# Relatório do Desafio DevOps Lacrei Saúde

Cleyton Nobre
Belo Horizonte/MG - Brasil
cleytonlnobr@gmail.com

## I. DESCRIÇÃO DO PROBLEMA

O presente relatório tem como objetivo apresentar as decisões de projeto tomadas para o problema proposto pela Lacrei Saúde. Trata-se de um desafio técnico cujo propósito é avaliar a capacidade de configurar e gerenciar ambientes de deploy seguros, estáveis e bem documentados. O desafio consiste em criar ambientes de staging e produção para uma aplicação fictícia desenvolvida em Node.js, assegurando que o processo de entrega seja automatizado, confiável.

Além da documentação da entrega da aplicação, esse documento contém uma a proposta de uma integração com a AssAs, uma API de pagamentos com suporte a split, e uma breve a descrição de um procedimento de rollback em caso de falhas no ambiente de produção.

## II. DECISÃO DE PROJETO

Para a realização desse desafio, foi criado um repositório contendo a aplicação Node.js com uma rota simples de status. A aplicação foi containerizada com Docker, permitindo sua execução de forma isolada e reprodutível tanto localmente quanto nos ambientes de staging e produção. Os ambientes foram configurados na AWS, utilizando instâncias EC2 para hospedar os serviços.

O processo de deploy foi automatizado por meio do GitHub Actions, permitindo que qualquer alteração na branch principal acione o pipeline de staging, enquanto o ambiente de produção depende de uma validação por pull request para maior controle.

Como medida adicional, foi descrita uma estratégia de rollback baseada em versões das imagens Docker. Cada nova versão gerada no pipeline é marcada com um timestamp, dessa forma permitido restaura o sistema.

Para melhor origanização o projeto foi estrutura da seguinte forma:

- Repositório Git contendo:
  - Aplicação Node.js (app.js) com rota `/status`.
  - Dockerfile configurado.
  - Workflows GitHub Actions separados para **staging** e **develop**.
  - README técnico explicando como reproduzir os ambientes localmente e em nuvem.

Com essa estrutura, adotou-se o seguinte Pipeline de Deploy:

1) Push na branch `develop` → testes automático.
2) Push na branch `main` → deploy automático para **staging**.
3) Notificações de sucesso ou erro no deploy via GitHub Actions.

Como medida de segurança adicional, é necessário uma estratégia de rollback baseada em versões das imagens Docker. Cada nova versão gerada no pipeline é marcada com um timestamp, e o sistema permite restaurar rapidamente versões anteriores em caso de falha, proporcionando maior segurança na manutenção do serviço.

## III. PROPOSTA DE INTEGRAÇÃO COM A API DA ASSAS

A viabilidade de integração com a API da Assas foi analisada com base na documentação pública disponibilizada pela plataforma. A seguir, apresenta-se uma proposta de fluxo de comunicação entre a aplicação e a Assas, com o objetivo de registrar transações, criar cobranças e realizar a divisão automática dos valores recebidos entre os beneficiários e a organização.

A arquitetura proposta prevê um módulo de pagamentos dedicado, responsável por orquestrar essas operações de forma segura, transparente e escalável.

A estrutura geral seria a seguinte:

O backend da aplicação contará com um módulo `payments.js`, encarregado das seguintes responsabilidades:

- Criação de cobranças;
- Registro de clientes.

Como prova de conceito, sugere-se a utilização do Postman para simular chamadas HTTP aos principais endpoints da API da Assas, como:

Criação de assinaturas (POST /subscriptions);

Consulta de pagamentos (GET /payments).

Abaixo, apresenta-se a estrutura da requisição para POST /subscriptions:

```
{
  "customer": {
    "name": "João da Silva",
    "email": "joao.silva@gmail.com",
    "cpf": "12345678909"
  },
  "billing_type": "BOLETO",
  "value": 4990,
  "cycle": "MONTHLY",
  "description": "Assinatura mensal
premium",
  "next_due_date": "2025-06-01"
}
```

```
{
 "customer": "cus_000005219613",
 "billingType": "BOLETO",
 "value": 2000.00,
 "dueDate": "2023-07-21",
 "installmentCount": 10,
 "installmentValue": 200.00
}
```

Abaixo ilustrado a estrutura de requisição para GET /payments

```
{
 "filters": {
  "start_date": "2025-05-01",
  "end_date": "2025-05-31",
  "status": "RECEIVED"
 }
}
```

## IV. CONSIDERAÇÕES

Por meio da utilização do Docker, GitHub Actions e infraestrutura na AWS, foi possível configurar ambientes isolados de staging e produção, com controle de versões e rollback eficiente baseado em tags com timestamp das imagens Docker.

A proposta de integração com a API da Assas, utilizando métodos REST documentados para criar cobranças recorrentes e consultar pagamentos, o que adicionaria valor à aplicação ao permitir a gestão automatizada de assinaturas com divisão de valores (split de pagamento).

Durante o desenvolvimento do projeto, algumas dificuldades foram enfrentadas, como:

Gerenciamento de credenciais sensíveis: foi necessário configurar variáveis de ambiente e secrets no GitHub Actions e nas instâncias da AWS, garantindo segurança sem comprometer a automação.

Configuração de rede na AWS: o provisionamento correto de instâncias EC2 com acesso público, firewall (security groups) e permissões específicas exigiu atenção e testes cuidadosos.

Controle de versões e rollback: implementar uma estratégia de rollback confiável exigiu a criação de um mecanismo robusto para versionar imagens Docker com tags baseadas em tempo e facilitar sua restauração.

Familiarização com a API da Assas: apesar de bem documentada, foi necessário um tempo inicial para entender os fluxos e formatos esperados, especialmente relacionados a operações de split.