EasyOrder - API Flask e Pipeline CI/CD Descrição

Este projeto é uma aplicação Flask simples, criada para testes iniciais e aprendizado de pipeline de CI/CD usando Docker e GitHub Actions.

O objetivo é buildar uma imagem Docker, enviar para o Amazon ECR e (opcionalmente) fazer o deploy em um cluster ECS da AWS.

Observação: O deploy no ECS está configurado, mas não foi testado em ambiente real devido a limitações de hardware.

Tecnologias

Python 3.11

Flask 3.1.2

Git

Docker

AWS CLI

GitHub Actions

(Opcional) VS Code ou outro editor de código

Estrutura do Projeto Projeto_Devops/ │ ├── .github/workflows/ │ └── docker-deploy.yml # Workflow do GitHub Actions ├── app.py # Código principal da aplicação ├── Dockerfile # Configuração do container Docker ├── requirements.txt # Dependências Python └── README.md # Este arquivo

Como o Pipeline Funciona

O workflow do GitHub Actions (docker-deploy.yml) realiza as seguintes etapas:

Checkout do código → Baixa os arquivos do repositório.

Configuração da AWS CLI → Conecta o workflow à AWS usando secrets.

Login no Amazon ECR → Autentica para enviar imagens Docker.

Build e push da imagem Docker → Cria a imagem e envia para o ECR.

Deploy no ECS (opcional) → Atualiza o serviço no cluster ECS.

Como o cluster ECS não foi criado, essa etapa apenas mostra uma mensagem de deploy ignorado.

Limitações

O deploy no ECS não pôde ser executado porque o ambiente de desenvolvimento não suporta virtualização.

O workflow está configurado com um nome fictício de cluster (Devops-pipeline) para permitir que o build e push funcionem normalmente.

O pipeline ficará 100% funcional assim que um cluster ECS real for criado e o nome atualizado no workflow.

Testando Localmente

Mesmo sem o cluster ECS, é possível testar todo o pipeline:

1. Instalar ferramentas

Docker Desktop

AWS CLI

docker --version aws --version

2. Configurar credenciais AWS

aws configure

Insira suas credenciais e região (sa-east-1).

3. Build da imagem Docker

docker build -t devops-pipeline:latest . docker images

4. Login e push para o ECR

aws ecr get-login-password --region sa-east-1 | docker login --username AWS --password-stdin <aws_account_id>.dkr.ecr.sa-east-1.amazonaws.com aws ecr create-repository --repository-name devops-pipeline docker tag devops-pipeline:latest <aws_account_id>.dkr.ecr.sa-east-1.amazonaws.com/devops-pipeline:latest docker push <aws_account_id>.dkr.ecr.sa-east-1.amazonaws.com/devops-pipeline:latest

5. Deploy ECS (opcional)

aws ecs update-service --cluster Devops-pipeline --service devops-service --force-new-deployment --image <aws_account_id>.dkr.ecr.sa-east-1.amazonaws.com/devops-pipeline:latest

Vai dar erro se o cluster não existir, mas no workflow do GitHub Actions isso não quebra o pipeline.

Conclusão

O pipeline está pronto e funcional para build e push da imagem Docker.

O deploy no ECS está configurado, mas não foi testado.

Basta criar um cluster ECS real e atualizar o nome no workflow para ter o pipeline completo e funcional.

Acesso ao Código

Todo o código do projeto está disponível no meu GitHub: https://github.com/Guilherme202151/devops-pipeline-projeto.git