

O que você deveria saber sobre Git

SERGIO CABRAL



sergiocabral.com

Índice

1. Perguntas e respostas rápidas	2
2. Git na prática.....	3
2.1. Como criar ou baixar um repositório Git.....	3
2.2. O básico do básico.....	5
2.3. <i>Branches</i> e <i>Tags</i>	7

Apresentação preparada para demonstrar o básico do Git para iniciantes numa palestra online na EBAC.

1. Perguntas e respostas rápidas

O que é o Git?

Um software de linha de comando conhecido como **VCS ou SCM**.



(D)VCS = (Distributed) Version Control System
SCM = Source Control Management

O que o Git faz?

Uma ferramenta que possibilita fazer alterações em um conjunto de arquivos e **registrar cada alteração numa linha do tempo**.

Por que preciso de um Git?

Além de manter um histórico de versões, torna possível **trabalhar com equipes** distribuídas.



Lembre-se que o seu **eu** de amanhã será diferente do seu **eu** de hoje e vocês precisam trabalhar em equipe.

Como começo a usar o Git?

Você instala no modo *avançar-avançar* e transforma qualquer pasta em um repositório Git com o comando **git init**. Ou você pode usar o comando **git clone** para clonar um repositório existente.

Saiba também

- Git não tem interface gráfica; é linha de comando.
- Git não é a única ferramenta do tipo, mas é a mais utilizada.
- Git não é GitHub.

2. Git na prática

2.1. Como criar ou baixar um repositório Git

Resultado do `git init`.

```
root :: pwsh :: my-super-app
D:\Git\ebac-talk
~#> mkdir my-super-app

Directory: D:\Git\ebac-talk

Mode                LastWriteTime         Length Name
----                -
d-----          04/11/2021  19:27             my-super-app

D:\Git\ebac-talk
~#> cd .\my-super-app\
D:\Git\ebac-talk\my-super-app
~#> git init
Initialized empty Git repository in D:/Git/ebac-talk/my-super-app/.git/
D:\Git\ebac-talk\my-super-app > |master #
~#> dir
D:\Git\ebac-talk\my-super-app > |master #
~#> dir -Attributes Hidden
Directory: D:\Git\ebac-talk\my-super-app

Mode                LastWriteTime         Length Name
----                -
d--h-             04/11/2021  19:28             .git

D:\Git\ebac-talk\my-super-app > |master #
~#>
```

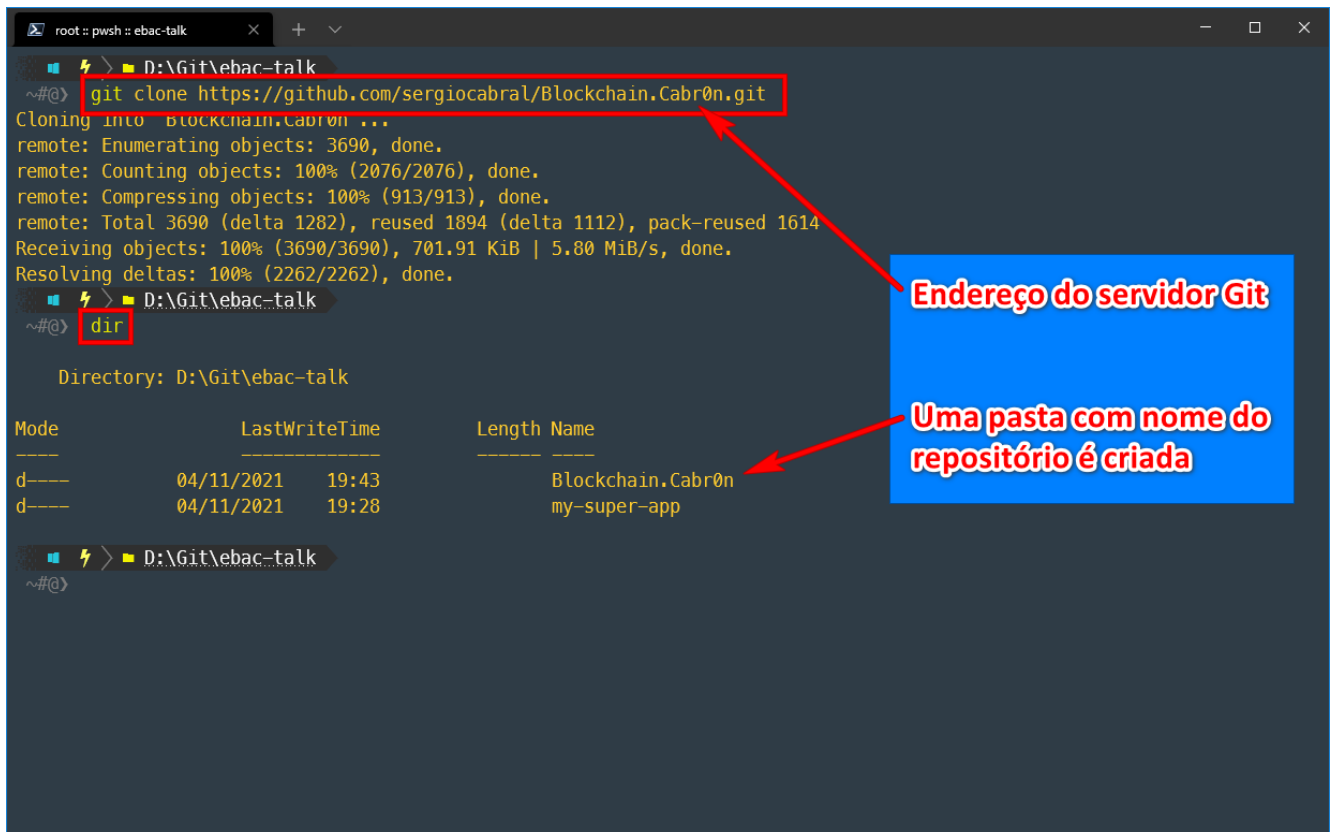
Cria uma pasta

Após o 'git init' a pasta continua sem conteúdo

Mas na verdade tem uma pasta oculta .git

Exemplo 1.

Resultado do `git clone`.



The image shows a Windows terminal window with the following content:

```
root :: pwsh :: ebac-talk
D:\Git\ebac-talk
~#> git clone https://github.com/sergiocabral/Blockchain.Cabr0n.git
Cloning into 'Blockchain.Cabr0n' ...
remote: Enumerating objects: 3690, done.
remote: Counting objects: 100% (2076/2076), done.
remote: Compressing objects: 100% (913/913), done.
remote: Total 3690 (delta 1282), reused 1894 (delta 1112), pack-reused 1614
Receiving objects: 100% (3690/3690), 701.91 KiB | 5.80 MiB/s, done.
Resolving deltas: 100% (2262/2262), done.
D:\Git\ebac-talk
~#> dir

Directory: D:\Git\ebac-talk

Mode                LastWriteTime         Length Name
----                -
d-----          04/11/2021    19:43             Blockchain.Cabr0n
d-----          04/11/2021    19:28             my-super-app
```

Annotations in the image:

- A red box highlights the command `git clone https://github.com/sergiocabral/Blockchain.Cabr0n.git`. A red arrow points from this box to a blue box containing the text "Endereço do servidor Git".
- A red box highlights the command `dir`. A red arrow points from this box to a blue box containing the text "Uma pasta com nome do repositório é criada".

Exemplo 2.



O Git grava seu banco de dados na pasta `.git`.

2.2. O básico do básico

`git status`

Estou num repositório Git?

Quais arquivos ainda não estão no repositório?

`git add <nome-do-arquivo>`

Marca arquivos para entrarem no repositório.

`git commit -m "<mensagem>"`

Grava arquivos no repositório.

`git log`

Mostra o histórico de alterações.

```
root :: pwsh : my-super-app
D:\Git\ebac-talk\my-super-app > |master ̸
~#@> echo "conteúdo qualquer" > my-file
D:\Git\ebac-talk\my-super-app > |master ̸ +1
~#@> git status
On branch master

No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)
D:\Git\ebac-talk\my-super-app > |master ̸ +1
~#@> git add .\my-file
D:\Git\ebac-talk\my-super-app > |master ̸ +1
~#@> git status
On branch master

No commits yet

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

D:\Git\ebac-talk\my-super-app > |master ̸ +1
~#@> git commit -m "uma mensagem descritiva aqui"
[master (root-commit) f019806] uma mensagem descritiva aqui
1 file changed, 1 insertion(+)
create mode 100644 my-file
D:\Git\ebac-talk\my-super-app > |master ̸
~#@> git log
commit f019806e7923e3477da92378afb0175f7f0aefed (HEAD -> master)
Author: Sergio Cabral <git@sergiocabral.com>
Date:   Tue Nov 16 09:22:52 2021 -0300

    uma mensagem descritiva aqui
D:\Git\ebac-talk\my-super-app > |master ̸
~#@>
```

⓪ que está prestes a entrar no repositório é chamado de "stage", "staging area" ou "index"

Exemplo 3.



Há três lugares onde seus arquivos podem estar
TTT, *The Three Trees* (as três árvores)

O que está na pasta do seu projeto	Working Directory	Gerenciado pelo sistema operacional.
O que vai para o repositório	Staging Area	Próximo <i>commit</i> ainda <u>não feito</u> .
O repositório	HEAD	Último <i>commit</i> <u>feito</u> e Pai do próximo <i>commit</i> .

2.3. Branches e Tags

Commits sempre são sequências de 40 caracteres (um *hash* gerado com *SHA1*).
Por exemplo: f019806e7923e3477da92378afb0175f7f0aefed

Branches e *tags* são apenas nomes para *commits*.
Ao apagar *branches* ou *tags* nenhum dado é perdido.

Criar branches

- `git branch <nome-do-branch>`
- `git checkout -b <nome-do-branch>`

Excluir branches

- `git branch -D <nome-do-branch>`



Quando se usa `-d` (minúsculo) o Git não vai excluir branches que apontam para commits que nunca sofreram *merge*.



Exemplo 4.

Criar tags

- `git tag <nome-da-tag> -m "<descrição-opcional>"`

Excluir tags

- `git tag -d <nome-da-tag>`



Branches e *tags* são semelhantes, mas a **diferença** está em que:

- Um **branch** acompanha cada novo *commit* feito a partir dele.
- Uma **tag** é permanente para um determinado *commit* na corrente do tempo.