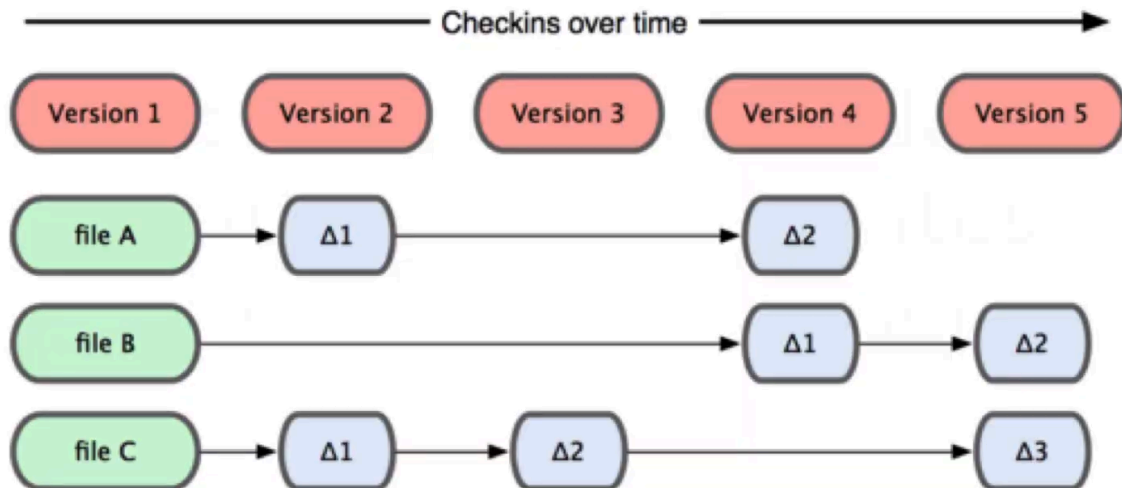


## Relatório 26 - Prática: Git e Github para Iniciantes (III)

Lucas Scheffer Hundsdorfer

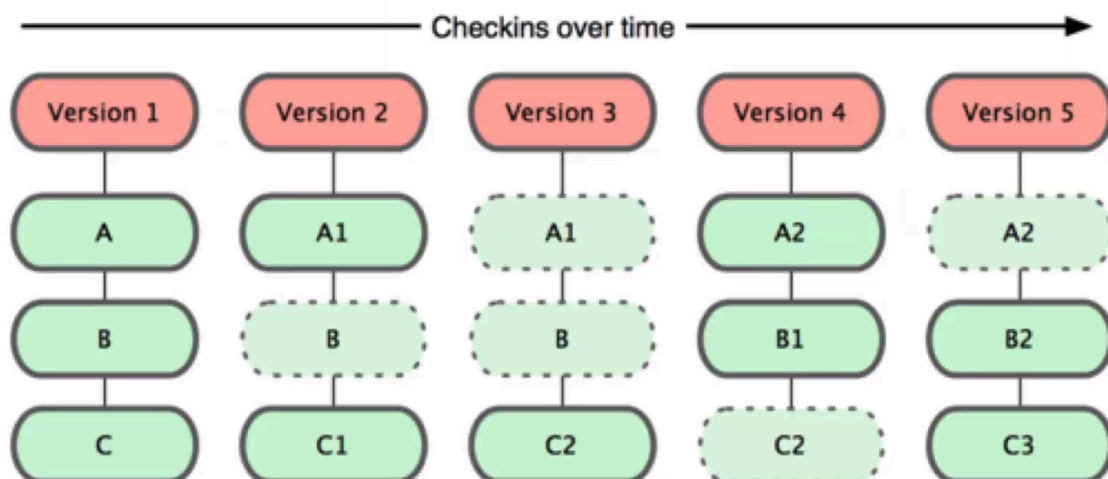
### Descrição da atividade

O curso começa explicando a importância do git para o controle de versões, o exemplo dado é que outros controladores de versões fazem assim:



Na versão 1 foi salvo os arquivos A, B, C, porém o arquivo só vai ser salvo de novo se ele for alterado, que nem foi na versão 2 com os arquivos A e C.

Já o git age de maneira diferente:



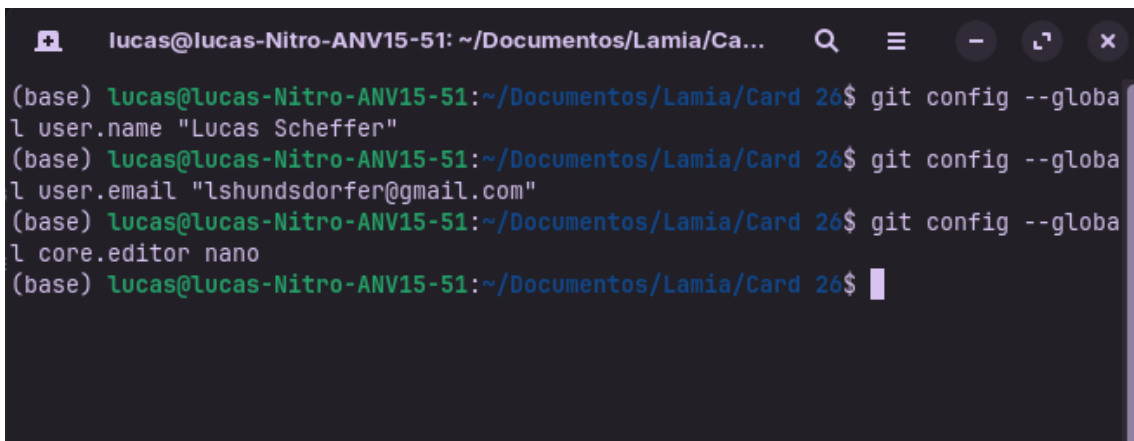
Aqui cada versão leva as versões correspondentes dos arquivos.

A origem do git veio de uma empresa chamada BitKeeper onde eles apresentavam uma ferramenta de controle de versão que hospedava o kernel do linux, porém com alguns acontecidos o criador do Linux o Linus Torvald

decidiu criar um software de controle de versões melhorado, com mais agilidade e sendo capaz de lidar com desenvolvimento não linear em projetos grandes como o próprio linux, assim nasceu o Git.

Git é a ferramenta que você instala e usa para controlar as versões do seu código, enquanto o GitHub é a plataforma online onde você armazena seus projetos e colabora com outras pessoas. Não se utiliza o GitHub sem o Git, mas o Git pode ser utilizado sem depender de qualquer plataforma de hospedagem.

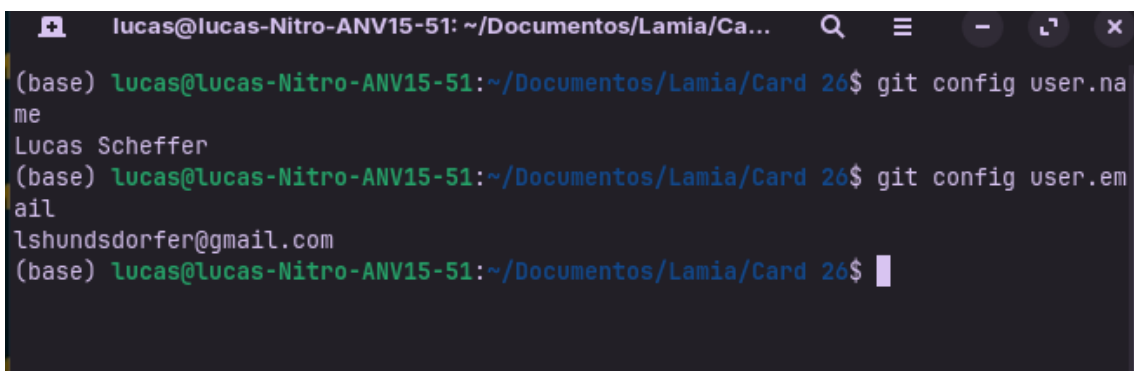
Após a instalação do git ele ensina a configuração padrão, que seria a do username e do email:

A terminal window with a dark background. The title bar shows 'lucas@lucas-Nitro-ANV15-51: ~/Documentos/Lamia/Ca...'. The prompt is '(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26\$'. The user enters 'git config --global user.name "Lucas Scheffer"', followed by 'git config --global user.email "lshundsdorfer@gmail.com"', and finally 'git config --global core.editor nano'. The prompt returns after each command.

```
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26$ git config --global user.name "Lucas Scheffer"
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26$ git config --global user.email "lshundsdorfer@gmail.com"
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26$ git config --global core.editor nano
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26$
```

No caso ele configurou qual editor ele utiliza como padrão, mas não é necessário.

Confirmação da configuração:

A terminal window with a dark background. The title bar shows 'lucas@lucas-Nitro-ANV15-51: ~/Documentos/Lamia/Ca...'. The prompt is '(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26\$'. The user enters 'git config user.name' and 'Lucas Scheffer' on the next line. Then the user enters 'git config user.email' and 'lshundsdorfer@gmail.com' on the next line. The prompt returns after each command.

```
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26$ git config user.name
Lucas Scheffer
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26$ git config user.email
lshundsdorfer@gmail.com
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26$
```

Após a configuração já começa a explicação de como inicializar um repositório:

```
lucas@lucas-Nitro-ANV15-51: ~/Documentos/Lamia/Card 26/git-cou...
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26$ mkdir git-course
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26$ cd git-course/
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26/git-course$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
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:   git branch -m <name>
Repositório vazio Git inicializado em /home/lucas/Documentos/Lamia/Card 26/git-course/.git
/
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26/git-course$ ls -la
total 12
drwxrwxr-x 3 lucas lucas 4096 set  8 12:56 .
drwxrwxr-x 3 lucas lucas 4096 set  8 12:56 ..
drwxrwxr-x 7 lucas lucas 4096 set  8 12:56 .git
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26/git-course$
```

Primeiramente é criada a pasta que desejamos iniciar o repositório, e entrada nela e dado o primeiro comando git 'git init', com esse comando é criado um diretório .git dentro da pasta que havia sido criada com os seguintes elementos:

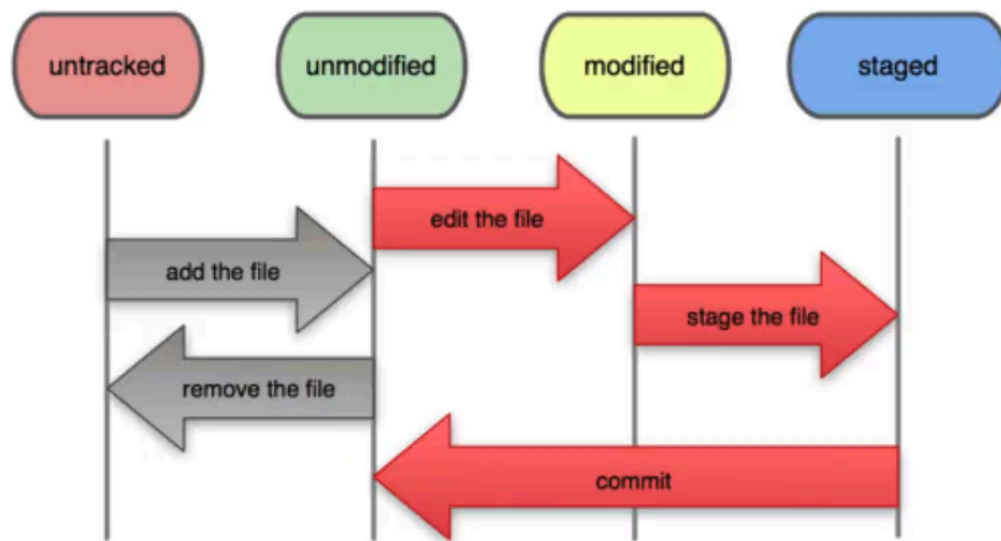
```
lucas@lucas-Nitro-ANV15-51: ~/Documentos/Lamia/Card 26/git-cou...
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26/git-course$ cd .git/
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26/git-course/.git$ ls
branches  config  description  HEAD  hooks  info  objects  refs
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26/git-course/.git$
```

Basicamente todos para a configuração e descrição do repositório.

Antes de começar a ensinar o git propriamente ele passa uma breve explicação de como usa o editor de texto dele, o vim, que é basicamente um editor de texto direto do terminal. Porém não é necessário ele deixa bem claro que se pode utilizar o editor que quiser.

Dentro do git existe o ciclo de vida dos status dos arquivos:

## File Status Lifecycle



Untracked: É quando o arquivo foi adicionado mas ainda não foi visto pelo git.

Unmodified: Quando o git já viu ele porém não foi modificado.

Modified: A partir de qualquer edição ele fica como modificado.

Staged: É o que ele vai manter salva a versão dos arquivos, quando você commit a todos os arquivos voltam para unmodified.

É importante ter o conhecimento desse ciclo de vida e dos logs para se ter noção do que está acontecendo com seus arquivos e as versões deles.

É possível ver os logs através do comando 'git log':

```
lucas@lucas-Nitro-ANV15-51: ~/Documentos/Lamia/Card 26/git-cou...
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26/git-course$ git log
commit 3fe06bf1cc29528bcb18667ec3ca183a6bf6c3dd (HEAD -> master)
Author: Lucas Scheffer <lshundsorfer@gmail.com>
Date: Mon Sep 8 13:08:34 2025 -0300

    Add testes

commit 5a818d31e7ea5c9c862c99ca1c92e6c80411666a
Author: Lucas Scheffer <lshundsorfer@gmail.com>
Date: Mon Sep 8 13:06:47 2025 -0300

    AddReadme.md
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26/git-course$
```

Dentro do log ele mostra a hash do commit, quem foi o autor do commit e o email dele a data do commit e também a mensagem passada. Utilizando um filtro para ver apenas os commits de tal autor:

```
lucas@lucas-Nitro-ANV15-51: ~/Documentos/Lamia/Card 26/git-cou...
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26/git-course$ git log --author="Lucas"
commit 3fe06bf1cc29528bcb18667ec3ca183a6bf6c3dd (HEAD -> master)
Author: Lucas Scheffer <lshundsdorfer@gmail.com>
Date: Mon Sep 8 13:08:34 2025 -0300

    Add testes

commit 5a818d31e7ea5c9c862c99ca1c92e6c80411666a
Author: Lucas Scheffer <lshundsdorfer@gmail.com>
Date: Mon Sep 8 13:06:47 2025 -0300

    AddReadme.md
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26/git-course$
```

Ou mostrar os shorts logs onde filtra por apenas o nome do autor dos commits e quantos commits fizeram e as mensagens.

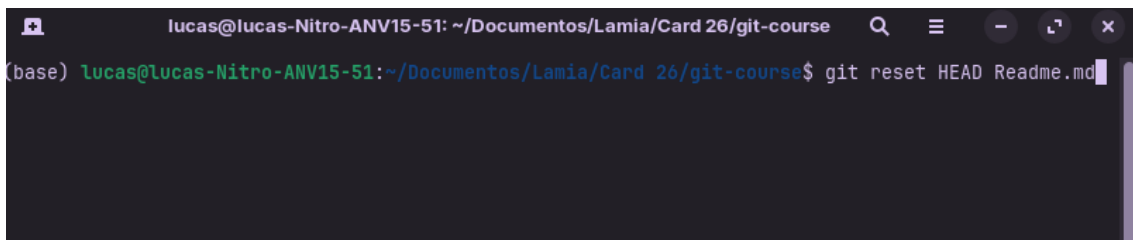
Além de ver o que foi mudado e comitado já é possível que se veja o que foi mudado antes de dar o commit, utilizando git diff ele consegue ver o que foi alterado dentro da pasta readme:

```
lucas@lucas-Nitro-ANV15-51: ~/Documentos/Lamia/Card 26/git-cou...
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26/git-course$ git diff 5a818d31e7ea5c9c862c99ca1c92e6c80411666a
diff --git a/Readme.md b/Readme.md
index 7d2e657..106c60e 100644
--- a/Readme.md
+++ b/Readme.md
@@ -2,3 +2,4 @@

Arquivo da aula de Git e Github para iniciantes
Este é um repositório teste
+Testes
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26/git-course$
```

Importante para revisar o que foi alterado sem subir erros para a versão principal.

Com o comando 'git checkout' dá para reverter mudanças feitas ao longo do processo que não foram adicionadas ainda com o 'git add'. Porém para mudanças que já foram para o stage o que precisa ser feito é:



```
lucas@lucas-Nitro-ANV15-51: ~/Documentos/Lamia/Card 26/git-course$ git reset HEAD README.md
```

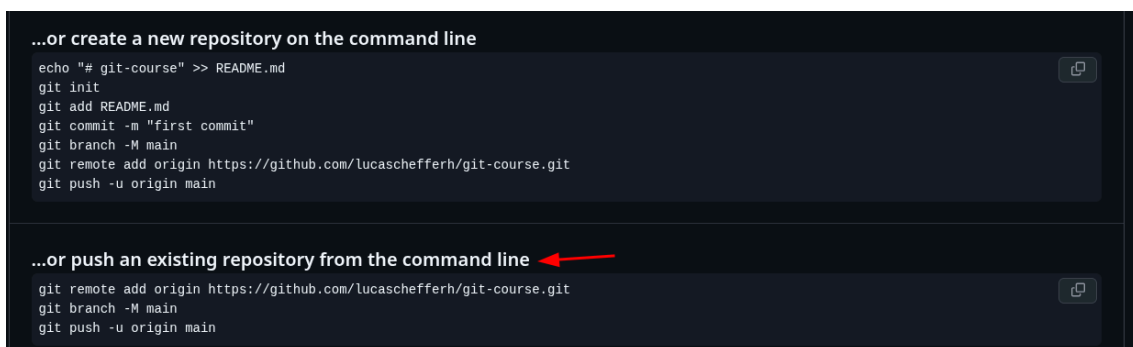
Com esse 'git reset HEAD arquivo' a adição do arquivo modificado é voltada atrás. Para desfazer um commit feito existe o 'git reset' com 3 parâmetros principais: soft, mixed e hard.

Soft ele vai reverter o commit porém a mudança do arquivo vai continuar no stage.

Mixed vai reverter o commit para o status de modificado antes de ser adicionado no stage, é o parâmetro padrão.

Hard vai ignorar a existência do commit feito, uma forma meio perigosa mas útil quando foi feito algo errado.

Em uma das aulas é ensinado a como criar um repositório remoto dentro da interface do github que é bem simples apenas clicar em criar, e o vídeo seguinte é apenas como configurar dentro do github para que ele reconheça sua chave ssh para permitir que você faça commits. Com o repositório remoto criado é necessário conectar com o seu repositório local com o remoto, a própria interface do github já te ajuda com isso:



```
...or create a new repository on the command line

echo "# git-course" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/lucaschefferh/git-course.git
git push -u origin main

...or push an existing repository from the command line
git remote add origin https://github.com/lucaschefferh/git-course.git
git branch -M main
git push -u origin main
```

Se for necessário criar um repositório local ele já dá os comandos, se o repositório já existe e só precisa conectar também dá os comandos necessários ou se você vai importar código de algum outro repositório. Com a conexão feita o que é necessário para subir as alterações do repositório local

para o remoto é um simples comando: 'git push':

```
lucas@lucas-Nitro-ANV15-51: ~/Documentos/Lamia/Card 26/git-course
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26/git-course$ git push -u origin main
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 12 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 764 bytes | 764.00 KiB/s, done.
Total 9 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/lucaschefferh/git-course.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26/git-course$
```

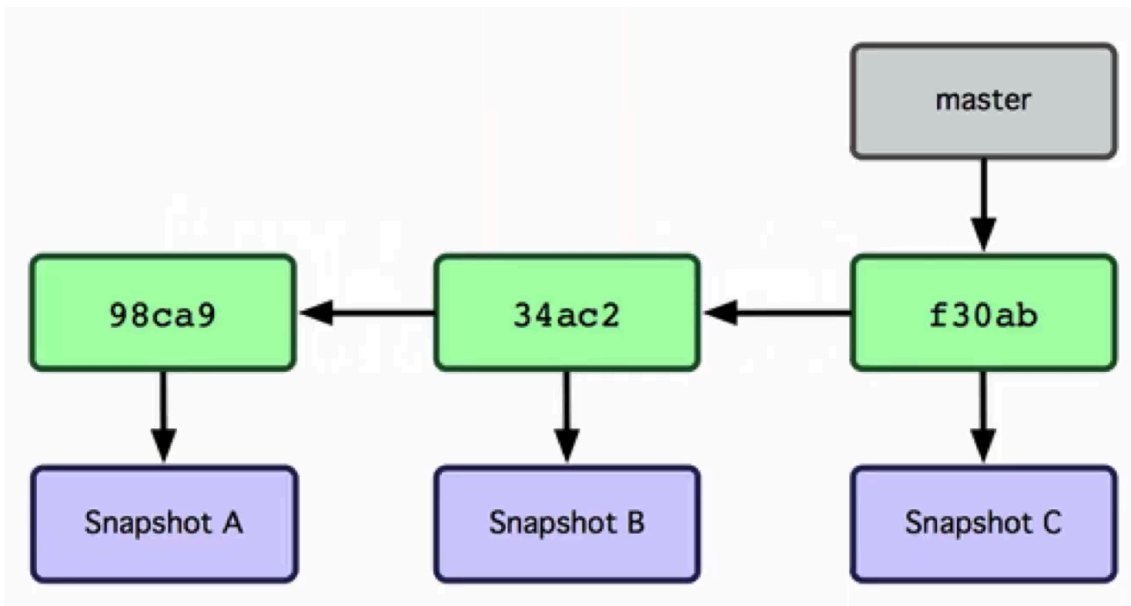
Esse 'origin' é um nome que fica normalmente como padrão, e esse master é de onde o arquivo está vindo que é a branch atual. Com o github é possível você pegar repositórios remotos públicos e clonar na sua máquina:

```
lucas@lucas-Nitro-ANV15-51: ~/Documentos/Lamia/Card 26/github-course-cl...
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26$ git clone https://github.com/lucaschefferh/git-course github-course-clone
Cloning into 'github-course-clone'...
remote: Enumerating objects: 12, done.
remote: Counting objects: 100% (12/12), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 12 (delta 3), reused 12 (delta 3), pack-reused 0 (from 0)
Receiving objects: 100% (12/12), done.
Resolving deltas: 100% (3/3), done.
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26$ ls
git-course  github-course-clone  'Relatório 26 - Lucas Scheffer.pdf'
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26$ cd github-course-clone/
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26/github-course-clone$ ls
Readme.md
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26/github-course-clone$ cat Readme.md
# Github

Arquivo da aula de Git e Github para iniciantes
Este é um repositório teste
Testes

Estou estudando os comandos Git
AAAAAAA AAAAAAAA
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26/github-course-clone$
```

O github também permite que se faça um fork de um repositório, seria parecido com um clone, porém você pode fazer as alterações que quiser e mandar um pull request para o dono do repositório, muito útil para projetos open source onde mais de uma pessoa trabalha no código. Ele se diferencia do clone porque o clone não é criado um repositório para isso e sim apenas copiado e colado os arquivos para sua máquina local.



O que uma branch faz é apontar para um commit, basicamente é isso, porém ele traz inúmeras vantagens, como poder modificar sem alterar o local principal (branch master), outro ponto é que é facilmente apagável, e é uma ferramenta primordial quando se tem várias pessoas trabalhando em um mesmo projeto evitando conflitos de código. Para criar uma nova branch é simples, basta dar 'git checkout -b {nome}' com esse comando a branch é criada e você já é levado até lá:

```
lucas@lucas-Nitro-ANV15-51: ~/Documentos/Lamia/Card 26/git-course
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26/git-course$ git checkout -b testing
Switched to a new branch 'testing'
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26/git-course$
```

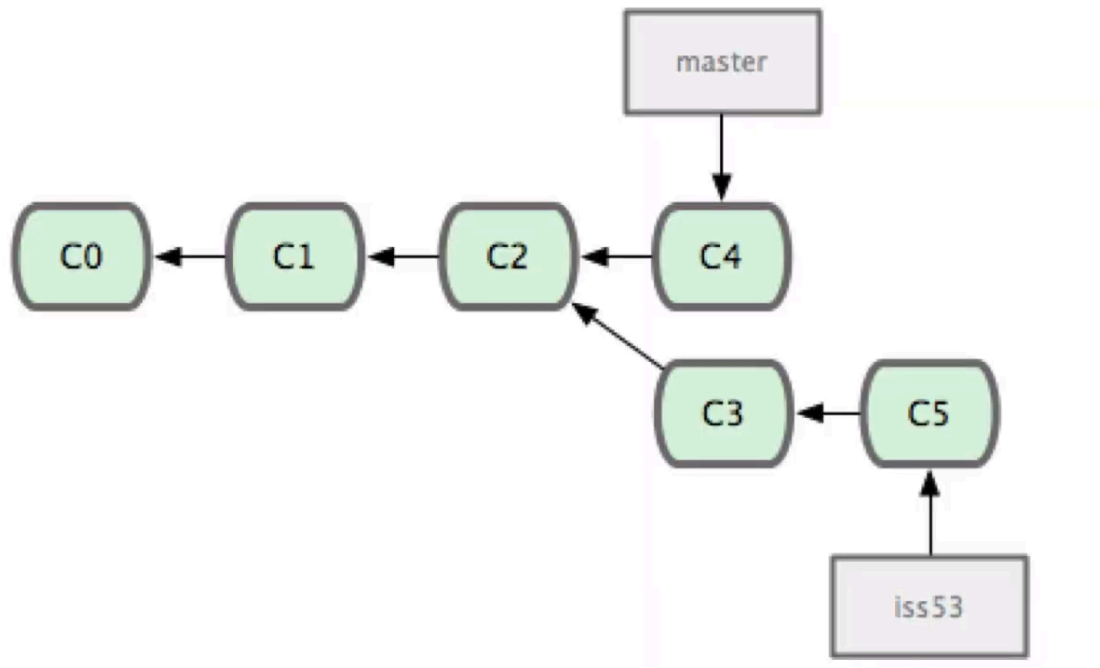
com 'git branch' o git vai listar todas as branches dentro do seu repositório, e vai destacar com um '\*' a branch onde você está. Para alterar sua branch é simples, basta dar o mesmo comando que foi dado para criar só que sem o '-b':

```
lucas@lucas-Nitro-ANV15-51: ~/Documentos/Lamia/Card 26/git-course
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26/git-course$ git branch
main
* testing
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26/git-course$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26/git-course$ git branch
* main
  testing
(base) lucas@lucas-Nitro-ANV15-51:~/Documentos/Lamia/Card 26/git-course$
```

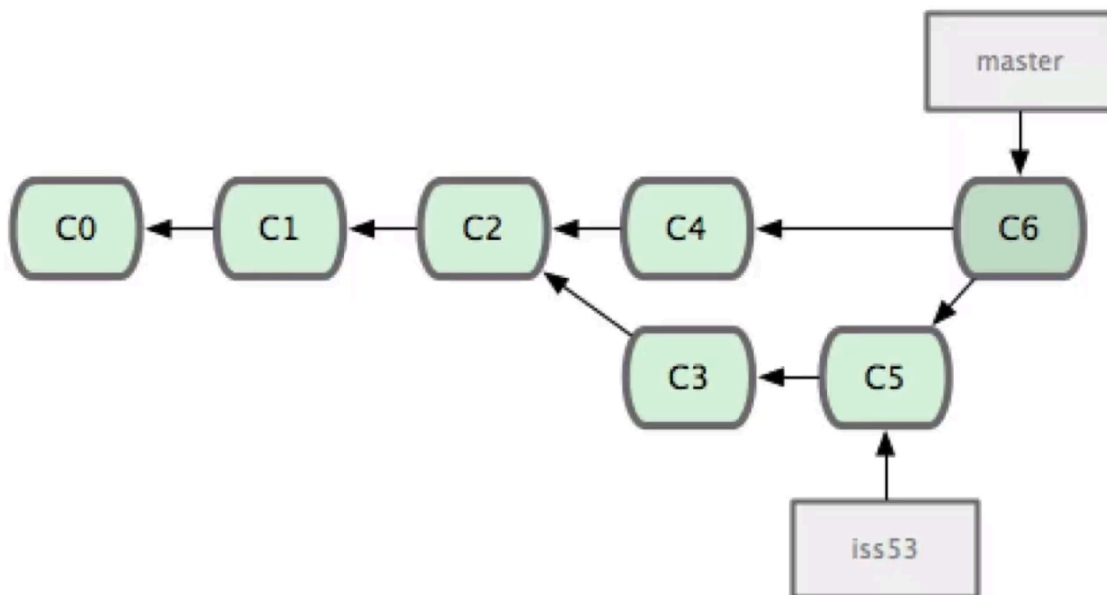
E para deletar uma branch basta dar 'git branch -D {nome}'.



Porém é necessário unir as branches para que o projeto seja finalizado e para isso dá para utilizar o dois comandos, o merge e o rebase. Começando pelo merge. Primeiramente vou mostrar o que acontece com as branches quando vão dando commits, as ramificações:

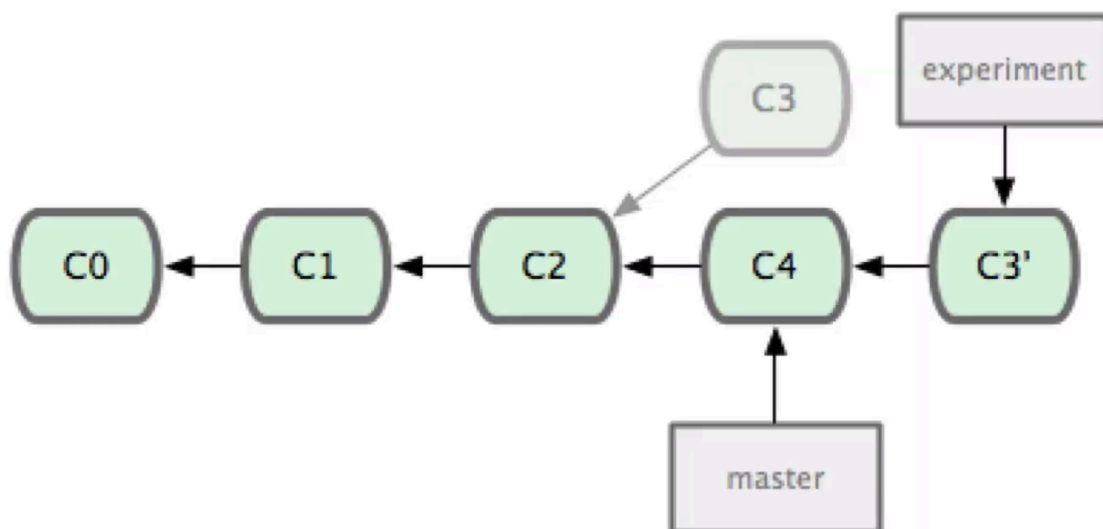


O merge vai trazer de volta a linearidade:

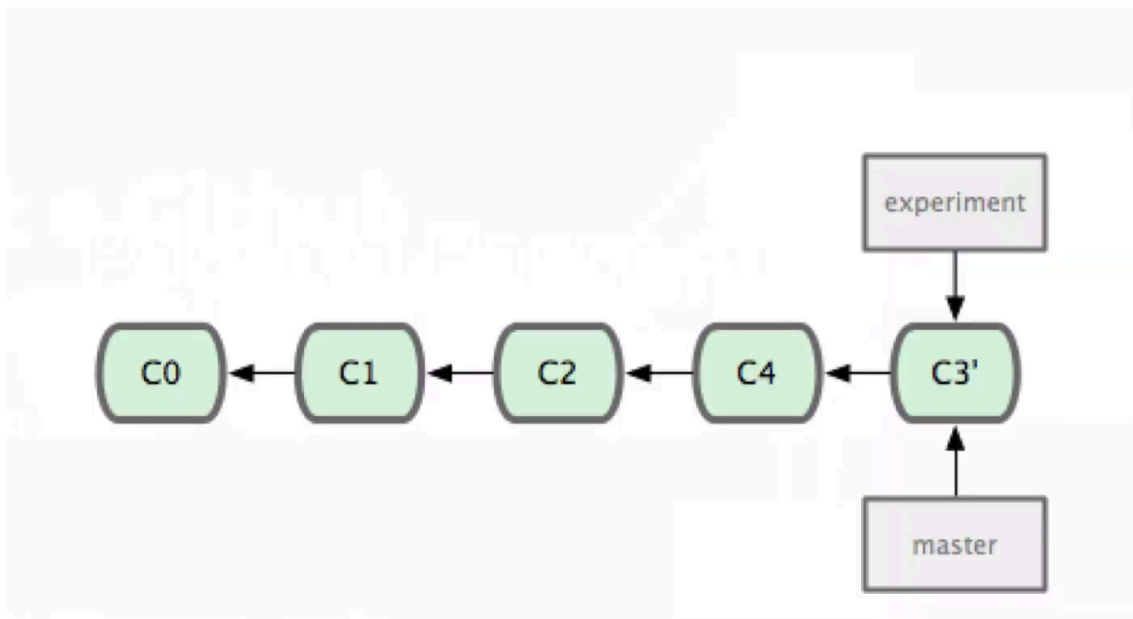


Porém é preciso fazer um commit na master que faz a junção de tudo novamente. A vantagem principal do merge é que ele não é destrutivo, não vai apagar nada que foi feito em nenhuma das branches, porém ele gera um commit a mais o que pode ser visto como poluir o histórico.

O rebase tem o mesmo intuito porém tem um processo diferente:

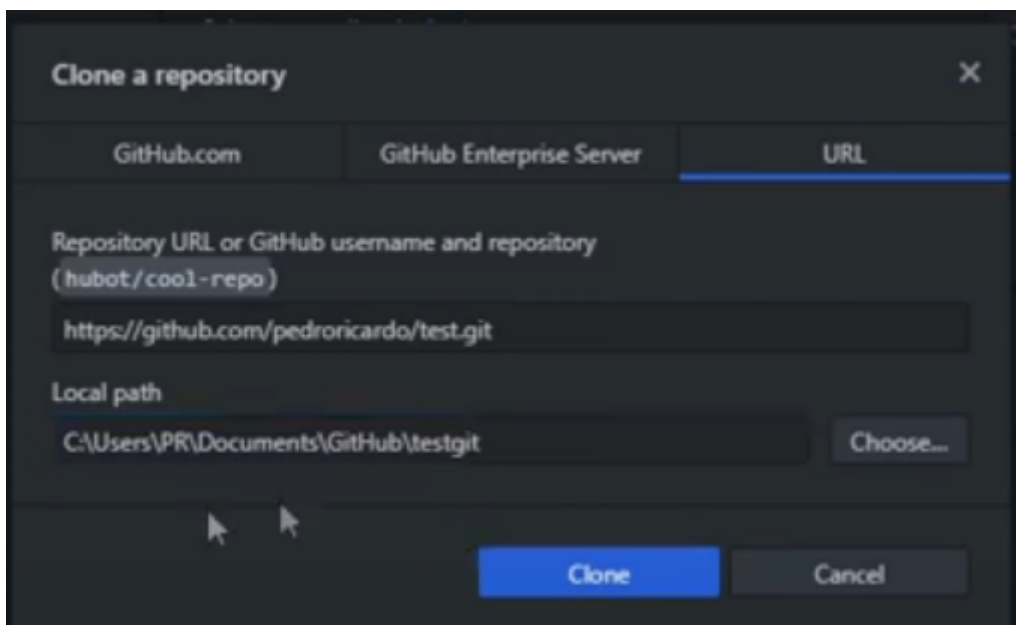


Ele basicamente destrói o commit c3 e cria uma cópia dele e coloca no começo da fila para voltar a ser linear. E no final do processo ele faz as branches apontarem para o mesmo commit:



As vantagens do rebase é não ter aquele commit extra para juntar tudo, e mantém tudo de forma linear as desvantagens é que ele perde a ordem em que foi inserido os commits.

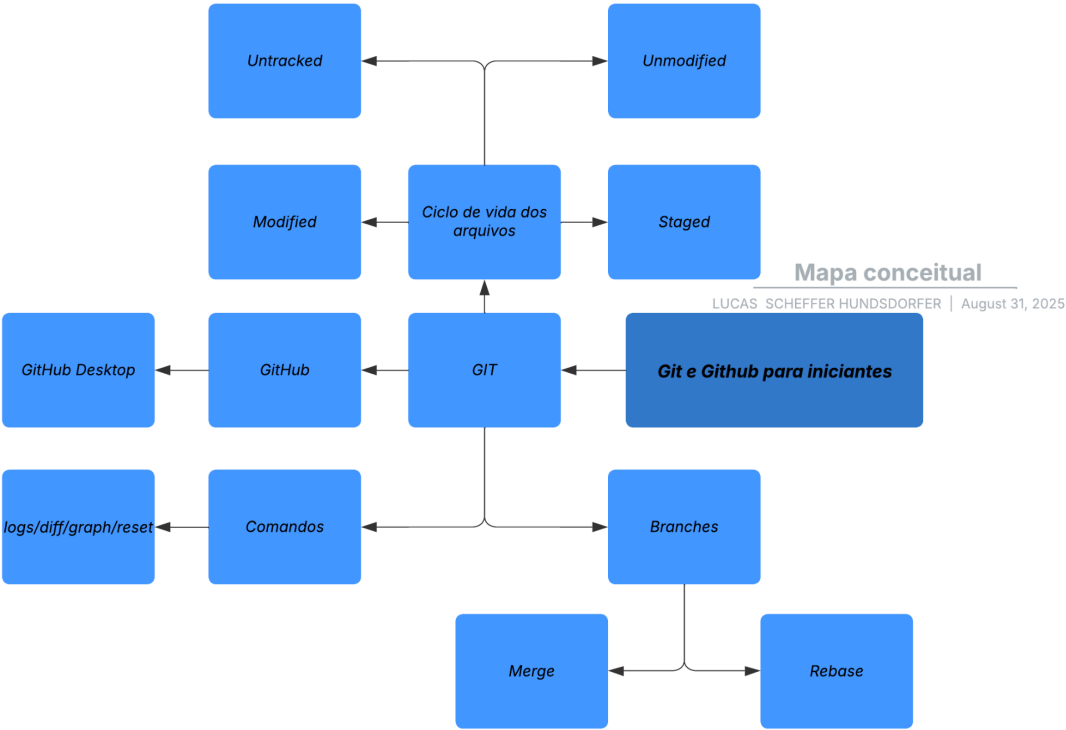
No vídeo de como utilizar o github desktop é começado apresentando a interface do programa e o que cada fração representa ali, como o histórico dos commits e as mudanças feitas onde clica para dar push para o github. Após isso ele cria um novo repositório remoto e copia o link do mesmo:



Ele clona o repositório remoto através do link e define a pasta local. E ensinado a criar o .gitignore para que faça com que o git ignore todos os arquivos que forem colocados dentro desta pasta, muito útil para repositórios públicos que possam conter informações que não podem ser vazadas. Após clonar o repositório é mostrado como ele mostra o que foi mudado dentro da pasta e

como fazer o commit e subir para o github e como fica o histórico das mudanças.

Insight visual original:



## Conclusões

Minha conclusão que fica é que o git juntamente do github são ferramentas muito poderosas para o controle de versão dos códigos, e que o domínio delas é de extrema importância para que além de não se perder em meio das branches e commits entender exatamente o que foi feito como foi feito e porque foi feito em meio a grandes projetos.

## Referências