

WorkShop: Entregando com Confiança

Primeiros passos em CI/CD com GitHub Actions

Alunos: Bruno Lacerra
Jonathan Davi
Thomaz Otávio

Sumário.

- 1 Introdução.
- 2 O que é CI/CD?
- 3 O Problema e a Solução.
- 4 Vantagens.
- 5 Ferramenta: GitHub Actions.
- 6 Começando com CI/CD.
- 7 Conclusão.

Contexto do Desenvolvimento de Software

- Projetos de software cada vez mais complexos
- Equipes grandes e distribuídas
- Pressão por rapidez e qualidade

Introdução

Desafios Enfrentados

- Erros acumulados ao longo do projeto
- Retrabalho e atrasos nas entregas
- Falta de confiança no software

O Papel da Automação

- Automação reduz erros manuais
- Processos repetíveis e confiáveis
- Base para práticas modernas

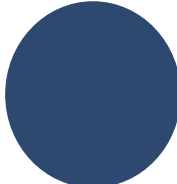
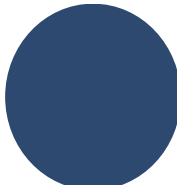
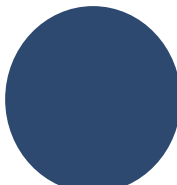
O que é CI/CD

Integração Contínua (CI)

- Testes automáticos a cada commit
- Detecta falhas rapidamente
- Facilita colaboração da equipe

O que é CI/CD

Entrega Contínua (CD)

-  Deploy automatizado
-  Software sempre pronto
-  Redução de riscos

O que é CI/CD

Resumindo CI/CD

- CI: integração e testes constantes
- CD: entrega contínua de versões
- Confiabilidade + rapidez

O problema: Desenvolvimento Tradicional

- Deploys manuais → demorados, propensos a erro.
- Falta de padronização entre equipes.
- Integração de código só no fim do projeto → muitos conflitos.
- Testes feitos tarde demais → bugs em produção.

O Problema na Prática

- Desenvolvedor A e B trabalhando em paralelo.
- Só integram semanas depois → falhas na compatibilidade.
- Deploy exige passos manuais (copiar arquivos, rodar scripts locais).
- Se algo quebra, não há rollback automático.

Exemplo prático

- CI (Integração Contínua): todo commit é testado e integrado automaticamente.
- CD (Entrega/Implantação Contínua): a aplicação vai para produção (ou staging) com segurança e repetibilidade
- Integrar cedo → detectar erros cedo → entregar rápido.

Problemas e solução

Solução na prática

- Dev faz push para repositório
- Pipeline inicia automaticamente.
- Executa build + testes
- Se aprovado → deploy em ambiente configurado.

Vantagens

Velocidade e eficiência

- Builds automáticos → tempo economizado.
- Deploys frequentes e confiáveis.
- Menos tempo gasto com tarefas manuais.

Vantagens

Qualidade e confiabilidade

- Testes automatizados em cada commit.
- Menor risco de bugs chegarem à produção.
- Feedback rápido para os desenvolvedores.

Colaboração e Padronização

- Todos seguem o mesmo pipeline → consistência.
- Facilita integração de novos membros.
- Reduz “funciona na minha máquina”.

Vantagens

Segurança e Escalabilidade

- Revisões e testes automáticos antes do deploy
- Ambientes reproduzíveis → fácil escalar.
- Logs e histórico centralizados no GitHub Actions.

GitHub Actions é uma plataforma de CI/CD (Integração Contínua e Entrega Contínua) integrada ao GitHub que permite automatizar, personalizar e executar workflows de desenvolvimento de software diretamente nos seus repositórios. Teve seu anúncio em 2018 e foi desenvolvido pela própria equipe do github.

Começando com CI/CD

No estudo de caso a seguir será demonstrado como criar e configurar um pipeline básico para integração contínua.

Também será abordado conceitos fundamentais para a utilização da ferramenta github actions.

Começando com CI/CD

Primeiros passos com GitHub Actions

- Criar / Clonar repositório que será utilizado
- Criar arquivos
- Criar pasta workflows e arquivo .yml
- Configurar arquivo .yml
- Testar e verificar resultados

Começando com CI/CD

Exercício prático

Crie um repositório no GitHub com um projeto Node.js que contenha uma função para verificar se um número é par ou ímpar. Configure CI usando GitHub Actions para rodar testes automatizados sempre que houver alterações na branch main.

Conclusão

Retomando os Pontos-Chave

- Desenvolvimento exige rapidez e qualidade
- Automação é essencial
- CI/CD aumenta confiança e eficiência

Conclusão

Encerramento

- CI/CD é também mudança cultural
- Pequenos passos já trazem ganhos
- Entregando software com confiança

Dúvidas?
Obrigado pela atenção!