



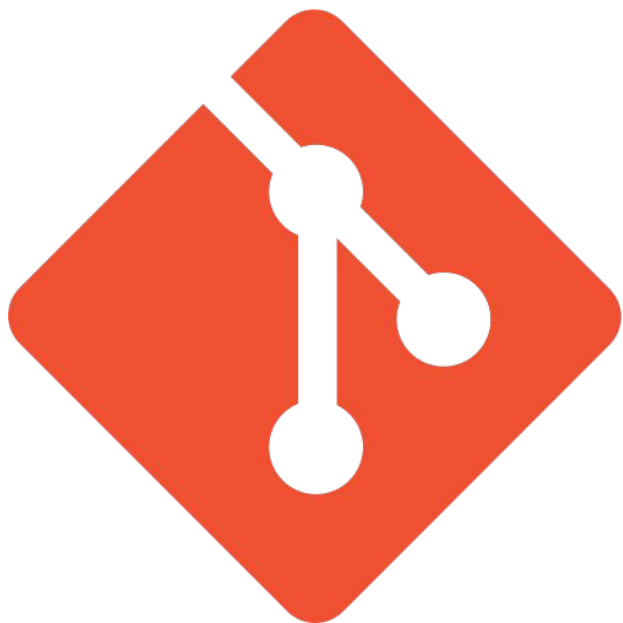
## Controle de Versionamento de Software

# Versionamento

- O versionamento é o gerenciamento de versões diferentes de um documento de texto qualquer. Perceba que não precisa ser código.
- O versionamento é controlado pelo o que chamamos de sistema de controle de versões. Normalmente, esses sistemas são utilizados no desenvolvimento de software para controlar as diferentes versões e histórico de desenvolvimento do código.

# Versionamento – Vantagens

- Controle de histórico
- Trabalho em equipe
- Marcação e resgate de versões estáveis
- Ramificação de projeto
- Segurança
- Confiança



# git

# GIT

- Antes de entender como versionar utilizando o Git, é importante saber o que é o Git de fato.
- O Git é uma **ferramenta de versionamento** muito poderosa que permite que desenvolvedores colaborem entre si de forma organizada na construção de um projeto que envolva código.
- Com ela é possível criar versões de arquivos e até mesmo projetos.

# GIT - História

- O Git foi inicialmente projetado e desenvolvido por **Linus Torvalds** para o desenvolvimento do **kernel Linux**, mas foi adotado por muitos outros projetos.
- O Git é um **software livre**.

# GIT - História

- “Git” é uma gíria em inglês britânico para cabeça dura, pessoas que acham que sempre têm razão, argumentativas, podendo ser também pessoa desagradável ou estúpida:
- “ I'm an egotistical bastard, so I name all my projects after myself. First Linux, now git. “

— Linus Torvalds

- “ *Eu sou um desgraçado egocêntrico, então batizo todos os meus projetos com meu nome. Primeiro Linux, agora Git.* ”

• — *Linus Torvalds*

# GIT - Instalação

- <https://git-scm.com/>



The screenshot shows the Git website homepage. At the top left is the Git logo (an orange diamond with a white 'G') followed by the text 'git --distributed-even-if-your-workflow-isnt'. To the right is a search bar with the placeholder text 'Search entire site...'. Below the header, there are two paragraphs of text. The first paragraph describes Git as a 'free and open source' distributed version control system. The second paragraph describes Git as 'easy to learn' and having a 'tiny footprint with lightning fast performance'. To the right of the text is a diagram showing a network of Git repositories represented as stacks of books, connected by colored lines (red, blue, yellow). Below the text and diagram, there are four sections: 'About' (with a gear icon), 'Documentation' (with a book icon), 'Downloads' (with a download arrow icon), and 'Community' (with a speech bubble icon). To the right of these sections is a monitor displaying the 'Latest source Release 2.39.1' and a 'Download for Windows' button.

**git** --distributed-even-if-your-workflow-isnt

Search entire site...

Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient **staging areas**, and **multiple workflows**.

**About**  
The advantages of Git compared to other source control systems.

**Documentation**  
Command reference pages, Pro Git book content, videos and other material.

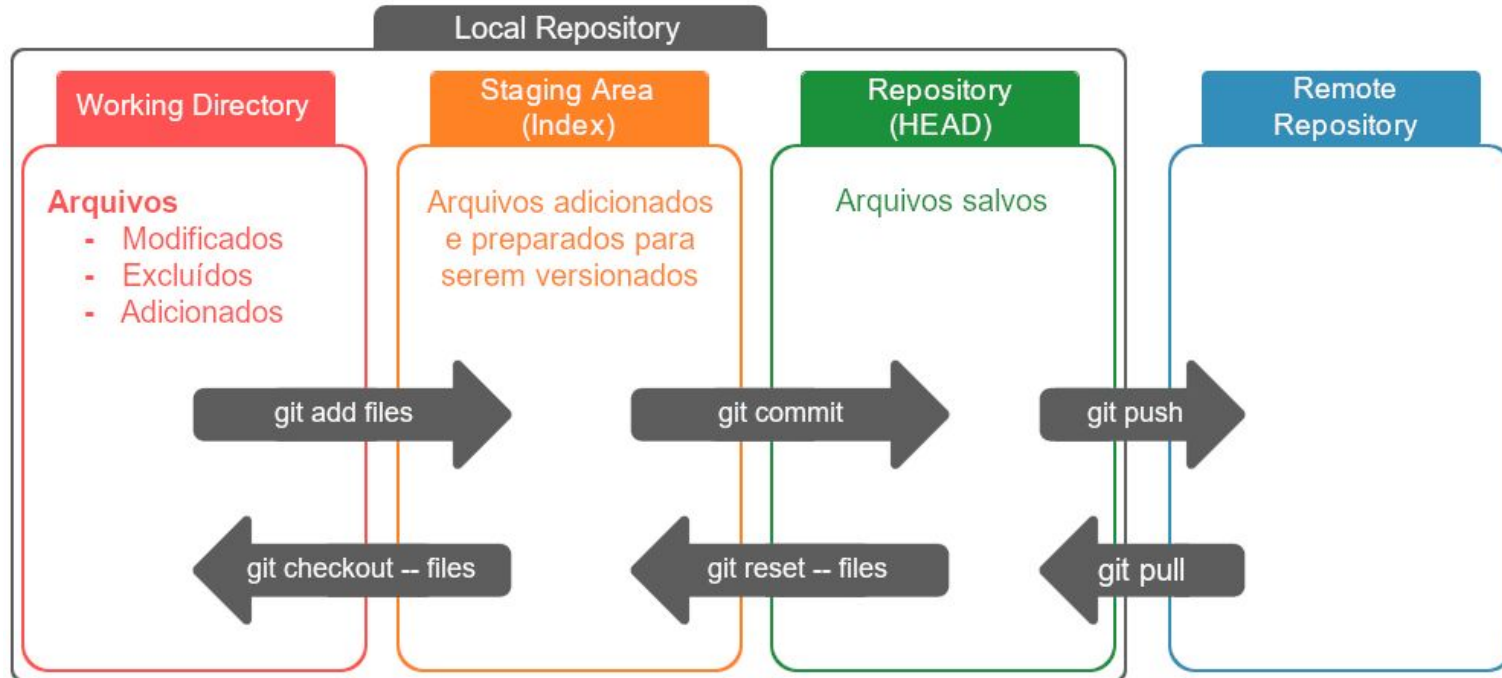
**Downloads**  
GUI clients and binary releases for all major platforms.

**Community**  
Get involved! Bug reporting, mailing list, chat, development and more.

Latest source Release  
**2.39.1**  
[Release Notes \(2022-12-13\)](#)  
[Download for Windows](#)



# GIT - Fluxo



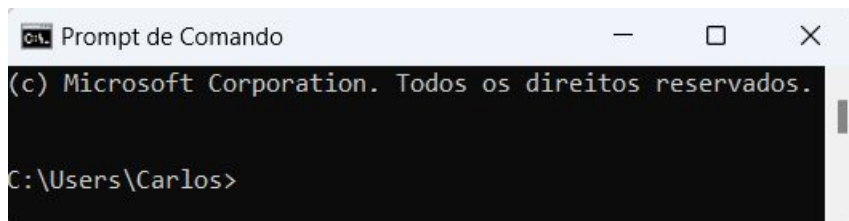
- **Working Directory** : Diretório de trabalho
- **Staging Area** : Área de preparação
- **Repository** : Repositório
- **Remote Repository** : Repositório remoto

# Termos – Git

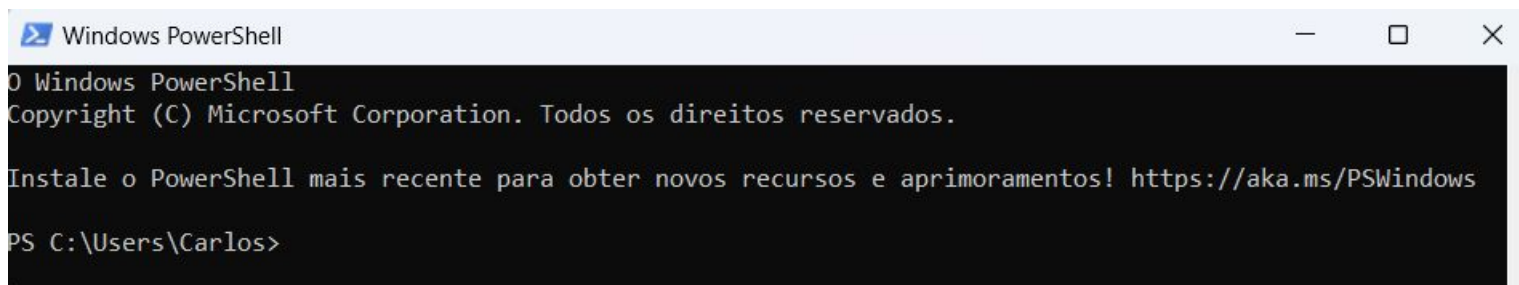
- **Repositório**
  - local onde são armazenados pacotes de software que podem ser recuperados e utilizados
- **Repositório remoto**
  - versões de nossos repositórios locais hospedadas na internet
- **Commits**
  - agrupar alterações realizadas em seu código sob um contexto
- **Branches**
  - diretórios que estão sempre apontando para o último commit de que possuem conhecimento

# Terminais

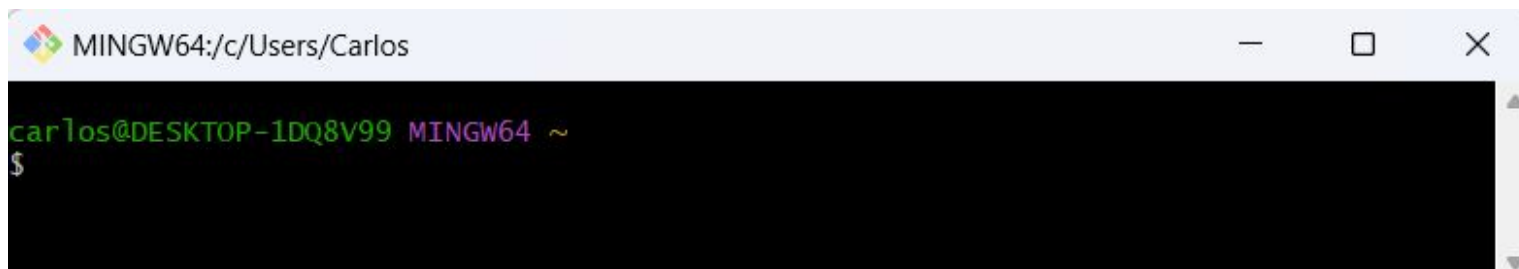
- CMD
- Power Shell
- Git Bash



A screenshot of the Windows Command Prompt (CMD) window. The title bar reads "C:\ Prompt de Comando". The window content shows the copyright notice "(c) Microsoft Corporation. Todos os direitos reservados." and the current directory "C:\Users\Carlos>".



A screenshot of the Windows PowerShell window. The title bar reads "Windows PowerShell". The window content shows the copyright notice "Copyright (C) Microsoft Corporation. Todos os direitos reservados.", a message to install the latest PowerShell version with a link "https://aka.ms/PSWindows", and the current directory "PS C:\Users\Carlos>".



A screenshot of the MINGW64 terminal window. The title bar reads "MINGW64:/c/Users/Carlos". The window content shows the prompt "carlos@DESKTOP-1DQ8V99 MINGW64 ~" and a dollar sign "\$".

# Terminais – Git Bash

*Diretório (pasta)*



```
MINGW64:/c/Users/Carlos/Desktop/git
carlos@DESKTOP-1DQ8V99 MINGW64 ~/Desktop/git
$
```

Exemplo de pasta aberta no terminal git bash

*Diretório (pasta)*      *Branch*



```
MINGW64:/c/Users/Carlos/Desktop/git
carlos@DESKTOP-1DQ8V99 MINGW64 ~/Desktop/git (master)
$
```

Exemplo de pasta com repositório aberta no terminal git bash

# GIT – Configuração

A primeira coisa que você deve fazer ao instalar Git é configurar seu nome de usuário e endereço de e-mail.

Isto é importante porque cada commit usa esta informação, e ela é carimbada de forma imutável nos commits que você começa a criar:

```
git config --global user.name "Fulano de Tal"
```

```
git config --global user.email fulanodetal@exemplo.br
```

<https://git-scm.com/book/pt-br/v2/Come%C3%A7ando-Configura%C3%A7%C3%A3o-Inicial-do-Git>

# GIT - Criação

- Criar pontos na história da produção do projeto:

# inicia a linha do tempo:

```
git init
```

# prepara as mudanças para irem para a linha do tempo:

```
git add .
```

# adiciona efetivamente um ponto na linha do tempo:

```
git commit -m "mensagem"
```

# GIT - Consultas

- Verificar mudanças feitas no projeto:

# visualiza os pontos na linha do tempo / commit:

`git log`

# informa o estado das alterações do nosso projeto:

`git status`

# apresenta determinado ponto na história:

`git show`

# GIT - Recuperação

- Voltar um arquivo para determinada ponto da linha do tempo:

# desfazer último commit alterando os arquivos:

```
git reset --hard HEAD~1
```

# voltar para determinado ponto da linha do tempo:

# ou recuperar arquivo deletado:

```
git checkout hash File.ext
```

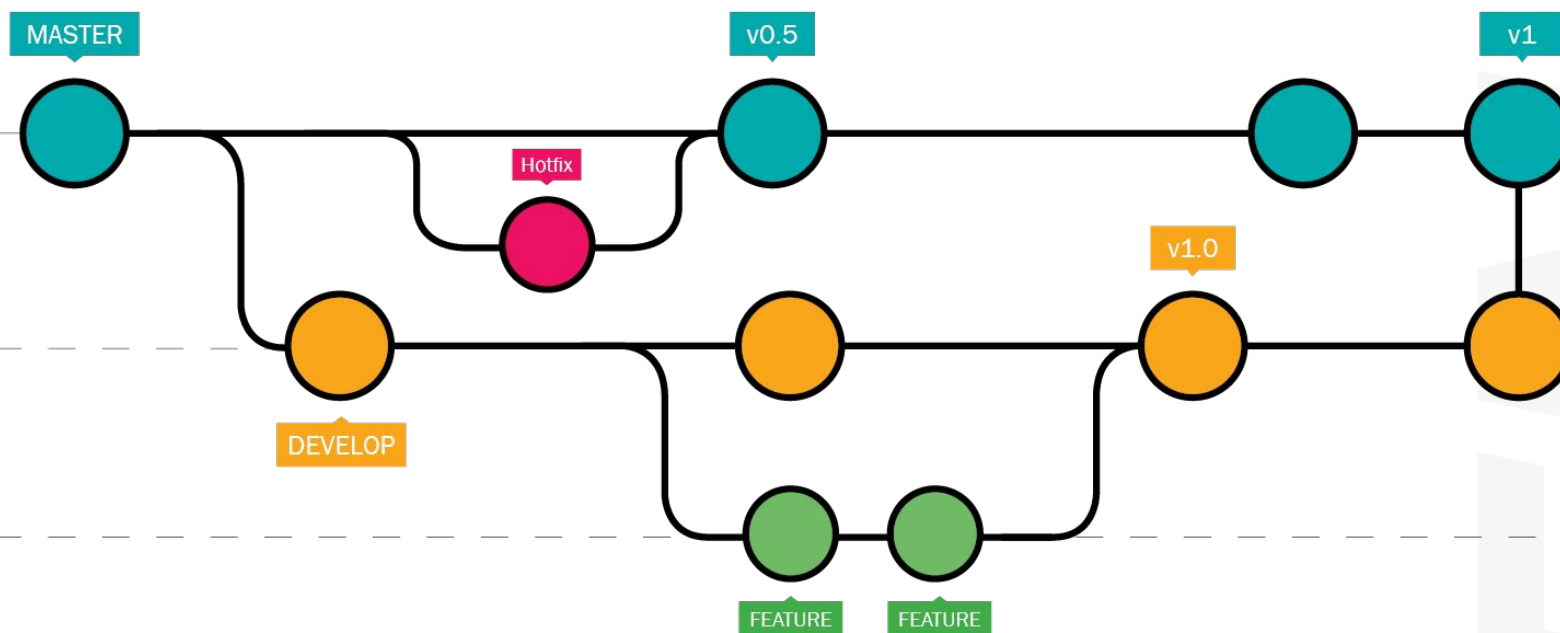
# reverte a preparação anterior:

```
git rm --cached File.ext
```



# GIT FLOW

SENAI-SP



# GIT - Branches

- Começar uma nova funcionalidade sem estragar o que já foi feito:

# criar novas linhas do tempo:  
git branch **feature/textos**

# entrar ou sair das linhas do tempo:  
git checkout **feature/textos**

# unir linhas do tempo (é necessário estar na branch “pai” Ex: main):  
git merge **feature/textos**

# GIT - Branches

- Deletar branch após a funcionalidade aplicada no projeto :

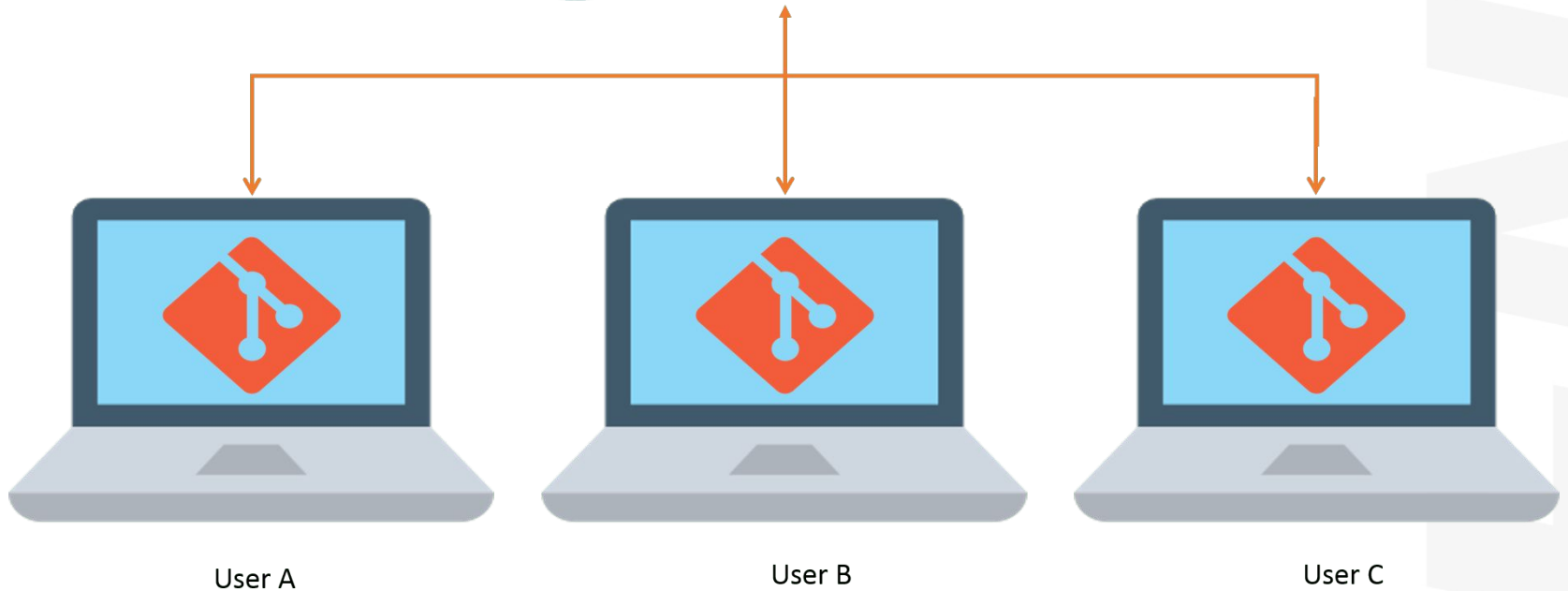
# deletar branch após implantada a solução:  
git branch -D [feature/textos](#)



# GITHUB

- O GitHub é um serviço baseado em nuvem que hospeda um sistema de controle de versão (VCS) chamado Git. Ele permite que os desenvolvedores colaborem e façam mudanças em projetos compartilhados enquanto mantêm um registro detalhado do seu progresso.

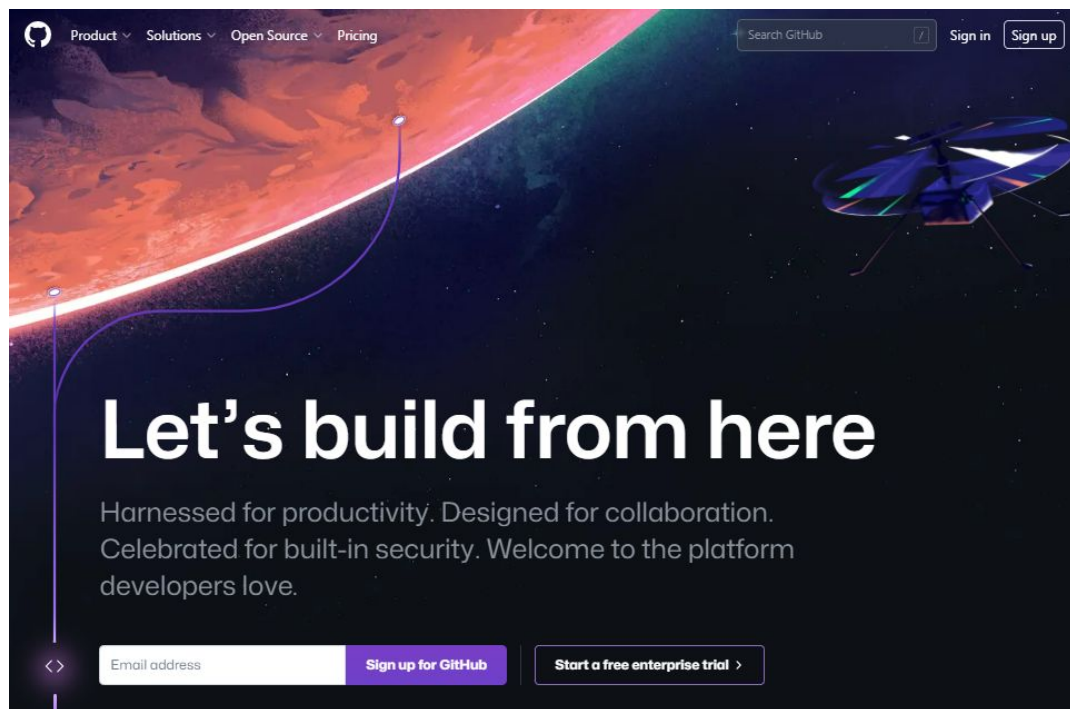
# GIT vs GITHUB



# GITHUB

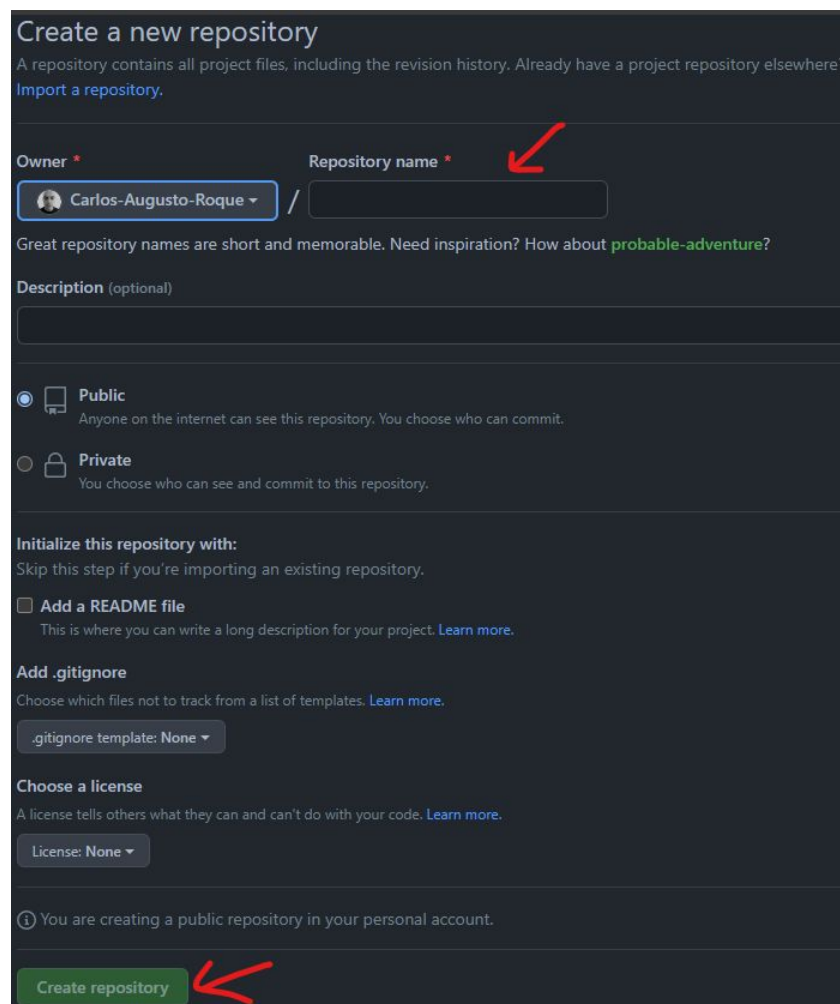
- Criar uma conta no GitHub

<https://github.com/>




# GITHUB

- Criar 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 \* Carlos-Augusto-Roque ▾ / Repository name \* 

Great repository names are short and memorable. Need inspiration? How about [probable-adventure?](#)

Description (optional)

☒ **Public**  
Anyone on the internet can see this repository. You choose who can commit.

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

**Initialize this repository with:**  
Skip this step if you're importing an existing repository.


☐ **Add a README file**  
This is where you can write a long description for your project. [Learn more.](#)


**Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: **None ▾**

**Choose a license**  
A license tells others what they can and can't do with your code. [Learn more.](#)

License: **None ▾**

 You are creating a public repository in your personal account.

**Create repository** 



# GITHUB

- Adicionar um repositório local já existente:

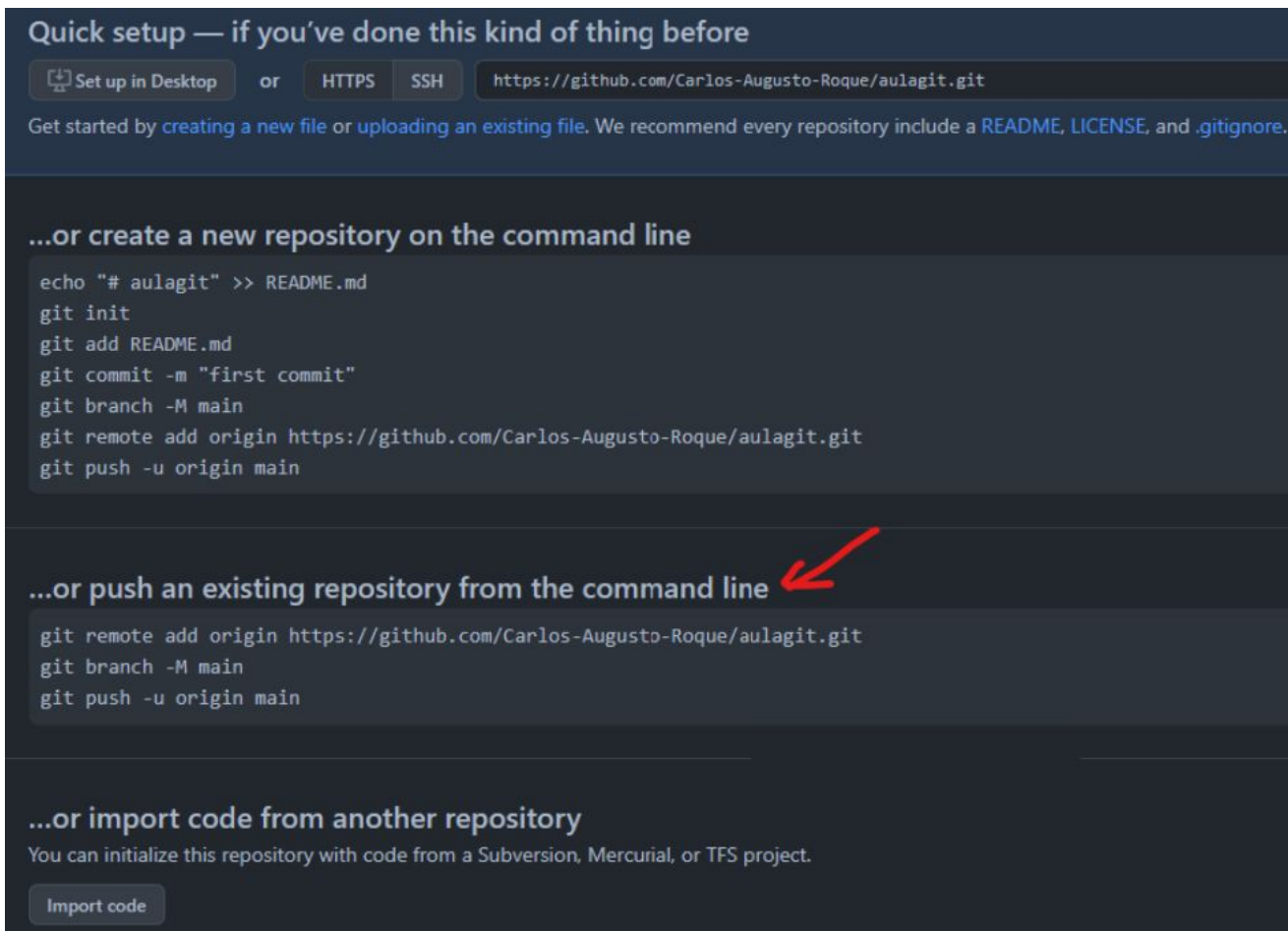
```
git remote add origin https://github.com....
```

```
git push -u origin main
```


*(-u = cria a main no github(remoto), usem somente no 1º push)*

# GITHUB

## - Criar uma conta repositório :




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

 Set up in Desktop or **HTTPS** **SSH** `https://github.com/Carlos-Augusto-Roque/aulagit.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 "# aulagit" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/Carlos-Augusto-Roque/aulagit.git
git push -u origin main
```

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

```
git remote add origin https://github.com/Carlos-Augusto-Roque/aulagit.git
git branch -M main
git push -u origin main
```

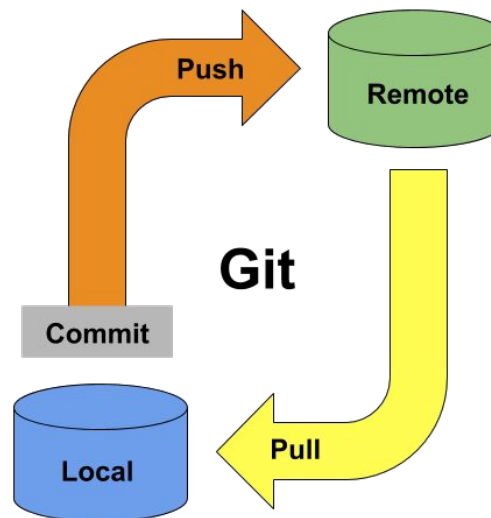
...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

[Import code](#)

# GITHUB

- pegar um projeto já iniciado :



# clona um repositório do github:

git clone <https://github.com>....

# pegar alterações feitas remotamente(Github) **ANTES** de enviar as alterações locais:

git pull

# enviar alterações locais para o repositório do Github:

git push origin main

Vamos praticar um pouco !