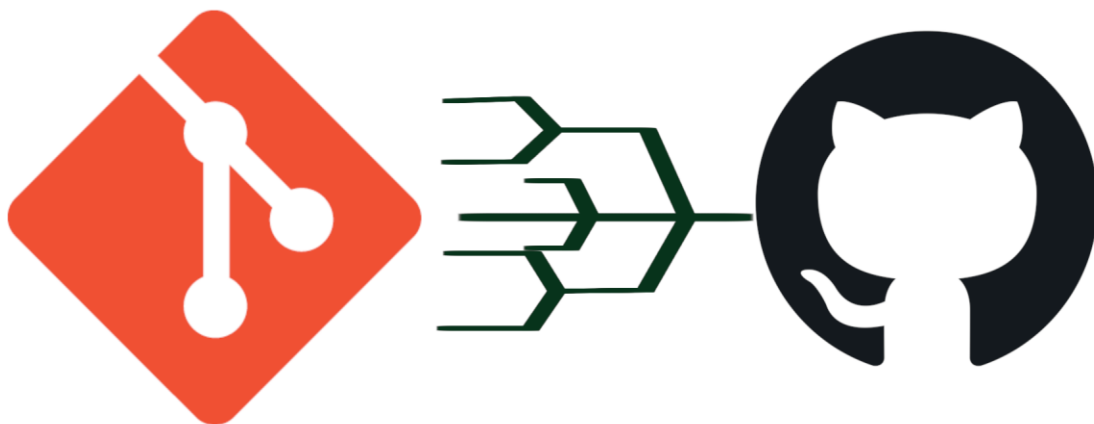


**Curso Básico de Git e Github**  
**Apostila Elaborada com Base nas Aulas**  
**[Git e Github para Iniciantes](#)**



# Curso de Git e Github

## Sumário

<b>Seção 1: Entendendo o que é o Git e Github .....</b>	<b>3</b>
<b>Seção 2: Configurando o Git .....</b>	<b>3</b>
<b>Seção 3: Essencial do Git .....</b>	<b>3</b>
Inicializando um repositório .....	3
Usando o editor do terminal .....	4
O ciclo de vida dos status de seus arquivos .....	4
Visualizando logs .....	5
Visualizando o diff .....	5
Desfazendo coisas .....	5
<b>Seção 4: Repositórios Remotos .....</b>	<b>6</b>
Criando e adicionando uma chave SSH .....	6
Ligando repositório local a um remoto .....	6
Enviando mudanças para um repositório remoto .....	7
Clonando repositórios remotos .....	7
Fazendo um fork de um projeto .....	7
<b>Seção 5: Ramificação (Branch) .....</b>	<b>7</b>
Criando um branch .....	8
Movendo e deletando branches .....	8
Entendendo o merge .....	9
Entendendo o rebase .....	10
Merge e rebase na prática .....	11
<b>Seção 6: Extras .....</b>	<b>16</b>
Criando o .gitignore .....	16
Git stash .....	17
Simplificação de comando .....	17
Versionamento com tags .....	17
Git revert .....	18
Apagando tags e branches remotos .....	18

## Seção 1: Entendendo o que é o Git e Github

Controle de versão: Sistema com a finalidade de gerenciar diferentes versões de um documento. É possível voltar para versões anteriores de um mesmo arquivo. Existem mais de um sistema além do git.

No caso do sistema git ele não verifica as diferenças do arquivo, ele tira snapshot dos estados do arquivo em diferentes versões.

O sistema de versionamento é responsável por “versionar” os arquivos do seu projeto, os outros sistemas trabalham com a diferença dos arquivos, enquanto o git trabalha com o estado dos arquivos.

Github: É o serviço de web compartilhado para projetos que utilizam o Git para versionamento. É um local na web que vai armazenar os projetos do git. O git é o sistema de controle de versão, enquanto o Github é apenas um local remoto para armazenamento.

## Seção 2: Configurando o Git

O git guarda as informações em três lugares: git config do sistema como um todo, git config do usuário e o git config do projeto específico.

Nome de usuário: `git config --global user.name "nome"`

Email: `git config --global user.email "email"`

Definir o editor: `git config --global core.editor NomeDoEditor`

Para saber o nome de usuário basta colocar `git config --global user.name`, de modo similar com os outros comando. Caso queira saber tudo basta: `git config --list`

## Seção 3: Essencial do Git

### Inicializando um repositório

Usará o comando de criar pasta: `mkdir`

Exemplo: `mkdir git-course`

Para fazer o repositório parte do ecossistema do git precisa usar o comando ***git init*** dentro da pasta em questão, a partir desse comando todas mudanças serão acompanhadas. Aparecerá uma pasta `.git`. Pode usar o comando `ls-la` e verá também. Dentro do diretório git verá que tem algumas pastas onde: temos `config` (guarda a configuração do repositório), `HEAD` (qual branch padrão), `branches` (branches existentes), `description`, `hooks` (gatilhos para fazer algumas ações).

## Usando o editor do terminal

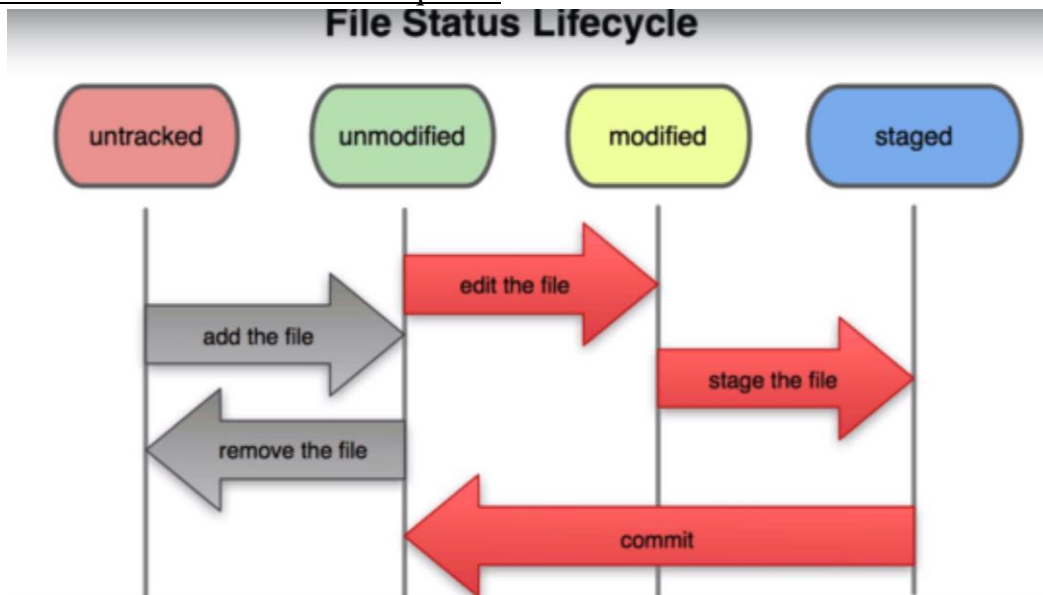
Pode usar um editor de texto separado, assim o terminal entra apenas na parte do git.

Para abrir um arquivo direto no terminal pode usar: `vi NomeDoArquivo.md` ou

usar `vim NomeDoArquivo.md`

Para inserir texto basta aperta a letra **i**, depois de escrever tudo aperta a tecla **esc** (sai do modo de inserção), **:** (indica que vai iniciar comando), **w** (indica que é para escrever e salvar) e **q** (indica sair), no fim aperta **enter**.

## O ciclo de vida dos status de seus arquivos



Comando para reportar como está o repositório no momento: `git status`

Provavelmente não vai aparecer nada para “commitar,” significa que não foi lançado para edição. Se houve alguma edição no arquivo, quando dar o comando `git status` vai aparecer o arquivo em untracked. Assim, precisa dar o comando `git add NomeDoArquivo`. Assim, o arquivo aparecerá na lista de committed, daí será criada uma versão. Caso tenha alguma modificação no arquivo, então precisa aplicar o comando `git add NomeDoArquivo` de novo para que as modificações sejam levadas para o committed. Finalmente, quando terminar tudo pode aplicar o comando `git commit -m “Add NomeDoArquivo”` para versionar/salvar as modificações, isso quer dizer que ele vai pegar todos os arquivos do repositório e criará uma imagem dele (snapshot). É uma boa prática colocar algum comentário das modificações feitas. Pois através dos logs as pessoas conseguem se localizar melhor.

Vai reparar que ele vai criar um commit segundo o nome que você dará no branch atual e uma hash criada no momento do comando, que é o número que fica ao lado do comentário feito. O hash é uma identificação do versionamento, nunca será repetido. Depois dá um descritivo do que foi alterado.

## Visualizando logs

Quando você deseja ver o histórico dos arquivos criados e modificados é interessante conhecer: *git log*, ele mostra o autor e data das modificações.

*git log --decorate*, mostra informações de qual branch para qual branch.

Pode fazer filtragens de acordo com o autor: *git log --author="nome"*

*git shortlog*, mostra para gente, em ordem alfabética, quais foram os autores, quantos commits fizeram e quais foram. Se quiser ver apenas a quantidade de commit e nome, basta: *git shortlog -sn*.

*git log --graph*, mostra de forma gráfica o que aconteceu com o branch e as versões.

Com o *git show hash* (*hash*: é o número do commit), você consegue ver os detalhes das modificações do arquivo.

## Visualizando o diff

Serve para ver as mudanças antes de enviar para o commit.

*git diff*, detalha as mudanças que houve no arquivo. É interessante fazer o diff antes do commit como uma forma de inspeção.

Dentro do *git diff*, temos a opção do name only, para dizer apenas o nome do arquivo que foi modificado. O comando: *git diff --name-only*. Ele vai listar.

Quando for comitar um arquivo que já existiu pode usar o comando: *git commit -am "comentário"*

## Desfazendo coisas

Resetar informações ou voltar no tempo. O comando *git checkout NomeDoArquivo*, ele retorna o arquivo para antes da última edição feita. Caso tenha levado essa mudança para o stage (lista do committed), pode aplicar o comando: *git reset HEAD NomeDoArquivo*.

Caso tenha dado um commit errado, pode voltar pelos comandos:

*git reset --soft HASH*, ele tira do commit e deixa no estágio de commit;

*git reset --mixed HASH*, ele tira do commit e deixa no estágio modified

*git reset --hard HASH*, ele vai ignorar a existência desse commit e deletar tudo que foi feito do commit.

Obs: precisa escolher a HASH que quer retornar.

## Seção 4: Repositórios Remotos

É uma nuvem onde vai ficar armazenado seus códigos, arquivos, etc.

### Criando e adicionando uma chave SSH

SSH é um protocolo que serve para autenticar um usuário remoto ao servidor. Existe uma chave privada e uma chave pública. O link detalha mais sobre o assunto:

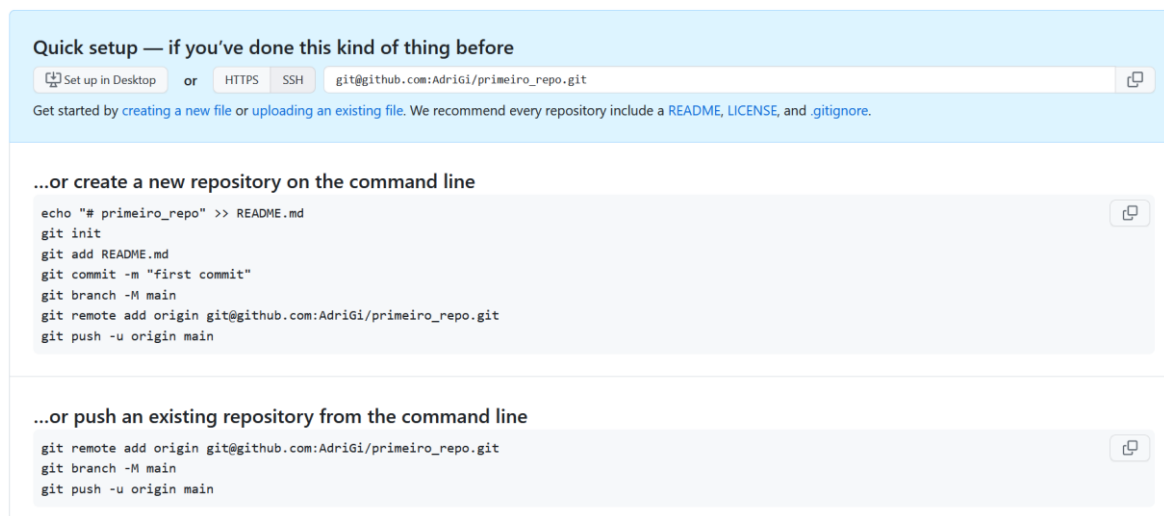
<https://docs.github.com/en/authentication/connecting-to-github-with-ssh>

No link mostra como criar e adicionar uma chave. A chave fica oculta, então precisa acessar via `cd ~/.ssh/`

Se der um comando `ls`, vai aparecer um arquivo `id_rsa.pub`. Para pegar a chave basta dar o comando `cat id_rsa.pub` ou `more id_rsa.pub` ou abrir em algum editor de texto. Para passar a chave no Github, vai em settings, procura SSH and GPG Keys, clica em New SSH key.

### Ligando repositório local a um remoto

Pode ir até o seu repositório e lá dá uma lista dos passos que você precisa fazer para criar/ligar um repositório. Por exemplo:



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

Set up in Desktop or HTTPS SSH `git@github.com:AdriGi/primeiro_repo.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 "# primeiro_repo" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:AdriGi/primeiro_repo.git
git push -u origin main
```

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

```
git remote add origin git@github.com:AdriGi/primeiro_repo.git
git branch -M main
git push -u origin main
```

O *origin* é apenas um nome, eu poderia mudar se quisesse.

O comando: `git remote`, mostra o repositório existente e o comando `git remote -v` vai mostrar os endereços.

O comando `git push -u origin master` envia os arquivos que tem, os logs e as modificações. O `u` serve para não precisar repetir todo o comando depois, os próximos comandos mostram da onde sai e pra onde vai. Por exemplo vai sair do master (a pasta atual) e ir para o origin (a nuvem no git).

## Enviando mudanças para um repositório remoto

Para enviar as mudanças usar o comando: *git push origin master*, onde origin é o nome do repositório, como foi falado, pode ser qualquer um. E o branch é o máster.

## Clonando repositórios remotos

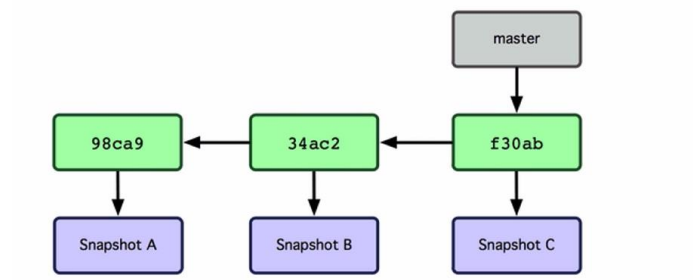
Clones de repositório é feito via comando *git clone link repositório (nome)*, onde em (nome) você pode dar qualquer nome. Útil quando está em uma máquina diferente e você deseja trabalhar no mesmo repositório.

## Fazendo um fork de um projeto

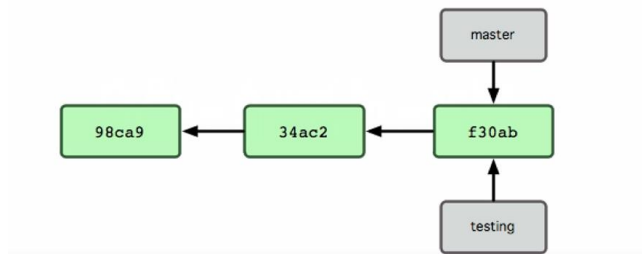
Fork pega um projeto que não é seu e faz uma cópia para você. É bom para fazer contribuições. Você faz o fork dentro da plataforma Github, a diferença dele para o clone, é que o clone você faz para arquivos que é seu e pode commitar, enquanto o fork você clona e deixa para o dono do projeto ver se pode commitar.

## **Seção 5: Ramificação (Branch)**

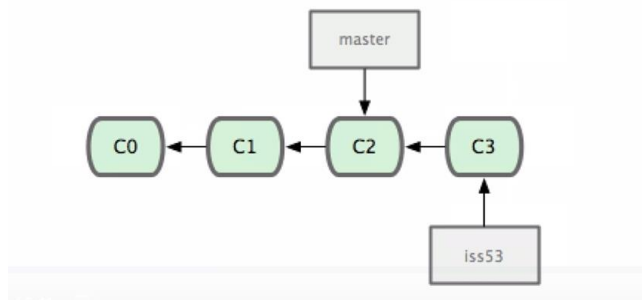
Branch é um ponteiro móvel que leva a um commit. Quando criamos um commit, cria uma HASH (conjunto de números e letras), cada HASH indica um snapshot daquele estado. Portanto o branch aponta para esse commit, assim o primeiro branch que temos se chama *master*. E o branch *master* segue o último commit dado.



É possível criar um novo branch que aponte para o mesmo commit:



Ou pode ter outro branch apontando para outro commit:



- O uso do branch serve para fazer alterações sem mudar o local principal (branch *master*);
- O branch é facilmente “desligável”, ou seja, você pode apagar rapidamente;
- É possível que várias pessoas trabalhem em diferentes branches, sem que um atrapalhe o outro;
- Por fim, evita conflitos, uma vez que cada um tem seu ambiente.

### Criando um branch

Para criar um branch, basta dar o comando:

*git checkout -b NomeDoBranch*

Para consultar quantos branches tem, basta:

*git branch* (a saída mostra a quantidade de branch, onde o asterisco indica o branch que você está usando no momento)

### Movendo e deletando branches

Para mover de um branch para outro basta:

*git checkout NomeDoBranch*

Para deletar o branch, basta:

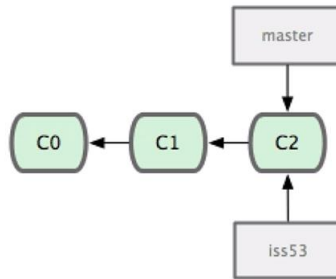


`git branch -D NomeDoBranch`

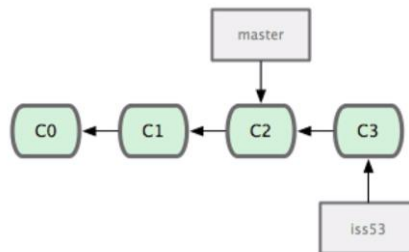
## Entendendo o merge

O merge é necessário para fazer a união do branch externo com o branch *master*. Existem duas formas de unir os branches, sendo: merge e rebase.

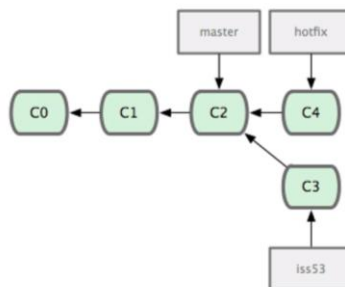
Ilustrando a situação inicial do arquivo:



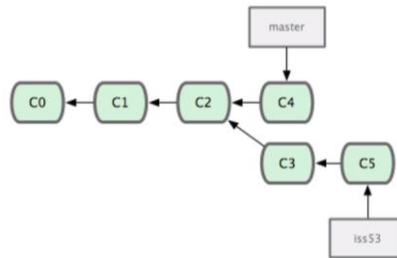
Onde temos o branch *master* e um branch externo *iss53*.



Na situação acima foi criado um novo commit (C3) com o branch externo (*iss53*). Repare que o *master* ainda continua apontando para o C2.

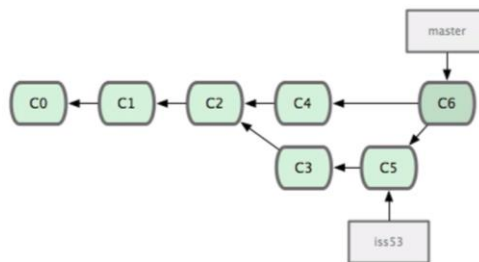


Na situação acima foi criado um novo commit a partir do branch *master* (C4).



Agora a situação mostra um commit(C5) criado a partir do branch *iss53*. Enquanto o branch *master* ficou com o último commit.

Para fazer o merge é necessário criar um novo commit (C6), como ilustrado na situação abaixo:



Nota-se que fecha um ciclo dos commit do branch externo e do *master*. Esse ciclo é chamado também de forma diamante, pois cria certas pontas.

## Pro

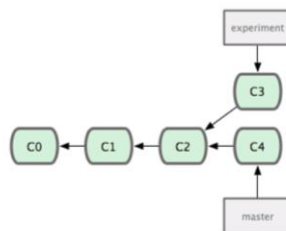
- Operação não destrutiva

## Contra

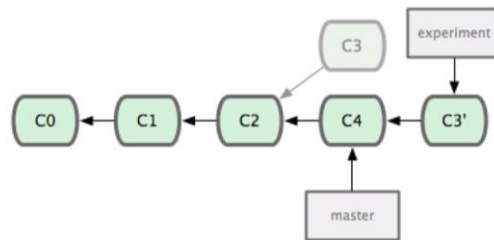
- Commit extra
- Histórico poluído

## Entendendo o rebase

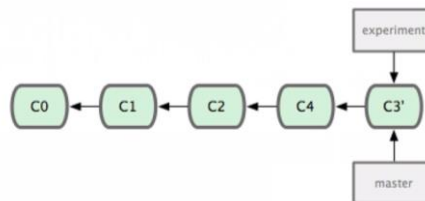
Com base no esquema inicial do merge (leia o item entendendo o merge), temos:



O rebase ele une as mudanças de forma linear, pega tudo que estava no branch separado e coloca no início da fila (fast forward):



Ao fim do processo você tem o branch externo e o branch *master* apontando para o mesmo commit.



### Pro

- Evita commits extras
- Histórico linear

### Contra

- Perde ordem cronológica

Conselho: Tem que tomar cuidado com conflito de arquivos uma vez que ele altera a ordem cronológica, assim ocasionando diferença de histórico para outra pessoa que esteja mexendo no mesmo arquivo.

## Merge e rebase na prática

### MERGE

- Crie uma pasta, sem seguida entre na pasta e inicie o git;
- Cria um arquivo foo com o vim;
- Com o git add e faça o commit;

```
reidoclash@DESKTOP-49HEDTI: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge
(Pytho3) reidoclash@DESKTOP-49HEDTI: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ vim foo
(Pytho3) reidoclash@DESKTOP-49HEDTI: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ ls
foo
(Pytho3) reidoclash@DESKTOP-49HEDTI: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ git status
On branch master

No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)
(Pytho3) reidoclash@DESKTOP-49HEDTI: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ git add foo
(Pytho3) reidoclash@DESKTOP-49HEDTI: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ git commit -m "add foo"
[master (root-commit) c288a08] add foo
 1 file changed, 1 insertion(+)
(Pytho3) reidoclash@DESKTOP-49HEDTI: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$
```

- Crie um branch test;
- Crie um arquivo bar, add e faça commit;
- Nota-se que quando dá o git log tem as duas mudanças feitas, mas quando olha pela ótica do branch máster aparece só o arquivo foo.

```
reidoclash@DESKTOP-49HEDTI: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge
(Pytho3) reidoclash@DESKTOP-49HEDTI: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ git checkout -b test
Switched to a new branch 'test'
(Pytho3) reidoclash@DESKTOP-49HEDTI: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ vim bar
(Pytho3) reidoclash@DESKTOP-49HEDTI: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ git add bar
(Pytho3) reidoclash@DESKTOP-49HEDTI: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ git commit -m "add bar"
[test de979f3] add bar
 1 file changed, 1 insertion(+)
 create mode 100644 bar
(Pytho3) reidoclash@DESKTOP-49HEDTI: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ git log
commit de679f3f6386e08097cae8da083f348fa3026ef (HEAD -> test)
Author: Adriano Giangardi <juniorhenrique96@hotmail.com>
Date:   Fri May 5 20:17:29 2023 -0300

    add bar

commit c288a08f7c10ecfdb1ca0b63f8bafc507b3b2d2d (master)
Author: Adriano Giangardi <juniorhenrique96@hotmail.com>
Date:   Fri May 5 20:10:03 2023 -0300

    add foo
(Pytho3) reidoclash@DESKTOP-49HEDTI: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ git checkout master
Switched to branch 'master'
(Pytho3) reidoclash@DESKTOP-49HEDTI: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ git log
commit c288a08f7c10ecfdb1ca0b63f8bafc507b3b2d2d (HEAD -> master)
Author: Adriano Giangardi <juniorhenrique96@hotmail.com>
Date:   Fri May 5 20:10:03 2023 -0300

    add foo
(Pytho3) reidoclash@DESKTOP-49HEDTI: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$
```

- Dentro do branch *master*, crie um arquivo fizz, add e faça o commit;
- Dê o git log e repare que as mudanças feitas no arquivo bar no branch test deve entrar entre o arquivo fizz e foo (de acordo com o histórico);

```
reidoclash@DESKTOP-49HEDTI: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge
(Pytho3) reidoclash@DESKTOP-49HEDTI: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ vim fizz
(Pytho3) reidoclash@DESKTOP-49HEDTI: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ git add fizz
(Pytho3) reidoclash@DESKTOP-49HEDTI: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ git commit -m "Add fizz"
[master bc26a88] Add fizz
 1 file changed, 1 insertion(+)
 create mode 100644 fizz
(Pytho3) reidoclash@DESKTOP-49HEDTI: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ git log
commit bc26a887e989c594d08c165cac8a8c146df1177 (HEAD -> master)
Author: Adriano Giangardi <juniorhenrique96@hotmail.com>
Date:   Fri May 5 20:22:10 2023 -0300

    Add fizz

commit c288a08f7c10ecfdb1ca0b63f8bafc507b3b2d2d
Author: Adriano Giangardi <juniorhenrique96@hotmail.com>
Date:   Fri May 5 20:10:03 2023 -0300

    add foo

add foo
```

- Faça o merge do branch test com o *master*, veja as mudanças e salve;

```
reidoclash@DESKTOP-49HEDTI: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge
(Python3) reidoclash@DESKTOP-49HEDTI: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ git merge test
Merge made by the 'ort' strategy.
 bar | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 bar
(Python3) reidoclash@DESKTOP-49HEDTI: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$
```

- Dê o git log e veja as mudanças, repare que houve a existência de um commit novo para adicionar o arquivo bar (add bar);
- Pode dar o comando git log –graph, e repare na estrutura de ciclo, apesar do histórico linear;

```
reidoclash@DESKTOP-49HEDTI: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge
(Python3) reidoclash@DESKTOP-49HEDTI: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ git log
commit 128f49d745ca3057c51643cc9b57ee3201400ee3 (HEAD -> master)
Merge: bc26a88 de679f3
Author: Adriano Giangardi <juniorhenrique96@hotmail.com>
Date: Fri May 5 20:26:47 2023 -0300

    Merge branch 'test'

commit bc26a887e989c594d08c165cac8a8c146df1177
Author: Adriano Giangardi <juniorhenrique96@hotmail.com>
Date: Fri May 5 20:22:10 2023 -0300

    Add fizz

commit de679f3f66386e00097cae8da083f348fa3026ef (test)
Author: Adriano Giangardi <juniorhenrique96@hotmail.com>
Date: Fri May 5 20:17:29 2023 -0300

    add bar

commit c288a08f7c10ecfdb1ca0b63f8bafc507b3b2d2d
Author: Adriano Giangardi <juniorhenrique96@hotmail.com>
Date: Fri May 5 20:10:03 2023 -0300

    add foo
(Python3) reidoclash@DESKTOP-49HEDTI: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$
```

## REBASE

- Crie um arquivo buz, add e faça o commit;
- Dê um log e veja;

```
reidoclash@DESKTOP-49HEDT1: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge
(Python3) reidoclash@DESKTOP-49HEDT1: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ vim buz
(Python3) reidoclash@DESKTOP-49HEDT1: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ git add buz
(Python3) reidoclash@DESKTOP-49HEDT1: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ git commit -m "add buzz"
[master 114abce] add buzz
1 file changed, 3 insertions(+)
create mode 100644 buz
(Python3) reidoclash@DESKTOP-49HEDT1: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ git log
commit 114abcecdad2d63f72f3918a38c7befd98e95b7d (HEAD -> master)
Author: Adriano Giangliardi <juniorhenrique96@hotmail.com>
Date: Fri May 5 21:08:51 2023 -0300

    add buz

commit 128f49d745ca3057c51643cc9b57ee3201400ee3
Merge: bc26a88 de679f3
Author: Adriano Giangliardi <juniorhenrique96@hotmail.com>
Date: Fri May 5 20:26:47 2023 -0300

    Merge branch 'test'

commit bc26a887e989c594d08c165cac8a8c146df1177
Author: Adriano Giangliardi <juniorhenrique96@hotmail.com>
Date: Fri May 5 20:22:10 2023 -0300

    Add fizz

commit de679f3f66386e08097cae8da083f348fa3026ef (test)
Author: Adriano Giangliardi <juniorhenrique96@hotmail.com>
Date: Fri May 5 20:17:29 2023 -0300

    add bar

commit c288a08f7c10ecfdb1ca0b63f8bafc507b3b2d2d
Author: Adriano Giangliardi <juniorhenrique96@hotmail.com>
Date: Fri May 5 20:10:03 2023 -0300

    add foo
(Python3) reidoclash@DESKTOP-49HEDT1: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$
```

- Crie um novo branch chamado rebase-branch;
- Crie um arquivo chamado bla, add e faça o commit;
- Veja o log, depois veja com o graph e perceberá que ainda estará linear;
- Caso dê um git log pelo branch *master* o arquivo do rebase-branch não vai aparecer no histórico;

```
reidoclash@DESKTOP-49HEDT1: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge
(Python3) reidoclash@DESKTOP-49HEDT1: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ git checkout -b rebase-branch
Switched to a new branch 'rebase-branch'
(Python3) reidoclash@DESKTOP-49HEDT1: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ vim bla
(Python3) reidoclash@DESKTOP-49HEDT1: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ git add bla
(Python3) reidoclash@DESKTOP-49HEDT1: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ git commit -m "add bla"
[rebase-branch 3e5ebca] add bla
1 file changed, 1 insertion(+)
create mode 100644 bla
(Python3) reidoclash@DESKTOP-49HEDT1: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ git log
commit 3e5ebcafa520bc14a8973a146dd3eb23458b75e (HEAD -> rebase-branch)
Author: Adriano Giangliardi <juniorhenrique96@hotmail.com>
Date: Fri May 5 21:12:24 2023 -0300

    add bla

commit 114abcecdad2d63f72f3918a38c7befd98e95b7d (master)
Author: Adriano Giangliardi <juniorhenrique96@hotmail.com>
Date: Fri May 5 21:08:51 2023 -0300

    add buz

commit 128f49d745ca3057c51643cc9b57ee3201400ee3
Merge: bc26a88 de679f3
Author: Adriano Giangliardi <juniorhenrique96@hotmail.com>
Date: Fri May 5 20:26:47 2023 -0300

    Merge branch 'test'

commit bc26a887e989c594d08c165cac8a8c146df1177
Author: Adriano Giangliardi <juniorhenrique96@hotmail.com>
Date: Fri May 5 20:22:10 2023 -0300

    Add fizz

commit de679f3f66386e08097cae8da083f348fa3026ef (test)
Author: Adriano Giangliardi <juniorhenrique96@hotmail.com>
Date: Fri May 5 20:17:29 2023 -0300

    add bar

commit c288a08f7c10ecfdb1ca0b63f8bafc507b3b2d2d
Author: Adriano Giangliardi <juniorhenrique96@hotmail.com>
Date: Fri May 5 20:10:03 2023 -0300

    add foo
(Python3) reidoclash@DESKTOP-49HEDT1: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$
```

No branch master crie um arquivo chamado sei la, add e faça o commit;

Dê o log, nota-se que entre o buzz e o seila deve ter o arquivo bla do branch do rebase;

```
reidoclash@DESKTOP-49HEDT1: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge
(Python3) reidoclash@DESKTOP-49HEDT1: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ vim seila
(Python3) reidoclash@DESKTOP-49HEDT1: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ git add.
git: 'add.' is not a git command. See 'git --help'.

The most similar command is
  add
(Python3) reidoclash@DESKTOP-49HEDT1: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ git add .
(Python3) reidoclash@DESKTOP-49HEDT1: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ git commit -m "add seila"
[master e79d60b] add seila
1 file changed, 1 insertion(+)
create mode 100644 seila
(Python3) reidoclash@DESKTOP-49HEDT1: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ git log
commit e79d60b95df15ce1301c2ee3e346470a59f65c74 (HEAD -> master)
Author: Adriano Gianiardi <juniorhenrique96@hotmail.com>
Date:   Fri May 5 21:18:27 2023 -0300

    add seila

commit 114abccdae2d63f72f3918a38c7befd90e05b7d
Author: Adriano Gianiardi <juniorhenrique96@hotmail.com>
Date:   Fri May 5 21:00:51 2023 -0300

    add buzz

commit 128f49d745ca3057c51643cc9b57ee3201400ee3
Merge: bc26a88 de679f3
Author: Adriano Gianiardi <juniorhenrique96@hotmail.com>
Date:   Fri May 5 20:26:47 2023 -0300

    Merge branch 'test'

commit bc26a8887e989c594d08c165cac8a8c146df1177
Author: Adriano Gianiardi <juniorhenrique96@hotmail.com>
Date:   Fri May 5 20:22:10 2023 -0300

    Add fizz

commit de679f3f66186e00997cae8da081f348fa3026ef (test)
Author: Adriano Gianiardi <juniorhenrique96@hotmail.com>
Date:   Fri May 5 20:17:29 2023 -0300

    add bar

commit c28a08f7c10ecfd81ca0b63f8baf507b3b2d2d
Author: Adriano Gianiardi <juniorhenrique96@hotmail.com>
Date:   Fri May 5 20:10:03 2023 -0300

    add foo
(Python3) reidoclash@DESKTOP-49HEDT1: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$
```

- Com o comando `git rebase NomeDoBranch`;
- Dê um log ou log --graph, nota-se que continuará linear.

```
reidoclash@DESKTOP-49HEDT1: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge
The most similar command is
  rebase
(Python3) reidoclash@DESKTOP-49HEDT1: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ git rebase rebase-branch
Successfully rebased and updated refs/heads/master.
(Python3) reidoclash@DESKTOP-49HEDT1: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ git log
commit b5d2a80f10ea32e545032670c91408c0b1a89 (HEAD -> master)
Author: Adriano Gianiardi <juniorhenrique96@hotmail.com>
Date:   Fri May 5 21:18:27 2023 -0300

    add seila

commit 3e5ec8a6a5528bc14a8973a146dd3eb23450b75e (rebase-branch)
Author: Adriano Gianiardi <juniorhenrique96@hotmail.com>
Date:   Fri May 5 21:12:24 2023 -0300

    add bla

commit 114abccdae2d63f72f3918a38c7befd90e05b7d
Author: Adriano Gianiardi <juniorhenrique96@hotmail.com>
Date:   Fri May 5 21:00:51 2023 -0300

    add buzz

commit 128f49d745ca3057c51643cc9b57ee3201400ee3
Merge: bc26a88 de679f3
Author: Adriano Gianiardi <juniorhenrique96@hotmail.com>
Date:   Fri May 5 20:26:47 2023 -0300

    Merge branch 'test'

commit bc26a8887e989c594d08c165cac8a8c146df1177
Author: Adriano Gianiardi <juniorhenrique96@hotmail.com>
Date:   Fri May 5 20:22:10 2023 -0300

    Add fizz

commit de679f3f66186e00997cae8da081f348fa3026ef (test)
Author: Adriano Gianiardi <juniorhenrique96@hotmail.com>
Date:   Fri May 5 20:17:29 2023 -0300

    add bar

commit c28a08f7c10ecfd81ca0b63f8baf507b3b2d2d
Author: Adriano Gianiardi <juniorhenrique96@hotmail.com>
Date:   Fri May 5 20:10:03 2023 -0300

    add foo
(Python3) reidoclash@DESKTOP-49HEDT1: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$
```

```
reidoclash@DESKTOP-49HEDT: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge
commit c288a0f7c10ecfdb1ca0b63f8bafc507b3b2d2d
Author: Adriano Giangliardi <juniorhenrique96@hotmail.com>
Date: Fri May 5 20:10:03 2023 -0300

    add foo

(Pytho3) reidoclash@DESKTOP-49HEDT: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$ git log --graph
* commit b5d2a80f10ea8a32e845032670c91408c0fb1a89 (HEAD -> master)
  Author: Adriano Giangliardi <juniorhenrique96@hotmail.com>
  Date: Fri May 5 21:18:27 2023 -0300

      add seila

* commit 3e5ecba6a5528bc14a8973a146dd3eb23458b75e (rebase-branch)
  Author: Adriano Giangliardi <juniorhenrique96@hotmail.com>
  Date: Fri May 5 21:12:24 2023 -0300

      add bla

* commit 114abcecdac2d63f72f3918a38c7befd90e05b7d
  Author: Adriano Giangliardi <juniorhenrique96@hotmail.com>
  Date: Fri May 5 21:08:51 2023 -0300

      add buzz

* commit 128f49d745ca3057c51643cc9b57ee3281400ee3
  Merge: bc26a88 de679f3
  Author: Adriano Giangliardi <juniorhenrique96@hotmail.com>
  Date: Fri May 5 20:26:47 2023 -0300

      Merge branch 'test'

* commit de679f3f66386e08097cae8da083f348fa3026ef (test)
  Author: Adriano Giangliardi <juniorhenrique96@hotmail.com>
  Date: Fri May 5 20:17:29 2023 -0300

      add bar

* commit bc26a887e909c594d08c165cac8a8c146df1177
  Author: Adriano Giangliardi <juniorhenrique96@hotmail.com>
  Date: Fri May 5 20:22:10 2023 -0300

      Add fizz

* commit c288a0f7c10ecfdb1ca0b63f8bafc507b3b2d2d
  Author: Adriano Giangliardi <juniorhenrique96@hotmail.com>
  Date: Fri May 5 20:10:03 2023 -0300

      add foo

(Pytho3) reidoclash@DESKTOP-49HEDT: /mnt/c/Users/junio/Dropbox/curso_git/git-course/rebase-merge$
```

Dica: Enquanto estiver adicionando novos commit e atualizando com outros branch use o rebase. Em caso de adição de novas informações é interessante usar o merge.

## Seção 6: Extras

### Criando o .gitignore

Comando usado quando tem arquivos incluído no repositório e você deseja que isso não seja compartilhado.

O uso é simples, primeiro você cria um arquivo usando algum editor e salva como *.gitignore*. O ponto serve para dizer que é um arquivo oculto. Dentro desse arquivo, você vai colocar as extensões ou o arquivo em específico que você deseja que não vá no commit. Por exemplo:

```
reidoclash@DESKTOP-49HEDT: /mnt/c/Users/junio/Dropbox/curso_git/git-course
(Pytho3) reidoclash@DESKTOP-49HEDT: /mnt/c/Users/junio/Dropbox/curso_git/git-course$ vi arquivo_qualquer.md
(Pytho3) reidoclash@DESKTOP-49HEDT: /mnt/c/Users/junio/Dropbox/curso_git/git-course$ ls
README.md  arquivo_qualquer.md  rebase-merge/
(Pytho3) reidoclash@DESKTOP-49HEDT: /mnt/c/Users/junio/Dropbox/curso_git/git-course$ vi .gitignore
(Pytho3) reidoclash@DESKTOP-49HEDT: /mnt/c/Users/junio/Dropbox/curso_git/git-course$ git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore
    arquivo_qualquer.md
    rebase-merge/

nothing added to commit but untracked files present (use "git add" to track)
(Pytho3) reidoclash@DESKTOP-49HEDT: /mnt/c/Users/junio/Dropbox/curso_git/git-course$
```



Repare que o arquivo *arquivo\_qualquer.md* não está aparecendo, pois dentro do *.gitignore* está especificado para ignorá-lo. Caso você queira ignorar um conjunto de extensões de arquivo, dentro do *.gitignore*, basta colocar *\*.extensão*, por exemplo: *\*.py*, *\*.xlsx*, *\*.json*, etc.

Alguns links interessantes que exploram esse comando:

<https://github.com/github/gitignore> (é uma lista de ignore)

<https://git-scm.com/docs/gitignore> (falando mais do comando)

## Git stash

Comando usado para guardar eventuais mudanças para depois fazer um commit. Pode ser usado, por exemplo, em situações que você precisa entrar em outro branch, mas não terminou suas modificações e não deseja subir as modificações para não poluir o log. Basta dar o comando:

*git stash* - Ele vai guardar essa modificação, deixando em progresso (WIP).

Nesse momento é possível sair e fazer outra coisa. Quando quiser, volte e dê o comando:

*git stash apply* - Ele retorna com os arquivos que ainda faltam add e dar o commit.

Existe também o comando *git stash list* que mostra os stash que está sendo feito. Caso queira limpar isso, basta dar o comando: *git stash clear*

## Simplificação de comando

Muitas vezes se torna exaustivo digitar todo o comando, por isso é possível simplificar algum comando, por exemplo:

- O comando *git status*, podemos pensar em apenas digitar *git s*;
- Para isso use o comando: *git config --global alias.simplificacao comando*;
- Por exemplo: *git config --global alias.s status*

## Versionamento com tags

Usado quando estamos trabalhando com bibliotecas ou sistemas muito grandes, estamos mexendo com o estado desses sistemas a cada commit dado. Então é interessante ter um marcador dessas versões, seja a cada 10 commit ou simplesmente assumir uma parte do projeto que tenha mudança suficiente para ser uma nova versão.

- Para criar uma tag, basta: *git tag* ;
- Se quiser passar uma tag com anotação: *git tag -a versão -m "anotação"* (por exemplo: *git tag -a 1.0.0 -m "arquivo finalizado"*);
- Quando quiser subir a tag para o github: *git push origin master --tags*, repare que lá no github terá uma release nova, inclusive com o código da versão da tag lançada.

## Git revert

Diferente do git reset, ele não apaga as modificações. É uma espécie de control z, é usado quando você não deseja apagar as modificações, pois nem tudo ali está errado. Mas ao mesmo tempo você não deseja corrigir no momento. O *git revert* é dado quando depois que você fez o commit, usando o comando:

*git revert HASH* (precisa buscar a hash do commit em questão)

Quando dar um *git log*, perceberá que o comando *git revert* cria um novo commit, e pode pegar a hash desse novo commit e ver as mudanças que ocorreram usando o comando *git show HASH*. Perceberá que ele retornou para o estado anterior à mudança, mas não apagou as mudanças que foram feitas, pois o commit “estragado” ainda estará lá (This reverts commit Hash).

## Apagando tags e branches remotos

Para deletar tag, pode usar o comando:

*g tag -d NumeroDaTag* – porém isso apaga apenas localmente.

Para apagar remotamente, usa o comando:

*git push origin : NumeroDaTag*

O mesmo vale para o branch:

*git push origin :NomeDoBranch*