

Assim que terminar de instalar e rodar a primeira vez, o git vai ter configurações bem padrão.

Vamos começar configurando o GIT na sua máquina.

Para isso, vamos definir seu nome, um e-mail e um editor de texto de sua preferência.

Vamos usar 3 comandos:

```
git config --global user.name "nome completo"
git config --global user.email email@provedor.com
git config --global core.editor notepad
```

No caso do editor, estou assumindo um usuário de Windows, e que ele prefira o notepad. Coloque o nome do seu editor preferido. Se não escolher nenhum, por padrão, o GIT usa o VIM na própria linha de comando.

Após configurar, podemos começar a mexer. No início, a tela vai estar aberta na raiz do Mingw (no Windows) ou na pasta do usuário.

Caso esteja no Windows, recomendo ir para a pasta do usuário digitando:

```
cd ~ <enter>
```

caso não esteja na pasta do usuário, mesmo estando no Linux, sugiro que vá para essa pasta.

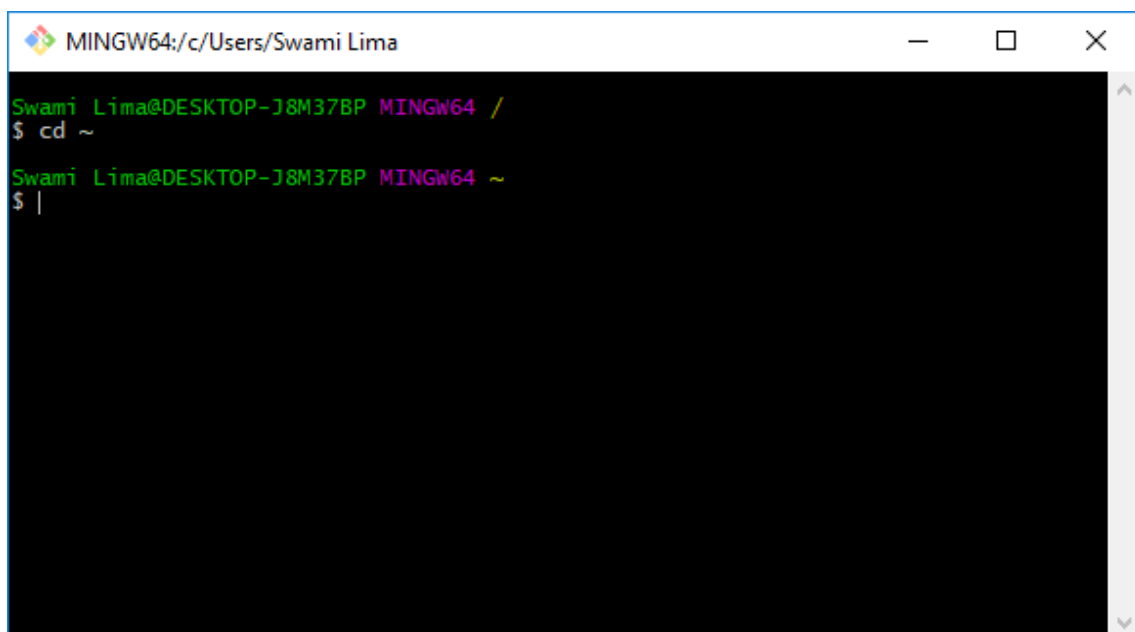
A screenshot of a terminal window titled "MINGW64; c:/Users/Swami Lima". The window has standard Windows window controls (minimize, maximize, close). The terminal content shows a prompt "Swami Lima@DESKTOP-J8M37BP MINGW64 /" followed by the command "\$ cd ~". The next line shows the prompt "Swami Lima@DESKTOP-J8M37BP MINGW64 ~" and a cursor on a new line "\$ |".

Figura 1: O GIT na linha de comando e a pasta "user".

Isso vai fazer com que o sistema se direcione para a pasta do Usuário.

Sempre que quiser listar o conteúdo da pasta, use o comando ls (LS).

Agora, para clonar o repositório, vamos usar um repositório de exemplo. O repositório de exemplo está no endereço <https://github.com/swamilima/ComoUsarGIT> e vou usá-lo como exemplo.

Para clonar, você deve dar um endereço com o final “.git” para o git. Dessa forma, ele vai copiar todos os arquivos para uma pasta com o mesmo nome do projeto exatamente na pasta em que estiver na linha de comando.

No meu desktop, eu criei uma pasta chamada “ComoUsar” e é dentro dela que vou clonar o meu GIT de exemplo.

Na página do projeto no GITHUB (ou em qualquer outro gerenciador de projetos que use o GIT) sempre vai haver um local onde se possa pegar o endereço para clonagem. No GITHUB, é bem fácil de ver, porque ele é verde e o botão é grande.

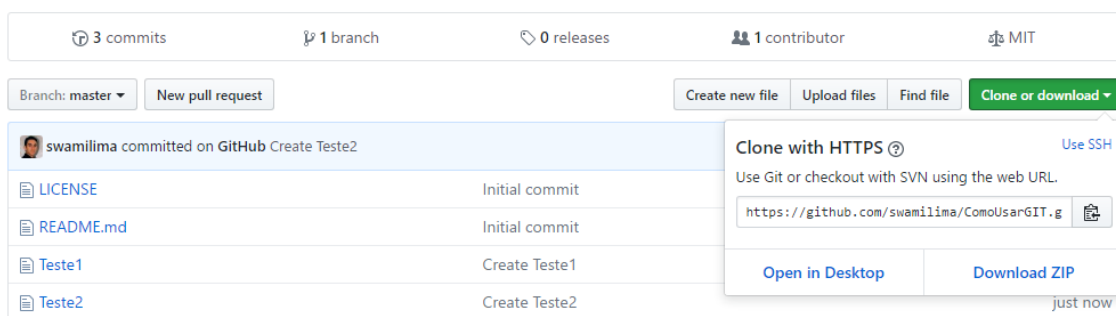


Figura 2: A página do projeto exemplo, no GITHUB.

Então, basta copiar esse endereço. No nosso caso, o endereço é:

<https://github.com/swamilima/ComoUsarGIT.git>

Veja que nosso projeto tem dois arquivos testes, o README do projeto e a licença.

Vamos clonar agora o projeto.

Usando seu gerenciador de arquivos (Windows Explorer, por exemplo) crie uma pasta em um local de sua escolha (no Desktop, nos documentos, dentro de uma pasta da disciplina, onde quiser). No meu caso, vou criá-la no Desktop. Como disse, vou dar o nome de “ComoUsar”. Note que não usei espaços, nem nessa pasta, nem nas pastas do GIT do projeto de CG. A linha de comando não se comporta bem com espaços, então use *underline* (_) ou hífen (-), ou apenas capitalizar (colocar letras maiúsculas quando mudar a palavra).

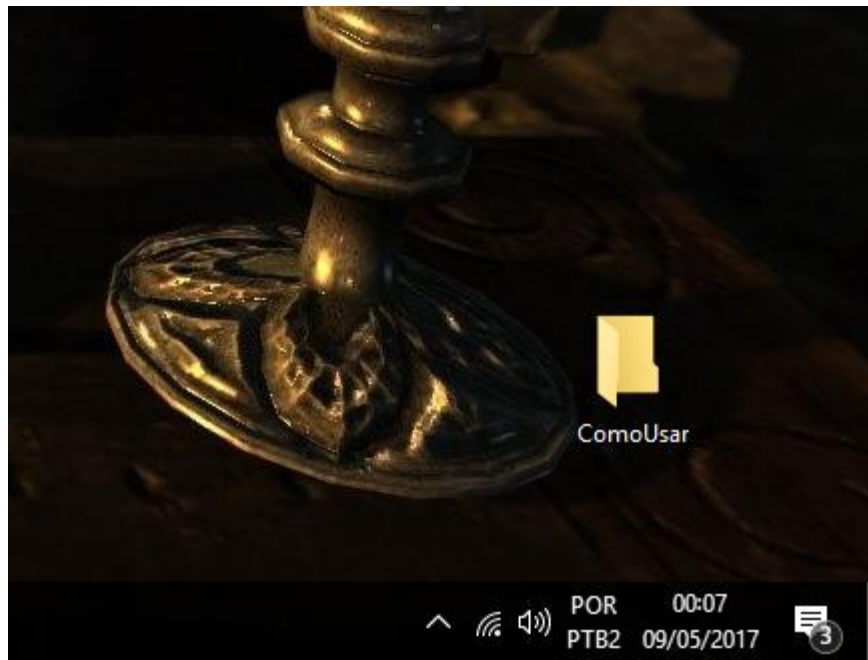


Figura 3: A pasta criada na minha área de trabalho (Desktop).

Agora, vamos navegar até essa pasta, usando a linha de comando.

Para isso, vamos usar o comando “cd”, seguido do nome da pasta.

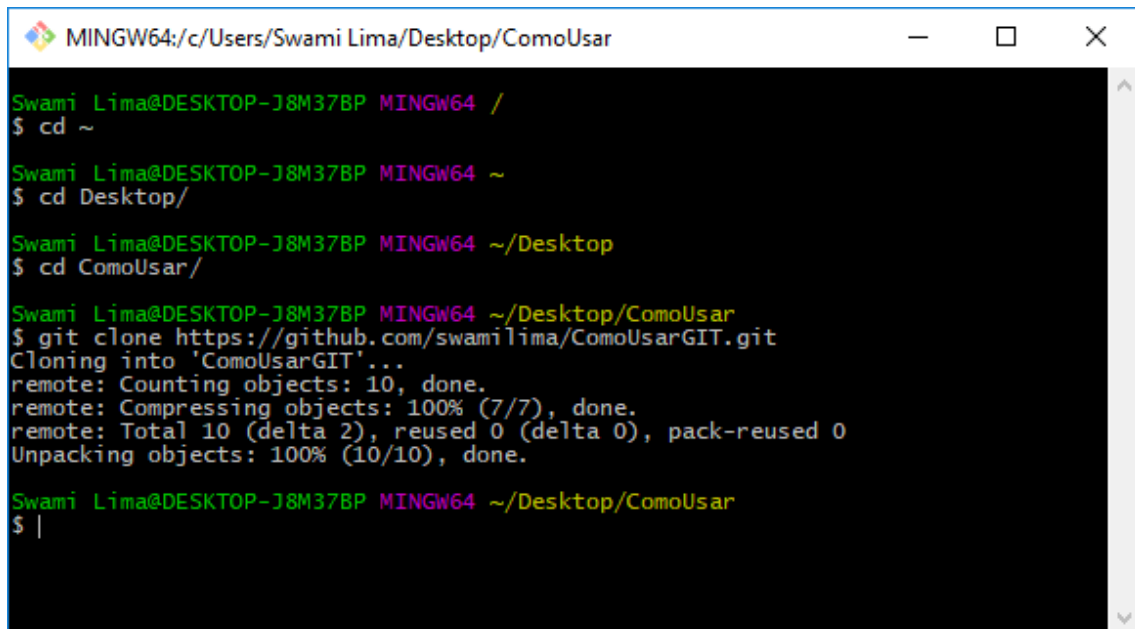
```
MINGW64:/c/Users/Swami Lima/Desktop/ComoUsar
Swami Lima@DESKTOP-J8M37BP MINGW64 /
$ cd ~
Swami Lima@DESKTOP-J8M37BP MINGW64 ~
$ cd Desktop/
Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop
$ cd ComoUsar/
Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop/ComoUsar
$ |
```

Figura 4: Como ir para a pasta desejada. Note que na imagem, a pasta atual é a "ComoUsar", exatamente a pasta onde vamos clonar nosso projeto.

Estando na pasta que queremos clonar, basta usar o comando “clone”, seguido do endereço onde queremos clonar. Lembre que todos os comandos começam com “git”, que é a chamada do programa.

Vamos usar o código:

```
git clone https://github.com/swamilima/ComoUsarGIT.git
```

A screenshot of a terminal window titled "MINGW64; c:/Users/Swami Lima/Desktop/ComoUsar". The terminal shows the following commands and output:

```
Swami Lima@DESKTOP-J8M37BP MINGW64 /  
$ cd ~  
  
Swami Lima@DESKTOP-J8M37BP MINGW64 ~  
$ cd Desktop/  
  
Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop  
$ cd ComoUsar/  
  
Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop/ComoUsar  
$ git clone https://github.com/swamilima/ComoUsarGIT.git  
Cloning into 'ComoUsarGIT'...  
remote: Counting objects: 10, done.  
remote: Compressing objects: 100% (7/7), done.  
remote: Total 10 (delta 2), reused 0 (delta 0), pack-reused 0  
Unpacking objects: 100% (10/10), done.  
  
Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop/ComoUsar  
$ |
```

Figura 5: Ao usar o comando "Clone", o git cria uma pasta com o nome do projeto na pasta atual, e copia todos os projetos para dentro dela.

Se eu for na minha pasta “ComoUsar” agora, ela contém uma pasta com o nome “ComoUsarGIT”, e dentro todos os arquivos.

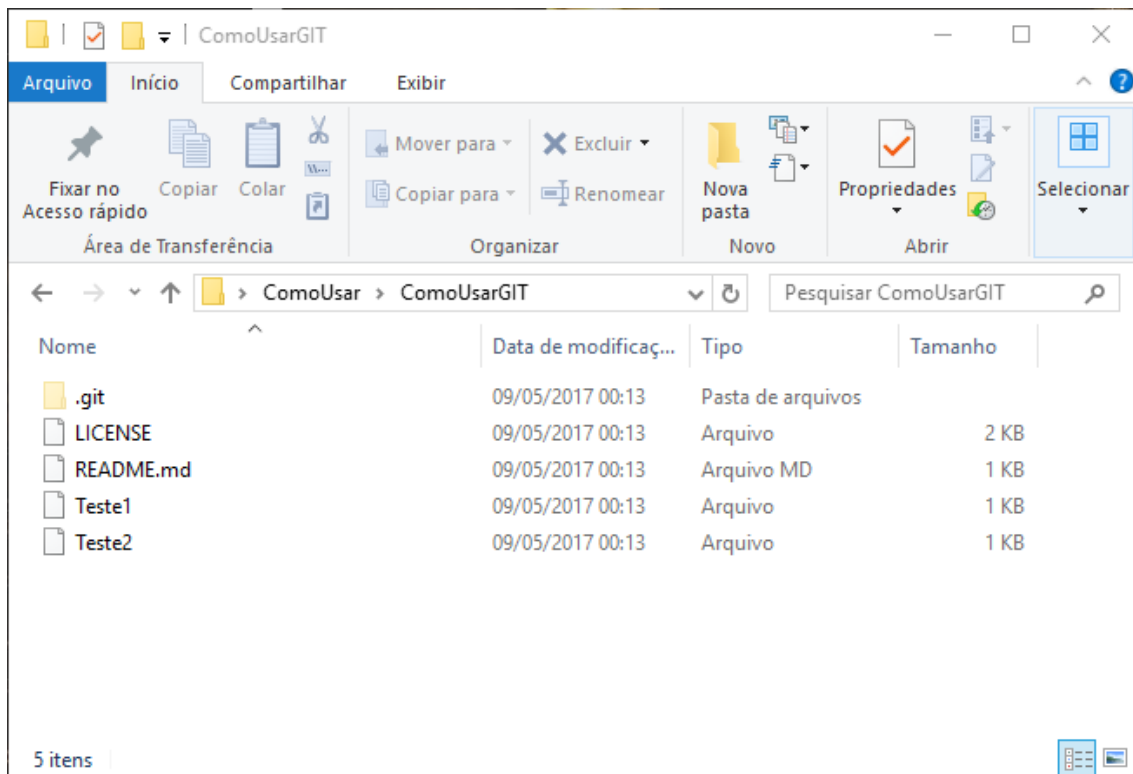


Figura 6: Minha pasta de trabalho. Esta pasta é uma cópia do projeto.

Para usar o GIT DENTRO da pasta do projeto, agora tenho que entrar na pasta de trabalho usando a linha de comando do GIT, com o comando “cd”:

```

MINGW64:/c:/Users/Swami Lima/Desktop/ComoUsar/ComoUsarGIT
Swami Lima@DESKTOP-J8M37BP MINGW64 /
$ cd ~

Swami Lima@DESKTOP-J8M37BP MINGW64 ~
$ cd Desktop/

Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop
$ cd ComoUsar/

Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop/ComoUsar
$ git clone https://github.com/swamilima/ComoUsarGIT.git
Cloning into 'ComoUsarGIT'...
remote: Counting objects: 10, done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 10 (delta 2), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (10/10), done.

Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop/ComoUsar
$ cd ComoUsarGIT/

Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop/ComoUsar/ComoUsarGIT (master)
$ |

```

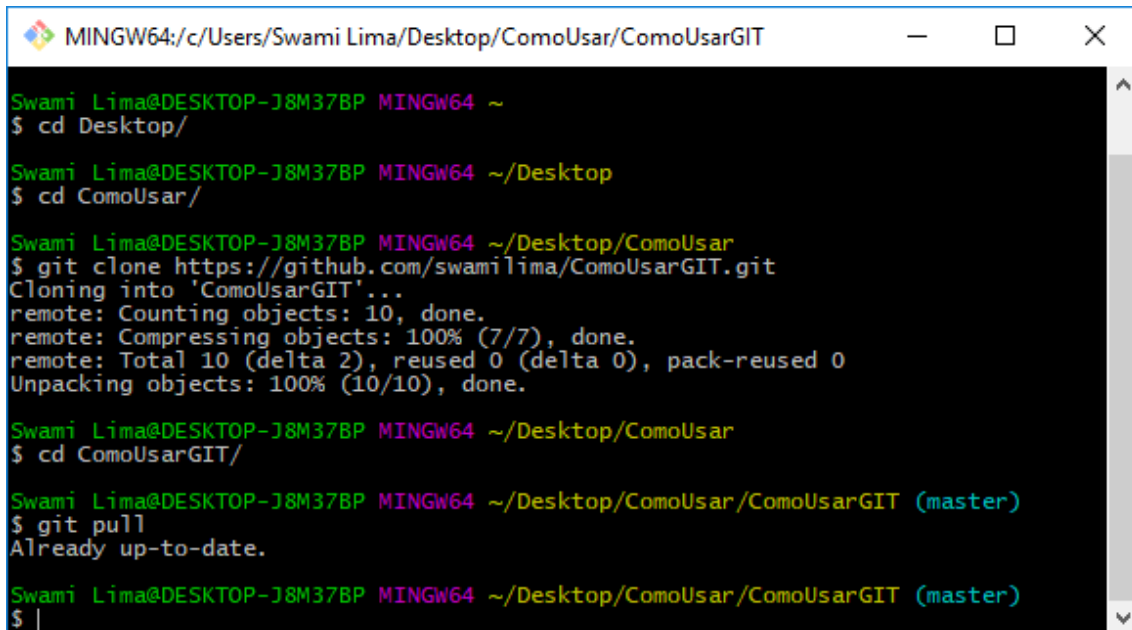
Figura 7: Usando o comando "cd" para entra na pasta de trabalho.

Note que agora, no final da linha existe um “master” em azul. Ele é UM dos indicativos que você está na pasta de trabalho. Ali pode estar outra coisa, mas não vamos falar sobre isso. Assuma apenas que se estiver no “master”, significa que está na pasta de trabalho.

Antes de começar a trabalhar na pasta, é interessante que se cheque se ela está atualizada. Para isso, use o comando

```
git pull
```

que é o comando para “puxar” o que está mais novo do projeto para sua pasta. Se usar o comando depois de ter alterado arquivos, ele vai dar um aviso dos arquivos que tem o mesmo nome, mas conteúdo diferente. Se sua pasta estiver atualizada, ele vai lhe informar com a mensagem abaixo.

A screenshot of a terminal window titled 'MINGW64:/c/Users/Swami Lima/Desktop/ComoUsar/ComoUsarGIT'. The terminal shows a series of commands and their outputs. The user starts by navigating to the desktop and then to the 'ComoUsar' directory. They then clone a repository from GitHub. After cloning, they navigate to the newly created 'ComoUsarGIT' subdirectory. Finally, they run 'git pull', which returns the message 'Already up-to-date.', indicating the local repository is synchronized with the remote one.

```
MINGW64:/c/Users/Swami Lima/Desktop/ComoUsar/ComoUsarGIT

Swami Lima@DESKTOP-J8M37BP MINGW64 ~
$ cd Desktop/

Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop
$ cd ComoUsar/

Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop/ComoUsar
$ git clone https://github.com/swamilima/ComoUsarGIT.git
Cloning into 'ComoUsarGIT'...
remote: Counting objects: 10, done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 10 (delta 2), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (10/10), done.

Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop/ComoUsar
$ cd ComoUsarGIT/

Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop/ComoUsar/ComoUsarGIT (master)
$ git pull
Already up-to-date.

Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop/ComoUsar/ComoUsarGIT (master)
$ |
```

Figura 8: O GIT informando que sua pasta está atualizada (up-to-date).

Agora, vou alterar o arquivo “Teste2”. Vou abrir e editar no notepad, esse arquivo, mas não vou postar imagem. Apenas estou alterando ele para poder “subir”. No nosso projeto, esses arquivos poderão ser códigos de JAVA, de C++ (copiados do LazyFoo) ou até mesmo os arquivos dos tutoriais, em DOCX ou PDF.

Após alterar o arquivo, estando ainda na pasta de trabalho vamos *adicionar* o arquivo alterado para que ele seja submetido. Podemos adicionar mais de um arquivo, antes de submeter, ou até mesmo submeter pastas inteiras, quando necessário (caso vá subir um projeto do NetBeans, por exemplo). Para isso, vamos usar o comando *add*, seguido do nome do(s) arquivo(s) ou pastas. No nosso exemplo, vamos adicionar apenas um arquivo.

```
MINGW64:/c/Users/Swami Lima/Desktop/ComoUsar/ComoUsarGIT

Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop
$ cd ComoUsar/

Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop/ComoUsar
$ git clone https://github.com/swamilima/ComoUsarGIT.git
Cloning into 'ComoUsarGIT'...
remote: Counting objects: 10, done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 10 (delta 2), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (10/10), done.

Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop/ComoUsar
$ cd ComoUsarGIT/

Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop/ComoUsar/ComoUsarGIT (master)
$ git pull
Already up-to-date.

Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop/ComoUsar/ComoUsarGIT (master)
$ git add Teste2

Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop/ComoUsar/ComoUsarGIT (master)
$
```

Figura 9: o uso do comando add e como ele não retorna nada.

Caso precisemos, podemos checar os arquivos que estão listados (adicionados) para serem submetidos. Para isso, usamos o comando *status*.

```
MINGW64:/c/Users/Swami Lima/Desktop/ComoUsar/ComoUsarGIT

Unpacking objects: 100% (10/10), done.

Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop/ComoUsar
$ cd ComoUsarGIT/

Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop/ComoUsar/ComoUsarGIT (master)
$ git pull
Already up-to-date.

Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop/ComoUsar/ComoUsarGIT (master)
$ git add Teste2

Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop/ComoUsar/ComoUsarGIT (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   Teste2

Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop/ComoUsar/ComoUsarGIT (master)
$
```

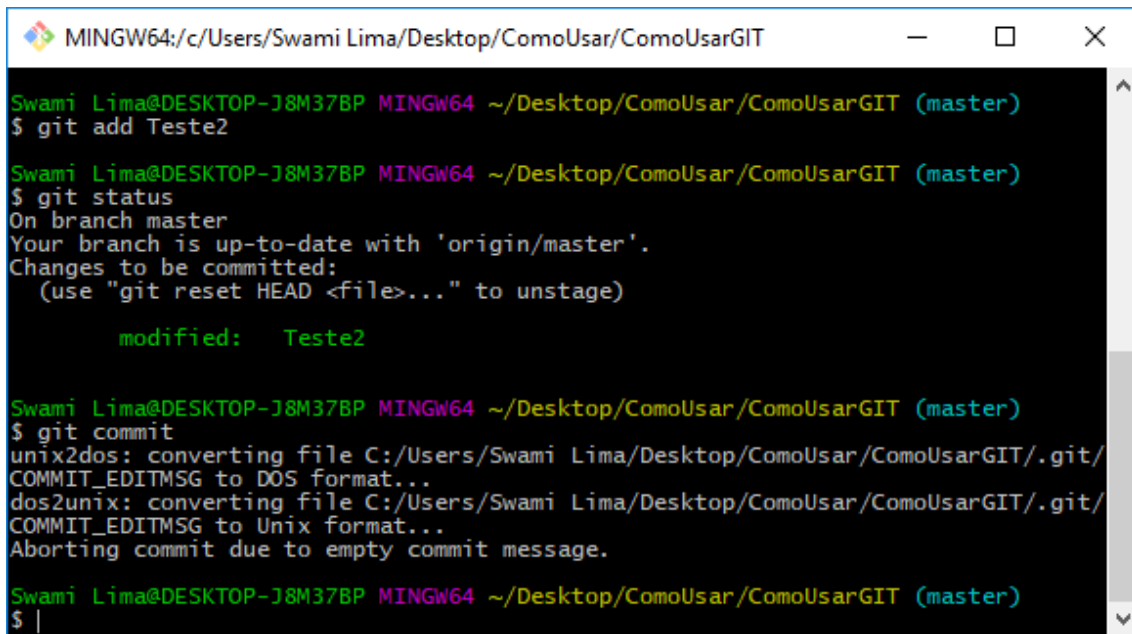
Figura 10: O comando Status.

Note que ele lista em verde, e logo nas linhas acima, tem escrito: *Changes ti be committed*:

O *Commit* citado é o submeter: é quando efetivamente avisamos o que vai ou não subir para o servidor como um arquivo modificado válido.

Para que um arquivo que foi *add* possa ser sinalizado como pronto para ir para o servidor, devemos *dar commit* no arquivo, usando o comando com esse nome.

Ao digitar o comando *commit*, uma janela do notepad deve abrir. Ela contém várias linhas com o símbolo #, que significa um comentário. Caso feche essa janela sem escrever nada, o Commit é cancelado.



```
MINGW64:/c/Users/Swami Lima/Desktop/ComoUsar/ComoUsarGIT
Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop/ComoUsar/ComoUsarGIT (master)
$ git add Teste2

Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop/ComoUsar/ComoUsarGIT (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   Teste2

Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop/ComoUsar/ComoUsarGIT (master)
$ git commit
unix2dos: converting file C:/Users/Swami Lima/Desktop/ComoUsar/ComoUsarGIT/.git/
COMMIT_EDITMSG to DOS format...
dos2unix: converting file C:/Users/Swami Lima/Desktop/ComoUsar/ComoUsarGIT/.git/
COMMIT_EDITMSG to Unix format...
Aborting commit due to empty commit message.

Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop/ComoUsar/ComoUsarGIT (master)
$ |
```

Figura 11: ao fechar a janela do notepad (ou qualquer outro editor de texto configurado), o GIT lhe informa que a mensagem foi vazia, e que o COMMIT foi cancelado.

Quando se alterar o arquivo de texto, tenha em mente o seguinte: a primeira linha será mostrada na página do projeto. Dessa forma, use a primeira linha como um “título” para a alteração que fez. Salve, após escrever o texto. Após escrito e salvo, feche a janela, para completar o *commit*.

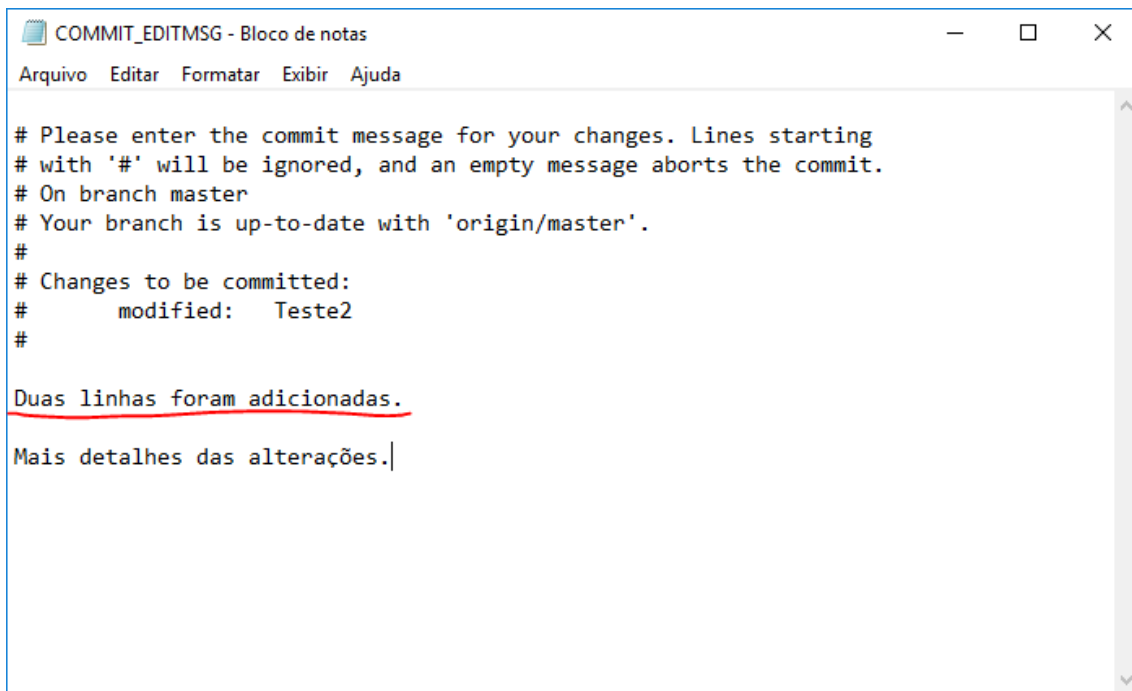


Figura 12: Usando a primeira linha como título (sublinhado em vermelho). O ideal é que a partir da segunda linha, as alterações sejam mais detalhadas.

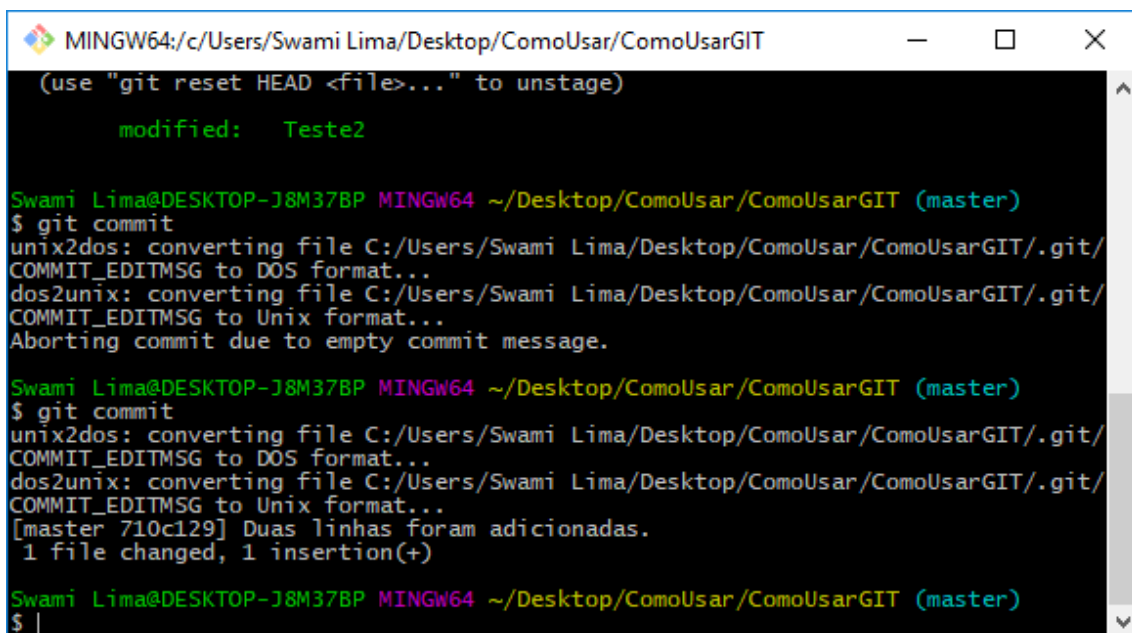
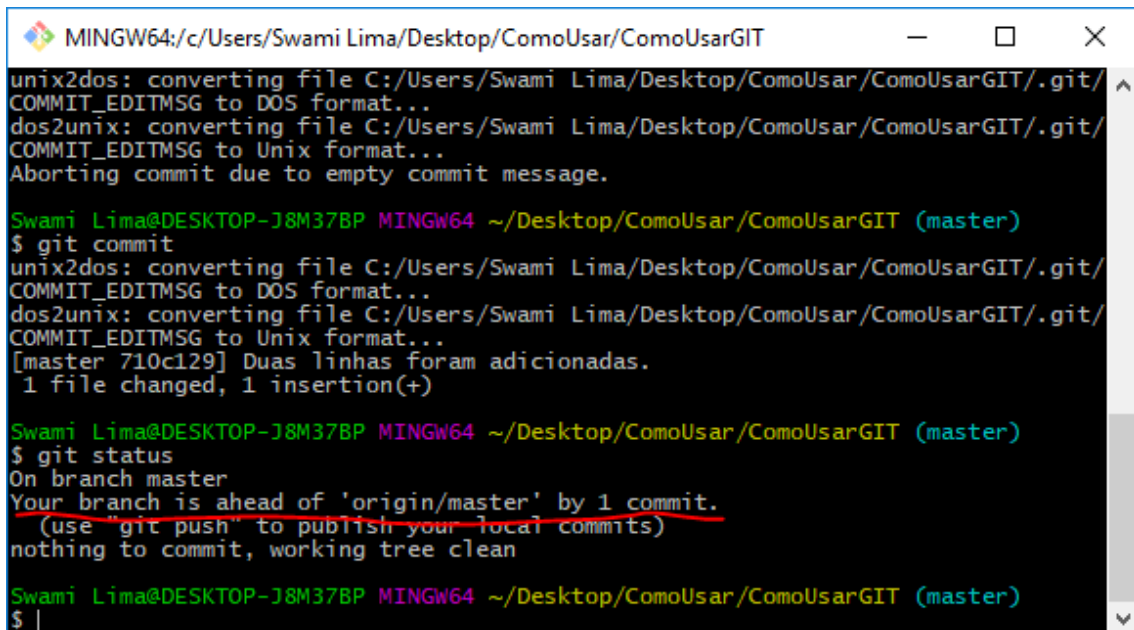


Figura 13: O GIT confirmando que um arquivo foi submetido (commit) e que está pronto para ser enviado ao servidor.

Apesar das alterações, o arquivo ainda não foi enviado para o servidor. Para que isso aconteça, deve-se usar o comando *push* (empurrar) para subir o que foi alterado.

Se usar o comando *status* ele vai informar que sua pasta está com um arquivo “a frente” do servidor master, ou seja: que os seus arquivos estão mais atuais.

A screenshot of a Windows command prompt window titled "MINGW64: c:/Users/Swami Lima/Desktop/ComoUsar/ComoUsarGIT". The window shows the output of several Git commands. First, a failed commit attempt is shown with messages about converting file formats and an empty commit message. Then, a successful commit is shown with the message "[master 710c129] Duas linhas foram adicionadas. 1 file changed, 1 insertion(+)". Finally, the 'git status' command is run, showing that the local branch is ahead of the origin/master by one commit, with a red line underlining the instruction to use 'git push' to publish the local commits.

```
MINGW64: c:/Users/Swami Lima/Desktop/ComoUsar/ComoUsarGIT
unix2dos: converting file C:/Users/Swami Lima/Desktop/ComoUsar/ComoUsarGIT/.git/COMMIT_EDITMSG to DOS format...
dos2unix: converting file C:/Users/Swami Lima/Desktop/ComoUsar/ComoUsarGIT/.git/COMMIT_EDITMSG to Unix format...
Aborting commit due to empty commit message.

Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop/ComoUsar/ComoUsarGIT (master)
$ git commit
unix2dos: converting file C:/Users/Swami Lima/Desktop/ComoUsar/ComoUsarGIT/.git/COMMIT_EDITMSG to DOS format...
dos2unix: converting file C:/Users/Swami Lima/Desktop/ComoUsar/ComoUsarGIT/.git/COMMIT_EDITMSG to Unix format...
[master 710c129] Duas linhas foram adicionadas.
1 file changed, 1 insertion(+)

Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop/ComoUsar/ComoUsarGIT (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)
nothing to commit, working tree clean

Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop/ComoUsar/ComoUsarGIT (master)
$
```

Figura 14: O comando STATUS informa que o usuário tem um arquivo para ser atualizado com o servidor.

Agora, vamos atualizar o servidor com o comando *push*. É normal que o servidor peça o usuário e a senha para autenticação no GITHUB (para saber se você é um usuário que pode alterar arquivos). No meu caso, ele me pediu apenas uma vez, e salvou a informação do meu login e senha do GITHUB.

Digite o comando:

```
git push
```

```
MINGW64:/c/Users/Swami Lima/Desktop/ComoUsar/ComoUsarGIT
Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop/ComoUsar/ComoUsarGIT (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)
nothing to commit, working tree clean

Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop/ComoUsar/ComoUsarGIT (master)
$ git pull
Already up-to-date.

Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop/ComoUsar/ComoUsarGIT (master)
$ git push
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 324 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/swamilima/ComoUsarGIT.git
17e966b..710c129 master -> master

Swami Lima@DESKTOP-J8M37BP MINGW64 ~/Desktop/ComoUsar/ComoUsarGIT (master)
$ |
```

Figura 15: Informações aparecem após usar o comando PUSH.

Agora, se formos na página do projeto no GITHUB, podemos ver que nosso projeto está atualizado, e a linha de título “nova”, aparece.

swamilima committed on GitHub Create Teste2		Latest commit 17e966b an hour ago
LICENSE	Initial commit	an hour ago
README.md	Initial commit	an hour ago
Teste1	Create Teste1	an hour ago
Teste2	Create Teste2	an hour ago

Figura 16: a página do projeto, antes do PUSH.

swamilima Duas linhas foram adicionadas. ...		Latest commit 710c129 12 minutes ago
LICENSE	Initial commit	an hour ago
README.md	Initial commit	an hour ago
Teste1	Create Teste1	an hour ago
Teste2	Duas linhas foram adicionadas.	12 minutes ago

Figura 17: a página do projeto, depois do PUSH.

Isso é o básico. Agora, é produzir.

Qualquer outra dúvida, falem comigo.