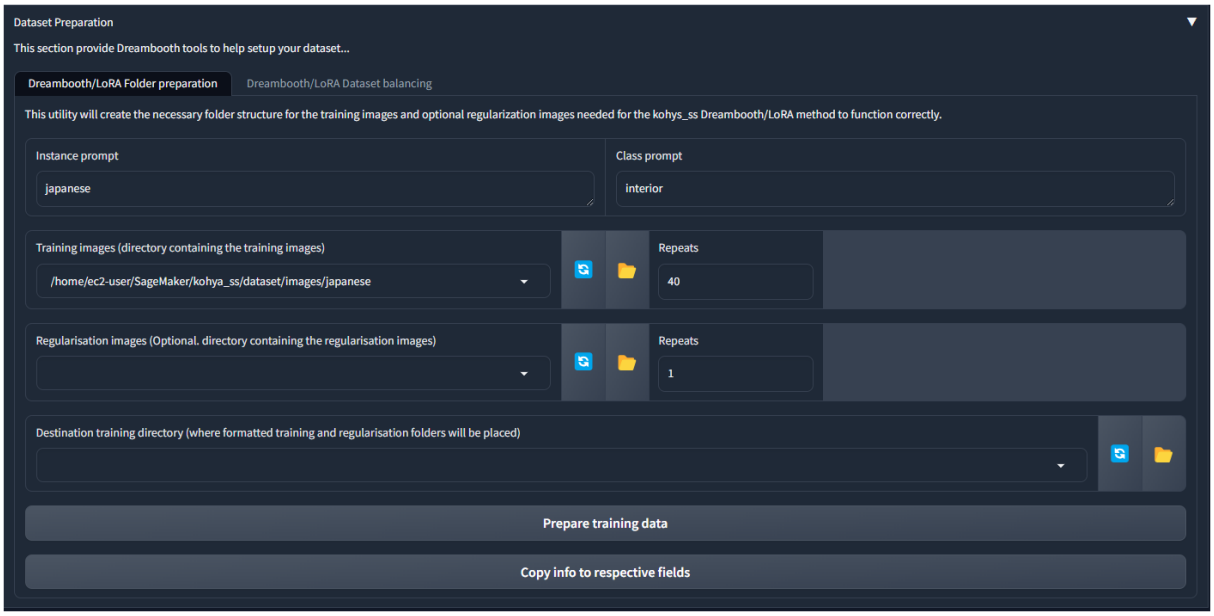
- Chia các loại hình ảnh đã caption thành từng thư mục con:

```
mkdir -p /home/ec2-user/SageMaker/kohya_ss/dataset/images/indochine
cp /home/ec2-user/SageMaker/kohya_ss/dataset/images/indochine* /home/ec2-user/SageMaker/kohya_ss/dataset/images/indochine/
mkdir -p /home/ec2-user/SageMaker/kohya_ss/dataset/images/japanese
cp /home/ec2-user/SageMaker/kohya_ss/dataset/images/japanese* /home/ec2-user/SageMaker/kohya_ss/dataset/images/japanese/
mkdir -p /home/ec2-user/SageMaker/kohya_ss/dataset/images/minimalist
cp /home/ec2-user/SageMaker/kohya_ss/dataset/images/minimalist* /home/ec2-user/SageMaker/kohya_ss/dataset/images/minimalist/
mkdir -p /home/ec2-user/SageMaker/kohya_ss/dataset/images/modernluxury
cp /home/ec2-user/SageMaker/kohya_ss/dataset/images/modernluxury* /home/ec2-user/SageMaker/kohya_ss/dataset/images/modernluxury/
mkdir -p /home/ec2-user/SageMaker/kohya_ss/dataset/images/neoclassic
cp /home/ec2-user/SageMaker/kohya_ss/dataset/images/neoclassic* /home/ec2-user/SageMaker/kohya_ss/dataset/images/neoclassic/
```



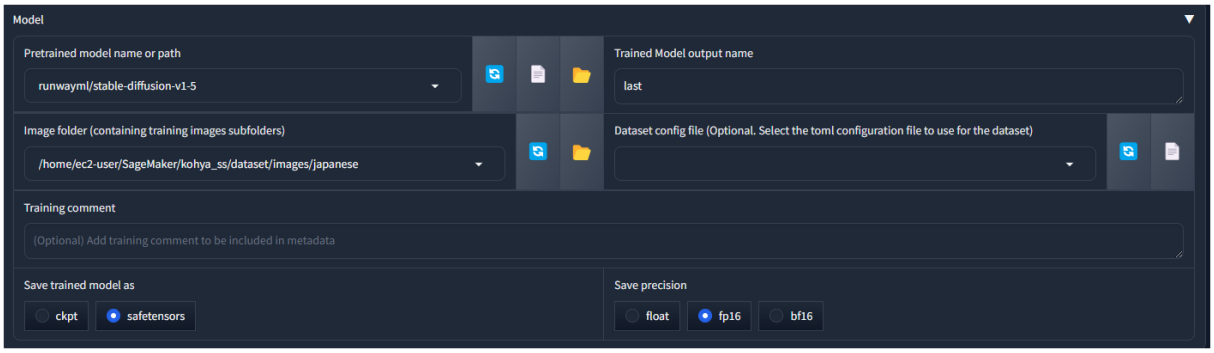
Hình 3.25: Chia các hình ảnh đã caption thành từng mục con

- Prepare data từ thư mục hình ảnh con đã tách ở bước trên



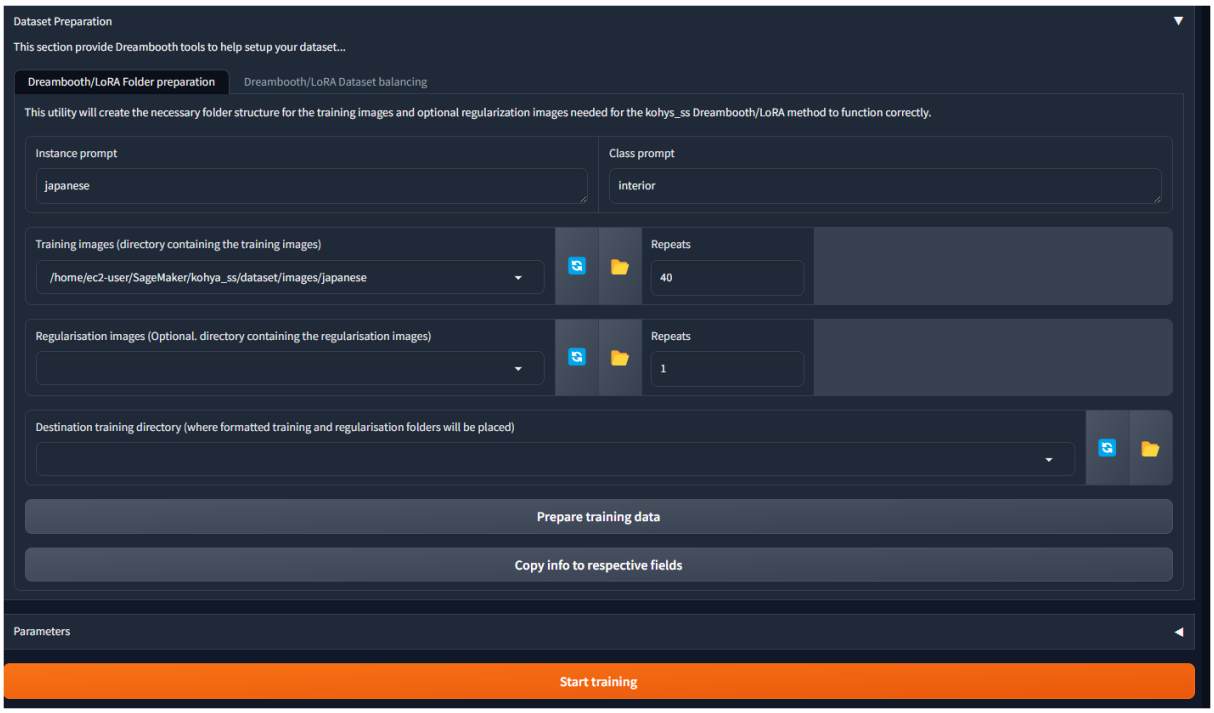
Hình 3.26: Chuẩn bị dữ liệu với Dreambooth/LoRA

- Chọn định dạng model (Dùng Base model sd-1.5)



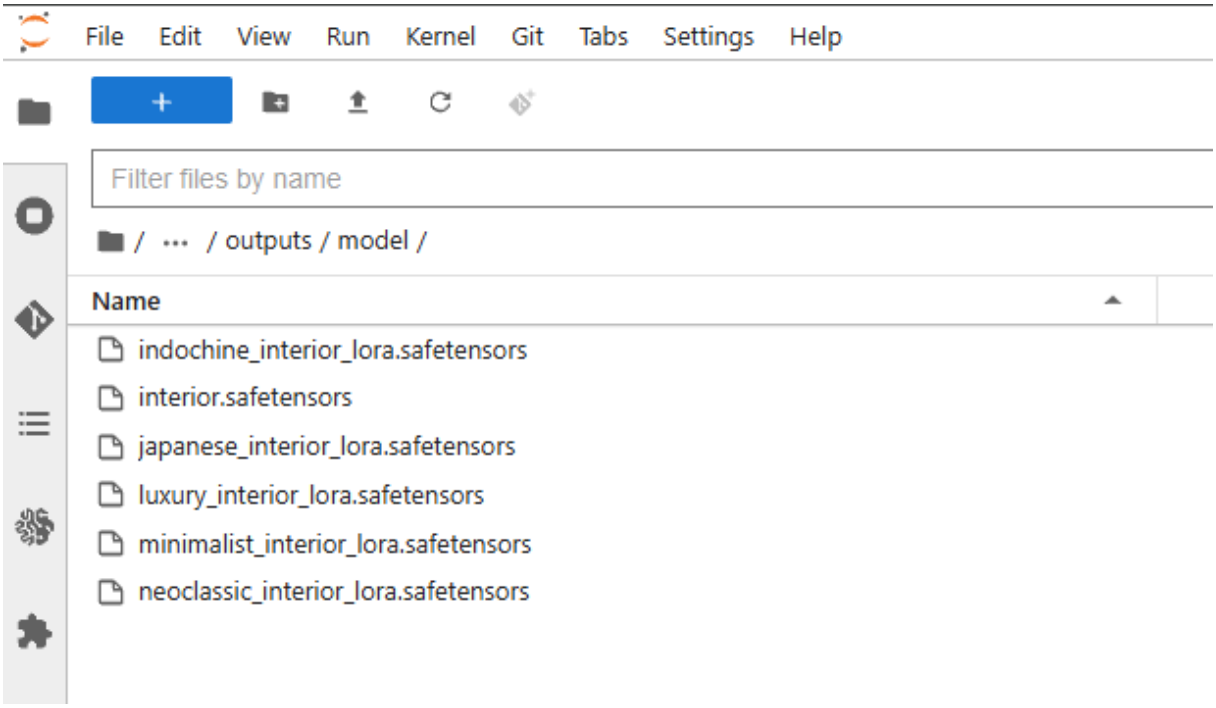
Hình 3.27: Chọn định dạng model

- Sau đó nhấn Start Training



Hình 3.28: Bắt đầu huấn luyện mô hình

- Sau khi xong, các model interior lora sẽ được lưu ở /home/ec2-user/SageMaker/kohya_ss/outputs/model/



Hình 3.29: Kết quả sau khi huấn luyện mô hình

- Người dùng có thể upload lên Amazon S3 để triển khai trên máy chủ Machine Learning (Comfy UI)
    - Dùng lệnh: *aws s3 sync ./ s3://sd-app-s3/loras/*

```
(kohya) (venv) sh-4.2$ aws s3 sync ./ s3://sd-app-s3/loras/


upload: ./interior.safetensors to s3://sd-app-s3/loras/interior.safetensors
upload: ./neoclassic_interior_lora.safetensors to s3://sd-app-s3/loras/neoclassic_interior_lora.safetensors
upload: ./luxury_interior_lora.safetensors to s3://sd-app-s3/loras/luxury_interior_lora.safetensors
upload: ./minimalist_interior_lora.safetensors to s3://sd-app-s3/loras/minimalist_interior_lora.safetensors
upload: ./japanese_interior_lora.safetensors to s3://sd-app-s3/loras/japanese_interior_lora.safetensors
upload: ./indochine_interior_lora.safetensors to s3://sd-app-s3/loras/indochine_interior_lora.safetensors
```

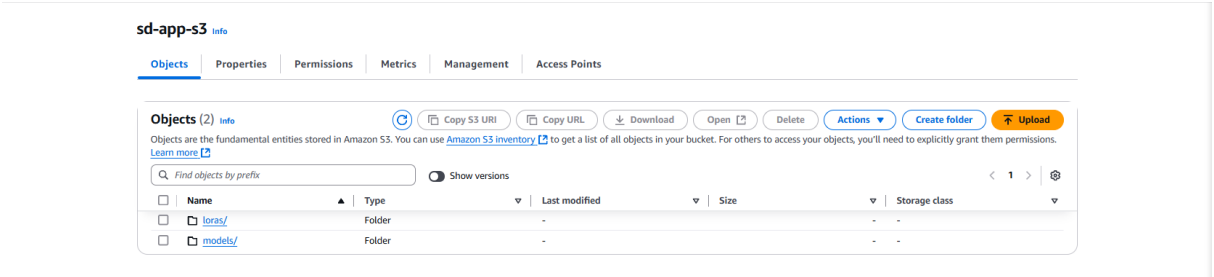
Hình 3.30: Upload model lên Amazon S3

3.3. TRIỂN KHAI MÁY CHỦ MACHINE LEARNING (COMFY UI)

3.3.1. Quản lý và chuẩn bị tài nguyên mô hình trên Amazon S3

Tài nguyên mô hình được lưu trữ trên Amazon S3 trong hai thư mục chính:

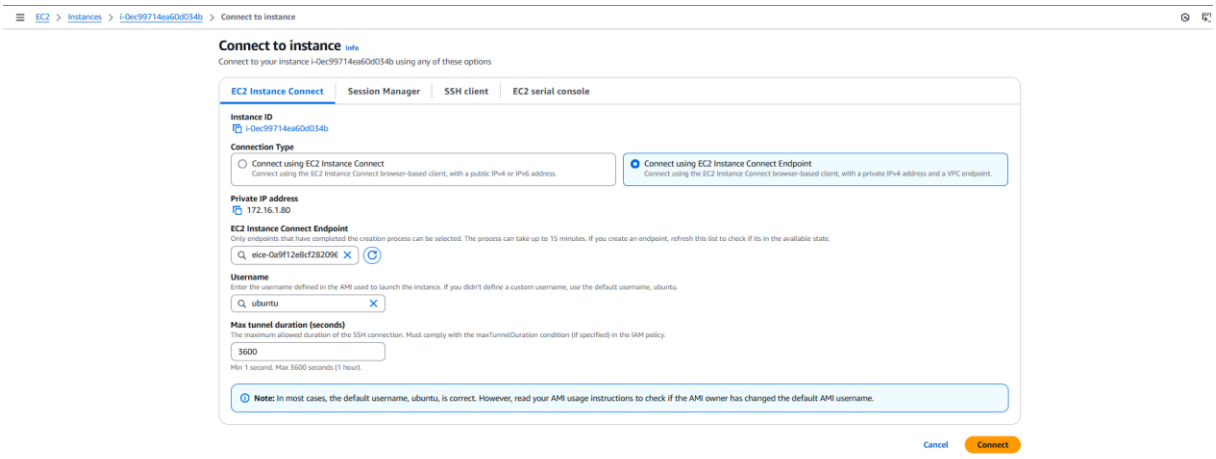
- Thư mục models/:
    - Chứa các tệp mô hình chính được sử dụng bởi ComfyUI để sinh ảnh.
    - Các mô hình phổ biến như Stable Diffusion (.ckpt hoặc .safetensors) được tổ chức trong thư mục này.
    - Mục đích là cung cấp mô hình gốc cho quá trình sinh ảnh hoặc fine-tune.
- Thư mục lora/:
    - Chứa các tệp LoRA (Low-Rank Adaptation), thường ở định dạng .safetensors, được dùng để bổ sung hoặc tùy chỉnh phong cách sinh ảnh.
    - Các LoRA này được chuẩn bị từ bước huấn luyện.



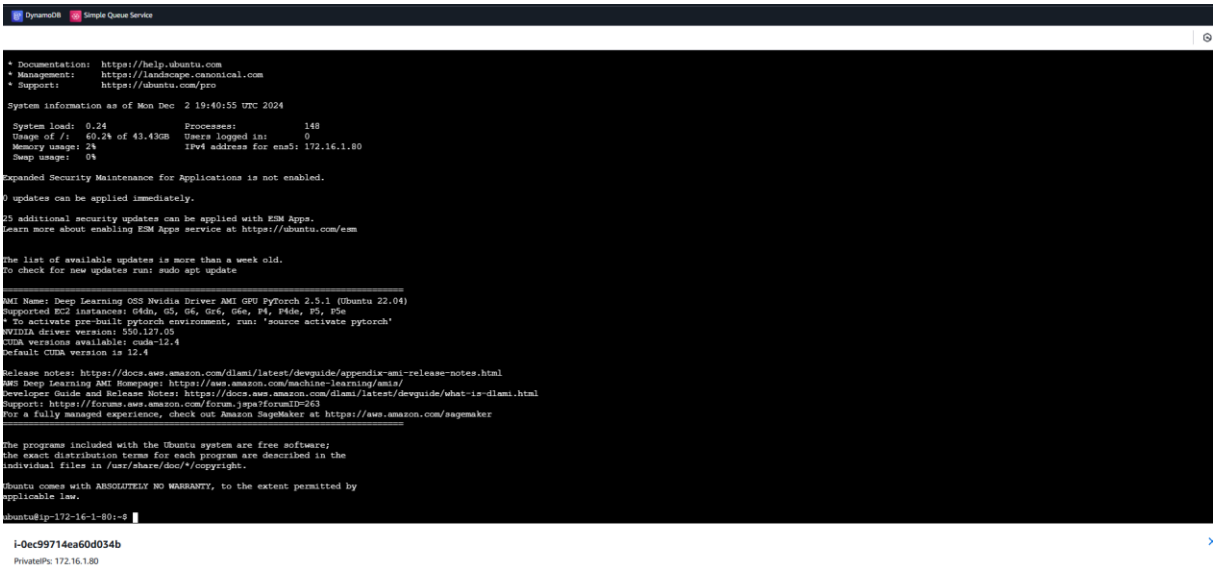
Hình 3.31: Tổ chức thư mục Amazon S3

3.3.2. Triển khai ComfyUI trên Máy chủ Machine Learning

Bước 1: Kết nối với instance EC2 của máy chủ Machine Learning thông qua AWS Management Console.



Hình 3.32: Kết nối máy chủ Machine Learning



Hình 3.33: Kết nối máy chủ Machine Learning thành công

Bước 2: Cài đặt và cấu hình môi trường PyTorch

```
sudo apt update && apt upgrade -y
source activate pytorch
pip install --upgrade pip -y
pip install --upgrade pip setuptools wheel -y
pip uninstall torch torchvision torchaudio -y
```

```
pip install --pre torch torchvision torchaudio --index-url
https://download.pytorch.org/whl/nightly/cu124 -y
```

```
ubuntu@ip-172-16-1-80:~$ sudo apt update && apt upgrade -y

source activate pytorch
pip install --upgrade pip -y
pip install --upgrade pip setuptools wheel -y

pip uninstall torch torchvision torchaudio -y
pip install --pre torch torchvision torchaudio --index-url https://download.pytorch.org/whl/nightly/cu124 -y
```

Hình 3.34: Cài đặt và cấu hình môi trường PyTorch

Bước 3: Cài đặt ComfyUI và các custom nodes

```
git clone https://github.com/comfyanonymous/ComfyUI.git /opt/dlami/nvme/ComfyUI
cd /opt/dlami/nvme/ComfyUI
pip3 install -r requirements.txt
cd /opt/dlami/nvme/ComfyUI/custom_nodes
git clone https://github.com/ltdrdata/ComfyUI-Manager.git
git clone https://github.com/Fannovel16/comfyui_controlnet_aux/
git clone https://github.com/pythongosssss/ComfyUI-Custom-Scripts.git
git clone https://github.com/Suzie1/ComfyUI_Comfyroll_CustomNodes
cd /opt/dlami/nvme/ComfyUI/custom_nodes/ComfyUI-Manager
pip3 install -r requirements.txt
cd /opt/dlami/nvme/ComfyUI/custom_nodes/comfyui_controlnet_aux
pip3 install -r requirements.txt
cd /opt/dlami/nvme/ComfyUI/custom_nodes/ComfyUI-Custom-Scripts
pip3 install -r requirements.txt
cd /opt/dlami/nvme/ComfyUI/custom_nodes/ComfyUI_Comfyroll_CustomNodes
pip3 install -r requirements.txt
```

```
(pytorch) ubuntu@ip-172-16-1-80:~$ git clone https://github.com/comfyanonymous/ComfyUI.git /opt/dlami/nvme/ComfyUI
cd /opt/dlami/nvme/ComfyUI
pip3 install -r requirements.txt

cd /opt/dlami/nvme/ComfyUI/custom_nodes
git clone https://github.com/ltdrdata/ComfyUI-Manager.git
git clone https://github.com/Fannovel16/comfyui_controlnet_aux/
git clone https://github.com/pythongosssss/ComfyUI-Custom-Scripts.git
git clone https://github.com/Suzie1/ComfyUI_Comfyroll_CustomNodes

cd /opt/dlami/nvme/ComfyUI/custom_nodes/ComfyUI-Manager
pip3 install -r requirements.txt

cd /opt/dlami/nvme/ComfyUI/custom_nodes/comfyui_controlnet_aux
pip3 install -r requirements.txt

cd /opt/dlami/nvme/ComfyUI/custom_nodes/ComfyUI-Custom-Scripts
pip3 install -r requirements.txt

cd /opt/dlami/nvme/ComfyUI/custom_nodes/ComfyUI_Comfyroll_CustomNodes
pip3 install -r requirements.txt
```

Hình 3.35: Cài đặt ComfyUI và các custom nodes

Bước 4: Load mô hình chính và mô hình lora từ Amazon S3

```
aws s3 sync s3://sd-app-s3/models/ /opt/dlami/nvme/ComfyUI/models/checkpoints/
aws s3 sync s3://sd-app-s3/loras/ /opt/dlami/nvme/ComfyUI/models/loras/
```

```
(pytorch) ubuntu@ip-172-16-1-80:~$ aws s3 sync s3://sd-app-s3/models/ /opt/dlami/nvme/ComfyUI/models/checkpoints/
aws s3 sync s3://sd-app-s3/loras/ /opt/dlami/nvme/ComfyUI/models/loras/
```

Hình 3.36: Load mô hình chính và mô hình lora từ Amazon S3

Bước 5: Tải ControlNet model và VAE model

```
cd /opt/dlami/nvme/ComfyUI/models/controlnet
wget
https://huggingface.co/lllyasviel/control_v11p_sd15_lineart/resolve/main/diffusion_pytorch_model.fp16.safetensors -O control_v11p_sd15_lineart.safetensors
wget
https://huggingface.co/lllyasviel/control_v11p_sd15_mlsd/resolve/main/diffusion_pytorch_model.fp16.safetensors -O control_v11p_sd15_mlsd.safetensors

cd /opt/dlami/nvme/ComfyUI/models/vae
wget https://huggingface.co/stabilityai/sd-vae-ft-mse-original/resolve/main/vae-ft-mse-840000-ema-pruned.safetensors
```

```
(pytorch) ubuntu@ip-172-16-1-80:~$ cd /opt/dlami/nvme/ComfyUI/models/controlnet
wget https://huggingface.co/lllyasviel/control_v11p_sd15_lineart/resolve/main/diffusion_pytorch_model.fp16.safetensors -O control_v11p_sd15_lineart.safetensors
wget https://huggingface.co/lllyasviel/control_v11p_sd15_mlsd/resolve/main/diffusion_pytorch_model.fp16.safetensors -O control_v11p_sd15_mlsd.safetensors

cd /opt/dlami/nvme/ComfyUI/models/vae
wget https://huggingface.co/stabilityai/sd-vae-ft-mse-original/resolve/main/vae-ft-mse-840000-ema-pruned.safetensors
```

Hình 3.37: Tải ControlNet model và VAE model

Bước 6: Tạo script khởi chạy ComfyUI

```
set +H
# Tạo script khởi chạy ComfyUI cho Text To Image
echo -e "#!/bin/bash\ncd /opt/dlami/nvme/ComfyUI && source activate pytorch && python3 main.py --listen 0.0.0.0 --port 8188" > /home/ubuntu/start_comfyui_text.sh
chmod +x /home/ubuntu/start_comfyui_text.sh

# Tạo script khởi chạy ComfyUI cho Sketch To Image
echo -e "#!/bin/bash\ncd /opt/dlami/nvme/ComfyUI && source activate pytorch && python3 main.py --listen 0.0.0.0 --port 8189" > /home/ubuntu/start_comfyui_sketch.sh
chmod +x /home/ubuntu/start_comfyui_sketch.sh
```

```
(pytorch) ubuntu@ip-172-16-1-80:~$ set +H
# Tạo script khởi chạy ComfyUI cho Text To Image
echo -e "#!/bin/bash\ncd /opt/dlami/nvme/ComfyUI && source activate pytorch && python3 main.py --listen 0.0.0.0 --port 8188" > /home/ubuntu/start_comfyui_text.sh
chmod +x /home/ubuntu/start_comfyui_text.sh

# Tạo script khởi chạy ComfyUI cho Sketch To Image
echo -e "#!/bin/bash\ncd /opt/dlami/nvme/ComfyUI && source activate pytorch && python3 main.py --listen 0.0.0.0 --port 8189" > /home/ubuntu/start_comfyui_sketch.sh
chmod +x /home/ubuntu/start_comfyui_sketch.sh
(pytorch) ubuntu@ip-172-16-1-80:~$ ls
BUILD_FROM_SOURCE_PACKAGES_LICENCES  LINUX_PACKAGES_LIST                    PYTHON_PACKAGES_LICENSES      nvidia-acknowledgements  start_comfyui_text.sh
LINUX_PACKAGES_LICENSES              OSSNvidiaDriver_v550.127.05_license.txt  THIRD_PARTY_SOURCE_CODE_URLS  start_comfyui_sketch.sh
(pytorch) ubuntu@ip-172-16-1-80:~$
```

Hình 3.38: Tạo script khởi chạy ComfyUI

Bước 7: Tạo dịch vụ khởi chạy ComfyUI

```
#Dịch vụ khởi chạy ComfyUI cho Text To Image
echo -e "[Unit]
Description=ComfyUI Service
After=network.target
[Service]
Type=simple
User=ubuntu
ExecStart=/home/ubuntu/start_comfyui_text.sh
WorkingDirectory=/opt/dlami/nvme/ComfyUI
Restart=always
```

[Install]

WantedBy=multi-user.target" | sudo tee /etc/systemd/system/comfyui_text.service

#Dịch vụ khởi chạy ComfyUI cho Sketch To Image

echo -e "[Unit]

Description=ComfyUI Sketch Service

After=network.target

[Service]

Type=simple

User=ubuntu

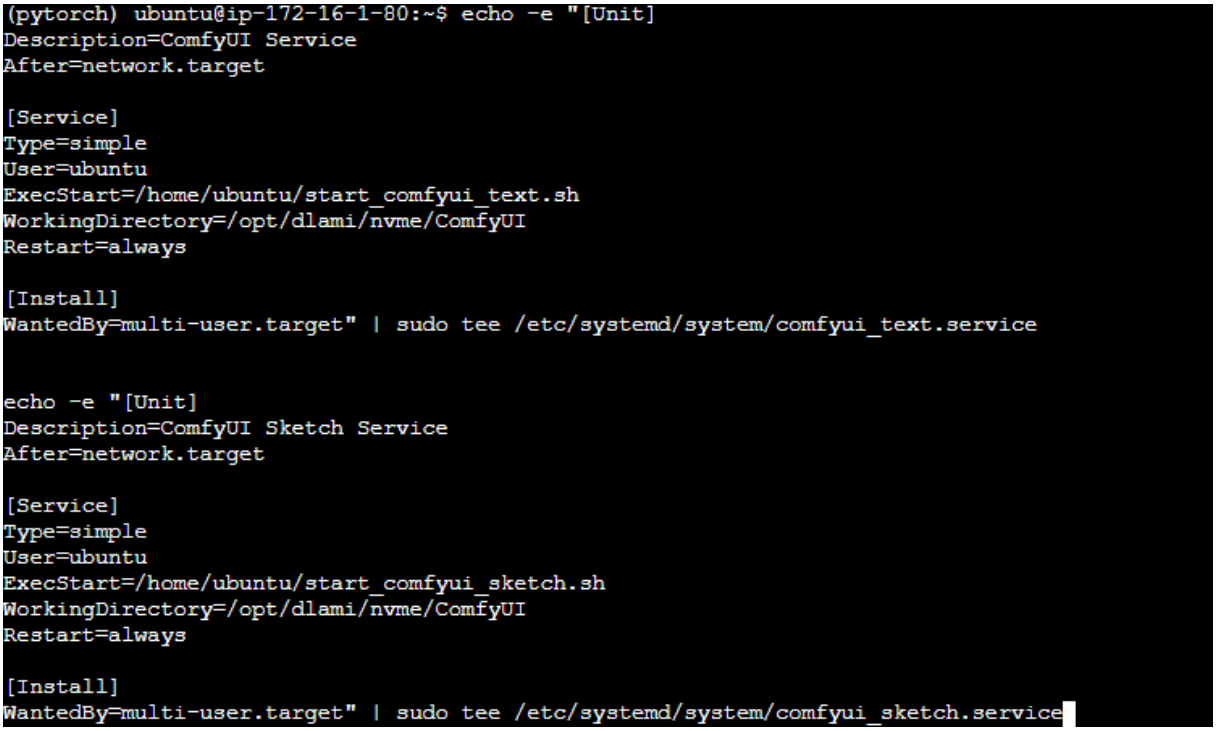
ExecStart=/home/ubuntu/start_comfyui_sketch.sh

WorkingDirectory=/opt/dlami/nvme/ComfyUI

Restart=always

[Install]

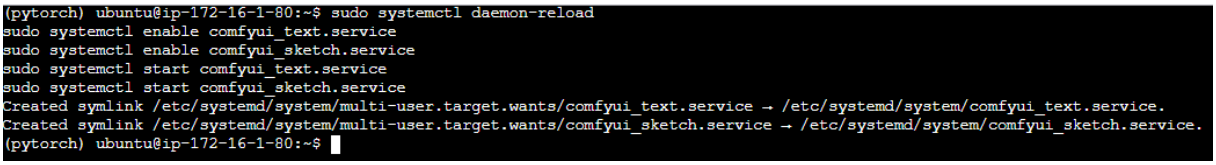
WantedBy=multi-user.target" | sudo tee /etc/systemd/system/comfyui_sketch.service



Hình 3.39: Tạo dịch vụ khởi chạy ComfyUI

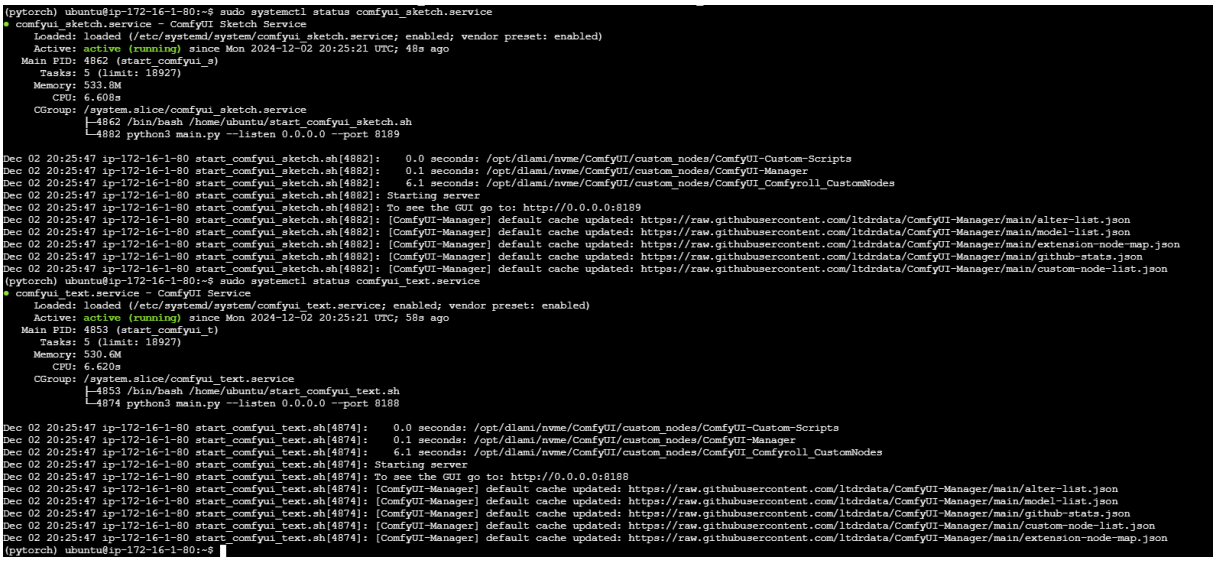
Bước 8: Khởi tạo và bật dịch vụ ComfyUI

```
sudo systemctl daemon-reload
sudo systemctl enable comfyui_text.service
sudo systemctl enable comfyui_sketch.service
sudo systemctl start comfyui_text.service
sudo systemctl start comfyui_sketch.service
```



Hình 3.40: Khởi tạo và bật dịch vụ ComfyUI

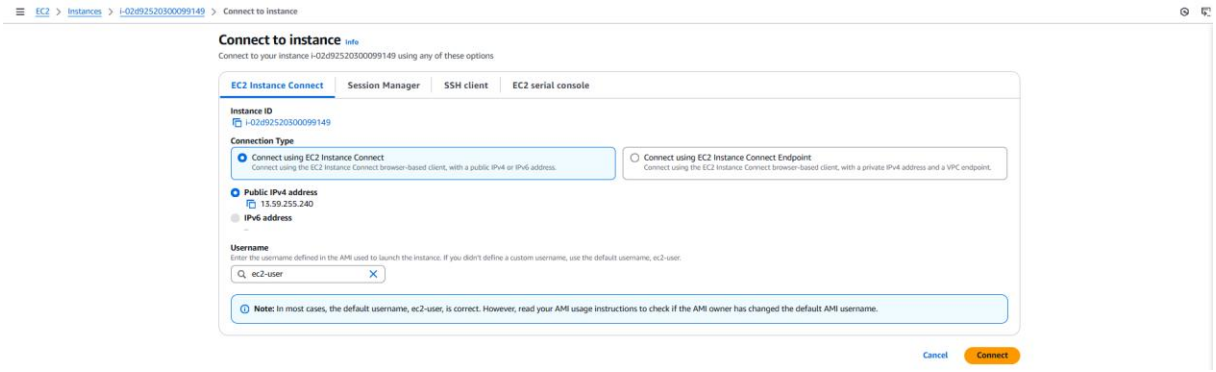
Hai dịch vụ của ComfyUI đã được chạy:



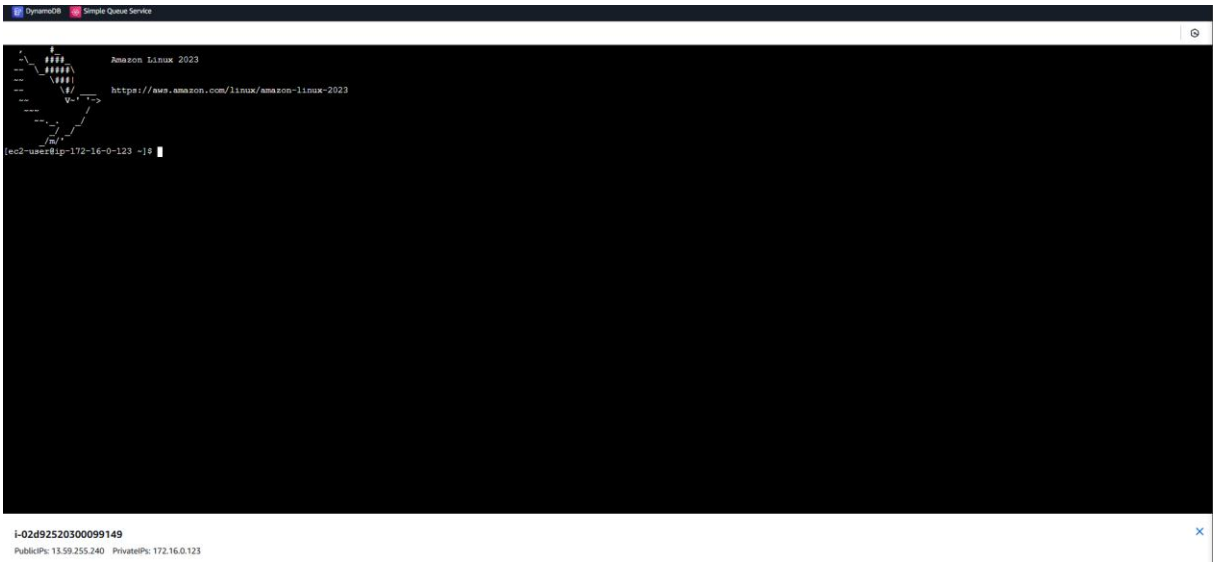
Hình 3.41: Dịch vụ ComfyUI đang chạy

3.4. TRIỂN KHAI MÁY CHỦ BACKEND

Bước 1: Kết nối với instance EC2 của máy chủ Backend thông qua AWS Management Console.



Hình 3.42: Kết nối máy chủ Backend



Bước 2: Cập nhật hệ thống và cài đặt Node.js

```
sudo yum update -y
curl -fsSL https://rpm.nodesource.com/setup_current.x | sudo bash -
sudo yum install -y nodejs
node -v
npm -v
```

```
[ec2-user@ip-172-16-0-123 ~]$ sudo yum update -y
curl -fsSL https://rpm.nodesource.com/setup_current.x | sudo bash -
sudo yum install -y nodejs
node -v
npm -v
```

Hình 3.44: Cập nhật hệ thống và cài đặt Node.js

Bước 3: Cài đặt Git và clone repo

```
sudo yum install git -y
git clone https://github.com/hiep20012003/generate-image-app
cd generate-image-app/server
```

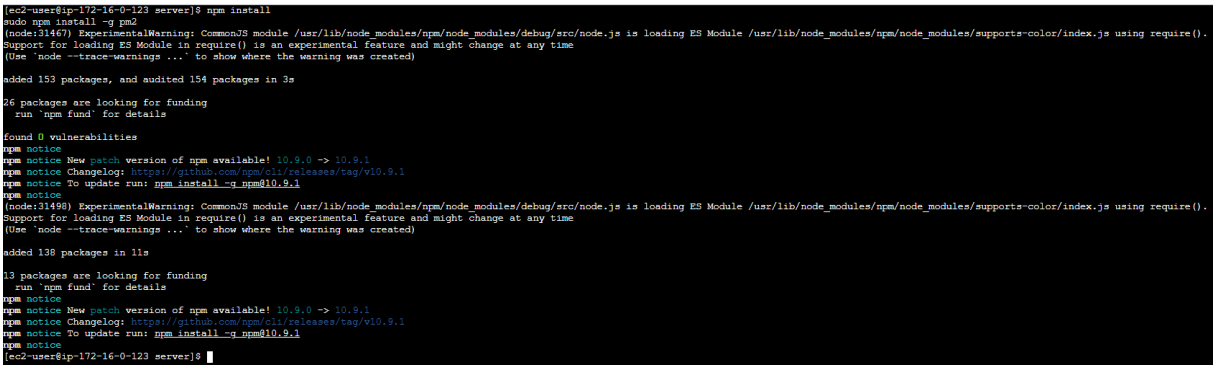
```
[ec2-user@ip-172-16-0-123 ~]$ sudo yum install git -y
git clone https://github.com/hiep20012003/generate-image-app
cd generate-image-app/server
```

Hình 3.45: Cài đặt Git và clone repo

Bước 4: Cài đặt thư viện phụ thuộc và PM2

- PM2 là một công cụ quản lý tiến trình cho ứng dụng Node.js, giúp khởi động, giám sát, và tự động khởi động lại ứng dụng khi có sự cố hoặc sau khi máy chủ khởi động lại.

```
npm install
sudo npm install -g pm2
```



Hình 3.46: Cài đặt thư viện phụ thuộc và PM2

Bước 5: Sửa biến môi trường **COMFY_UI_ADDRESS** trong file .env theo private ip của máy chủ machine learning (comfyui server)

- Trong trường hợp này là : **COMFY_UI_ADDRESS=172.16.1.80**

Bước 6: Cấu hình và khởi chạy backend server với PM2

```
pm2 startup systemd --user $USER
pm2 start index.js --name sd-app
pm2 save
```

```
[ec2-user@ip-172-16-0-123 server]$ sudo pm2 startup systemd --user $USER
sudo pm2 start index.js --name sd-app
sudo pm2 save
```

Hình 3.47: Cấu hình và khởi chạy backend server với PM2

```
[PM2] Remove init script via:
$ pm2 unstartup systemd
[PM2] Spawning PM2 daemon with pm2_home=/root/.pm2
[PM2] PM2 Successfully daemonized
[PM2] Starting /home/ec2-user/generate-image-app/server/index.js in fork_mode (1 instance)
[PM2] Done.
```

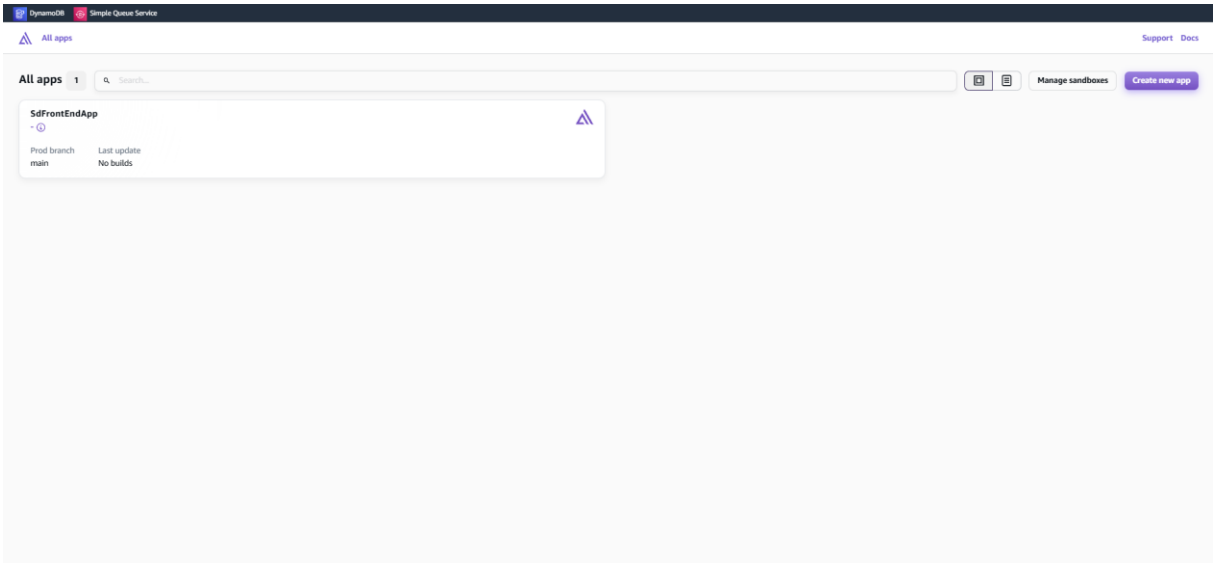
id	name	namespace	version	mode	pid	uptime	↺	status	cpu	mem	user	watching
0	sd-app	default	1.0.0	fork	32233	0s	0	online	0%	50.7mb	root	disabled

```
[PM2] Saving current process list...
[PM2] Successfully saved in /root/.pm2/dump.pm2
[ec2-user@ip-172-16-0-123 server]$ sudo pm2 log
[TAILING] Tailing last 15 lines for [all] processes (change the value with --lines option)
/root/.pm2/pm2.log last 15 lines:
PM2        | 2024-12-02T20:53:05: PM2 log: PM2 version         : 5.4.3
PM2        | 2024-12-02T20:53:05: PM2 log: Node.js version     : 23.2.0
PM2        | 2024-12-02T20:53:05: PM2 log: Current arch        : x64
PM2        | 2024-12-02T20:53:05: PM2 log: PM2 home            : /root/.pm2
PM2        | 2024-12-02T20:53:05: PM2 log: PM2 PID file        : /root/.pm2/pm2.pid
PM2        | 2024-12-02T20:53:05: PM2 log: RPC socket file     : /root/.pm2/rpc.sock
PM2        | 2024-12-02T20:53:05: PM2 log: BUS socket file     : /root/.pm2/pub.sock
PM2        | 2024-12-02T20:53:05: PM2 log: Application log path : /root/.pm2/logs
PM2        | 2024-12-02T20:53:05: PM2 log: Worker Interval     : 30000
PM2        | 2024-12-02T20:53:05: PM2 log: Process dump file   : /root/.pm2/dump.pm2
PM2        | 2024-12-02T20:53:05: PM2 log: Concurrent actions  : 2
PM2        | 2024-12-02T20:53:05: PM2 log: SIGTERM timeout     : 1600
PM2        | 2024-12-02T20:53:05: PM2 log: ================================================================================
PM2        | 2024-12-02T20:53:05: PM2 log: App [sd-app:0] starting in -fork mode-
PM2        | 2024-12-02T20:53:06: PM2 log: App [sd-app:0] online

/root/.pm2/logs/sd-app-error.log last 15 lines:
/root/.pm2/logs/sd-app-out.log last 15 lines:
0|sd-app   | Server running on: http://localhost:80
```

Hình 3.48: Backend server đang chạy trên cổng 80

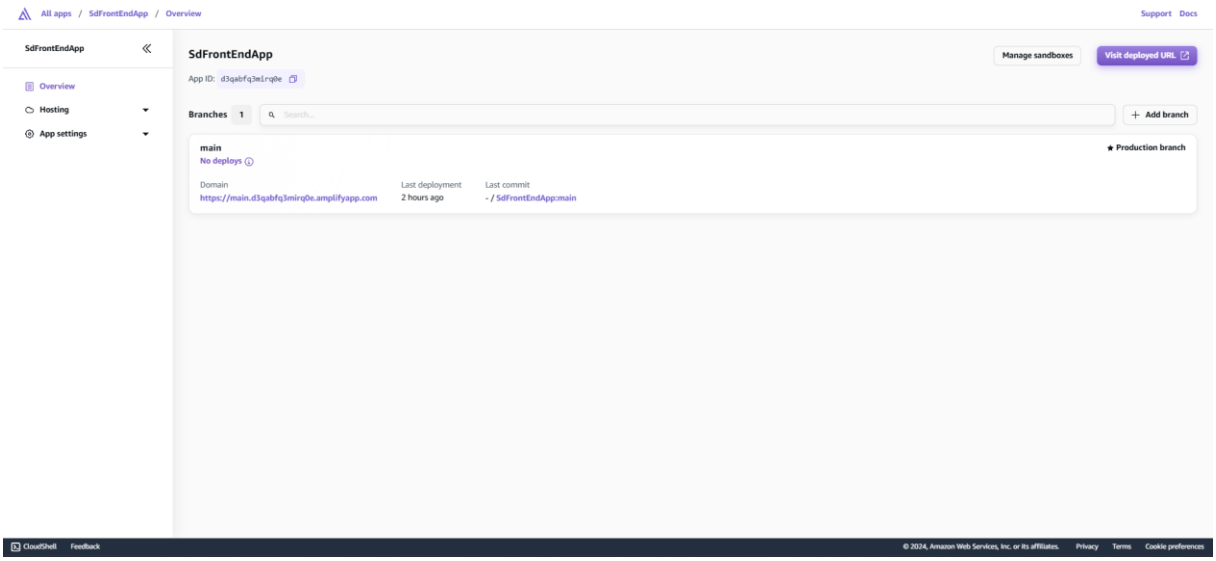
3.5. TRIỂN KHAI FRONTEND ỨNG DỤNG VỚI AWS AMPLIFY

Bước 1: Truy cập dịch vụ AWS Amplify thông qua AWS Management Console và nhấn vào SdFrontEndApp đã dược tạo với CloudFormation.



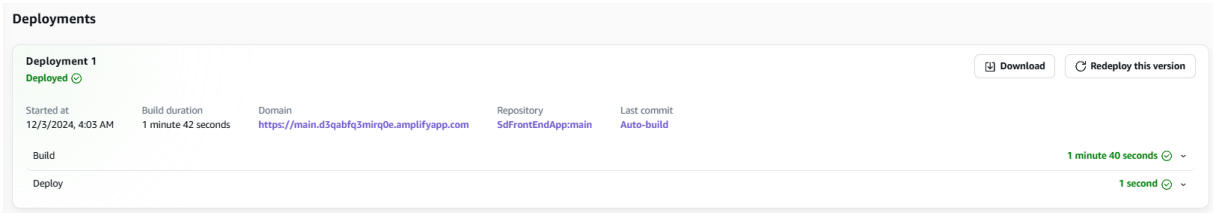
Hình 3.49: Giao diện All apps của dịch vụ AWS Amplify

Bước 2: Nhấn vào main branch → Run job



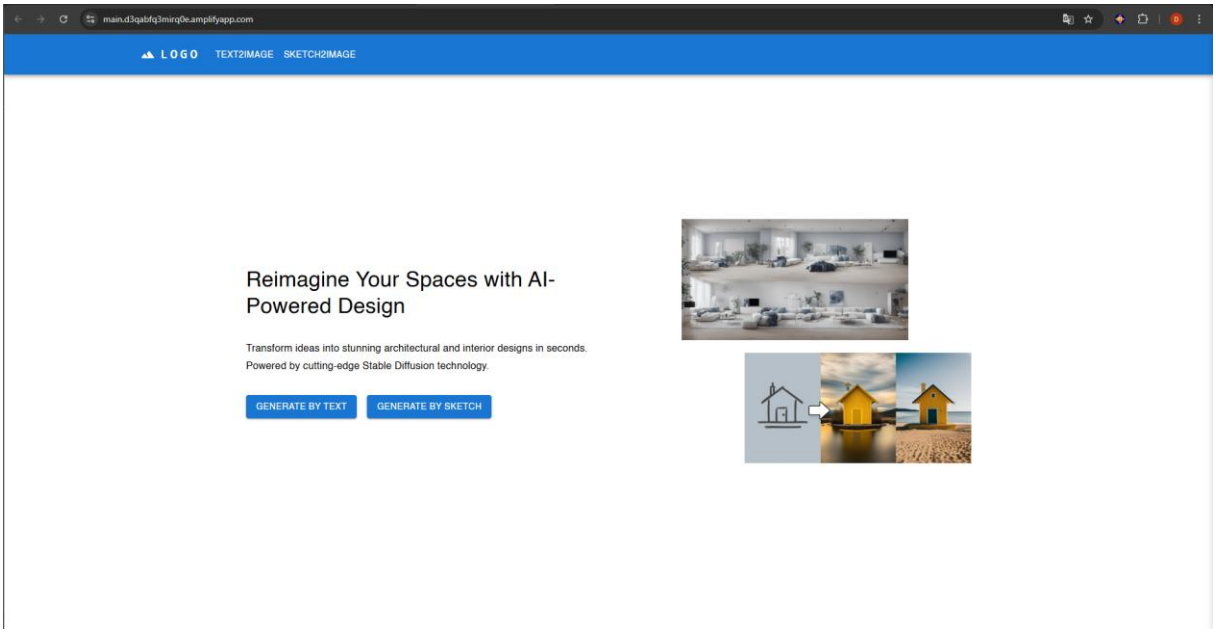
Hình 3.50: Giao diện chính của dịch vụ AWS Amplify

Hình 3.51: Run job để bắt đầu deploy



Hình 3.52: Deploy thành công

Bước 3: Truy cập vào Domain được hiển thị để truy cập Web app



Hình 3.53: Truy cập Web app qua domain được cấp