

Como instalar o git

Vá para o site: <https://git-scm.com/> e baixe a última versão. A instalação irá perguntar várias coisas, se estiver na dúvida em o que escolher, não mexa em nada e apenas clique em “Next”.

Por padrão é instalado o “bash” que é uma linha de comandos muito útil, e para abrir ela em qualquer diretório, basta clicar com o botão direito na pasta e selecionar “Git Bash Here”, e é nela que você entrará os comandos git.

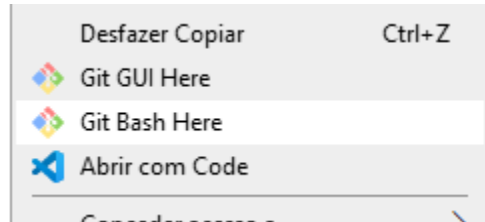


Figura 1: Opção "Git Bash Here"

Como utilizar o git

Navegue até a pasta do projeto e execute o comando: “**git init**” para inicializar o repositório git, uma pasta oculta “.git” será criada e nela será armazenado o repositório.

Crie os arquivos do seu projeto.

Cada mudança do repositório é chamada de “commit”. Para colocar um arquivo no commit, é necessário rodar o comando “**git add <nome do arquivo>**”. Para adicionar todos os arquivos do projeto, rode “**git add .**”.

Os arquivos adicionados são chamados de “staged files”, stage significa palco, é como se os arquivos que vão ser “commitados” estejam em um palco, e os arquivos que não serão commitados (unstaged files), não estão no palco. Para saber quais arquivos foram modificados e quais estão staged ou unstaged, rode o comando: “**git status**”. Este comando também te ensina como remover arquivos do “stage” (palco).

Para criar um commit, simplesmente rode o comando: **git commit -m “Mensagem do commit”**. É muito importante que a mensagem explique bem e resumidamente quais alterações foram feitas no código. Para uma utilização mais profissional do git também, é recomendável fazer pequenos commits frequentemente a cada modificação do código.

É possível criar “branches” também, que são ramificações do projeto. Vamos supor que você quer adicionar uma nova funcionalidade ao seu programa, mas não quer arriscar modificar o código original. Para isso que servem as branches: você cria uma nova linha do tempo para o seu projeto, trabalha na nova funcionalidade, e quando ela estiver perfeitamente funcional, aí sim pode fazer o “merge” que mescla a nova linha do tempo com a original, que é chamada de “master”. Caso a funcionalidade atrapalhe o programa, você descarta a linha do tempo criada e volta a trabalhar na master, que ficou inalterada. Para criar uma nova

“branch” rode o comando: **git checkout -b <nome da branch>**. Para mudar de branch, rode o mesmo comando só que sem o **-b**.

Como salvar um repositório no GitHub


Crie um novo repositório no GitHub, não inicialize o repositório com README.md, nem adicione o .gitignore ou license. Assim que você criar, irá aparecer uma mensagem com instruções. Siga a instrução que começa com “...or push an existing repository from the command line”. Se for a primeira vez sua utilizando o Git e GitHub, aparecerá uma janela pedindo para se logar no GitHub. Na próxima vez que você utilizar, as suas credenciais já estarão salvas no computador e não será necessário se logar novamente. Caso queira remover suas credenciais do computador, vá em Painel de Controle → Contas de Usuário → Gerenciar suas credenciais → Credenciais do Windows.

O nome “origin” do comando utilizado para salvar o repositório no GitHub, existe pois ao criar um clone de um repositório remoto na sua máquina local, o git cria um nome para você, quase sempre esse nome é “origin”, esse nome nada mais é do que um encurtamento da URL do repositório remoto, para você não precisar toda vez digitar a URL completa.

Create a new repository

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

Owner

 pauloeps ▾

Repository name *

/ nome ✓

Great repository names are short and memorable. Need inspiration? How about **reimagined-funicular**?

Description (optional)



Public

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



Captura Retangular



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

Figura 2: Criação de um repositório no GitHub

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


```
git remote add origin https://github.com/pauloeps/nome.git
git push -u origin master
```


Figura 3: Instrução ao criar um novo repositório

Um Pull Request é uma maneira de alertar o dono do repositório que você deseja fazer mudanças no código. Isso permite eles fazerem a revisão do seu código antes de fazer um “merge” com a master branch.

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

 base: **master** ... compare: **mnelson** ✓ **Able to merge.** These branches can be automatically merged.



Added my name file

Write

Preview

Markdown supported

Edit in fullscreen

Leave a comment

Attach images by dragging & dropping or [selecting them](#).

Figura 4: Pull Request

Added my name file #1

 **Open** cubeton wants to merge 1 commit into **master** from **mnelson**



 Conversation **0**  Commits **1**  Files changed **1**





cubeton commented a minute ago

Owner

Adding a new file with my name as the title name.

  Added my name file 166b73a

Add more commits by pushing to the **mnelson** branch on **cubeton/git101**.

  **This branch is up-to-date with the base branch**
Merging can be performed automatically.


 **Merge pull request** You can also [open this in GitHub for Mac](#) or view [command line instructions](#).

Figura 5: Como o dono do repo vê seu Pull Request

Todo commit possui um código hash que é um identificador único. Para desfazer mudanças, basta usar o comando: **git revert <código hash do commit>** que o código vai “voltar no tempo” para aquele commit específico.

Para transferir mudanças de código do repositório remoto no seu repositório local, utilize o comando **git pull**. Para transferir mudanças de código do seu repositório local para o repositório remoto, utilize o comando **git push**.

Ao utilizar um repositório https, o git pode ficar pedindo suas credenciais toda vez que você fizer um push ou pull. Para resolver isso, digite o seguinte comando: **git config --global credential.helper cache -timeout=600**. O valor de timeout pode ser modificado de acordo com a sua preferência. É importante colocar o timeout pois sua senha será armazenada em um arquivo de texto no seu computador, sem criptografia, então é um risco de segurança se você esquecer de apagar ela.

Para fazer um merge de uma branch com a master branch, utilize: **git merge <nome da branch> -m “mensagem explicando o merge”**

Muitas vezes você pode achar um repositório online e desejar cloná-lo em sua máquina, para fazer isso basta rodar o famoso comando: **git clone <url do repositório>**.

Outra coisa importante é o arquivo **.gitignore** onde você coloca o nome de outros arquivos e diretórios que não deseja que sejam adicionados no repositório. Ele é útil pois alguns arquivos não são necessários de se adicionar no repositório e só estariam ocupando espaço e tempo de processamento à toa ao se fazer os commits.

É recomendável também criar um arquivo **README.md** que explicará sobre o projeto em questão. Isso é especialmente útil ao criar repositórios públicos que estarão disponíveis para outras pessoas colaborarem, pois elas precisam saber sobre o que o projeto se trata.

Algumas IDEs como o Visual Studio Code, possuem integração com o Git, onde é possível através da própria interface gráfica fazer os comandos Git, sem precisar digitá-los no terminal.

Para aprender mais sobre Git, consulte a documentação no site oficial: <https://git-scm.com/doc>.

Para aprender mais sobre GitHub, consulte a ajuda do site oficial: <https://help.github.com/en/github>