

O que são branches?

terça-feira, 20 de junho de 2023

15:12

- As branches são ramificações feitas através de outros desenvolvedores para trabalhar no projeto sem que utilize a branch master/main
- Nunca deve ser utilizada a branch master/main por que se feita uma modificação definitiva na master e for perdida, é mais complicado de se recuperar
- Branchs são as divisões do projeto em versões diferentes
- Após a finalização do projeto as branches são unidas para ter o código-fonte final (Merge)
- Branch dev - features mais novas adicionadas nessa branch
- Branch staged - é a branch onde entrarão em testes

Criando uma branch - git branch

terça-feira, 27 de junho de 2023

14:26

- Visualizando branches disponíveis - **git branch**
- Criando branches - **git branch nome**
- Quando criamos uma branch ela parte de qual branch você criou.
 - **Ex:** Criamos uma branch dev a partir da main, a branch dev vai ter tudo que a branch main tem e o que você adicionar nessa nova branch.

Deletando uma branch - git branch -d

terça-feira, 27 de junho de 2023

14:35

- Utiliza-se a flag **-d** ou **--delete**
- Deve-se usar o comando `git branch -d nome`
- É pouco usado, pois serve de histórico
- Usado geralmente quando criamos errado a branch

```
PS D:\github_teste\repo_teste> git branch -d master
Deleted branch master (was 0169740).
PS D:\github_teste\repo_teste> git branch master
PS D:\github_teste\repo_teste> git branch --delete master
```

Mudando de branch - git checkout nomebranch

terça-feira, 27 de junho de 2023

14:39

- Utiliza-se a flag **-b**
- Também é utilizado para dispensar mudanças de um arquivo
- As alterações que não foram commitadas vão juntas para essa nova branch
- O comando **git checkout -b nomebranch** - cria uma nova branch e entra nela

Unindo branches - git merge nomebranch

terça-feira, 27 de junho de 2023

14:56

- Deve-se usar na branch que queremos adicionar o conteúdo
- Utiliza-se o comando **git merge <nome>**
- Também pode ser utilizado para atualizar uma branch atrasada

Trazendo conteúdo de uma branch master para a main

```
PS D:\github_teste\repo_teste> git merge master
Merge made by the 'ort' strategy.
 oi.txt | 1 +
 2 files changed, 2 insertions(+)
 create mode 100644 alow.txt
 create mode 100644 oi.txt
PS D:\github_teste\repo_teste> git status
On branch main
```

Utilizando uma stash - git stash

terça-feira, 27 de junho de 2023

15:14

- Funciona como se estivesse jogando o código no lixo, ou seja as modificações feitas serem salvas juntos com o **git stash**
- A branch após o stash volta para o sua versão original do repositório remoto
- Os commits também são perdidos

```
● PS D:\github_teste\repo_teste> git stash
● Saved working directory and index state WIP on master: 80c030f Merge branch 'master'
○ PS D:\github_teste\repo_teste> █
```

Recuperando um stash - git stash apply

terça-feira, 27 de junho de 2023

15:39

- Deve-se usar o comando **git stash list** para verificar a lista de stash
- Para recuperar devemos utilizar o comando **git stash apply numerostash**
- Para verificar no terminal as modificações feitas nessa stash podemos usar o comando
git stash show -p numerostash

Recuperando stash 5

```
● PS D:\github_teste\repo_teste> git stash list
stash@{0}: WIP on nome: 80c030f Merge branch 'master'
● stash@{1}: WIP on nome: 80c030f Merge branch 'master'
stash@{2}: WIP on nome: 80c030f Merge branch 'master'
stash@{3}: WIP on nome: 80c030f Merge branch 'master'
stash@{4}: WIP on master: 80c030f Merge branch 'master'
stash@{5}: WIP on master: 80c030f Merge branch 'master'
PS D:\github_teste\repo_teste> git stash apply 5
● On branch nome
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   nome.txt

no changes added to commit (use "git add" and/or "git commit -a")
○ PS D:\github_teste\repo_teste> █
```

Removendo as stash- git stash clear ou git stash drop numerostash

terça-feira, 27 de junho de 2023

15:44

- Para excluir todas as stash devemos usar o comando **git stash clear**
- Para excluir uma stash específica devemos usar o comando **git stash drop numerostash**

```
PS D:\github_teste\repo_teste> git stash list
stash@{0}: WIP on main: 086bb66 Criando arquivo
PS D:\github_teste\repo_teste> git stash drop 0
● Dropped refs/stash@{0} (0a6b23e736d5d5099f8a8a043dcf0a79fca5580f)
● PS D:\github_teste\repo_teste> git stash list
```


Criando tags - `git tag -a <nome> -m "<msg>"`

terça-feira, 27 de junho de 2023

16:07

- Serve para criar tags e salvar algo na branch
- Diferente do stash a tag serve como um checkpoint da branch
- Utilizado para demarcar estágios do desenvolvimento de algum recurso, funciona como se fosse um ponto de recuperação
- Salva em versões
- Elas funcionam como branches de um branches praticamente

Criando uma tag

```
PS D:\github_teste\repo_teste> git tag -a versão_4.0 -m "Criando versão 4.0"
PS D:\github_teste\repo_teste> git tag
versão_1.0
versão_2.0
versão_3.0
versão_4.0
```

Alterando entre tags - git show ou git checkout

terça-feira, 27 de junho de 2023

16:19

- Usamos o comando **git show nome** - para ver as modificações da tag
- Usamos o comando **git checkout nome** - para alterar entre as tags

Alterando entre tags

```
PS D:\github_teste\repo_teste> git checkout versão_1.0
HEAD is now at 7986eac Criando arquivo
● PS D:\github_teste\repo_teste> git tag
● versão_1.0
  versão_2.0
  versão_3.0
  versão_4.0
○ PS D:\github_teste\repo_teste> |
```

Enviando tags ao repositório - `git push origin nome`

terça-feira, 27 de junho de 2023

16:28

- As tags podem ser compartilhadas entre outros devs, quando enviada para o repositório online
- Deve-se usar o comando **`git push origin <nometag>`**
- Para enviar todas as tags devemos usar **`git push origin --tags`**