🚀 Desafio 1 DevOps:
Do Commit ao Deployment Automatizado

Objetivo: Desenvolver uma aplicação simples e implementar um pipeline completo de CI/CD que garanta a qualidade do código e a entrega automatizada em um ambiente de staging/produção, utilizando **containers** e **GitHub Actions**.

Formação da Equipe (Até 5 pessoas): Para simular um ambiente real e quebrar silos organizacionais, sugere-se a divisão de papéis:
1. **Engenheiro de CI/CD:** Responsável pela configuração dos workflows no GitHub Actions.
2. **Desenvolvedor(a) Backend:** Responsável pela lógica da aplicação e testes unitários.
3. **Especialista em Containers:** Responsável pelo Dockerfile e otimização da imagem.
4. **Engenheiro(a) de QA/Qualidade:** Responsável pela análise estática e testes de integração.
5. **Especialista em Observabilidade:** Responsável por garantir que a aplicação possa ser monitorada após o deploy

🛠️ O Desafio Técnico

Etapa 1: Desenvolvimento e Integração Contínua (CI)
A equipe deve criar um repositório no GitHub para uma aplicação (ex: Node.js ou Python).
• **Ação:** Configurar um workflow de CI que, a cada *push* ou *pull request*, execute:
  ◦ **Build:** Instalação de dependências e compilação.
  ◦ **Testes:** Execução de testes unitários e de integração (cobertura mínima recomendada de 80%).
  ◦ **Análise Estática:** Verificação de padrões de código e vulnerabilidades.
• **Regra de Ouro:** O código só pode ser integrado ao repositório principal se o status do pipeline estiver "verde".
Etapa 2: Conteinerização e Registry
Para garantir consistência entre ambientes:
• **Ação:** Criar um **Dockerfile** otimizado (preferencialmente usando imagens *alpine*).
• **Automação:** O pipeline deve realizar o build da imagem Docker e fazer o **push** para um Container Registry (como Docker Hub ou GitHub Packages) após o sucesso dos testes.
Etapa 3: Continuous Delivery (CD) com GitHub Actions
Implementar a extensão do CI para a preparação automatizada de *releases*.
• **Ação:** Configurar o deployment automatizado em um ambiente de *staging* que espelhe a produção.
• **Diferencial (CD Manual):** A decisão de deploy para o ambiente de **produção** deve ser automatizada no processo, mas pode exigir um clique manual (aprovação) no GitHub Actions para simular o modelo de **Continuous Delivery**.
• **Segurança:** Utilizar **GitHub Secrets** para gerenciar chaves de API e credenciais de acesso de forma segura.
Etapa 4: Observabilidade Básica
Após o deploy, a equipe deve demonstrar que a aplicação está saudável.
• **Ação:** Integrar um agente de monitoramento simples ou expor métricas básicas (latência, taxa de erro) que possam ser visualizadas em um dashboard ou logs.

📝 Critérios de Avaliação

1. **Automação Extensiva:** O pipeline executa todas as etapas sem intervenção manual humana até o staging?
2. **Velocidade do Build:** O tempo total de execução do pipeline é inferior a 10 minutos?
3. **Tratamento de Falhas:** Se um teste falha, o deploy é bloqueado corretamente?
4. **Documentação:** Existe um arquivo YAML claro no diretório `.github/workflows/` descrevendo os *jobs* e *steps*?
5. **Cultura DevOps:** A equipe demonstrou colaboração e compartilhamento de responsabilidades?

Data de entrega/apresentação: 04 / Março / 2026
Local: Sala 09 ou Lab. 2.

Observação: Este desafio foca na criação de uma infraestrutura que evite o "integration hell" e permita entregas rápidas e confiáveis, conforme os princípios de desenvolvimento moderno

Respeitosamente,
Prof. Cloves Rocha