



Versionamento, git e github

Projeto de Software

Fabio Moraes



Versionamento

- Consiste em gerenciar / controlar diferentes versões dos arquivos que compõem o sistema desenvolvido
 - Código, configurações, dados, imagens, etc
- Permite saber quando mudanças foram realizadas, por quem e por quais motivos
 - Melhora a organização e segurança do processo



Versionamento

- Permite recuperar facilmente versões passadas dos arquivos
 - Para acessar versões de releases anteriores do sistema
 - Em caso de falhas ou erros inseridos no código
- As principais ferramentas são o sistema **git** e o serviço **github**
 - github é um serviço web baseado no sistema git



Primeiros passos

- Crie um novo repositório a partir de um diretório local
 - O repositório só existe localmente e se for removido tudo se perde

```
> git init
```

- Também é possível criar uma cópia local de um repo remoto

```
> git clone <endereço do repo remoto>
```



Repositórios locais

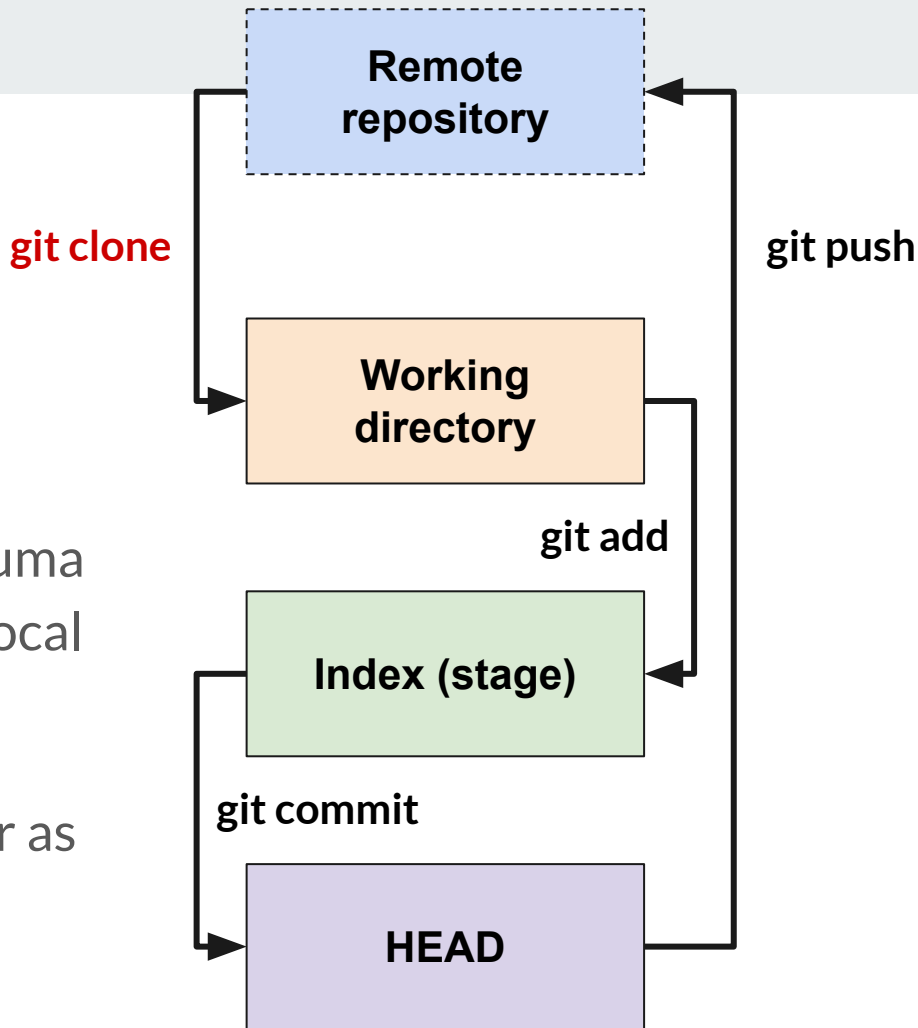
- Uma vez com um repositório local configurado o git mantém três estruturas em árvore para gerenciar mudanças
 - **Working directory:** contém os arquivos atuais em modificação
 - **Index (stage):** área que mantém mudanças temporárias
 - **HEAD:** aponta para a última modificação confirmada (commit)

Funcionamento básico

Ao **clonar** um repositório remoto uma cópia é criada em um repositório local

- No **working directory**

Nesse espaço o usuário vai realizar as mudanças desejadas



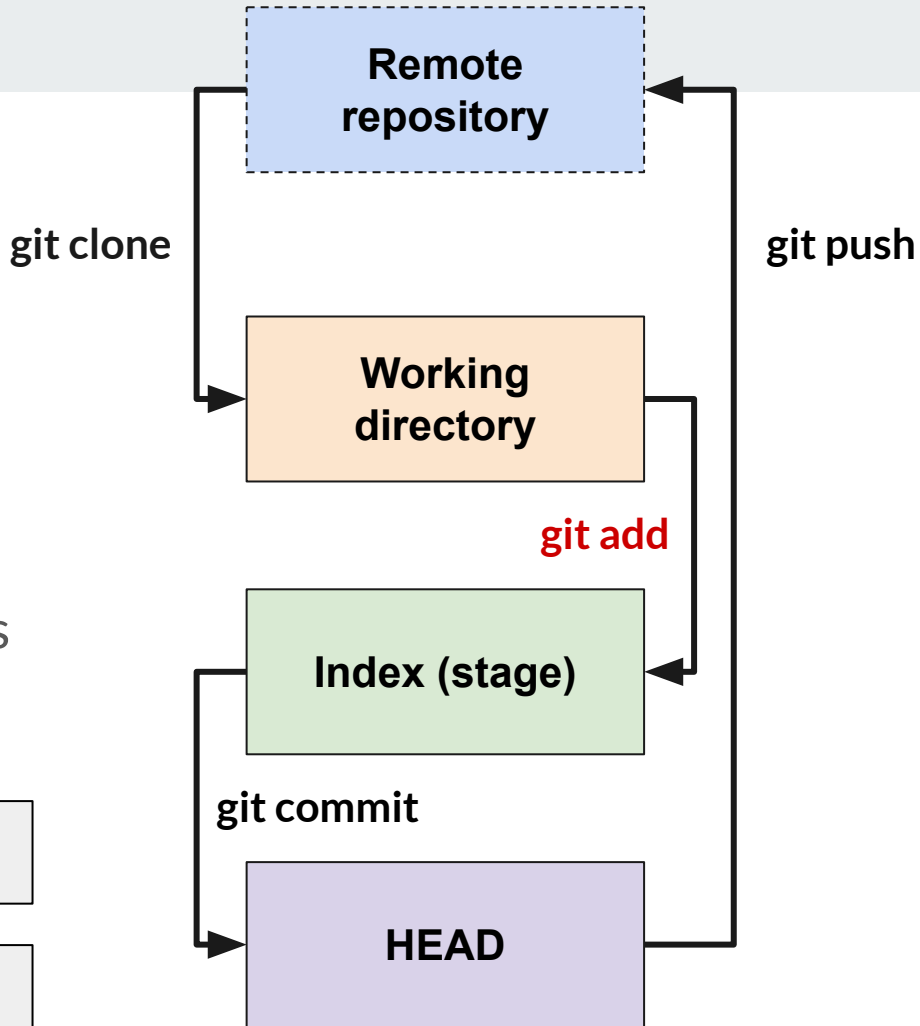
Funcionamento básico

Mudanças temporárias podem ser salvas antes de serem confirmadas

- No **index**

```
> git add <arquivo modificado>
```

```
> git add *
```

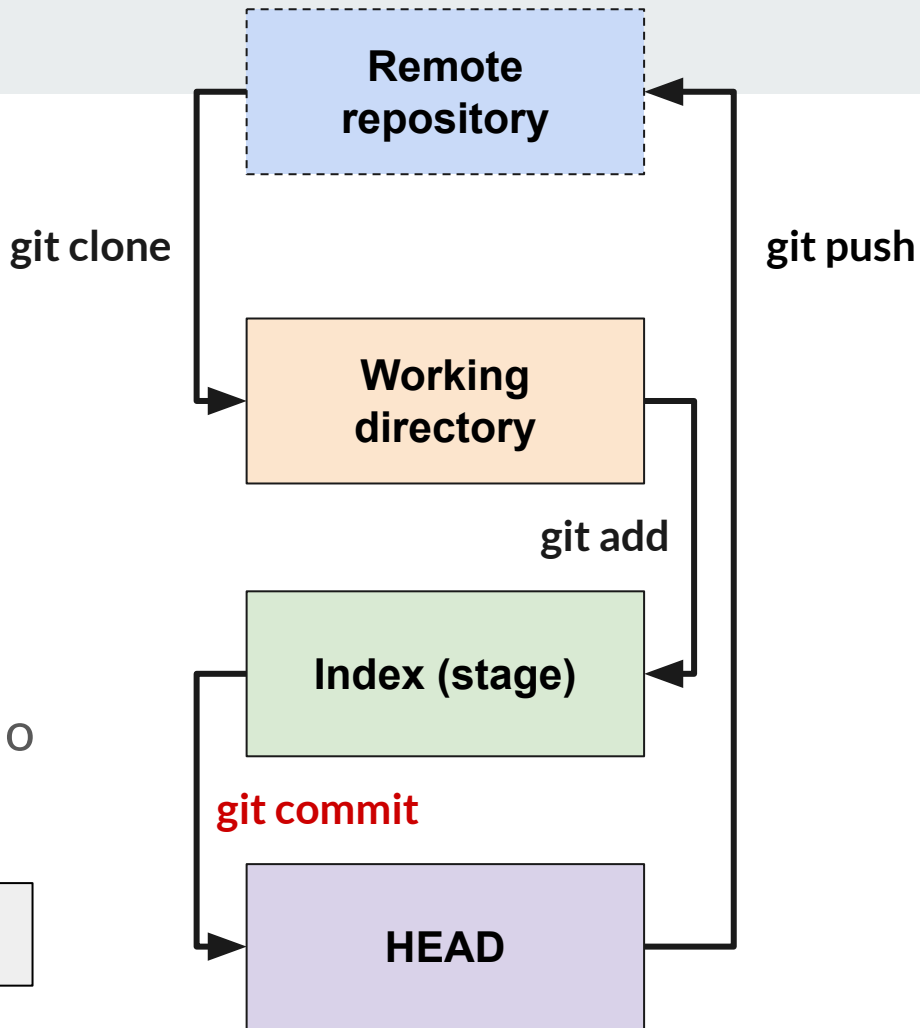


Funcionamento básico

Mudanças temporárias são confirmadas por meio de commits

- O **HEAD** passa a apontar para o último commit realizado

```
> git commit -m "comentário"
```

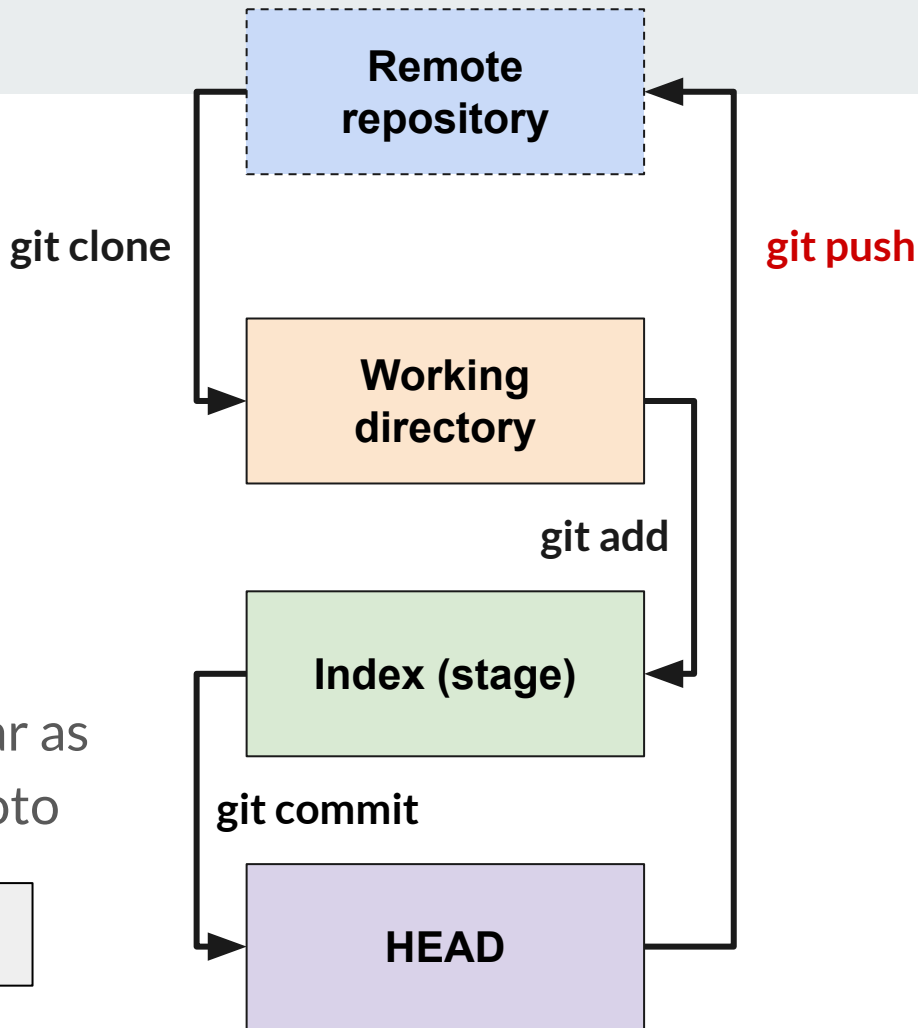


Funcionamento básico

Todas as mudanças ainda estão no repositório local

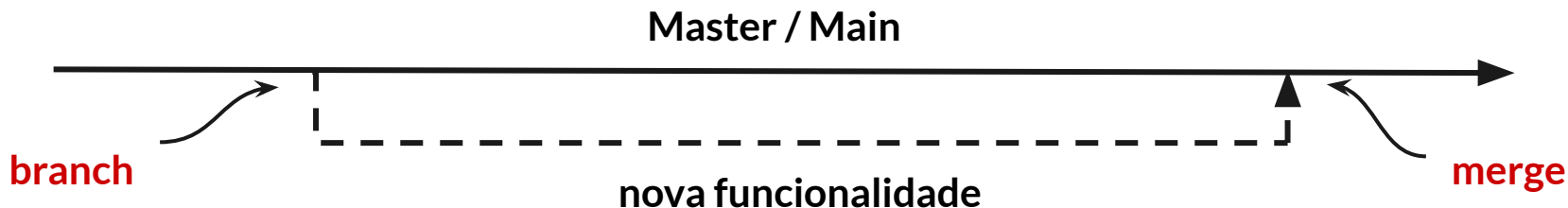
É preciso fazer um push para enviar as alterações para o repositório remoto

```
> git push origin master
```



Usando branches (ramos)

- Branches são usadas para o desenvolvimento isolado de funcionalidades
 - Cada branch é um fluxo de código independente
 - *master* ou *main* é branch padrão de um repositório (em produção)





Usando branches (ramos)

- Criando uma nova branch chamada “nova_funcionalidade”

```
> git checkout -b nova_funcionalidade
```

- Retornando para a branch master

```
> git checkout master
```



Usando branches (ramos)

- Removendo uma determinada branch

```
> git checkout -d nova_funcionalidade
```


- Enviando a branch local para o repositório remoto
 - Ficará disponível para os outros colaboradores


```
> git push origin <nome da branch local>
```


Criando pull-requests


- Após um push, um pull-request pode ser criado da branch de trabalho para a branch de destino (por exemplo a master)



 psoft had recent pushes less than a minute ago [Compare & pull request](#)

 master ▾

 4 branches

 0 tags


[Go to file](#)


[Add file ▾](#)


[Code ▾](#)


[Use this template](#)

This branch is even with psoft-2020-1:master.

 [Contribute ▾](#)

 [Fetch upstream ▾](#)

 fabiomorais Merge pull request [psoft-2020-1#4](#) from psoft-2020-1/add_client ...

b082d92 1 hour ago  20 commits



Atualizando repositórios locais

- Se uma nova versão remota está disponível é possível atualizar o repositório local

```
> git pull
```

- O git traz as atualizações e tenta fazer merge automático das versões
 - Conflitos podem ocorrer e são resolvidos manualmente



Resolvendo conflitos

- Os conflitos devem ser resolvidos manualmente, editando os arquivos conflitantes (exibidos pelo git)
 - Após as modificações os arquivos devem ser adicionados ao index para serem considerados na atualização

```
> git add <arquivo modificado>
```



Resolvendo conflitos

```
README.md x
1 # How to create a merge conflict ↵
2 ↵
3 <<<<<<< HEAD ↵
4 First you add a file, but create a conflicting
  • change ↵
5 ↵
6 on another branch. ↵
7 ===== ↵
8 First you add a file. ↵
9 ↵
10 Then you add something on another branch and
  • commit it. ↵
11 >>>>>>> new_branch ↵
12
```




Descartando alterações locais

- Caso algo de errado em um arquivo é possível reverter modificações locais
 - O comando abaixo sobrescreve o arquivo no seu working directory com a versão que consta no HEAD
 - Mudanças temporárias já adicionadas ao index são mantidas

```
> git checkout -- <arquivo a ser sobrescrito>
```



Descartando alterações locais

- Para descartar todas as alterações e commits locais é preciso recuperar o histórico do servidor
 - O working directory passa a apontar para o último commit no repositório remoto

```
> git fetch origin
```

```
> git reset --hard origin/master
```

—

Crie sua conta no github! 🤓



Referências

- [https://rogerdudler.github.io/git-guide/index.pt BR.html](https://rogerdudler.github.io/git-guide/index.pt_BR.html)
- <https://git-scm.com/docs/gittutorial>
- <https://guides.github.com/activities/hello-world/>
- <https://learngitbranching.js.org/>



Versionamento, git e github

Projeto de Software

Fabio Moraes