

Prof. Me. Maicon dos Santos Senac

Como você organiza suas mudanças?



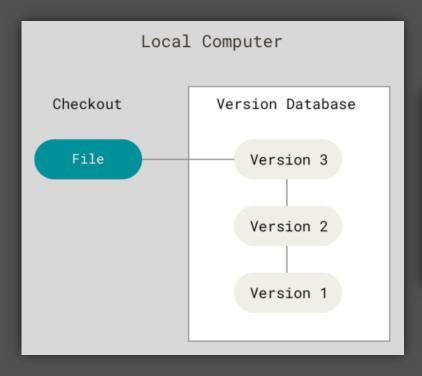
Última Versão Versão Final Versão Final **Final** Última Versão Versão Final 1 Última Versão 1 Final 4 Melhor e última Última Versão 2 Versão Final 2 versão de todas

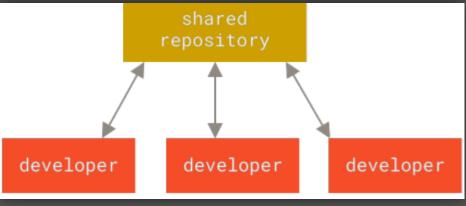
Esse é um modelo que muitas pessoas utilizando para armazenar e copiar arquivos movimentando os mesmos para outros diretórios. Esta abordagem é muito comum porque é muito simples, mas incrivelmente propensa a erros.

CONTROLE DE VERSÃO

Controle de versão é um sistema que registra alterações em um arquivo ou conjunto de arquivos ao longo do tempo para que você possa lembrar versões.

Controle de Versão Centralizado



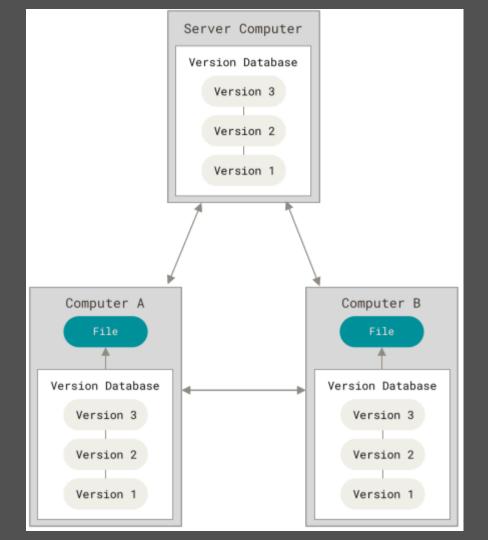


No entanto, esta configuração também tem algumas desvantagens graves. O mais óbvio é o ponto único de falha que o servidor centralizado representa.

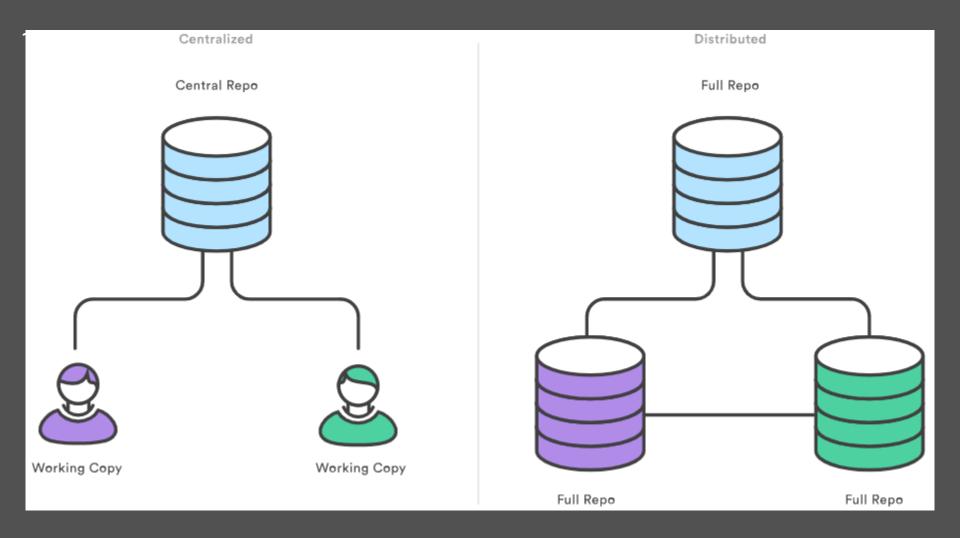
Se o servidor der problema por uma hora, ninguém pode colaborar ou salvar as alterações de versão.

Se o disco rígido do banco de dados central for corrompido, e backups apropriados não foram mantidos, você perde absolutamente tudo, toda a história do projeto.

Controle de Versão Distribuído



Um controle de versão distribuído permite que você configure vários tipos de fluxos de trabalho, colaborar com diferentes repositórios, que podem inclusive ser restaurados em caso de perda.



Uma Breve História do GIT

Desenvolvido/Criado em 2005 pelo Linus Torvalds (o homem conhecido por criar o núcleo, ou kernel, do SO Linux).

Hoje, GIT é um dos mais famosos sistemas de controle de versão de código aberto.

Desde sua criação, a ferramenta evoluiu e amadureceu para ser fácil de usar e ainda reter essas qualidades iniciais.

Incrivelmente rápido, é muito eficiente para projetos grandes, essa ferramenta tem um incrível sistema de ramos para desenvolvimento não linear.

Companies & Projects Using Git























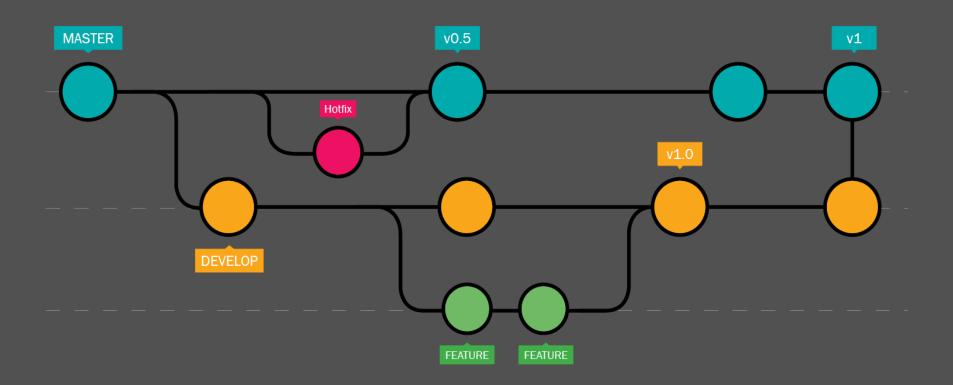


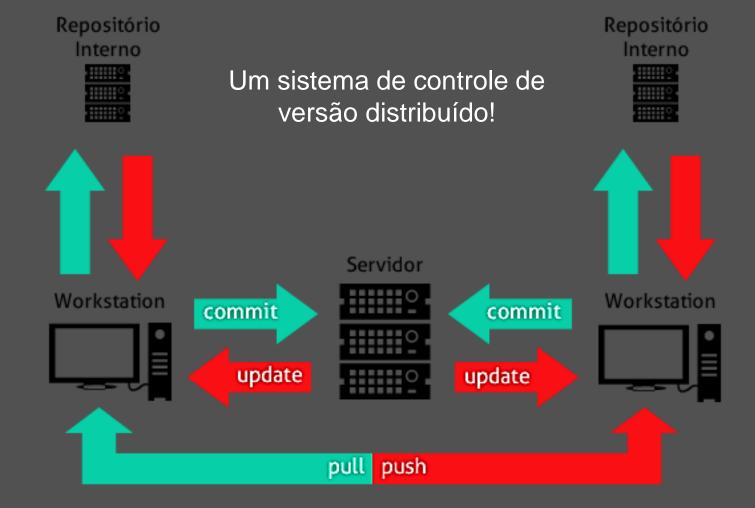




Fluxo de trabalho

- Você modifica arquivos no seu diretório de trabalho.
- Você adiciona arquivos modificados para uma área de transferência.
- Você salva os arquivos que estão na área de preparo e armazena esses arquivos de forma permanente para o diretório Git.







Instalação do GIT

- □ Instalação do Git
 - Para utilizarmos o Git, é fundamental torná-lo disponível em seu computador.
 - https://git-scm.com/download
 - □ Escolha o SO e mãos à obra!

Downloads



Older releases are available and the Git source repository is on GitHub.



Temos a possibilidade de utilizar GIT em qualquer SO

Download for Windows

Click here to download the latest (2.40.1) 64-bit version of Git for Windows. This is the most recent maintained build. It was released about 1 month ago, on 2023-04-25.

Other Git for Windows downloads

Standalone Installer

32-bit Git for Windows Setup.

64-bit Git for Windows Setup.

Portable ("thumbdrive edition")

32-bit Git for Windows Portable.

64-bit Git for Windows Portable.

Using winget tool

Install winget tool if you don't already have it, then type this command in command prompt or Powershell.

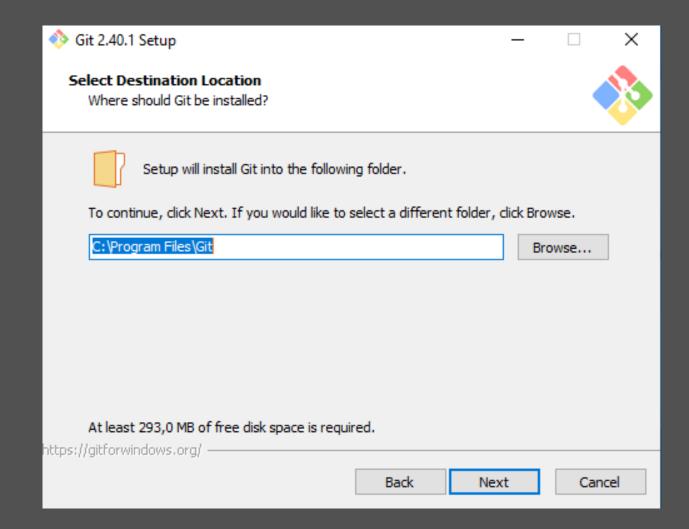
```
winget install --id Git.Git -e --source winget
```

The current source code release is version 2.40.1. If you want the newer version, you can build it from the source code.

Escolher a forma de instalação do GIT para Windows.



Basta seguir as instruções na tela, clicando em Next. Ao término, clique em Finish para concluir com êxito a instalação.



Select Components

Which components should be installed?



Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

	✓ Additional icons	\wedge
	On the Desktop	
	✓ Windows Explorer integration	
	·· ☑ Git Bash Here	
	☑ Git GUI Here	
	☑ Git LFS (Large File Support)	
	✓ Associate .git* configuration files with the default text editor	
	Associate .sh files to be run with Bash	
	Check daily for Git for Windows updates	
	(NEW!) Add a Git Bash Profile to Windows Terminal	v
1		

Current selection requires at least 292,9 MB of disk space.

https://gitforwindows.org/

Only show new options

Next

Back

Git 2.40.1 Setup	-	-		×	
Select Start Menu Folder Where should Setup place the program's shortcuts?			•		
Setup will create the program's shortcuts in the following Start Menu folder.					
To continue, click Next. If you would like to select a different fold	der, did	k Brow	se.		
Git		Brov	vse]	
Don't create a Start Menu folder					
Back	Next		Can	icel	



Choosing the default editor used by Git

Which editor would you like Git to use?



Use Vim (the ubiquitous text editor) as Git's default editor

The <u>Vim editor</u>, while powerful, <u>can be hard to use</u>. Its user interface is unintuitive and its key bindings are awkward.

Note: Vim is the default editor of Git for Windows only for historical reasons, and it is highly recommended to switch to a modern GUI editor instead.

Note: This will leave the 'core.editor' option unset, which will make Git fall back to the 'EDITOR' environment variable. The default editor is Vim - but you may set it to some other editor of your choice.

https://gitforwindows.org/

Back

Next

https://gitforwindows.org/

Back Next

Git 2.40.1 Setup

Adjusting your PATH environment

How would you like to use Git from the command line?



Ouse Git from Git Bash only

This is the most cautious choice as your PATH will not be modified at all. You will only be able to use the Git command line tools from Git Bash.

• Git from the command line and also from 3rd-party software

(Recommended) This option adds only some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools. You will be able to use Git from Git Bash, the Command Prompt and the Windows PowerShell as well as any third-party software looking for Git in PATH.

O Use Git and optional Unix tools from the Command Prompt

Both Git and the optional Unix tools will be added to your PATH. Warning: This will override Windows tools like "find" and "sort". Only use this option if you understand the implications.

https://gitforwindows.org/ –

Back

Next

Choosing the SSH executable

Which Secure Shell client program would you like Git to use?



• Use bundled OpenSSH

This uses ssh.exe that comes with Git.

Use external OpenSSH

NEW! This uses an external ssh.exe. Git will not install its own OpenSSH (and related) binaries but use them as found on the PATH.

Only show new options

https://gitforwindows.org/ =

Back

Next



Choosing HTTPS transport backend

Which SSL/TLS library would you like Git to use for HTTPS connections?



Use the OpenSSL library

Server certificates will be validated using the ca-bundle.crt file.

Ouse the native Windows Secure Channel library

Server certificates will be validated using Windows Certificate Stores. This option also allows you to use your company's internal Root CA certificates distributed e.g. via Active Directory Domain Services.

https://gitforwindows.org/

Back

Next

Configuring the line ending conversions

How should Git treat line endings in text files?



Checkout Windows-style, commit Unix-style line endings

Git will convert LF to CRLF when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Windows ("core.autocrlf" is set to "true").

Checkout as-is, commit Unix-style line endings

Git will not perform any conversion when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Unix ("core.autocrlf" is set to "input").

O Checkout as-is, commit as-is

Git will not perform any conversions when checking out or committing text files. Choosing this option is not recommended for cross-platform projects ("core.autocrlf" is set to "false").

https://gitforwindows.org/

Back

Next



Configuring the terminal emulator to use with Git Bash

Which terminal emulator do you want to use with your Git Bash?



• Use MinTTY (the default terminal of MSYS2)

Git Bash will use MinTTY as terminal emulator, which sports a resizable window, non-rectangular selections and a Unicode font. Windows console programs (such as interactive Python) must be launched via `winpty` to work in MinTTY.

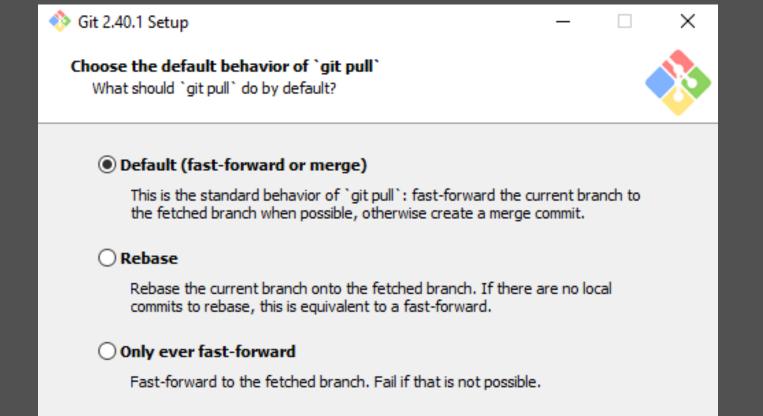
O Use Windows' default console window

Git will use the default console window of Windows ("cmd.exe"), which works well with Win32 console programs such as interactive Python or node.js, but has a very limited default scroll-back, needs to be configured to use a Unicode font in order to display non-ASCII characters correctly, and prior to Windows 10 its window was not freely resizable and it only allowed rectangular text selections.

https://gitforwindows.org/

Back

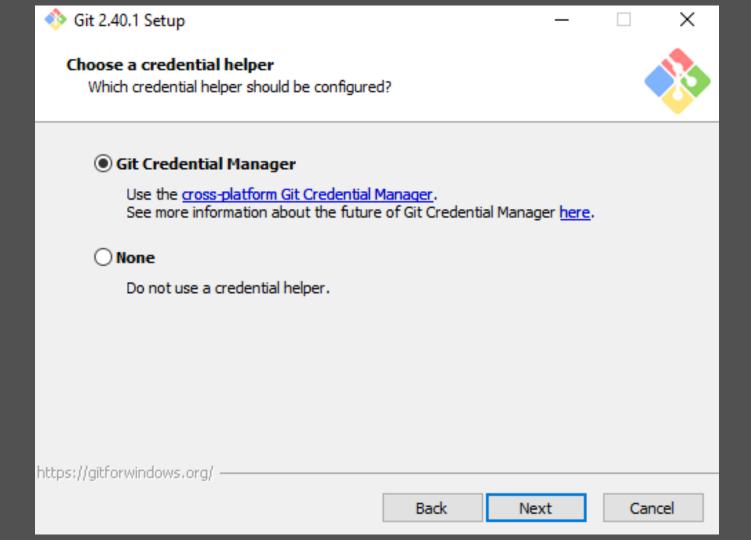
Next

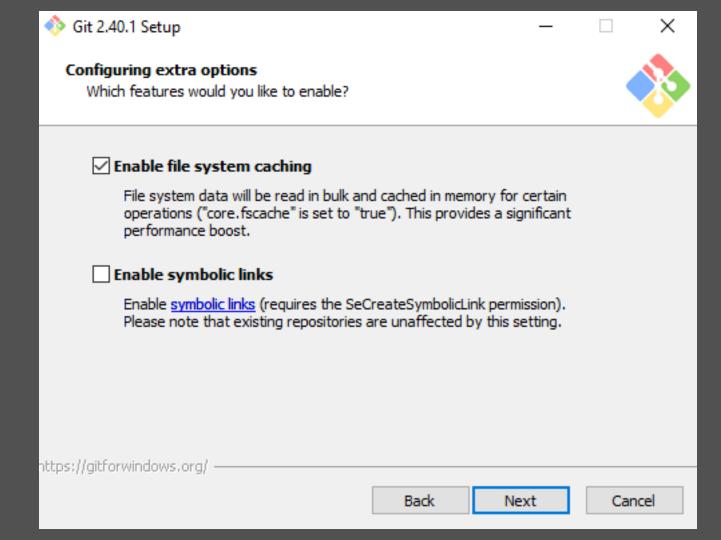


https://gitforwindows.org/

Next

Back







Configuring experimental options

These features are developed actively. Would you like to try them?



☐ Enable experimental support for pseudo consoles.

(NEW!) This allows running native console programs like Node or Python in a Git Bash window without using winpty, but it still has known bugs.

Enable experimental built-in file system monitor

(NEW!) Automatically run a <u>built-in file system watcher</u>, to speed up common operations such as `git status`, `git add`, `git commit`, etc in worktrees containing many files.

https://gitforwindows.org/

Only show new options

Back Install

Cancel

Completing the Git Setup Wizard



Setup has finished installing Git on your computer. The application may be launched by selecting the installed shortcuts.

Click Finish to exit Setup.

- Launch Git Bash
- ✓ View Release Notes

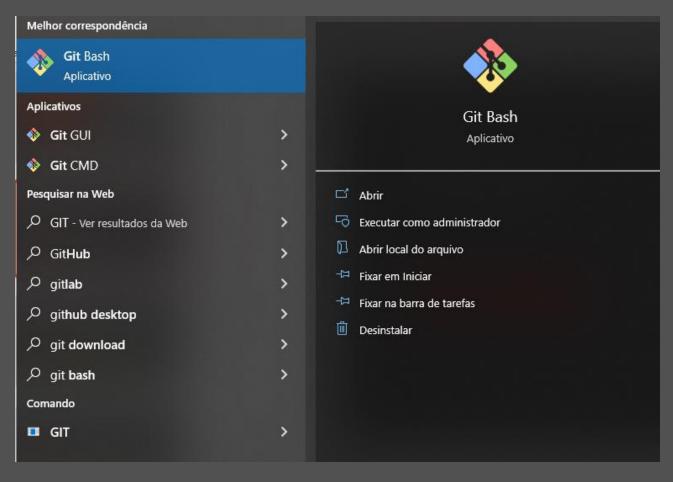
Only show new options

Finish

□ Uma vez que o GIT está instalado e configurado no seu dispositivo, vamos explorar os conceitos básicos do GIT e como começar a usar o GIT.

Inicializando o

GII BASH



42





Mãos à obra!

Para configurar inicialmente suas credenciais no GIT, é necessário executar os comandos abaixo:

git config --global user.name "usuario" git config --global user.email "email"

Pronto, o GIT já está com suas credenciais.

Para configurar inicialmente seu repositório no GIT, é necessário executar o comando abaixo:

git init

Pronto, o projeto já é um repositório Git.

Uma pasta chamada .git foi criada no diretório em que o comando foi executado.

Para que todos os arquivos sejam versionados, você pode adiciona-los através do seguinte comando:

git add.

Será adicionado todos arquivos do diretório que ainda não foram salvos em comando add no Git.

Caso desejar adicionar somente um arquivo, deverá informar o nome do arquivo. Você pode adicionar através do seguinte comando:

git add "nomedoarquivo"

Será adicionado somente o arquivo em que foi informado no comando add no Git.

Podemos ver a situação dos arquivos do repositório GIT, acompanhando o status de cada modificação com o comando a seguir:

git status

Será retornado o status dos arquivos que foram adicionados ou não foram ainda adicionados que constam no repositório.

Para gravarmos as mudanças no repositório Git, devemos executar o comando:

git commit -m "mensagem"

Será salvo uma versão do "arquivo.txt" no repositório do Git. Será exibido mensagem que foi adicionado um *commit* identificado e ele carrega a seguinte mensagem de *commit* "Arquivo inicial".

Um recurso muito utilizado do GIT é sua capacidade de permitir que criem vários ramos (branches) independentes dentro de um único projeto.

O branch padrão em qualquer projeto é sempre o master branch. Um novo ramo (branch) pode ser criado usando o seguinte comando:

git checkout -b nomebranch

A branch será criada no REPOSITÓRIO LOCAL.

Outra maneira de se criar uma nova branch é utilizando o comando Branch. Um novo ramo (branch) pode ser criado usando o seguinte comando:

git branch nomebranch

A nova branch será criada no REPOSITÓRIO LOCAL.

Para excluir uma Branch que já existe, é possível utilizar o comando abaixo informando o parâmetro –d antes do nome da branch.

git branch -d nomebranch

A branch será deletada do REPOSITÓRIO LOCAL.

Porém caso você desejar retornar para uma branch existente, pode ser utilizado apenas o comando checkout da seguinte forma:

git checkout master

Seu repositório local agora será utilizado com os arquivos do repositório Git master.

Agora que temos duas branches, para juntar duas branches, com diferentes *commits*, podemos unir as *branches* precisamos rodar o comando:

git merge outrabranch

Para que o git merge funcione, precisamos estar na *branch* que irá receber os *commits*. Ao rodar o comando a branch será atualizada

Porém, caso você queira obter a cópia de um repositório do Git já existente:

git clone url

Pronto, será obtido uma cópia/clone com todos os dados deste repositório remoto no diretório onde você executou o comando.

Para tornar o *branch* disponível para outros usuários, você terá que **EMPURRAR** para o repositório remoto. Para fazer isso, use o seguinte comando:

git push origin nomebranch

A branch será enviada para o repositório remoto Git.

Caso você queira atualizar seu diretório de trabalho local para uma versão mais recente do repositório remoto, você pode **PUXAR** com o simples comando:

git pull git fetch

Será atualizado o repositório local com os dados do repositório remoto do Git.

Desfaz o commit do repositório remoto. Realiza a reversão de um commit de forma fácil e segura.rever

git revert <commit>

O comando git revert sempre precisa de uma refência de commit para que seja revertido. Caso não seja informado, deverá passar o último commit como referência para seu correto funcionamento.

Desfaz as alterações nos arquivos de trabalho. Permite limpar por completo as alterações que não estão no repositório.

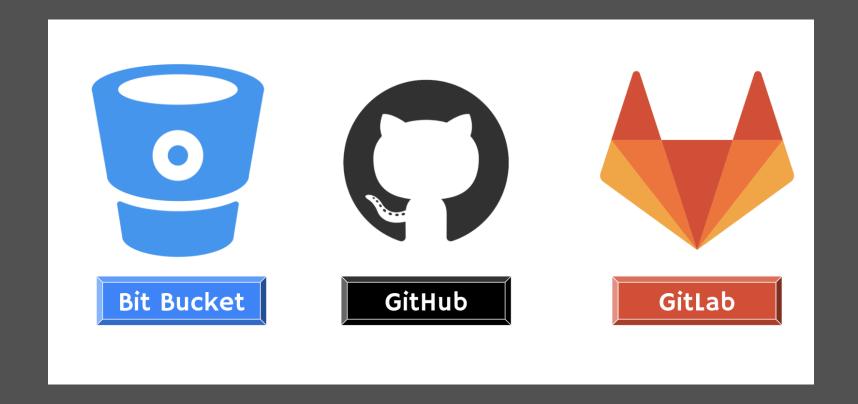
git reset

Para que o git reset funcione é necessário que exista arquivos na branch adicionados.

Para visualizar o histórico de commits existentes, basta utilizar o comando:

git log

Será listado todos os commits que foram realizados no repositório do Git. É um histórico!



NA PRÁTICA

GITHUB Desktop

GitHub, uma aplicação Web que possibilita criar de repositórios Git e de forma visual.



GitHub é um site no qual você pode fazer criação e atualização de projetos Git. Crie uma conta no GitHub.

Acesse: https://desktop.github.com/



IMPORTANTE!!!

A diferença entre GIT e GITHUB é que o GIT é uma ferramenta para versionar projetos, enquanto o GITHUB é a aplicação (site) no qual você colocará esses projetos Git versionados.

ATIVIDADES

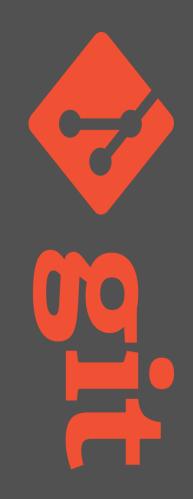
1. Atividades com git:

- ✓ Crie uma conta no GitHub e um novo repositório;
- ✓ Faça o clone do repositório criado para o seu computador;
- ✓ Crie um arquivo README.md e faça o commit para o repositório;
- ✓ Faça o push das mudanças para o repositório no GitHub.

- ✓ Crie uma nova branch no seu repositório;
- ✓ Faça algumas mudanças em um arquivo existente no seu repositório;
- ✓ Faça o commit das mudanças na nova branch;
- ✓ Abra um pull request para mesclar a nova branch com a branch principal do repositório;
- ✓ Peça para um colega de equipe revisar o seu pull request;
- ✓ Se houver comentários ou alterações sugeridas, faça as mudanças necessárias e atualize o pull request;
- ✓ Depois que o pull request for aprovado, faça o merge da nova branch com a branch principal do repositório.

<u>Aprendemos</u>

- ✓ Instalar o Git,
- ✓ Criar repositórios,
- ✓ Adicionar arquivos,
- ✓ Verificar históricos,
- ✓ Visualizar estados,
 - ✓ Fazer commits,
- ✓ Obter e Enviar versões,
 - ✓ Criar branches,
- ✓ Navegar entre branches.



Obrigado!



Prof: Me. Maicon dos Santos