Versionamento de código

GIT II

Professor - Renato Naumann

O que vamos aprender hoje?

- Como reverter um **commit**
- O que é e como trabalhar com **branches**
- Possíveis operações de um branch
- O que é e como fazer um **merge**
- Como trabalhar com tags e a sua importância

Como podemos reverter um commit?

Pode acontecer casos em que vamos precisar voltar a versão da nossa aplicação para um momento específico. Isso é chamado de **reverter commit**.
Lembram que vimos que quando damos o comando **git log** ele aparece uma chave de commit?

Essa chave é chamada de **ID de commit**.
Pelas mensagens do commit é possível identificar qual o commit que queremos voltar a versão, com o ID de commit em mãos é possível voltar.

O comando para isso é: **git reset --hard "Id do Commit"**

1

Vamos reverter um commit?

Abra o terminal e vamos fazer juntos

O que é e como trabalhar com branches

Um branch no git, é um ponteiro para as alterações feitas nos arquivos do projeto.
É útil em situações nas quais você deseja adicionar um novo recurso ou corrigir um erro, gerando um novo branch garantindo que o código instável não seja mesclado nos arquivos do projeto principal.
Depois de concluir a atualização dos códigos do branch, você pode mesclar o branch com o principal, geralmente chamado de **master**.

Possíveis operações de um branch

Listar

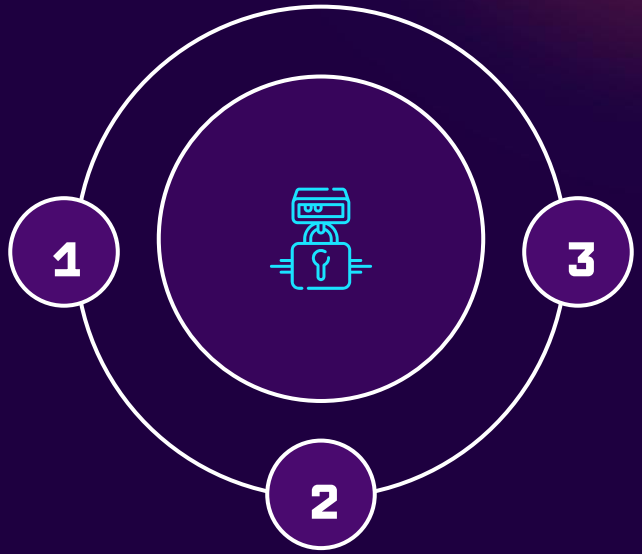
Criar

Acessar

Renomear

Remover

Comandos para listar branches



Listar local

git branch --list

Listar ambos

git branch -a

Listar remotos

git branch -r

Criar novo branch

Criar

git branch <nome_do_branch>

Criar e acessar

git checkout -b <nome_do_branch>

Criar baseado em um remoto

git checkout -b <nome_branch> <branch_remoto>

Acessar um branch

Acessar branch

```
git checkout <nome_do_branch>
```

Renomear um branch

Dentro do branch

git branch -m <novo_nome>

Fora do branch

git branch -m <antigo> <novo>

Remover um branch

Remoção padrão

`git branch -d <nome_do_branch>`

Remoção Forçada

`git branch -D <nome_do_branch>`

2

Vamos ver na prática?

Abram o terminal e vamos testar todos os comandos?

O que é um merge?

O **git merge** serve para mesclar alterações de dois branches diferentes.
Sempre ao executar o comando **git merge** você deve estar no **branch de origem** e trazer o **branch destino**. Então, sempre antes de executar o comando, certifique se que está no branch correto.
O comando git merge trás as informações, ele não envia.

O comando é: **git merge <branch-origem>**

3

Vamos ver na prática?

Abram o terminal e vamos fazer juntos

O que é o Git Flow?

Git flow é um modelo de organização de branches, isso significa que o **Git Flow** estabelece algumas regras de nomenclaturas para tipos de branches enquanto, ao mesmo tempo, define o que cada tipo de branch faz.

Por que utilizar o Git Flow?

É muito comum vermos pessoas utilizando apenas um branch para fazer commits em projetos pessoais. Isto não é errado, é muito tranquilo de se controlar tudo em um branch quando se está desenvolvendo sozinho, mas o cenário muda bastante quando temos que interagir com mais desenvolvedores em um projeto.
Nessas horas é de suma importância que se tenha total controle do que está sendo produzido por sua equipe, onde, ao mesmo tempo são corrigidos falhas, implementado novas funcionalidades e ter o seu código de produção com total funcionamento.
É aí que o fluxo de git flow nos ajuda.

Principais branches do Git Flow

1

Master

É o branch que contém código em nível de produção, ou seja, o código mais maduro existente na sua aplicação.

2

Develop

É o branch que contém código em nível preparatório para o próximo deploy. Quando features são terminadas, elas são mergeadas com o branch develop

3

Feature/*

São branches no qual são desenvolvidos recursos novos para o projeto em questão.

4

Hotfix/*

São branches no qual são realizadas correções de bugs críticos encontrados em ambiente de produção, e que por isso são criadas a partir da branch master

4

Vamos ver na prática?

Abram o terminal e vamos fazer juntos

O que são Git Tags?

São etiquetas que demarcam um ponto (commit) que representa alguma mudança significativa no seu código, ou seja, uma versão (ou release) do seu projeto.

Comando	Descrição
Criar	git tag -a <name> -m <message>
Listar	git tag
Subir remoto	git push origin v1.0
Exclusão local	git tag -d v1.0
Exclusão remota	git push --delete origin v1.0

5

Vamos ver na prática?

Abram o terminal e vamos fazer juntos

OBRIGADO!

Duvidas?
Nosso canal oficial de comunicação é o Slack.

academico@geradordedevs.com.br
https://www.geradordedevs.com.br