

# Git

Do noob até o Wizard lv12

# Por que aprender git?

- ▶ Ferramenta de controle de versão
- ▶ Ótimo para trabalhar em equipe
- ▶ Uma ótima forma de backup
- ▶ Mantem o histórico de tudo
- ▶ Muito melhor que salvar código em GDrive ou Dropbox
- ▶ É de longe a ferramenta mais adotada para versionamento no meio open-source
- ▶ Para ser o/a top da balada

## Um pouco de história

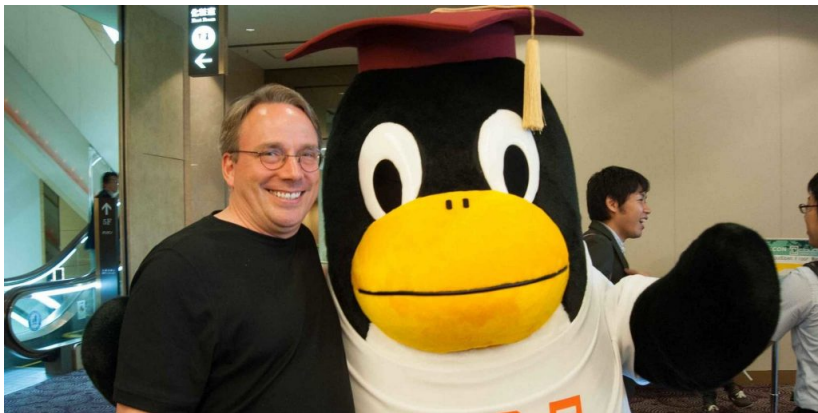


Figure 1: Linus Torvalds criador do Git

Foi criado em 2005 por Linus Torvalds para o versionamento do kernel Linux após desentendimento com a empresa responsável pelo DVCS. Foi rapidamente adotado por outros projetos.

# Sobre a apresentação

## Slides

[github.com/mateusKoppe/git-guia-basico](https://github.com/mateusKoppe/git-guia-basico)

## Links (muito) úteis

- ▶ [git-scm.com/book/pt-br/v1](https://git-scm.com/book/pt-br/v1)
- ▶ [github.github.com/training-kit/downloads/pt\\_BR/github-git-cheat-sheet.pdf](https://github.github.com/training-kit/downloads/pt_BR/github-git-cheat-sheet.pdf)
- ▶ [github.github.com/training-kit/downloads/pt\\_BR/github-git-cheat-sheet/](https://github.github.com/training-kit/downloads/pt_BR/github-git-cheat-sheet/)

# Um pouco sobre bash

`pwd` *# Exibe o diretório atual*

`ls` *# Lista os arquivos e pastas no diretório*

`mkdir` `<pasta>` *# Cria um diretório*

`cd` `<pasta>` *# Entra no diretório informada*

`mv` `<atual>` `<novo>` *# Renomea um arquivo ou diretório*

# Como instalar

## Linux

```
sudo apt install git # Debian based
```

## Mac

```
brew install git
```

## Windows

Baixe o executável no site oficial e instale

# Criando um repositório

Repositório é o local onde você armazenará e versionará o seu código.

Navege até o diretório do seu projeto e digite:

```
# Inicia um repositório vazio  
git init
```

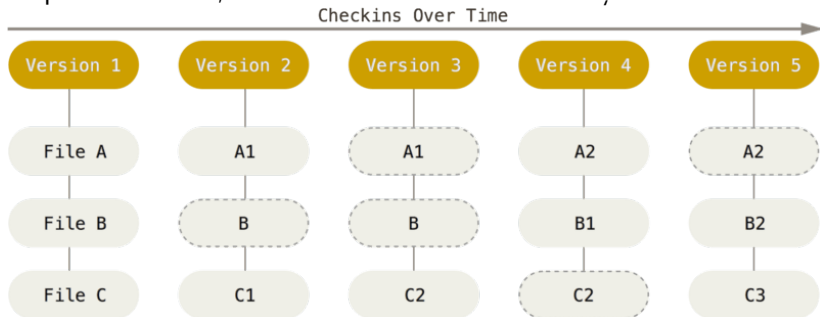
Quando você executar esse comando uma pasta `.git` será criada no diretório e essa pasta conterá todos os meta dados para gerenciar o seu repositório.

# Um pouco do conceito

O git funciona com base em snapshots, que armazenam as modificações que foram feitas ao longo do projeto.

Essas snapshots agem como “fotografias” que irão salvar o estado do projeto

naquele momento, e então irão salvar essas mudanças em um commit.





# Setup

Antes de começarmos é necessário configurar algumas informações no git, como por exemplo o nome e o e-mail do usuário já que esses dados que estarão atrelados ao commit.

```
git config --global user.name <name>
```

```
git config --global user.email <email>
```

## Sobre `--global`

Utilize apenas no seu computador pessoal, caso você queria definir a configuração apenas para um repositório específico use `--local` (já é o padrão)

## Precisando de ajuda?

```
git help # Lista comandos possíveis
```

```
git help [comando] # Exibe informações sobre o comando
```

# Comandos e conceitos básicos

O básico para “se virar”.

- ▶ `git status`
- ▶ `git add`
- ▶ `git commit`
- ▶ `git reset`
- ▶ `git diff`

## git status

`git status` é o comando que você utilizará sempre que quiser informações sobre as modificações no repositório.

*# Recebe status do repositório, dos arquivos*

`git status`

Os arquivos no seu repositório estarão em um dos dois estados: `untracked` (não monitorado) ou `tracked` (monitorado).

## git status

### Tracked (monitorado)

Arquivos em tracked são os arquivos que já estavam na seu último commit, ou seja, o repositório já sabe da existencia desse arquivo. O arquivo pode estar em um dos seguintes estados: unmodified (não modificado), modified (modificado), ou staged (selecionado).

### Untracked (não monitorado)

São os arquivos que ainda não fazem parte do histórico do repositório ou ainda não foram selecionados.

## Dicas

Você pode passar a flag `-s` para ver uma versão curta e mais amigável do status.

```
git status -s
```

## git add

Quando você quiser adicionar/monitorar um arquivo para ser commitado você deve utilizar o comando `git add`, esse comando alterará o status do arquivo para staged.

(vai deixar o arquivo verdinho na lista)

```
# Adiciona arquivos para serem trackeados  
git add <arquivo ou pasta>
```

## Dica

Caso esteja com preguiça de digitar manualmente todos os arquivos que foram modificados você pode simplesmente adicionar o diretório inteiro utilizando `.`, mas tome cuidado para não adicionar arquivo indesejáveis.

```
git add .
```



## git reset

*Adicionei um arquivo in staged sem querer, e agora? Você pode usar `git reset` para retornar um arquivo para o seu status original.*

Git status por padrão não irá fazer você perde código.

Outras funcionalidades do `git reset` serão listadas futuramente.

```
git reset <arquivo>
```

## git commit

Tudo pronto? Chegou a hora de salvar as mudanças em um commit com uma mensagem dizendo o que esse commit faz, para isso use o comando `git commit`.

O commit é justamente o que da essa ideia de snapshots, será com commits que você irá salvar determinados momentos do seu código e irá “encapsular” as mudanças selecionadas.

```
# Cria um commit abrindo-o em um editor  
git commit
```

```
# De forma rápida:  
git commit -m "<message>"
```

## git log

*E para ver isso aí??*

Utilize git log para ver quais são os logs de commits do seu repositório.

*# Exibe os logs*

```
git log
```

*# Uma linha por log*

```
git log --oneline
```

## Dica

Você pode usar a flag `--oneline` para visualizar os commits em apenas uma linha e `--graph` para visualizar o histórico de commit em forma de gráfico.

```
git log --oneline
```

```
git log --graph
```

```
git log --oneline --graph # Why not both?
```

## git diff

*# Exibe as diferenças que não foram adicionadas*

```
git diff
```

*# Exibe as que foram adicionadas:*

```
git diff --cached
```

# Branches

- ▶ `git-branch`
- ▶ `git-checkout`
- ▶ `git-stash`
- ▶ `git-merge`

# git-branch

*# Lista as branches criadas e exibe a branch atual*

`git branch`

*# Cria uma nova branch baseada na branch atual*

`git branch <name>`

*# Deleta uma branch*

`git branch -d <name>`

# git-checkout

*# Troca de branch*

```
git checkout <name>
```

*# Cria e troca de branch*

```
git checkout -b <name>
```



## git-stash

*# Salva as mudanças de uma branch e reseta-a*

`git stash`

*# Retorna as mudanças que foram salvas para a branch*

`git stash apply`

## git-merge

*# Junta os commits da branch atual com a branch alvo*  
`git merge <name>`

# Remote


Git não faria sentido se não houvesse uma forma de armazenar os repositório em algum lugar onde possa ser compartilhado para outras pessoas.

Crie um repositório online

- ▶ `git-remote`
- ▶ `git-push`
- ▶ `git-pull`
- ▶ `git-clone`

# Criando um repositório no Github

Owner

 mateusKoppe ▾

 / 


Repository name \*

Great repository names are short and memorable. Need inspiration? How about **fantastic-guide**?


Description (optional)

---

☒

 **Public**  
Anyone can see this repository. You choose who can commit.

☐

 **Private**  
You choose who can see and commit to this repository.


---

☐

**Initialize this repository with a README**  
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾

Add a license: **None** ▾

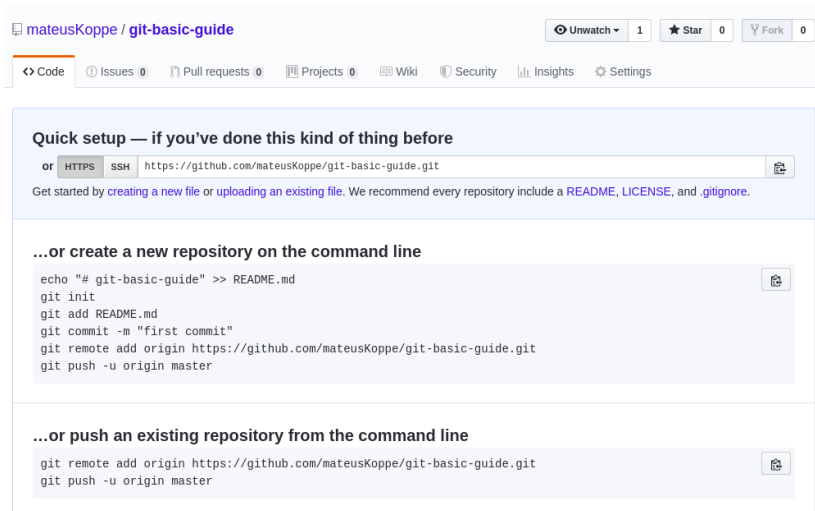


---

Create repository

Figure 2:

# Criando um repositório no Github



The screenshot shows the GitHub repository page for 'mateusKoppe / git-basic-guide'. The repository has 1 watch, 0 stars, and 0 forks. The navigation bar includes links for Code, Issues (0), Pull requests (0), Projects (0), Wiki, Security, Insights, and Settings.

**Quick setup — if you've done this kind of thing before**

or **HTTPS** **SSH** `https://github.com/mateusKoppe/git-basic-guide.git`

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

**...or create a new repository on the command line**

```
echo "# git-basic-guide" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/mateusKoppe/git-basic-guide.git
git push -u origin master
```

**...or push an existing repository from the command line**

```
git remote add origin https://github.com/mateusKoppe/git-basic-guide.git
git push -u origin master
```

Figure 3:

## git-remote

*# Lista os repositórios adicionados*

```
git remote
```

*# Adiciona um repositório remoto*

```
git remote add <name> <url>
```

*# Por convenção o repositório principal geralmente*

*# é nomeado como origin*

```
git remote add origin <url>
```

# git-push

```
# Push = Empurra  
# Envia os commits da branch selecionada  
# para o remote selecionado  
git push <remote> <branch>  
  
# O mais comum é  
git push origin master
```

# git-pull

```
# Pull = Puxa  
# Atualiza a branch selecionada de acordo com o  
# remote selecionado  
git pull <remote> <branch>
```



# git-clone

*# Clona um repositório online*

`git clone <url> [<folder>]`

# Gitignore

Caso seja necessário que o repositório ignore algum arquivo ou algumas pasta é possível criar um `.gitignore`, nele você insere quais arquivos deverão ser ignoradas no repositório.

## Variável HEAD

Reflete o branch e commit atual. Você também pode utilizar ~<n> para referenciar commits anteriores.

# Macetes

```
git push origin HEAD
```

```
git reset --head HEAD~1
```