**Guia Rápido:**



[O que é Git.](#_cupp7sms6u9q)

[Por que usar o Git](#_4qb64ynx345t)

[Instalação](#_tmkhhcf0vuxk)

[Mas antes de começar](#_85kuwuwrperf)

[Configuração](#_qkc8ayu2oqwg)

[Iniciando Repositório](#_jbc1u4zfp5kb)

[Rastreando arquivo](#_tf72r2x6ekxu)

[Salvando Arquivo no Repositório](#_97bvslohkvqy)

[Alterando Arquivo](#_2vxeihxuq2fj)

[Verificando Alterações Feitas](#_fje4qf9yb7id)

[Desfazendo mensagem errada](#_j77s75y7einx)

[Desfazendo Commit](#_96xgzxud3ncu)

[Desfazendo Alterações não rastreadas](#_g0f9gxc6osrh)

[Desfazendo Alterações rastreadas](#_jgw4x42b2a3b)

[Verificando alterações não rastreadas](#_sgjnk5vkwi6n)

[Verificando alterações rastreadas](#_iu5gmm65t8lm)

[Verificando alterações em commits](#_7givtyq8c52a)

[Removendo Arquivos](#_doi8k8lan0k4)

[Renomeando/Movendo Arquivos](#_grmox3epa01s)

[Como funciona](#_sjdenlosgrjl)

[Criando repositório remoto no GitHub](#_hmvqxycpjbtt)

[Compartilhando repositório remoto](#_9idn9eidlwa4)

[Algumas regras ao fazer push](#_u5t95f7zb6iy)

[Recebendo atualização do repositório remoto](#_sbxyvagq6xre)

**Resumão**

# O que é Git.

Git e um versionamento de software, responsável por manter salvo várias versões de código e arquivos alterados em seu projeto.

Criado pelo [Linus Torvalds](https://pt.wikipedia.org/wiki/Linus_Torvalds), depois de farpas com desenvolvedores de outros software de versionamento por uma possível melhora, resolveu criar sua própria com intuito de facilitar o trabalho com kernel do linux ([uma breve história](https://git-scm.com/book/pt-br/v1/Primeiros-passos-Uma-Breve-Hist%C3%B3ria-do-Git)).

O nome Git e uma gíria do inglês britânico para alguém “estúpido”, “cabeça dura”, mas como o próprio criador disse, pode significar várias coisas dependendo do seu humor.

# Por que usar o Git

Além de manter todo histórico de alteração salva do arquivo é possível descartar alterações indesejadas (onde seu ctrl+z não tem poder), recuperar arquivos deletados, comparar duas versões do mesmo arquivo para ter noção do que foi modificado.

Trabalho em equipe fica muito mais dinâmico e fácil utilizando corretamente o git, evitando perda de código ou arquivo, economizando tempo (upload, download), facilitando o compartilhamento do projeto.

**Setup**

# Instalação

Guia do próprio Git e bem completo e eu estou com preguiça de colocar e explicar aqui, [clique aqui](https://git-scm.com/book/pt-br/v1/Primeiros-passos-Instalando-Git) para acessar.

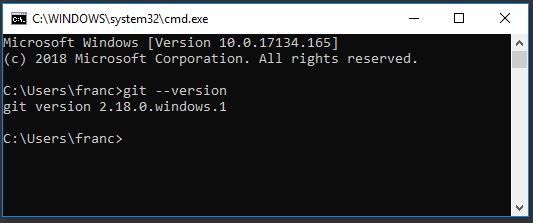
# Mas antes de começar

O Git tem uma interface gráfica para trabalhar com ele, onde irei abordar em algum capítulos aí, durante a explicação vou utilizar o Prompt/Terminal onde maioria dos programadores utiliza.

Os comando do git são os mesmo para qualquer plataforma, seja windows, linux ou mac.

Se você fez a instalação padrão já é possível utilizar o git em sua máquina, abra o prompt e digitar:

|  |
| --- |
| git --version |



(Tudo estiver ok precisa imprimir a versão do Git)

**Na prática**

# Configuração

Para identificar a pessoa que alterou o documento o Git precisa de duas informações suas, nome e email, para configurar:

|  |
| --- |
| git config --global user.name "Meu nome" git config --global user.email "MeuEmail@dominio.com" |

Em seguida criei uma parte e um documento em texto para trabalhar em cima:

|  |
| --- |
| mkdir Git cd Git copy NUL AprendendoGit.txt |

# Iniciando Repositório

Para iniciar um repositório apenas digite:

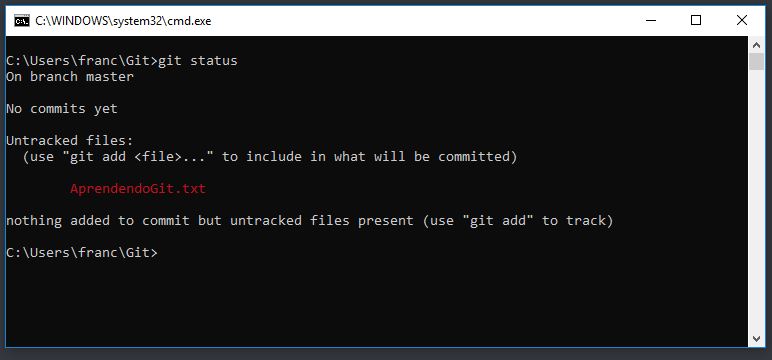
|  |
| --- |
| git init |

# 

# Rastreando arquivo

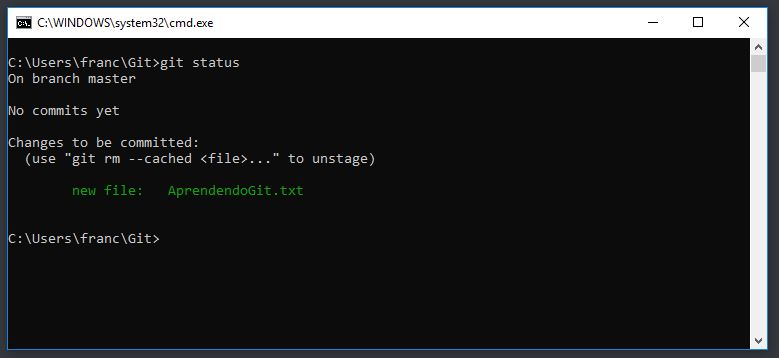
Criamos nosso repositório porém novo arquivo ainda não está sendo rastreado pelo Git, para visualizar o estado do arquivo digite:

|  |
| --- |
| git status |



Para o Git rastrear nosso arquivo, precisamos digitar o comando:

|  |
| --- |
| git add AprendendoGit.txt |



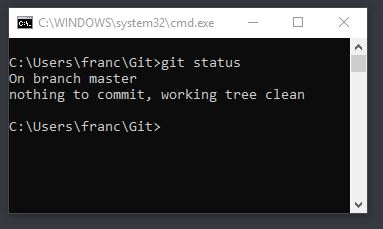
(Executando: ‘git status’)

# Salvando Arquivo no Repositório

Criamos o repositório e o arquivo está sendo rastreado pelo Git, precisamos dizer ao Git que queremos salvar a alteração no repositório, para isso precisamos executar um comando e descrever a ação.

|  |
| --- |
| git commit -m "Primeiro Commit" |

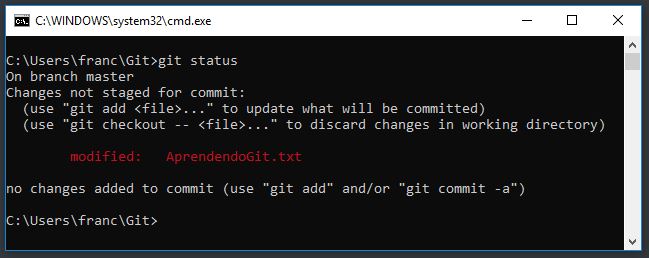
Ao verificar o estado do repositório o Git nos informa que não existe nenhuma nenhuma alteração no repositório.



# 

# Alterando Arquivo

Eu alterei o arquivo “AprendendoGit.txt”, colocando os comandos que aprendendo até agora, salvei e verifiquei o estado do repositório:



Git reclamou que existe um arquivo que foi modificado, estão vamos salvar essas alterações no nosso repositório, e preciso executar:

|  |
| --- |
| git add AprendendoGit.txt git commit -m "Salvando os comandos aprendidos do git" |

Porém dessa vez vamos encurtar o processo:

|  |
| --- |
| git commit -am "Salvando os comandos aprendidos do git" |

Adicionamos -am, onde ‘a’ seria ‘add’, ‘m’ informando que vamos colocar a mensagem do commit direto no comando.

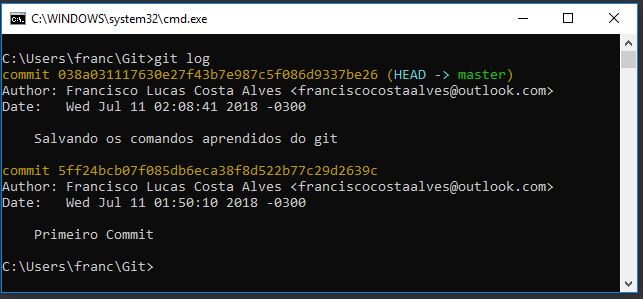
(Se adicionamos ‘git commit’, sem o -m vai aparecer um editor para adicionar a mensagem do commit)

# Verificando Alterações Feitas

Para informar de todo commit já feito, podemos utilizar:

|  |
| --- |
| git log |

Vai aparecer um log mostrando toda os commit feitos, com nome, email e data feita do commit que o autor salvou, junto com mensagem do commit.



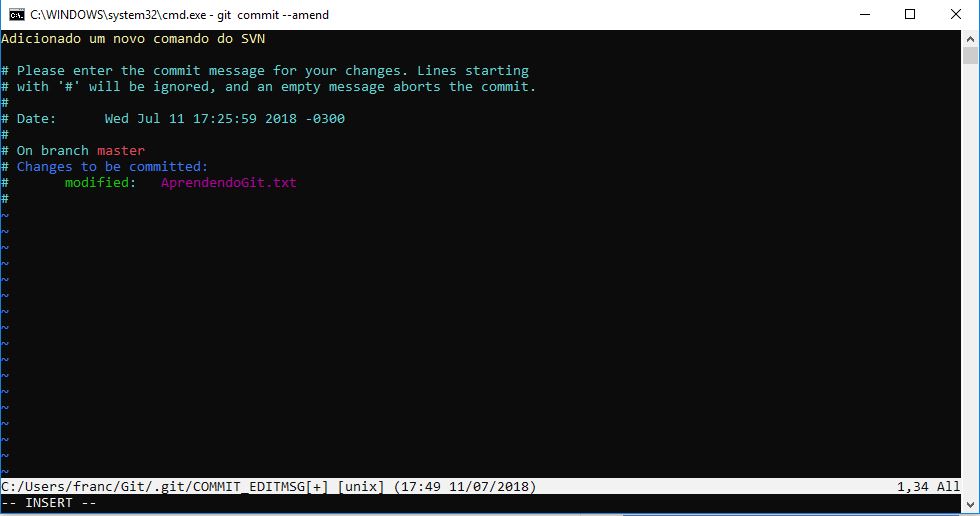
# Desfazendo mensagem errada

Alterei o arquivo de texto novamente e em seguida fiz o commit para salvar alteração no repositório:

|  |
| --- |
| git commit -m "Adicionado um novo comando do SVN" |

Porém eu errei na mensagem do commit colocando nome de um outro software de versionamento, para corrigir basta executar o comando:

|  |
| --- |
| git commit --amend |



Vai abrir um editor de texto, e a primeira linha está a mensagem do commit errado, para editar pressione “Insert” no teclado, Quando corrigir a mensagem pressione “Esc”, saindo da edição, para salvar digite “:wq” e pressione “Enter”, caso queira sair sem salvar digite “:q!”.

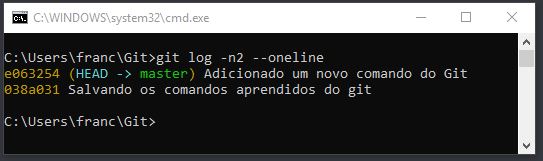
Finalizado, quando ver o log novamente, vai ver sua mensagem corrigida.

# Desfazendo Commit

Sempre bom ter certeza que está tudo certo antes de fazer um commit, mas como errar é humano vamos ver procedimento para resetar isso.

No meu último commit eu salvei um link duvidoso então vamos da uma olhada no log:

|  |
| --- |
| git log -n2 --oneline |



Utilizando estes 2 parâmetros para o “git log”, retorna um log bem simples, com o código do commit e a mensagem, isso é importante para pegar o código do commit que queremos resetar nesse caso seria o último, “e063254”.

Com código em mãos é só executar o comando:

|  |
| --- |
| git revert --no-edit e063254 |

Se estiver certeza que vai resetar o **último commit,** podemos simplificar com o comando:

|  |
| --- |
| git revert --no-edit HEAD |

(HEAD sempre aponta para o último commit)

Quando visualizar log novamente, vai aparecer um novo commit escrito ‘ *Revert “Mensagem do commit resetado”* ’, caso não tenha colocado o parâmetro “--no-edit”, o git ia abrir editor de texto para você digitar a mensagem.

O Git simplesmente restaurou a versão antiga do seu arquivo e fez um commit para essa ação, abre ou recarregue seu arquivo e você vai perceber que a sua última alteração sumiu.

Outra maneira de desfazer o commit é utilizando o comando:

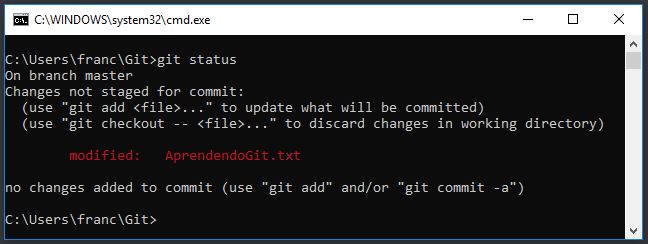
|  |
| --- |
| git reset --hard e063254 |

Dessa maneira o git simplesmente eliminou o commit no histórico, se for procurar no log você não vai encontrar o novamente commit.

Entre as duas opções eu recomendo a primeira por manter registrado no histórico, já que é essa a intenção do de utilizar o Git além de ser menos destruidora, caso erre no segundo modo você acaba perdendo as alterações do commit.

# Desfazendo Alterações não rastreadas

Caso tenha alterado seu arquivo e não tenha rastreado ele, utilizando o “git add”:



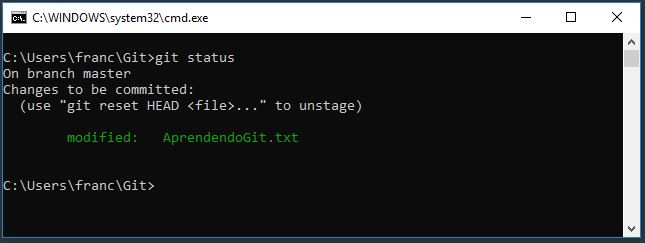
Só executar o comando:

|  |
| --- |
| git checkout -- AprendendoGit.txt |

Git restaurar o arquivo na última versão, dessa forma ele não irá aparecer no “git status”

# Desfazendo Alterações rastreadas

Se você alterou e rastreou o seu arquivo, utilizando o “git add”:



Só executar o comando:

|  |
| --- |
| git reset -- AprendendoGit.txt |

Git restaurar o arquivo na última versão, dessa forma ele não irá aparecer no “git status”.

# Verificando alterações não rastreadas

Se tiver que visualizar as diferenças feitas no arquivo com a última versão, executamos o comando:

|  |
| --- |
| git diff |

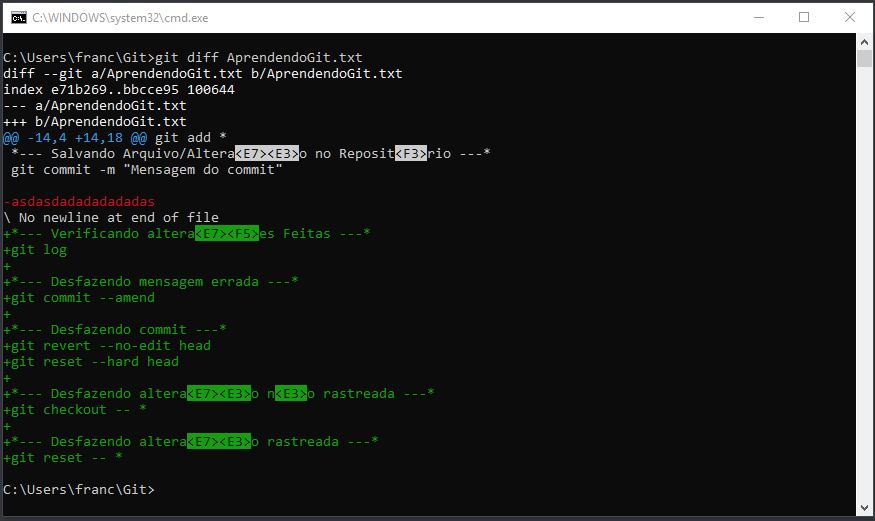
(Todos os arquivos não rastreados)

ou:

|  |
| --- |
| git diff AprendendoGit.txt |

(um arquivos específico não rastreados)

O Git vai mostrar apenas a parte que vou alterada, com um ‘+’ e na cor verde para linhas que foi adicionado, ‘-’ em vermelho para linhas removidas.



(Aperte “enter” para mostrar as linhas quando finalizado pressione ‘q’)

# Verificando alterações rastreadas

Se tiver que visualizar as diferenças feitas no arquivo rastreado com a última versão, executamos o comando:

|  |
| --- |
| git diff --staged |

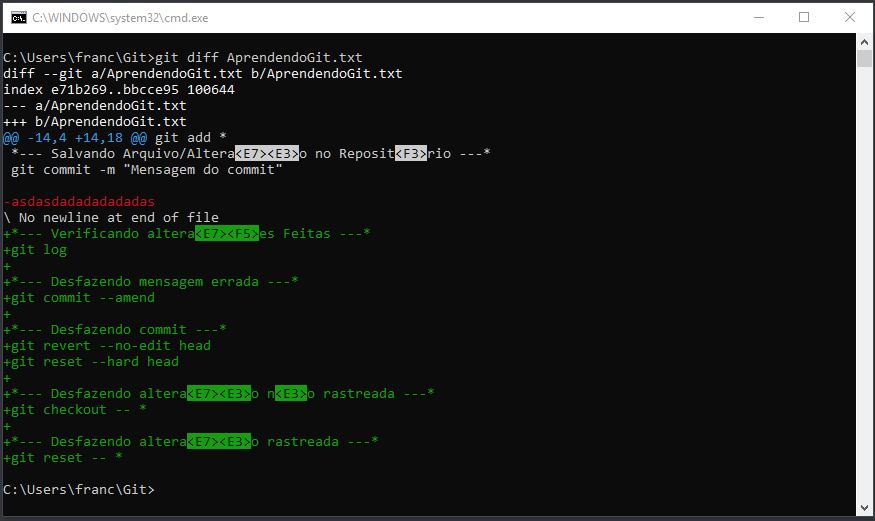
(Todos os arquivos não rastreados)

ou:

|  |
| --- |
| git diff --staged AprendendoGit.txt |

(um arquivos específico não rastreados)

O Git vai mostrar apenas a parte que vou alterada, com um ‘+’ e na cor verde para linhas que foi adicionado, ‘-’ em vermelho para linhas removidas.



(Aperte “enter” para mostrar as linhas quando finalizado pressione ‘q’)

# Verificando alterações em commits

Se tiver que visualizar as diferenças feitas entre commit, precisamos pegar seus respectivos códigos e executar:

|  |
| --- |
| git diff 038a031..e063254 |

Vai aparecer toda alteração feita entre “038a031’’ até “e063254”.

# Removendo Arquivos

Para remover um arquivo com o Git, execute:

|  |
| --- |
| git rm NomeDoArquivo |

Vai aparecer no seu “git status”, nos arquivos não rastreados:

|  |
| --- |
| deleted: NomeDoArquivo |

Não tem necessidade de deletar o arquivo pelo Git, caso tenha removido o arquivo na maneira comum/Externa o Git não vai encontrar o arquivo e declarar como deletado.

# Renomeando/Movendo Arquivos

Para renomear um arquivo com o Git, execute:

|  |
| --- |
| git mv NomeDoArquivo NovoNomeDoArquivo |

No “git status”:

|  |
| --- |
| renamed: NomeDoArquivo -> NovoNomeDoArquivo |

Para mover o arquivo executamos comando anterior, colocando apenas o diretorio:

|  |
| --- |
| git mv NomeDoArquivo pasta/NomeDoArquivo |

No “git status”:

|  |
| --- |
| renamed: NomeDoArquivo -> pasta/NomeDoArquivo |

Para as duas ações o git vai mostrar que o arquivo foi renomeado e ele já vai estar **rastreado.**

Caso tenha feito isso de maneira externa o Git é capaz de entender, porem já aconteceu comigo algo como:

|  |
| --- |
| deleted: NomeDoArquivo  new file: pasta/NomeDoArquivo |

Apenas adicione as duas alterações e no commit escreve sua intenção como, “renomeado/movido x para y”

**Repositório remoto**

# Como funciona

Se você leu capítulo “na prática”, deve ter perguntado “como compartilhar meu repositório?”, para compartilhar você precisa de um repositório remoto que seria um servidor, assim você vai enviar todas as alterações feita por você desta forma um outro desenvolvedor do seu time vai poder baixar todas alterações enviados para o repositório remoto e também podendo enviar as alterações feitas por ele.

Para criar um repositório remoto você vai precisar de uma conta em um serviço para essa finalidade, uma delas são:

1. [GitHub](https://github.com/)
2. [BitBucket](https://bitbucket.org/)

No exemplo vou utilizar o [GitHub](https://github.com/), mas o processo é igual para qualquer serviço.

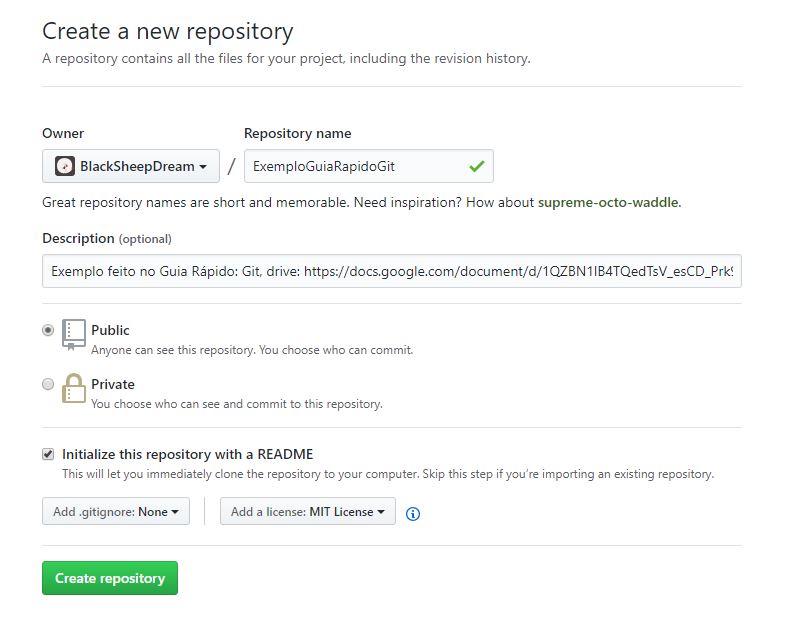
No exemplo vou enviar repositório local para repositório remoto sem chave SSH, No capítulo posterior irei ensina a configurar a chave.

# Criando repositório remoto no GitHub

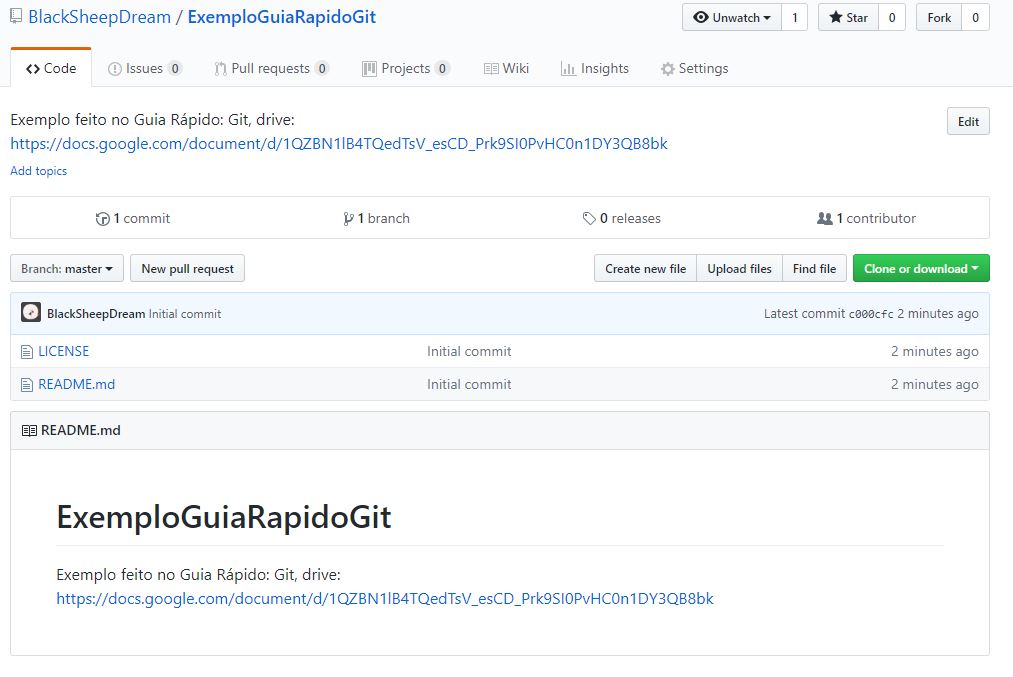
Na home do GitHub na aba de repositórios aperta botão “Novo repositório”:



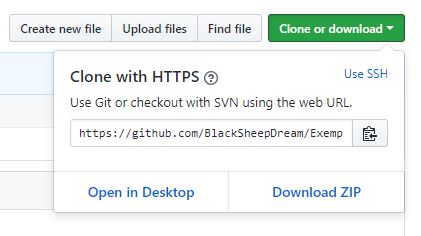
Preencha o nome do repositório, outras opções são opcionais:



Ao confirmar vai aparecer uma nova tela com diversas informações sobre o repositório que não vou entrar em detalhe:



Aperte clone or download, e copie o link exibido: (Se aparecer “Clone with SSH”, aperta o link “Use HTTPS”)



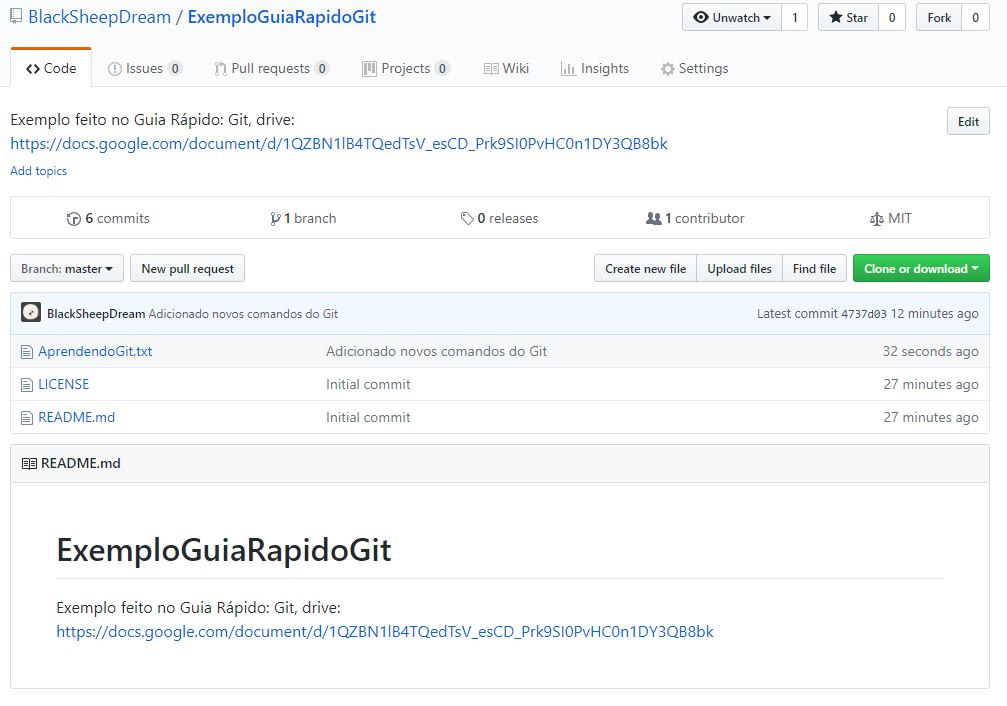
Feito isso podemos voltar no nosso terminal, agora vamos fazer a conexão com repositório local com o remoto executando o comando:

|  |
| --- |
| git remote add origin https://github.com/BlackSheepDream/ExemploGuiaRapidoGit.git |

Óbvio que vc terá que substituir o link do meu repositório para o seu, feito isso basta enviar agora toda informação da sua máquina para o remoto:

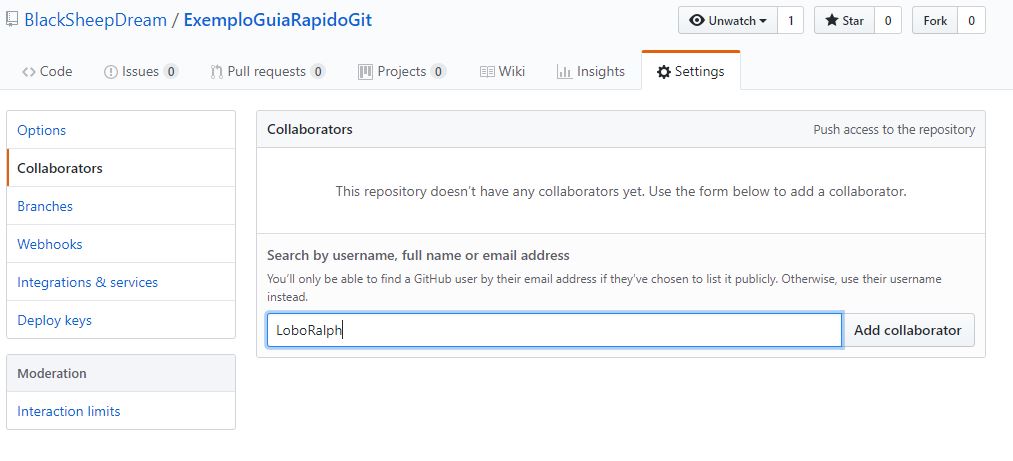
|  |
| --- |
| git push origin master |

Como estamos utilizando https, o Git vai pedir username e a senha da conta do GitHub. (Com a chave SSH não é necessário). Quando completar verifique o repositório remoto, deve conter todos os arquivos que foram salvos:

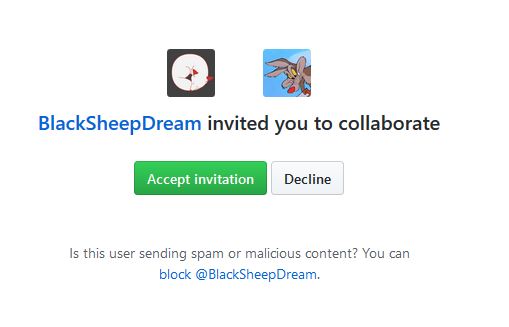


# Compartilhando repositório remoto

Para compartilhar repositório remoto para outros desenvolvedores e preciso adicionar colaboradores, sabendo disso na página do seu repositório na aba “Configurações”, procura sua equipe com username, nome completo ou email deles:



Na visão da sua equipe vai receber um convite para ser colaborador:



Ao aceitar vai ser redirecionado para página do repositório local, recebendo a permissão de alterar o repositório da maneira que achar correto, agora é necessário fazer caminho inverso, pegar repositório remoto para o repositório local.

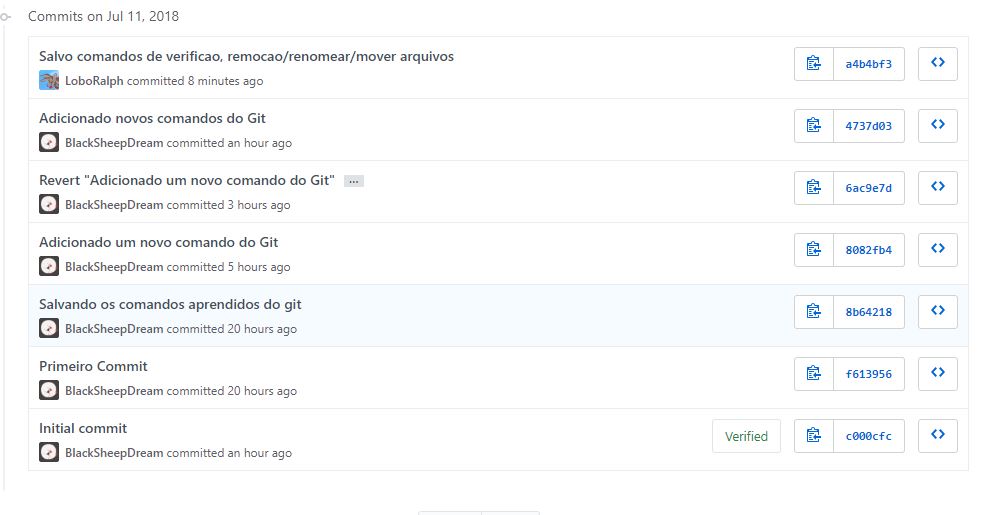
Dessa vez vamos clonar o repositório com seguinte comando:

|  |
| --- |
| git clone https://github.com/BlackSheepDream/ExemploGuiaRapidoGit.git |

Agora o “LoboRalph” tem repositório local podendo fazer oque achar melhor, depois de um dia de trabalho ele precisa enviar as alterações para repositório local, que já vimos antes:

|  |
| --- |
| git push origin master |

Checando seu repositório remoto deve aparecer commit do “LoboRalph”:



# Enviando Alterações para repositório remoto

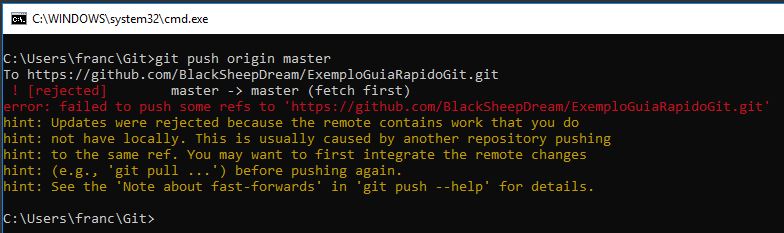
Se você leu as etapa de [criação do repositório remoto](#_hmvqxycpjbtt), você já deve ter usado comando “push” do Git:

|  |
| --- |
| git push origin master |

Isso faz com Git envie todos os commits feitos para o repositório remoto (todos os commits que não estão salvo no repositório remoto).

# Algumas regras ao fazer push

Agora sua equipe pode trabalhar com repositório remoto, vamos continuar modificando nosso arquivo e tentar fazer um push:



Recebemos um erro, isso ocorreu porque nosso repositório remoto foi modificado e não temos essa modificação no nosso repositório local, precisamos baixar essa modificação e depois enviar as suas para repositório remoto. veja como atualizar seu repositório local no próximo tópico: [Recebendo atualização do repositório remoto](#_sbxyvagq6xre).

# 

# Recebendo atualização do repositório remoto

Para baixar todas as alterações do repositório remoto para o repositório local é necessário executar:

|  |
| --- |
| git pull origin master |