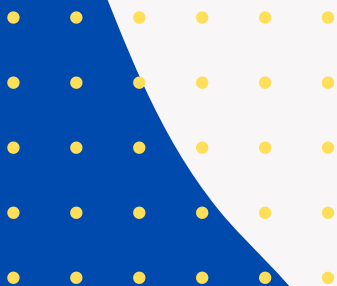


GIT: DE PADAWAN A JEDI



Micaely Gusmão

SUMÁRIO

1. Como utilizar esse material
2. O que é git?
3. Configuração
4. GitHub
5. Entendendo como funciona a força
 - a. O que são repositórios?
 - b. O que são commits?
 - c. O que são branches?
6. Na prática: comandos mais utilizados
7. Resolvendo um conflito
8. Lidando com branches
9. Pull request, o famoso PR
10. Finalização

COMO UTILIZAR ESSE MATERIAL

Olá!

Esse material foi feito com a intenção de ajudar a quebrar a abstração do trabalho no dia-a-dia em relação a utilização do Git no controle de versão do código-fonte dos projetos que atuamos.

No início terá algumas explicações teóricas sobre o que é o git e a explicação de termos e comandos que são mais utilizados. Depois terá um passo a passo na prática para utilizar alguns comandos que serão apresentados aqui.

É de extrema importância que tanto a parte teórica quanto a parte prática seja feita para a obtenção de melhores resultados no aprendizado!

Espero que esse material te ajude. Com carinho,
Micaely Gusmão!



Email para contato: gusmaomicaely@gmail.com

O QUE É GIT?

O git é uma ferramenta utilizada para se ter um histórico de alterações em arquivos e o controle das versões do código-fonte de um software.

Imagina que três pessoas estão desenvolvendo um sistema de cadastro de clientes e resolveram que um faz uma listagem e o outro faz a tela de cadastro. Porém, alguém sobe um código errado e o sistema que antes funcionava agora não está funcionando mais. Com o git, por conta do controle de versão é possível voltar até o momento em que o código estava funcionando porque existe um histórico de alterações e controle de cada momento que um código novo foi enviado.

Com o git o trabalho em equipe se tornou extremamente mais fácil e o controle de qualidade para entrega também aumentou consideravelmente.

Logo vamos começar a por a mão na massa então já faça o download:

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

CONFIGURAÇÃO

Depois que você tiver instalado o git é necessário fazer a configuração dele no seu computador/ambiente.

Para ter certeza que a instalação ocorreu como esperado abra um terminal e digite o seguinte comando:

```
Windows PowerShell
PS C:\Users\Micaely> git --version
git version 2.39.0.windows.1
PS C:\Users\Micaely> |
```

Se essas informações aparecerem a instalação ocorreu com sucesso!

Agora é necessário configurar com os seus dados:

```
Windows PowerShell
PS C:\Users\Micaely> git config --global user.name "Seu Nome"

PS C:\Users\Micaely> git config --global user.email "seuemail@email.com"
```

Coloque cada linha separada e pressione enter, caso não apareça alguma mensagem de erro está pronto para usar!

GITHUB

O GitHub é uma plataforma que hospeda o sistema de controle de versão git. Assim como o GitHub, existem outras como: GitLab, Azure Devops, BitBucket, entre outros.

É utilizando essas plataformas que conseguimos ter de forma visual a clareza de como está a nossa versão remota do arquivo/código-fonte que está sendo versionado.

Aqui nós iremos utilizar o GitHub então é de extrema importância que você tenha uma conta criada lá para que consiga acompanhar os exemplos.

Logo vamos começar a por a mão na massa então já faça a sua conta:

<https://github.com/>

ENTENDENDO COMO FUNCIONA A FORÇA

Agora vamos começar a entender como funciona o git e a explicação de alguns termos que utilizamos no dia-a-dia que facilmente você se esbarrará neles caso busque alguma solução de algum problema no google!

Veremos então os seguintes tópicos:

- O que são repositórios?
- O que são commits?
- O que são branches?

Vamos lá!



O QUE SÃO REPOSITÓRIOS?

O repositório é o local onde fica armazenado os seus arquivos, pensa no repositório como uma pasta. Igual as pastas que temos no nosso computador!

Dessa forma, teremos o repositório local que é a pasta no seu computador que contém os seus arquivos e o repositório remoto que é a pasta que estão os arquivos com a última versão que foi disponibilizada, que outras pessoas da sua equipe ou se for público qualquer pessoa, podem acessar e fazer modificações no mesmo.

Resumindo:

Repositório local: a pasta no seu computador

Repositório remoto: a sua pasta, mas no github

Ou seja, já que a pasta no github é sua, vai ter lá aquilo que você mandar para lá..

O QUE SÃO COMMITTS?

Quando você manda as suas coisas do repositório local para o repositório remoto você precisa identificar o que você está mandando.

Exemplo:

Imagine que você e mais uma pessoa estão fazendo um trabalho da faculdade e você resolveu escrever a introdução do trabalho. Depois de finalizada a introdução, você vai ter que enviar o trabalho atualizado para o seu amigo poder continuar de onde você parou, o commit é o que você vai usar para dizer que a mudança que você está mandando se refere a adição da introdução.

Funciona como se fosse uma anotação para um histórico.

Versão 1 - Criação da estrutura inicial do trabalho feita por fulano

Versão 2 - Adição da introdução feita por ciclano

Se em algum momento der errado é fácil identificar qual versão subiu o que e voltar para a parte que tudo estava certo!

O QUE SÃO BRANCHS?

Imagina que você e seu colega da faculdade chegaram em uma versão do trabalho que vocês gostaram e o professor também. Porém, chegou uma nova parte e vocês irão ter que recomeçar a partir do que está pronto.

Para manter a primeira parte sem modificações é possível deixar um backup do trabalho até aquela parte e vocês vão reiniciar a partir desse backup, ou seja, vai ter tudo que tem na primeira parte do trabalho e vocês irão adicionar novas coisas sem alterar o backup.

Depois que o professor aprovar as coisas novas vocês vão precisar pegar a versão de vocês, que tem a segunda parte, e juntar com a versão de backup fazendo o que chamamos de **merge**.

Resumindo: o backup é uma branch principal que normalmente é chamada de main ou master e a versão sua e do seu colega é uma branch dessa versão principal. O merge é a junção da sua branch com outra, nesse caso a master.

NA PRÁTICA: COMANDOS MAIS UTILIZADOS

O git tem inúmeros comandos e possibilidades, porém tem alguns que são os principais no dia-a-dia e que é difícil encontrar alguém que trabalhe com git que não tenha utilizado eles pelo menos uma vez:

Vamos falar um pouco sobre cada um dos seguintes comandos:

- git init
- git add remote
- git remote -v
- git status
- git add .
- git commit
- git stash
- git stash pop
- git pull
- git push
- git checkout -b
- git checkout

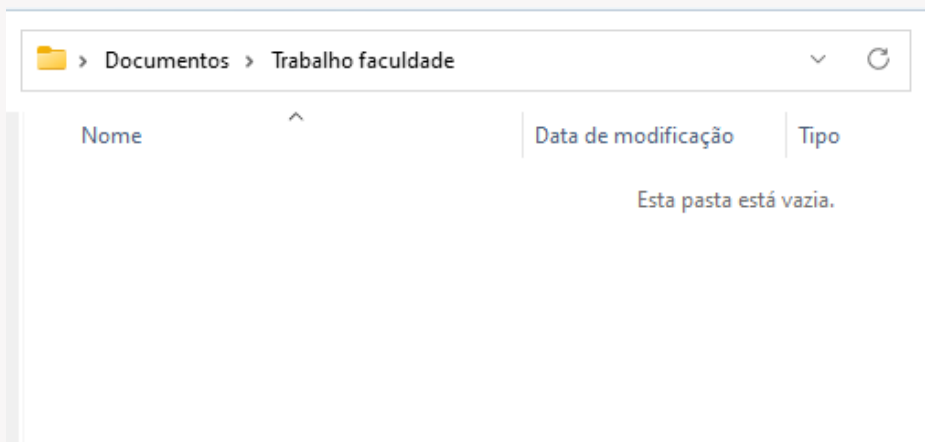
Teremos uma breve explicação de quando utilizar e como utilizar cada um!

Vamos lá!

GIT INIT

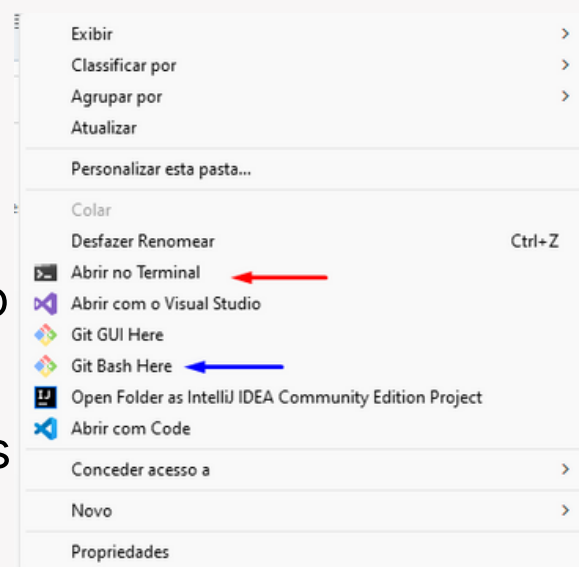
Quando você quer começar armazenar algum arquivo seu que está no seu repositório local, ou seja, no seu computador você precisa dizer para o git que você quer iniciar um repositório ali e isso você faz com o git init

Abri meus Documentos e criei um pasta Trabalho faculdade para poder criar o nosso trabalho e versionar ele com git

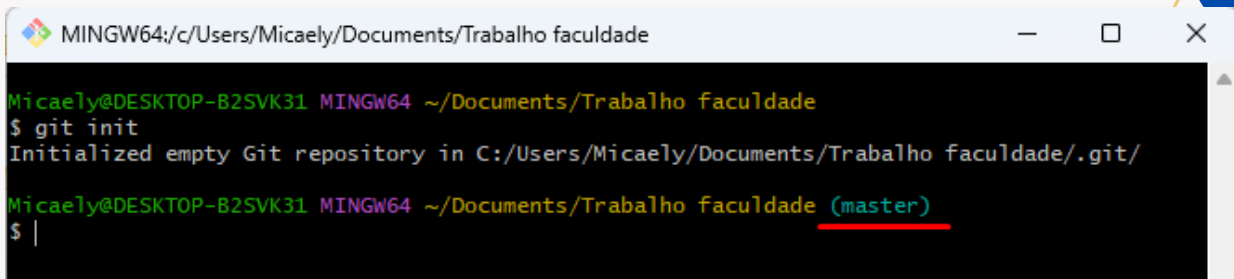


Dentro dessa pasta, aperte com o botão direito do mouse e irá aparecer duas opções:

Abrir no terminal **ou** Git Bash Here, eu vou usar o git bash porque ajuda a ver algumas informações

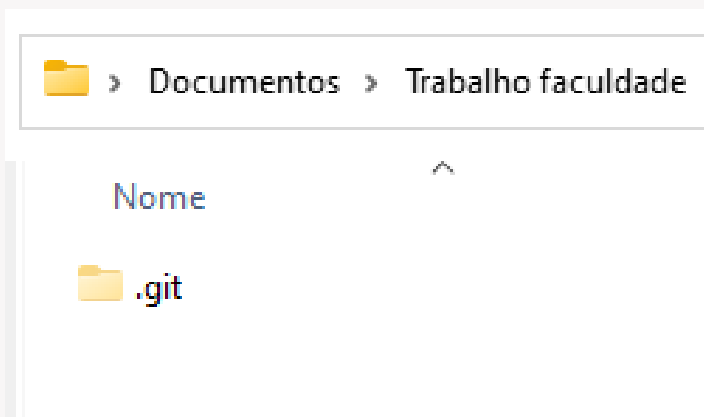


Depois de aberto o terminal vamos dizer para o git iniciar um repositório ali com o git init:

A screenshot of a Windows terminal window titled 'MINGW64; c:/Users/Micaely/Documents/Trabalho faculdade'. The prompt is 'Micaely@DESKTOP-B2SVK31 MINGW64 ~/Documents/Trabalho faculdade'. The command '\$ git init' has been entered, and the output is 'Initialized empty Git repository in C:/Users/Micaely/Documents/Trabalho faculdade/.git/'. The prompt now shows '(master)' in red text, which is underlined in the original image.

```
MINGW64; c:/Users/Micaely/Documents/Trabalho faculdade
Micaely@DESKTOP-B2SVK31 MINGW64 ~/Documents/Trabalho faculdade
$ git init
Initialized empty Git repository in C:/Users/Micaely/Documents/Trabalho faculdade/.git/
Micaely@DESKTOP-B2SVK31 MINGW64 ~/Documents/Trabalho faculdade (master)
$ |
```

Perceba que depois que enviamos o comando git init apareceu uma mensagem avisando que um repositório vazio foi criado naquele caminho e foi adicionado uma pasta .git nele.. vamos conferir:



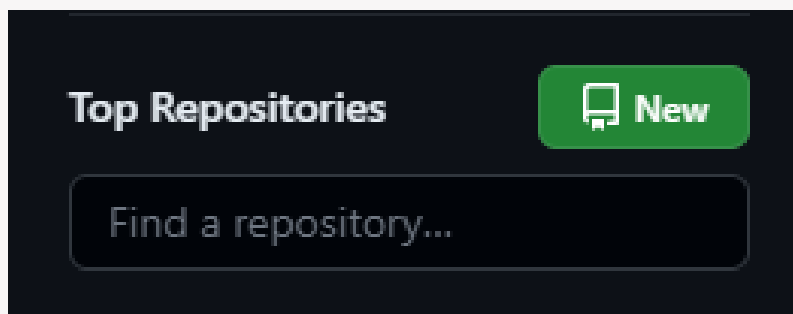
Pronto, a sua pasta começará a ter um controle de versão.

Perceba que na linha de baixo no terminal já apareceu um item a mais: **master**. Sendo assim, o conteúdo que você adicionar ali esta na branch master.

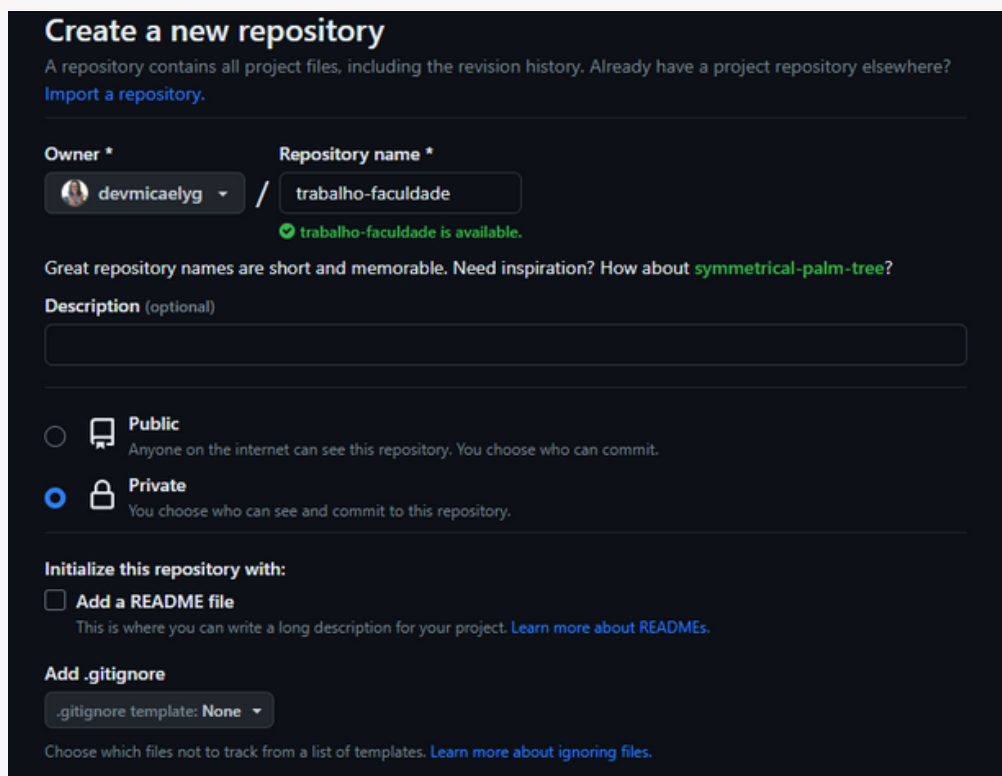
Agora vamos dizer qual será o nosso repositório remoto.

GIT ADD REMOTE

Acesse a sua conta do github e na tela inicial terá a seguinte opção

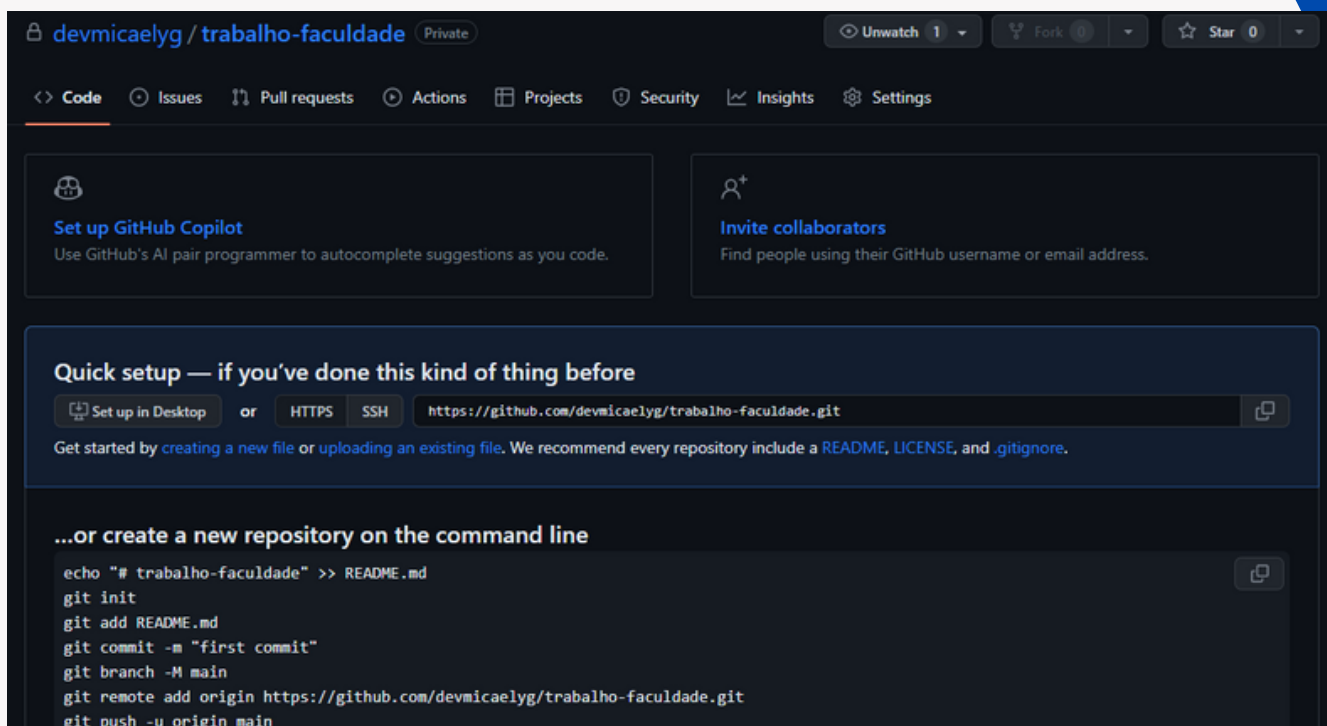


Aperte no botão verde new. Com isso, irá abrir a seguinte tela:

A screenshot of the 'Create a new repository' form on GitHub. The form has a dark background with white text. At the top, it says 'Create a new repository' in bold. Below this is a subtitle: 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.' The form has two main sections: 'Owner *' and 'Repository name *'. The 'Owner *' section has a dropdown menu with 'devmicaelyg' selected. The 'Repository name *' section has a text input field with 'trabalho-faculdade' entered. Below the repository name field, there is a green checkmark and the text 'trabalho-faculdade is available.' Below these fields is a text input field for 'Description (optional)'. Below the description field are two radio buttons: 'Public' (selected) and 'Private'. Below the radio buttons is a section titled 'Initialize this repository with:' with a checkbox for 'Add a README file' and a dropdown menu for 'Add .gitignore' with 'None' selected. At the bottom, there is a link to 'Learn more about ignoring files.'

É nessa tela que iremos colocar o nome do nosso repositório e se ele vai ser publico ou privado, coloque essas informações e depois que finalizar role até o final da página e clique no botão *create repository*

Em seguida irá abrir a seguinte tela:



Isso significa que o nosso repositório remoto foi criado, porém está vazio.

O próximo passo será relacionar o nosso repositório local com o nosso repositório remoto e é para isso que serve o **git remote add**, vamos dizer o seguinte para o git: "essa pasta minha aqui tem como repositório remoto esse aqui.."

Vamos lá configurar isso então. Temos que pegar a url do nosso repositório e podemos fazer isso na mesma página do print acima:



Depois de pegar a url voltaremos para o terminal que foi aberto anteriormente e vamos fazer o seguinte comando:

```
MINGW64:/c/Users/Micaely/Documents/Trabalho faculdade  
Micaely@DESKTOP-B2SVK31 MINGW64 ~/Documents/Trabalho faculdade (master)  
$ git remote -v  
  
Micaely@DESKTOP-B2SVK31 MINGW64 ~/Documents/Trabalho faculdade (master)  
$ |
```

O git remote -v vai pedir para o git listar quais são os repositórios remotos do nosso repositório local criado dentro da nossa pasta Trabalho faculdade. Como ainda não adicionamos o remoto o comando retorna nada.

Então vamos adicionar com o seguinte comando:

git remote add origin <a url do seu repositório>

```
MINGW64:/c/Users/Micaely/Documents/Trabalho faculdade  
Micaely@DESKTOP-B2SVK31 MINGW64 ~/Documents/Trabalho faculdade (master)  
$ git remote add origin https://github.com/devmicaelyg/trabalho-faculdade.git
```

Agora vamos repetir o git remote -v para ter certeza que o repositório remoto foi adicionado:

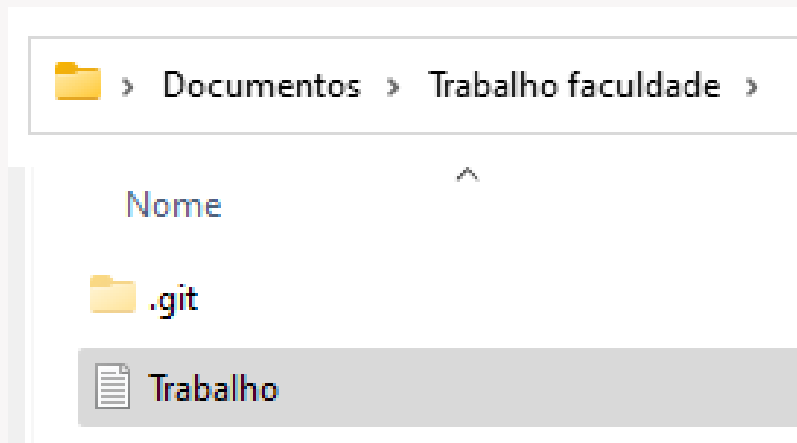
```
MINGW64:/c/Users/Micaely/Documents/Trabalho faculdade  
Micaely@DESKTOP-B2SVK31 MINGW64 ~/Documents/Trabalho faculdade (master)  
$ git remote -v  
origin https://github.com/devmicaelyg/trabalho-faculdade.git (fetch)  
origin https://github.com/devmicaelyg/trabalho-faculdade.git (push)
```

Pronto, agora nosso repositório local tem um repositório remoto que foi dado o nome de **origin**.

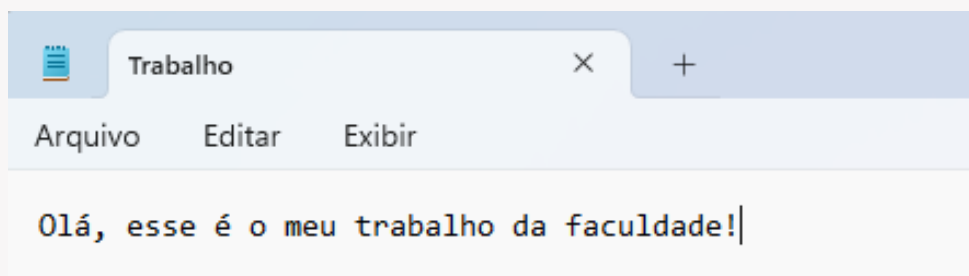
Qualquer alteração que iremos mandar para o remoto iremos utilizar o nome origin para dizer que é para lá que está sendo enviado.

GIT STATUS

Agora vamos adicionar um arquivo ao nosso repositório local porque ele ainda está vazio



Adicionei um arquivo txt somente com uma frase dentro dele



Agora, fizemos uma modificação no nosso repositório local e para verificar quais mudanças eu tenho nele podemos ir até o terminal e pedir para o git dizer qual é o status do nosso repositório local

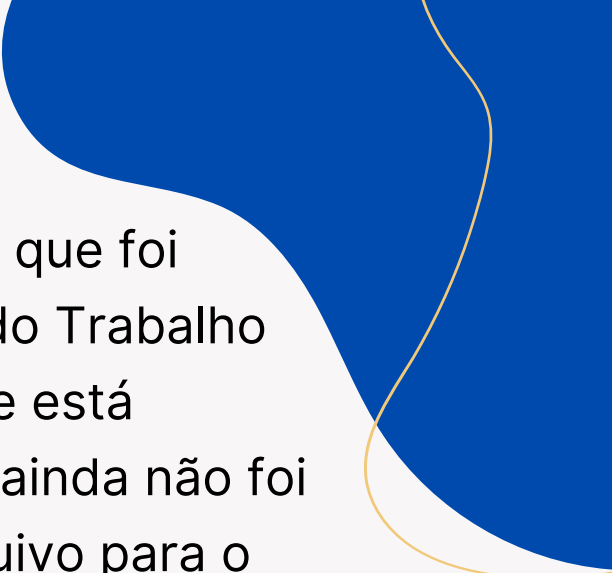
```
MINGW64:/c:/Users/Micaely/Documents/Trabalho faculdade

Micaely@DESKTOP-B2SVK31 MINGW64 ~/Documents/Trabalho faculdade (master)
$ git status
On branch master

No commits yet

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

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




O que o terminal está dizendo ai é que foi adicionado um arquivo txt chamado Trabalho nesse repositório remoto e que ele está untracked, isso significa que o git ainda não foi avisado se é para enviar esse arquivo para o repositório remoto.

Para enviar é necessário seguir o seguinte procedimento:

1. Dizer para o git adicionar na lista dele de arquivos que serão enviados que o seu arquivo deverá ser enviado
2. Dizer para ele o que você está enviando, dar um nome a esse envio para ficar em um histórico de mudanças
3. Enviar dizendo para onde você está enviando

GIT ADD

Para o primeiro passo iremos conhecer o comando git add . que irá dizer para o git: "pegue todos os arquivos modificados e adicione na sua lista de arquivos que serão enviados". Vamos fazer isso no nosso e depois pedir para ver o status



```
MINGW64:/c/Users/Micaely/Documents/Trabalho faculdade

Micaely@DESKTOP-B2SVK31 MINGW64 ~/Documents/Trabalho faculdade (master)
$ git add .

Micaely@DESKTOP-B2SVK31 MINGW64 ~/Documents/Trabalho faculdade (master)
$ git status
On branch master

No commits yet

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

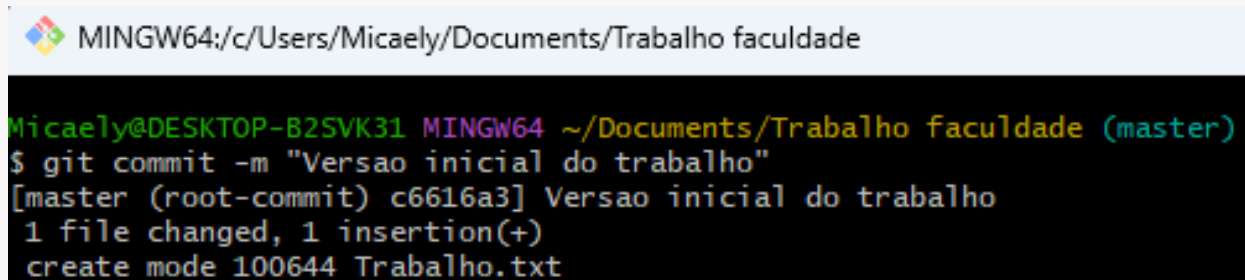
Perceba que depois do **git add .** o **git status** mudou de resultado.

Agora ele sabe que o seu arquivo deve ir para o repositório remoto e ele está dizendo: "No commits yet" que quer dizer que nenhum commit foi feito. Lembra do commit que foi explicado anteriormente? Chegou a hora dele!

GIT COMMIT

Como disse anteriormente, o git commit é usado para que possamos identificar as alterações que estamos enviando para o nosso repositório remoto. Dessa forma, qualquer problema fica mais fácil achar em qual envio aconteceu

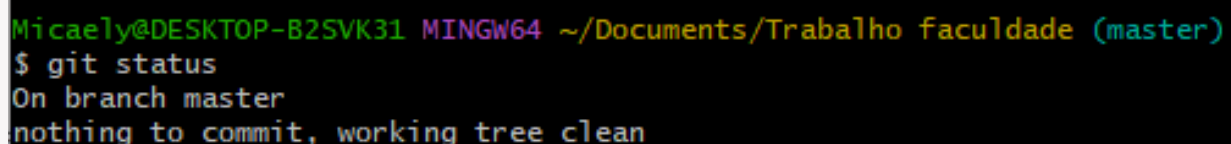
Vamos fazer o commit com o seguinte comando:
git commit -m "mensagem que vai identificar o commit"



```
MINGW64:/c/Users/Micaely/Documents/Trabalho faculdade  
Micaely@DESKTOP-B2SVK31 MINGW64 ~/Documents/Trabalho faculdade (master)  
$ git commit -m "Versao inicial do trabalho"  
[master (root-commit) c6616a3] Versao inicial do trabalho  
1 file changed, 1 insertion(+)  
create mode 100644 Trabalho.txt
```

Perceba que depois de enviar o comando aparecem algumas informações que é um log dizendo que 1 arquivo foi modificado e é uma inserção.

Vamos repetir o git status para podermos acompanhar a mudança do status do repositório local



```
Micaely@DESKTOP-B2SVK31 MINGW64 ~/Documents/Trabalho faculdade (master)  
$ git status  
On branch master  
nothing to commit, working tree clean
```

Agora o status nos diz que não tem mudanças mais na nossa branch master que precisam ser adicionadas ou commitadas.

Bom, agora falta enviar para o repositório remoto.
Vamos lá!

GIT PUSH

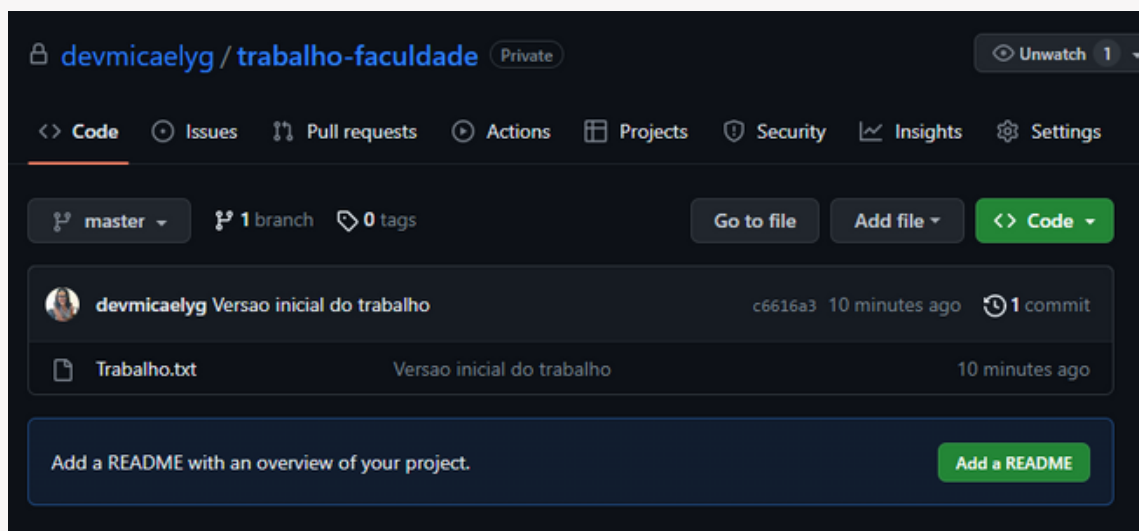
Já adicionamos na lista de envios do git e já fizemos o commit para identificar a mudança. Agora para fazer o envio é bem simples, o comando é:

git push -u origin <branch>

Estamos falando: "git envie as modificações para o meu repositório remoto, que foi denominado origin quando adicionamos ele, para a branch que se chama X"

```
Micaely@DESKTOP-B2SVK31 MINGW64 ~/Documents/Trabalho faculdade (master)
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 277 bytes | 138.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/devmicaelyg/trabalho-faculdade.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

Agora nós podemos ir até o github e recarregar a tela que abrimos depois de criar o repositório e ele estará assim:



GIT PULL

Quando estamos trabalhando em equipe é extremamente comum que a outra pessoa suba alterações no repositório remoto fazendo com que a sua versão no seu repositório local fique desatualizada.

O problema disso é que se você continuar mexendo no seu arquivo com ele estando em uma versão desatualizada depois na hora de enviar para o repositório remoto vai ter um problema conhecido como **conflito**. O conflito é quando você tenta subir uma modificação em cima de uma modificação que foi feita antes da sua.

Para isso é sempre bom ter o costume de atualizar a sua versão local com a ultima versão remota e fazemos isso com o git pull

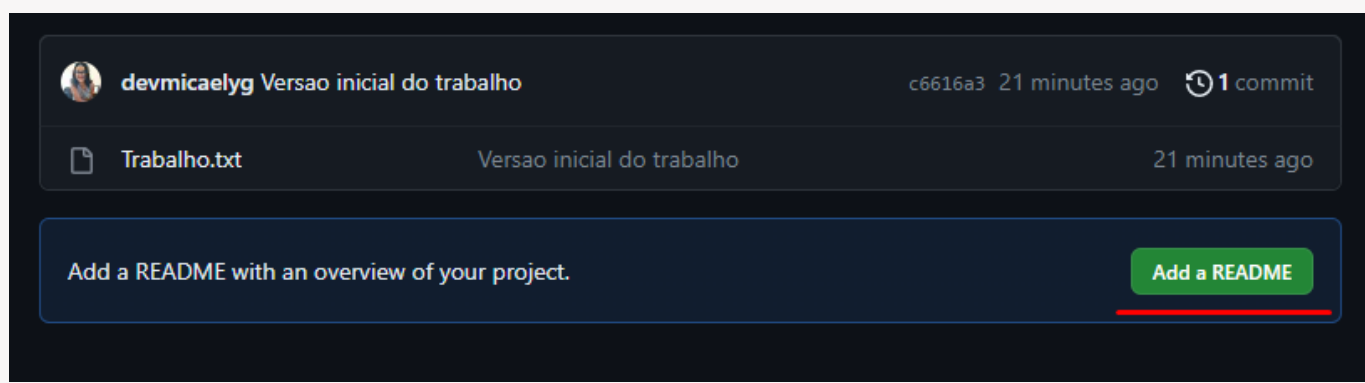
git pull origin <branch>

Estamos dizendo para o git o seguinte: "git pegue todas as mudanças que tem no meu repositório local chamado de origin que estão na branch X"

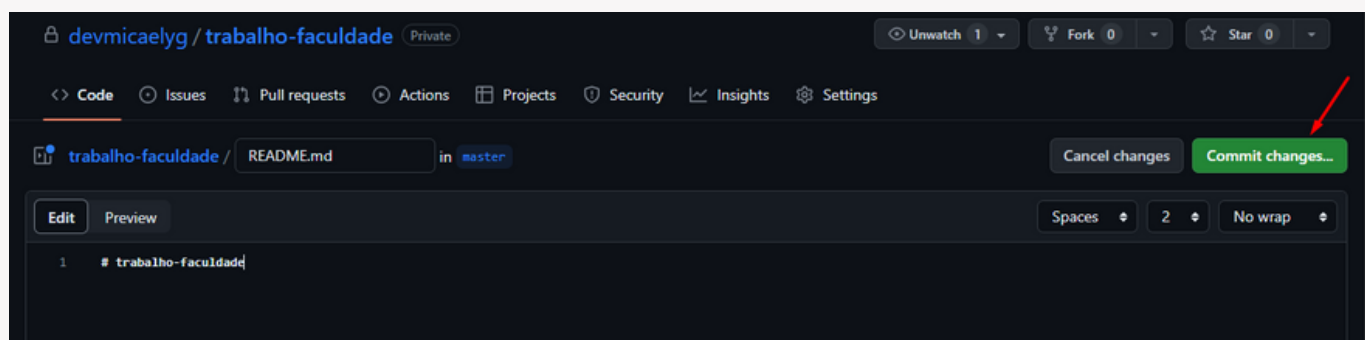
```
Micaely@DESKTOP-B2SVK31 MINGW64 ~/Documents/Trabalho faculdade (master)
$ git pull origin master
From https://github.com/devmicaelyg/trabalho-faculdade
 * branch          master       -> FETCH_HEAD
Already up to date.
```

Nesse caso o nosso repositório remoto não tem nenhuma modificação então a resposta do comando é: "A sua versão local está atualizada com o seu repositório remoto"

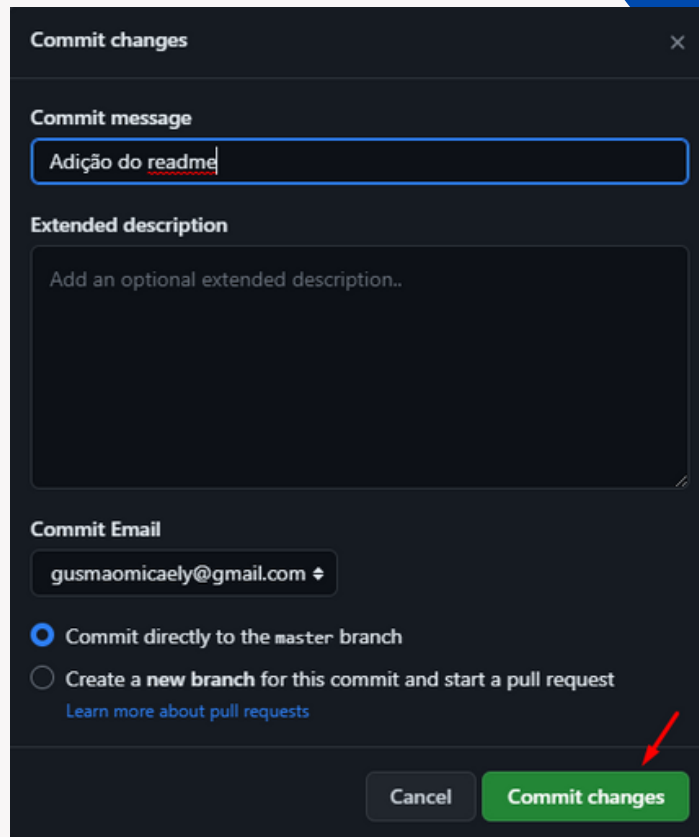
Vamos adicionar uma modificação para podermos ver o git pull em ação.



Volte na página do seu repositório no github e clique em add a README



Como o readme abre com esse texto padrão # trabalho-faculdade clique direto em Commit changes



A screenshot of the 'Commit changes' dialog box in a code editor. The dialog has a dark theme. At the top, it says 'Commit changes' with a close button. Below is a 'Commit message' field containing the text 'Adição do readme'. Underneath is an 'Extended description' field with a placeholder text 'Add an optional extended description..'. Below that is a 'Commit Email' field containing 'gusmaomicaely@gmail.com'. There are two radio buttons: the first is selected and labeled 'Commit directly to the master branch', and the second is labeled 'Create a new branch for this commit and start a pull request' with a link 'Learn more about pull requests'. At the bottom right, there is a green 'Commit changes' button, which is pointed to by a red arrow, and a grey 'Cancel' button.

Commit changes

Commit message

Adição do readme

Extended description

Add an optional extended description..

Commit Email

gusmaomicaely@gmail.com

☒ Commit directly to the master branch

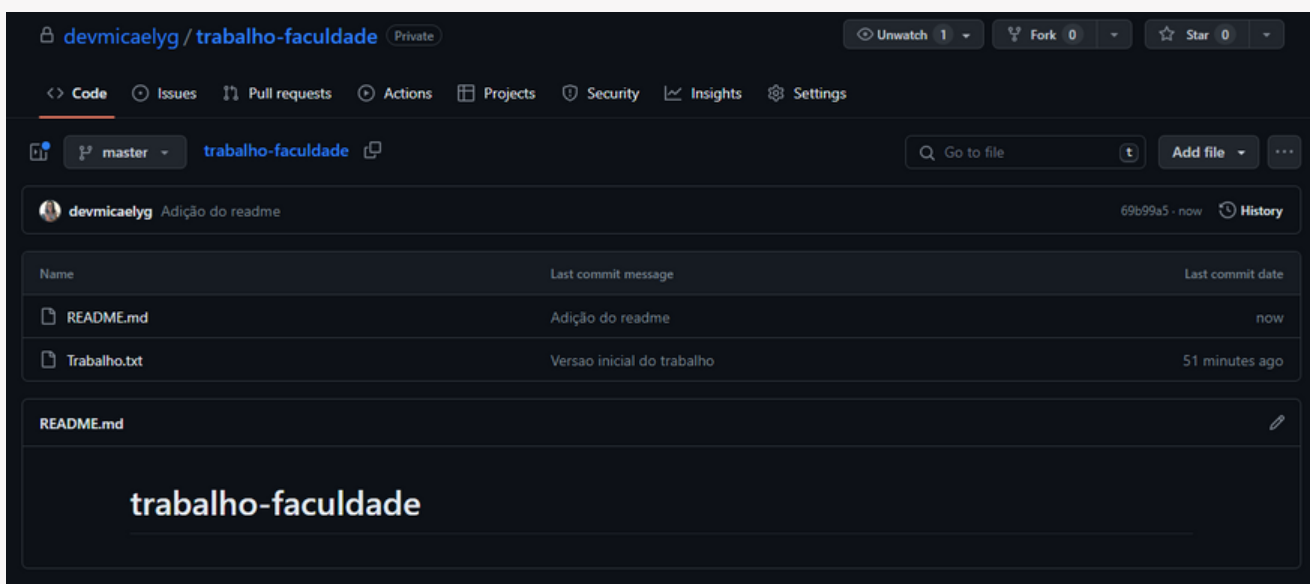
☐ Create a new branch for this commit and start a pull request

[Learn more about pull requests](#)

Cancel Commit changes

Adicione a mensagem do commit e aperte em Commit changes.

Ele vai recarregar e voltar para a página do seu repositório com o Readme lá



A screenshot of a GitHub repository page for 'devmicaelyg / trabalho-faculdade'. The page shows the commit history for the 'master' branch. The most recent commit is by 'devmicaelyg' with the message 'Adição do readme' and commit hash '69b99a5', dated 'now'. Below the commit history is a table with columns 'Name', 'Last commit message', and 'Last commit date'. The table lists two files: 'README.md' with message 'Adição do readme' and 'Trabalho.txt' with message 'Versao inicial do trabalho'. Below the table is a preview of the 'README.md' file, showing the text 'trabalho-faculdade'.

devmicaelyg / trabalho-faculdade Private

Unwatch 1 Fork 0 Star 0

Code Issues Pull requests Actions Projects Security Insights Settings

master trabalho-faculdade

devmicaelyg Adição do readme 69b99a5 · now History

Name	Last commit message	Last commit date
README.md	Adição do readme	now
Trabalho.txt	Versao inicial do trabalho	51 minutes ago

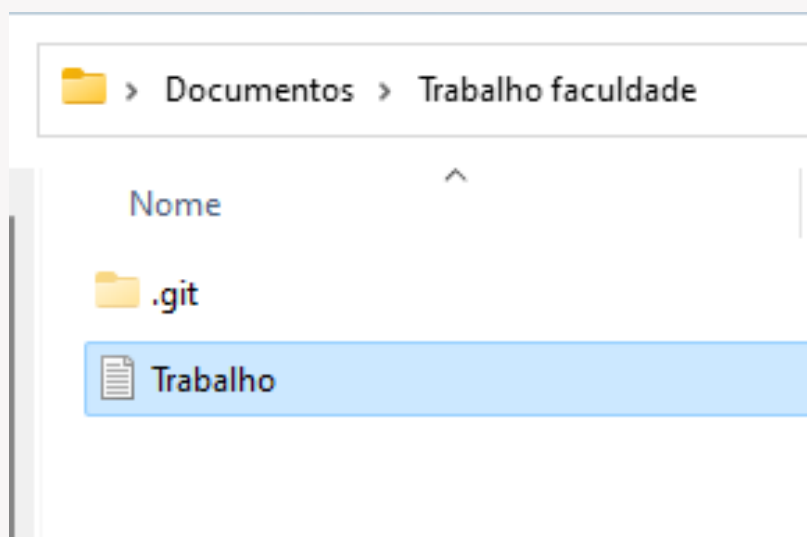
README.md

trabalho-faculdade

O README é um arquivo markdown (.md) que fica na página inicial dos repositórios com uma introdução/descrição do que se refere aquele repositório. Exemplo: quais são as tecnologias utilizadas, como preparar o ambiente de desenvolvimento ou como utilizar o item que foi versionado.

No nosso caso adicionamos ele com um texto padrão somente para modificar a nossa versão remota.

Agora vamos atualizar a nossa versão local com as mudanças que fizemos na remota. Só para lembrar a nossa pasta está assim:



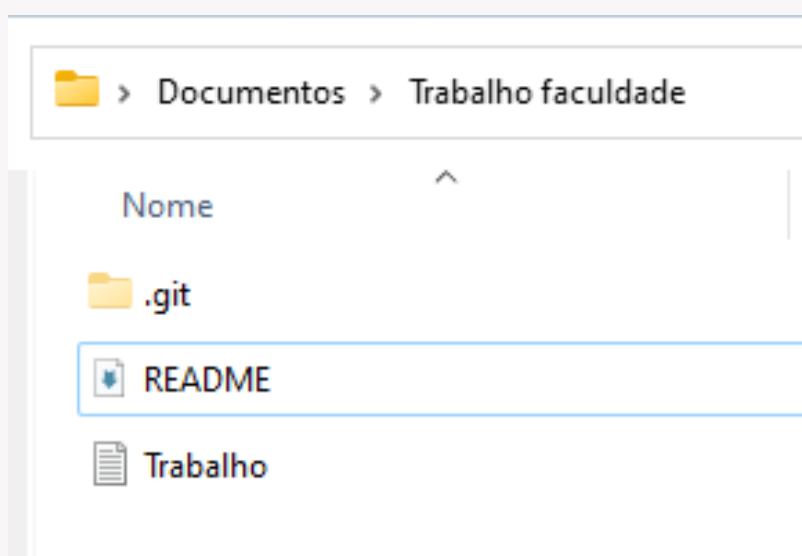
Lembra do git pull? Vamos fazer ele agora

```
Micaely@DESKTOP-B2SVK31 MINGW64 ~/Documents/Trabalho faculdade (master)
$ git pull origin master
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 686 bytes | 40.00 KiB/s, done.
From https://github.com/devmicaelyg/trabalho-faculdade
* branch          master      -> FETCH_HEAD
  c6616a3..69b99a5 master      -> origin/master
Updating c6616a3..69b99a5
Fast-forward
 README.md | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
```

Diferente da última vez, agora tínhamos itens a serem puxados da versão remota para a nossa local e o log do git pull está mostrando exatamente isso ali.

Se você olhar com atenção verá que o arquivo que veio do remoto foi exatamente o README.md que criamos pelo github.

Vamos olhar agora a nossa pasta como ficou



Agora o nosso repositório local na branch master está sincronizada com a do repositório remoto!





Bom, até agora aprendemos:

- Como iniciar um repositório git em uma pasta do seu computador
- Como criar um repositório remoto no github
- Como vincular esse repositório remoto com o seu repositório local
- Como enviar as suas mudanças locais
- Como atualizar a sua versão local com a versão remota do seu repositório

Esses comandos são os mais utilizados no dia-a-dia no desenvolvimento de software, porém além deles tem mais dois comandos que também serão bem úteis quando você estiver fazendo os comandos que aprendemos até agora, são eles:

- git stash
- git stash pop

Vamos começar..



GIT STASH

As vezes acontece de você precisar atualizar a sua versão local com a versão remota no meio de um trabalho.

Exemplo:

Você já está com a sua versão local do trabalho modificada e o seu colega da faculdade atualizou a versão remota e para continuar fazendo o seu você precisa dessa atualização para poder continuar. Se você tentar atualizar com o pull tendo modificações no seu local vai acontecer um problema que é o seguinte:

OBS: Para essa demonstração eu fiz uma modificação no nosso arquivo txt remoto e no nosso local antes de tentar fazer o pull seguinte.

Com um git status vocês podem ver que tem modificações no meu repositório local

```
Micaely@DESKTOP-B2SVK31 MINGW64 ~/Documents/Trabalho faculdade (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Trabalho.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

No meu repositório remoto o meu arquivo txt está desse jeito:



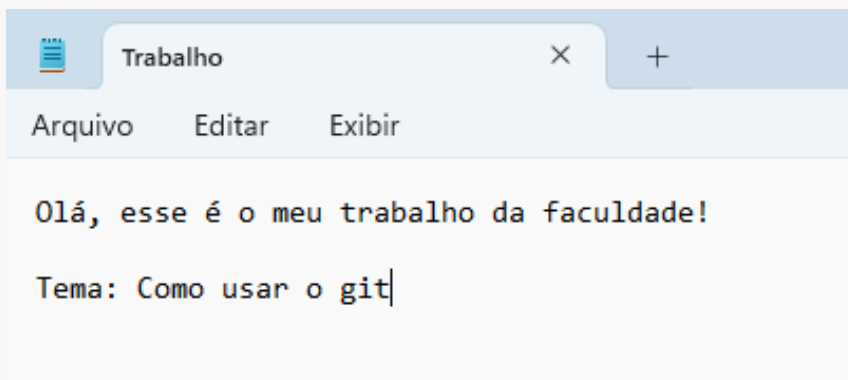
```
trabalho-faculdade / Trabalho.txt

devmicaelyg Parte 2 do trabalho

Code Blame 4 lines (3 loc) · 76 Bytes

1 Olá, esse é o meu trabalho da faculdade!
2
3 Grupo: Fulano e Ciclano
4 Tema:
```

E no repositório local desse jeito:



```
Trabalho

Arquivo Editar Exibir

Olá, esse é o meu trabalho da faculdade!

Tema: Como usar o git|
```

Está claramente diferente e depois de perceber vamos ter que sincronizar as duas versões, olha a mensagem que aparece:

```
Micaely@DESKTOP-B2SVK31 MINGW64 ~/Documents/Trabalho faculdade (master)
$ git pull origin master
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 733 bytes | 48.00 KiB/s, done.
From https://github.com/devmicaelyg/trabalho-faculdade
* branch master -> FETCH_HEAD
69b99a5..67e08fa master -> origin/master
error: Your local changes to the following files would be overwritten by merge:
Trabalho.txt
Please commit your changes or stash them before you merge.
Aborting
Updating 69b99a5..67e08fa
```

Ele começa o processo de atualização e quando percebe que tem modificações no mesmo arquivo e que isso pode acabar se misturando ele avisa: "Por favor, commita as suas mudanças OU faça um stash antes de fazer o merge"

Como o que você começou fazer no seu repositório local ainda não foi finalizado é conveniente que você faça o stash

O git stash é basicamente dizer para o git guardar as suas modificações enquanto você faz a atualização do seu repositório local com o repositório remoto, ou seja, o merge

```
Micaely@DESKTOP-B2SVK31 MINGW64 ~/Documents/Trabalho faculdade (master)
$ git stash
Saved working directory and index state WIP on master: 69b99a5 Adição do readme
```

Ele guardou as suas modificações e disse que o ultimo commit nessa versão do stash é a do hash 69b99a5 com a mensagem "Adição do readme" (por isso a importância de se colocar uma mensagem clara no commit)

Agora vamos tentar o pull novamente

```
Micaely@DESKTOP-B2SVK31 MINGW64 ~/Documents/Trabalho faculdade (master)
$ git pull origin master
From https://github.com/devmicaelyg/trabalho-faculdade
 * branch          master      -> FETCH_HEAD
Updating 69b99a5..67e08fa
Fast-forward
 Trabalho.txt | 5 ++++-
1 file changed, 4 insertions(+), 1 deletion(-)
```

Agora ele conseguiu fazer o pull e mostrou ali que um arquivo foi modificado e foi o Trabalho.txt e teve 4 inserções e uma deleção dentro dele

Bom, agora precisamos pegar a nossa versão que foi guardada para juntar com essa que veio e para isso vamos utilizar o git stash pop

GIT STASH POP

A diferença do git stash para o git stash pop é que o git stash guarda e o git stash pop pega o que foi guardado.

Observe um detalhe que aconteceu:

```
Micaely@DESKTOP-B2SVK31 MINGW64 ~/Documents/Trabalho faculdade (master)
$ git stash pop
Auto-merging Trabalho.txt
CONFLICT (content): Merge conflict in Trabalho.txt
On branch master
Your branch is up to date with 'origin/master'.

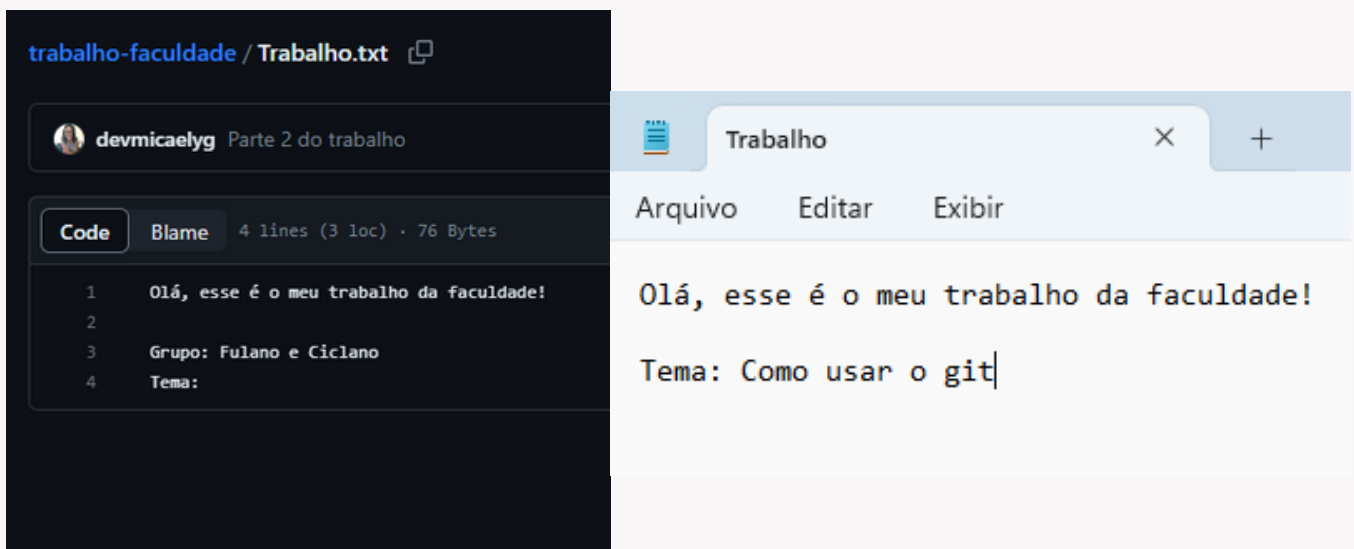
Unmerged paths:
  (use "git restore --staged <file>..." to unstage)
  (use "git add <file>..." to mark resolution)
    both modified:   Trabalho.txt

no changes added to commit (use "git add" and/or "git commit -a")
The stash entry is kept in case you need it again.
```

RESOLVENDO UM CONFLITO

Ele fez o git stash pop e no momento em que estava fazendo o auto-merging ele identificou um CONFLICT, o conflito que eu tinha comentado anteriormente.

Analizando as duas versões poderemos ver o seguinte:



The image displays two side-by-side screenshots of a code editor window titled 'Trabalho'. The left screenshot shows the remote version of the file 'Trabalho.txt' with the following content:

```
1 Olá, esse é o meu trabalho da faculdade!
2
3 Grupo: Fulano e Ciclano
4 Tema:
```

The right screenshot shows the local version of the file 'Trabalho.txt' with the following content:

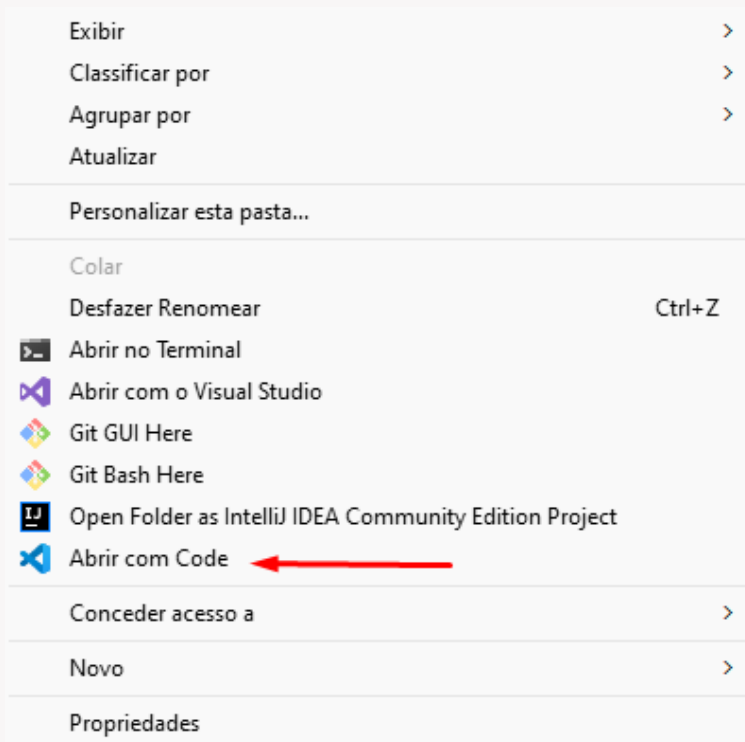
```
Olá, esse é o meu trabalho da faculdade!
Tema: Como usar o git
```

Na versão remota já tinha a linha Grupo: Fulano e Ciclano e nessa mesma linha do nosso local já tínhamos colocado o tema, quando ele tentou juntar os dois ele ficou em dúvida: qual linha eu devo colocar?

Como ele não sabe a resposta ele avisa: CONFLICT

Para resolver conflitos é mais seguro se você utilizar uma ferramenta como o visual studio code que se você não tiver no seu computador pode fazer o download através desse link
<https://code.visualstudio.com/download>

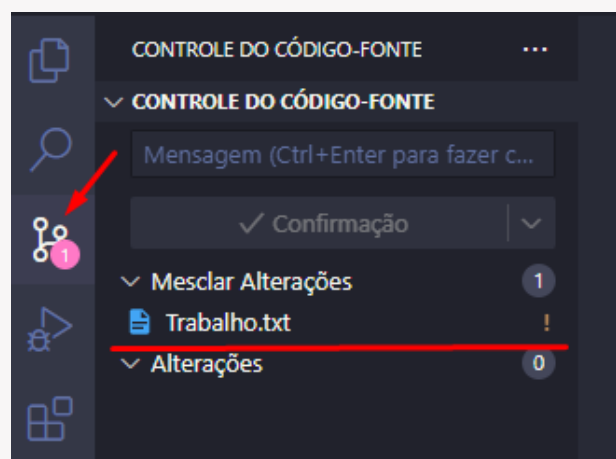
Com o botão direito na pasta que você tem o repositório git iniciado



Clique em Abrir com Code ou no terminal que está aberto digite code . que também abrirá

```
Micaely@DESKTOP-B2SVK31 MINGW64 ~/Documents/Trabalho faculdade (master)
$ code .
```

Clicando na opção indicada pela seta será possível ver a linha que está destacada



Essa exclamação no final dela está demonstrando que é necessário uma ação para mesclar as alterações, clique no arquivo e abrirá a seguinte visualização:

```
You, há 2 segundos | 1 author (You)
1 Olá, esse é o meu trabalho da faculdade! You, há 24 minutos • Parte 2 do trabalho
2
3 Aceitar Alteração Atual | Aceitar Alteração de Entrada | Aceitar Ambas as Alterações | Comparar Alterações
4 <<<<<<< Updated upstream (Alteração Atual)
5 Grupo: Fulano e Ciclano
6 Tema:
7 =====
8 Tema: Como usar o git
9 >>>>>>> Stashed changes (Alteração da Entrada)
```

Ele está te mostrando de forma colorida o conflito e pedindo que você escolha:

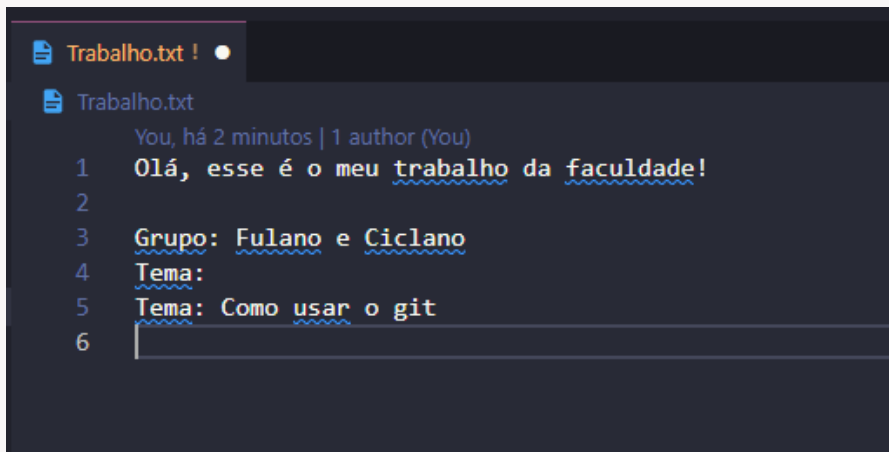
- Aceitar a sua alteração que é a roxa
- Aceitar a alteração do remoto que é a verde
- Aceitar as duas
- Comparar as duas

Decidir isso é resolver o conflito.

No nosso caso eu quero aceitar a que está vindo porque tem o grupo e também aceitar a minha porque já tem o tema escrito, logo eu vou clicar em:

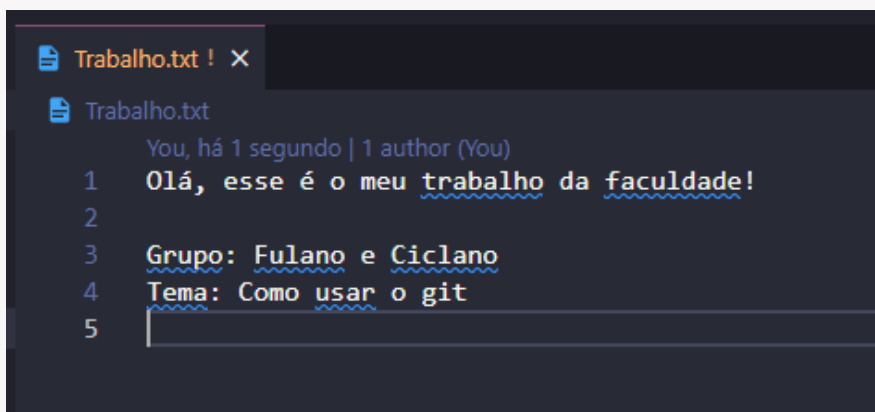
```
You, há 2 segundos | 1 author (You)
1 Olá, esse é o meu trabalho da faculdade! You, há 24 minutos • Parte 2 do trabalho
2
3 Aceitar Alteração Atual | Aceitar Alteração de Entrada | Aceitar Ambas as Alterações | Comparar Alterações
4 <<<<<<< Updated upstream (Alteração Atual)
5 Grupo: Fulano e Ciclano
6 Tema:
7 =====
8 Tema: Como usar o git
9 >>>>>>> Stashed changes (Alteração da Entrada)
```

Com isso o arquivo vai ficar assim:



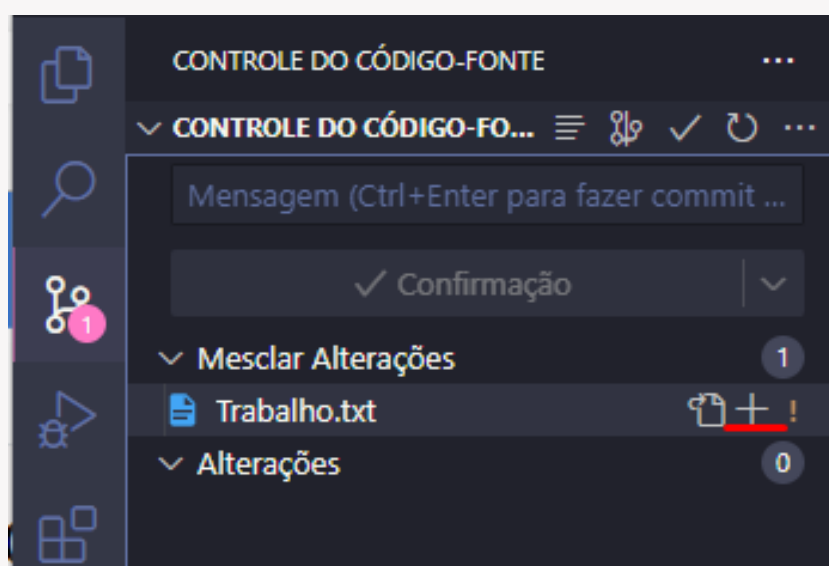
```
Trabalho.txt !
Trabalho.txt
You, há 2 minutos | 1 author (You)
1 Olá, esse é o meu trabalho da faculdade!
2
3 Grupo: Fulano e Ciclano
4 Tema:
5 Tema: Como usar o git
6
```

Vamos arrumar como queremos que fique

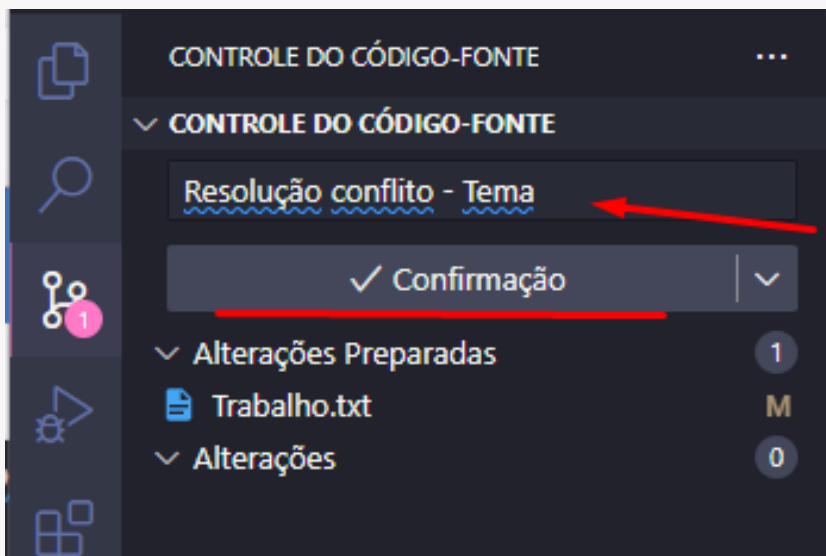


```
Trabalho.txt ! X
Trabalho.txt
You, há 1 segundo | 1 author (You)
1 Olá, esse é o meu trabalho da faculdade!
2
3 Grupo: Fulano e Ciclano
4 Tema: Como usar o git
5
```

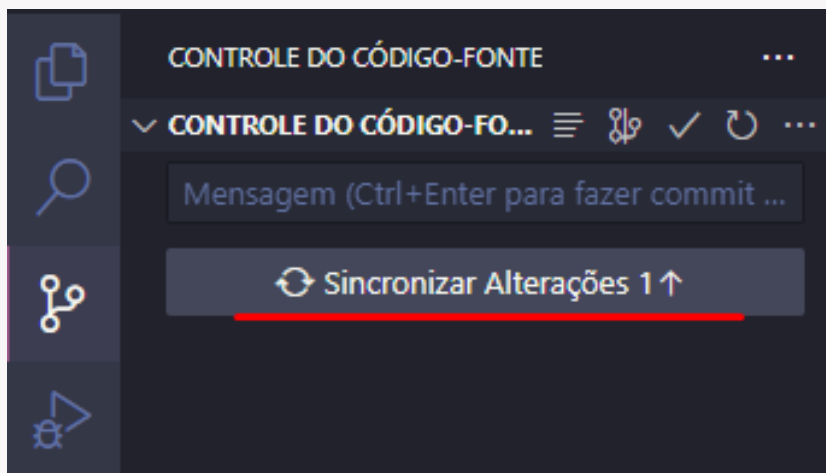
Podemos agora adicionar essa modificação apertando no mais



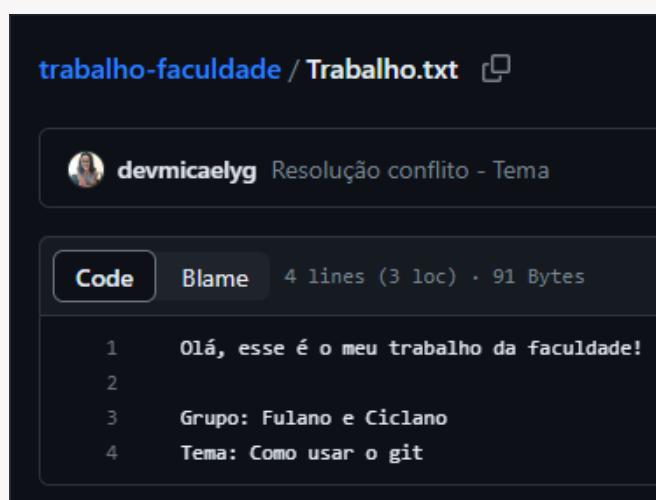
Adicionar uma mensagem e confirmar:



Agora vamos sincronizar as mudanças do local com o remoto:



Depois de carregado se você voltar no seu github e olhar o seu arquivo Trabalho.txt verá que ele está assim:



Isso significa que os últimos três passos que fizemos em uma interface gráfica do visual studio code é exatamente os comandos que tínhamos aprendido anteriormente

- `git add .`
- `git commit -m "Resolução conflito - Tema"`
- `git push -u origin master`



Fizemos isso tudo na branch master, mas e se tiver outra branch?

LIDANDO COM BRANCHS

Vamos dizer que a versão que temos agora do nosso trabalho seja a primeira que foi aprovada pelo professor

A screenshot of a GitHub commit page. The repository is 'trabalho-faculdade' and the file is 'Trabalho.txt'. The commit is by user 'devmicaelyg' with the message 'Resolução conflito - Tema'. It shows the 'Code' tab selected, displaying 4 lines of code. The code content is: 'Olá, esse é o meu trabalho da faculdade!', 'Grupo: Fulano e Ciclano', and 'Tema: Como usar o git'.

```
trabalho-faculdade / Trabalho.txt  
  
devmicaelyg Resolução conflito - Tema  
  
Code Blame 4 lines (3 loc) · 91 Bytes  
  
1 Olá, esse é o meu trabalho da faculdade!  
2  
3 Grupo: Fulano e Ciclano  
4 Tema: Como usar o git
```

Porém, agora precisamos da continuidade nesse trabalho e não queremos correr o risco de modificar o que já foi aprovado pelo professor. Para isso, nós vamos criar uma nova branch a partir da versão da master

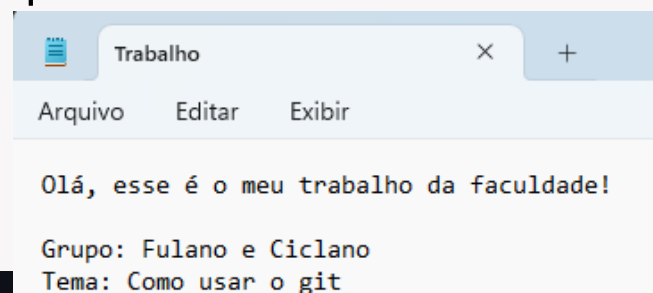
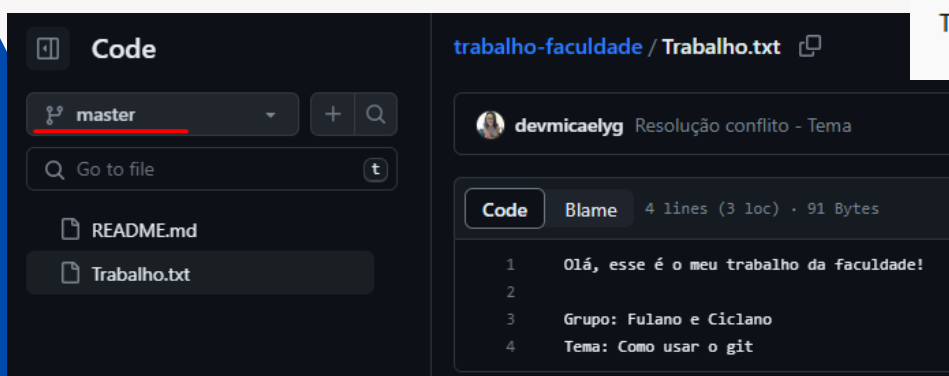
Para isso utilizaremos o comando:
`git checkout -b <nome da branch nova>`

```
Micaely@DESKTOP-B2SVK31 MINGW64 ~/Documents/Trabalho faculdade (master)
$ git checkout -b explicacao-git
Switched to a new branch 'explicacao-git'

Micaely@DESKTOP-B2SVK31 MINGW64 ~/Documents/Trabalho faculdade (explicacao-git)
$ |
```

Observe que a nova branch foi criada e agora no lugar da master aparece explicacao-git que é branch que estamos agora vendo a versão.

Se abrirmos o nosso txt você verá que o conteúdo é exatamente igual ao da master que está no repositório remoto



Agora vamos fazer a alteração no nosso arquivo local e enviar para o remoto

```
Micaely@DESKTOP-B2SVK31 MINGW64 ~/Documents/Trabalho faculdade (explicacao-git)
$ git status
On branch explicacao-git
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Trabalho.txt

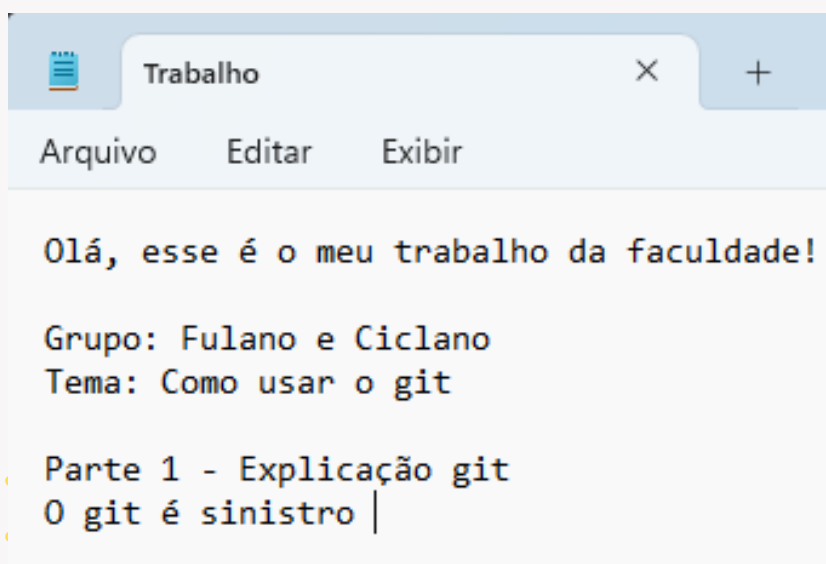
no changes added to commit (use "git add" and/or "git commit -a")

Micaely@DESKTOP-B2SVK31 MINGW64 ~/Documents/Trabalho faculdade (explicacao-git)
$ git add .

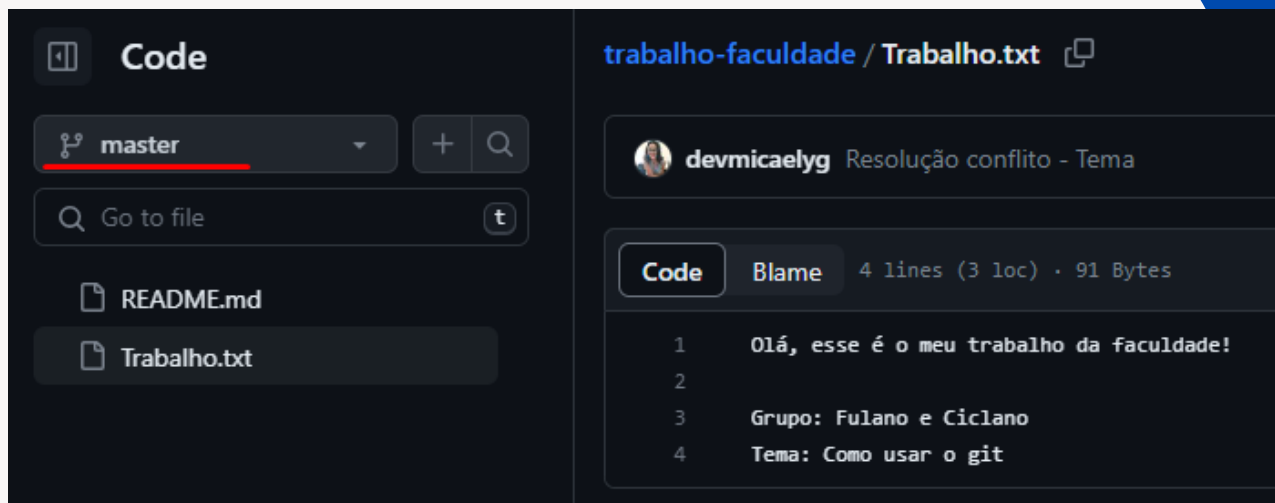
Micaely@DESKTOP-B2SVK31 MINGW64 ~/Documents/Trabalho faculdade (explicacao-git)
$ git commit -m "Adição da parte 1"
[explicacao-git 4867e24] Adição da parte 1
1 file changed, 3 insertions(+)

Micaely@DESKTOP-B2SVK31 MINGW64 ~/Documents/Trabalho faculdade (explicacao-git)
$ git push -u origin explicacao-git
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 418 bytes | 209.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'explicacao-git' on GitHub by visiting:
remote:   https://github.com/devmicaelyg/trabalho-faculdade/pull/new/explicacao-git
remote:
To https://github.com/devmicaelyg/trabalho-faculdade.git
 * [new branch]      explicacao-git -> explicacao-git
branch 'explicacao-git' set up to track 'origin/explicacao-git'.
```

O arquivo que foi enviado está assim



Vamos olhar a master remoto agora ..



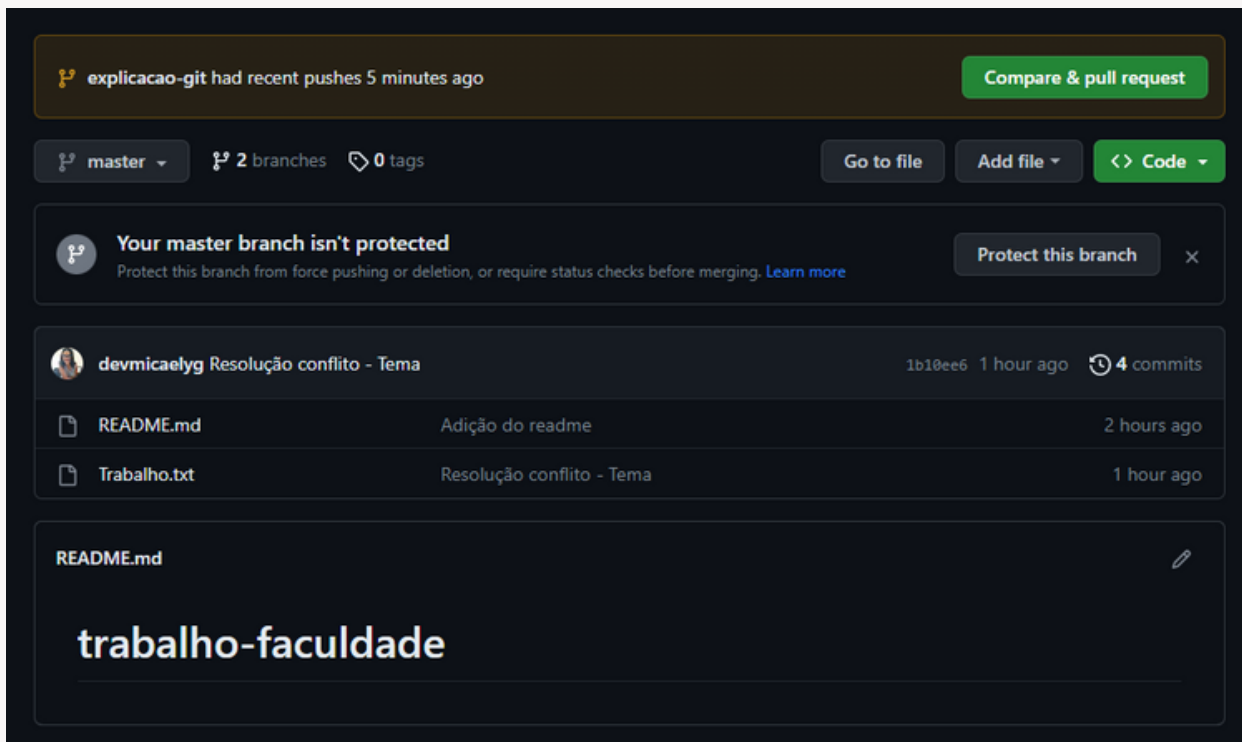
Está do mesmo jeito, porquê?

Ela está assim porque a branch que nós atualizamos foi a explicacao-git e não a master.. Nós criamos outra branch para poder preservar a versão da master e não correr o risco de estragar o que já está pronto.

Foi enviado a nova versão e agora o professor precisa dizer se está como ele queria. Assim que ele liberar é necessário fazer um merge da master com a explicacao-git para que a nova versao estavel contenha tanto o que o professor corrigiu na primeira vez quanto o que ele corriu na segunda vez

E para fazer esse merge, essa junção, utilizamos o conhecido PR ou pull request

PULL REQUEST, O FAMOSO PR

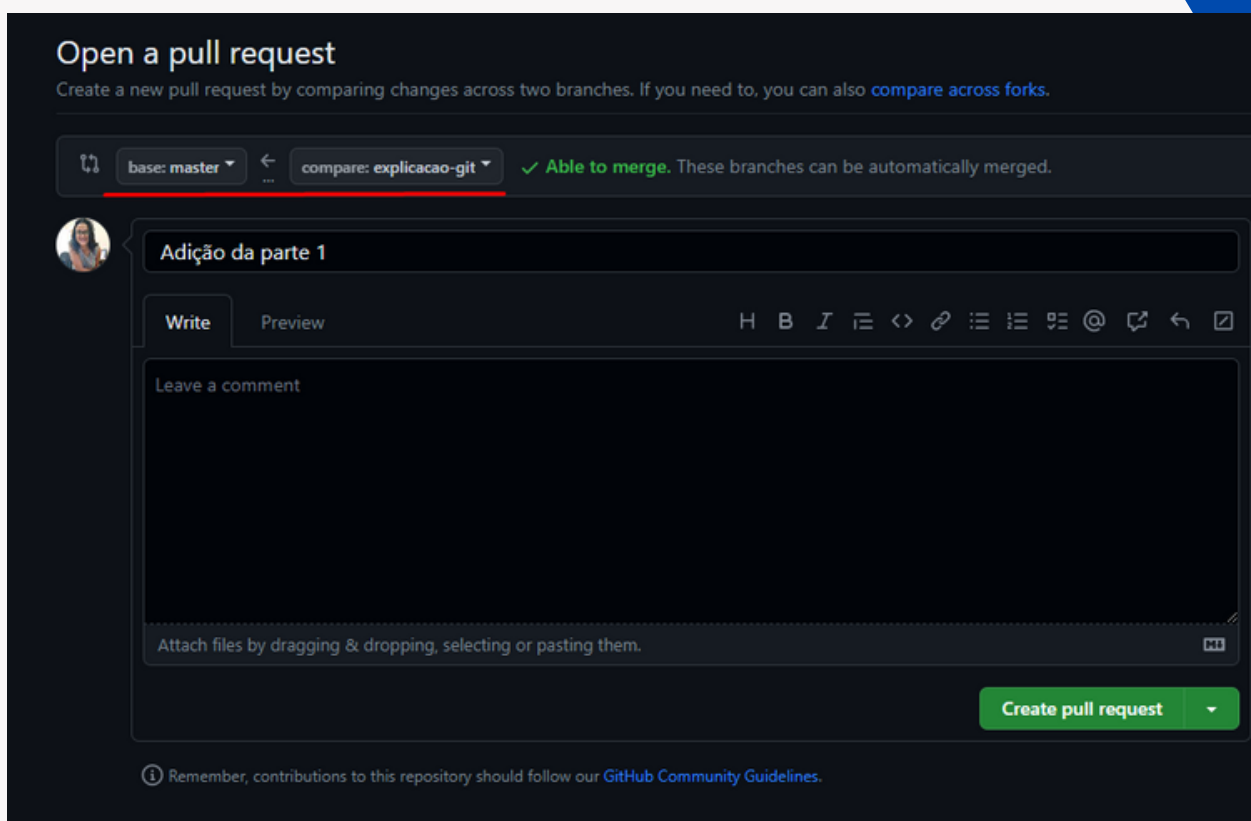


Se você entrar no seu repositório depois de ter mandando uma atualização de uma outra branch você verá essa mensagem que antes não aparecia

É basicamente um aviso falando que uma atualização foi enviada e que pode ser feito um pull request.

Clique no botão Compare & pull request

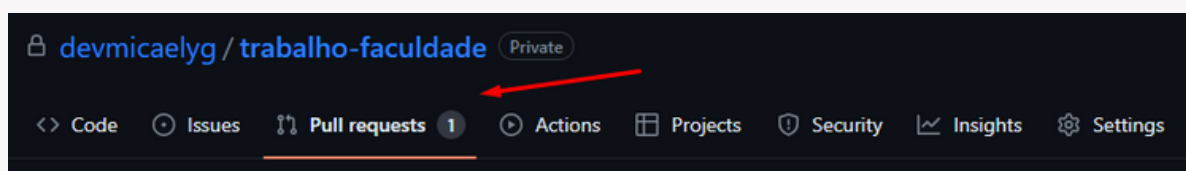
Você será redirecionado para essa tela:



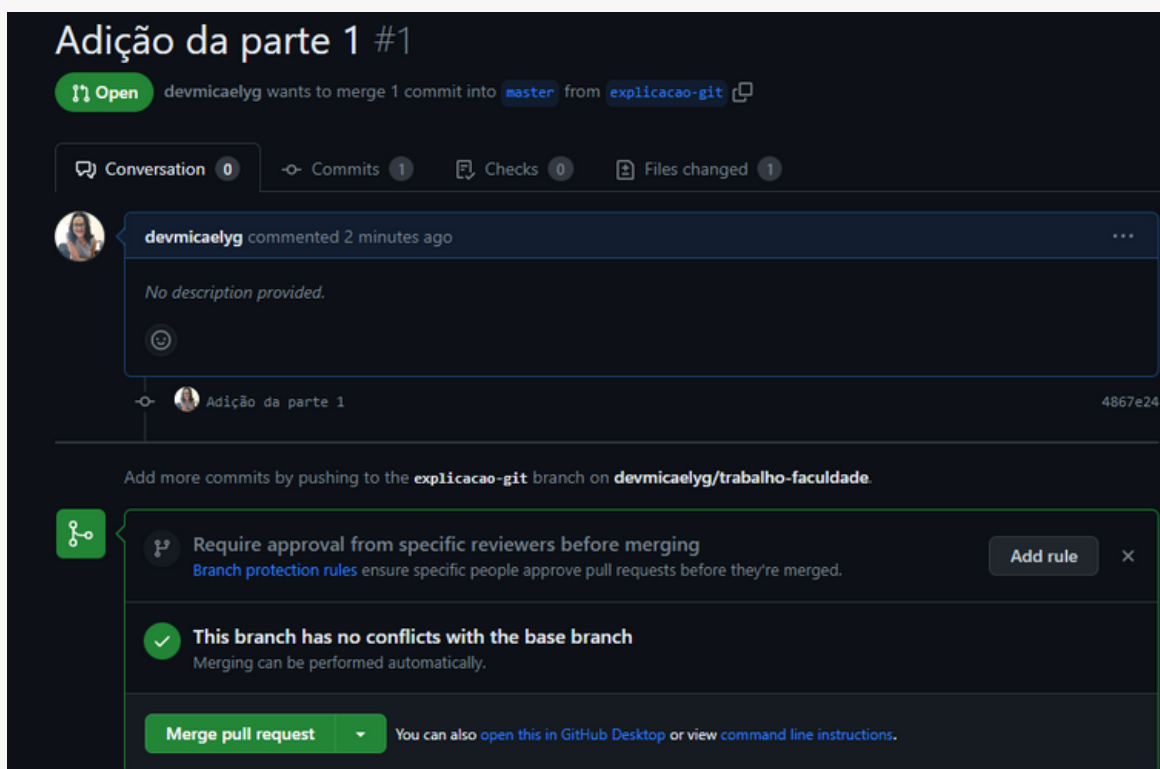
A parte grifada está mostrando que o pull request é referente a pegar as coisas que tem na branch explicacao-git e mandar para a master.

Em seguida você adiciona uma mensagem onde ali está sendo "Adição da parte 1" e aperta no botão Create pull request

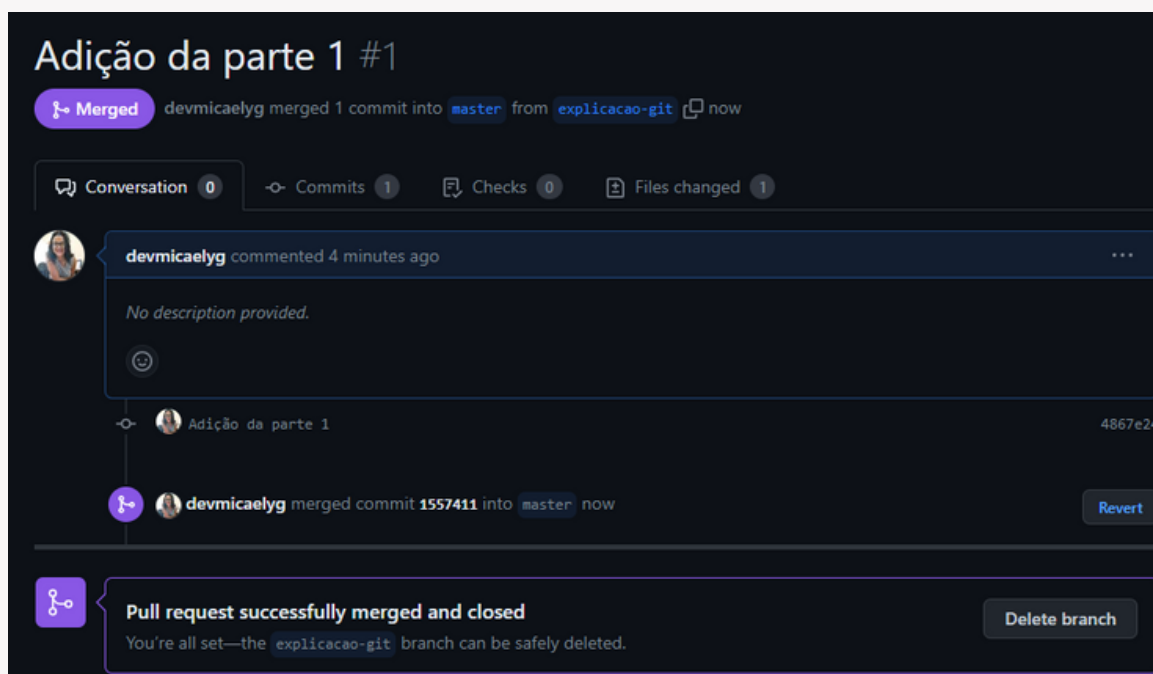
Depois que foi aberto ele fica disponível em uma aba que está sendo indicada pela seta vermelha na imagem seguinte



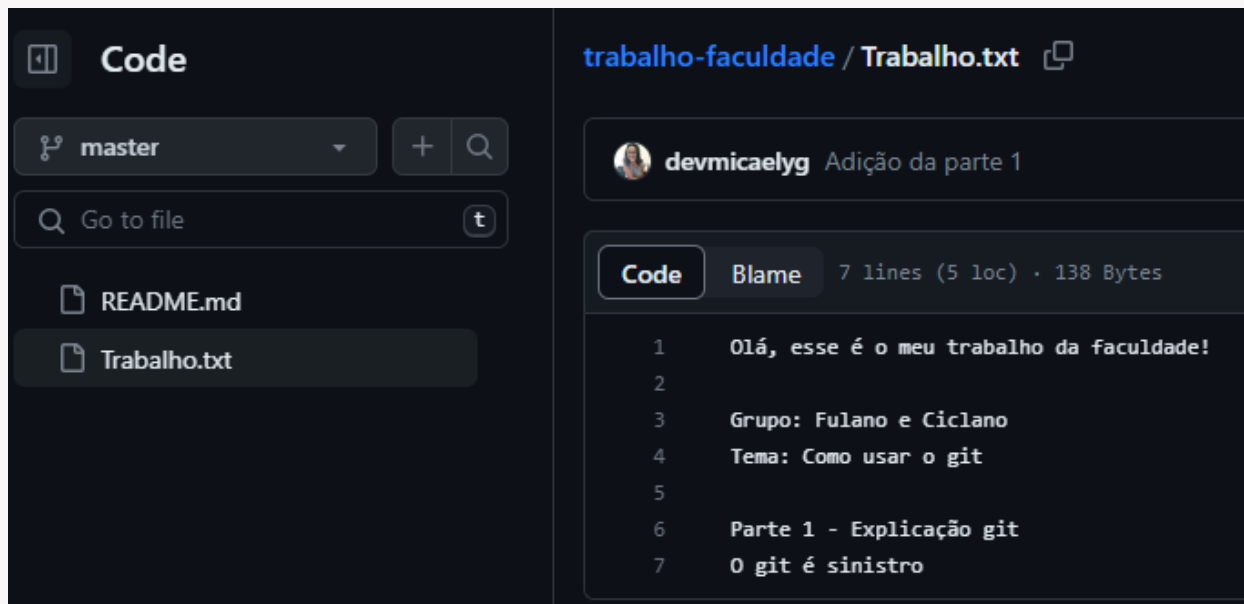
Clicando nessa aba você verá essa tela:



Nessa área é possível ver os detalhes do pull request, quais modificações estão sendo enviadas, se tem algum conflito ou se está livre para que o merge aconteça. Nesse caso está livre, então é só apertar no botão Merge pull request > Confirm merge e está feito



Agora voltando a tela inicial será possível ver que a master recebeu as alterações feitas na branch explicacao-git

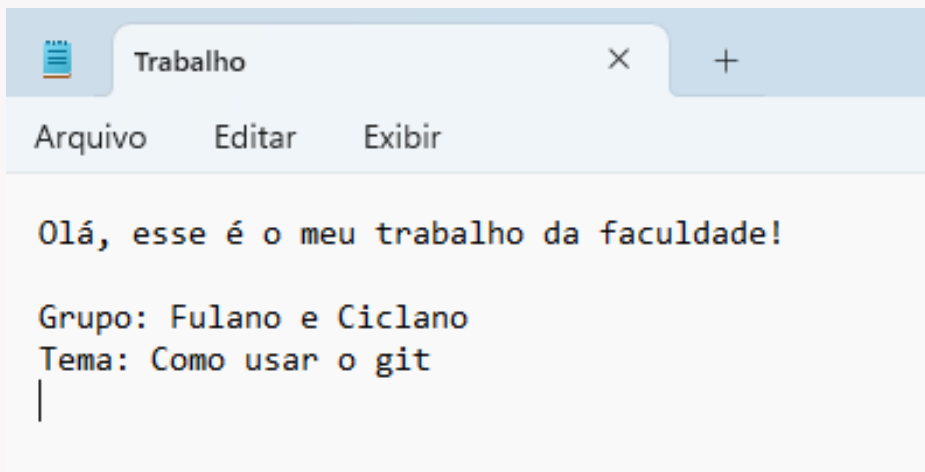


E agora voltando para o nosso repositório local queremos voltar para a master, para isso é só faltar o git checkout mas sem o -b que foi utilizado para criar a branch

```
Micaely@DESKTOP-B2SVK31 MINGW64 ~/Documents/Trabalho faculdade (explicacao-git)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

Micaely@DESKTOP-B2SVK31 MINGW64 ~/Documents/Trabalho faculdade (master)
$ |
```

Agora o nosso repositório local está de volta na master, que tal conferir o nosso Trabalho.txt?



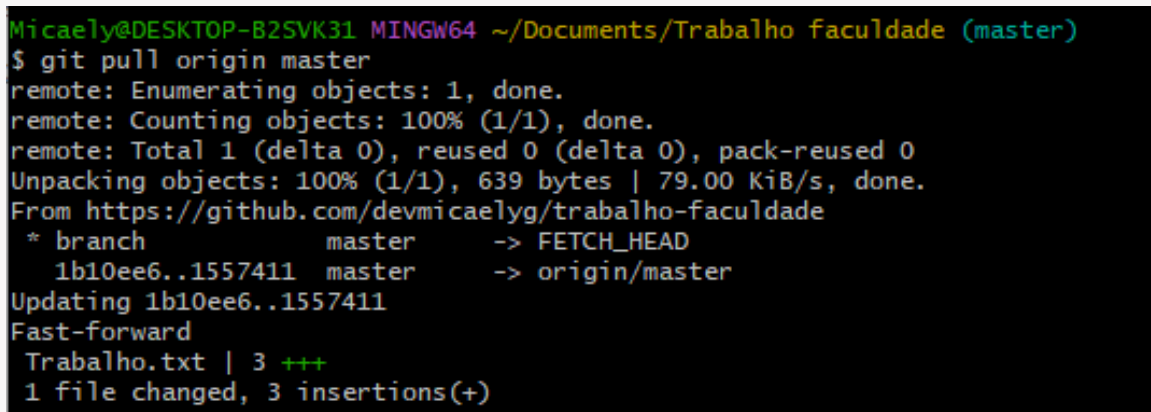
```
Trabalho
Arquivo  Editar  Exibir

Olá, esse é o meu trabalho da faculdade!

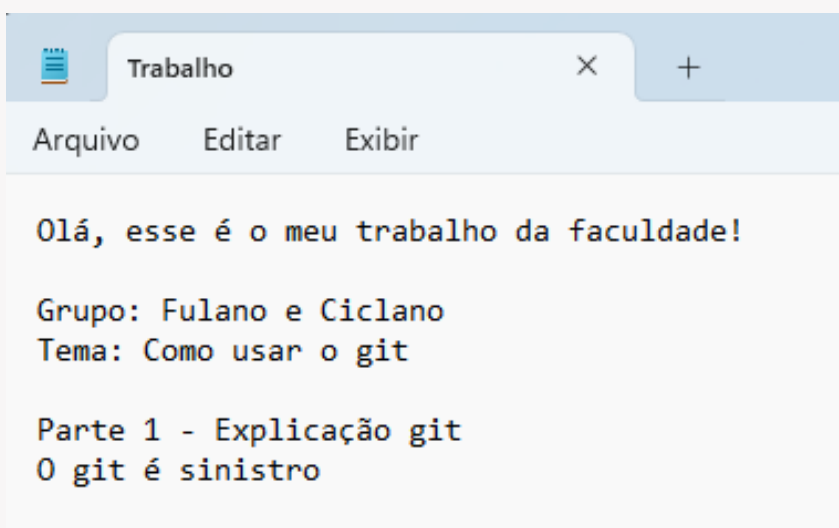
Grupo: Fulano e Ciclano
Tema: Como usar o git
|
```

Ué.. O que aconteceu?

Aconteceu que atualizamos a master remoto e não a local, mas já sabemos que para atualizar é só utilizar um git pull origin master



```
Micaely@DESKTOP-B2SVK31 MINGW64 ~/Documents/Trabalho faculdade (master)
$ git pull origin master
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (1/1), 639 bytes | 79.00 KiB/s, done.
From https://github.com/devmicaelyg/trabalho-faculdade
* branch      master      -> FETCH_HEAD
   1b10ee6..1557411 master    -> origin/master
Updating 1b10ee6..1557411
Fast-forward
 Trabalho.txt | 3 +++
 1 file changed, 3 insertions(+)
```



```
Trabalho
Arquivo  Editar  Exibir

Olá, esse é o meu trabalho da faculdade!

Grupo: Fulano e Ciclano
Tema: Como usar o git

Parte 1 - Explicação git
O git é sinistro
```

Agora sim..!



FINALIZAÇÃO

Agora além de conseguir versionar os seus arquivos e códigos, você também conseguirá trabalhar em equipe sem ter medo dos comandos!

O git tem uma porção de outros comandos a mais além desses apresentados, mas para começar a utilizar ele e se acostumar esses já são mais do que o suficiente.

Espero que esse material tenha te ajudado e que agora você consiga utilizar a força para combater os rebeldes que ainda insistem em não utilizar git

Até a próxima e que a força esteja sempre com você!

