

Usando o git

Olá, alunos(as),

Bem-vindos(as) a nossa sétima aula da disciplina “Gerência de Configuração”. Nesta primeira aula, vamos ver como fazer uma operação básica de um repositório do Git. Você vai ver como inicializar um repositório do Git, como salvar as suas primeiras alterações, fazendo os seus primeiros commits e listando quais foram os commits que foram feitos.

Assim, não se esqueça de ler esta aula, assistir aos nossos vídeos e fazer as atividades. Se tiverem alguma dúvida, vocês podem usar o nosso quadro de avisos, que eu responderei.

🔥 Bons estudos!

Objetivos de aprendizagem

Ao término desta aula, vocês serão capazes de:

- saber como iniciar um repositório do Git;
- entender como funciona o fluxo de armazenamento das alterações, com os commits;
- saber como listar os commits feitos.

Seções de estudo

- 1 – Iniciando um repositório do git
- 2 – Commitando os seus primeiros arquivos
- 3 – Ignorando arquivos com gitignore

1 - Iniciando um repositório do git

Agora que você sabe o que é o Git e entendeu como funciona o processo de instalação, vamos agora ver como funciona o processo de inicialização de um repositório do Git. Um repositório é o local onde as mudanças serão salvas, permitindo que possamos fazer o versionamento das mudanças e das versões dos arquivos.

Um repositório pode ficar na pasta raiz de um projeto, junto com os arquivos de código-fonte e configuração do projeto. Não existem restrições de quantas pastas podem conter nessa pasta raiz, pois o Git salva e controla as alterações através de uma pasta oculta denominada de “.git”.

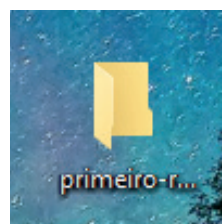
Mas, como abordamos na aula 06, não adianta apenas criar a pasta “.git”. Dentro dessa pasta, existe um conjunto de outras subpastas e arquivos, que o Git usa para salvar as alterações e os metadados das alterações. Assim, precisamos executar um comando para que todo esse conjunto seja criado. Com isso, precisamos fazer o que chamamos de **inicialização do repositório**, onde toda a estrutura de arquivos do Git será criada dentro da pasta do projeto.

Para isso, precisamos fazer essas duas operações:

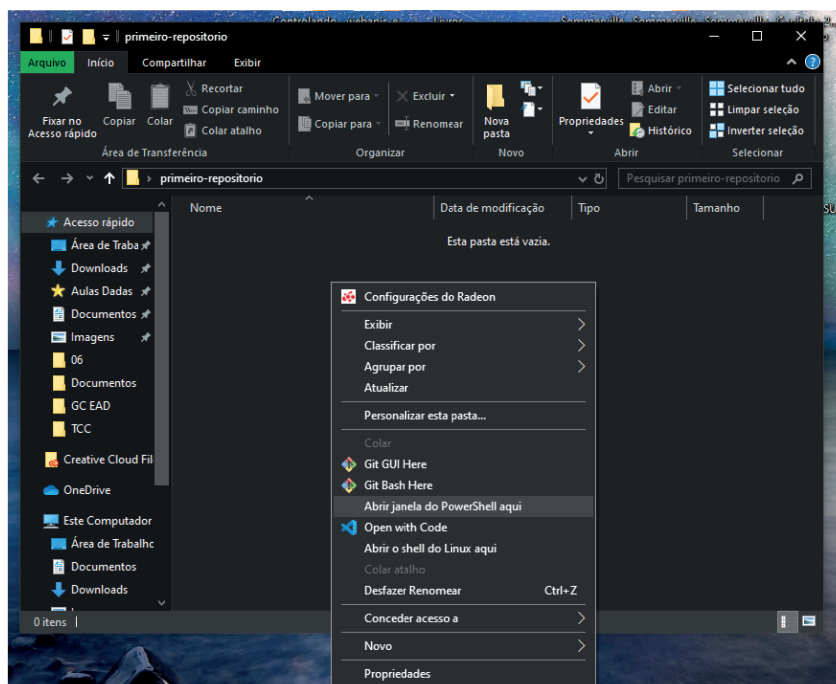
1. criar uma pasta, onde os arquivos do projeto serão salvos – ou selecionar uma pasta com um código-fonte existente, onde inicializaremos o versionamento;
2. inicializar a estrutura de arquivos da pasta “.git”, usando o comando **git init**.

Com isso dito, vamos mostrar como fazer isso. Com o git instalado na máquina, crie uma pasta em alguma pasta de referência do seu sistema operacional: Pode ser a Área de Trabalho (Desktop), pode ser a pasta de usuário do seu sistema ou outra em que você tenha fácil acesso.

No nosso cenário, vamos usar o sistema Windows e criar a nossa pasta na Área de Trabalho. A pasta será chamada de “primeiro-repositorio”. Crie essa pasta no lugar da sua preferência, como foi dito.



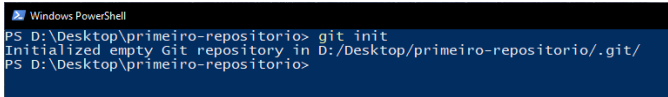
Agora, entre nessa pasta usando o Explorador de Arquivos e abra um terminal dentro dela. No Windows, você abre um novo terminal apertando a tecla SHIFT, enquanto você clica no botão direito do mouse na janela da pasta no Explorer. Clique na opção “Abrir janela do Terminal aqui” ou “Abrir janela do PowerShell aqui”, de acordo com o sistema operacional que você esteja usando.



Como podemos ver, você já criou a pasta onde armazenaremos os arquivos. Agora vamos inicializar o nosso repositório usando o comando:

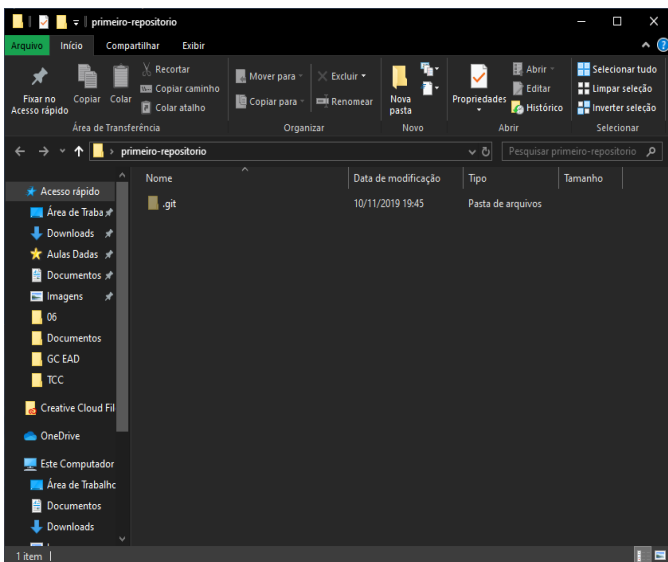
```
git init
```

Digite esse comando e tecla Enter. Aguarde o retorno do terminal, indicando se houve erro ou deu tudo certo.



```
Windows PowerShell
PS D:\Desktop\primeiro-repositorio> git init
Initialized empty Git repository in D:/Desktop/primeiro-repositorio/.git/
PS D:\Desktop\primeiro-repositorio>
```

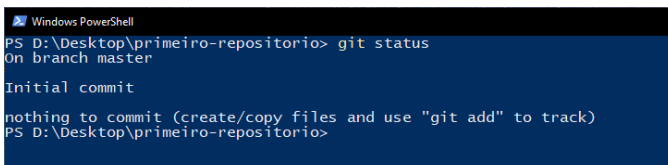
Se você vir a mensagem “Initialized empty Git repository in (...)”, você criou com sucesso a estrutura de pastas do git, transformando a pasta do seu “projeto” em um repositório do Git. Para comprovar isso, você pode ver na pasta (caso no seu explorador esteja com a opção de ver pastas e arquivos ocultos habilitada) a pasta “.git”.



Caso não tenha, vamos executar um outro comando, que mostra o status do repositório. Digite:

```
git status
```

E tecla Enter. Aguarde o retorno do terminal. Se o Git informar “nothing to commit”, quer dizer que o Git está pronto para receber as nossas alterações.

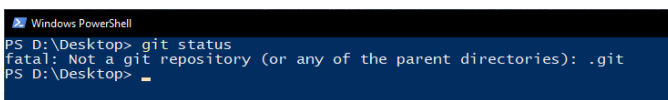


```
Windows PowerShell
PS D:\Desktop\primeiro-repositorio> git status
On branch master

Initial commit

nothing to commit (create/copy files and use "git add" to track)
PS D:\Desktop\primeiro-repositorio>
```

Se o repositório não inicializar, o Git reclamará com a seguinte mensagem, caso seja digitado o comando status: “fatal: Not a git repository (or any of the parent directories): .git”



```
Windows PowerShell
PS D:\Desktop> git status
fatal: Not a git repository (or any of the parent directories): .git
PS D:\Desktop>
```

Se isso acontecer, verifique o que foi feito antes, para procurar possíveis erros.

Com isso feito, precisamos configurar qual vai ser o autor dos commits. Se isso não for feito, o Git não permitirá que os commits sejam realizados, ou poderá usar o nome de usuário da máquina como sendo o autor dos commits. É importante que você faça essa configuração, pois as hospedagens de repositórios usam essas informações para fins de estatísticas e histórico.

Para fazer isso, ainda no terminal, digite o seguinte comando (e depois tecla Enter):

```
git config --global user.name "<seu nome>"
```

Evidentemente, troque <seu nome>, pelo seu nome. Mantenha as aspas, para que o comando funcione corretamente. Por exemplo, meu nome é Luis Aurelio Casoni. Assim, devo digitar o seguinte:

```
git config --global user.name "Luis Aurelio Casoni"
```

Digite o comando e tecla Enter. O comando não emite nenhuma resposta. Agora, vamos colocar o e-mail para identificar o autor do commit. Use um *e-mail* que você esteja utilizando muito. Para configurar o e-mail do autor do commit, digite (e depois tecla Enter):

```
git config --global user.email "<seu email>"
```

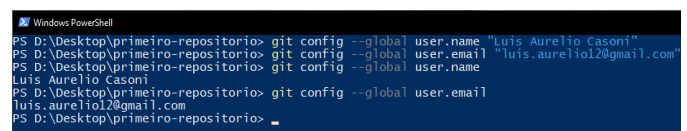
Da mesma forma, troque <seu email> pelo seu endereço de e-mail, mantendo as aspas, para que o comando funcione corretamente. Assim, vamos a um exemplo de e-mail:

```
git config --global user.email "luis.aurelio12@gmail.com"
```

Para verificar se as alterações deram certo, digite os seguintes comandos (e ao final de cada um deles, tecla Enter):

```
git config --global user.name
git config --global user.email
```

Esses comandos mostram o valor atual para as configurações. Se baterem com os dados que você informou, meus parabéns! Você está pronto para fazer os primeiros commits.



```
Windows PowerShell
PS D:\Desktop\primeiro-repositorio> git config --global user.name "Luis Aurelio Casoni"
PS D:\Desktop\primeiro-repositorio> git config --global user.email "luis.aurelio12@gmail.com"
PS D:\Desktop\primeiro-repositorio> git config --global user.name
Luis Aurelio Casoni
PS D:\Desktop\primeiro-repositorio> git config --global user.email
luis.aurelio12@gmail.com
PS D:\Desktop\primeiro-repositorio>
```

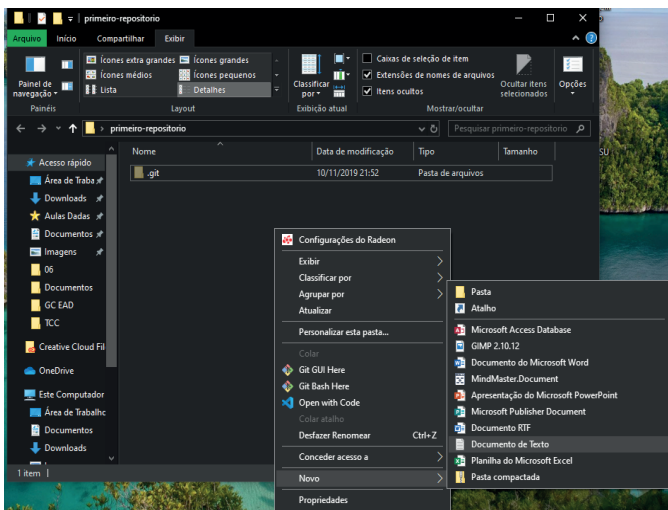
O processo de configuração do nome e do e-mail só é necessário fazer apenas uma única vez, quando o Git é instalado. Se por acaso o Git for desinstalado e instalado novamente, você precisa passar a máquina para um outro programador que usar o Git, ou instalar o Git em uma outra máquina e repetir o processo.

Agora, vamos mostrar como fazer os seus primeiros

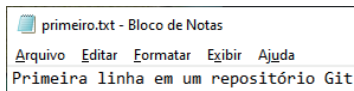
commits. Vamos lá?

2 - Commitando os seus primeiros arquivos

Vamos agora aprender como fazer os seus primeiros commits no repositório do Git. Para isso, precisamos criar um arquivo, que irá servir para as nossas demonstrações. Vamos criar um arquivo de texto na pasta primeiro-repositorio, denominado de “primeiro.txt”. No Windows, basta apenas clicar com o botão direito na janela da pasta no Windows Explorer, clicar em “Novo” e depois em “Novo Documento de Texto”:



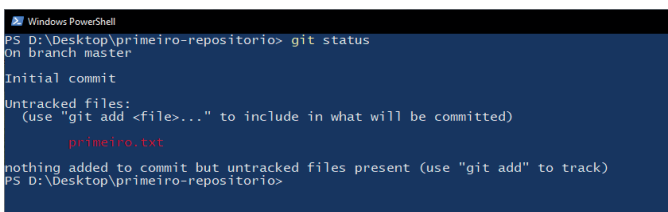
Depois, dê o nome de “primeiro.txt” e abra ele no editor de texto simples. Escreva qualquer coisa nele. Por exemplo, no nosso caso, vou escrever: “Primeira linha em um repositório Git”.



Salve o arquivo (no bloco de notas, “Arquivo” > “Salvar”, ou o atalho Ctrl + S) e volte ao terminal. Digite o comando (e tecla Enter):

```
git status
```

Provavelmente, você vai ver uma tela com os seguintes dizeres, “Untracked Files” e “nothing added to commit but untracked files present (use “git add” to track)”, indicando o arquivo “primeiro.txt” no texto.



O que isso significa é que o Git detectou o novo arquivo, mas ainda não está rastreado no repositório. Assim, para que

possamos adicionar essa versão do arquivo ao repositório, precisamos seguir os seguintes passos:

1. Adicionar o arquivo à área de stage, deixando a alteração pronta para ser commitada;
2. Commitar o arquivo, registrando a versão do arquivo no repositório.

Assim, vamos aos passos. Primeiramente, precisamos adicionar o arquivo à área de stage, para indicar ao Git que o arquivo está pronto para ser commitado. Para isso, usamos o comando **git add**, que pode ser seguido por um nome de arquivo ou um nome de uma pasta do repositório (nesse caso todos os arquivos da pasta são registrados). Também podemos usar o caractere curinga asterisco (*) que indica que vamos adicionar todos os arquivos modificados no repositório a essa área.

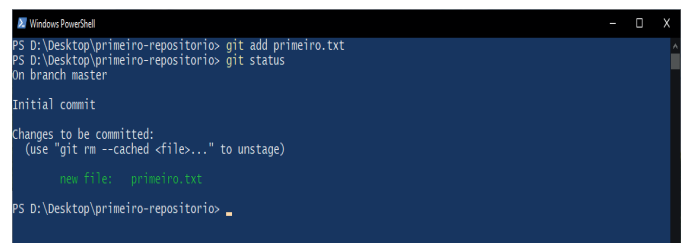
Vamos então adicionar o arquivo “primeiro.txt” a nossa área de stage. Digitamos o seguinte comando (e teclamos Enter após a digitação):

```
git add primeiro.txt
```

Em seguida, vamos certificar o estado atual do arquivo primeiro.txt. Com isso, vamos digitar novamente o comando:

```
git status
```

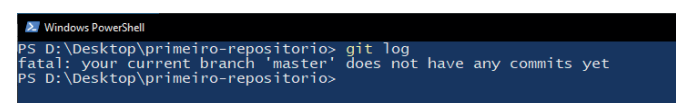
Agora, uma mensagem dizendo “Changes to be committed” aparece, indicando os arquivos que estão prontos para serem salvos no repositório. Nesse caso, apenas aparece um arquivo, que é o nosso primeiro.txt.



Vamos agora certificar que não existe nenhum commit no nosso repositório. Para isso, vamos digitar o seguinte comando:

```
git log
```

Esse comando mostra a lista dos commits feitos. Mas, por enquanto, não temos nenhum commit feito, como pode ser visto na resposta do comando:

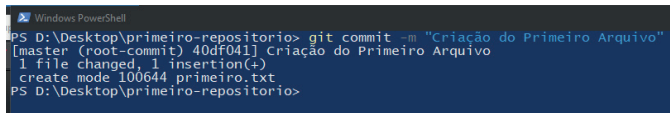


Agora, vamos registrar a nossa primeira versão do arquivo no nosso repositório. Vamos usar o comando **git commit**, usando o parâmetro -m, que será a mensagem do commit. A mensagem é um texto breve que explica as alterações que foram feitas. Assim, no nosso caso, temos a criação do primeiro arquivo. Dessa forma, podemos commitar o arquivo

usando o comando:

```
git commit -m "Criação do Primeiro Arquivo"
```

O Git fará o armazenamento da versão e informará o êxito da operação.

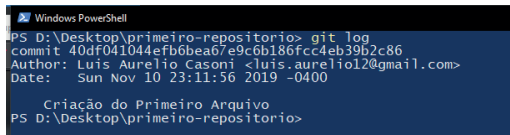


```
PS D:\Desktop\primeiro-repositorio> git commit -m "Criação do Primeiro Arquivo"
[master (root-commit) 40df041] Criação do Primeiro Arquivo
1 file changed, 1 insertion(+)
 create mode 100644 primeiro.txt
PS D:\Desktop\primeiro-repositorio>
```

Vamos agora ver o histórico de commits, para ver como ficou. Digite o comando:

```
git log
```

Você vai ver um item, que representa o commit que acabamos de fazer.



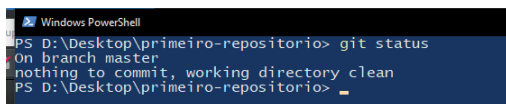
```
PS D:\Desktop\primeiro-repositorio> git log
commit 40df041044efb6bea67e9c6b186fcc4eb39b2c86
Author: Luis Aurelio Casoni <luis.aurelio12@gmail.com>
Date: Sun Nov 10 23:11:56 2019 -0400

    Criação do Primeiro Arquivo
PS D:\Desktop\primeiro-repositorio>
```

Agora, vamos ver o status do nosso repositório, usando o comando:

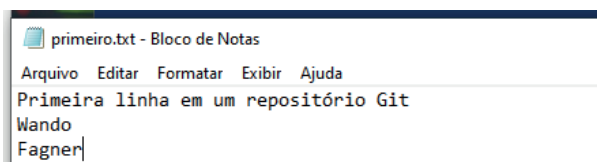
```
git status
```

Você vai ver que não temos arquivos modificados pendentes. Assim, o nosso diretório de trabalho está limpo (clean).



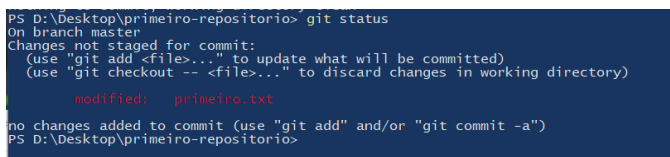
```
PS D:\Desktop\primeiro-repositorio> git status
On branch master
nothing to commit, working directory clean
PS D:\Desktop\primeiro-repositorio>
```

Vamos agora fazer uma segunda alteração no nosso arquivo primeiro.txt. Vou adicionar duas linhas a ele, uma dizendo “Wando” e outra dizendo “Fagner”, como mostra essa imagem a seguir:



```
primeiro.txt - Bloco de Notas
Arquivo Editar Formatar Exibir Ajuda
Primeira linha em um repositório Git
Wando
Fagner
```

Salve o arquivo e volte ao prompt de comando. Digite o comando **git status** para ver como está o repositório agora. Agora, temos uma modificação pendente em nosso repositório:



```
PS D:\Desktop\primeiro-repositorio> git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   primeiro.txt

no changes added to commit (use "git add" and/or "git commit -a")
PS D:\Desktop\primeiro-repositorio>
```

Bem, agora para efetivar essa commit, precisamos

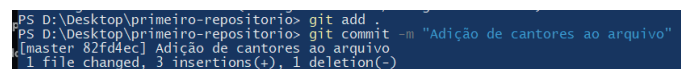
seguir duas etapas. Primeiro, adicionamos o arquivo a área de stage, usando o comando **git add ***. Observe que nós estamos usando o curinga, para adicionar todos os arquivos não modificados no repositório para ficarem disponíveis para serem commitados.

Observe que quando fazemos o comando **git commit**, apenas os arquivos que foram adicionados à área de stage serão commitados como uma versão. Assim, se tivermos dois arquivos modificados em um repositório e apenas um for adicionado a área de stage, quando for feito o commit, apenas o arquivo que estava na área de stage será commitado.

Em seguida, vamos commitar o arquivo, usando o comando **git commit**, informando no parâmetro -m, a alteração feita, que foi a adição de nomes de dois cantores ao texto. O comando é o seguinte:

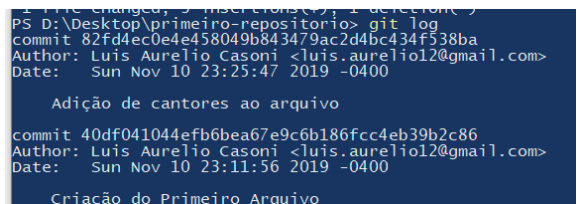
```
git commit -m "Adição de cantores ao arquivo"
```

Novamente, o Git informará os arquivos que serão efetivados nesse commit.



```
PS D:\Desktop\primeiro-repositorio> git add .
PS D:\Desktop\primeiro-repositorio> git commit -m "Adição de cantores ao arquivo"
[master 82fd4ec] Adição de cantores ao arquivo
1 file changed, 3 insertions(+), 1 deletion(-)
PS D:\Desktop\primeiro-repositorio>
```

Agora, se rodarmos o comando **git log**, teremos dois commits, ao invés de um:



```
1 file changed, 3 insertions(+), 1 deletion(-)
PS D:\Desktop\primeiro-repositorio> git log
commit 82fd4ec0e4e458049b843479ac2d4bc434f538ba
Author: Luis Aurelio Casoni <luis.aurelio12@gmail.com>
Date: Sun Nov 10 23:25:47 2019 -0400

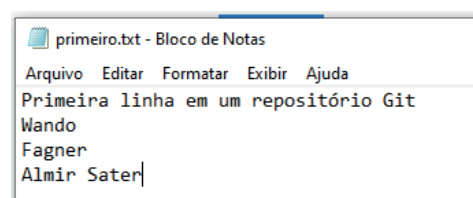
    Adição de cantores ao arquivo

commit 40df041044efb6bea67e9c6b186fcc4eb39b2c86
Author: Luis Aurelio Casoni <luis.aurelio12@gmail.com>
Date: Sun Nov 10 23:11:56 2019 -0400

    Criação do Primeiro Arquivo
PS D:\Desktop\primeiro-repositorio>
```

Mas, vamos supor que você esqueceu de alterar alguma coisa e deseja salvar essa alteração no último commit que você fez. Para isso, temos o recurso chamado de **amend**, que permite alterar o conteúdo e a mensagem do último commit. Para isso, precisamos usar o argumento **--amend** no comando de commit.

Para exemplificar isso, vamos fazer uma pequena alteração em nosso arquivo de testes, adicionando mais um grande cantor à lista. Vamos adicionar à lista do arquivo “primeiro.txt” o nome de “Almir Sater”.



```
primeiro.txt - Bloco de Notas
Arquivo Editar Formatar Exibir Ajuda
Primeira linha em um repositório Git
Wando
Fagner
Almir Sater
```

Salve-o. Agora, retornemos ao nosso terminal. Vamos adicionar a nossa alteração, digitando **git add ***, e depois teclando Enter. Em seguida, digite:

```
git commit --amend -m "Adição de três cantores ao arquivo"
```


Agora, observe a saída. Ele vai indicar que o commit foi feito com êxito:

```
Windows PowerShell
PS D:\Desktop\primeiro-repositorio> git add *
PS D:\Desktop\primeiro-repositorio> git commit --amend -m "Adição de três cantores ao arquivo"
[master 43f9d96] Adição de três cantores ao arquivo
Date: Sun Nov 10 23:25:47 2019 -0400
1 file changed, 4 insertions(+), 1 deletion(-)
PS D:\Desktop\primeiro-repositorio>
```

Mas, se agora digitarmos o comando **git log**, veremos que ao invés de três commits, temos apenas dois. É que essa operação modificou o conteúdo e a mensagem do último commit feito.

```
Windows PowerShell
PS D:\Desktop\primeiro-repositorio> git log
commit 43f9d96d42b0fd561b1a1060a021cbab96a0b793
Author: Luis Aurelio Casoni <luis.aurelio12@gmail.com>
Date: Sun Nov 10 23:25:47 2019 -0400

    Adição de três cantores ao arquivo

commit 40df041044efb6bea67e9c6b186fcc4eb39b2c86
Author: Luis Aurelio Casoni <luis.aurelio12@gmail.com>
Date: Sun Nov 10 23:11:56 2019 -0400

    Criação do Primeiro Arquivo
PS D:\Desktop\primeiro-repositorio>
```

Mas, e se quisermos voltar a uma versão anterior? Para isso é simples. Primeiramente, precisamos observar a hash do commit. A hash é uma cadeia de 40 caracteres, que identifica o commit. A cadeia completa de cada commit é exibida quando digitamos o comando **git log**.

Para que possamos voltar para um commit, precisamos do hash desse commit o para qual queremos voltar. Mas, não é necessário ter em mente os quarenta caracteres. Apenas os sete primeiros caracteres são necessários para acessar o commit. Como já dissemos, para ver quais são os caracteres, basta apenas digitar **git log** (e teclar Enter) no terminal para vermos as cadeias de caracteres. A última imagem mostra o resultado desse log de commits.

O primeiro commit tem como sete primeiros dígitos aqui nesse exemplo a string “40df041”. Assim, precisamos o seguinte comando para retornar ao nosso primeiro commit:

```
git checkout 40df041
```

```
PS D:\Desktop\primeiro-repositorio> git checkout 40df041
HEAD is now at 40df041... Criação do Primeiro Arquivo
PS D:\Desktop\primeiro-repositorio> git status
```

E, com isso, voltamos ao estado do primeiro commit. Feche e abra o arquivo para verificar isso:

```
Windows PowerShell
PS D:\Desktop\primeiro-repositorio> git checkout 40df041
Note: checking out '40df041'.
You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.
If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:
git checkout -b <new-branch-name>

HEAD is now at 40df041... Criação do Primeiro Arquivo
PS D:\Desktop\primeiro-repositorio> git status
HEAD is now at 40df041... Criação do Primeiro Arquivo
nothing to commit, working directory clean
```

feito anteriormente? Bem, precisamos saber qual é a hash do commit que era o último da lista. Porém, um simples **git log** não basta:

```
Windows PowerShell
PS D:\Desktop\primeiro-repositorio> git log
commit 40df041044efb6bea67e9c6b186fcc4eb39b2c86
Author: Luis Aurelio Casoni <luis.aurelio12@gmail.com>
Date: Sun Nov 10 23:11:56 2019 -0400

    Criação do Primeiro Arquivo
```

Isso se deve ao fato de que agora o commit atual do repositório é o primeiro commit. Assim, os commits que vieram depois do primeiro estão ignorados. Mas, podemos vê-los, usando o argumento **-all**. Assim, digite:

```
git log --all
```

```
PS D:\Desktop\primeiro-repositorio> git log --all
commit 43f9d96d42b0fd561b1a1060a021cbab96a0b793
Author: Luis Aurelio Casoni <luis.aurelio12@gmail.com>
Date: Sun Nov 10 23:25:47 2019 -0400

    Adição de três cantores ao arquivo

commit 40df041044efb6bea67e9c6b186fcc4eb39b2c86
Author: Luis Aurelio Casoni <luis.aurelio12@gmail.com>
Date: Sun Nov 10 23:11:56 2019 -0400

    Criação do Primeiro Arquivo
```

Com isso, passamos a ver todos os commits feitos. Agora, para deixar o commit em que adicionamos os três cantores como o commit atual, precisamos rodar o comando **git checkout**, informando os sete primeiros dígitos do hash do commit desejado.

Como podemos ver, o commit em que desejamos voltar tem o hash inicial “43f9d96”. Assim, para voltar a ele, digitamos o seguinte comando:

```
git checkout 43f9d96
```

```
PS D:\Desktop\primeiro-repositorio> git checkout 43f9d96
Previous HEAD position was 40df041... Criação do Primeiro Arquivo
HEAD is now at 43f9d96... Adição de três cantores ao arquivo
```

Assim, o que nós fizemos antes da volta dos commits está restaurado. Abra novamente o arquivo “primeiro.txt” para comprovar isso:

```
Windows PowerShell
PS D:\Desktop\primeiro-repositorio> git status
nothing to commit, working directory clean
PS D:\Desktop\primeiro-repositorio> git log
commit 43f9d96d42b0fd561b1a1060a021cbab96a0b793
Author: Luis Aurelio Casoni <luis.aurelio12@gmail.com>
Date: Sun Nov 10 23:25:47 2019 -0400

    Adição de três cantores ao arquivo

commit 40df041044efb6bea67e9c6b186fcc4eb39b2c86
Author: Luis Aurelio Casoni <luis.aurelio12@gmail.com>
Date: Sun Nov 10 23:11:56 2019 -0400

    Criação do Primeiro Arquivo
PS D:\Desktop\primeiro-repositorio> git status
nothing to commit, working directory clean
```

Outra forma de retornar ao último commit feito é executar o comando:

```
git checkout master
```

Esse comando retorna automaticamente à última

E se nos arrependemos e decidir voltar para o commit

alteração feita, sem precisar acessar o histórico de commits ignorados.

```
PS D:\Desktop\primeiro-repositorio> git checkout master
Previous HEAD position was 40df041... Criação do Primeiro Arquivo
Switched to branch 'master'
PS D:\Desktop\primeiro-repositorio>
```

E, com isso, finalizamos a nossa aula. Na próxima, vamos falar a respeito de Github. Até lá!

Retomando a aula

Chegamos ao final da nossa sétima aula. Vamos relembrar?

1 – Iniciando um repositório do git

Nesta seção, você viu como fazer a inicialização do repositório do Git. Ele é feito através do comando **git init**, que faz o processo de criação da estrutura de pastas do diretório “.git”, que armazena as diferenças e os metadados do repositório.

2 – Commitando os seus primeiros arquivos

Nesta aula, você aprendeu como funciona o processo de commit, que consiste em adicionar os arquivos à área de stage, deixando prontos para a realização do commit. Vimos também como retornar a um ponto anterior a um commit feito.

Vale a pena

Vale a pena ler,

AQUILES, Alexandre; FERREIRA, Rodrigo. *Controlando versões com Git e GitHub*. São Paulo: Casa do Código, 2017.

Vale a pena acessar,

DUDLER, Roger. git - guia prático. 2019. Disponível em: https://rogerdudler.github.io/git-guide/index.pt_BR.html. Acesso em 19 nov. 2019.

GIT. .3 Primeiros passos - Noções Básicas de Git. Disponível em: <https://git-scm.com/book/pt-br/v1/Primeiros-passos-No%C3%A7%C3%B5es-B%C3%A1sicas-de-Git>. Acesso em 19 nov. 2019.

H., Rafael. Tutorial do GIT Básico – Introdução ao

GIT. Hostinger, 2019. Disponível em: https://rogerdudler.github.io/git-guide/index.pt_BR.html. Acesso em 19 nov. 2019.

JANUSKA, Antonin. My Personal Git Tricks Cheatsheet. DEV, 2019. Disponível em: <https://dev.to/antjanus/my-personal-git-tricks-cheatsheet-23j1>. Acesso em 19 nov. 2019.

SCHMITZ, Daniel. Tudo que você queria saber sobre Git e GitHub, mas tinha vergonha de perguntar. Tableless, 2015. Disponível em: <https://tableless.com.br/tudo-que-voce-queria-saber-sobre-git-e-github-mas-tinha-vergonha-de-perguntar/>. Acesso em 19 nov. 2019.

Minhas anotações