Princípios de boas práticas de versionamento de código

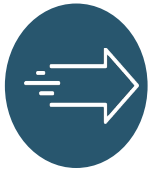
Aula 3

Professor Gustavo Dias

Aplicar fluxo de trabalho utilizando Git

O Git oferece **flexibilidade** para equipes adotarem diferentes fluxos de trabalho. Esses fluxos de trabalho ditam como as alterações são feitas, revisadas e integradas no projeto.

Aplicar fluxos de trabalho com Git.



Introduzir o conceito do GitHub e suas possibilidades no contexto da comunidade de desenvolvimento.

Conhecer tags e como podem ser utilizadas.

Git Flow

Release Branches

- Preparam o código para um **novo lançamento**.
- **Permitem correções** de bugs e preparação para o lançamento.

Hotfix Branches

- Usadas para **correções rápidas** diretamente na versão de produção.

Git Flow

Vamos relembrar!
Principais branches utilizados no Git e práticas de utilização.

- **Branch main (ou master):** Representa a **versão estável** e pronta para produção do software.
- **Branch develop:** Onde o desenvolvimento diário ocorre. Uma vez que os recursos estejam estáveis, são **mesclados de volta** aqui.
- **Feature Branches:** Criados a partir da *branch develop* e são usados para desenvolver **novas funcionalidades**.

Git Flow

Vamos conhecer algumas de suas características.

- Fluxo de trabalho mais **simples** e **linear** comparado ao Git Flow.
- Todo trabalho ocorre em **branches de funcionalidades**.
- Quando uma funcionalidade ou correção é finalizada, um **pull request** é aberto.
- Após revisão e testes, o *pull request* é **mesclado** no **branch principal**.

Vídeo

Assista ao vídeo “O que são Git e GitHub? ” no YouTube

ALURA. O que são Git e GitHub? #HipstersPontoTube. Disponível em: https://www.youtube.com/watch?v=P4BNi_yPehc. Acesso em: 20 dez. 2023.

© Getty Images

Exposição

Secretaria da Educação SÃO PAULO GOVERNO DO ESTADO

Exposição

Características *Feature Branching* e *Trunk Based Development*

Feature Branching	***Trunk Based Development***
Cada nova funcionalidade ou correção é desenvolvida em **seu próprio branch**.	Os desenvolvedores trabalham em **cópias curtas** do *branch* principal, chamada de "tronco".
Permite que vários desenvolvedores **trabalhem** em diferentes funcionalidades **simultaneamente** sem interferir uns nos outros.	Os *commits* são feitos regularmente para este tronco, garantindo que o código esteja sempre em um estado **"pronto para produção"**.
Uma vez que a funcionalidade esteja **completa e testada**, ela **é mesclada** de volta ao *branch* principal.	*Branches* de funcionalidades são evitados ou são de **vida muito curta**.
É **flexível** para a maioria dos projetos e é particularmente benéfico em equipes que trabalham em várias funcionalidades ou correções simultaneamente.	É indicado para times que usam **integração contínua** e querem prevenir conflitos ao unir *branches* de funcionalidades extensas. Ele foca em agilidade e simplicidade.

Exposição

Tags no conceito de versionamento de código

No desenvolvimento de software, é essencial ter **pontos de referência claros** no histórico de desenvolvimento, principalmente quando se trata de manter e lançar diferentes versões de um software.

É aqui que as **“tags”** desempenham um papel fundamental.

Tome nota

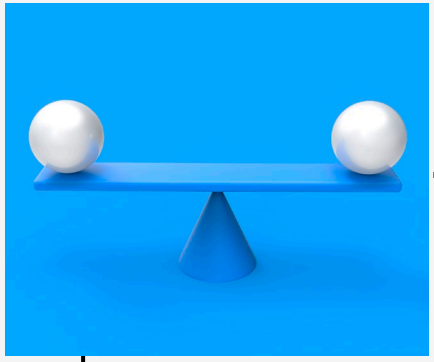
No contexto do Git, uma tag atua como um marcador para um commit específico. Tags são permanentes e indicam um local específico no histórico do repositório. Elas só são alteradas se forem explicitamente atualizadas.

Exposição

Por que usar tags no conceito de versionamento de código:

Versões estáveis

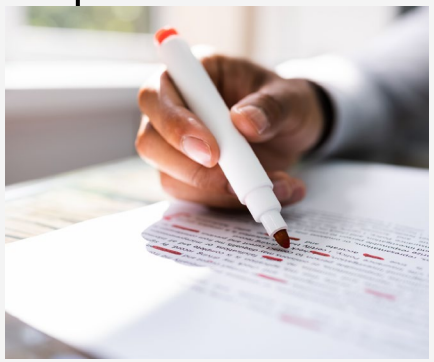
Em um software, a tag marca o commit, indicando o histórico da versão lançada.



© Getty Images

Facilidade de acesso

Permite revisitar ou corrigir uma versão anterior.



© Getty Images

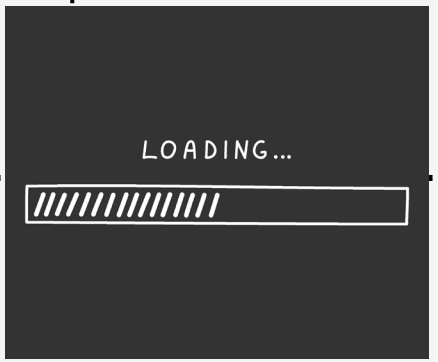
Documentação

Acompanha mensagem com uma nota acerca da versão ou motivo de criação daquela tag.

© Getty Images

Distribuição

Disponibiliza downloads das versões de seu software baseadas nas tags.



© Getty Images

Vamos fazer um **quiz**

Registro

TESTE SEU CONHECIMENTO:

Maria está começando a contribuir para um projeto *open-source*. Ela realizou um *fork* do repositório oficial e deseja enviar suas mudanças para serem avaliadas.

Relato fictício elaborado especialmente para o curso

Qual fluxo de trabalho Maria está usando no projeto *open-source*?

Registro

Vamos fazer um **quiz**

- **Fluxo de trabalho centralizado**
- **Fluxo de trabalho de *Feature Branching***
- ***Git flow***
- ***Forking workflow***

Registro

Vamos fazer um **quiz**

Feedback da atividade

✗ **Fluxo de trabalho centralizado**

RESPOSTA ERRADA! No fluxo de trabalho centralizado, todos os desenvolvedores fazem *push* diretamente para o *branch* principal.

✗ **Fluxo de trabalho de *Feature Branching***

RESPOSTA ERRADA! No fluxo de trabalho de *Feature Branching*, os desenvolvedores criam *branches* para cada nova funcionalidade ou correção.

✓ ***Forking workflow***

RESPOSTA CORRETA! No *forking workflow*, cada desenvolvedor tem seu próprio repositório público, fazendo contribuições por meio de "forks". Esse é o fluxo típico para projetos open-source.

✗ ***Git flow***

RESPOSTA ERRADA! Git flow é uma extensão do fluxo de *feature branching* com uma estrutura mais definida para funcionalidades, lançamentos e correções.

Vamos fazer um **quiz**

TESTE SEU CONHECIMENTO:

Rodrigo lançou uma nova versão estável do software e quer marcar essa versão específica no histórico do Git para fácil referência no futuro.

Relato fictício elaborado especialmente para o curso.

Qual recurso do Git, Rodrigo deve usar para atingir seu objetivo?

Vamos fazer um **quiz**

- ***Branch***
- ***Commit***
- ***Tag***
- ***Checkout***

Vamos fazer um **quiz**

Feedback da atividade

✗ **Branch**

RESPOSTA ERRADA! *Branch* é uma linha de desenvolvimento paralela. Não seria a melhor opção para marcar versões estáveis do software.

✗ **Commit**

RESPOSTA ERRADA! *Commit* é uma alteração individual no código. Não serve para marcar versões específicas de forma referenciável.

✓ **Tag**

RESPOSTA CORRETA! *Tag* é usada para marcar pontos específicos no histórico do Git como uma versão lançada. É a opção correta para o objetivo de Rodrigo.

✗ **Checkout**

RESPOSTA ERRADA! *Checkout* é usado para mudar de *branch* ou voltar para um *commit* anterior. Não é usado para marcar versões.

Vamos fazer um **quiz**

TESTE SEU CONHECIMENTO:

O que representa uma “tag” em um sistema de controle de versão como o Git ao lidar com versões estáveis de software?

Registro

Secretaria da Educação SÃO PAULO GOVERNO DO ESTADO

Vamos fazer um **quiz**

Uma tag marca a última mudança feita no código antes de um novo desenvolvedor se juntar à equipe.

Um indicador de que o repositório de código foi arquivado e não receberá mais atualizações.

Uma referência a um ponto específico no histórico do repositório que corresponde à liberação de uma versão estável do software.

Uma anotação que indica o início do desenvolvimento de uma nova funcionalidade no projeto.

Vamos fazer um **quiz**

Feedback da atividade

Alternativa 1

RESPOSTA ERRADA! Uma tag é usada para marcar versões estáveis, e não está relacionada à adição de membros à equipe de desenvolvimento.

Alternativa 2

RESPOSTA ERRADA! Uma tag serve para marcar versões estáveis para referência futura, e não indica que o repositório foi arquivado.

Alternativa 3

RESPOSTA CORRETA! Em sistemas de controle de versão, uma tag é comumente usada para marcar um commit específico que representa uma versão estável do software, facilitando o acesso a esse estado do código no futuro.

Alternativa 4

RESPOSTA ERRADA! Tags são usadas para marcar lançamentos de versões estáveis e não estão diretamente relacionadas ao início do desenvolvimento de novas funcionalidades.