# RESUMO:

**RAMON CASTRO BARBOSA**

O necessário para usar o Git com plataformas como GitLab ou GitHub é necessário instalar o Git, disponível na plataforma base: https://git-scm.com

O VSCode possui interação com o Git através de plugins, assim como o Git Bash para Windows ou o Git via terminal nos SO de Kernel Linux.

Usamos o git resumidamente para copiar repositórios de outros usuários, nossos mesmos que estejam hospedados em plataformas com suporte a ao Git como o GitHub, para realizarmos alterações em nossos projetos usando qualquer máquina que possua integração do git e que tenha vínculo em nosso perfil de hospedagem, com o git podemos voltar também a versões anteriores de um projeto, ok, mas como fazemos isso? Bom, temos comandos para realizar essas ações citadas e muitas outras.

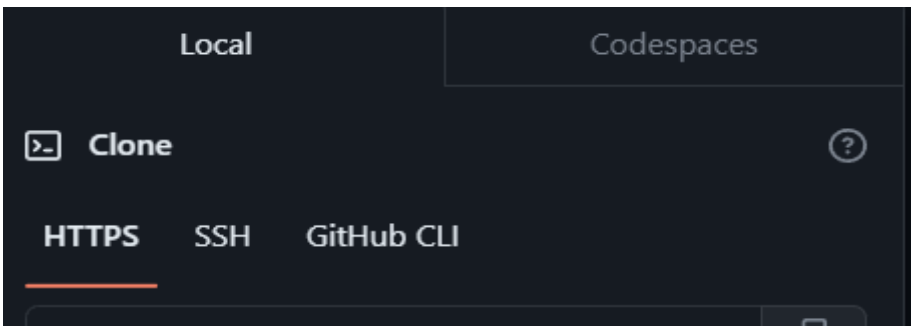
No sistema de git temos os branchs, isso serve para quando temos um tipo grande, isso para termos tarefas bem divididas e cada pessoa tenha seu branch com seus códigos, o git possui outros comandos entre os de clonagem de repositórios, atualização do repositório local com base no repositório principal e muito mais, o git é usado por desenvolvedores pelo mundo todo e em especial para usar do mesmo para trabalho em equipe como por exemplo entre um brasileiro e um irlandes.

# OS COMANDOS

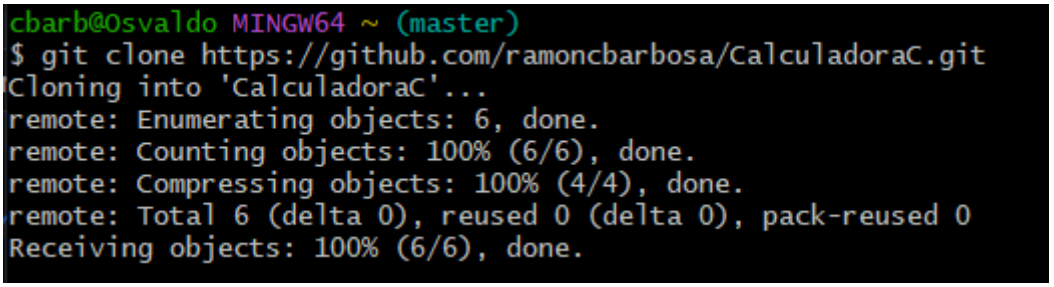
Para vermos a versão instalada do Git, temos co comando git version, ele vale para Windows e SO's Linux:

```
cbarb@Osvaldo MINGW64 ~
$ git --version
git version 2.40.0.windows.1
```

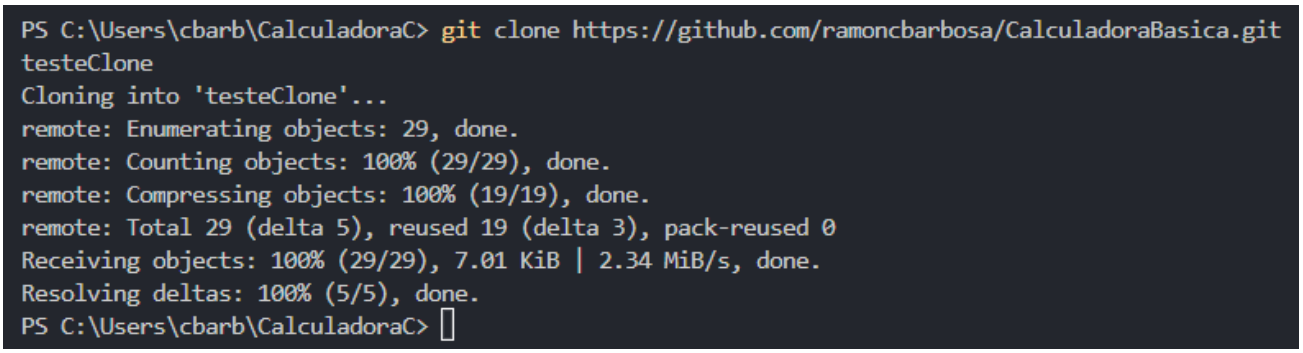
Mas antes disso, temos formas de clonar um repositório para nossa máquinas, como faremos isso? Precisaremos da url do repositório, por exemplo:



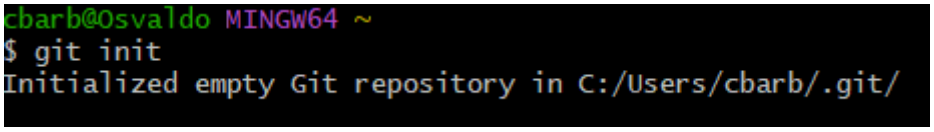
E agora com o endereço do repositório irei usar o comando de clonagem do git, o git clone:




Isso irá clonar o repositório para a pasta padrão do sistema, caso queiramos mudar o local, temos como definir colocando o caminho depois do endereço do repositório, assim:

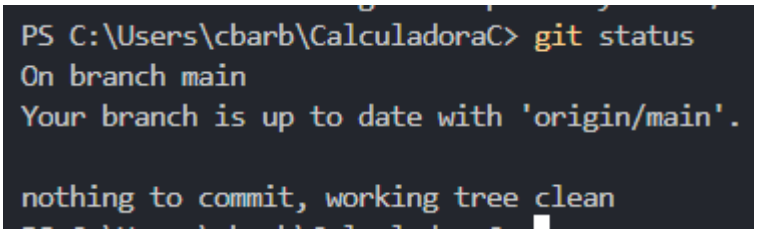



Para iniciarmos uma repositório na pasta atual do sistema, usamos o comando para usarmos um projeto que queremos trabalhar nele:




DÚVIDAS: Como vou salvar o projeto o qual clonei para a minha máquina, como vou subir essas alterações?

Usaremos primeiro um comando para ver o status atual do projeto com o git status, para sabermos o que será versionado:



Como faço para adicionar e salvar o que foi feito no arquivo, bom, usarmos um comando com duas variações, o git add e git add .

Git Add onde irei adicionar um arquivo especifico ao repositório:

```
C:\Users\cbarb\CalculadoraC> git add .\calculadora.c
```

Git Add . (ponto) onde irei adicionar todos os arquivos novos ou alterados ao repositório:

```
PS C:\Users\cbarb\CalculadoraC> git add .
PS C:\Users\cbarb\CalculadoraC> git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   lixo.c
        new file:   lixoDois.c
```

OBS: Usando o comando git status podemos ver o que foi salvo, mas vale ressaltar que ainda não fizemos de fato o commit, que seria de fato subir para o repositório as alterações.

Como vou salvar de fato então?

Usaremos o comando git commit com duas variações:

Git commit –a: Esse comando fará todos os arquivos de diversas extensões subirem.

Git commit -m “***”: Esse comando fará um comentário sobre o que estou subindo, deixando uma mensagem no que eu subi ou no que foi alterado, como mensagem para outros dev, essa mensagem fica entre aspas duplas:

```
PS C:\Users\cbarb\CalculadoraC> git commit -a -m "estudo para trainee"
[main 50dcc96] estudo para trainee
 3 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 lixo.c
 create mode 100644 lixoDois.c
 create mode 100644 lixoTres.c
PS C:\Users\cbarb\CalculadoraC>
```

---

## CRIAÇÃO DE REPOSITÓRIOS NO GITHUB:

No GitHub depois de ter criado um perfil, você irá:

1) Aba “Repositórios”;
2) New;
3) E então na caixa “Repository name” e colocará um nome para o seu repositório;
4) Colocar uma descrição no campo “Description” (opcional);

5) Terá duas opções do tipo de repositório, “Public” ou “Private”;
6) Pode adicionar uma tipo de arquivo de texto para dizer os detalhes do repositório na aba “Add a README file” (opcional);
7) Pode adicionar também um outras coisas...
8) Por fim, no final vá até o “Create repository” para a criação do novo repositório;

OBS¹: Claramente este repositório estará vazio e ao criar esse novo repositório, teremos alguns comandos como o git init e outros mais;

OBS²: Teremos comandos para montarmos o repositório em nossa máquina e assim usarmos ele;

---

Comandos Caso Já Tenhamos Algo e Linkarmos o Reposiório Local Com o do GitHub:

Git Remotee é para adicionarmos uma origem remota a um repositório:

```
PS C:\Users\cbarb> git remote add origin https://github.com/ramoncbarbosa/TraineeJR.git
>>
```

O comando Git branch –M main é para definir qual será o meu branch principal:

```
PS C:\Users\cbarb> git branch -M main
```

O comando Git Push –u origin main é para enviarmos os códigos do main da máquina virtual para o repositório oficial:

```
PS C:\Users\cbarb\CalculadoraC> git push -u origin main
info: please complete authentication in your browser...
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 336 bytes | 336.00 KiB/s, done.
```

Temos o comando git checkout –b “***” é para criarmos um branch, seria como um branch dos devs que estão testando uma versão beta que só “eu” poderei ver enquanto o branch geral continue visível a todos:

```
PS C:\Users\cbarb\CalculadoraC> git checkout -b "teste"
Switched to a new branch 'teste'
```

OBS: Se ediar o arquivo no branch teste e eu quiser fazer um push, terei o retorno como erro, já que o que eu editei esta no novo branch e não no branch geral, para corrigir o que faremos?

Usaremos o comando “git push --set-upstream origin teste”, assim vamos subir as alterações, mas faremos isso criando um branch chamado “teste”. Ok, mas como posso ver isso no repositório? Basta entrar no repositório e clicar no “main” e verá agora uma aba escrita main e outra escrita teste;

```
PS C:\Users\cbarb\CalculadoraC> git push --set-upstream origin teste
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 237 bytes | 237.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
```

OBS: Mas caso o que estiver em teste seja autorizado para uso dos usuários, como se uni os códigos do branch teste para o branch main?

Primeiro vamos entender como mudar de branchs, para isso usaremos o comando “git branch” para vermos quais branch temos:

```
PS C:\Users\cbarb\CalculadoraC> git branch
  main
* teste
```

Agora para mudarmos entre nossos branch usaremos o comando “git main” ou git NomeDoBranch:

```
PS C:\Users\cbarb\CalculadoraC> git checkout main
Switched to branch 'main'
```

OBS: Saberemos em qual branch estamos quando o branch estiver com uma estrela (*) antes do nome dele, assim:

```
* main
  teste
```

Para unirmos os códigos usarmos o comando merge, assim uniremos o código da branch teste na branch main, como no comando abaixo:

```
PS C:\Users\cbarb\CalculadoraC> git merge teste
Updating 50dcc96..35b35d8
Fast-forward
 README.md | 1 -
 1 file changed, 1 deletion(-)
 delete mode 100644 README.md
```

OBS: Pós testar o código depois do merge, enviaremos ele para o repositório no main via git push, depois será possível ver isso no repositório indo no main.

Como trazer ou manter sempre o repositório local em igualdade com o repositório principal? Para isso podemos usar o comando “git pull” para trazer as alterações do repositório principal para o local:

```
PS C:\Users\cbarb\CalculadoraC> git pull
Already up to date.
PS C:\Users\cbarb\CalculadoraC>
```

Caso tenha um arquivo não útil ao projeto que irá remover arquivos não desejados, ele saíra do  untracked, para isso basta usar o comando “git clean –f”:

```
PS C:\Users\cbarb\CalculadoraC> git clean -f
PS C:\Users\cbarb\CalculadoraC>
```

OBS: Arquivos untracked estarão sempre em vemelho, para vermos isso basta usar o “git status”.

Temos um comando para sabermos todos os logs do repositório.

OBS: Log é um “caminho” que o repositório já percorreu.

```
commit 35b35d8124727fcb23082f348eeda6447d8ee3fa (HEAD -> main, origin/teste, origin/main, origin/HEAD, test
e2, teste)
Author: Ramon C. Barbosa <ramon.cobarbosa@gmail.com>
Date:   Sat Apr 15 15:26:57 2023 -0300

    teste com js para estudar

commit 50dcc96d722acc5cca64cb6786388692bfc7b1e4
Author: Ramon C. Barbosa <ramon.cobarbosa@gmail.com>
Date:   Sat Apr 15 13:48:41 2023 -0300

    estudo para trainee

commit 28f0dd916d6e643eece51ca7b1f8e4781f35dba9
Author: Ramon Barbosa <83264505+ramoncbarbosa@users.noreply.github.com>
Date:   Sat Mar 4 14:41:05 2023 -0300

    Add files via upload

commit 28725beeab05f2ab297778b7307bb02ae958a421
Author: Ramon Barbosa <83264505+ramoncbarbosa@users.noreply.github.com>
Date:   Sat Mar 4 14:40:26 2023 -0300

    Initial commit
```