

Git e GitHub

Aprenda sobre o sistema de controle de versões, Git e como ele funciona com o GitHub.

Sobre controle de versão e o Git

Um sistema de controle de versão, ou **VCS**, monitora o histórico de alterações à medida que as pessoas e equipes colaboram em projetos em conjunto. Como os desenvolvedores fazem alterações no projeto, qualquer versão anterior do projeto pode ser recuperada a qualquer momento.

Os desenvolvedores podem revisar o histórico do projeto para descobrir:

- Quais alterações foram feitas?
- Quem fez as alterações?
- Quando as alterações foram feitas?
- Por que as alterações foram necessárias?

Os VCSs fornecem a cada colaborador uma visão unificada e consistente de um projeto, evidenciando o trabalho que já está em andamento.

Em um sistema de controle de versão distribuído, cada desenvolvedor tem uma cópia completa do projeto e do histórico do projeto. Ao contrário dos sistemas de controle de versão centralizados conhecidos, os **DVCSs** não precisam de uma conexão constante com um repositório central. **Git** é o sistema de controle de versão distribuída mais popular. O **Git** é comumente usado para o desenvolvimento de software de código aberto e comercial, com benefícios significativos para indivíduos, equipes e negócios.

- O **Git** permite que os desenvolvedores vejam toda a linha do tempo das suas alterações, decisões e progressão de qualquer projeto em um só lugar. Desde o momento em que acessam a história de um projeto, o desenvolvedor tem todo o contexto de que precisa para entender e começar a contribuir.
- Os desenvolvedores trabalham em todos os fusos horários. Com um DVCS como o **Git**, a colaboração pode acontecer a qualquer momento enquanto mantém a integridade do código-fonte. Ao usar branches, os desenvolvedores podem propor alterações no código de produção.
- As empresas que usam o Git podem derrubar as barreiras de comunicação entre equipes e mantê-las focadas em fazer o melhor trabalho. Além disso, o Git possibilita alinhar especialistas em todos os negócios para colaborar em grandes projetos.

Instalando o Git

Antes de começar a usar o Git, você tem que torná-lo disponível em seu computador. Mesmo se ele já tiver sido instalado, é provavelmente uma boa ideia atualizar para a versão mais recente. Você pode instalá-lo como um pacote ou através de outro instalador, ou baixar o código fonte e comprá-lo.

Instalando no Windows

Para instalar o Git no Windows basta ir ao <http://git-scm.com/download/win> e o download começará automaticamente.

Instalando no Mac

Existem várias maneiras de instalar o Git em um Mac. O mais fácil é provavelmente instalar as ferramentas de linha de comando Xcode. No Mavericks (10,9) ou acima, você pode fazer isso simplesmente rodando **git** a partir do Terminal pela primeira vez. Se você não tiver o Git instalado, ele irá pedir-lhe para instalá-lo.

Se você quiser uma versão mais atualizada, você também pode instalá-lo através de um instalador binário. Um instalador OSX Git é mantido e disponível para download no site do Git, pelo <http://git-scm.com/download/mac>.

Instalando no Linux

Se você deseja instalar o Git no Linux através de um instalador binário, você pode geralmente fazê-lo através da ferramenta básica de gerenciamento de pacotes que vem com sua distribuição.

Se você usar Fedora por exemplo, você pode usar o **yum** digite no seu terminal:

```
$ sudo yum install git-all
```

Se você usar uma distribuição baseada em **Debian** como o **Ubuntu**, use o **apt-get**:

```
$ sudo apt-get install git-all
```

Para mais opções de instruções de como instalar o Git em outros vários sistemas Unix, veja na página do Git, em <http://git-scm.com/download/linux>.

Configurando o Git

Após a instalação precisamos de fazer algumas configurações básicas no git como o nosso **username** e **email**.

Configurando username

Abra o seu terminal ou command line e digite:

```
git config --global user.name "seu nome"
```

Para confirmar:

```
git config --global user.name
```

Configurando email

Abra o seu terminal ou command line e digite:

```
git config --global user.email "seu email"
```

Para confirmar:

```
git config --global user.email
```

Configurando nome da branch principal

```
git config --global init.defaultBranch main
```

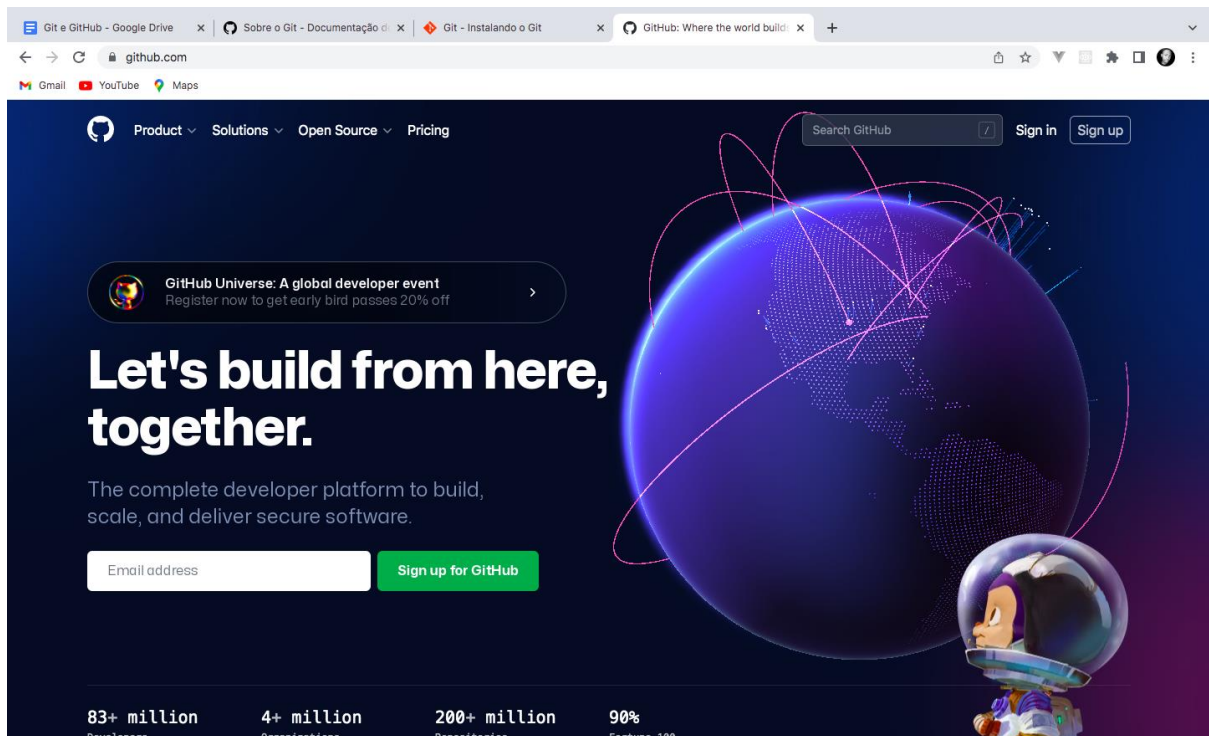
Como Funciona o GitHub

GitHub hospeda repositórios do Git e fornece aos desenvolvedores ferramentas para enviar um código por meio das funcionalidades de linha de comando, problemas(discussões encadeadas), pull requests, revisão de código ou o uso de uma coleção de aplicativos grátis e para compra em GitHub Marketplace.

Criar Conta no Github

É possível criar uma conta pessoal, que serve como sua identidade em GitHub.com.

Para tal visite o site oficial do GitHub <https://github.com/> no Clicando em SignUp poderá ser direcionado para página de registo e ali é só seguir os passos para criação de conta.



Comandos básicos do Git

Para usar o Git, os desenvolvedores usam comandos específicos para copiar, criar, alterar e combinar código. Esses comandos podem ser executados diretamente na linha de comando ou usando um aplicativo como GitHub Desktop. Aqui estão alguns comandos comuns para usar o Git:

- **git init** inicializa um novo repositório Git e começa a acompanhar um diretório existente. Ele adiciona uma subpasta oculta dentro do diretório existente que contém a estrutura de dados interna necessária para o controle de versão.
- **git clone** cria uma cópia local de um projeto que já existe remotamente. O clone inclui todos os arquivos, histórico e branches do projeto.
- **git add** prepara uma alteração. O Git controla as alterações na base de código de um desenvolvedor, mas é necessário testar e tirar um instantâneo das alterações para incluí-las no histórico do projeto. Este comando executa o teste, a primeira parte do processo de duas etapas. Todas as mudanças que são testadas irão tornar-se parte do próximo instantâneo e parte do histórico do projeto. O teste e o commit separados dão aos desenvolvedores total controle sobre o histórico do seu projeto sem alterar como eles codificam e funcionam.
- **git commit** salva o instantâneo no histórico do projeto e conclui o processo de controle de alterações. Em suma, um commit funciona como tirar uma

foto. Qualquer item que tenha sido preparado com `git add` passará a fazer parte do instantâneo com `git commit`.

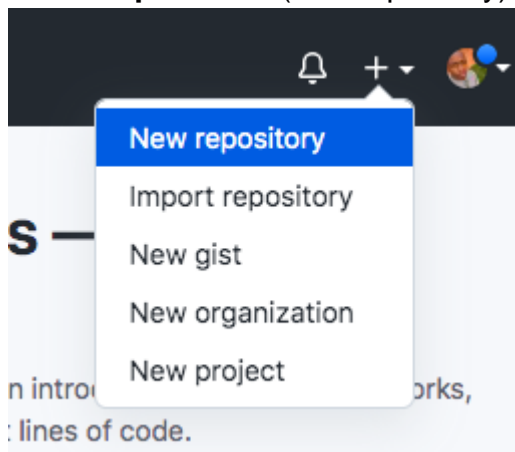
- `git status` mostra o status das alterações como não controladas, modificadas ou preparadas.
- `git branch` mostra os branches que estão sendo trabalhados localmente.
- `git merge` mescla as linhas de desenvolvimento. De modo geral, esse comando é usado para combinar alterações feitas em dois branches distintos. Por exemplo, um desenvolvedor faria merge quando quisesse combinar alterações de um branch de recurso no branch principal para implantação.
- `git pull` atualiza a linha de desenvolvimento local com atualizações do equivalente remoto. Os desenvolvedores usam este comando se um colega fez commits em um branch remoto, e eles gostaria de refletir essas alterações no seu ambiente local.
- `git push` atualiza o repositório remoto com todos os commits feitos localmente em um branch.

Para obter mais informações, confira o [guia de referência completo de comandos do Git](#).

Exemplo: Contribuir para um repositório existente


Para começar vamos aprender primeiro como criar **um repositório no GitHub**. Para tal faça login na sua conta se ainda não estiver logado.

1. No canto superior direito do menu clicar no botão mais (+) e depois **novo repositório** (new repository).



2. Dê um nome ao seu repositório
3. Dê uma descrição ao sobre o projecto (opcional)
4. Selecciona a opção create README.md
5. Criar repositório (create repository)


Owner * **Repository name ***


 IsaacNdala / nome-do-repositorio ✓

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

Description (optional)

Uma descrição do seu projecto

☒  **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 ▾

This will set  **main** as the default branch. Change the default name in your [settings](#).

 You are creating a public repository in your personal account.

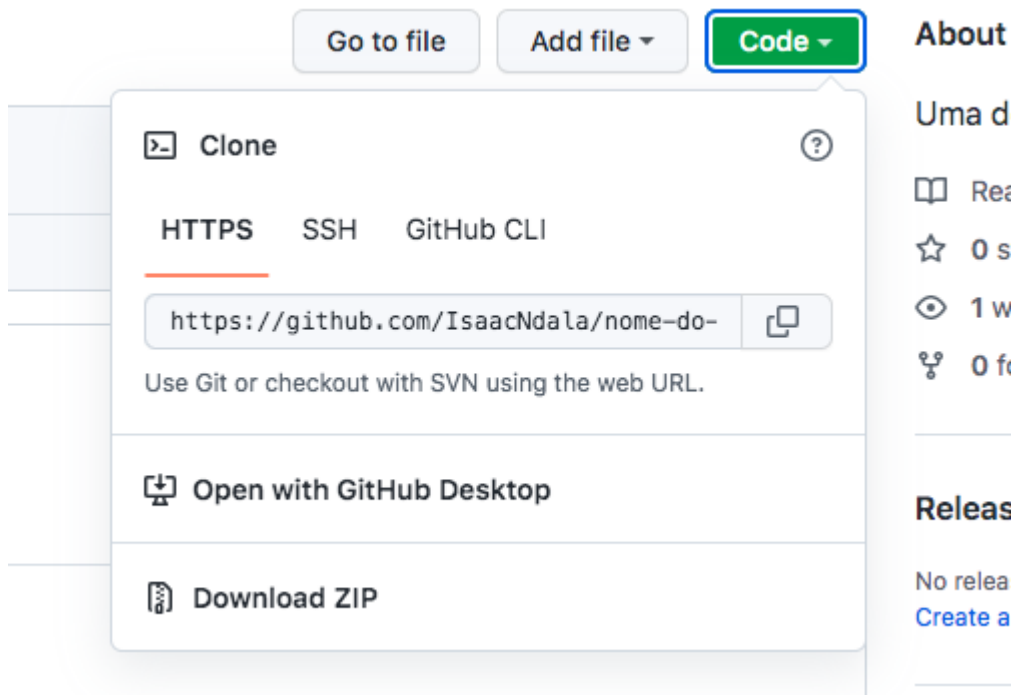
[Create repository](#)

Agora abra o seu terminal ou linha comando navegue para o seu ambiente de trabalho.

```
● Isaac-MacBook-Air:~ isaacndala$ cd desktop
○ Isaac-MacBook-Air:desktop isaacndala$
```

O comando `cd` permite navegar sobre diretórios

Agora copie a url do seu projecto no seu github:



Com isso feito vamos começar a usar os comandos do Git, começando por baixar o repositório usando o comando **git clone** e o endereço copiado no repositório.

```
Isaac-MacBook-Air:desktop isaacndala$ git clone https://github.com/IsaacNdala/nome-do-repositorio.git
```

Depois de pressionar ENTER será feito o download do projecto no ambiente de trabalho. agora navegamos para a pasta do projecto.

```
Isaac-MacBook-Air:desktop isaacndala$ cd nome-do-repositorio
Isaac-MacBook-Air:nome-do-repositorio isaacndala$
```

Agora vamos criar um arquivo na nossa pasta usando o comando **touch index.html** (**Window echo index.html**) ou podemos mesmo criar de forma manual como aprendemos anteriormente.

Em seguida vamos ver o estado das alterações no nosso repositório usando o **git status**

```
● Isaac-MacBook-Air:nome-do-repositorio isaacndala$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        index.html

nothing added to commit but untracked files present (use "git add" to track)
○ Isaac-MacBook-Air:nome-do-repositorio isaacndala$
```

A informação que nos dá é que nós criamos um arquivo e precisamos de adicionar usando o comando **git add**. Vamos então adicionar:

```
● Isaac-MacBook-Air:nome-do-repositorio isaacndala$ git add .
○ Isaac-MacBook-Air:nome-do-repositorio isaacndala$
```

Com isso nos adicionamos o arquivo no repósitorio, e note o ponto (.) depois do comando, nos poderia usar **git add index.html**, mas usando o ponto isso indica que adicionaria todos os arquivos no caso se criássemos vários. Depois disso vamos novamente verificar o estado:

```
● Isaac-MacBook-Air:nome-do-repositorio isaacndala$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   index.html

○ Isaac-MacBook-Air:nome-do-repositorio isaacndala$
```

Ao verificarmos o estado nos informa que precisamos de fazer um **commit** dessa nova alteração.

```
● Isaac-MacBook-Air:nome-do-repositorio isaacndala$ git commit -m "Adiciona o arquivo index.html"
[main 2043696] Adiciona o arquivo index.html
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 index.html
○ Isaac-MacBook-Air:nome-do-repositorio isaacndala$
```

Ao executarmos o comando **git commit** nos criamos por assim dizer uma nova versão do nosso project. E o para **-m** serve para descrever o a alteração feita no projecto. Até esse momento o GitHub não sabe das alterações que fizemos em nosso projecto para tal precisamos de executar o comando **git push**.


```
● Isaac-MacBook-Air:nome-do-repositorio isaacndala$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 294 bytes | 294.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/IsaacNdala/nome-do-repositorio.git
   bccf3d0..2043696  main -> main
```

A partir desse momento nós já temos o nosso código no GitHub e se visitarmos o nosso repositório nesse momento veremos o novo arquivo que adicionamos como mostra a imagem.

IsaacNdala Adiciona o arquivo index.html		2043696 13 minutes ago	🕒 2 commits
📄 README.md	Initial commit	1 hour ago	
📄 index.html	Adiciona o arquivo index.html	13 minutes ago	

Parabens você adicionou o seu código no GitHub. Nota que ao fazer o clone do projecto automaticamente você já inicializou o um repositório usando o comando **git init**, para saber como iniciar um novo repositório no seu computador e publicar no gitHub visite <https://docs.github.com/pt/get-started/using-git/about-git> no segundo exemplo apresenta como pode fazer.

Links Uteis

Sobre o Git:

<https://docs.github.com/pt/get-started/using-git/about-git>

Começando - Instalando o Git: <https://git-scm.com/book/pt-br/v2/Come%C3%A7ando-Instalando-o-Git>