

Documentação para auxílio Git

Nesta documentação não abordarei o passo a passo da instalação do mesmo. Para isso, acesse diretamente o site do Git: <https://git-scm.com/>

1.1 O que é o GIT?

GIT é uma ferramenta criada para o versionamento de códigos, isto é, uma ferramenta onde você pode fazer uma alteração no seu projeto atual e, caso necessário, retorná-lo a versão que você tinha anteriormente.

Exemplo:

Criamos o projeto chamado “Curso java” e nele possuímos diversos arquivos para trabalharmos. Eu crio um código para automatizar a página inicial do LinkedIn e subo no repositório da equipe no GitLab. Dois dias depois, vejo que a minha modificação que fiz não será mais necessária e está causando um bug na hora da execução, porém, já não lembro mais como estava o código antes. Com o Git eu posso retornar a versão do meu código para uma anterior a versão que houve a alteração.

1.2 Qual a diferença de Git para Github / Gitlab?

Essa é uma dúvida muito comum no início, pois fica confuso qual a diferença e funcionalidade de cada um deles. Então vou tentar explicar de uma maneira simples para que você não tenha mais essa dúvida.

Git é o nome dado para a ferramenta de versionamento de código, já o Github ou o Gitlab são sites que funcionam como uma “rede social” para programadores e diversas áreas. Eles funcionam basicamente para armazenar e compartilhar o seu código. Com eles nós conseguimos compartilhar nosso código com toda a nossa equipe e trabalhar **simultaneamente** no mesmo código.

2. Como iniciar meu projeto?

Vamos lá, hora de finalmente iniciarmos o nosso projeto com o Git. Para isso, você precisará ter o Git instalado e navegar até a pasta que você quer iniciar seu projeto. Então faça o seguinte:

1. Abra o CMDER. (<https://cmder.net/>)
2. Digite no console o seguinte comando: “cd (digite o caminho até a pasta)” ou navegue uma a uma até lá.
3. Na pasta desejada digite “git init” para que o sistema identifique que aquele diretório é para ser reconhecido como um projeto git.

Agora você verá que aparecerá a informação “master” ao lado do caminho.

3. Como sincronizar meu projeto com o Github/Gitlab?

Para isso é bem simples, primeiramente vamos criar uma chave SSH em nossa máquina, pois será desta forma que o Github/Gitlab identificará nosso computador como confiável e autorizará que façamos as modificações.

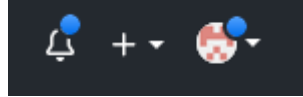
Para criar a chave, siga o seguinte passo a passo:

- Abra o cmder;
- Digite no console: `ssh -keygen -t ed25519 -C “nome_desejado”;`
- Após, digite: `cat ~/.ssh/id_25519.pub | clip`

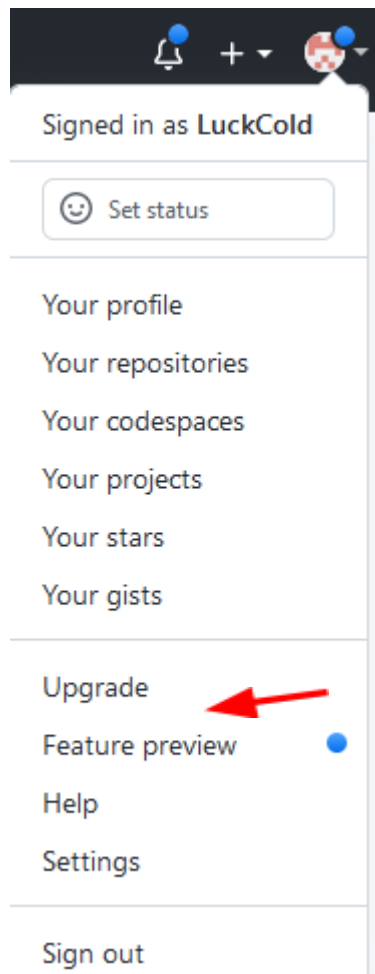
- Esse comando copiará automaticamente a sua chave SSH.

Perfeito, agora com a chave SSH em mãos, vamos acessar a página do Github/Gitlab. Para exemplo, usaremos o Github.

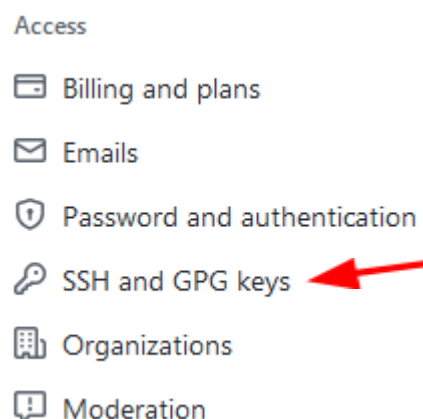
No site do [Github](https://github.com) acesse a sua conta ou crie, caso você não tenha ainda. Com a conta acessada, clique na foto do seu usuário, no canto superior direito.



E clique na opção “*settings*”:



Em *settings* clique em “SSH and GPG keys”:



Clico em “New SSH key”:

New SSH key

Em title você pode colocar o que quiser, porém é recomendado colocar a identificação de onde é aquela chave, por exemplo, “PC DE CASA”.

Em key você vai colar (CTRL+V) as informações copiadas com o comando utilizado anteriormente, lá no nosso console.

SSH keys / Add new

Title

PC de casa

Key

```
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIEfSKEhvQWpG8oMOQvj/ec6kyl3UdpRGR8S98WNYr14H
lucasmoreno389@gmail.com
```

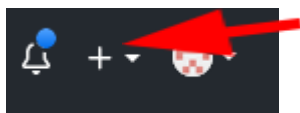
Add SSH key

Maravilha! Agora estamos conversando com o Github e será possível subir seus arquivos para ele. 😊

4. Como subo meu projeto para o Github?

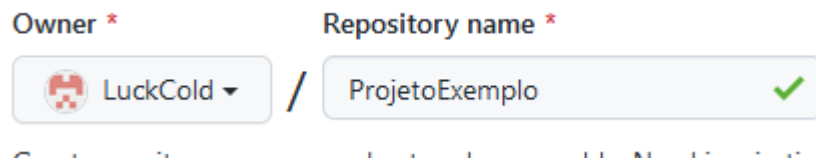
Muito bem, agora vamos colocar aquela pasta que criamos no começo no Github.

Para isso, vamos primeiro criar um novo repositório no Github. Acesse sua conta e clique no botão “+” no canto superior direito.



Após isso, clique na opção de “New repository”.

Na nova tela, você informará os dados do novo repositório. Logo na primeira opção, informe o nome dele.



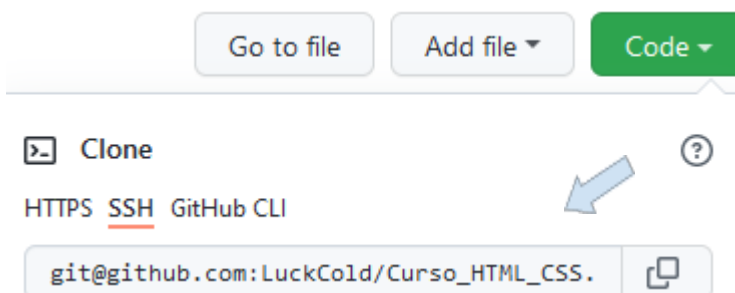
O nome do repositório pode ser qualquer um, porém lembre-se que ele será o que vai ser informado para que você possa distribuir seu projeto.

Logo em seguida, você pode colocar uma descrição do projeto (não obrigatória) e definir se ele será público ou privado.

Caso escolha **PÚBLICO** lembre-se que **QUALQUER** pessoa na internet poderá vê-lo, então nada de colocar o projeto do cliente público, beleza?

As demais opções você pode deixar da forma que está pois serão abordadas mais pra frente.

Certo, agora com o diretório criado, vamos sincronizá-lo com a pasta criada no seu computador. Para isso, no github, vamos até o repositório criado, clicamos em “Code” e, com a opção SSH selecionada, copiamos o endereço passado.



Com o console já na pasta, e o endereço copiado, vamos utilizar o seguinte comando:

```
git remote add origin "endereço SSH"
```

Pronto, com isso você vai ter sincronizado o repositório GIT com o seu computador. :D

5. Puxando um repositório do git para sua máquina

Vamos supor agora que você não tem nenhum projeto no PC, mas entrou em um time que já tem um repositório pronto no git e você só precisa trazer ele pro seu PC para começar a trabalhar, como fazemos?

Para isso é muito simples também, pelo CMDER navegue até o diretório que você queira salvar seu arquivo.

Geralmente, as equipes criam um diretório padrão para que não haja conflito nos arquivos que vão trabalhar. Por exemplo, se você está trabalhando numa automação web, precisamos configurar os drivers dentro do projeto, ele ficaria em C:/Projects/{nome_do_projeto}/drivers. Quando você puxa o projeto para a sua máquina, se você não colocá-lo na pasta C:/Projects, o código não vai conseguir localizar o driver e não iniciará.

Tá, mas agora como vou trazer o projeto pra minha máquina?

Vamos lá, agora que você está na pasta que você quer, pegue o endereço SSH do projeto, conforme explicado anteriormente e execute o seguinte comando:

```
git clone <endereço SSH>
```

Ele pedirá a senha configurada anteriormente e após isso fará o download dos arquivos do projeto.

6. Fiz meu código, como compartilho com minha equipe?

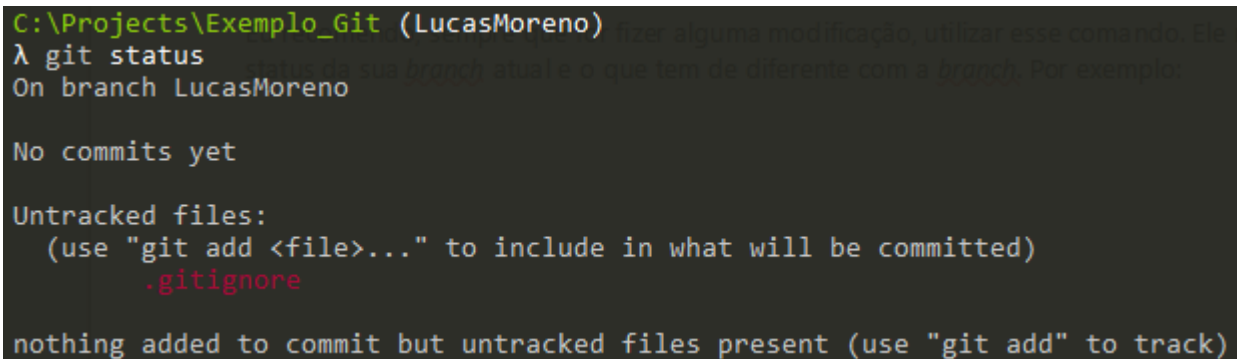
Você baixou o código no seu PC para trabalhar, fez lá um script muito fod@ e está perfeito.... Mas como você compartilha com o resto da sua equipe?

Para fazer isso, vamos abrir novamente o famoso CMDER. Já com ele na pasta do projeto e tudo arrumado, você vai precisar usar alguns comandos para verificar se tudo está certo.

Primeiro, você vai digitar:

```
git status
```

Eu recomendo, sempre que for fazer alguma modificação, utilizar esse comando. Ele mostrará como está o status da sua *branch* atual e o que tem de diferente com a *branch* que está no Github. Por exemplo:



```
C:\Projects\Exemplo_Git (LucasMoreno)
λ git status
On branch LucasMoreno

No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)
```

Você pode ver que nesse exemplo nós adicionamos o arquivo ".gitignore". Mas por que ele está em vermelho? Isso ocorre pois ainda não informamos que queremos adicionar aquele arquivo para ser "observado" pelo git. Para que façamos isso, precisamos usar o seguinte comando:

```
git add . (sim, apenas um ponto)
```

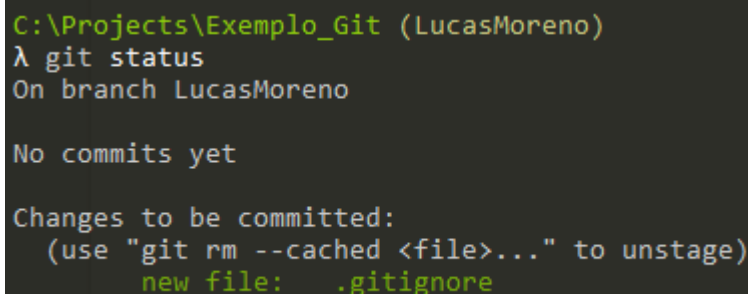
Com esse comando, você vai falar pro git: "Ou, todos os arquivos que você está falando que não estão sendo observados, pode observar, beleza?"

Se você quer adicionar apenas 1 ou outro arquivo, você pode fazer da seguinte forma:

```
git add "nome_do_arquivo"
```

Dessa forma você vai falar pro git: "Ou, começa a observar apenas esse arquivo aqui, beleza?"

Depois que você fez um dos comandos acima, o console vai ficar dessa forma:



```
C:\Projects\Exemplo_Git (LucasMoreno)
λ git status
On branch LucasMoreno

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   .gitignore
```

Olha, ele mudou de cor! Mas esse meu arquivo ainda não está no projeto, o que houve? :(

Esse comando que utilizamos, ainda não sobe o arquivo para o nosso repositório online, para isso precisamos utilizar mais alguns comandos. Rs

Primeiro, vamos preparar nosso arquivo pra subir pro repositório:

```
git commit -m "descrição RESUMIDA das mudanças"
```

Agora, use novamente o git status:

```
C:\Projects\Exemplo_Git (LucasMoreno)
λ git status
On branch LucasMoreno
nothing to commit, working tree clean
```

Wow! Agora não está mais aparecendo nada, então é apenas isso? Não :D

Agora, com todos os comandos, nós falamos pro git “Olha, observa esses arquivos aqui e prepara pra subir no repositório, beleza?”

Para enviarmos pro nosso repositório, precisamos **empurrar** os arquivos para ele. Use o seguinte comando:


```
git push origin {nome_da_sua_branch}
```

Digite a senha da sua chave SSH.

E pronto, agora você subiu seus arquivos para o seu repositório. Mas para adicionar as mudanças nos arquivos do projeto, você precisa solicitar um *merge*.

Para isso, acesse a página do projeto e vá em “Pull requests”

Você verá o seguinte aviso:

 **main** had recent pushes less than a minute ago

Compare & pull request

Clique em “Compare & pull request”


Você será direcionado para a página de comparação:

 base: LucasMoreno ← compare: main

✓ **Able to merge.** These branches can be automatically merged.

Onde está escrito “Able to merge” significa que não existe nenhum conflito entre o seu arquivo e o arquivo que está no repositório, ou seja, não houve nenhuma mudança que gere um conflito dentro daquele arquivo e você pode unir ele no repositório.

Clique em “Create pull request”:

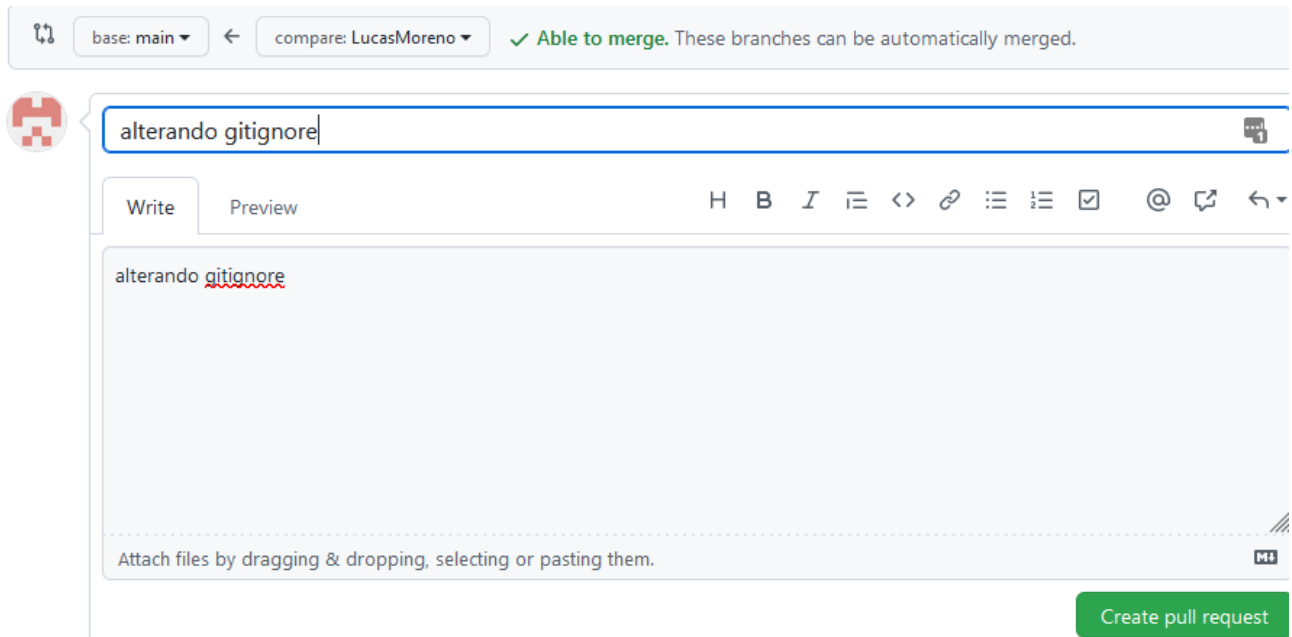
 base: main ← compare: LucasMoreno

✓ **Able to merge.** These branches can be automatically merged.

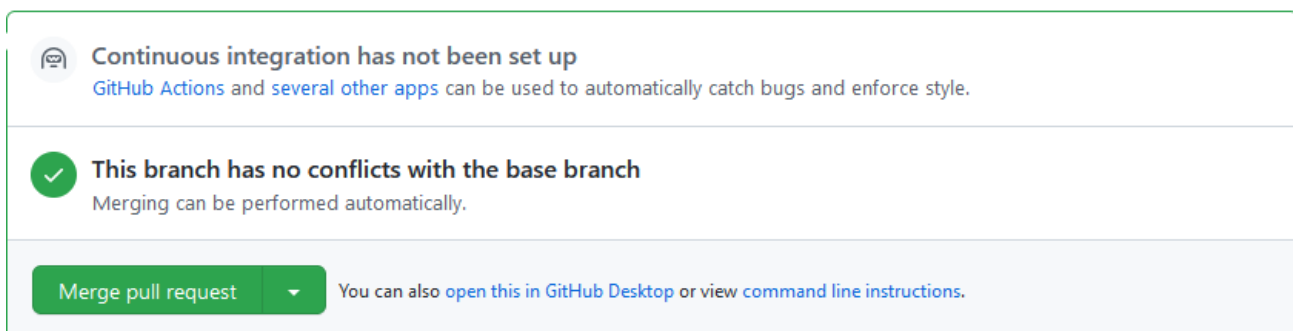
Discuss and review the changes in this comparison with others. [Learn about pull requests](#)

Create pull request

Adicione um título e um resumo da alteração e clique novamente em “Create Pull request”::



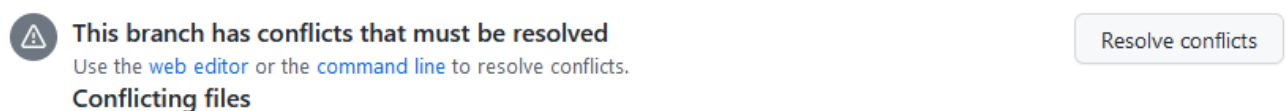
Depois de feito isso, ele verificará os arquivos e, se tudo estiver correto e não precisar de alteração manualmente, você vai poder clicar no botão “Merge Pull Request”¹:



¹Muitos projetos deixam essa tarefa de aprovar o “merge” para uma pessoa específica, pois com isso fica controlado e não vira uma bagunça onde cada um mistura com o código do outro sem se importar com o resto.

Agora, vamos para um outro cenário... E se não estivesse tudo certo e eu precisar corrigir algum problema manualmente? Vamos usar como exemplo uma automação. Em nosso projeto, vamos automatizar a home de um site, então estamos mexendo no arquivo “homePage”.

Mapeamos diversos elementos lá da página e tudo perfeito até então, mas, um colega nosso, precisou usar um elemento lá e acabou modificando uma linha do nosso código e subiu no git. Então, quando você subir seu arquivo você vai dar de cara com a seguinte mensagem de erro:



Oh my god, e agora o que faço? Calma... Esse erro é apresentado quando o seu arquivo teve um conteúdo adicionado junto com o conteúdo de um coleguinha... Então você precisa resolver esses conflitos manualmente para que consiga subir no Git.

Como faço isso? :(

7. Resolvendo conflitos no código

Você foi subir seu código e ele está bonitinho e rodando perfeitamente, mas o Git não deixou eu subir ele e agora?

Primeiramente, você precisa resolver os conflitos apresentados pelo GIT.

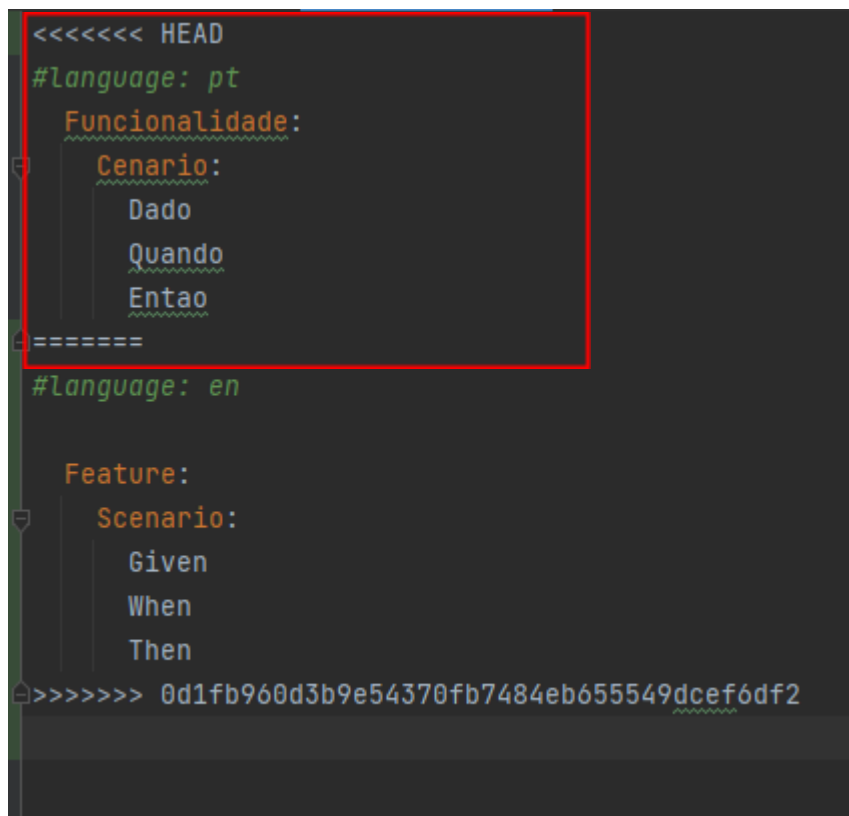
Então, você tem que atualizar seus arquivos com a branch principal. Para isso, é muito simples, só utilizar o comando:

```
git pull origin {nome_da_branch_principal}
```

Esse comando irá pegar os arquivos da branch principal e baixá-los na sua branch.

No nosso exemplo, eu criei um arquivo chamado “*github.feature*” e subi ele pra minha branch principal, porém, um colega meu também criou um arquivo com o mesmo nome e já havia subido. Então o meu apresentou conflito.

Ao atualizar com a branch principal, o arquivo que houve conflito apresentou as seguintes informações:



```
<<<<<< HEAD
#language: pt
Funcionalidade:
  Cenario:
    Dado
    Quando
    Entao
=====
#language: en
Feature:
  Scenario:
    Given
    When
    Then
>>>>>> 0d1fb960d3b9e54370fb7484eb655549dcef6ddf2
```

A parte em destaque é o seu código e o restante é o código que já estava no diretório.

Agora, você precisa ir até o arquivo que está com os conflitos e, juntamente com sua equipe, corrigir as mudanças.

Depois de conversar com seu time e arrumar os conflitos, basta realizar o procedimento anterior e solicitar o “merge” normalmente.