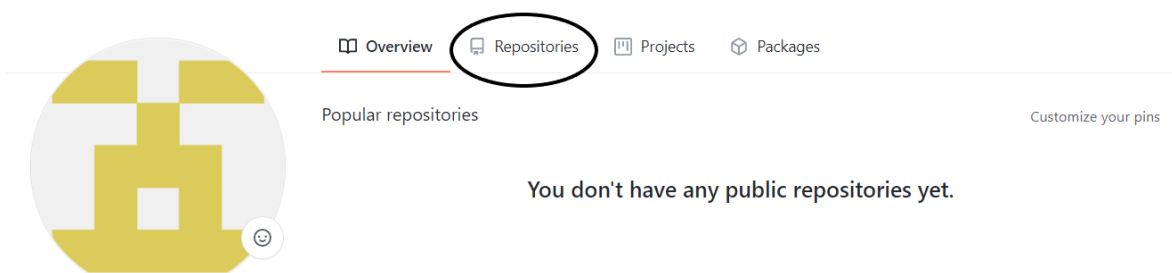


Primeiros passos para clonar um repositório - Mapa Mental

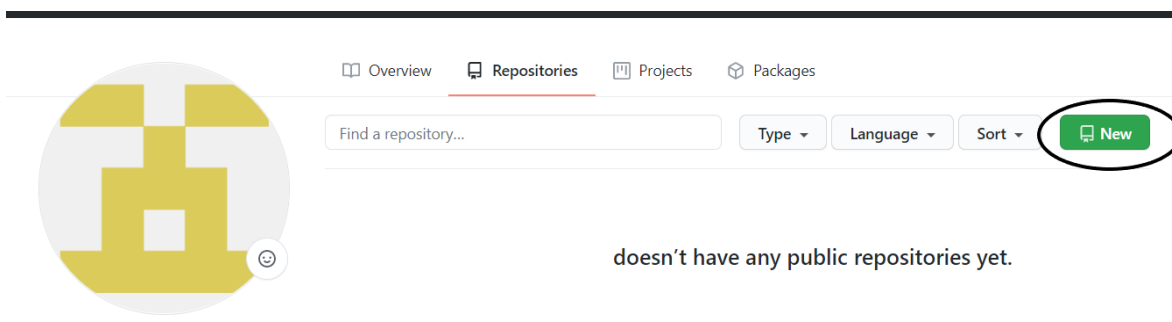


Criando um repositório

- No github lá, na sua conta, clique em “Repositories”.





- Clique no botão “New”




- Coloque o nome do repositório: **Projetos** e pode deixar como **Private**.


Owner * Repository name *

 Aqui aparece seu usuário / Projetos 


Great repository names are short and memorable. Need inspiration? How about [friendly-umbrella?](#)

Description (optional)

☐  **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.

- Abaixo tem o botão Create Repository, clique nele.

☒  **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.](#)

☐ **Choose a license**
A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository

Criando dois arquivos no main

- Iremos criar dois arquivos no main, mas primeiro certifique-se que está em:
~/Projetos/MeuPrimeiroGit (main)

Com o editor nano criaremos os dois arquivos abaixo:

nano index.html

nano style.css

- Vamos visualizar os arquivos

git status

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

- Vamos rastrear os arquivos no git:

git add .

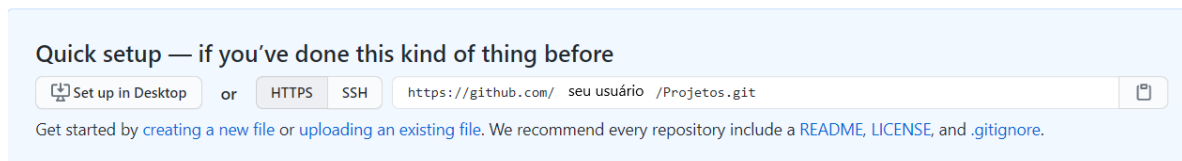
- Na sequência criar um ponto de salvamento:

```
git commit -m "Criando os arquivos index.html e style.css"
```

Enviando arquivos para o repositório Projetos

Para enviar os arquivos remotamente, precisa acessar o github remotamente.

- Copie o código do repositório



```
git remote add origin https://github.com/seuusuario/Projetos.git
```

- Vamos enviar todos os commit do branch local para o repositório remoto identificado pelo nome origin em seu branch remoto nomeado main.

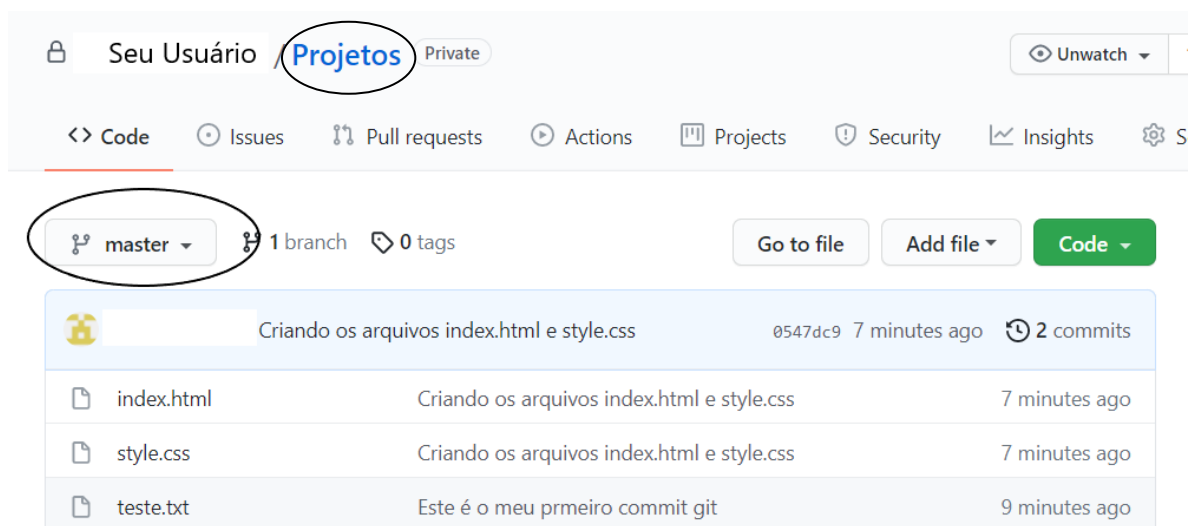
git push

O comando **git push** é usado para enviar o conteúdo do repositório local para um repositório remoto. O comando **push** transfere commits do repositório local a um repositório remoto, ou seja, exporta commits para branches remotos.

```
git push -u origin main
```

```
To https://github.com/seuUsuario/Projetos.git
* [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

Se clicar no link “Projetos” verá o branch “main” com os 3 arquivos.



Obs.: Quando estiver no main, use git push para as próximas vezes.

Criando branches

A branch basicamente são versões diferentes do sistema, a branch principal se chama main.

Para trabalhar em linha de produção cria-se outro branch que servirão de ambiente de teste ou linha de produção.

- Podem existir dois branches:

1º Trabalhando na versão atual do sistema (main)

2º Trabalhando na versão 2.0 do sistema (produção)

- Iremos criar um novo branch

git branch producao (será uma cópia do main)

- Com o comando: git branch, visualiza todas as branches.

git branch

```
* main (mostra qual branch está trabalhando)
  producao
```

- Agora mudaremos de branch main para branch teste.

git checkout produção

```
Switched to branch 'producao'
~/Projetos/MeuPrimeiroGit (producao)
```

- Pode executar o comando git branch para visualizar a alteração.

git branch

```
main
* producao
```

- Vamos enviar todos os commit no branch local para o repositório remoto identificado pelo nome origin em seu branch remoto nomeado producao:

git push -u origin producao

```
To https://github.com/SeuUsuario/Projetos.git
 * [new branch]      producao -> producao
Branch 'producao' set up to track remote branch 'producao' from 'origin'.
```

- Com o comando "ls" você visualiza os arquivos

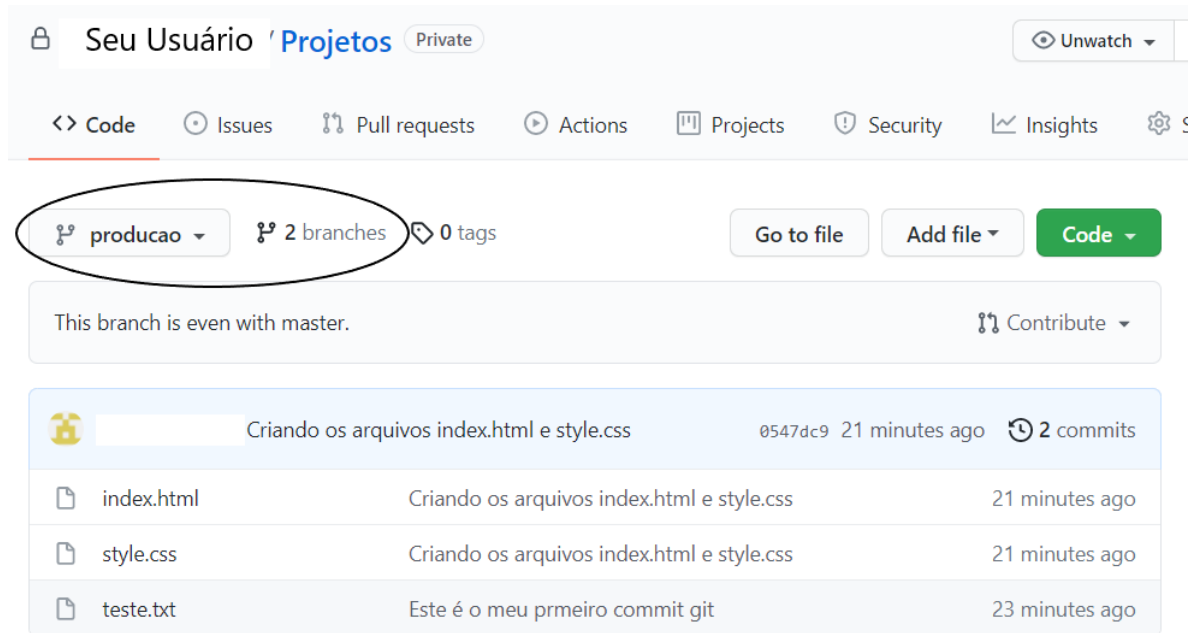
ls

index.html style.css teste.txt

- Agora vamos visualizá-los no github

Mude o DropDown do branch para producao, e veja que ele fez uma cópia dos arquivos que estão na branch main. Note também que mostram 2 branches.

Agora a equipe de desenvolvimento pode trabalhar com uma versão do sistema ou site enquanto a versão atual está em uso.



Desfazendo commit

- Estando no **branch producao**, iremos alterar o index.html, rastrear e cria um ponto de commit.

```
nano index.html (altere o arquivo)
git add index.html ou git add .
git commit -m "Alterando o arquivo index.htm"
```

```
[producao e70c79a] Alterando o arquivo index.html
1 file changed, 1 insertion(+)
```

Observe que trouxe os 7 primeiros números do registro de log do commit, eles são suficientes para desfazer o commit, mas nada impede de usar todos os números do registro de log.

Vamos supor que algo deu errado na alteração do index.html ou o analista falou que não era necessário realizar essa alteração, você precisará voltar o arquivo de origem.

Primeiramente vamos visualizar o log com o git dos commits.

```
git log
```

```
commit e70c79a50919e2a45949ba76f192be584d83dea7 (HEAD -> producao,
origin/producao)
Author: Seuusuario <seu email>
```

Date: Wed Aug 18 22:04:33 2021 -0400

Alterando o arquivo index.html

commit 0547dc910c4470976ea6debd5b5cfcb492128b85 (origin/main, main, checkout)

Author: Seuusuario <seu email>

Date: Wed Aug 18 21:13:42 2021 -0400

Criando os arquivos index.html e style.css

commit 5e5852e1c41d066e19665e6e636e3779aafaf35c

Author: Seuusuario <seu email>

Date: Wed Aug 18 21:11:32 2021 -0400

Este é o meu primeiro commit git

Para voltar tem-se 2 comandos: **soft** e **reset**.

- O **soft** desfaz o commit sem apagar o que foi feito, podem ser alterados e novamente comitado, ideal para um ambiente de produção.

git reset --soft 0547dc9

ou pode ser usado o commit completo

git reset --soft 0547dc910c4470976ea6debd5b5cfcb492128b85

- O **hard** apaga tudo o que foi feito.

git reset --hard 0547dc9

ou pode ser usado o commit completo

git reset --hard 0547dc910c4470976ea6debd5b5cfcb492128b85

- Iremos usar o hard

git reset --hard 0547dc910c4470976ea6debd5b5cfcb492128b85

Vamos ver o log e observe que agora existem apenas 2 commit.

git log

commit 0547dc910c4470976ea6debd5b5cfcb492128b85 (origin/main, main, checkout)

Author: Seuusuario <seu email>

Date: Wed Aug 18 21:13:42 2021 -0400

Criando os arquivos index.html e style.css

commit 5e5852e1c41d066e19665e6e636e3779aafaf35c

Author: Seuusuario <seu email>

Date: Wed Aug 18 21:11:32 2021 -0400

Este é o meu primeiro commit git

Se abrir o index.html notará que ele voltou para o que estava antes do comando hard.

Aqui em um tutorial como excluir um repositório no github

<https://docs.github.com/pt/github/administering-a-repository/managing-repository-settings/deleting-a-repository>

Você pode excluir o repositório do github e a pasta Projetos do seu computador para repetir as etapas novamente. Lembre-se de começar pelo mapa mental.

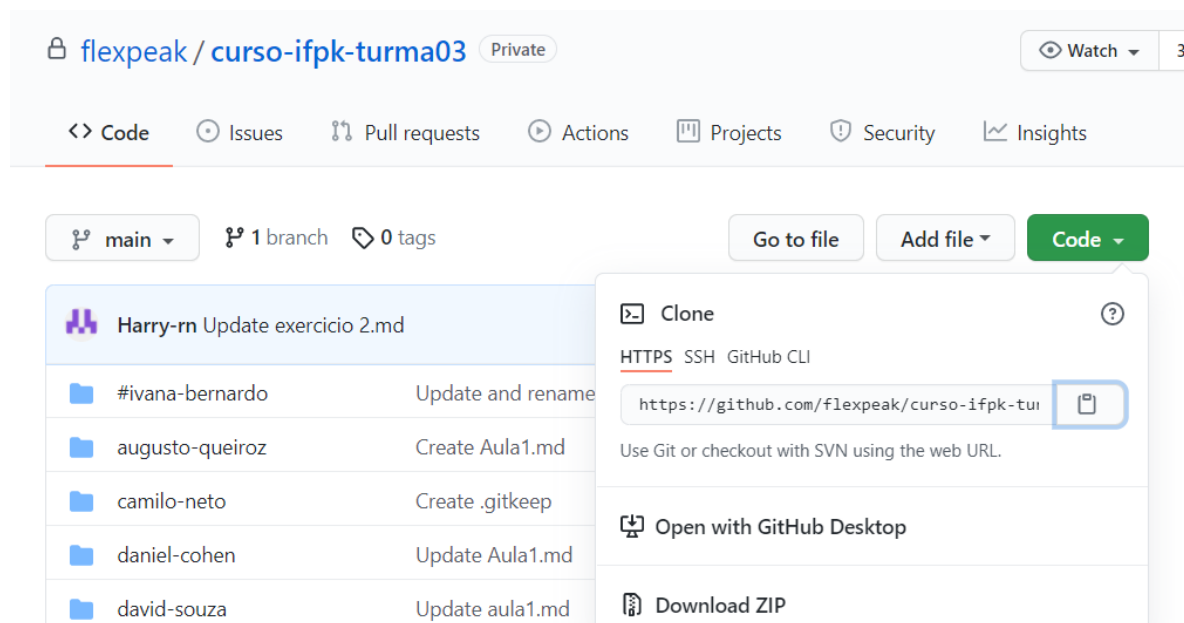
Clonando um repositório

Iremos aprender a clonar do repositório para a máquina local, mas iremos fazer na pasta Projetos que foi criado na máquina. Execute o comando “cd ..” para sair da subpasta.

```
cd ..  
~/Projetos
```

Para exemplificar iremos clonar o repositório do curso.

Entre no repositório e copie o código.



O comando “git clone” é usado para copiar um repositório existente para um diretório local

```
git clone https://github.com/flexpeak/curso-ifpk-turma03.git
```

```
Cloning into 'curso-ifpk-turma03'...  
remote: Enumerating objects: 1219, done.  
remote: Counting objects: 100% (714/714), done.  
remote: Compressing objects: 100% (538/538), done.  
remote: Total 1219 (delta 516), reused 168 (delta 168), pack-reused 505  
Receiving objects: 100% (1219/1219), 9.15 MiB | 661.00 KiB/s, done.  
Resolving deltas: 100% (664/664), done.
```

Prontinho, se listar a pasta Projetos, verá a pasta “curso-ifpk-turma03” e dentro dela todos as pastas e arquivos que estão no github.

```
cd curso-ifpk-turma03
```

```
~/Projetos/curso-ifpk-turma03 (main)
```

```
ls
```

```
'#ivana-bernardo'/  fabio-cantuario/  marla-guedes/  rebecca-souza/
augusto-queiroz/   flavio-silva/     matheus-lima/  robson-lima/
camilo-neto/       francisco-junior/ mauricio-mathias/ thiago-farias/
daniel-cohen/      isabela-monteiro/ peterson-albuquerque/ thiago-santos/
david-souza/       joao-rocha/       rafael-oliveira/ wolfgang-aly/
elton-oliveira/    lorrain-farias/   raimundo-neto/
emilly-araujo/     lucas-lobes/      raissa-lima/
erike-martins/     marcos-costa/     readme.md
```

Ignorando Arquivos

Vamos supor que você tenha um backup de banco de dados ou de senhas e não queira que fique no repositório. Para tal crie um arquivo .gitignore, vamos aos passos:

- 1) Crie o arquivo bd.sql

```
nano bd.sql
```

- 2) Crie o arquivo senhas.txt

```
nano senhas.txt
```

```
Com o git status
```

```
Untracked files:
```

```
(use "git add <file>..." to include in what will be committed)
```

```
bd.sql
```

```
senhas.txt
```

- 3) Crie o arquivo .gitignore

```
nano .gitignore
```

```
coloque nele os nomes dos arquivos que devem ser ignorados:
```

```
bd.sql
```

```
senhas.txt
```

```
salve o arquivo .gitignore
```

```
git status
```

```
Untracked files:
```

```
(use "git add <file>..." to include in what will be committed)
```

```
.gitignore
```


git revert

É utilizado para registrar alguns commits novos para reverter o efeito de alguns commits anteriores (geralmente quando tem um problema). Em vez de remover o commit do histórico do projeto, ele descobre como inverter as alterações introduzidas pelo commit e anexa um commit novo com o conteúdo resultante. Assim, ele evita que o Git perca o histórico, o que é importante para a integridade do histórico de revisão e para uma colaboração confiável.

O git reset não voltar as alterações feitas?

Sim, mas a diferença que com o git reverte ainda tem acesso ao commit feito errado. O git revert é conhecido como o “salvador das sextas-feiras”, pois vamos supor que você fez várias alterações, inseriu novas funcionalidades e no fim do dia você envia o código para a produção (coloca no ar) e acusa um problema. Com o revert volta para o estado anterior para na segunda-feira seja verificado o que pode ter dado errado no código.

Exemplo:

Vamos alterar o style.css

nano style.css, altere o conteúdo

```
git add .
```

```
git commit -m "Revert style.css"
```

```
git log
```

```
commit Odd5b605ad8632688160234d05d1140eb2457440 (HEAD -> main, origin/main)
```

```
Author: SeuUsuario <seu_email>
```

```
Date: Fri Jan 14 18:42:29 2022 -0400
```

```
Revert style.css
```

Para reverter a alteração

```
git revert --no-edit Odd5b605ad8632688160234d05d1140eb2457440
```

```
[main 5533c2c] Revert "Revert style.css"
```

```
Date: Fri Jan 14 18:56:34 2022 -0400
```

```
1 file changed, 2 deletions(-)
```

--no-edit => serve para não abrir o editor

Abrindo o editor verificar que voltou ao estado anterior.

nano style.css

Visualizando o log, verificará que as alterações permanecem e futuramente pode voltar para o commit corrigir o erro e colocar em produção a nova alteração.

```
git log
```

author: SeuUsuario <seu_email>

Date: Fri Jan 14 18:56:34 2022 -0400

```
Revert "Revert style.css"
```

This reverts commit Odd5b605ad8632688160234d05d1140eb2457440.

commit Odd5b605ad8632688160234d05d1140eb2457440 (origin/main)

Author: SeuUsuario <seu_email>

Date: Fri Jan 14 18:42:29 2022 -0400

```
Revert style.css
```

Removendo branches do repositório remoto

Para remover o branch do repositório basta dar o comando:

```
git push origin :producao
```

removendo branches locais

Para deletar um branch precisa estar fora dele, vamos criar a branch teste e depois deletar.

Criando

```
git branch teste
```

Deletando

```
git branch -D teste
```

Deleted branch teste (was 5533c2c).

git pull

O comando git pull é usado para buscar e baixar conteúdo de repositórios remotos e fazer a atualização imediata ao repositório local para que os conteúdos sejam iguais.

Vamos a um exemplo:

Como não temos outra pessoa usando, então iremos ao repositório e criaremos um arquivo, por exemplo, teste.js

Add file + Create new file + coloca o nome teste.js e botão Commit new file

O arquivo foi criado no repositório, ou seja, alguém criou diretamente no repositório ou outra pessoa está usando o mesmo repositório e executou o comando push.

Se executar o comando git status

```
git status
```

On branch main

```
Your branch is up to date with 'origin/main'.
```

```
nothing to commit, working tree clean
```

Informará que nada precisa ser comitado, mas na verdade o remoto foi alterado.

Então você executará o comando pull para pegar as atualizações do repositório remoto e enviar para o repositório local.

```
git pull origin main
```

```
Username for 'https://github.com': usuario
```

```
Password for 'https:// usuario@github.com':
```

```
remote: Enumerating objects: 7, done.
```

```
remote: Counting objects: 100% (7/7), done.
```

```
remote: Compressing objects: 100% (4/4), done.
```

```
remote: Total 6 (delta 2), reused 0 (delta 0), pack-reused 0
```

Unpacking objects: 100% (6/6), 1.26 KiB | 2.00 KiB/s, done.

From https://github.com/usuario/Projetos_

```
* branch          main          -> FETCH_HEAD
   5533c2c..dd828de  main          -> origin/main
```

Updating 5533c2c..dd828de

Fast-forward

```
teste.js | 1 +
1 file changed, 1 insertion(+)
create mode 100644 teste.js
```

Com o git log ele mostra que fez o pull no repositório local

git log

commit dd828de0cb8e23bf1dfd897fec366544e3f2f830 (HEAD -> main, origin/main)

Author: Usuario <84885503+usuario@users.noreply.github.com>

Date: Mon Jan 17 21:10:03 2022 -0400

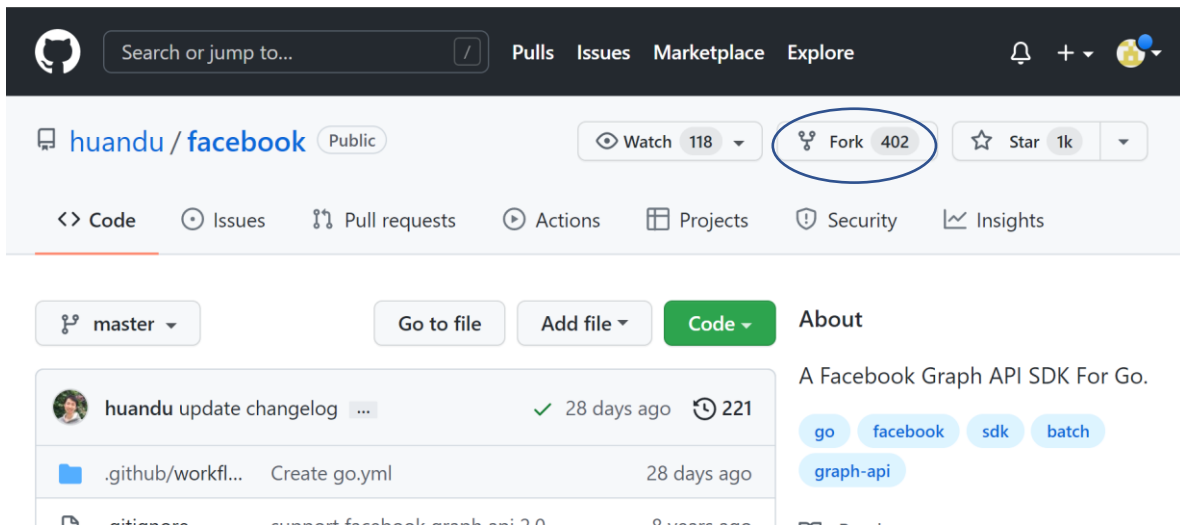
Update teste.js

Uma recomendação quando se trabalha em equipe é você dar o comando pull para enviar para o local e depois um push para o remoto, dessa forma irá garantir que o sistema está atualizado.

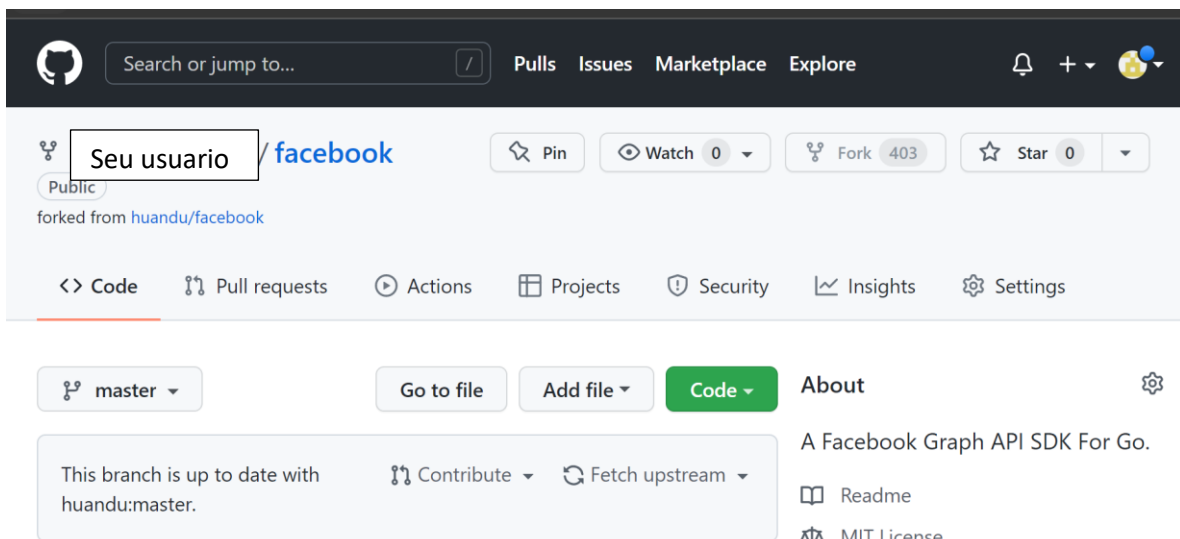
Comando fork

Contribuindo com outros projetos que não são os nossos. Por exemplo, uma API do facebook: <https://github.com/huandu/facebook>

Para contribuir precisa fazer um fork



Quando o comando fork é feito, já aparece no seu usuário, ou seja, um repositório idêntico dentro do seu usuário.



É possível clonar para o eu repositório local:

```
git clone https://github.com/SeuUsuario/facebook.git
```

```
Cloning into 'facebook'...
```

```
remote: Enumerating objects: 827, done.
```

```
remote: Counting objects: 100% (21/21), done.
```

```
remote: Compressing objects: 100% (15/15), done.
```

```
remote: Total 827 (delta 5), reused 12 (delta 4), pack-reused 806
```

```
Receiving objects: 100% (827/827), 410.66 kiB | 829.00 kiB/s, done.
```

```
Resolving deltas: 100% (525/525), done.
```

```
ls -l
```

```
total 12
```

```
-rw-r--r-- 1 mega 197121 43 Jan 17 14:15 README.md
```

```
drwxr-xr-x 1 mega 197121 0 Jan 17 21:34 facebook
```

```
-rw-r--r-- 1 mega 197121 60 Jan 17 14:15 index.html
```

```
-rw-r--r-- 1 mega 197121 107 Jan 17 14:15 style.css
```

```
-rw-r--r-- 1 mega 197121 24 Jan 17 21:16 teste.js
```

Entra na pasta do facebook

```
cd facebook
```

```
ls -l
```

```
total 207
```

```
-rw-r--r-- 1 mega 197121 9569 Jan 17 21:34 CHANGELOG.md
```

```
-rw-r--r-- 1 mega 197121 186 Jan 17 21:34 CONTRIBUTING.md
```

```
-rw-r--r-- 1 mega 197121 1070 Jan 17 21:34 LICENSE
```

```
-rw-r--r-- 1 mega 197121 14719 Jan 17 21:34 README.md
```

```
-rw-r--r-- 1 mega 197121 6495 Jan 17 21:34 api.go
```

```
-rw-r--r-- 1 mega 197121 4238 Jan 17 21:34 api_test.go
```

```
-rw-r--r-- 1 mega 197121 7427 Jan 17 21:34 app.go
```

```
-rw-r--r-- 1 mega 197121 671 Jan 17 21:34 app_test.go
```

```
-rw-r--r-- 1 mega 197121 1103 Jan 17 21:34 batch_result.go
```

```
-rw-r--r-- 1 mega 197121 1954 Jan 17 21:34 conversion.go
```

```
-rw-r--r-- 1 mega 197121 774 Jan 17 21:34 conversion_test.go
```

```
-rw-r--r-- 1 mega 197121 1266 Jan 17 21:34 error.go
```

```
-rw-r--r-- 1 mega 197121 7750 Jan 17 21:34 facebook_test.go
```

```
-rw-r--r-- 1 mega 197121 1880 Jan 17 21:34 filedata.go
```

```
-rw-r--r-- 1 mega 197121 1324 Jan 17 21:34 filedata_test.go
```

```
-rw-r--r-- 1 mega 197121 49 Jan 17 21:34 go.mod
```

```
-rw-r--r-- 1 mega 197121 3360 Jan 17 21:34 paging_result.go
```

```
-rw-r--r-- 1 mega 197121 7256 Jan 17 21:34 params.go
```

```
-rw-r--r-- 1 mega 197121 1952 Jan 17 21:34 params_test.go
```

```
-rw-r--r-- 1 mega 197121 38131 Jan 17 21:34 result.go
```

```
-rw-r--r-- 1 mega 197121 27901 Jan 17 21:34 result_test.go
```

```
-rw-r--r-- 1 mega 197121 17671 Jan 17 21:34 session.go
```

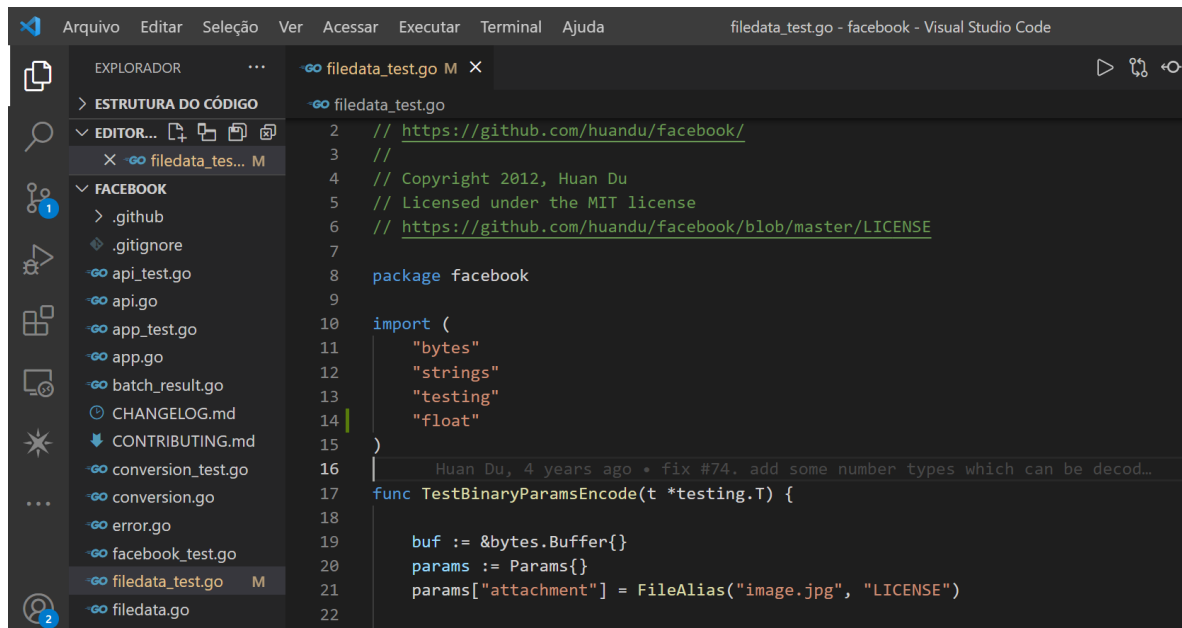
```
-rw-r--r-- 1 mega 197121 10434 Jan 17 21:34 session_test.go
```

Na pasta do facebook iremos abrir o Vcode.

code .

Veja que está escrito em linguagem “go” e o vcode pedirá para instalar, não se faz necessário a instalação para o nosso exemplo.

Vamos alterar o arquivo filedata_test.go e colocar o tipo “float” no import. Salve a alteração.



Com o comando git status veremos a modificação no arquivo.

```
git status
```

```
on branch main
```

```
Your branch is up to date with 'origin/main'.
```

```
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:   filedata_test.go
```

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

Vamos adicionar e comitar.

```
git add .
```

```
git commit -m "Alterando o arquivo filedata_test.go"
[main b8c3a7e] Alterando o arquivo filedata_test.go
1 file changed, 1 insertion(+)
```

Vamos verificar o nome da branch: main

```
git branch
* main
```

Vamos verificar o nome remoto: origin

```
git remote -v
origin https://github.com/seuUsuario/facebook.git (fetch)
origin https://github.com/seuUsuario/facebook.git (push)
```

Agora que já sabemos o nome da branch e o repositório remoto podemos fazer o push para o repositório.

```
git push origin main
Username for 'https://github.com': seuUsuario
Password for 'https://seuUsuario@github.com':
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), 318 bytes | 159.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/seuUsuario/facebook.git
    01c37fb..b8c3a7e  main -> main
```


Showing 1 changed file with 1 addition and 0 deletions.

Split Unified

```
▼ 1 filedata_test.go
```

↑		@@ -11,6 +11,7 @@ import (
11	11	"bytes"
12	12	"strings"
13	13	"testing"
14	+	"float"
14	15)
15	16	
16	17	func TestBinaryParamsEncode(t *testing.T) {

↓

As alterações ainda estão no seu repositório. Para enviar: Pull request

[Code](#) [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Insights](#) [Settings](#)

master

Go to file

Add file

Code

About

This branch is 1 commit ahead of huandu:master.

Contribute

Fetch upstream

Alterando o arquivo fil... 15 minutes ago 222

.github/workfl... Create go.yml 28 days ago

.gitignore support facebook graph api 2.0 8 years ago

A Facebook Graph API SDK For Go.

Readme

MIT License

0 stars

0 watching

403 forks

Em seguida botão New pull request

[Code](#) [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Insights](#) [Settings](#)

Filters

is:pr is:open

Labels 9

Milestones 0


New pull request

0 Open 0 Closed

Verifique se o destino está correto e se realmente é o que você vai enviar. Rolando mais um pouco a página verá as suas alterações no código. Estando tudo certo botão Create pull request.

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).



base repository: huandu/facebook ▾

base: master ▾

←

head repository: [redacted]/facebook ▾

compare: master ▾


✓ **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

Ele mostrará a tela abaixo onde você descreverá o que alterou e o porquê da alteração, em seguida, botão Create pull request. Onde será enviado para o repositório de origem e irão avaliar se irão aceitar ou rejeitar a alteração. No nosso exemplo, não iremos enviar 😊.

Se o pull request for aceito vai entrar no projeto original.



Alterando o arquivo filedata_test.go


Write


Preview

H B I ≡ <> 🔗 ≡ ≡ ☑

@ ↗ ↶ ▾

Leave a comment

Attach files by dragging & dropping, selecting or pasting them. 

☒ Allow edits by maintainers 

Create pull request ▾