

An aerial photograph of a university campus. The image shows a large green lawn in the center, surrounded by various buildings, some with red roofs and others with white roofs. There are several trees and a winding path. A body of water is visible in the lower right corner. The text "WEB ACADEMY" is overlaid in large, bold, dark blue letters.

WEB ACADEMY

Integração Contínua

Daniel Augusto Nunes da Silva

Apresentação

Ementa

- **Entrega de software:** problemas, princípios e *pipelines*. **DevOps**. Controle de versões. **Integração contínua**. Boas práticas no uso de integração contínua. Desenvolvimento Baseado no Trunk (TBD). Servidores de integração contínua. **GitHub Actions**. Fluxo de trabalho no GitHub. **Implantação contínua**. Entrega contínua.

Objetivos

- **Geral:** Capacitar o aluno na utilização de técnicas de **integração e implantação contínua** em projetos de software, utilizando ferramentas para **automatizar o processo de entregar software**.
- **Específicos:**
 - Discutir os problemas relacionados ao processo de entrega de software;
 - Relacionar os conceitos integração e implantação contínua, com ênfase no conceito de DevOps;
 - Apresentar técnicas e ferramentas para automatizar tarefas relacionadas a integração e implantação contínua.
 - Construir um pipeline de integração e implantação contínua em um projeto de software.

Conteúdo programático

Introdução

- O problema de entregar software;
- Princípios para entrega de software;
- DevOps;
- Pipeline para entrega de software.

Integração Contínua

- Introdução a Integração Contínua;
- Boas práticas no uso de CI;
- Desenvolvimento Baseado no Trunk (TBD);
- GitHub Actions;
- Fluxos de trabalho no GitHub Actions;
- Abordagens para implementar CI;
- Quando não usar CI?

Implantação Contínua

- Introdução a Implantação Contínua;
- Fluxo de trabalho;
- Vantagens;
- Criação de workflow no GitHub Actions.

Sites de referência

- Software Delivery Guide (Martin Fowler).
 - <https://martinfowler.com/delivery.html>
- GitHub Docs: GitHub Actions.
 - <https://docs.github.com/pt/actions>
- Docker Docs.
 - <https://docs.docker.com/get-started/overview/>

Contato



<https://linkme.bio/danielnsilva/>

Introdução

O problema de entregar software

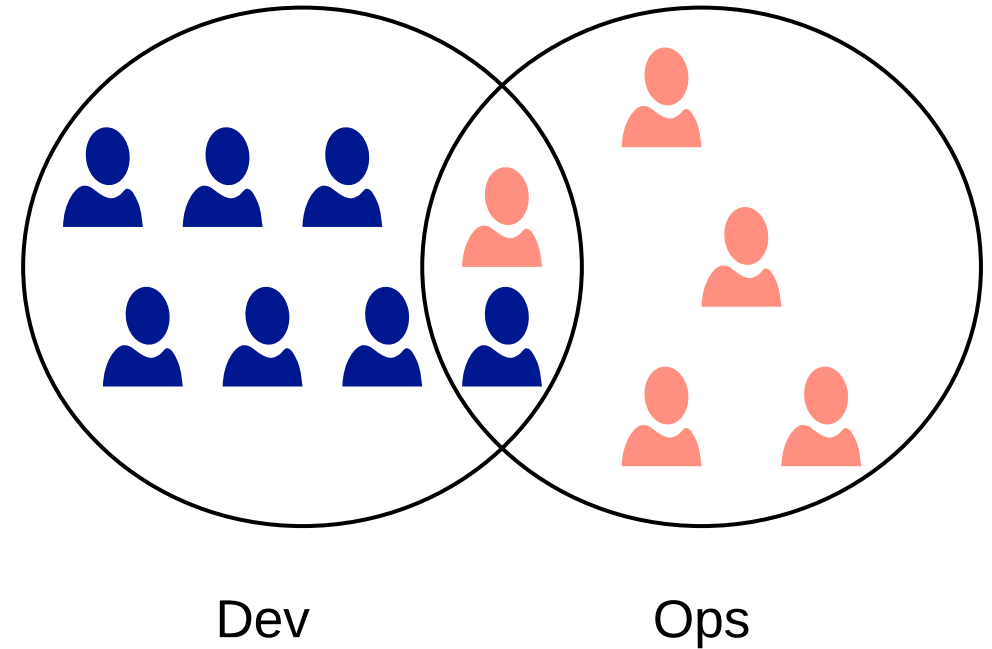
- **Colocar um software em produção pode implicar em muitas dificuldades:** problemas de compilação, testes, configuração de ambientes, etc.
- **Todo modelo de desenvolvimento de software descreve uma etapa na qual o software entra em operação** e, ainda, recebe novas versões durante seu ciclo de vida.
- Até meados dos anos 2000, normalmente as metodologias estavam concentradas em técnicas de gerenciamento de projetos e requisitos, além de práticas de desenvolvimento e testes.
- Essas boas práticas aplicadas ao processo de desenvolvimento de software devem envolver uma **maneira eficiente de entregar software**.

Princípios para entrega de software

- Crie um **processo repetível e confiável** para entrega de software.
- **Automatize** tudo que for possível.
- Mantenha tudo em um **sistema de controle de versões**.
- Se uma tarefa pode causar problemas, deve ser executada com **mais frequência** e o quanto antes.
- “**Concluído**” significa pronto para entrega.
- **Todos são responsáveis** pela entrega do software.

DevOps

- Historicamente, tarefas de **desenvolvimento** e de **operações** (infraestrutura) são separadas em equipes diferentes.
- Essa divisão dificulta o processo de entregar software com rapidez e qualidade.
- A proposta do **DevOps** é integrar as duas áreas, otimizando o processo de entregar software (colocar em produção).



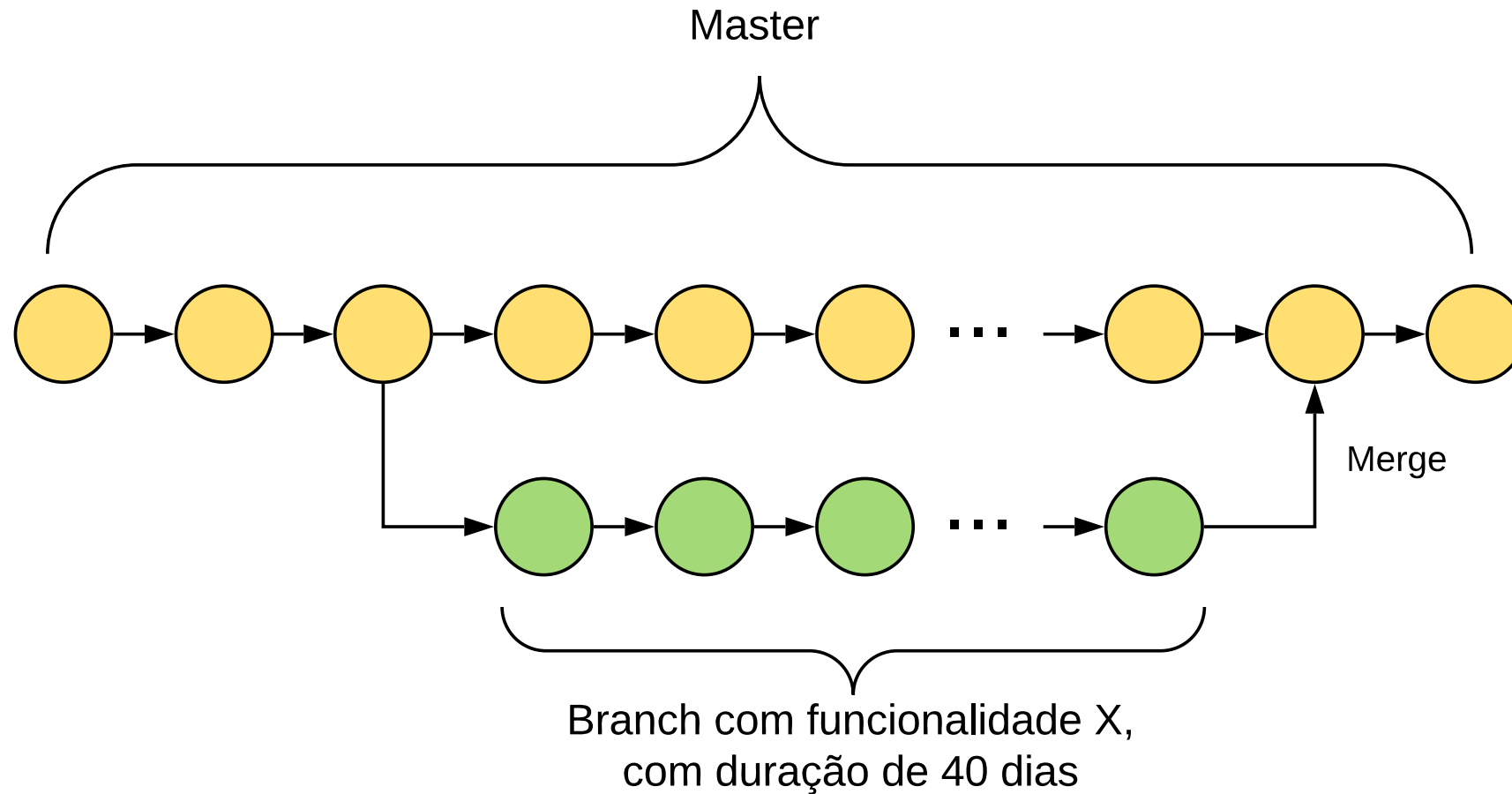
Fonte: VALENTE, 2020.

Pipeline para entrega de software



Integração Contínua

Integração Contínua



Fonte: VALENTE, 2020.

Integração Contínua

- **Integração Contínua** (*Continuous Integration* ou **CI**) é uma prática proposta pela metodologia ágil *Extreme Programming* (XP).
- Princípio motivador: **se uma tarefa pode causar problemas, não podemos deixar que ela acumule.**
 - **Grandes integrações de código** podem gerar mais conflitos.
 - Se **integrar o código de forma frequente**, isto é, contínua, as integrações serão pequenas e irão **gerar menos conflitos**.
- Devemos **quebrar uma tarefa em subtarefas** que possam ser **realizadas de forma frequente**.

Boas Práticas para Uso de CI

- **Build Automatizado**

- É importante que **seja o mais rápido possível**, pois com integração contínua ele será executado com frequência.

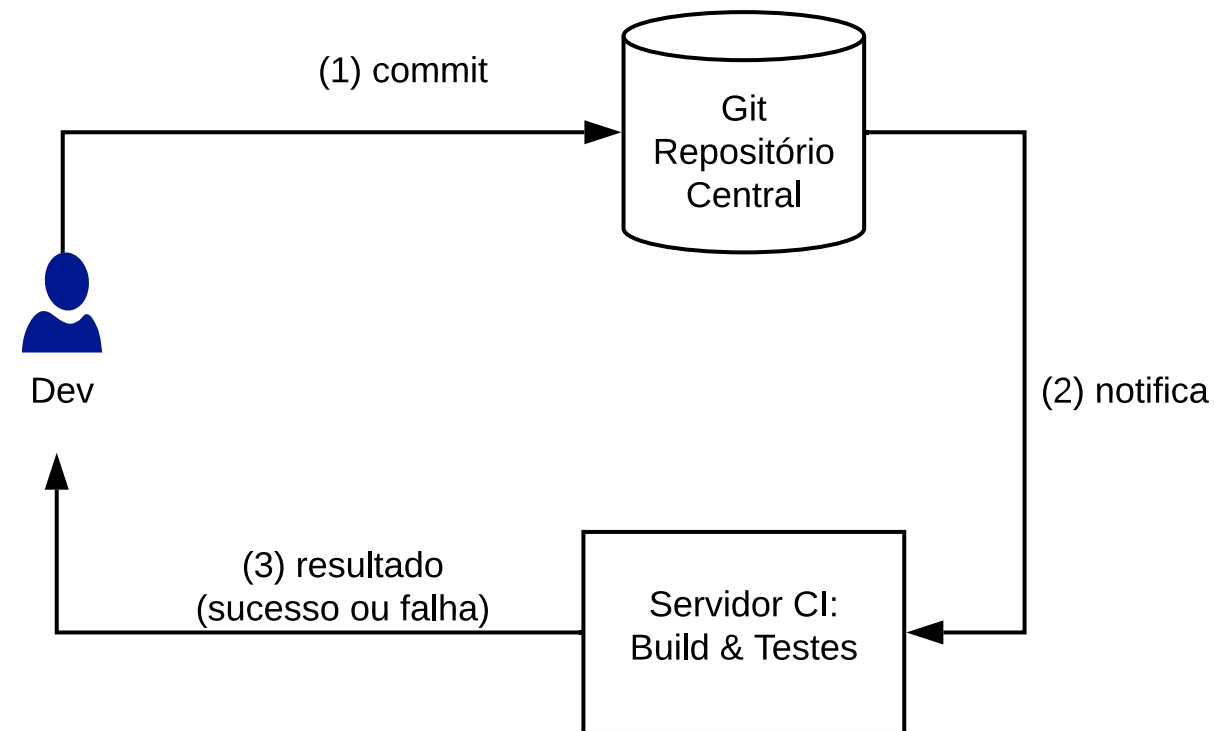
- **Testes Automatizados**

- Além de garantir que o software compila sem erros após cada novo commit, é importante garantir também que ele **continua com o comportamento esperado**.

Boas Práticas para Uso de CI

▪ Servidores de Integração Contínua

- Os builds e testes automatizados devem ser executados com frequência.
- Após um novo commit, o sistema de controle de versões avisa o servidor de CI, que clona o repositório e executa um build completo, bem como roda todos os testes.
- Após a execução do build e dos testes, o servidor notifica o usuário.



Fonte: VALENTE, 2020.

Boas Práticas para Uso de CI

- **Desenvolvimento Baseado no Trunk**

- CI é compatível com o uso de branches desde que sejam integrados de forma frequente no master (todo dia).
- Quando migram para CI, é comum que as organizações usem também desenvolvimento baseado no trunk (**TBD** – *trunk based development*).
- **Não existem mais branches** para implementação de novas funcionalidades ou para correção de bugs.
- **Todo desenvolvimento ocorre no branch principal**, também conhecido como trunk ou master (main).

GitHub Actions

- Ferramenta integrada ao GitHub para automatizar a execução de fluxos de trabalho de desenvolvimento de software, dando suporte a utilização CI.
- **Uma ação (Github Action) é um aplicativo** que executa uma tarefa complexa (envolve vários passos), mas que é repetitiva.
- É possível combinar ações personalizadas e ações já disponíveis na plataforma.
- **O GitHub faz o papel de Servidor de CI**, fornecendo uma máquina virtual para executar as ações.
- O arquivos de configuração utilizam a linguagem YAML e devem ser armazenados no diretório **`.github/workflows/`** em cada projeto.

Linguagem YAML

- **YAML** é uma **linguagem de serialização de dados** muito usada em arquivos de configuração, e projetada com **ênfase na legibilidade**.
- Assim como JSON, é baseada em pares de **chave e valor**;
- É a linguagem padrão de ferramentas como GitHub Actions, Docker Compose e Ansible.

```
# Comentário
nome: "Exemplo de YAML"
professor:
  nome: "Daniel"
  disciplinas:
    - nome: "Frameworks Back-end"
      carga_horaria: 15
    - nome: "Frameworks Front-end"
      carga_horaria: 15
descricao: |
  Este é um exemplo de valor
  multilinha no YAML.
```


GitHub Actions Workflow

```
1. name: Primeiro Workflow
2. on:
3.   push: ← - - - - - Sempre que um novo arquivo
4. jobs:                                     é enviado ao repositório.
5.   primeiro-job:
6.     runs-on: ubuntu-latest ← - - - - - Provisão de uma máquina virtual
7.     steps:                                     para funcionar como Servidor CI.
8.     - -> - run: echo "Primeiro passo"
9.     - name: Checkout
10.    - uses: actions/checkout@v4 ← - - - - - O action "checkout" faz o checkout
11.    - name: Lista arquivos                                     do branch atual no Servidor CI.
12.    - run: ls ${{ github.workspace }}
13.    - name: Status
14.    - run: echo "Status ${{ job.status }}"

- O hífen indica um passo, que pode agrupar diferentes configurações.
```

Continua...

Referências

- HUMBLE, Jez; FARLEY, David. **Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation**. 1. ed. [S. I.]: Pearson Addison-Wesley, 2010. 512 p.
- DUVALL, Paul M. **Continuous Integration: Improving Software Quality and Reducing Risk**. 1. ed. [S. I.]: Pearson Addison-Wesley, 2007. 336 p.
- GITHUB (ed.). **GitHub Docs: GitHub Actions**. [S. I.], 2025. Disponível em: <https://docs.github.com/pt/actions>.
- MARCO TULIO VALENTE. **Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade**, 2020. Disponível em: <https://engsoftmoderna.info/>.
- SOMMERVILLE, Ian. **Engenharia de Software**. 9. ed. São Paulo: Pearson Addison-Wesley, 2011.