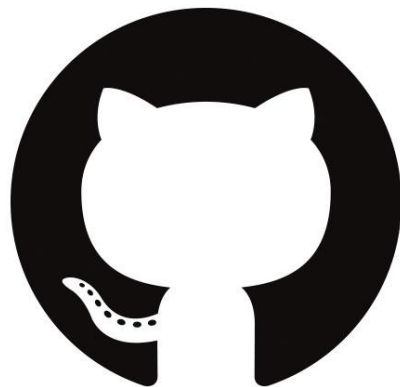


# GIT & GITHUB

Um pequeno orientador para web-navegantes de primeira  
viagem



# AFINAL DE CONTAS, O QUE É GIT?

Git é uma ferramenta muito utilizada com o objetivo de obter melhor controle das versões de um mesmo código. O que também é chamado de versionamento de código.

GitHub é um complemento que permite acessar as funcionalidades do GIT online e permite compartilhar projetos e colaboração de diferentes equipes neles.



Na wiki oficial, há explicações alternativas do próprio Torvald, co-fundador do Git, explicando que..

"(...)O *git* pode significar qualquer coisa, depende de seu humor. Pode ser, meramente uma combinação de três letras pronunciáveis e não utilizadas atualmente por nenhum comando comum do UNIX; o retrô acrônimo de *Global information tracker*, em português, *Rastreamento global de informações*; ou, quando trava, "Goddamn idiotic truckload of sh\*t" <sup>1</sup>



Hoje não há como negar sua  
tamanho importância para  
desenvolvedores, e empresas de  
tecnologia desde as pequenas quanto  
as grandes..

Atualmente os serviços prestados pela GIT e GitHub pertencem à Microsoft.

(1) [https://git.wiki.kernel.org/index.php/GitFaq#Why\\_the\\_.27git.27\\_name.3F](https://git.wiki.kernel.org/index.php/GitFaq#Why_the_.27git.27_name.3F)

# VEJA SÓ ESSES PROJETOS DO GIT!

## **Conheça algumas listas de projetos Open Source para você contribuir**

<https://medium.com/trainingcenter/saiba-como-contribuir-com-projetos-open-source-e-conhe%C3%A7a-o-ct-d916a30fcdb7>

## **10 grandes empresas que estão no GitHub e o que elas estão fazendo lá**

<https://www.neuroniodigital.com.br/empresas-no-github/>

## **10 projetos que têm conquistado desenvolvedores no GitHub**

<https://computerworld.com.br/2018/02/16/10-projetos-que-tem-conquistado-desenvolvedores-no-github/>

Sério, pode  
abrir que não  
é vírus!





Para fazer o download e instalação do GIT é preciso identificar o seu sistema: [neste link](#) para Windows; [neste](#) para sistemas Mac então através do comando `sudo apt-get install git` para plataformas Linux/Debian, escolha sabiamente.

Se você usa uma (Máquina Virtual)VM na nuvem, como o cloud9 ou koding, o git já estará disponível em sua linha de comando.

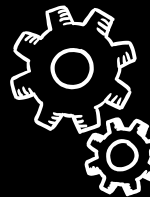
Para verificar, abra uma sessão de terminal e usando o comando:

```
git --version
```

Se retornou algo como "`git --version X.XX.X`" você já possui o git instalado.

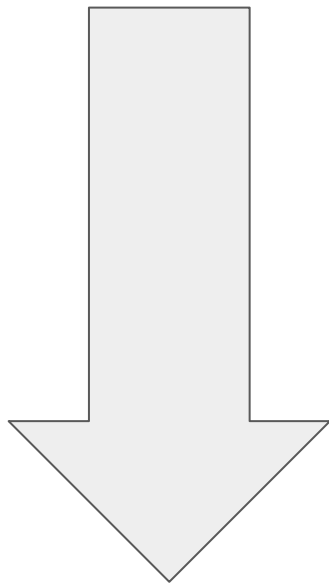


<https://pt.wikihow.com/Criar-uma-Conta-no-GitHub>

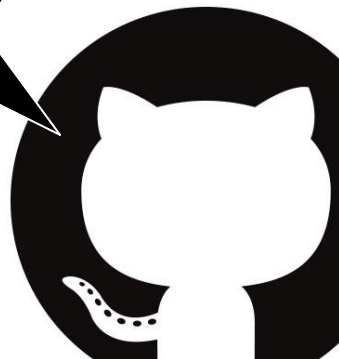


# CONFIGURANDO O GIT

Após criar a conta, você verá um botão chamado `+New Repository` na qual poderá criar um repositório de acordo com a tela a seguir.



**Segue no  
próximo  
slide..**



Muitas informações?

Fica tranquilx pq vamos detalhar...é fundamental criar um repositório como este para os nossos testes.

Nesta imagem estamos criando um repositório **OlaMundo**, a opção **Public** está marcada então o repositório é de domínio público.


Se a opção, **README** estiver ativa automaticamente será incluída nele a descrição acima.

Proprietário

Nome do repositório

Owner


Repository name \*

 solidarizesw ▾

/

OlaMundo

✓




Great repository names are short and memorable. Need inspiration? How about **fluffy-engine**?

Description (optional)


BREVE Descrição do projeto

Teste do Repositório "OlaMundo"

☒  Public

O repositório é Público (qualquer um pode ver)

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

☐  Private

O repositório é Privado (somente a organização podem ver)


Your current plan does not support private repositories. Your organization's owners will need to upgrade to Team.

Skip this step if you're importing an existing repository.

☒ Initialize this repository with a README

This will let you immediately clone the repository to your computer.

Add .gitignore: None ▾

Add a license: None ▾ 

Create repository

Criar repositório

Se tudo deu certo por aqui, e eu espero que sim, o link abre a tela principal do seu projeto:

The screenshot shows the GitHub interface for a repository named 'olaMundo'. At the top, there's a header with the repository name and navigation links like 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. Below this, the repository name 'olaMundo' is displayed, followed by statistics: 1 commit, 1 branch, 0 packages, 0 releases, and 1 contributor. A section for the initial commit shows the file 'README.md' with a commit message 'Nome do arquivo' and a timestamp '15 minutes ago'. The main content area displays the repository name 'olaMundo' and the subtitle 'Teste do Repositório "olaMundo"'. A speech bubble points to the repository name, stating: 'Lembrando que nome do usuário é o login que você usou para criar sua conta!'. The GitHub logo is visible in the bottom right corner.

Nome de usuário/olaMundo

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Actions Projects 0 Wiki Security Insights Settings

Teste do Repositório "olaMundo" **Após a criação do repositório, ele estará disponível em**  
<https://github.com/nome do usuário/site> **.** Edit

Manage topics

1 commit 1 branch 0 packages 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

Nome de usuário initial commit **Data de atualização** Latest commit 05a87d6 15 minutes ago

README.md **Nome do arquivo** Initial commit 15 minutes ago

README.md

**olaMundo**

Teste do Repositório "olaMundo"

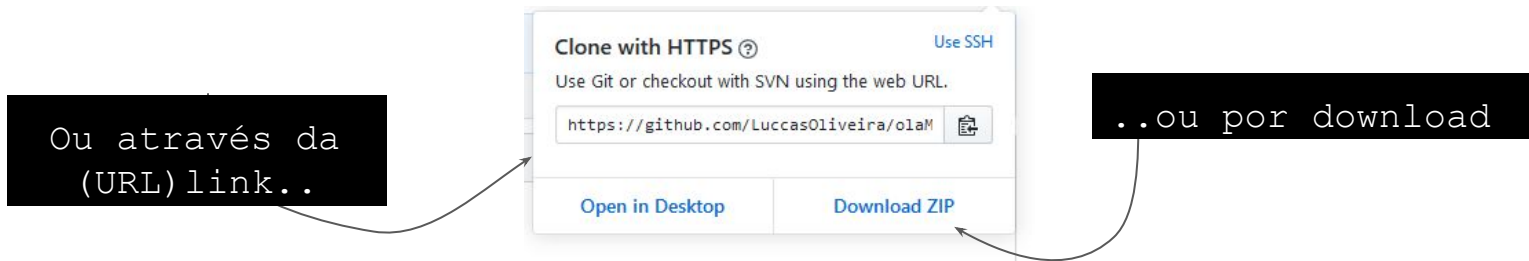
Lembrando que **nome do usuário** é o login que você usou para criar sua conta!



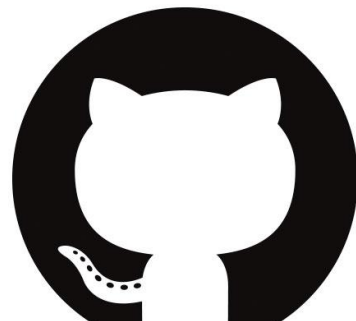
Observe o último à direita; o botão de cor verde. Este permite que baixe o projeto inteiro através de dois modos:



## 1) Clique no botão de copiar URL:



Essa operação será necessária para que possamos “clonar” este projeto em nosso ambiente de trabalho (sua máquina windows, mac, linux ou a vm).

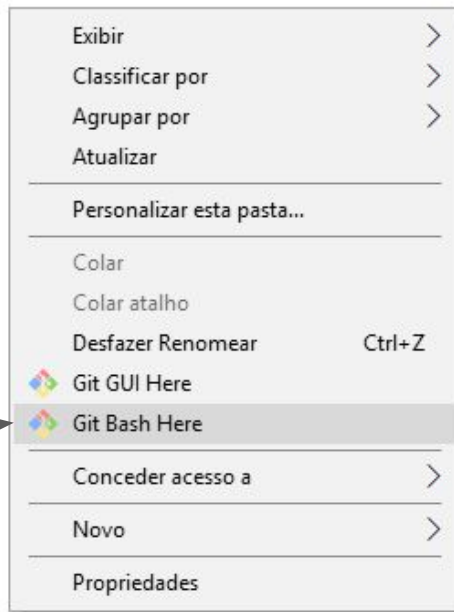


Antes de iniciar, é necessário configurar um usuário, que realizará as alterações nos projetos para isso use os comandos:

2) Clique uma pasta no sistema:

2.1) Clique com o botão esquerdo dentro da pasta

3) Clique na opção **Git Bash Here**



ATENÇÃO

No **Linux** ou no **MAC** o git estará lá disponível no terminal/console, sendo este passo desnecessário.

Antes de iniciar, é necessário configurar um usuário, que realizará as alterações nos projetos, para isso:

4) Digite os seguintes comandos:

```
git config --global user.name "Nome de usuário"  
git config --global user.email "Meu email cadastrado na conta"
```

Estas configurações ficam alocadas no arquivo `~/.gitconfig`, onde o `~` é o seu diretório principal.

..para confirmar as configurações é possível através do comando:

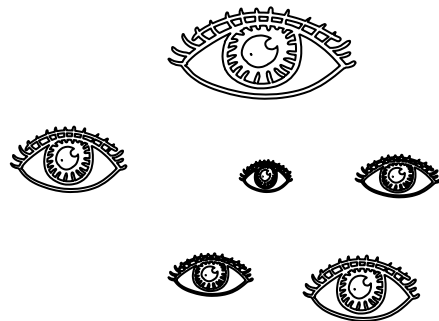
```
git config --list
```

..vai exibir uma série de configurações e por último seu nome de usuário e email.

# ALGUNS COMANDOS ÚTEIS

Se, em algum momento, trocamos de máquina, isso pode estragar nosso histórico no Git , pois estaremos com um usuário ou email diferente.

Pensando nisso, listamos os seguintes comandos:



Verificando as configurações locais

**Para encontrar o nome de usuário**

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

**Para encontrar o email**

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

Alterando as configurações locais  
o novo valor passado como parâmetro entre  
aspas!

**Alterar o nome de usuário**

```
git config --global user.name "nome do usuário"
```

**Alterar o email**

```
git config --global user.email "email do usuário"
```

# CLONAR!

Vamos trazer este projeto para o nosso git, este processo se chama **clonar**, com o terminal aberto, cole:

```
git clone https://github.com/nomedeusuario/olaMundo.git
```

Ao clonar, uma pasta com o nome do projeto é criada na máquina com o nome do projeto baixado, é esperado os seguintes elementos de execução:

```
Cloning into 'olaMundo'...  
remote: Enumerating objects: 3, done.  
remote: Counting objects: 100% (3/3), done.  
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0  
Unpacking objects: 100% (3/3), done.
```

Perceba que na pasta criada estará incluída **mais uma nova pasta** com o nome do projeto e os arquivos dele.

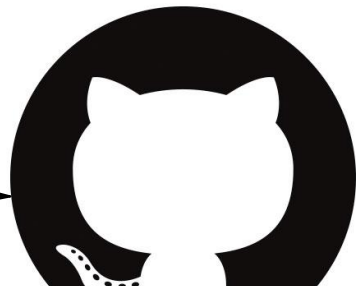


OMG! MEU PROJETO É PÚBLICO?!

QUALQUER UM PODE BAIXAR MEU PROJETO?!?! ..

VOU PERDER MEU PROJETO, VEY! ..

CALMA! Se o seu repositório está **público**, qualquer um pode `clonar` ele para si, mas não podem alterar os seus arquivos, exceto que você permita!



# COMANDOS IMPORTANTES

Com o projeto baixado, nós vamos manipulá-lo através dos seguintes comandos:

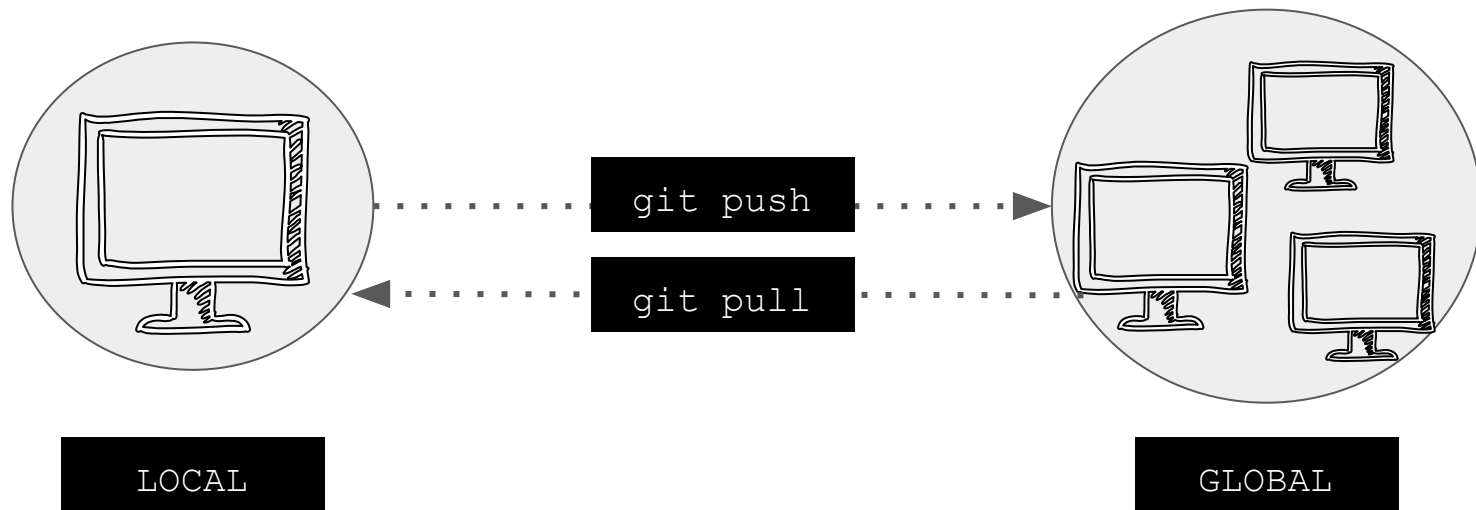
`git add <arquivos...>` Este comando permite adicionar arquivos ao projeto. Esses arquivos ficam em uma pasta separada chamada INDEX, ao digitar `git push` desses arquivos serão adicionados.

`git commit -m "comentário qualquer"` Todos os arquivos que estão no INDEX que o comando `git add` adicionou e criar uma revisão com um número e um comentário para que todos possam visualizar as alterações feitas.

`git status` Para sabermos se tem algo que foi modificado enquanto estamos mexendo, rodamos este comando.

É importante estar atento que os comandos acima **somente** funcionarão dentro da pasta baixada.



O comando `git push` empurra as suas modificações para o servidor, incluindo todas no histórico do projeto, para todos os novos acessantes receberem.



..é importante perceber que há dois “estados” no projeto: antes e após a execução do comando `git push`. Caso eu não execute o comando, não haverá nenhuma atualização



Após realizar o `git push` podemos ver no site do github as mudanças realizadas no projeto:


 LuccasOliveira adicionando comentário ao aloMundo		Latest commit 7022b3a 1 hour ago
nome de usuário	Initial commit	6 hours ago
 documentoNovo.txt	adicionando comentário ao aloMundo	1 hour ago

Até aqui, conseguimos ver todo o processo de **versionamento de arquivos** com git e github.

VISH, errei no `commit`! E agora?

`git commit --amend` modifica a mensagem do último `commit` feito por você, usuário do projeto.

Ele abrirá um editor de texto para editar faça as alterações necessárias e digite `:w` para gravar as atualizações e `:q` para sair do editor. APÓS o `git push` é possível ver as atualizações feitas no site.

 documentoNovo.txt	adicionando documento ao aloMundo	10 minutes ago
---	-----------------------------------	----------------

# Avançando EM COMANDOS

O comando `git add`, adiciona qualquer arquivo ao commit, quando este for realizado. Isso é necessário porque nem sempre queremos que todos os arquivos que alteramos sejam “comitados”.

Por exemplo: ao salvar outro arquivo na pasta do projeto. No terminal, receberemos a seguinte mensagem ao executar o `git status`

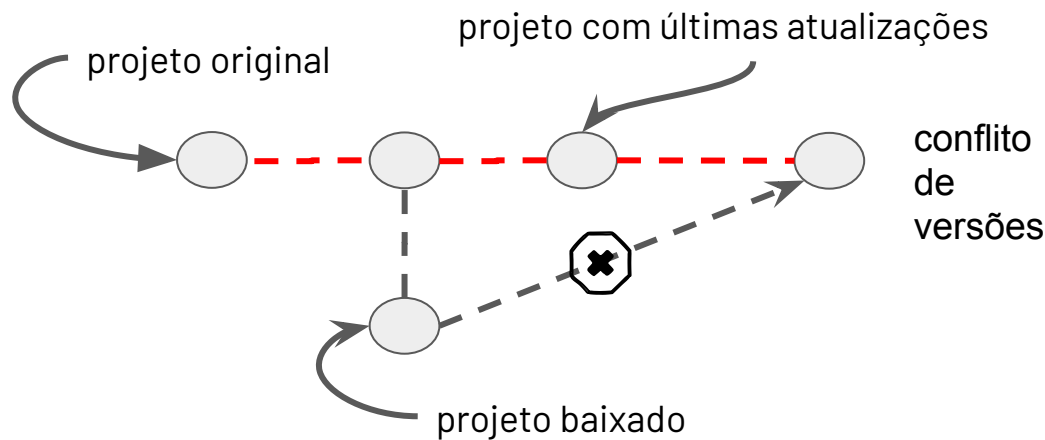
```
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    outroArquivo.txt

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

Agora, altere seu conteúdo, e use o seguinte comando no terminal `git commit -m "Alteração feita no: outroArquivo.txt"` e depois, `git push`. Interessante observar que a ultima atualização, não foi feita, até que você faça novamente o comando `git add`.

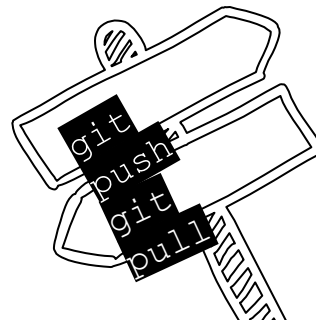
# "BRANCH" (RAMIFICAÇÃO)

E quando há “dilatações temporais” entre a nova atualização e a atualização anterior..



O comando `git pull` é utilizado para receber na máquina atual, a versão recente do projeto. Por isso, ao iniciar a atividade é importante verificar se há atualizações e manter sincronia.

Além disso é muito importante que a comunicação entre os membros da equipe para que todos estejam cientes de suas funcionalidades. Em geral, os desenvolvedores envolvidos no projeto, recebem a demanda através de **issues**. Deixadas no site do git, as tarefas podem ser divididas em atividades menores para resolvê-las.



# "Branch" (RAMIFICAÇÃO)

*Branch*, ou "ramo" que é ramificar o seu projeto em 2, como se cada um deles fosse um repositório, e depois juntá-los novamente. Observe abaixo o print do site:



Você já está em um "ramo"

Até agora fizemos as alterações somente no master.

Repare também que o comando `git status` também exibe em qual "ramo" estamos!

```
$ git status
On branch master
Your branch is up to date with
'origin/master'.

nothing to commit, working tree clean
```

# "BRANCH" (RAMIFICAÇÃO)

Inicialmente, pelo github, criar o branch "novo\_galho". Veremos a frente como criar novos *branches* através de linhas de comando.



## ATENÇÃO

Em caso de QUALQUER alteração do projeto, é necessário utilizar o comando `git pull` para atualizar a máquina com essas atualizações!

Após criar, esta frase aparece abaixo no novo *branch*

This branch is even with master.

# "Branch" (RAMIFICAÇÃO)

Para “pular entre os galhos”, é necessário o comando `git checkout` e escrever o nome do novo *branch* para onde queremos ir.

NESTE NOVO *BRANCH*, vamos criar um arquivo para testes, ele vai ser diferente, pois não importa o que façamos, o galho principal, permanece intacto.


Depois de adicionado este novo arquivo através do comando `git add`, vamos “commitar”.





...é esperado a seguinte saída depois disto.

```
$ git commit
[novo galho 89a1a4e] adicionando arquivo de teste na ramificação
secundária
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 arquivoDeTeste.txt
```

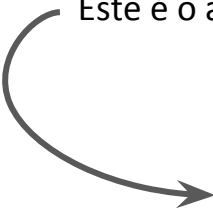
# "Branch" (RAMIFICAÇÃO)




Este é o arquivo com o branch "novo\_galho" perceba que o arquivo de teste está ali



 README.md	Initial commit	11 hours ago
 arquivoDeTeste.txt	adicionando arquivo de teste na ramificação secundária	12 minutes ago
 documentoNovo.txt	adicionando documento ao aloMundo	5 hours ago
 outroArquivo.txt	alterações novas feitas em: outroArquivo.txt	4 hours ago

Este é o arquivo com o branch **original** "master" perceba que o arquivo de teste **não existe!**



 README.md	Initial commit	12 hours ago
 documentoNovo.txt	adicionando documento ao aloMundo	5 hours ago
 outroArquivo.txt	alterações novas feitas em: outroArquivo.txt	4 hours ago

# "MERGE" (MESCLAR)

Quando se trabalha em dois ramos ("*branches*"), duas versões de arquivos diferentes, em dado algum momento essas segmentações precisam se unir. Há duas maneiras de resolvermos tal situação:

## 1) Merge **com** conflitos

Quando o conteúdo dos arquivos é alterado de alguma forma.

## 2) Merge **sem** conflitos

Quando não alteremos a mesma linha de um arquivo em branches diferentes, conseguimos realizar um merge sem ocasionar conflitos.



# "MERGE" (MESCLAR)

## 1) Merge **com** conflitos

Estando na ramificação secundária que aqui chamamos de **novo galho**, é necessário executar o seguinte comando: **git merge master**. Caso existam alterações entre os arquivos, é esperado o seguinte resultado:

```
$ git merge master
CONFLICT (add/add): Merge conflict in info.html
Auto-merging info.html
Automatic merge failed; fix conflicts and then commit the
result.
```

No site não seria diferente, uma vez com as duas ramificações conflituosas atualizadas, a mensagem de erro é eminente

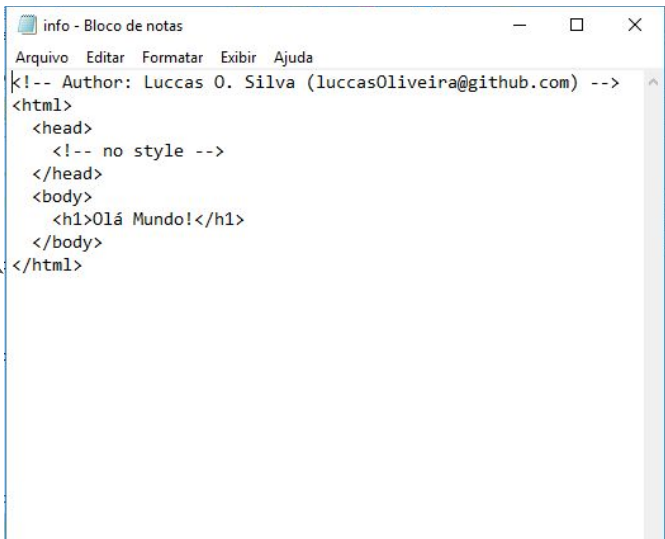


# "MERGE" (MESCLAR)

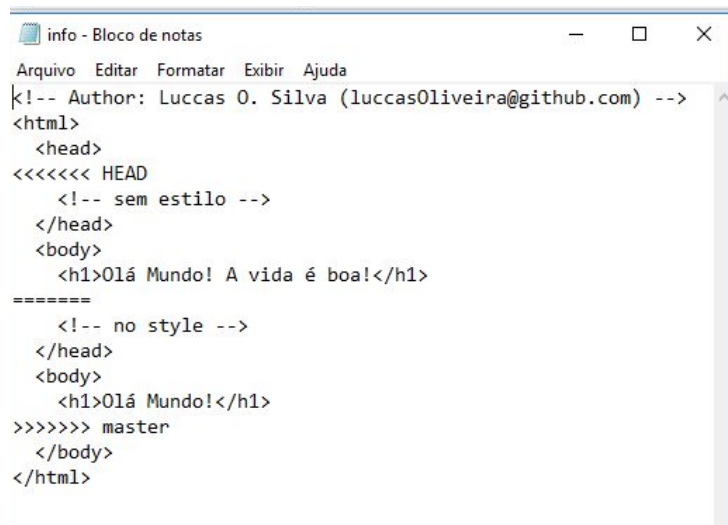
A causa do conflito: Foi criado um arquivo em formato html que torna impossível para o Git mesclar, diferente dos editores de texto.

Esta versão está no *branch* master...

...e esta versão, está no *branch* "novo\_galho"



```
info - Bloco de notas
Arquivo  Editar  Formatar  Exibir  Ajuda
|!-- Author: Luccas O. Silva (luccasOliveira@github.com) -->
<html>
  <head>
    <!-- no style -->
  </head>
  <body>
    <h1>Olá Mundo!</h1>
  </body>
</html>
```



```
info - Bloco de notas
Arquivo  Editar  Formatar  Exibir  Ajuda
|!-- Author: Luccas O. Silva (luccasOliveira@github.com) -->
<html>
  <head>
<<<<<<< HEAD
    <!-- sem estilo -->
  </head>
  <body>
    <h1>Olá Mundo! A vida é boa!</h1>
=====
    <!-- no style -->
  </head>
  <body>
    <h1>Olá Mundo!</h1>
>>>>>> master
  </body>
</html>
```

# "MERGE" (MESCLAR)

Você precisa resolver o conflito manualmente. Compare as duas versões e faça as alterações necessárias.

Assim que terminado é esperado a seguinte saída após o comando `git merge master`

```
$ git merge master  
Merge made by the 'recursive' strategy.
```

Depois, use o comando abaixo para "commitar" a mesclagem.

```
$ git commit -m "Merged master fixed  
conflict"
```

Não esqueça de  
"commitar" as  
alterações realizadas!



# ENTRE GALHOS E RAMOS

Na aba Insights, é possível ver o progresso de mesclagem e ramificação do projeto, incluindo picos de atividade, para isso, observe a figura a seguir:



DÚVIDAS?



