Desenvolvimento Web II

Aula 06 - Integração e Entrega Contínua - CI/CD

Prof. Fabricio Bizotto

Instituto Federal Catarinense

fabricio.bizotto@ifc.edu.br

Ciência da Computação
27 de janeiro de 2024

Roteiro

Manifesto Ágil

- O Manifesto Ágil foi escrito em 2001 por 17 desenvolvedores de software que se reuniram para discutir métodos leves de desenvolvimento de software.
- O Manifesto Ágil é baseado em 4 valores e 12 princípios.
- Os 4 valores do Manifesto Ágil são:
  1. Indivíduos e interações mais que processos e ferramentas;
  2. Software em funcionamento mais que documentação abrangente;
  3. Colaboração com o cliente mais que negociação de contratos;
  4. Responder a mudanças mais que seguir um plano.

Integração Contínua - *Continuous Integration (CI)*

- CI é uma prática de desenvolvimento de software em que os desenvolvedores integram pequenos pedaços de código em um repositório compartilhado frequentemente, de preferência várias vezes ao dia.
- Cada integração é verificada por uma **build** automatizada (incluindo testes) para detectar erros de integração o mais rápido possível.
- Esse tipo de abordagem leva a uma redução significativa nos problemas de integração e permite que uma equipe desenvolva software coeso mais rapidamente.
- O CI pode ser falhar por: **builds** quebradas, testes quebrados, cobertura de testes insuficiente, etc.
- **Ferramentas**: Git, GitHub, GitLab, Jenkins, Circle CI, GitHub Actions, GoCD, Semaphore, Travis CI, etc.

Pipeline - Fluxo de Trabalho

1. **Controle de Versão**: Desenvolvedor faz **commit** no repositório compartilhado;
2. **Gatilho Automático**: Servidor de CI monitora o repositório e faz o **pull** do código;
3. **Compilação e Testes Automatizados**: Servidor de CI executa a **build** e os testes;
4. **Relatório**: Servidor de CI notifica o desenvolvedor sobre o resultado.

Integração Contínua

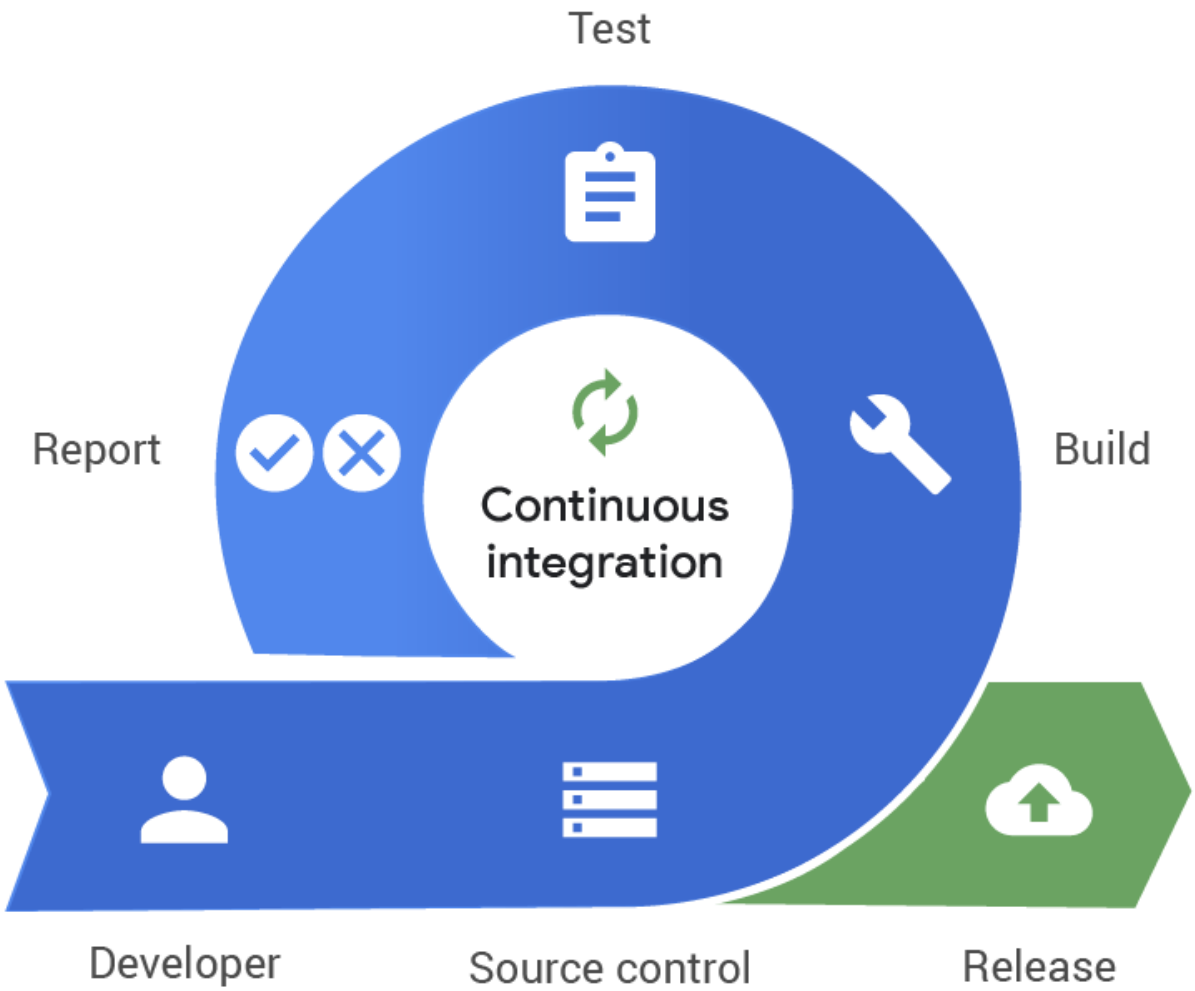

Figura: Integração Contínua

Entrega Contínua - *Continuous Delivery (CD)*

- CD é uma abordagem de engenharia de software na qual as equipes produzem software em ciclos curtos, garantindo que o software possa ser implantado de forma confiável a qualquer momento.
- CD visa garantir que um aplicativo esteja sempre no estado pronto para produção após passar com sucesso em testes automatizados e verificações de qualidade.
- O código é desenvolvido, testado e preparado para ser implantado automaticamente em um ambiente de produção a qualquer momento. No entanto, a decisão final de implantar no ambiente de produção ainda é manual e controlada pelo time de operações.

Deploy Contínuo - *Continuous Deployment*

- O **Deploy Contínuo** é uma extensão da Entrega Contínua que permite que cada alteração de código que passe com sucesso pelos testes automatizados seja implantada automaticamente em produção.
- **Ferramentas**: AWS, Azure, Google Cloud, Heroku, Digital Ocean, Surge, GitHub Pages, Netlify, Vercel, etc.

Pipeline de Integração e Entrega Contínua

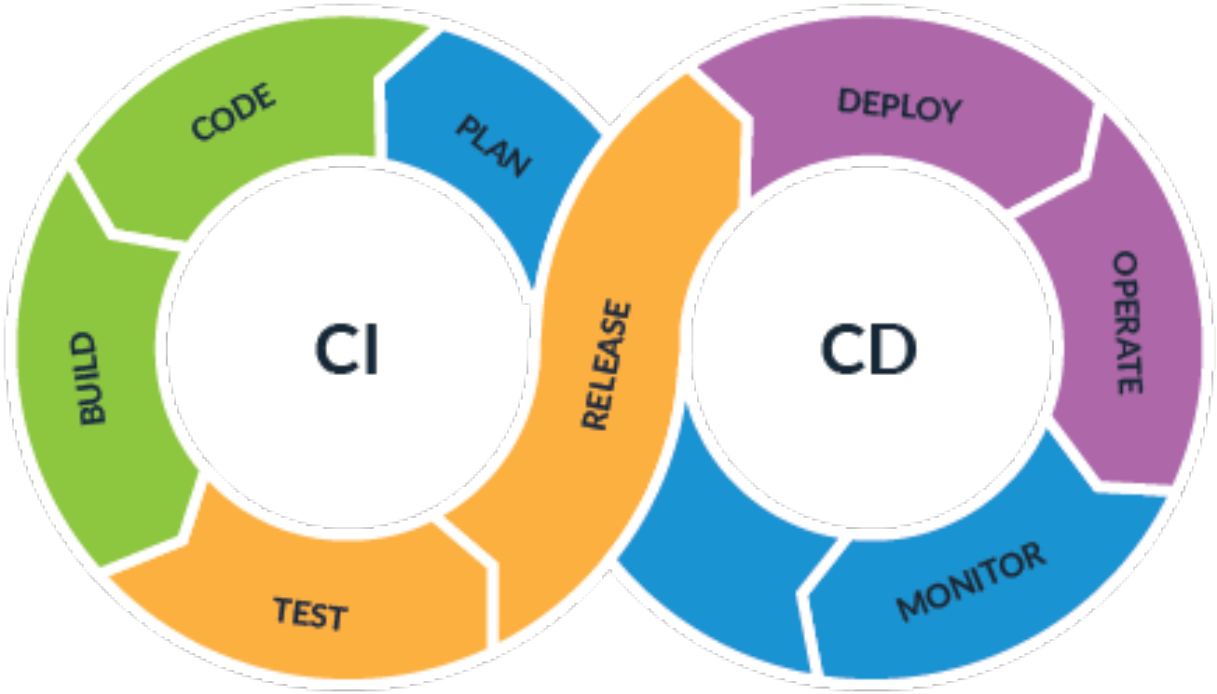

Figura: Pipeline de Integração e Entrega Contínua

Pipeline de Integração e Entrega Contínua

- O pipeline introduz monitoramento e automação para melhoria no processo de desenvolvimento de software, principalmente nas fases de integração e teste.
- Embora seja possível executar o pipeline manualmente, o objetivo é que ele seja executado automaticamente sempre que houver uma alteração no código.

Pipeline - Fluxo de Trabalho

As etapas típicas de um pipeline de CI/CD são:

1. **Teste**: Testes unitários, testes de integração, testes de aceitação, etc;
2. **Build**: Compilação, empacotamento, etc;
3. **Release**: Implantação em ambientes de homologação e produção;
4. **Monitoramento**: Monitoramento de desempenho, disponibilidade, etc.
5. **Feedback**: Notificação de falhas, relatórios de desempenho, etc.

Testes

- **Testes Unitários**: Testam uma unidade de código (método, classe, etc) de forma isolada.
- **Testes de Integração**: Testam a integração entre duas ou mais unidades de código.
- **Testes de Aceitação do Usuário (UAT)**: São testes realizados pelos usuários finais ou representantes de negócios para garantir que a aplicação atenda às expectativas e requisitos do usuário.
- **Testes de Regressão**: Testam se as alterações no código não quebram funcionalidades existentes.
- **Testes de Desempenho**: Testam o desempenho do sistema em diferentes condições de carga.
- **Smoke Tests**: Testes básicos para verificar se o sistema está funcionando. Para testas, por exemplo, se o sistema está no ar, se o banco de dados está acessível, etc.
- **Testes Funcionais**: Testam se o sistema atende aos requisitos funcionais. Para automatizar esses testes, é necessário simular a interação do usuário com o sistema.

Ferramentas

- **Ferramentas básicas**: Git, GitHub, Docker, Docker Hub;
- **CI/CD**: GitLab, Jenkins, Circle CI, GitHub Actions, GoCD, Semaphore, Travis CI;
- **Cloud**: AWS, Azure, Google Cloud, Heroku, Digital Ocean, Surge, GitHub Pages, Netlify, Vercel, etc.
- **Monitoramento**: New Relic, Datadog, AppDynamics, etc.
- **Testes**: Selenium, Cypress, Jest, JUnit, Vitest, etc.
- **Cobertura de Testes**: Istanbul, Jacoco, SonarQube, etc.
- **Análise de Código**: SonarQube, Codacy, Code Climate, etc.

Integração Contínua

Exemplo prático

Vamos ver um exemplo prático de CI/CD usando o GitHub Actions

https://github.com/fabricioifc/ci-cd-example

O que é Integração Contínua (CI)?

Resposta

a. Um processo que integra diferentes sistemas em uma única aplicação.
b. Um método para integrar continuamente novos recursos em uma aplicação.
c. Uma prática que visa integrar e testar automaticamente as alterações no código fonte.

O que é Integração Contínua (CI)?

Resposta

a. Um processo que integra diferentes sistemas em uma única aplicação.

b. Um método para integrar continuamente novos recursos em uma aplicação.

c. Uma prática que visa integrar e testar automaticamente as alterações no código fonte.

Qual é o principal objetivo da Integração Contínua?

Resposta

a. Garantir que as alterações no código sejam integradas e testadas frequentemente.
b. Reduzir o número de desenvolvedores na equipe.
c. Aumentar o tempo entre as versões da aplicação.

Qual é o principal objetivo da Integração Contínua?

Resposta

a. Garantir que as alterações no código sejam integradas e testadas frequentemente.
b. Reduzir o número de desenvolvedores na equipe.
c. Aumentar o tempo entre as versões da aplicação.

Qual é a diferença entre Integração Contínua e Entrega Contínua?

Resposta

a. Não há diferença, os termos são usados de forma intercambiável.

b. Entrega Contínua é apenas uma fase avançada da Integração Contínua.

c. Integração Contínua se refere à integração frequente de código, enquanto Entrega Contínua envolve a entrega automatizada de software funcional.

Qual é a diferença entre Integração Contínua e Entrega Contínua?

Resposta

a. Não há diferença, os termos são usados de forma intercambiável.

b. Entrega Contínua é apenas uma fase avançada da Integração Contínua.

c. Integração Contínua se refere à integração frequente de código, enquanto Entrega Contínua envolve a entrega automatizada de software funcional.

Qual é a importância dos testes automatizados na Integração Contínua?

Resposta

a. Não são necessários, pois a Integração Contínua já cobre todos os aspectos.
b. Garantir que as alterações no código não quebrem funcionalidades existentes.
c. Apenas agilizar o processo de integração, sem impacto na qualidade do software.

Qual é a importância dos testes automatizados na Integração Contínua?

Resposta

a. Não são necessários, pois a Integração Contínua já cobre todos os aspectos.

b. Garantir que as alterações no código não quebrem funcionalidades existentes.

c. Apenas agilizar o processo de integração, sem impacto na qualidade do software.

Quais benefícios podem ser obtidos ao implementar a Integração Contínua?

Resposta

a. Maior número de bugs.

b. Maior confusão entre os membros da equipe.

c. Detectar e corrigir problemas de integração mais cedo, facilitando o desenvolvimento contínuo e a entrega de software de alta qualidade.

Quais benefícios podem ser obtidos ao implementar a Integração Contínua?

Resposta

a. Maior número de bugs.

b. Maior confusão entre os membros da equipe.

c. Detectar e corrigir problemas de integração mais cedo, facilitando o desenvolvimento contínuo e a entrega de software de alta qualidade.

Experimentos

- **Experimento 1**
  Siga esse passo a passo para criar um repositório no GitHub e configurar o GitHub Actions para executar os testes e o deploy de forma automática no GitHub Pages.
- **Experimento 2**
  Faça alterações no código e observe o resultado no GitHub Actions.
- **Experimento 3**
  Faça alterações que quebrem os testes e observe o resultado no GitHub Actions.

Desafio

- Crie um novo projeto que faça uso de CI/CD e publique no GitHub.
- Use o GitHub Actions ou outra ferramenta de CI/CD.
- Use o GitHub Pages ou outra ferramenta de hospedagem.
- Crie um ambiente de homologação e um de produção.
- Faça uso de testes automatizados.