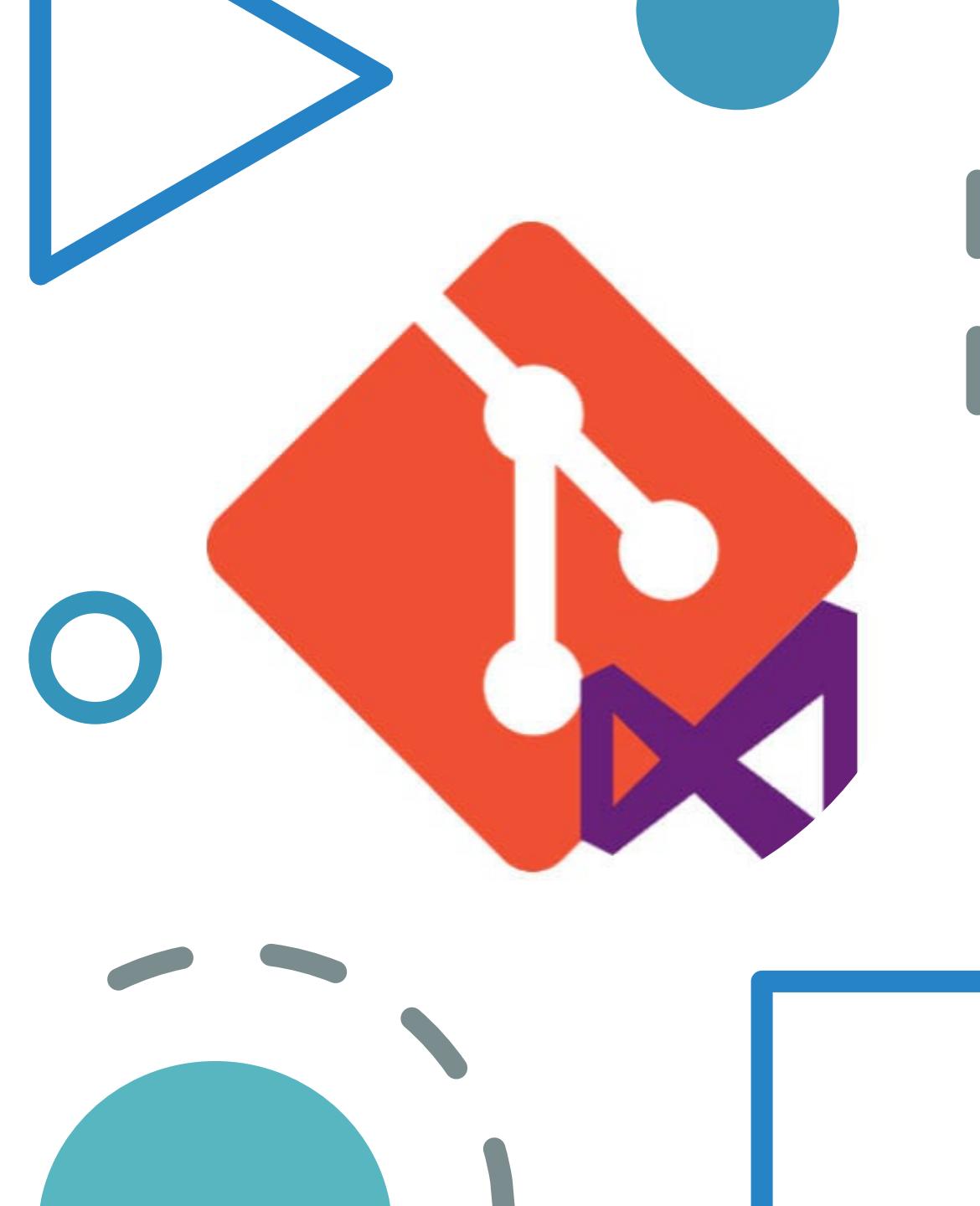


Fluxo de Trabalho do Git

- Mas atenção, se novas alterações acontecerem no seu ambiente de projeto, ou seja, nos casos em que algum arquivo rastreado (**TRACKED**) tenha sido alterado e que não passaram por um novo commit, ele passa para um estado chamado de **MODIFIED**
- O Git está te alertando que seu arquivo rastreado foi alterado desde seu último commit, você deverá ter uma estratégia para a realização de novos commits, assegurando um controle eficaz de seu projeto e não tirando Snapshots sem o menor critério



Fluxo de Trabalho do Git

- Quando nosso projeto controlado (Tracked) sofreu alterações (Modified),
precisamos repetir o processo, isto é, adicionar à Stage e depois realizar o Commit
- Fique atento, todos os passos são necessários, veja a seguir o resumo de tudo o que foi discutido até agora



Fluxo de Trabalho do Git

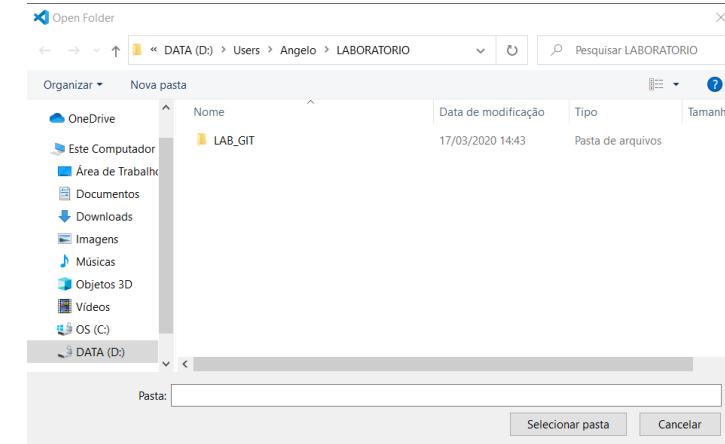
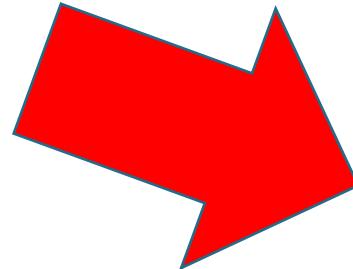
Parte II

Os comandos GIT ADD, GIT COMMIT & GIT STATUS

Os comandos GIT ADD, GIT COMMIT & GIT STATUS



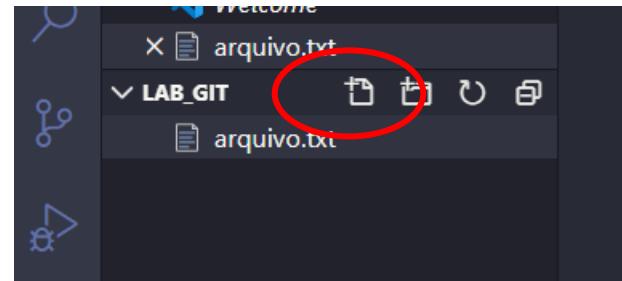
- Para testar este aprendizado, vamos agora fazer um pequeno laboratório
- Abra o Visual Studio Code
- Abra uma nova pasta, no meu caso, a pasta tem esse caminho



Os comandos GIT ADD, GIT COMMIT & GIT STATUS



- Vamos criar um arquivo, editá-lo e salvá-lo, veja os passos:



- Vamos iniciar o Git com o comando git init

```
accol@DESKTOP-TMBP1KD MINGW64 /d/Users/Angelo/LABORATORIO/LAB_GIT
$ git init
Initialized empty Git repository in D:/Users/Angelo/LABORATORIO/LAB_GIT/.git/
accol@DESKTOP-TMBP1KD MINGW64 /d/Users/Angelo/LABORATORIO/LAB_GIT (master)
$ 
```



Os comandos GIT ADD, GIT COMMIT & GIT STATUS

- Para saber o status do seu arquivo/projeto você deve usar o comando **git status** (*sempre use git status para acompanhar seu repositório*)
- Lembre-se o Git se recorda das configurações feitas anteriormente, logo, seu projeto possui configurações que foram herdadas, e isto, permanecerá até que resolva alterá-las com um novo **git config**

Observe que seu novo arquivo, no caso "arquivo.txt" esta na modalidade untracked

```
accol@DESKTOP-TMBP1KD MINGW64 /d/Users/Angelo/LABORATORIO/LAB_GIT (master)
$ git status
On branch master

No commits yet

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

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

accol@DESKTOP-TMBP1KD MINGW64 /d/Users/Angelo/LABORATORIO/LAB_GIT (master)
$ git config --global -l
user.name=accolombinifake
user.email=accolombinifake@hotmail.com
core.editor=notepad++
```



Os comandos GIT ADD, GIT COMMIT & GIT STATUS

- O próximo passo é colocar este arquivo na área de **Stage**, para que possamos a partir daí, **versioná-lo efetivamente**.
- Para isso vamos precisar do comando: **git add <nome_do_arquivo>**

Observe agora que
novo arquivo
passou para o
status de: No
commits yet

```
accol@DESKTOP-TMBP1KD MINGW64 /d/Users/Angelo/LABORATORIO/LAB_GIT (master)
$ git add arquivo.txt

accol@DESKTOP-TMBP1KD MINGW64 /d/Users/Angelo/LABORATORIO/LAB_GIT (master)
$ git status
On branch master
No commits yet

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

Os comandos GIT ADD, GIT COMMIT & GIT STATUS



- Feito isso, podemos continuar criando/modificando novos/antigos arquivos para depois adicioná-los na **Stage** e realizar um único **commit** mais tarde, ou podemos fazer um commit no nosso arquivo e dar início ao processo de rastreamento
- Para realizarmos o commit vamos precisar do comando **git commit -m** (a opção -m indica que podemos inserir uma mensagem em nosso commit para relatarmos o que foi feito até o momento). Esta mensagem deverá estar entre aspas simples ou dupla (depende de seu terminal) 'mensagem aqui' ou "mensagem aqui"
- Essa mensagem é importante para tornar o histórico dos arquivos mais amigável. Procure ser claro sucinto e objetivo nesta mensagem



Os comandos GIT ADD, GIT COMMIT & GIT STATUS



- Observe nosso exemplo, veja que após o commit o **git status** nos devolve uma mensagem alegando que não há nenhum documento rastreável sem commit até o momento, isso significa que seu projeto está com o controle de versão assegurado

Após o commit o git status nos indica que na nossa branch master não há documentos que precisem de commit

```
1 accol@DESKTOP-TMBP1KD MINGW64 /d/Users/Angelo/LABORATORIO/LAB_GIT (master)
$ git commit -m 'adição do arquivo.txt'
[master (root-commit) ca9e81e] adição do arquivo.txt
 1 file changed, 1 insertion(+)
 create mode 100644 arquivo.txt

accol@DESKTOP-TMBP1KD MINGW64 /d/Users/Angelo/LABORATORIO/LAB_GIT (master)
$ git status
On branch master
nothing to commit, working tree clean
```

Os comandos GIT ADD, GIT COMMIT & GIT STATUS



- Vamos observar o que acontece quando fazemos uma alteração em nosso arquivo
- Observe que nosso arquivo agora estará no **status Modified**, isso é, ainda não foi para a **Stage**

Observe que o Git reconheceu que houve alteração no documento e que o documento ainda não foi para a área de Stage

```
1 accol@DESKTOP-TMBP1KD MINGW64 /d/Users/Angelo/LABORATORIO/LAB_GIT (master)
$ git status
On branch 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:   arquivo.txt

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

Os comandos GIT ADD, GIT COMMIT & GIT STATUS



- Neste momento, a dica é, avalie se vale a pena fazer um commit de seu arquivo, não é recomendado que se faça commits a cada alteração
- A melhor dica aqui é avaliar o contexto, se o contexto demanda uma foto, demanda que registre esse fato, se essa for a situação, então faça um commit, caso não siga em frente atento ao controle do seu projeto, saiba que o Git não poderá recuperar nenhum trecho de seu projeto que não esteja no status de committed
- **Uma boa referência é deixar para realizar seus commits quando as alterações realizadas fizerem algum sentido em seu projeto**

Os comandos GIT ADD, GIT COMMIT & GIT STATUS



- Como estamos num laboratório vamos realizar mais um commit
- Desta vez vamos usar o comando `(.)` que dirá ao Git o seguinte, adicione à *Stage* todos os arquivos/documentos que sofreram alteração. Veja a seguir todos os passos:

Observe que o Git reconheceu todas as alterações que ocorreram no arquivo, inclusive uma delação de uma palavra

1

```
accol@DESKTOP-TMBP1KD MINGW64 /d/Users/Angelo/LABORATORIO/LAB_GIT (master)
$ git add .

accol@DESKTOP-TMBP1KD MINGW64 /d/Users/Angelo/LABORATORIO/LAB_GIT (master)
$ git commit -m 'Alteração na segunda linha do arquivo.txt'
[master f9e346e] Alteração na segunda linha do arquivo.txt
 1 file changed, 2 insertions(+), 1 deletion(-)

accol@DESKTOP-TMBP1KD MINGW64 /d/Users/Angelo/LABORATORIO/LAB_GIT (master)
$ git status
On branch master
nothing to commit, working tree clean
```

Tudo bem, mas como vejo as versões anteriores?

Git checkout parte III



Git Checkout

- Vamos entender como podemos visualizar o histórico dos arquivos criados, quem os criou e em que momento foram criados
- O comando responsável por isso é o **git log**, veja a seguir:

Observe o snapshot gerado para o commit. O Git exibe todos os commits que foram realizados, do mais recente para o mais antigo

```
accol@DESKTOP-TMBP1KD MINGW64 /d/Users/Angelo/LABORATORIO/LAB_GIT (master)
$ git log
commit 2aaaf4a0bf3f19d622a076c96e2f130c65609c974 (HEAD -> master)
Author: accolombinifake <accolombinifake@hotmail.com>
Date:   Tue Mar 17 16:02:05 2020 -0300
        Últimas alterações
commit f9e346ef5b1b834163daefb6a0ed8570e81b1c14
Author: accolombinifake <accolombinifake@hotmail.com>
Date:   Tue Mar 17 15:52:37 2020 -0300
        Alteração na segunda linha do arquivo.txt
commit ca9e81e1d217ac09c70525f90aa12a07ce6d85a7
Author: accolombinifake <accolombinifake@hotmail.com>
Date:   Tue Mar 17 15:25:00 2020 -0300
        adição do arquivo.txt
```

Git Checkout

- O comando `git log` possui uma série de variações que serão apresentadas com o tempo
- Caso seu log seja muito longo seu terminal ficará paralisado. Para sair você deve digitar a letra **q** (quit)
- O nome que se dá ao histórico é **Project History**
- Para visualizar qualquer arquivo de seu Project History, basta copiar seu hash code e usar o comando **git checkout <hash code>**, que o Git irá exibir seu documento no instante em que a foto foi tirada. Veja a seguir:



Git Checkout



- Texto atual:

```
arquivo.txt
1  Texto
2  Mais uma linha
3  Texto
4  Mais uma linha
5  Texto
6  Mais uma linha
```

- Texto no primeiro commit

Texto commit

Checkout

```
1 acc01@DESKTOP-TMPD1KD: MINGW64 /d/Users/Angelo/LAB_GIT (master)
$ git checkout ca9e81e1d217ac09c70525f90aa12a07ce6d85a7
Note: switching to 'ca9e81e1d217ac09c70525f90aa12a07ce6d85a7'.
```

You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may do so (now or later) by using -c with the switch command. Example:

```
git switch -c <new-branch-name>
```

Or undo this operation with:

```
git switch -
```

Turn off this advice by setting config variable advice.detachedHead to false

```
HEAD is now at ca9e81e adição do arquivo.txt
```



Git Checkout

- Se observar mais de perto, verá que o Git está lhe informando que ele está apontando agora para seu arquivo no estado em que ele se encontrava quando você tirou aquela foto
- Se você digitar o comando git log novamente poderá levar um susto, observe:

```
accol@DESKTOP-TMBP1KD MINGW64 /d/Users/Angelo/LABORATORIO/LAB_GIT ((ca9e81e...))  
$ git log  
commit ca9e81e1d217ac09c70525f90aa12a07ce6d85a7 (HEAD)  
Author: accolombinifake <accolombinifake@hotmail.com>  
Date:   Tue Mar 17 15:25:00 2020 -0300  
  
    adição do arquivo.txt
```

O Git esta apontando para esse head e não mais para a branch master



Git Checkout

- Para você voltar para o status atual ou dar uma verificada em outros commits realizado você deverá usar o comando **git checkout master** (olha a branch master aí)

```
acco1@DESKTOP-TMBP1KD MINGW64 /d/Users/Angelo/LABORATORIO/LAB_GIT ((ca9e81e...))  
$ git checkout master  
Previous HEAD position was ca9e81e adição do arquivo.txt  
Switched to branch 'master'  
  
acco1@DESKTOP-TMBP1KD MINGW64 /d/Users/Angelo/LABORATORIO/LAB_GIT (master)  
$ 
```

Agora o Git trouxe você novamente para sua branch master, você está de volta ao status atual!

1

Atenção você sempre deverá voltar para a branch master, ok

Introdução ao GitHub

Vamos agora elevar nosso controle para a nuvem

GitHub



Introdução ao GitHub

Why GitHub? ▾ Enterprise Explore Marketplace Pricing ▾

Search GitHub



Sign in Sign up

Built for developers

GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 40 million developers.

Para iniciar neste tópico vamos ao site <https://github.com/>

Individuals

Everything a developer needs to contribute to the future of software.

[View individual plans →](#)

Organizations

Essential management and security for teams of all sizes.

[View organization plans →](#)

Introdução ao GitHub



- O GitHub é uma plataforma remota de hospedagem de projetos
- É baseado no Git permitindo que continuemos a trabalhar normalmente como se estivéssemos localmente
- Há outras ferramentas com esse propósito como o BitBucket <https://bitbucket.org/> e outros ...
- **Mas o GitHub é o Melhor!!!**

Introdução ao GitHub



- Poder do GitHub → além de integrar todas as ferramentas do Git, incorpora outras ferramentas que potencializam ainda mais as ações de projeto em times
- Você também irá encontrar um gigantesca comunidade trabalhando com GitHub, e mais um número imenso de projetos compartilhados só esperando por você, em outras palavras, você nem sempre precisará partir do zero, terá a seu alcance muito do trabalho que já foi desenvolvido e que poderá lhe ser útil

Introdução ao GitHub



1

Atenção crie sua conta com suas credenciais de trabalho

Para trabalhar com o GitHub você antes de mais nada deverá criar uma conta, o processo é bem simples, basta seguir os passos:



Join GitHub

Create your account

Username *

Email address *

Password *

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more](#).

Email preferences

Send me occasional product updates, announcements, and offers.

Verify your account

Por favor resolva esse desafio para que possamos saber que você é uma pessoa de verdade.

Introdução ao GitHub

Na sequência seremos questionados a cerca de qual plano queremos?



No caso, vamos escolher o plano Unlimited public repositório for free



A light gray rectangular area containing a background pattern of black paw prints from a cat, arranged in a diagonal line from bottom-left to top-right.

Welcome to GitHub

You've taken your first step into a larger world, @fakebmarchete.

Completed
Set up a personal account

Step 2:
Choose your plan

Step 3:
Tailor your experier

Choose your personal plan

Unlimited public repositories for free.

Unlimited private repositories for \$7/month.

Don't worry, you can cancel or upgrade at any time.

Help me set up an organization next

Organizations are separate from personal accounts and are best suited for businesses who need to manage permissions for many employees.
[Learn more about organizations.](#)

Both plans include:

- Collaborative code rev
- Issue tracking
- Open source commun
- Unlimited public repos
- Join any organization

Introdução ao GitHub



- Por último apresenta-se umas questões, que poderemos deixar de lado e submeter a criação da nossa conta



What are you interested in?

e.g. tutorials, android, ruby, web-development, machine-learning, open-source

Submit

[skip this step](#)

Introdução ao GitHub



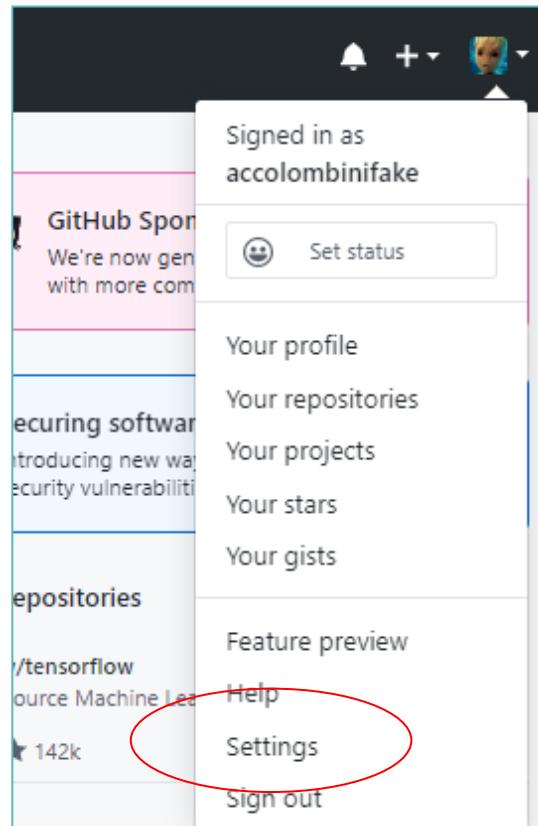
- Há muitos tutoriais que ensinam a trabalhar com o GitHub, observe que o primeiro está à sua disposição ao finalizar sua conta
- Há também um guia para você iniciar seu projeto
- Antes disso, vamos fazer algumas configurações básicas



Introdução ao GitHub



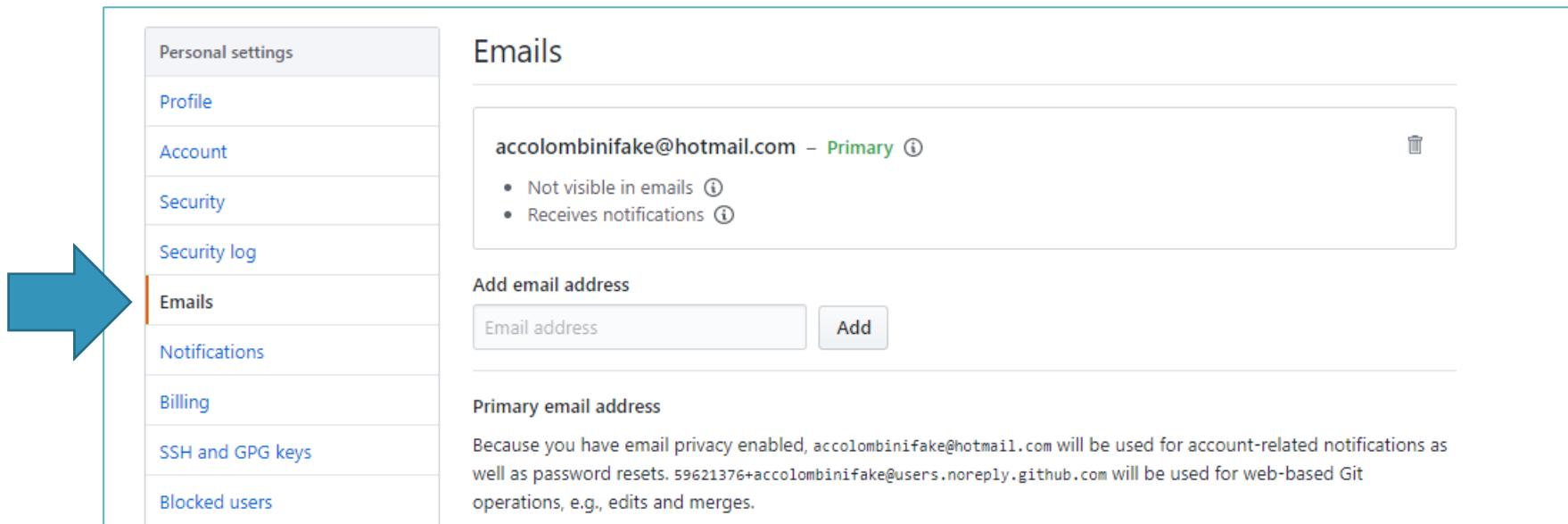
- Em profiles vamos em settings, veja como:



Introdução ao GitHub



- Neste ponto há muito o que preencher, mas o essencial é **confirmar seu e-mail**. Faça isso e estará pronto para trabalhar. **Atenção você precisa confirmar seu e-mail, ok!**



A screenshot of the GitHub Personal settings page, specifically the 'Emails' section. A large blue arrow points from the left towards the 'Emails' tab in the sidebar. The sidebar also lists 'Profile', 'Account', 'Security', 'Security log', 'Notifications', 'Billing', 'SSH and GPG keys', and 'Blocked users'. The 'Emails' tab is selected, showing a list of emails. The first email listed is 'accolombinifake@hotmail.com – Primary'. Below it, there's a list of notifications: 'Not visible in emails' and 'Receives notifications'. Below this list is a section titled 'Add email address' with a text input field labeled 'Email address' and a 'Add' button. At the bottom, there's a note about primary email addresses and privacy settings.

Personal settings

Profile

Account

Security

Security log

Emails

Notifications

Billing

SSH and GPG keys

Blocked users

Emails

accolombinifake@hotmail.com – Primary ⓘ

- Not visible in emails ⓘ
- Receives notifications ⓘ

Add email address

Email address

Add

Primary email address

Because you have email privacy enabled, accolombinifake@hotmail.com will be used for account-related notifications as well as password resets. 59621376+accolombinifake@users.noreply.github.com will be used for web-based Git operations, e.g., edits and merges.

GitHub → criando e entendendo repositórios

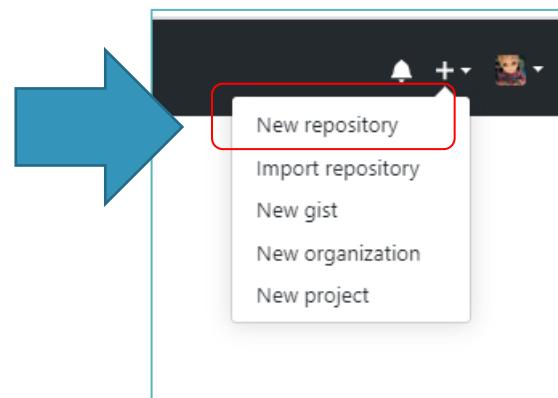
- Todo projeto em GitHub demanda a criação de um repositório
- Logo, você deverá criar tantos repositórios quantos projetos que estiver trabalhando
- Vamos começar com um, já é suficiente para nosso aprendizado
- Existem diversas maneiras de criarmos esses repositórios e sincronizar com nosso computador, nosso **ambiente Git, lembra dele!!!**



GitHub → criando e entendendo repositórios



- Criação de repositórios
 - Técnica simplificada → criar o repositório a partir de sua página no GitHub
 - Basta clicar na opção de + e em seguida em novo repositório



GitHub → criando e entendendo repositórios

- Criando um repositório com o nome inicial
- Este repositório será público, para privado é necessário uma conta especial. Hoje já é possível mesmo numa conta free.
- Demais opções serão exploradas nas próximas aulas

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner / Repository name 

Great repository names are short and memorable. Need inspiration? How about [“initial”](#)?

Description (optional)

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

 Private
You choose who can see and commit to this repository.

Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you don't need it.

Add .gitignore: None  Add a license: None  





[acolombinifake / inicial](#)

Unwatch 1 Star 0 Fork 0

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

Quick setup — if you've done this kind of thing before

[Set up in Desktop](#) or [HTTPS](#) [SSH](#) <https://github.com/acolombinifake/inicial.git>

Get started by creating a new file or uploading an existing file. We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# inicial" >> README.md  
git init  
git add README.md  
git commit -m "first commit"  
git remote add origin https://github.com/acolombinifake/inicial.git  
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/acolombinifake/inicial.git  
git push -u origin master
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

[Import code](#)

GitHub → criando e entendendo repositórios

Esta tela nos mostra como ligamos nosso diretório no GitHub com nosso diretório local

Introdução ao GitHub

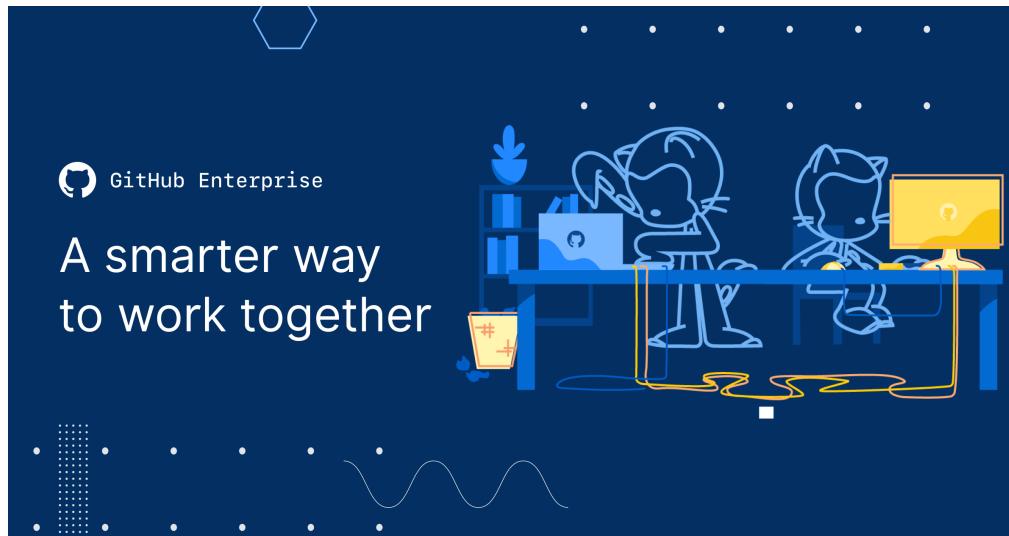
- Vamos fazer uma pausa e falar um pouco sobre o arquivo **Readme.md**
- Este arquivo é muito interessante para você especificar o por quê de seu projeto, você pode colocar instruções de instalação, normas que estão sendo empregadas em seu desenvolvimento, etc.
- Você tem a opção de criar o arquivo de **Readme** no momento em que está criando seu Repositório



Introdução ao GitHub



- Veja a tela a seguir:



Create a new repository

A repository contains all the files for your project, including the revision history.

Owner / Repository name ✓

Great repository names are short and memorable. Need inspiration? How about super-duper-giggle.

Description (optional)

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

Private You choose who can see and commit to this repository.

Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: Add a license: ⓘ

Introdução ao GitHub



- Para o GitHub todo arquivo com extensão **.md** possui um esquema de marcação específico criado pelo GitHub
- Para editar esse arquivo usamos diretamente a plataforma do GitHub, clicando sobre o arquivo

A screenshot of a GitHub repository page. At the top, there are statistics: 1 commit, 1 branch, 0 packages, 0 releases, and 1 contributor. Below this are buttons for Branch: master, New pull request, Create new file, Upload files, Find file, and Clone or download. The main area shows a commit history. The first commit, by user 'accolombinifake' with the message 'Initial commit', has a file named 'README.md' highlighted with a red box. Below it is another commit with the same file name. The word 'Teste' is visible in the commit message of the second commit. The entire screenshot is enclosed in a light blue border.

| Manage topics | | | | |
|--------------------------------|------------------|-----------------|-------------------------------------|---------------|
| 1 commit | 1 branch | 0 packages | 0 releases | 1 contributor |
| Branch: master | New pull request | Create new file | Upload files | Find file |
| accolombinifake Initial commit | | | Latest commit 1e7dc8e 3 minutes ago | |
| README.md | | | Initial commit | 3 minutes ago |
| README.md | | | | |
| Teste | | | | |

Introdução ao GitHub



- Em seguida clique no **ícone para edição** e pronto, é só digitar tudo o que considera relevante para uma primeira leitura a cerca o que se trata seu repositório

A screenshot of a GitHub commit page for a file named 'index.html'. The commit details are as follows:

- Author: accolombini fake
- Commit message: Initial commit
- Date: 1e7dc8e 15 seconds ago
- Contributors: 1 contributor
- File statistics: 1 lines (1 sloc), 7 Bytes
- Actions: Raw, Blame, History, Copy, Delete, Edit (highlighted with a red box)

The content of the file is displayed below the statistics:

```
Teste
```

Introdução ao GitHub



- Ao clicar em editar, o editor do GitHub é aberto e seu arquivo já recebe uma marcação gerada automaticamente com o título de seu repositório, não delete essa marcação, nas linhas seguintes insira suas informações. **Mas atenção este arquivo não se limita apenas a essa descrição**, falaremos mais nas próximas aulas

The screenshot shows a GitHub repository page for 'accolombinifake / Teste'. The 'Code' tab is selected. A modal window is open over the repository header, showing the file 'Teste / README.md'. The modal has tabs for 'Edit file' and 'Preview changes'. The preview area contains the following text:

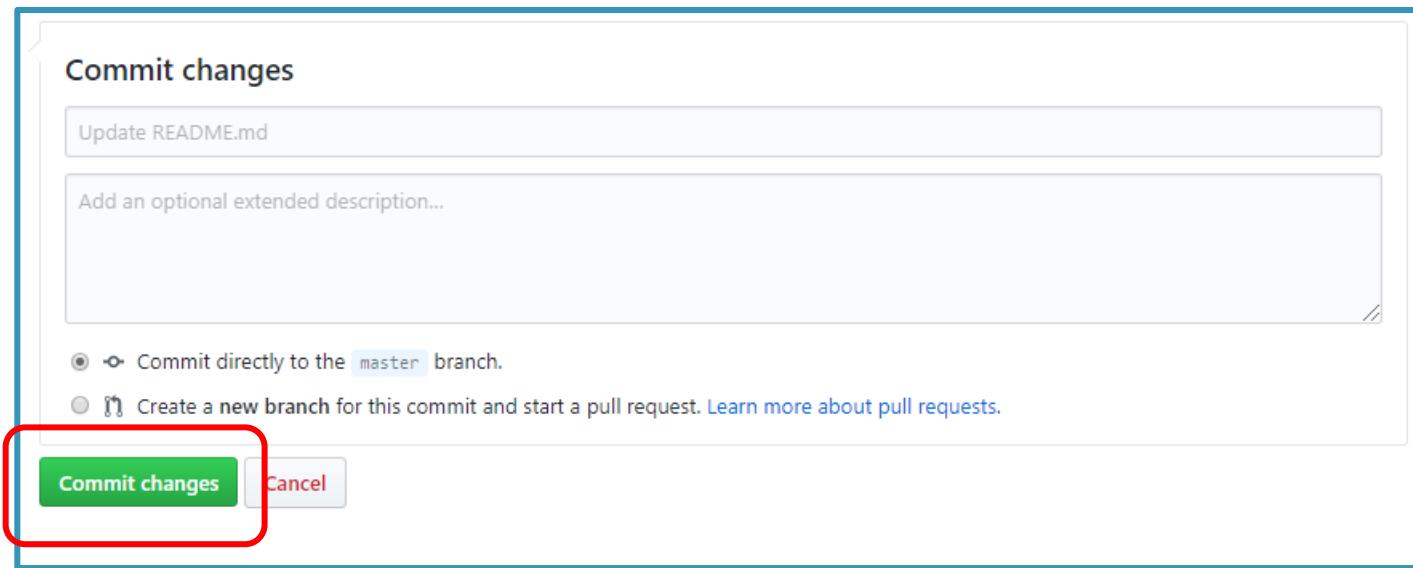
```
1 # Teste
2
3 Este projeto .....
```

A red rectangular box highlights the first line of the preview, '# Teste', which is the automatically generated title from the repository name.

Introdução ao GitHub



- Quando concluir sua digitação você deverá fazer o commit do seu trabalho, sim é o mesmo processo que viu com o Git. O GitHub já vai começar a controlar seu projeto, falaremos mais sobre isso, aguarde



Introdução ao GitHub

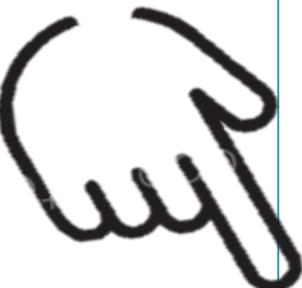
- Vamos agora conhecer outro arquivo importante, o arquivo **.gitignore**, lembra que comentamos que em algum momento podemos optar por não controlar algum arquivo que usamos em nosso projeto, pois é chegou a hora.
- Esse arquivo pode ser criado diretamente no nosso diretório local, ou a partir da criação de um novo repositório na plataforma do GitHub, para exemplificar, faremos isso na plataforma, acompanhe



Introdução ao GitHub



- Quando criamos um novo repositório temos a opção além de criar o arquivo de README.md, criar o gitignore, observe



A screenshot of the GitHub 'Create repository' form. The 'Repository name' field contains 'gitignore'. Below it, there's a text input for 'Description (optional)'. Under 'Visibility', the 'Public' option is selected. In the 'Initialize this repository with a README' section, the checkbox is checked and the note 'This will let you immediately clone the repository to your computer.' is visible. A dropdown menu for 'Add .gitignore' is open, showing 'None' as the current selection. A red box highlights this dropdown. At the bottom, there's a 'Create repository' button.

Owner Repository name *

accolombinifake / gitignore ✓

Great repository names are short and memorable. Need inspiration? How about shiny-waddle?

Description (optional)

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

Private You choose who can see and commit to this repository.

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.

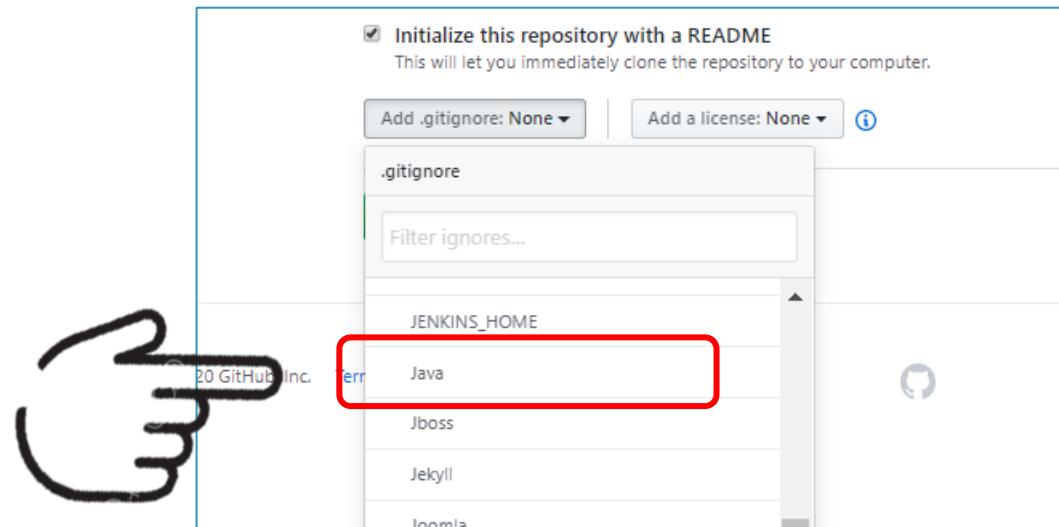
Add .gitignore: None ▾ Add a license: None ⓘ

Create repository

Introdução ao GitHub



- Ao clicar na opção Add .gitignore o GitHub trás uma série de opções pré-configuradas que podem apoiá-lo nessa tarefa (trata-se de arquivos comuns de serem ignorados), mas sim é claro, você poderá incluir as suas configurações
- Como exemplo, vamos admitir que estamos num projeto Java



Introdução ao GitHub



- Clicando sobre o nome do arquivo `.gitignore` o editor se abre com uma série de opções previamente estabelecidas pelo GitHub, pense nisso como um suporte a seu trabalho, cabe a você analisar se concorda ou não, acrescentando ou tirando arquivos



Branch: master [gitignore](#) / `.gitignore`

acolombinifake Initial commit
1 contributor

23 lines (18 sloc) | 278 Bytes

```
1 # Compiled class file
2 *.class
3
4 # Log file
5 *.log
6
7 # BlueJ files
8 *.ctxt
9
10 # Mobile Tools for Java (J2ME)
11 .mtj.tmp/
12
13 # Package Files #
14 *.jar
15 *.war
16 *.nar
17 *.ear
18 *.zip
19 *.tar.gz
20 *.rar
21
22 # virtual machine crash logs, see http://www.java.com/en/download/help/error_hotspot.xml
23 hs_err_pid*
```



Todos esses arquivos serão automaticamente ignorados pelo GitHub sempre que um sincronismo for realizado. Portanto, tenha sempre um arquivo `.ignore` em seu projeto



Introdução ao GitHub

Vamos praticar - Laboratório

Sincronizando seu projeto



- Dando início ao nosso laboratório, vamos a partir da plataforma GitHub criar um novo repositório
- Vamos chamar esse repositório de `projetolab`, inicializar o `readme.md` e adicionar o `.gitignore` como se fosse um projeto Java, você já sabe fazer isso, mãos à obra. Teremos algo como:

The screenshot shows a GitHub repository page for the user `acolombinifake` named `projetolab`. The repository has 1 commit, 1 branch, 0 packages, 0 releases, and 1 contribution. The `master` branch is selected. The commit history shows three files: `.gitignore`, `README.md`, and another `README.md`. The repository description is: "Projeto para ações práticas em laboratório, sincronizando Git com GitHub." Below the repository details, there is a section titled "projetolab" with the same description.

Sincronizando seu projeto



- Precisamos agora, de alguma forma, descarregar esse projeto em nossa máquina, mas como?
- A primeira coisa que nos vem à mente é ...

A screenshot of a GitHub repository page for a user named 'accolombinifake'. The page shows several files: '.gitignore' (Initial commit), 'README.md' (Initial commit), and another 'README.md' file. At the top right, there's a 'Clone or download' button with a dropdown menu. The menu includes 'Clone with HTTPS' (selected), 'Use SSH', and a 'Download ZIP' button, which is highlighted with a red rectangle. A large black hand cursor points at the 'Download ZIP' button. A blue callout bubble at the bottom left of the image contains the text: 'Resista a tentação e não faça isso, você perderá as ferramentas de gestão e sincronismo'.

Branch: master New pull request

Create new file Upload files Find file Clone or download

Clone with HTTPS Use SSH
Use Git or checkout with SVN using the web URL.
<https://github.com/accolombinifake/proje>

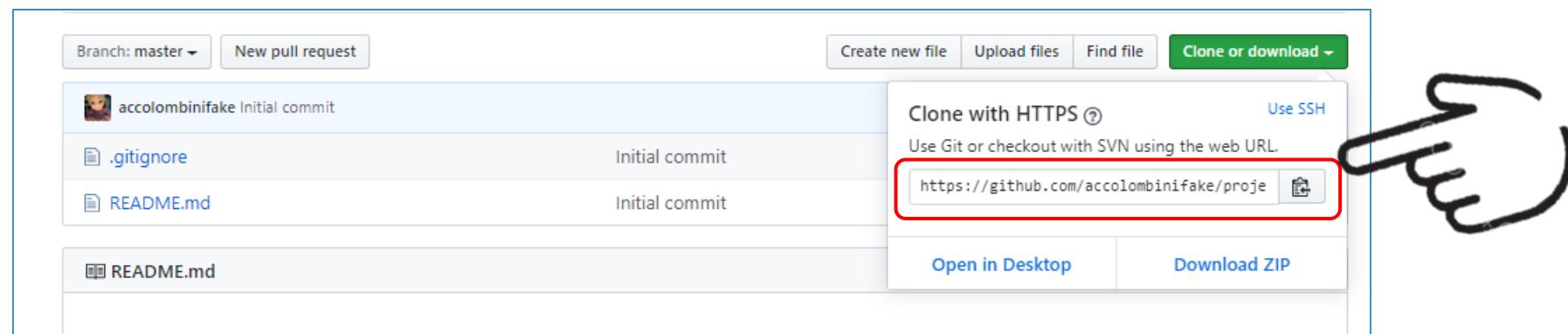
Open in Desktop Download ZIP

Resista a tentação e não faça isso,
você perderá as ferramentas de gestão
e sincronismo

Sincronizando seu projeto



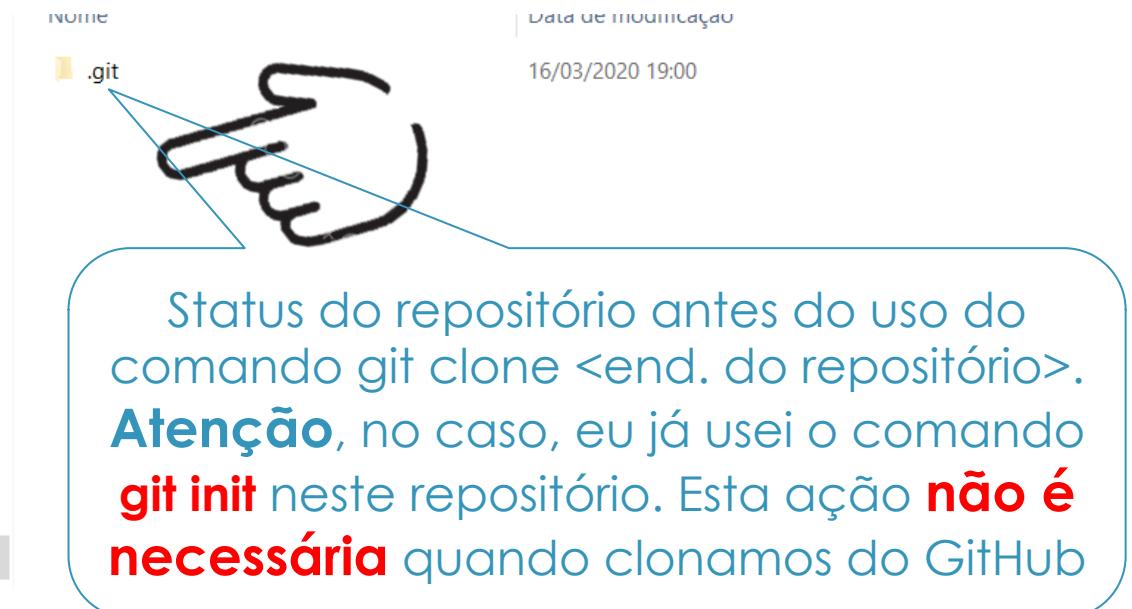
- Para garantir o pleno uso dos recursos e ferramentas do Git e GitHub, você deverá copiar o endereço do seu repositório no GitHub



Sincronizando seu projeto



- Uma vez copiado, volte para seu Visual Studio Code, abra o terminal, vá para a pasta criada anteriormente, no caso, criamos a pasta GIT
- Uma vez na pasta criada usamos o comando **git clone <endereço do repositório no GitHub>**
- Com essa ação, todo repositório criado no GitHub será baixado para sua máquina



Status do repositório antes do uso do comando `git clone <end. do repositório>`.
Atenção, no caso, eu já usei o comando `git init` neste repositório. Esta ação **não é necessária** quando clonamos do GitHub

Sincronizando seu projeto



- Posicionando no diretório destino e usando o comando
git clone <endereço do repositório do GitHub>

```
acco1@DESKTOP-TMBP1KD MINGW64 /  
$ cd D:/Users/Angelo  
  
acco1@DESKTOP-TMBP1KD MINGW64 /d/Users/Angelo  
$ cd GIT
```

1

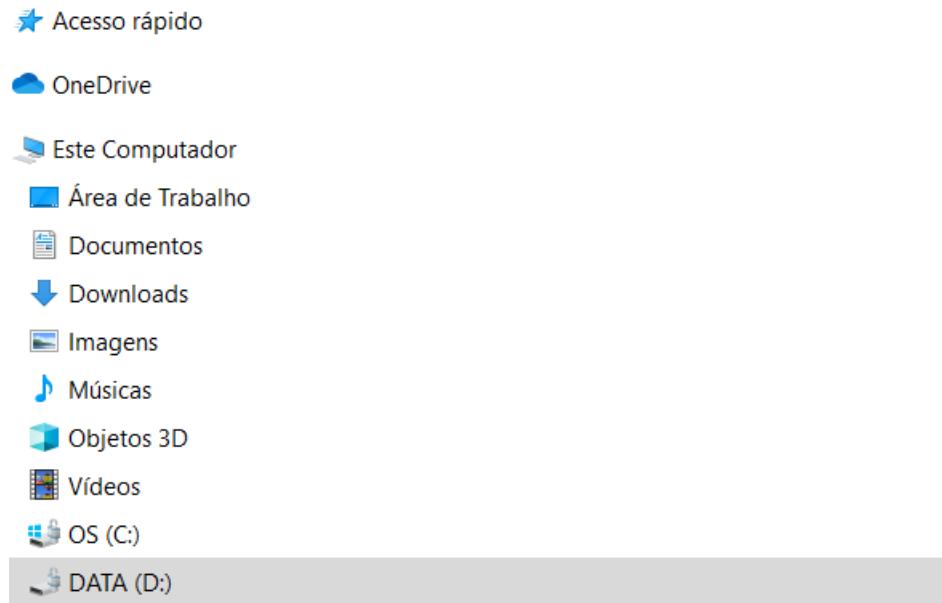
```
acco1@DESKTOP-TMBP1KD MINGW64 /d/Users/Angelo/GIT (master)  
$ git clone https://github.com/acco1ombinifake/projetolab.git  
Cloning into 'projetolab'...  
remote: Enumerating objects: 4, done.  
remote: Counting objects: 100% (4/4), done.  
remote: Compressing objects: 100% (4/4), done.  
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0  
Unpacking objects: 100% (4/4), 917 bytes | 2.00 KiB/s, done.
```

Tudo certo,
podemos
prosseguir

Sincronizando seu projeto



- Diretório após o uso do comando **git clone <end do repositório GitHub>**



Sincronizando seu projeto



- Abrindo o diretório local (pasta GIT) poderemos visualizar todos os arquivos criados no contexto do GitHub, veja ...

