



Controle de versionamento de software



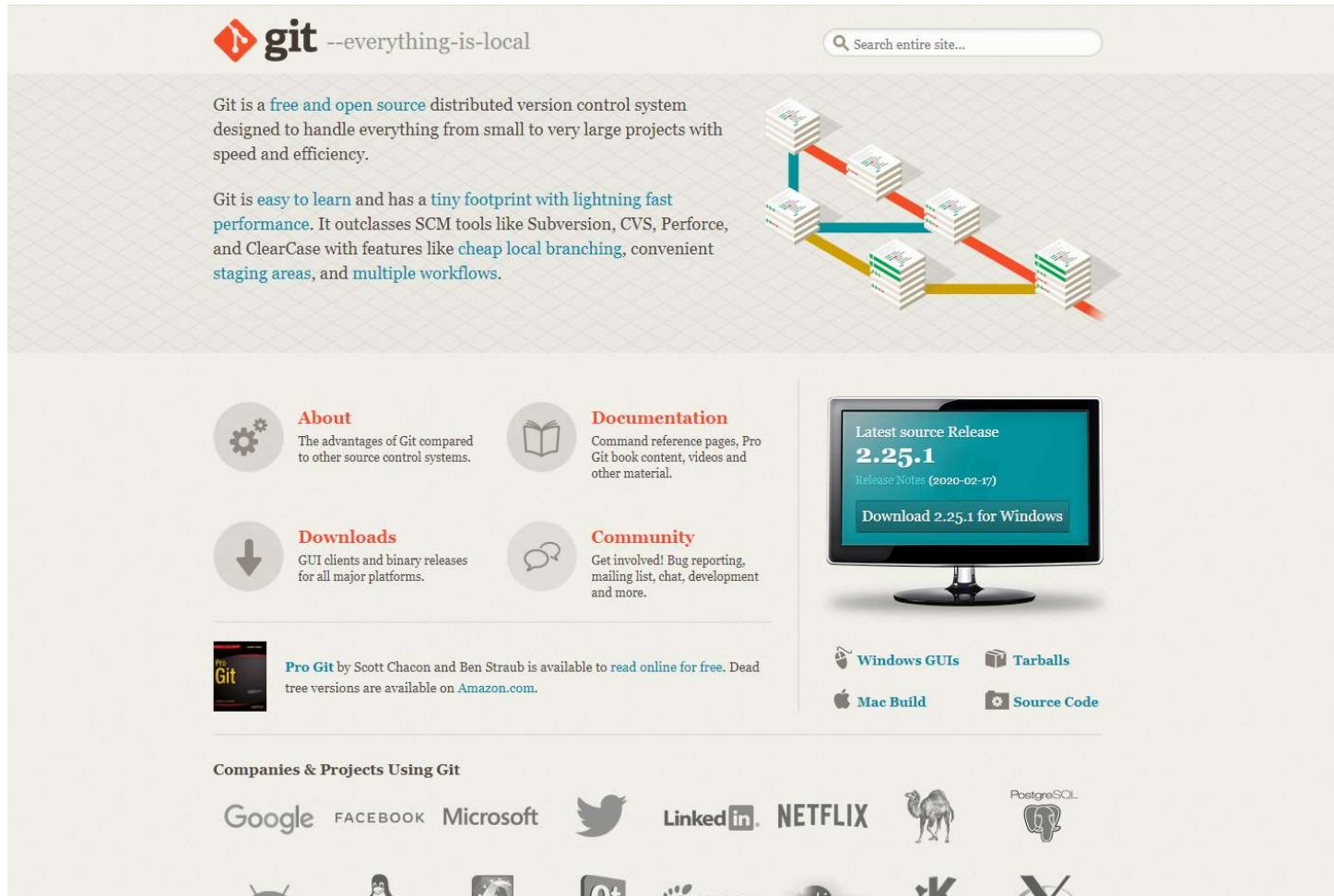
git

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**, distribuído sob os termos da versão 2 da **GNU General Public License**.

GIT - Instalação

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



The screenshot shows the Git website homepage. At the top, the Git logo is followed by the tagline "--everything-is-local". A search bar is located in the top right corner. The main content area describes Git as a free and open source distributed version control system. It highlights features like being easy to learn, having a tiny footprint, and lightning fast performance. A diagram illustrates the distributed nature of Git with multiple stacks of code connected by lines. Below this, there are four sections: "About" (advantages of Git), "Documentation" (command reference, Pro Git book), "Downloads" (GUI clients, binary releases), and "Community" (bug reporting, mailing list). A monitor displays the latest source release (2.25.1) and a button to download for Windows. At the bottom, there are links for Windows GUIs, Tarballs, Mac Build, and Source Code. The footer lists companies and projects using Git, including Google, Facebook, Microsoft, Twitter, LinkedIn, Netflix, and PostgreSQL.

git --everything-is-local

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
2.25.1
Release Notes (2020-02-17)
[Download 2.25.1 for Windows](#)

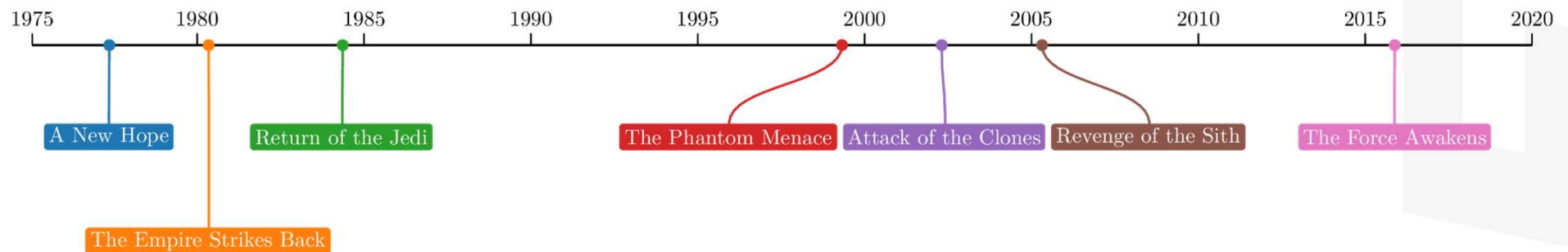
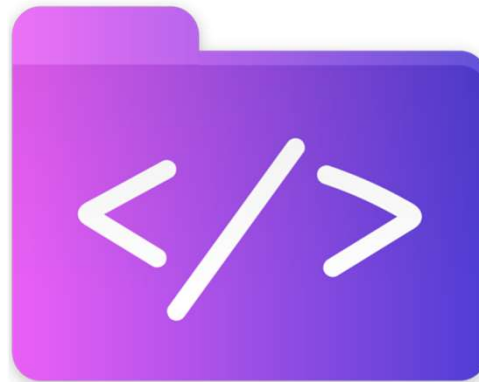
Pro Git by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).

Windows GUIs **Tarballs**
Mac Build **Source Code**

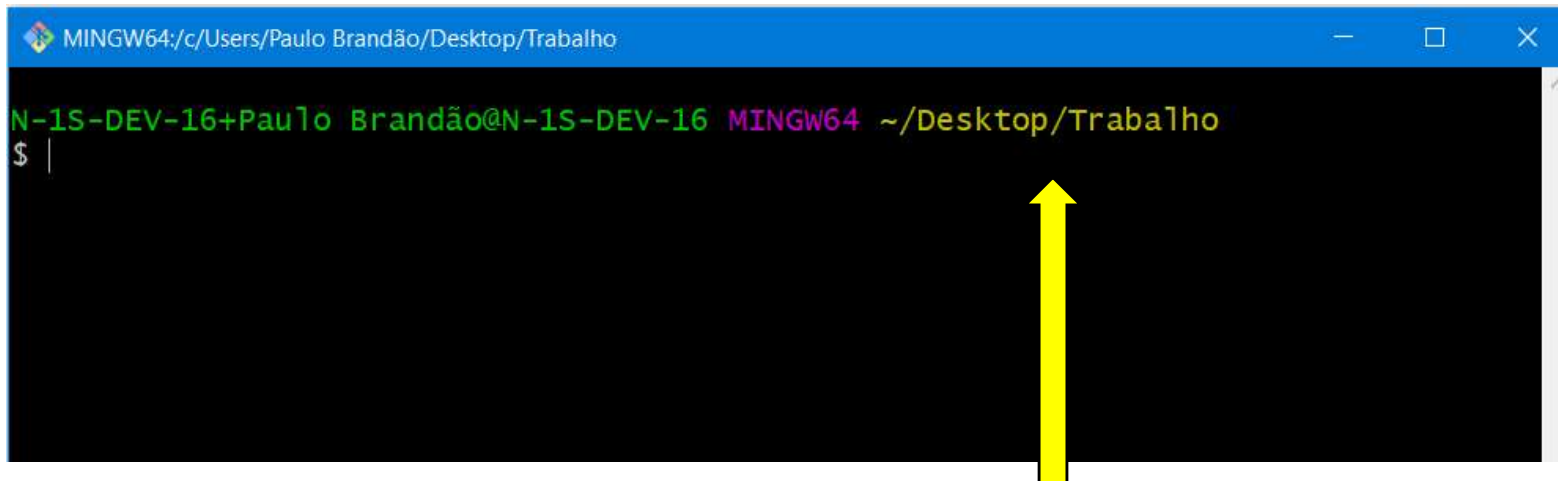
Companies & Projects Using Git
Google FACEBOOK Microsoft Twitter LinkedIn NETFLIX PostgreSQL

GIT – Repositório(pasta) + Timeline

SENAI-SP



Terminais - GitBash



A terminal window titled "MINGW64:/c/Users/Paulo Brandão/Desktop/Trabalho". The prompt shows the user is at the root of the "Trabalho" directory: "N-1S-DEV-16+Paulo Brandão@N-1S-DEV-16 MINGW64 ~/Desktop/Trabalho". A yellow arrow points to the "Trabalho" part of the path.

Diretório (pasta)



A terminal window titled "MINGW64:/c/Users/Paulo Brandão/Desktop/Trabalho". The prompt shows the user is on the "master" branch: "N-1S-DEV-16+Paulo Brandão@N-1S-DEV-16 MINGW64 ~/Desktop/Trabalho (master)". A cyan arrow points to the "(master)" part of the prompt.

Branch

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:

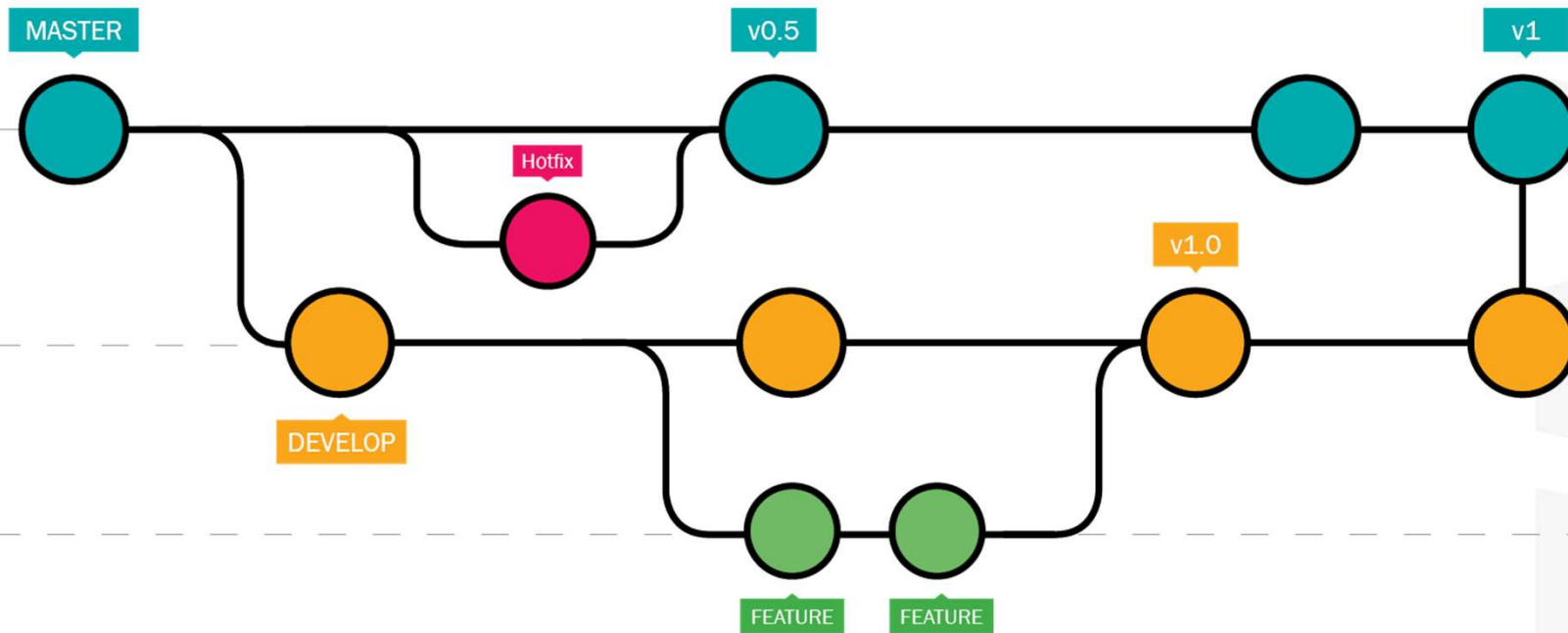
ou recuperar arquivo deletado:

```
git checkout hash nomedoarquivo.ext
```

```
git checkout hash .
```

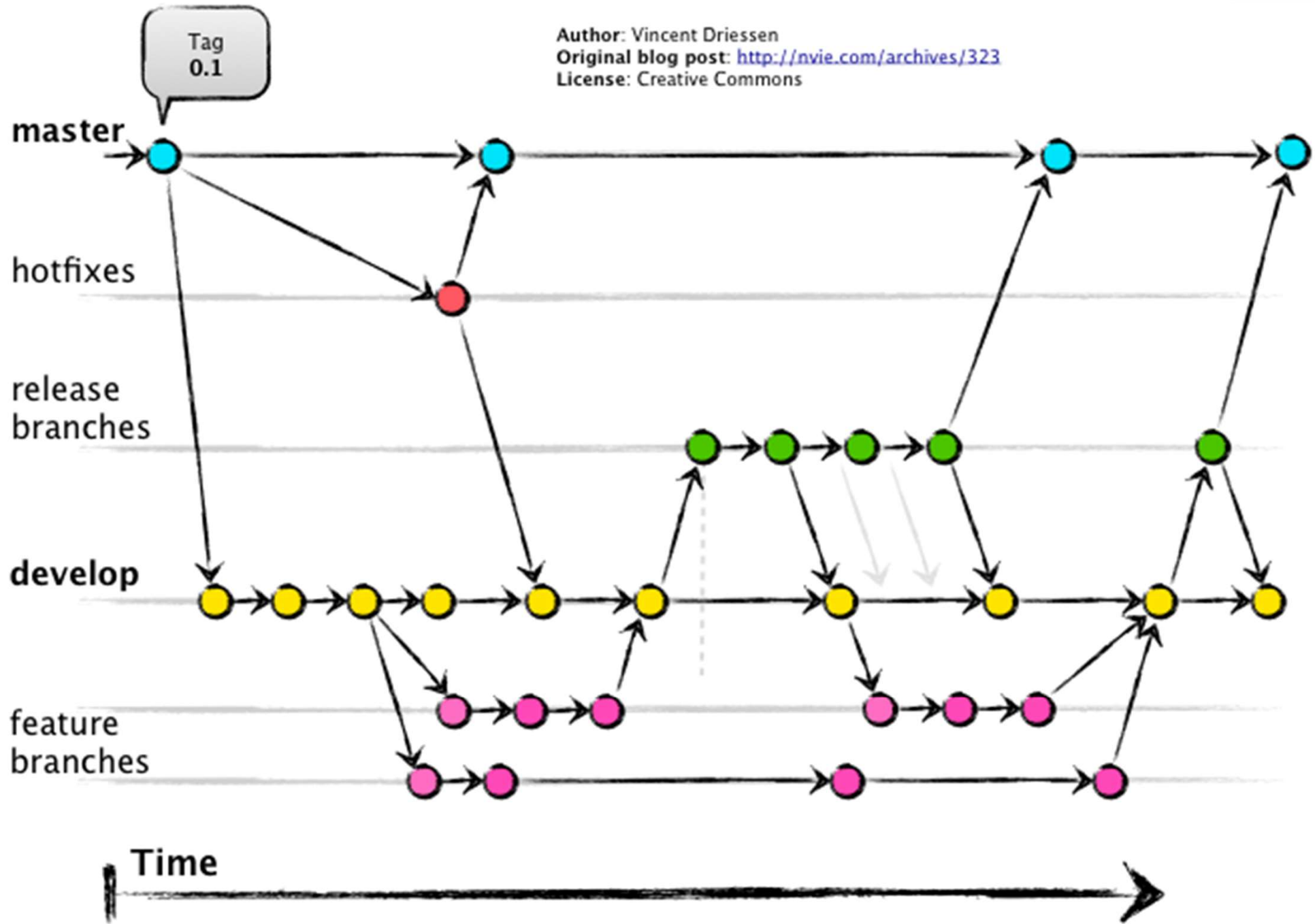
GIT FLOW

SENAI-SP



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**

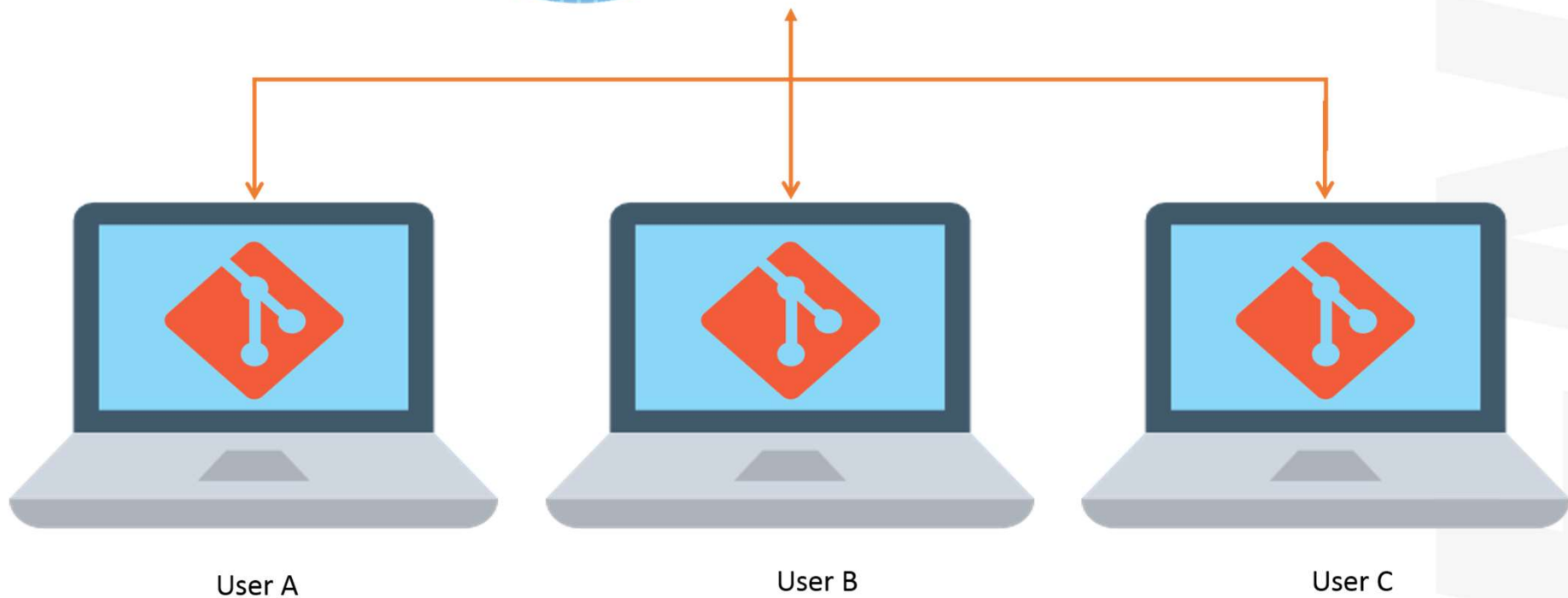


GitHub

GIT vs GITHUB

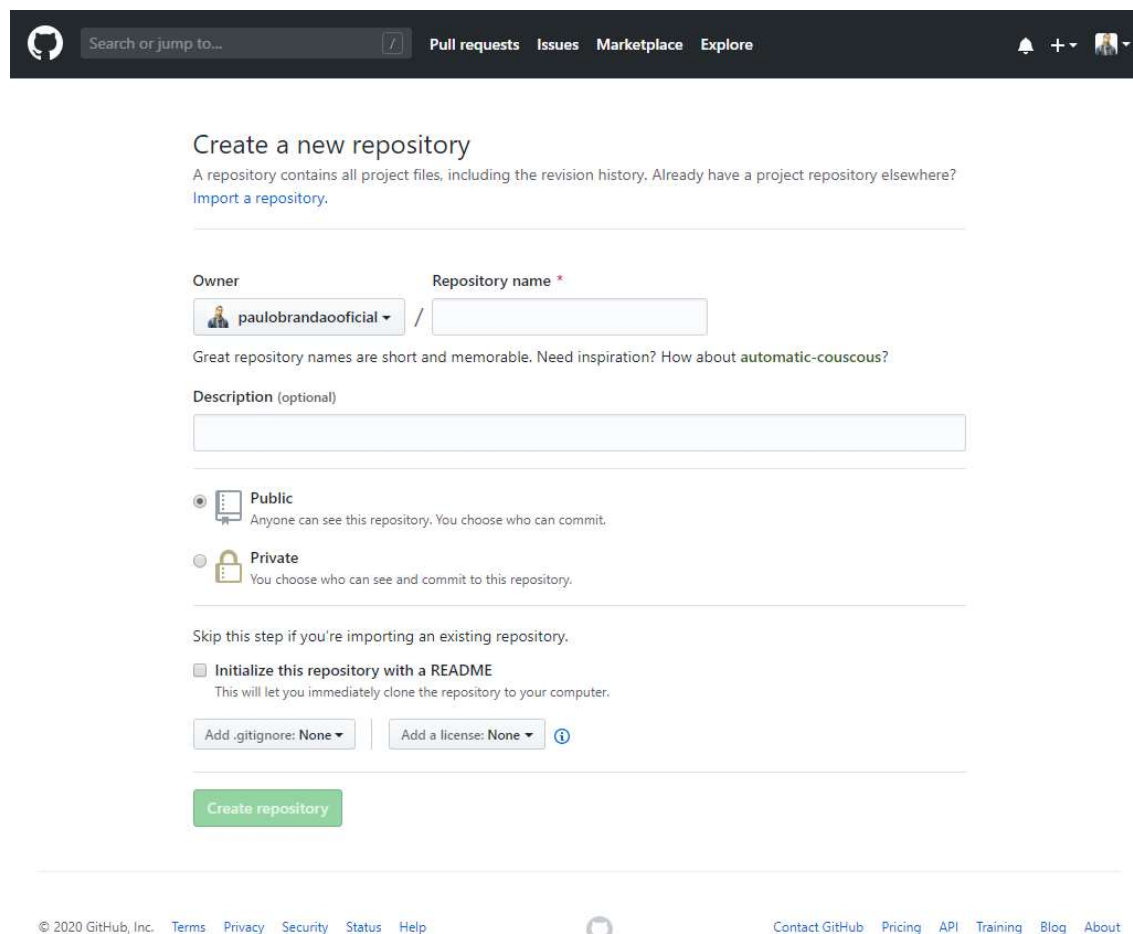


GitHub



GITHUB

- Criar um repositório :



The screenshot shows the GitHub interface for creating a new repository. At the top is a dark navigation bar with the GitHub logo, a search bar, and links for Pull requests, Issues, Marketplace, and Explore. Below this is the 'Create a new repository' section. It includes a sub-header, a brief explanation of repositories, and a link to 'Import a repository'. The main form has two input fields: 'Owner' (with a dropdown menu showing 'paulobrandaooficial') and 'Repository name *'. Below these fields is a tip about repository names. There is a 'Description (optional)' text area. The 'Visibility' section has two radio buttons: 'Public' (selected) and 'Private'. Below this is a checkbox for 'Initialize this repository with a README'. At the bottom of the form are two dropdown menus for '.gitignore' and 'License', and a green 'Create repository' button. The footer contains copyright information, links to Terms, Privacy, Security, Status, and Help, the GitHub logo, and links to Contact GitHub, Pricing, API, Training, Blog, and About.

Search or jump to... / Pull requests Issues Marketplace Explore

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner Repository name *

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

Description (optional)

☒ **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

© 2020 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Help](#) [Contact GitHub](#) [Pricing](#) [API](#) [Training](#) [Blog](#) [About](#)

GITHUB

- Adicionar um repositório local já existente:

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

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

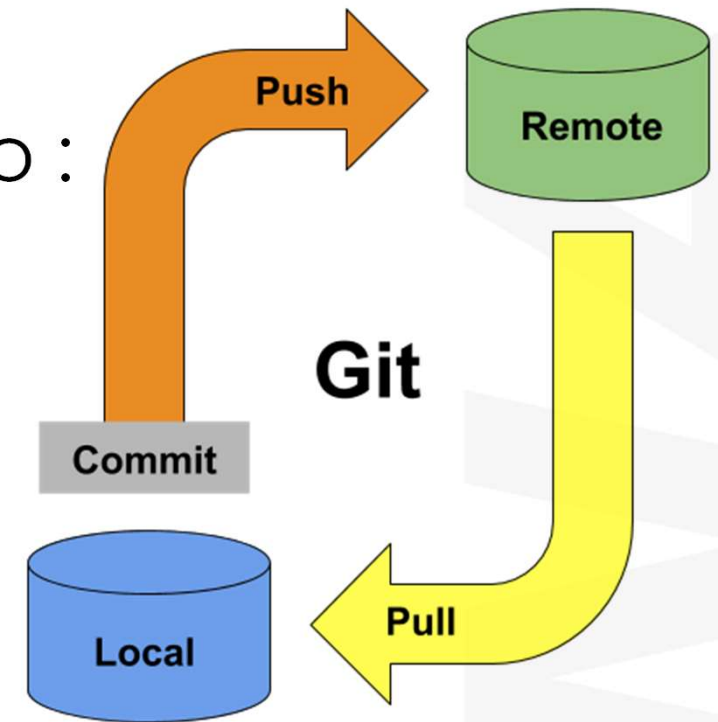
(-u = cria a main no github(remoto), usem somente no 1º push)

- Salvar login do github localmente:

```
git config credential.helper store
```

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



Github flow

