Projeto DevOps

Deploy automatizado de API com FastAPI, Jenkins e Kubernetes

Data de desenvolvimento	28/05/2025
Data da última atualização	04/06/2025
Autores	Davi Santos Cardoso da Silva (davi.silva@compasso.com.br) Thiago Geremias de Oliveira (thiago.geremias@compasso.com.br)

Descrição Geral

Este projeto tem como objetivo ensinar na prática os conceitos de CI/CD utilizando as tecnologias FastAPI, Docker, Jenkins e Kubernetes. Os bolsistas trabalharão com um código base em FastAPI e desenvolverão uma esteira de automação que fará o deploy automatizado da aplicação em um cluster Kubernetes local.

Tecnologias Utilizadas

- **FastAPI**: Framework web em Python.
- **Docker**: Para conteinerização da aplicação.
- **Docker Hub**: Registro público de imagens.
- **Jenkins**: Ferramenta de CI/CD.
- **Kubernetes local**: Minikube, Kind, Docker Desktop ou Rancher Desktop.

Código Base da Aplicação (Backend)

projeto-kubernetes-pb-desafio-jenkins/backend/main.py at main · box-genius/projeto-kubernetes-pb-desafio-jenkins · GitHub

Fases do Projeto

Fase 1: Preparação do projeto

Nesta fase será necessário crias as seguintes atividades:

- Criar um repositório de código no Github para inserir a aplicação de exemplo;
- Criar conta no Docker Hub.
- Verificar acesso ao cluster Kubernetes local.
- Validar execução local com uvicorn.

Entregáveis: Código rodando localmente, repositório do github criado e ambiente preparado.

Fase 2: Conteinerização com Docker

Nesta fase é esperado que se execute esse código em um container e depois faça a publicação deste container no Dockerhub

- Criar o dockerfile;
- Fazer build: docker build -t usuario/fastapi-hello:latest .
- Fazer push: docker push usuario/fastapi-hello:latest
- Versionar o dockerfile junto com o código da aplicação no github.
-

Entregáveis: Imagem publicada no Docker Hub.

Fase 3: Arquivos de Deploy no Kubernetes

Nesta fase vamos manualmente criar o deployment e service do kubernetes para que rode a imagem de container que acabamos de subir na fase anterior, no Kubernetes local.

- Criar o yaml de deployment da aplicação e aplicá-lo no cluster
- Criar o yaml de service do deploymento e aplicá-lo no cluster

Entregáveis: Aplicativo exposto em localhost:30001 via NodePort ou rodando via port-forward. O aplicativo precisa estar funcionando a partir do Kubernetes.

Fase 4: Jenkins - Build e Push

Nesta fase vamos criar uma pipeline no jenkins que quando acionada, seja manualmente ou por alguma mudança no repositório do github, execute o build e o push da imagem de container.

- Criar a pipeline no Jenkins;
- Realizar o stage de build;
- Realizar o stage de push;

Entregáveis: Pipeline funcional no Jenkins até o push da imagem.

Fase 5: Jenkins - Deploy no Kubernetes

Nesta fase o Jenkins será configurado para acessar o kubectl e acesso ao cluster local, assim como uma etapa de deploy será incluída no pipeline.

- Jenkins precisa acessar o kubectl (usar agent com kubectl e kubeconfig configurados);
- Adicionar etapa de deploy no Jenkinsfile;
- Testar a pipeline completa;

Entregáveis: Pipeline completo com deploy automatizado.

Fase 6: Documentação

Criar README.md com:

- Passos para reproduzir o projeto.
- Tecnologias usadas.
- Print do Jenkins.
- Apresentar pipeline funcionando: build, push e deploy.

Entregáveis: Documentação completa e apresentação final.

Desafios Extras:

- Criar uma etapa após o push da imagem de container, realizar o scanner de vulnerabilidades, utilizando o Trivy.
- Criar um webhook com o Slack ou Discord para avisar quando a pipeline for atualizar no ambiente Kubernetes.
- Subir o Sonarqube em ambiente Docker e conectá-lo com o Jenkins e enviar todo o código da aplicação para a análise SAST.
- Utilizar Helm Chart para implantar a aplicação no Kubernetes.