

CI/CD com o Github Actions - Programa de Bolsas - DevSecOps

Time Compass	Thiago Geremias De Oliveira, Yurisay Josefina Escudero Guevara, Davi Santos Cardoso Da Silva, Keila Linhares Alves e Patricia Maria Moura dos Santos
--------------	--

CI/CD com o Github Actions

Atualmente, empresas de todos os portes estão adotando **práticas de automação no ciclo de desenvolvimento**, conhecidas como **CI/CD (Integração Contínua e Entrega Contínua)**. O objetivo é simples: entregar código com **velocidade, segurança e consistência**.

Nesse cenário, ferramentas como **GitHub Actions** e **ArgoCD** ganham destaque:

- O **GitHub Actions** permite automatizar o build, os testes e a publicação de imagens Docker diretamente a partir dos commits.
- O **ArgoCD**, por sua vez, implementa o conceito de **GitOps**, onde o próprio Git é a “fonte de verdade” da infraestrutura e dos deploys em Kubernetes.

Dominar essas ferramentas e práticas é essencial para profissionais que desejam atuar com DevOps, SRE, Cloud ou Desenvolvimento moderno, pois é isso que sustenta as entregas contínuas de grandes empresas

Objetivo

Automatizar o ciclo completo de desenvolvimento, build, deploy e execução de uma aplicação FastAPI simples, usando GitHub Actions para CI/CD, Docker Hub como registry, e ArgoCD para entrega contínua em Kubernetes local com Rancher Desktop.

Pré-requisitos

- **Conta no GitHub** (repo público)
- **Conta no Docker Hub** com token de acesso
- **Rancher Desktop com Kubernetes habilitado**
- **kubectl configurado corretamente** (`kubectl get nodes`)
- **ArgoCD instalado no cluster local**
- **Git instalado**
- **Python 3 e Docker instalados**

Etapas do Projeto

Etapa 1 – Criar a aplicação FastAPI

- Criar um repositório Git para o projeto (Exemplo: hello-app) com o arquivo `main.py`;
- Criar um `Dockerfile` para executar esse aplicativo;
- Criar um repositório Git para os manifestos do ArgoCD (exemplo: hello-manifests).

`main.py`

```
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
async def root():
    return {"message": "Hello World"}
```

Etapa 2 – Criar o GitHub Actions (CI/CD)

- Criar o arquivo de workflow no github actions para buildar e fazer a publicação da imagem no Docker Hub, que será o nosso container registry neste projeto.
- A pipeline também deve realizar um Pull Request no repositório de manifestos, fazendo a alteração da imagem.
- Criar os segredos no GitHub: `DOCKER_USERNAME`, `DOCKER_PASSWORD`, `SSH_PRIVATE_KEY`
- Ter acesso de gravação ao repositório de manifestos usado pelo ArgoCD;

Etapa 3 – Repositório Git com os manifests do ArgoCD

- Criar os manifests do kubernetes de deployment e service para o Hello App no repositório de manifests;

Esse repositório será usado pelo ArgoCD para sincronizar o deploy.








Etapa 4 – Criar App no ArgoCD

- Na interface do ArgoCD criar o vínculo com o repositório de manifests;
- Criar o app no ArgoCD

Etapa 5 – Acessar e testar a aplicação localmente

- Usando o port-forward para rotear as portas do serviço, acesse via navegador: <http://localhost:8080/>
- Alterar o repositório da aplicação, modificando a mensagem dentro do código python de Hello World para qualquer outra mensagem e verificando se após o processo de CI/CD a imagem foi atualizada no ambiente Kubernetes.

Entregáveis esperados

-  Link do repositório Git com a aplicação FastAPI + Dockerfile + GitHub Actions
-  Link do repositório com os manifests (deployment.yaml, service.yaml)
-  Evidência de build e push da imagem no Docker Hub
-  Evidência de atualização automática dos manifests com a nova tag da imagem
-  Captura de tela do ArgoCD com a aplicação sincronizada
-  Print do `kubectl get pods` com a aplicação rodando
-  Print da resposta da aplicação via `curl` ou navegador.