

Card25_Git e Github para Iniciantes

Davi Bezerra Barros

O sistema de versionamento Git

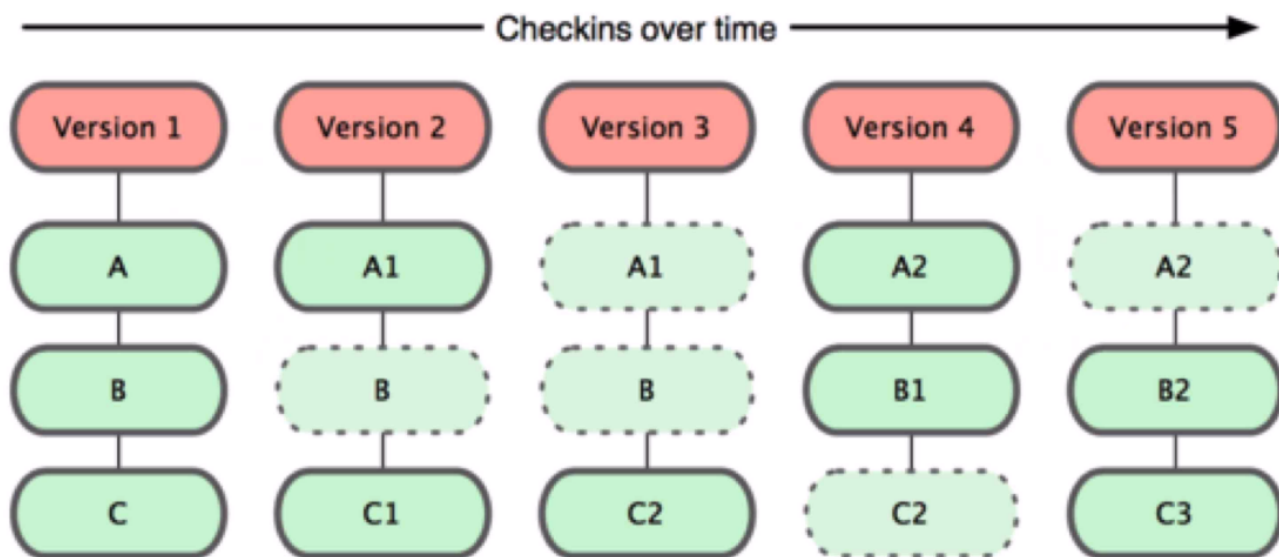
O que é Git?

Git é um sistema de controle de versão open source criado por Linus Torvalds, e é utilizado para desenvolvimento de software. Ele permite a colaboração de vários desenvolvedores no mesmo projeto de forma assíncrona, e rastreia as alterações no código fonte ao longo do tempo, permitindo que as mudanças feitas sejam revertidas ou alteradas.

Funcionamento

Diferentemente de outros sistemas que trabalham com as diferenças entre arquivos, o Git trabalha com seus **estados**; ele tira *snapshots* dos estados dos arquivos para cada versão, independentemente de terem sido alterados ou não. Caso o arquivo não tenha sido alterado, sua referência será guardada para manter a integridade de todo o conjunto de arquivos, e a versão completa.

Sistema Git



Os arquivos e os históricos são guardados localmente em cada repositório clonado, fazendo com que o projeto possa evoluir de forma assíncrona.

Comandos e definições básicas

A seguir estão descritas algumas definições básicas do git:

Git config: Define as configurações do Git em três níveis diferentes: Para todos os repositórios do usuário(global), para o repositório atual(local), ou para todos os usuários e repositórios da

máquina(System). É utilizado para definir o nome de usuário, email, listar configurações, definir o editor de texto, etc.

```
[davi@daviPc ~]$ git config --global user.name "Davi Bezerra"
[davi@daviPc ~]$ git config --global user.name
Davi Bezerra
[davi@daviPc ~]$ git config --global user.email "bbarrosdavi@gmail.com"
[davi@daviPc ~]$ git config --global user.email
bbarrosdavi@gmail.com
[davi@daviPc ~]$ git config --global core.editor nano
[davi@daviPc ~]$ git config --list
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
user.name=Davi Bezerra
user.email=bbarrosdavi@gmail.com
core.editor=nano
[davi@daviPc ~]$
```

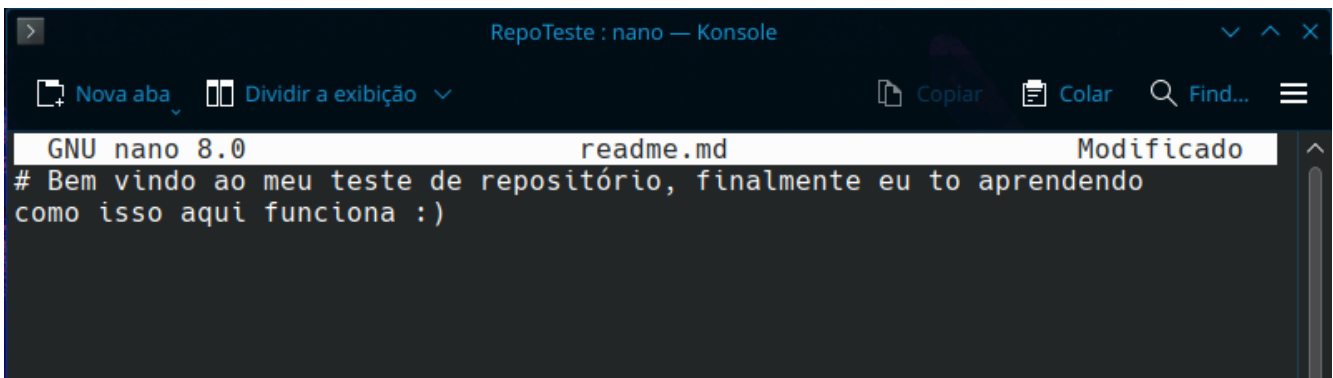
Inicializar repositório: Repositório é o local onde estão guardados os arquivos de um projeto, e seu histórico de versões. O **git init** é o comando utilizado para converter o diretório atual em um repositório git:

```
[davi@daviPc Prática_GitHub]$ mkdir RepoTeste
[davi@daviPc Prática_GitHub]$ cd RepoTeste
[davi@daviPc RepoTeste]$ git init
hint: Using 'master' as the name for the initial branch. This default branch
hint: name
hint: is subject to change. To configure the initial branch name to use in a
hint: ll
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command
hint: :
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/davi/Documentos/GitHub/LAMIA/Bootc
amp_MachineLearning/Prática_GitHub/RepoTeste/.git/
[davi@daviPc RepoTeste]$ ls -la
total 12
drwxr-xr-x 3 davi davi 4096 out  9 18:30 .
drwxr-xr-x 3 davi davi 4096 out  9 18:30 ..
drwxr-xr-x 7 davi davi 4096 out  9 18:30 .git
[davi@daviPc RepoTeste]$ cd .git/
[davi@daviPc .git]$ ls
branches  config  description  HEAD  hooks  info  objects  refs
[davi@daviPc .git]$
```

- **branches:** Pasta onde estão armazenados os *branches* do repositório; são ramificações do projeto original utilizados para desenvolver funcionalidades ou corrigir bugs. Os branches são como ponteiros que apontam para um commit específico na história do projeto.
- **config:** Contém as configurações específicas do repositório local. Ele define qual branch está ativa, qual é o repositório remoto (se houver), e outras configurações.

- **description:** Utilizado por interfaces de repositórios remotos como GitWeb e GitHub.
- **head:** Indica qual branch ou commit em que o usuário está trabalhando no momento.
- **hooks:** Contém scripts de gatilhos que são executados em momentos específicos no repositório, para ações específicas.
- **info :** Armazena dados auxiliares sobre o repositório, como o arquivo `exclude` , que permite configurar exclusões de arquivos específicas desse repositório, funcionando de forma similar ao `.gitignore` , mas local.
- **objects:** Contém todos os objetos do repositório, incluindo commits, blobs (conteúdo dos arquivos), árvores (estrutura de diretórios) e outros objetos Git.
- **refs:** Contém referências para heads (branches), tags, e repositórios remotos. É onde o Git mantém as pontas das branches e tags no histórico do repositório.

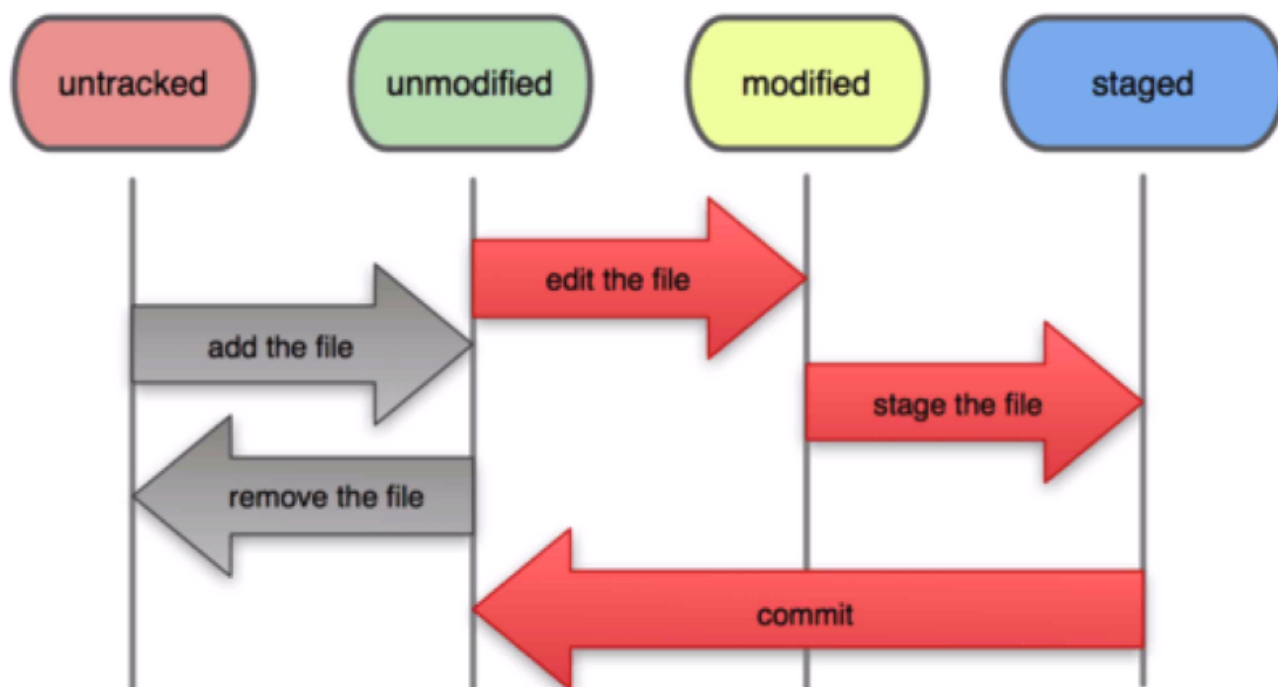
Também é possível utilizar um editor de texto para alterar os arquivos git diretamente no terminal(ou não):



Ciclo de vida e status dos arquivos:

O git separa os arquivos em quatro status bem definidos:

File Status Lifecycle



- **untracked**: Arquivo que acabou de ser adicionado no repositório, e o git não o reconhece em nenhuma versão do repositório.
- **unmodified**: O arquivo existe, mas não sofreu nenhuma alteração desde o último commit, está em seu estado mais recente.
- **modified**: Arquivo que já está sendo rastreado e que foi modificado desde sua última versão. O Git detecta que o conteúdo do arquivo foi alterado, mas essas modificações ainda não foram preparadas para serem comitadas.
- **staged**: Arquivo que foi modificado e está preparado para commit.
- **commit**: É o processo de salvar as alterações rastreadas no histórico do repositório. O git registra o estado atual dos arquivos que estão preparados(staged) e salva uma fotografia destes arquivos, criando um ponto de restauração.

Rastreando novos arquivos e alterações

git status: Reporta a situação atual do repositório:

```
[davi@daviPc RepoTeste]$ touch Readme.md
[davi@daviPc RepoTeste]$ nano Readme.md
[davi@daviPc RepoTeste]$ git status
On branch master

No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)
[davi@daviPc RepoTeste]$ git add Readme.md
[davi@daviPc RepoTeste]$ git status
On branch master

No commits yet

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

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

Arquivo novo

1: Arquivos não rastreados

2: Arquivo alterado, pronto para commit

0. Arquivo "Readme.md" foi criado
1. O arquivo criado foi reconhecido mas não está rastreado
2. O arquivo foi alterado e sua alteração foi reconhecida

```

[davi@daviPc RepoTeste]$ nano Readme.md
[davi@daviPc RepoTeste]$ git status
On branch master

No commits yet

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

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:   Readme.md
3: Arquivo alterado novamente

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        readme.md
4: Modificação reconhecida

[davi@daviPc RepoTeste]$ git add Readme.md
[davi@daviPc RepoTeste]$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   Readme.md
5: Adicionado novamente ao stage

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

[davi@daviPc RepoTeste]$ █

```

3. O arquivo foi alterado novamente
4. A modificação foi reconhecida e o arquivo está marcado como *Modified*
5. Arquivo adicionado novamente ao stage com o git add

Git commit: Cria um snapshot de todos os arquivos que foram alterados. Comentar o commit é sempre uma prática recomendada. O parâmetro -m define a mensagem.

```

[davi@daviPc RepoTeste]$ git commit -m "Add Readme"
[master (root-commit) 1013800] Add Readme
 1 file changed, 2 insertions(+)
 create mode 100644 Readme.md
[davi@daviPc RepoTeste]$ █

```

- 100644: Identificador do commit.
- 2 insertions: Quantidades de linhas que foram alteradas.

```

[davi@daviPc RepoTeste]$ git status
On branch master
nothing to commit, working tree clean
[davi@daviPc RepoTeste]$ █

```

Após o git commit, não há mais alterações rastreadas no repositório. Não é possível fazer o commit de alguma alteração sem antes adicioná-la ao staged.

Visualizando alterações

git log:

```
[davi@daviPc RepoTeste]$ git log
commit 1013800b4b659e821ff6f36a6749cbf829101cf1 (HEAD -> master)
Author: Davi Bezerra <bbarrosdavi@gmail.com>
Date:   Wed Oct 9 19:21:06 2024 -0300

    Add Readme
[davi@daviPc RepoTeste]$
```

O comando **git log** é utilizado para verificar as alterações no repositório. Exibe os commits, autores, data, código hash e outras informações importantes. Também é possível listar as alterações de modo mais sucinto com **git shortlog**, filtrar autores com **git log --author = "autor"**, e outros parâmetros úteis.

git show "hash": Exibe as informações de uma commit específica a partir de sua hash de identificação.

git diff: Exibe as linhas que foram alteradas desde a versão anterior:

```
[davi@daviPc RepoTeste]$ nano Readme.md
[davi@daviPc RepoTeste]$ git diff
diff --git a/Readme.md b/Readme.md
index c6a8258..bb4101f 100644
--- a/Readme.md
+++ b/Readme.md
@@ -1,2 +1,3 @@
 oh, testezinho hiii
 teste teste teste
+teste para mostrar a diff
[davi@daviPc RepoTeste]$
```

- **+++**: arquivos adicionados
- **---**: arquivos removidos
- **+**: linhas modificadas
- **git diff --name only**: Parâmetro que mostra apenas o nome dos arquivos que foram alterados, caso haja muitos.

Desfazendo alterações

Modificação no arquivo:

```
[davi@daviPc RepoTeste]$ git status
On branch master
nothing to commit, working tree clean
[davi@daviPc RepoTeste]$ nano Readme.md
[davi@daviPc RepoTeste]$ 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:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
[davi@daviPc RepoTeste]$ git diff
diff --git a/Readme.md b/Readme.md
index bb4101f..53f6e4b 100644
--- a/Readme.md
+++ b/Readme.md
@@ -1,3 +1,4 @@
 oh, testezinho hiii
 teste teste teste
 teste para mostrar a diff
+Teste: remover alteração
[davi@daviPc RepoTeste]$ #nao quero
[davi@daviPc RepoTeste]$ git checkout.[]
```

git checkout: Retorna o arquivo especificado para sua última versão:

```
[davi@daviPc RepoTeste]$ #nao quero
[davi@daviPc RepoTeste]$ git checkout Read.md
error: pathspec 'Read.md' did not match any file(s) known to git
[davi@daviPc RepoTeste]$ git checkout README.md
Updated 1 path from the index
[davi@daviPc RepoTeste]$ git diff
[davi@daviPc RepoTeste]$ cat README.md
oh, testezinho hiii
teste teste teste
teste para mostrar a diff
[davi@daviPc RepoTeste]$ ~
```

git reset:

- --HEAD: Remove alterações do stage.
- --soft: Remove o commit, e retorna o arquivo para *staged*, com a alteração realizada.
- --mixed: Remove o commit e retorna o arquivo para *modified*, a alteração ainda está lá.
- --hard: Remove o commit e mata todas as alterações feitas em todos os arquivos no commit.

soft:

```
[davi@daviPc RepoTeste]$ git log
commit 09c983ec839f4c03954fceb0fa196de876068c72 (HEAD -> master)
Author: Davi Bezerra <bbarrosdavi@gmail.com>
Date:   Wed Oct 9 19:58:26 2024 -0300

    teste reset soft

commit 2b0cb280c1f04375e96e1583005db3ad721d15f5
Author: Davi Bezerra <bbarrosdavi@gmail.com>
Date:   Wed Oct 9 19:52:30 2024 -0300

    lalalala

commit 9f1aa7a9a091c30c2c727e907c976b504a99476e
Author: Davi Bezerra <bbarrosdavi@gmail.com>
Date:   Wed Oct 9 19:43:10 2024 -0300

    teste diff

commit 1013800b4b659e821ff6f36a6749cbf829101cf1
Author: Davi Bezerra <bbarrosdavi@gmail.com>
Date:   Wed Oct 9 19:21:06 2024 -0300

    Add Readme
[davi@daviPc RepoTeste]$ git reset --soft 2b0cb280c1f04375e96e1583005db3ad721d15f5
[davi@daviPc RepoTeste]$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md
[davi@daviPc RepoTeste]$
```

mixed:

```
[davi@daviPc RepoTeste]$ git reset --mixed 9f1aa7a9a091c30c2c727e907c976b504a99476e
Unstaged changes after reset:
M       README.md
[davi@daviPc RepoTeste]$
```

hard:

```
[davi@daviPc RepoTeste]$ cat Readme.md
adibgabdga0
oh, testezinho hiii
teste teste teste
teste para mostrar a diff
tira git soft
[davi@daviPc RepoTeste]$ git log
commit 3045226f765f04319cf2e58df69ab30a1ebacfc1 (HEAD -> master)
Author: Davi Bezerra <bbarrosdavi@gmail.com>
Date:   Wed Oct 9 20:04:10 2024 -0300

    pronto pro hard

commit 9f1aa7a9a091c30c2c727e907c976b504a99476e
Author: Davi Bezerra <bbarrosdavi@gmail.com>
Date:   Wed Oct 9 19:43:10 2024 -0300

    teste diff

commit 1013800b4b659e821ff6f36a6749cbf829101cf1
Author: Davi Bezerra <bbarrosdavi@gmail.com>
Date:   Wed Oct 9 19:21:06 2024 -0300

    Add Readme
[davi@daviPc RepoTeste]$ git reset --hard 1013800b4b659e821ff6f36a6749cbf829101cf1
HEAD is now at 1013800 Add Readme
[davi@daviPc RepoTeste]$ git status
On branch master
nothing to commit, working tree clean
[davi@daviPc RepoTeste]$ cat Readme.md
oh, testezinho hiii
teste teste teste
[davi@daviPc RepoTeste]$
```

As alterações de todos os commits feitos após o 1013800 foram completamente deletadas.

Repositório remoto

É uma versão de um repositório que está hospedado em um servidor, e pode ser acessada por outros desenvolvedores para colaborar em um projeto. Um grande exemplo de repositório remoto é o GitHub, plataforma de versionamento de código que utiliza o Git como seu sistema de versionamento. Os comandos utilizados para interagir com repositórios remotos são:

- **push:** Envia as alterações do repositório local para o remoto.

```
[davi@daviPc RepoTeste]$ git remote add origin git@github.com:Gwafflezz/CursoGit.git
git branch -M main
git push -u origin main
The authenticity of host 'github.com (20.201.28.151)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvC0qU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 240 bytes | 240.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:Gwafflezz/CursoGit.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
[davi@daviPc RepoTeste]$ git push -u origin main
branch 'main' set up to track 'origin/main'.
Everything up-to-date
[davi@daviPc RepoTeste]$
```

- **branch:** Cria uma linha separada de desenvolvimento, para trabalhar em paralelo e de forma isolada de outros desenvolvedores, evitando conflitos e sem alterar o código principal.

```
[davi@daviPc RepoTeste]$ git checkout -b teste
Switched to a new branch 'teste'
[davi@daviPc RepoTeste]$ git branch
main
* teste
[davi@daviPc RepoTeste]$
```

- **fork:** É uma cópia do repositório de um projeto. Utilizado para fazer mudanças sem alterar o projeto original, e possibilita propor estas mudanças ao dono do projeto.
- **merge:** Combina dois branches diferentes integrando os commits de uma branch na outra, a partir de um novo commit.
- **rebase:** Diferente do merge, que mantém o histórico de commits das branches, o rebase reescreve este histórico, aplicando as commits linearmente branch principal.

Github Desktop

O GitHub desktop é uma implementação gráfica do git integrado aos repositórios remotos do github. Todas as funcionalidades discutidas anteriormente estão presentes na aplicação, de forma visual e mais intuitiva.