



Git e GitHub

Principais comandos

André Estevam

a166348@dac.unicamp.br

Notas

1. Não sou especialista em Git, assistam com dúvidas e apontem possíveis erros;
2. O conteúdo foi desenvolvido com base em experiência pessoal, experimentação nas ferramentas e pesquisa na documentação do git;
3. Como este é um assunto essencialmente técnico, existem inúmeras fontes que discorrem sobre o mesmo assunto. Preferi deixar nos slides links para as fontes que utilizei ao invés de uma seção de bibliografia pois considero não confiável toda fonte que não seja a documentação oficial, assim, esses links devem ser interpretados como um “saiba mais” pois testei nas devidas ferramentas as funcionalidades que comento.

Objetivos

1. Mostrar os principais tópicos que envolvem Git/GitHub de forma prática;
2. Motivar a **pesquisa** mais profunda sobre ferramentas e comandos;
3. Exemplificar os **benefícios** da ferramenta no dia a dia de um desenvolvedor;
4. Expor minha visão sobre estas ferramentas.

Mão na

massa

1. Instalar o Git

(ferramenta para linha de comando)

Tutorial da Atlassian:

<https://www.atlassian.com/br/git/tutorials/install-git>

Mac e Linux: linha de comando;

Windows: arquivo instalável.

```
1 # No Linux
2 # abrir o terminal: ctrl + alt + t
3
4 # atualizar os pacotes da máquina
5 $ sudo apt-get update
6 # instalar o git
7 $ sudo apt-get install git
8
9 # Dica: use o argumento `-y` (sudo
  apt-get install git -y) para
  responder "sim" para as perguntas
  do instalador
10
11 # Executar um comando genérico
  para verificar se foi instalado
12 $ git --version
13 git version 2.9.2
```

2. Conta no GitHub

Acesse: <https://github.com/join>

Join GitHub

Create your account

Username *

Email address *

Password *

Make sure it's at least 15 characters OR at least 8 characters including a number and a letter. [Learn more.](#)

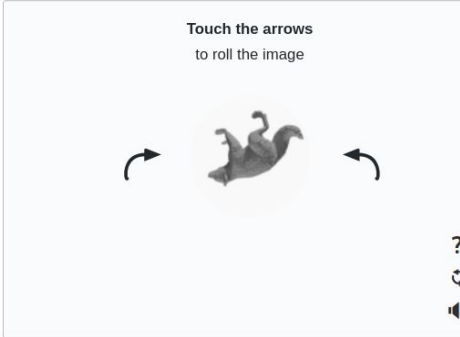
Email preferences

☐ Send me occasional product updates, announcements, and offers

Verify your account

Verify your account

Touch the arrows to roll the image




Individual

Pick the plan that's right for you, personally.

Team

Choose a plan to help your team grow and collaborate.




Free

\$0 USD

The basics of GitHub for every developer

Choose Free

- ✓ Unlimited public repositories
- ✓ Unlimited private repositories
- ✓ Limited to 3 collaborators for private repositories
- ✓ 2,000 total Action minutes/month
- ✓ 500MB of GitHub Packages storage
- ✓ Advanced vulnerability scanning for public repositories
- ✓ Automated security updates
- ✓ GitHub Security Advisories



Pro

\$7 USD

Per month

Pro tools for developers with advanced requirements

Choose Pro

- ← Includes everything in Free
- ✓ Unlimited collaborators
- ↑ 3,000 total Action minutes/month
- ↑ 1GB of GitHub Packages storage
- ✓ Private GitHub Pages and Wikis
- ✓ Private protected branches
- ✓ Code owners
- ✓ Repository insights

3. Associar Git e GitHub

```
1 # Setup do seu nome e e-mail global
2 # Utilize o username e e-mail do GitHub
3 # (default para todos os repositórios da máquina)
4 $ git config --global user.name "Fulano"
5 $ git config --global user.email "fulano@fulanomail.com"
```

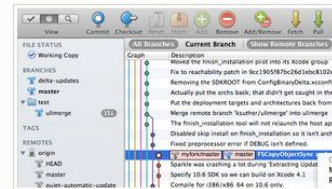
4. Instalar uma GUI e extensões

- Existem muitas interfaces disponíveis:
 - Veja: <https://git-scm.com/downloads/guis>.

GUI Clients

Git comes with built-in GUI tools for committing (**git-gui**) and browsing (**gitk**), but there are several third-party tools for users looking for platform-specific experience.

If you want to add another GUI tool to this list, just [follow the instructions](#).

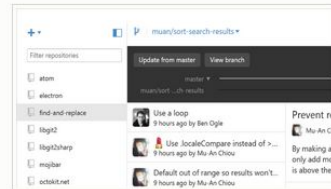


SourceTree

Platforms: Mac, Windows

Price: Free

License: Proprietary

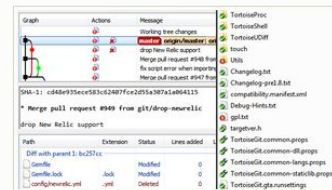


GitHub Desktop

Platforms: Mac, Windows

Price: Free

License: MIT

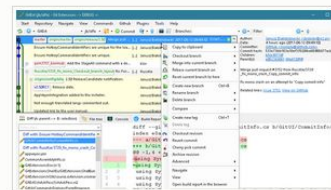


TortoiseGit

Platforms: Windows

Price: Free

License: GNU GPL

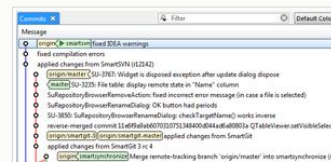
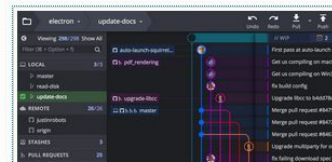


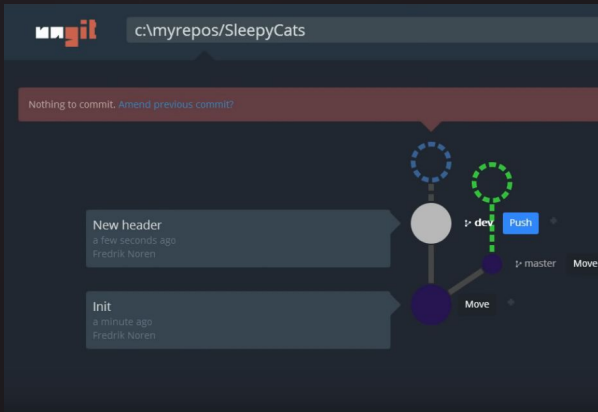
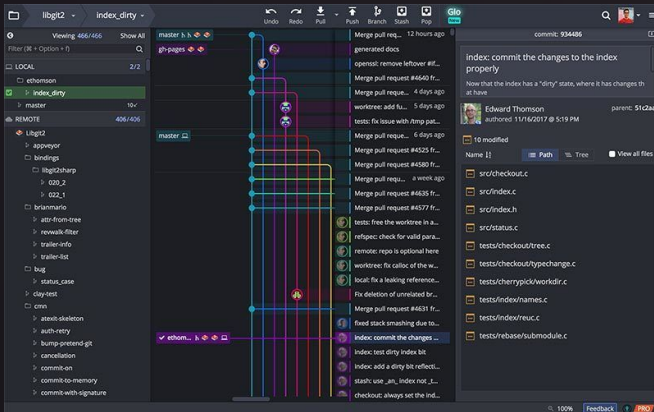
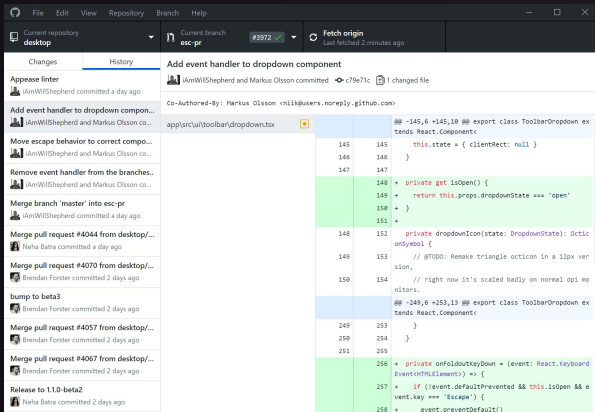
Git Extensions

Platforms: Linux, Mac, Windows

Price: Free

License: GNU GPL





GitHub Desktop

<https://desktop.github.com>

Só para windows;

Desenvolvido pelo GitHub (é o cliente **oficial**).

GitKraken

<https://www.gitkraken.com>

Todas as plataformas;

Mais comum entre desenvolvedores **profissionais**.

Ungit

<https://github.com/FredrikNoren/ungit>

Roda no browser e no VSCode;

Foco em ser **amigável** ao usuário ;

Operações com **drag & drop**.

```
1 # Instalar o GitKraken
2
3 # Por Snap
4 $ sudo apt install snapd # se não tiver instalado (`snap --version` não funcionará)
5 $ snap install gitkraken
6 $ gitkraken
7
8 # Por Flatpak
9 $ flatpak install flathub com.axosoft.GitKraken
10 $ flatpak run com.axosoft.GitKraken
```

```
1 # Instalar o Ungit
2
3 # Se não tiver o node e npm instalados:
4 $ wget -qO- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.1/install.sh | bash
5
6 # REINICIE O TERMINAL
7
8 $ nvm install node
9
10 # Checando a instalação
11 $ node --version
12 $ npm --version
13
14 # Instalando e rodando o ungit
15 $ npm install -g ungit
16 $ ungit
```

[➤ Open in browser](#)

GitKraken Release

Behold the evolution of GitKraken! Find out what's new and how we're remembering those bugs of yesterday.

Version 6.5.4

Monday, March 9th, 2020

The latest improvements to the GitKraken

Improvements 🙌

Who needs luck when you have improvements?

- When users select a parent SHA 1, the child SHA 1s are now highlighted.
- The `Cmd / Ctrl1 + F` keyboard shortcut now works.

Bug Fixes 🐛

Blarney! These bugs were making us green.

Welcome to GitKraken

[Sign in with GitHub](#)[Create a GitKraken Account](#)

or sign in with an existing GitKraken account

Email

Password

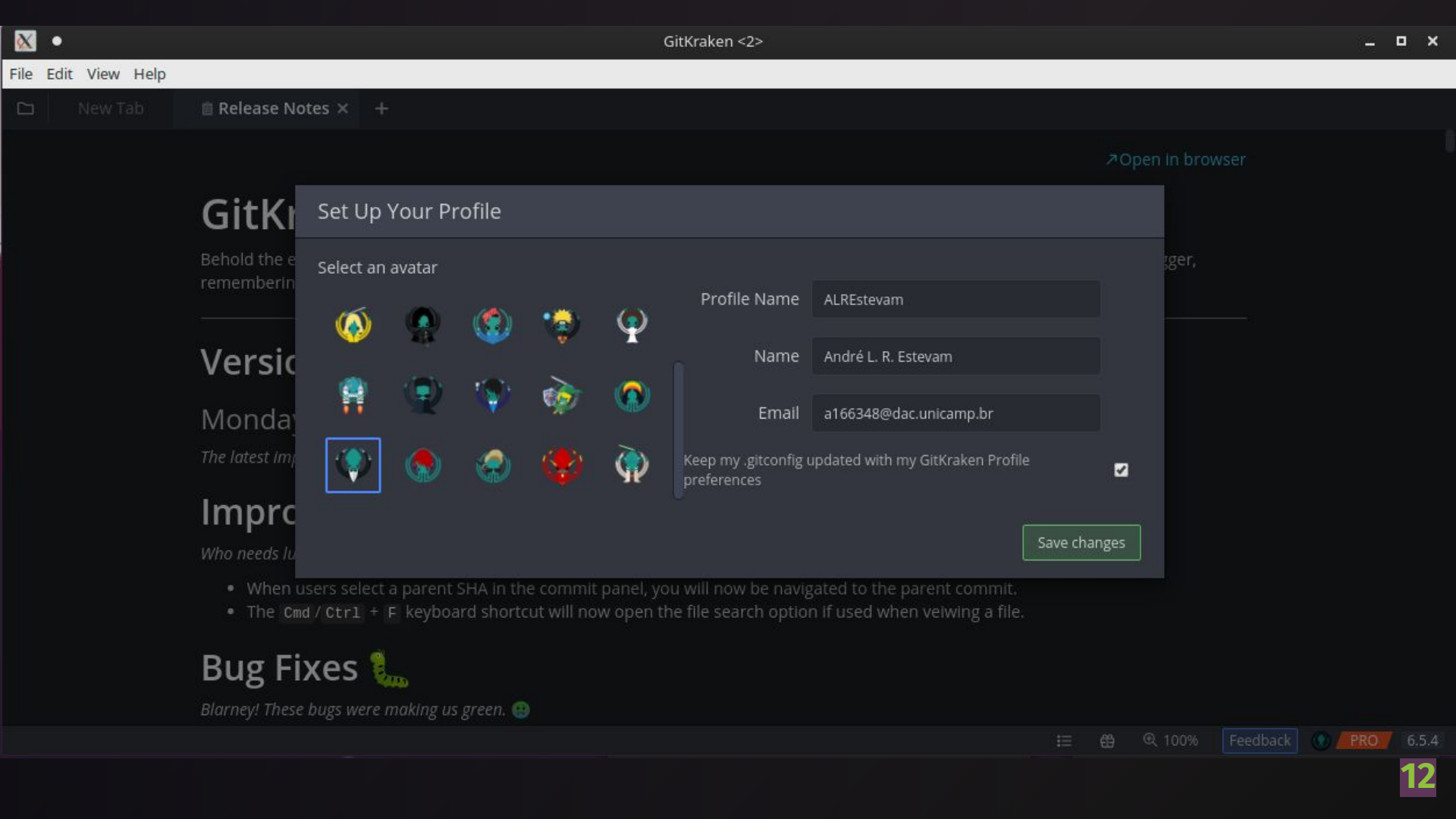
[I forgot my password](#)

100%

[Feedback](#)

FREE

6.5.4



Open In browser

GitKraken

Behold the e...
rememberin

Version

Monday

The latest Imp

Impro

Who needs lu

- When users select a parent SHA in the commit panel, you will now be navigated to the parent commit.
- The `Cmd / Ctrl1 + F` keyboard shortcut will now open the file search option if used when velwing a file.

Bug Fixes

Blarney! These bugs were making us green. 🐛

Set Up Your Profile

Select an avatar



Profile Name

Name

Email

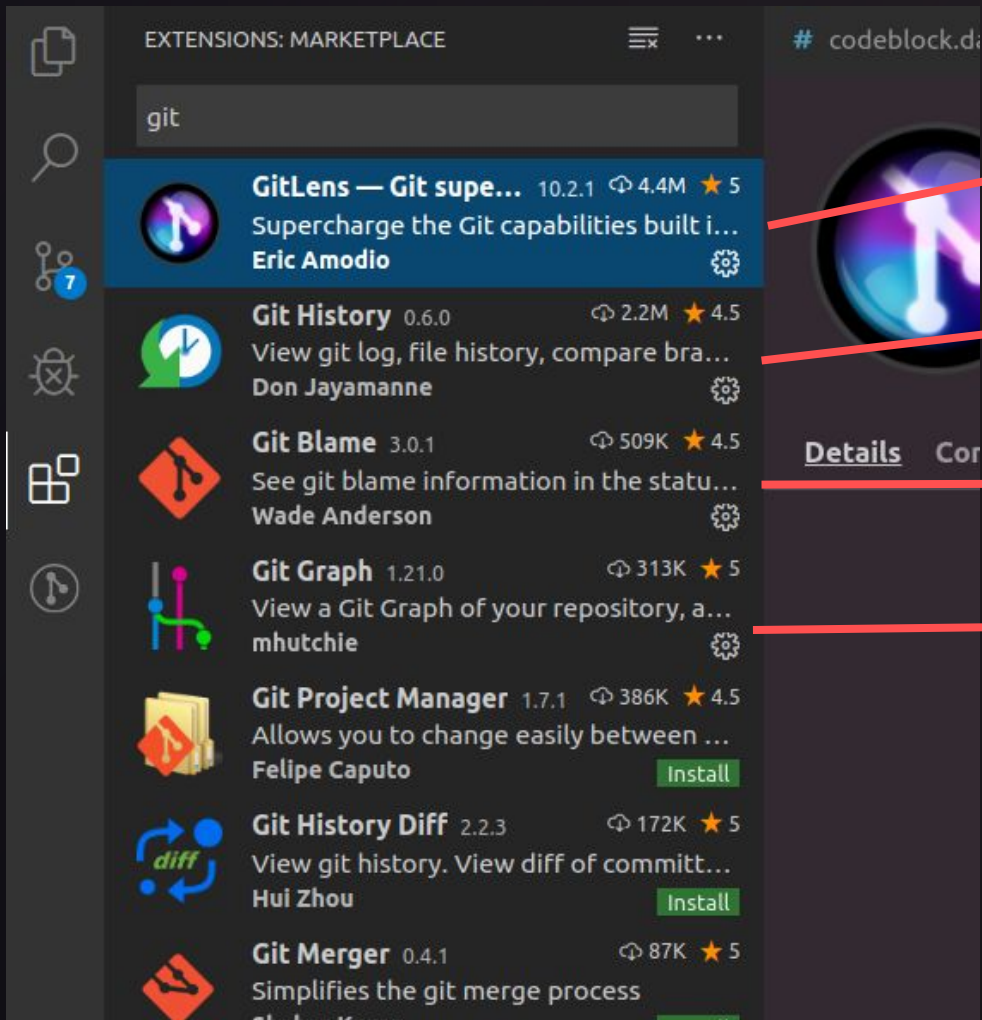
Keep my .gitconfig updated with my GitKraken Profile preferences ☒

Save changes

IDEs e Extensões

- Diversas *IDEs* (*Integrated development environment*) possuem opções visuais para fazer as operações mais comuns.





Muitas funcionalidades através de **botões**
Auxílio para **conflitos de merge**;

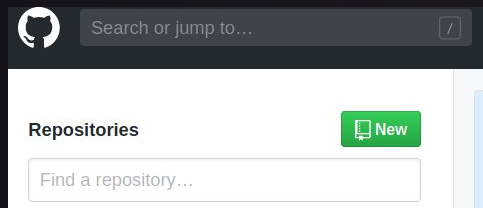
Histórico de alterações
Visualização de **versões antigas**;

Mostra linha a linha quem e quando a
escreveu ou alterou (**quem é o culpado**);

Mostra a árvore de **branches** de forma
gráfica.

Muitas outras...

Criando um Repositório



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?

[Import a repository.](#)

Owner



alreest-sec ▾

Repository name *

repositorio-de-teste ✓

Great repository names are short and memorable. Need inspiration? How about [expert-octo-disco](#)?

Description (optional)

Um repositório para testar as funcionalidades do Git e GitHub



Public

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



Private

You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

☒ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer.

Add .gitignore: **None** ▾

Add a license: **None** ▾ ⓘ

Create repository

Um repositório para testar as funcionalidades do Git e GitHub

[Edit](#)[Manage topics](#)

1 commit

1 branch

0 packages

0 releases

1 contributor

Branch: master ▾

[New pull request](#)[Create new file](#)[Upload files](#)[Find file](#)[Clone or download ▾](#)

alreest-sec Initial commit

[README.md](#)

Initial commit



README.md

Clone with HTTPS ⓘ[Use SSH](#)

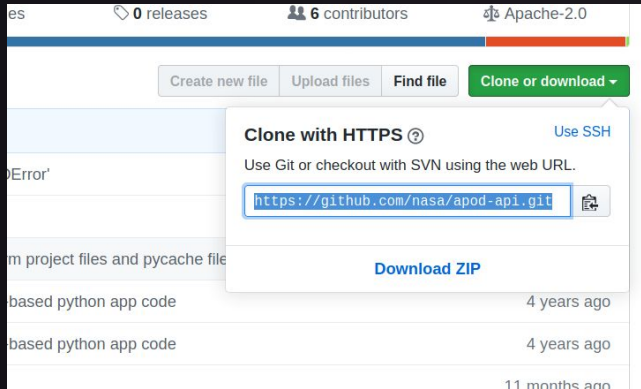
Use Git or checkout with SVN using the web URL.

<https://github.com/alreest-sec/reposi>[Download ZIP](#)

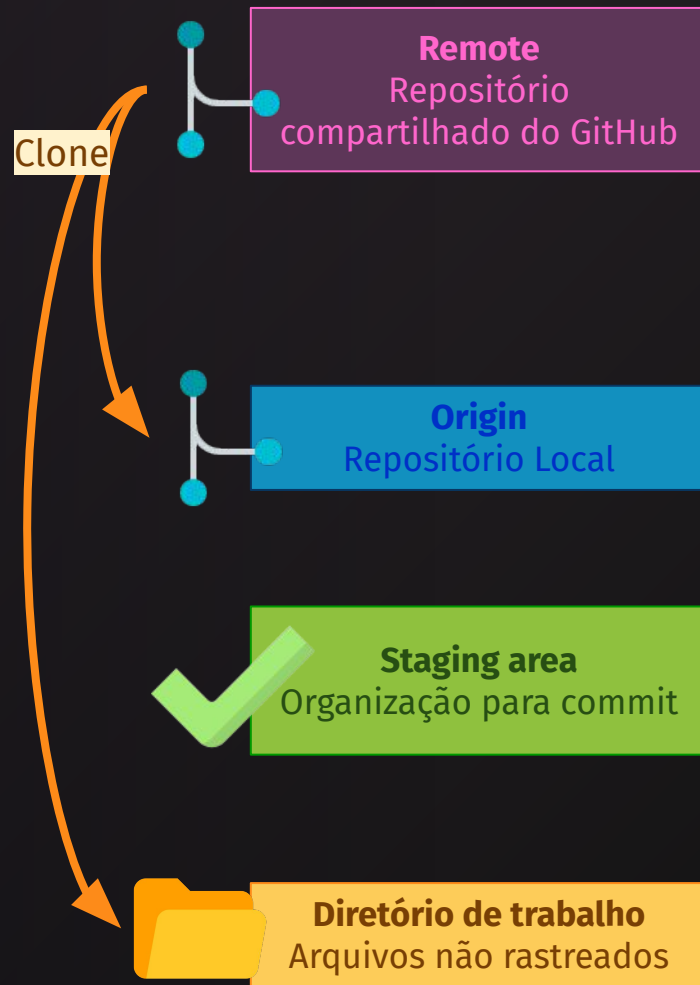
repositorio-de-teste

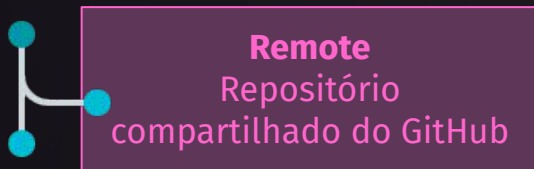
Um repositório para testar as funcionalidades do Git e GitHub

Copie todo o repositório do GitHub
para sua máquina



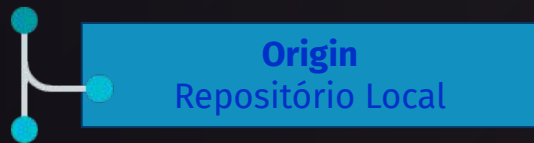
```
1 $ git clone https://github.com/[...]git
```





Faça suas alterações, acompanhe os arquivos não rastreados, na staging area e “commitados” com o comando:

```
1 $ git status
```



Adicione os arquivos alterados (apenas alguns ou todos) na staging area:

```
1 # Adicionar todos os arquivos não rastreados
2 $ git add .
3 # Arquivos de um diretório
4 $ git add ./diretorio/*
5 # Arquivos com a extensão .c
6 $ git add *.c
7 # Arquivo específico (ex main.c)
8 $ git add main.c
```





Crie um pacote de alterações com os arquivos da staging area;

Utilize uma mensagem bem descritiva:

- **Não faça:** “atualizações diversas”;
- **Faça no mínimo:** “este commit implementa a funcionalidade x”;

```
1 $ git commit -m "mensagem"
```

Neste ponto você criou um pacote com uma série de alterações atreladas a um texto explicativo;

Este “pacote” está no repositório local, você é o único com acesso;

Você pode repetir o ciclo Adicionar/alterar, *Git Add*, *Git Commit* o quanto quiser.



Quando você decidir publicar estes “pacotes” no repositório compartilhado use os comandos;

```
1 # Conectar origin (sua máquina) com remote (GitHub)
2 $ git remote add <origin> <remote>
3 $ git remote add origin https://github.com/[...].git
4
5 # Envie as mudanças em origin para remote
6 $ git push <remote> <branch>
7 $ git push origin master
```

Execute o comando `git push`, caso ocorra algum erro relacionado a falta do remote, use o `git remote add`;

Para não precisar digitar a senha a cada *push*:

Git Cache (menos seguro)

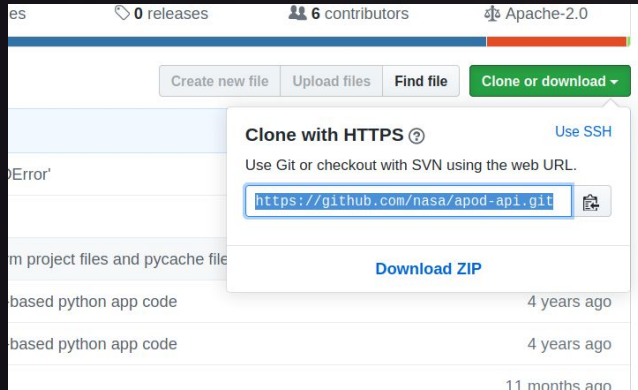
<https://pt.stackoverflow.com/a/168974>

```
1 # Git, a senha deve estar na memória
2 $ git config --global credential.helper cache
3
4 # Definir o tempo de armazenamento (1h)
5 $ git config --global credential.helper 'cache --timeout=3600'
```

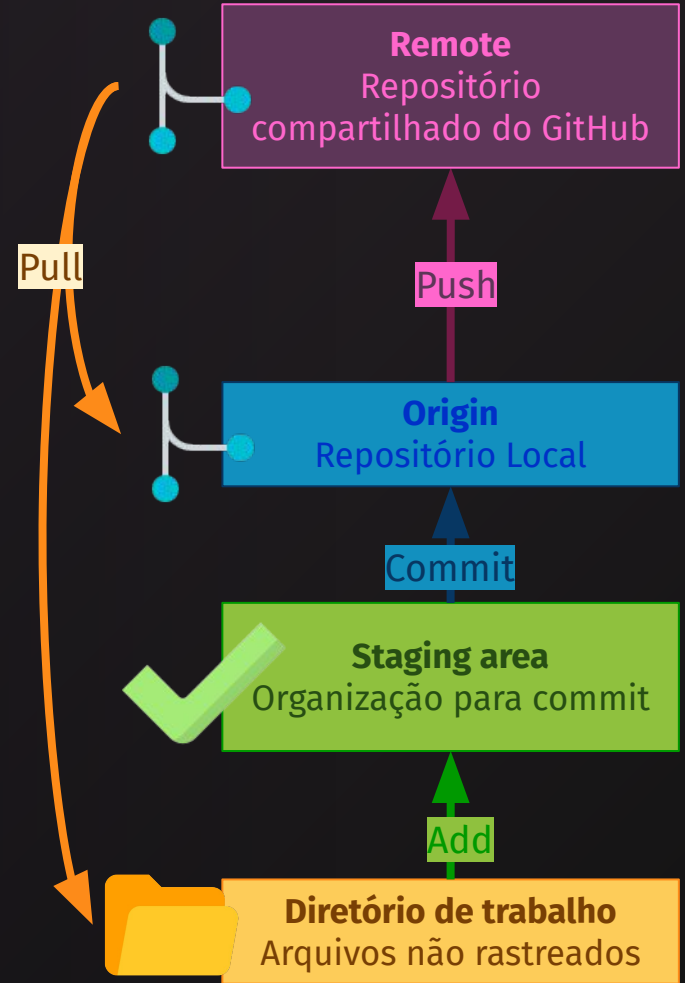
SSH Keys (mais seguro)

<https://www.atlassian.com/git/tutorials/git-ssh>

Quando outros desenvolvedores
enviarem novas mudanças, use o pull
para:



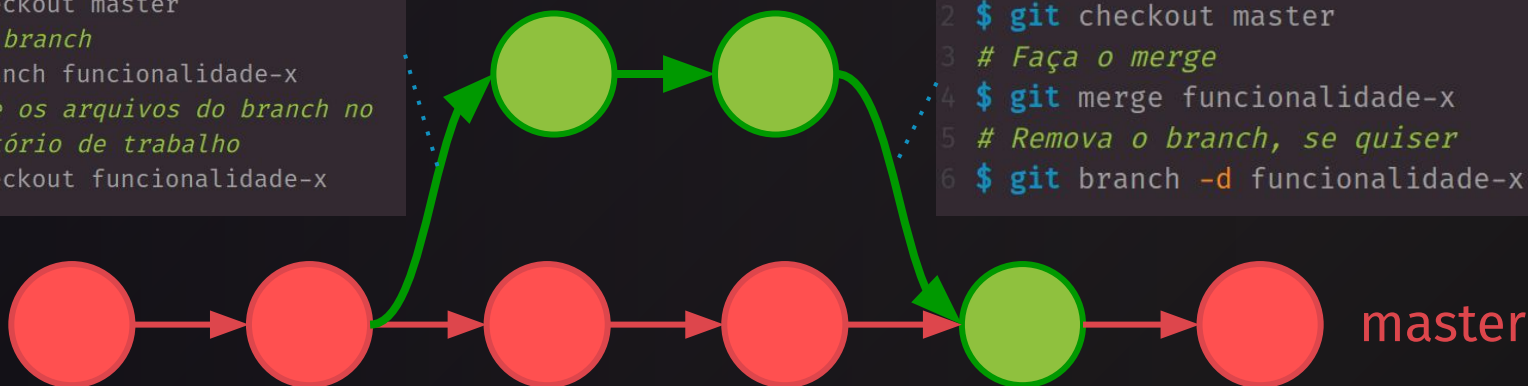
```
1 # Baixa atualizações e faz um  
2 # merge com o código local  
3 $ git pull <remote>
```



```
1 # listar todos os branches em remote
2 $ git branch -a
```

```
1 # Dê checkout para o branch base
  do seu novo branch
2 $ git checkout master
3 # Crie o branch
4 $ git branch funcionalidade-x
5 # Coloque os arquivos do branch no
  seu diretório de trabalho
6 $ git checkout funcionalidade-x
```

funcionalidade-x



```
1 # Faça checkout para o branch ALVO do merge
2 $ git checkout master
3 # Faça o merge
4 $ git merge funcionalidade-x
5 # Remova o branch, se quiser
6 $ git branch -d funcionalidade-x
```

```
1 # Bônus
2 # Cria e faz checkout em um branch
3 $ git checkout -b <nome-branch>
```

Linha de comando vs *GUIs*

- *minha opinião*

1. **Entenda** os conceitos através da linha de comando - descubra o que ele pode fazer;
2. Use as *GUIs* quando tiver **dificuldade**, mas tenha em mente que elas têm limitações;
3. Então **escolha** entre o que for mais prático, fácil, rápido etc.
4. **Em suma:** use a ferramenta certa para o trabalho certo.