energize+

ENERGIA SUSTENTÁVEL

# Projeto de DevOps: Implementação de Pipeline CI/CD para a Aplicação Energize API

Relatório Técnico de Entrega

**Nome:** Maysa Angélica de Paula da Fonseca
**RM:** 557328
**Curso:** Análise e Desenvolvimento de Sistemas
**Instituição:** FIAP

Brasília, 08 de Outubro de 2025

**Sumário**

## 1. Introdução ao Projeto

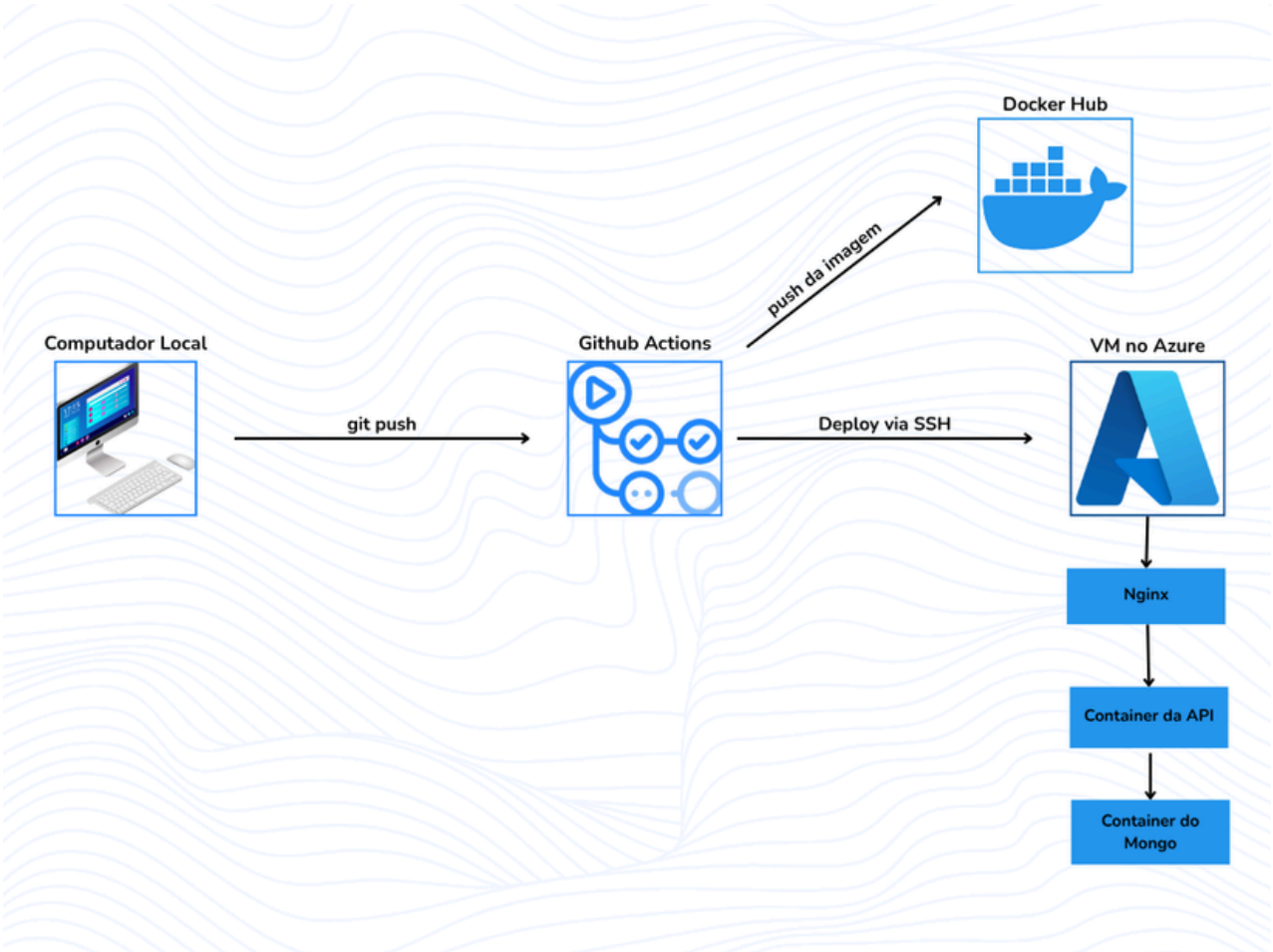
Este documento detalha a implementação de um ciclo de vida de DevOps completo para a aplicação "Energize API", um projeto desenvolvido em Java com Spring Boot e MongoDB. O objetivo principal foi aplicar práticas modernas de automação para criar um fluxo de integração e entrega contínuas (CI/CD), partindo de uma infraestrutura de nuvem criada do zero na Microsoft Azure.

O projeto abrange desde a configuração da infraestrutura, passando pela conteinerização da aplicação e do banco de dados, até a criação de um pipeline totalmente automatizado que realiza o build, a publicação e o deploy da aplicação a cada nova alteração no código-fonte.

## 2. Arquitetura da Solução Implementada

A solução foi arquitetada para ser robusta, escalável e segura, utilizando os seguintes componentes:

- **Provedor de Nuvem:** Microsoft Azure.
- **Infraestrutura:** Uma Máquina Virtual (VM) com sistema operacional Ubuntu 22.04 LTS, configurada com uma Rede Virtual (VNet) e um Network Security Group (NSG) para controle de acesso via firewall.
- **Orquestração:** Docker e Docker Compose foram utilizados para criar e gerenciar os containers da aplicação (API) e do banco de dados (MongoDB), garantindo a portabilidade e a consistência do ambiente.
- **CI/CD:** O GitHub Actions foi a ferramenta escolhida para orquestrar a automação, integrada diretamente ao repositório do código-fonte.
- **Registro de Imagens:** O Docker Hub foi utilizado como registro para armazenar as imagens Docker geradas pelo pipeline.



## 3. Conteinerização da Aplicação

### 3.1. Dockerfile

Foi empregada uma estratégia de multi-stage build no Dockerfile para otimizar o tamanho e a segurança da imagem final.

- **Estágio 1 (Builder):** Utiliza uma imagem completa do Maven e JDK para compilar o código-fonte Java, executar testes e gerar o arquivo executável .jar.
- **Estágio 2 (Final):** Inicia a partir de uma imagem JRE (Java Runtime Environment) mínima, que é muito mais leve. Apenas o .jar compilado no estágio anterior é copiado para esta imagem final. Essa abordagem remove ferramentas de build e código-fonte desnecessários, reduzindo a superfície de ataque e o tamanho da imagem.

### 3.2. Docker Compose

O docker-compose.yml foi utilizado para orquestrar os serviços locais e no ambiente de nuvem. Ele define dois serviços principais:

- **api:** O container da aplicação Energize API.
- **mongo:** O container do banco de dados MongoDB.

Foram utilizados recursos essenciais como networks para a comunicação isolada entre os containers, volumes para garantir a persistência dos dados do MongoDB, e env_file para carregar as variáveis de ambiente de forma segura.

## 4. Pipeline de CI/CD com GitHub Actions

O coração da automação é um workflow do GitHub Actions, definido em .github/workflows/main.yml.

### 4.1. Visão Geral do Pipeline

O pipeline é acionado automaticamente a cada push na branch main. Ele é composto por dois jobs sequenciais: build-and-push e deploy-to-staging.

A estratégia de CI/CD adotada neste projeto focou na implementação da Entrega Contínua (Continuous Delivery), com deploy automatizado para o ambiente de Staging. Esta abordagem segue as melhores práticas de mercado para garantir a segurança e a estabilidade do ambiente de produção.

Neste modelo, toda alteração no código-fonte é automaticamente implantada no ambiente de Staging, que funciona como um espelho do de produção, permitindo uma validação final completa. A promoção da versão de Staging para o ambiente de Produção seria então realizada de forma controlada, após a verificação manual, minimizando o risco de introduzir bugs para o usuário final. Para o escopo deste desafio, a automação até o ambiente de Staging demonstrou com sucesso a competência na criação de um pipeline de CI/CD robusto e seguro.

### 4.2. Segredos e Autenticação Segura

Para garantir a segurança, nenhuma credencial foi exposta no código. Foram utilizados os "Repository Secrets" do GitHub para armazenar de forma criptografada:

- Credenciais do Docker Hub (DOCKER_USERNAME, DOCKER_TOKEN).
- Credenciais de acesso à VM no Azure (AZURE_SSH_HOST, AZURE_SSH_USER, AZURE_SSH_KEY).

### 4.3. Etapas do Workflow

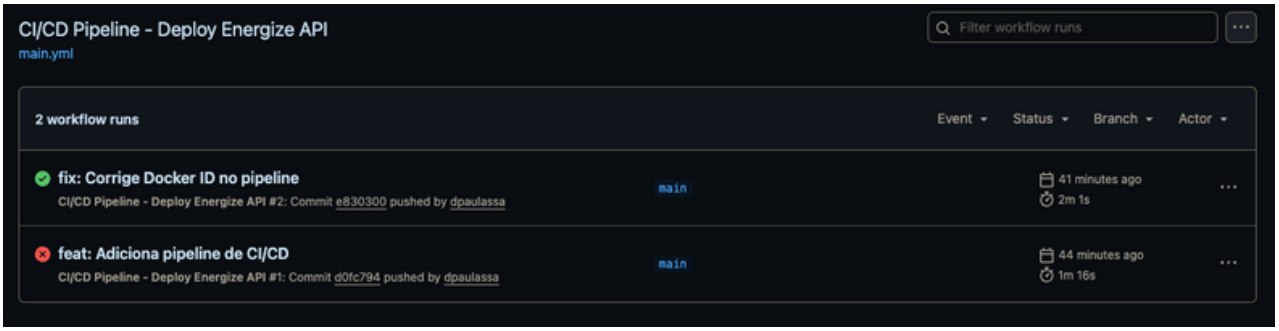
- **Job: build-and-push:**
  - Faz o checkout do código.
  - Autentica-se no Docker Hub.
  - Constrói a imagem Docker da aplicação.
  - Envia a nova imagem para o repositório no Docker Hub com a tag latest.
- **Job: deploy-to-staging:**
  - Aguarda o sucesso do job anterior.
  - Conecta-se à VM no Azure via SSH.

Executa um script que baixa a nova imagem (docker pull) e reinicia os serviços com o Docker Compose (docker compose up -d), atualizando a aplicação.

## 5. Evidências de Funcionamento

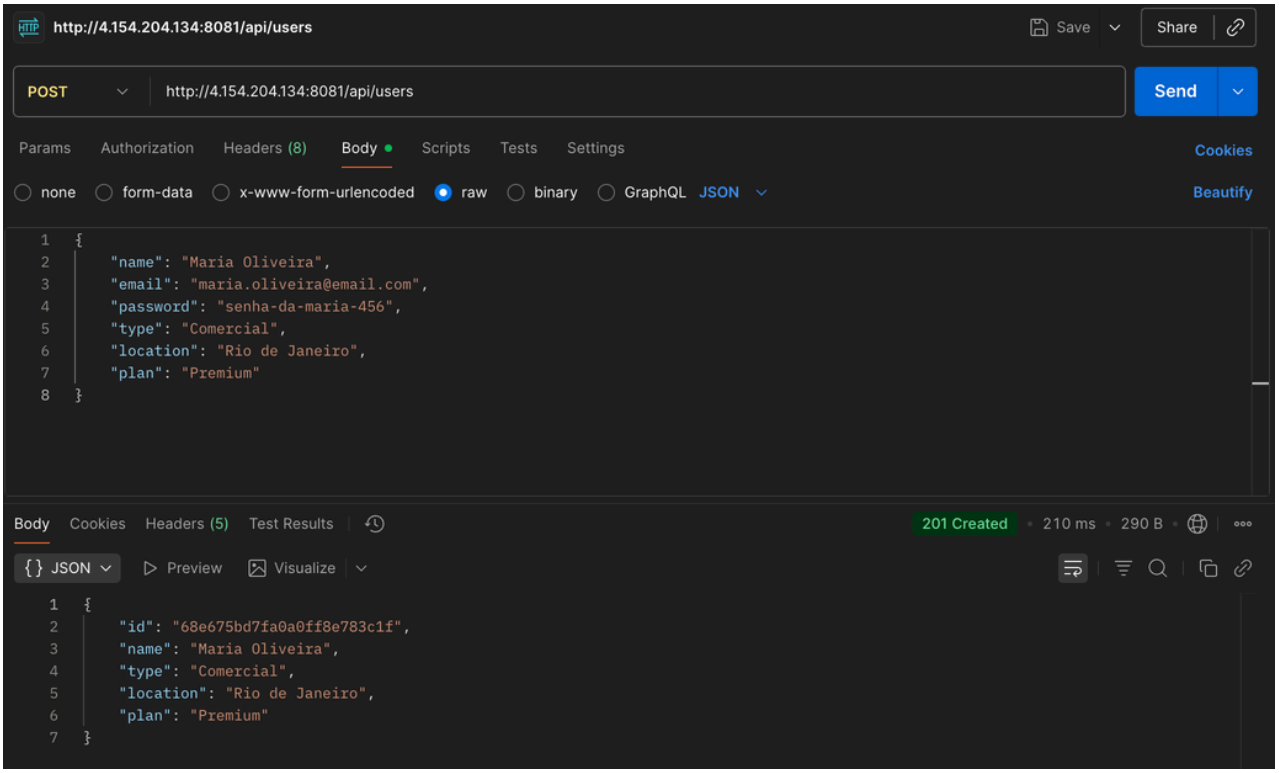
### 5.1. Execução do Pipeline

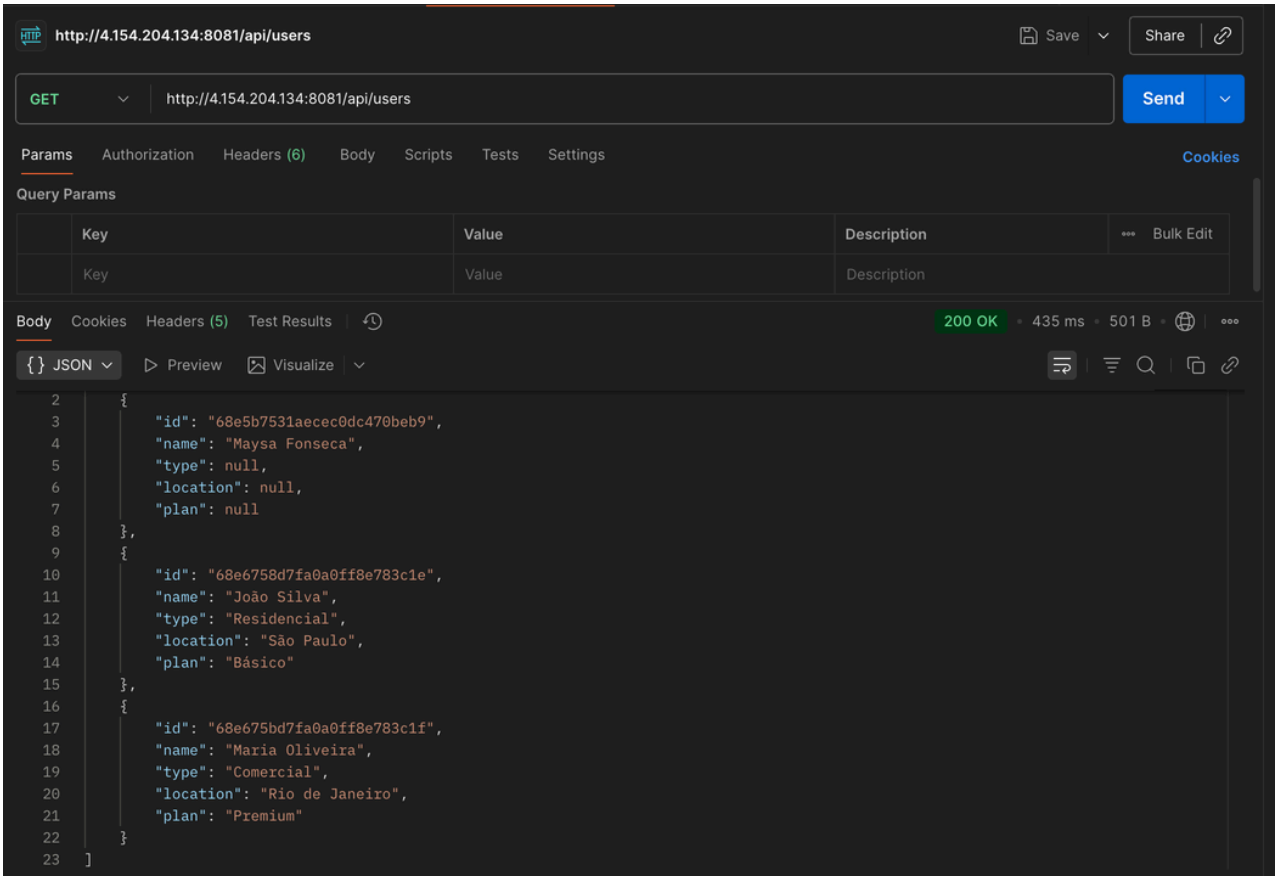
A imagem abaixo evidencia a execução bem-sucedida do pipeline no GitHub Actions, com ambos os jobs (Build e Deploy) concluídos.




### 5.2. Ambientes em Execução

Abaixo, a prova de que o ambiente de staging está no ar, acessível publicamente e retornando dados JSON do banco de dados após a execução do pipeline.






Estilos de formatação ☑

```
[
  {
    "id": "68e5b7531aecec0dc470beb9",
    "name": "Maysa Fonseca",
    "type": null,
    "location": null,
    "plan": null
  },
  {
    "id": "68e6758d7fa0a0ff8e783c1e",
    "name": "João Silva",
    "type": "Residencial",
    "location": "São Paulo",
    "plan": "Básico"
  },
  {
    "id": "68e675bd7fa0a0ff8e783c1f",
    "name": "Maria Oliveira",
    "type": "Comercial",
    "location": "Rio de Janeiro",
    "plan": "Premium"
  }
]
```

## 6. Desafios Encontrados e Soluções Aplicadas

Durante o projeto, diversos desafios práticos de um ambiente de nuvem real foram encontrados e solucionados:

- **Desafio 1: Bloqueio de Região na Assinatura Azure.**
  - **Problema:** A assinatura do tipo "Universitária" possuía uma política que restringia a criação de recursos em certas regiões, como a do Brasil.
  - **Solução:** A solução foi criar uma nova assinatura pessoal do tipo "Free Tier", que ofereceu maior flexibilidade e permitiu a criação dos recursos na região desejada.

- **Desafio 2: Acesso Externo Bloqueado (Firewall).**
  - **Problema:** Após o deploy, a aplicação estava rodando, mas não era acessível pela internet (erro ERR_CONNECTION_TIMED_OUT).
  - **Solução:** O problema foi diagnosticado como o firewall do Azure (Network Security Group). A solução foi criar regras de entrada (inbound rules) para liberar explicitamente as portas da aplicação (8080 para produção e 8081 para staging).

- **Desafio 3: Falha de Permissão no Pipeline (sudo).**
  - **Problema:** O pipeline de CI/CD falhava na etapa de deploy com o erro sudo: a terminal is required to read the password.
  - **Solução:** O erro ocorria porque o comando sudo exigia uma senha em um ambiente não interativo. A solução foi configurar o arquivo sudoers na VM (via visudo) para permitir que o usuário azureuser executasse comandos do Docker especificamente, sem a necessidade de senha (NOPASSWD).

## 7. Conclusão

Este projeto demonstrou com sucesso a aplicação prática de um ciclo de DevOps completo. A partir da configuração de uma infraestrutura de nuvem, foi possível containerizar uma aplicação complexa e automatizar completamente seu processo de entrega. Os desafios encontrados não foram impedimentos, mas sim oportunidades de aprendizado profundo sobre governança de nuvem, configuração de redes, segurança e administração de sistemas Linux, competências essenciais para um profissional da área.

## 8. Checklist de Entrega

| Item | OK |
|---|---|
| Projeto compactado em .ZIP com estrutura organizada | ✅ |
| Dockerfile funcional | ✅ |
| docker-compose.yml | ✅ |
| Pipeline com etapas de build, teste e deploy | ✅ |
| README.md com instruções e prints | ✅ |
| Documentação técnica com evidências (PDF ou PPT) | ✅ |
| Deploy realizado nos ambientes staging e produção | ✅ |