

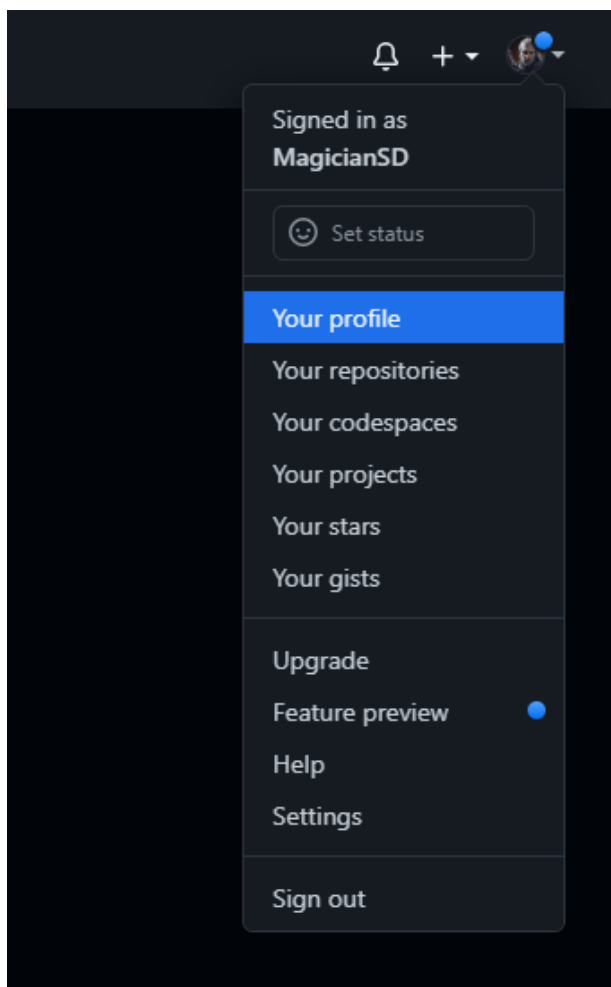
GUIA DE USO DO GUIT HUB

CONFIRMANDO DADOS DE ACESSO AO GITHUB

Acesse o GitHub e crie sua conta

CONFIGURANDO SEU PERFIL NO GITHUB

É muito recomendável que você configure seu perfil no GitHub para servir como seu portfólio das suas habilidades. Se assim desejar, acesse sua conta no canto superior direito onde ficará sua foto de identificação após edição, como mostra na imagem abaixo.

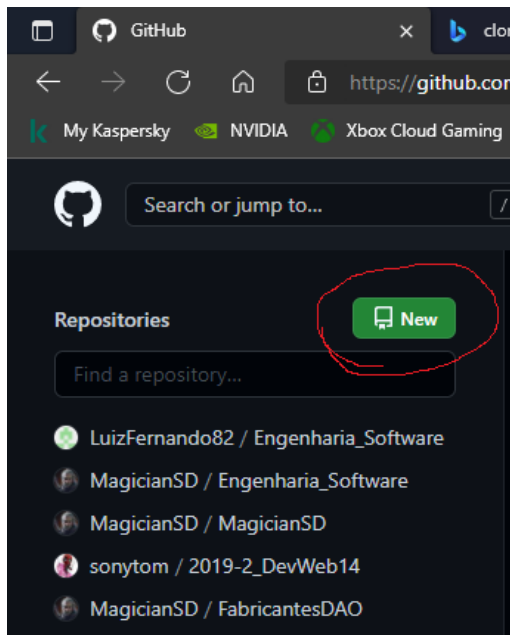


Menu no canto superior direito da tela

Feito isso, vá na opção *Your Profile*. Em seguida, clique em *Edit profile* e informe seus dados/foto.

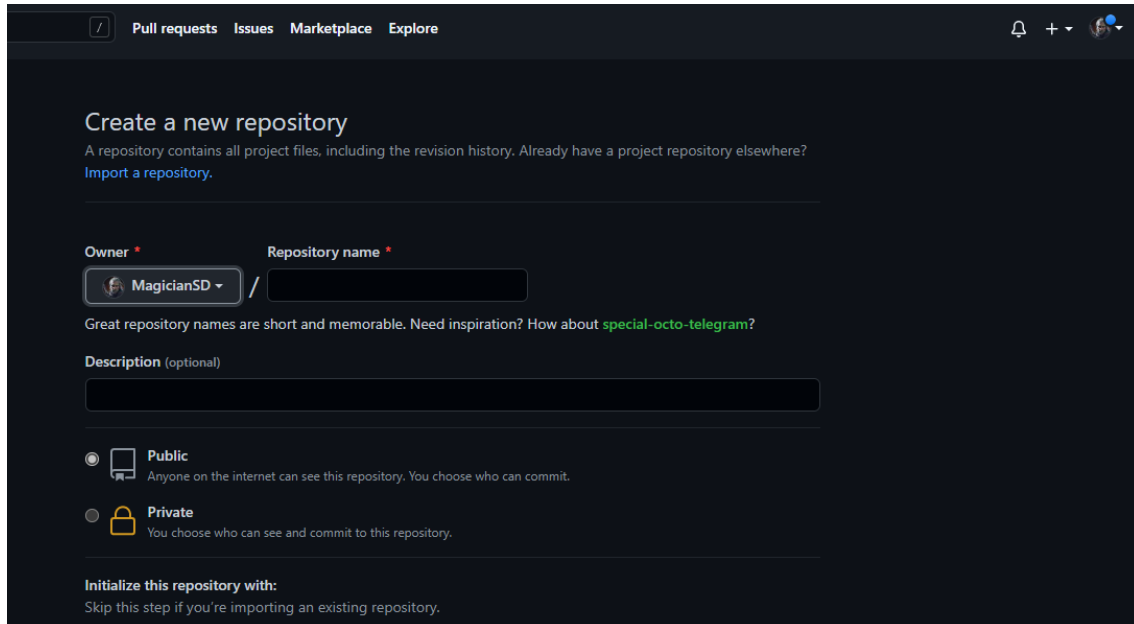
INICIANDO UM REPOSITÓRIO PELO GITHUB

Logo após configurar seu perfil, vá no mesmo menu e clique em *Your repositories*. Lá, clique no botão *New* para iniciar o processo de criação.



Página: Your repositories

Após clicar em New, você será direcionado para a página de criação do repositório, conforme imagem abaixo.



The screenshot shows the GitHub interface for creating a new repository. At the top, there are navigation links: Pull requests, Issues, Marketplace, and Explore. Below these, the main heading is 'Create a new repository'. A subtext explains that a repository contains all project files, including revision history, and offers a link to 'Import a repository'. The form includes fields for 'Owner' (set to 'MagicianSD') and 'Repository name'. A tip suggests using short and memorable names, with an example 'special-octo-telegram'. There is a 'Description (optional)' text area. Below this, two radio buttons allow selecting the repository's visibility: 'Public' (selected) and 'Private'. At the bottom, there is a section 'Initialize this repository with:' and a note to skip this step if importing an existing repository.

criação de repositório (opções)

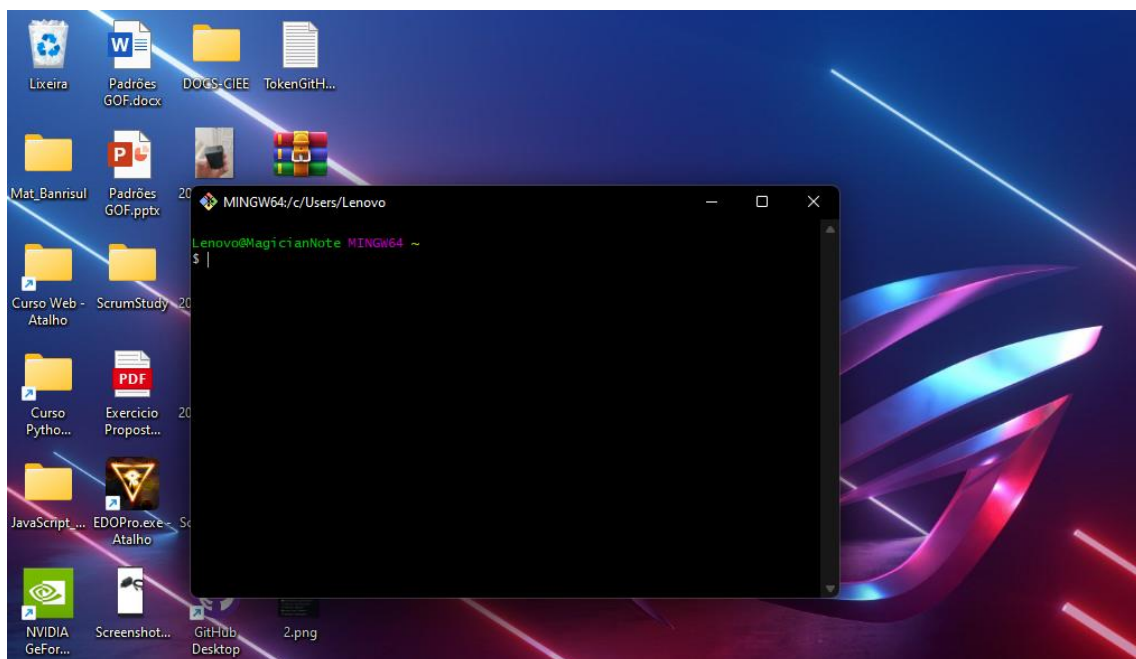
1. Informe o nome do seu repositório. Comumente será o nome do seu projeto. Apenas considere que se seu projeto tiver mais de uma solução, frontend e backend por exemplo, será interessante identificar.
2. Opcionalmente, você pode dar informações sobre o que se trata este repositório que está sendo criado, como objetivo de negócio ou as tecnologias usadas se for apenas para seu próprio portfólio.
3. Escolha *Public* se este repositório será aberto para a comunidade acessando o seu perfil. Caso selecione *Private*, somente você

poderá visualizá-lo. Esta opção poderá ser alterada depois, não se preocupe.

4. O README.md é um arquivo Markdown usado para documentar o projeto/produto. Se já possuir um, apenas deixe desmarcado.
5. O arquivo .gitignore serve para não subir arquivos compilados ou gerados automaticamente para o repositório economizando tempo de upload/download e espaço. O GitHub te dará opções de acordo com a tecnologia que você está desenvolvendo, selecione o seu, caso não tenha.
6. Quando estiver pronto, clique em *Create repository* para criar seu repositório.

CONFIGURANDO O GIT LOCAL

Agora, crie a pasta onde ficará armazenado localmente na sua máquina o repositório do seu projeto. Em seguida, clique com o botão direito do mouse e escolha a opção *Git Bash Here* e abrirá um terminal como o ilustrado abaixo.



Terminal do Git

Os comandos a seguir serão para que você verifique e configure, caso necessário, seus dados de acesso ao GitHub.

```
git config --list
```

Serve para verificar as configurações locais, como usuário e e-mail. Assim você terá certeza que as versões serão armazenadas com o autor correto.

```
git config --global user.name [nome do usuário entre aspas duplas]
```

Serve para definir o nome do usuário local.

```
git config --global user.email [email entre aspas duplas]
```

Serve para definir o e-mail do usuário local.

INICIANDO E ASSOCIANDO UM REPOSITÓRIO

Para criar um repositório local, basta executar o código abaixo na pasta raiz do projeto.

```
git init
```

Para associar ao repositório remoto, execute o comando abaixo.

```
git remote add origin [cole aqui a URL copiada na etapa anterior]
```

Associa o repositório local ao repositório do remoto com o apelido “**origin**”.

COMANDOS PARA BAIXAR DO REPOSITÓRIO REMOTO

```
git pull origin master
```

Considere utilizá-lo inicialmente se o .gitignore foi criado pelo GitHub, afinal este

comando serve para “baixar” do repositório remoto todos os arquivos que não existem no repositório local. Lembrando que “origin” é um *alias*.

```
git clone [cole aqui a URL copiada na etapa anterior]
```

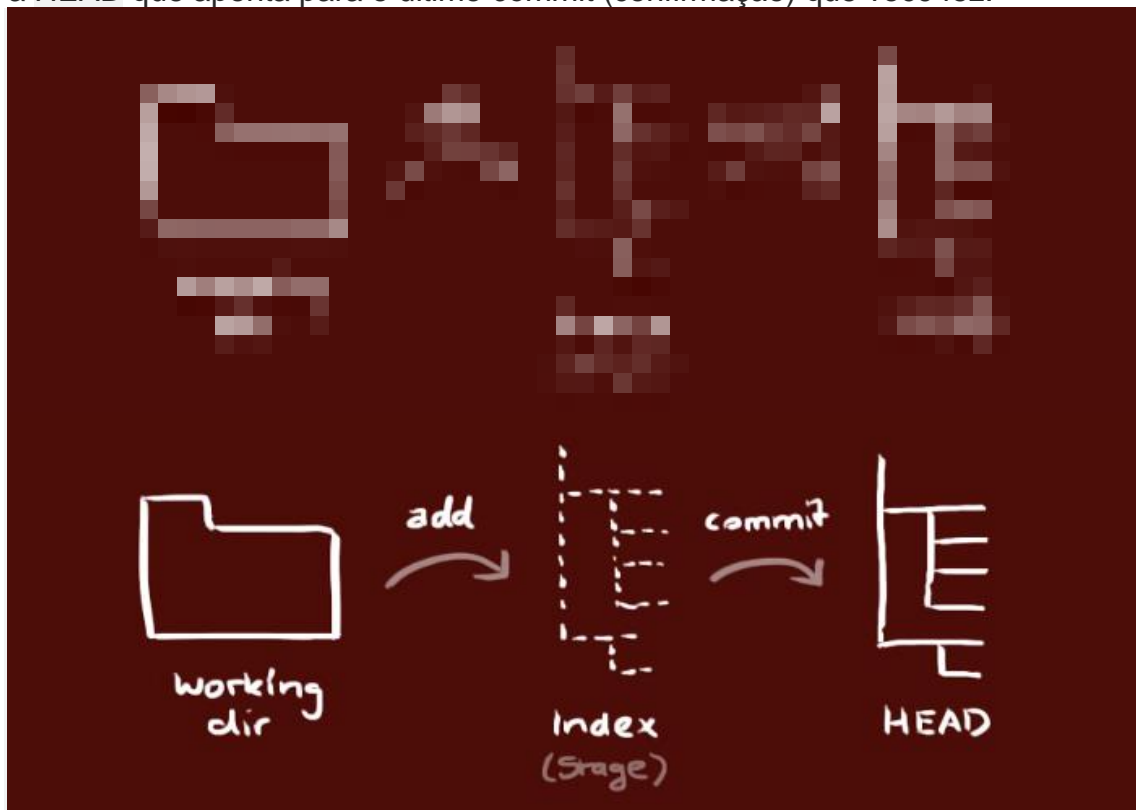
Cria uma cópia do trabalho em um repositório local.

```
git clone usuário@servidor:/caminho/para/o/repositório
```

Quando usar um servidor remoto, este será o comando.

ENTENDENDO O FLUXO DE TRABALHO

Seus repositórios locais consistem em três “árvores” mantidas pelo git. A primeira delas é sua **Working Directory** que contém os arquivos vigentes. A segunda é o **Index** que funciona como uma área temporária. Finalmente a **HEAD** que aponta para o último *commit* (confirmação) que você fez.



Fluxo de Trabalho

ADICIONAR E CONFIRMAR O INCREMENTO

Você pode propor mudanças (adicioná-las ao **Index**) usando os comandos:

```
git add <arquivo>
```

```
git add *
```

Este é o primeiro passo no fluxo de trabalho básico do Git. Para realmente confirmar estas mudanças, precisará realizar um *commit* através do comando

```
git commit -m "comentários das alterações"
```

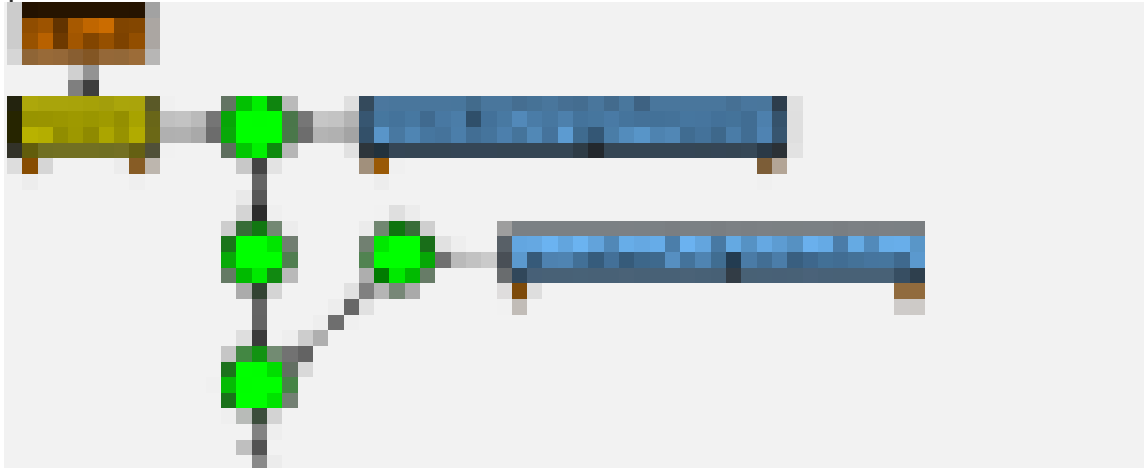
Agora o arquivo é enviado para o **HEAD**, mas ainda não para o repositório remoto.

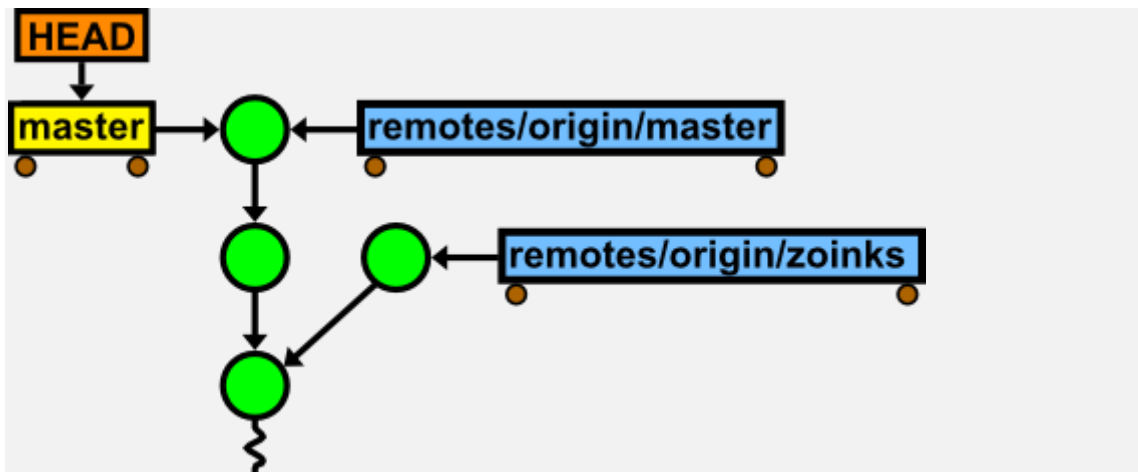
ENVIANDO ALTERAÇÕES

As alterações que estão no **HEAD** da sua cópia de trabalho local podem ser enviadas para o seu repositório remoto com o comando:

```
git push origin master
```

Altere *master* para qualquer ramo (*branch*) desejado enviando suas alterações para ele.

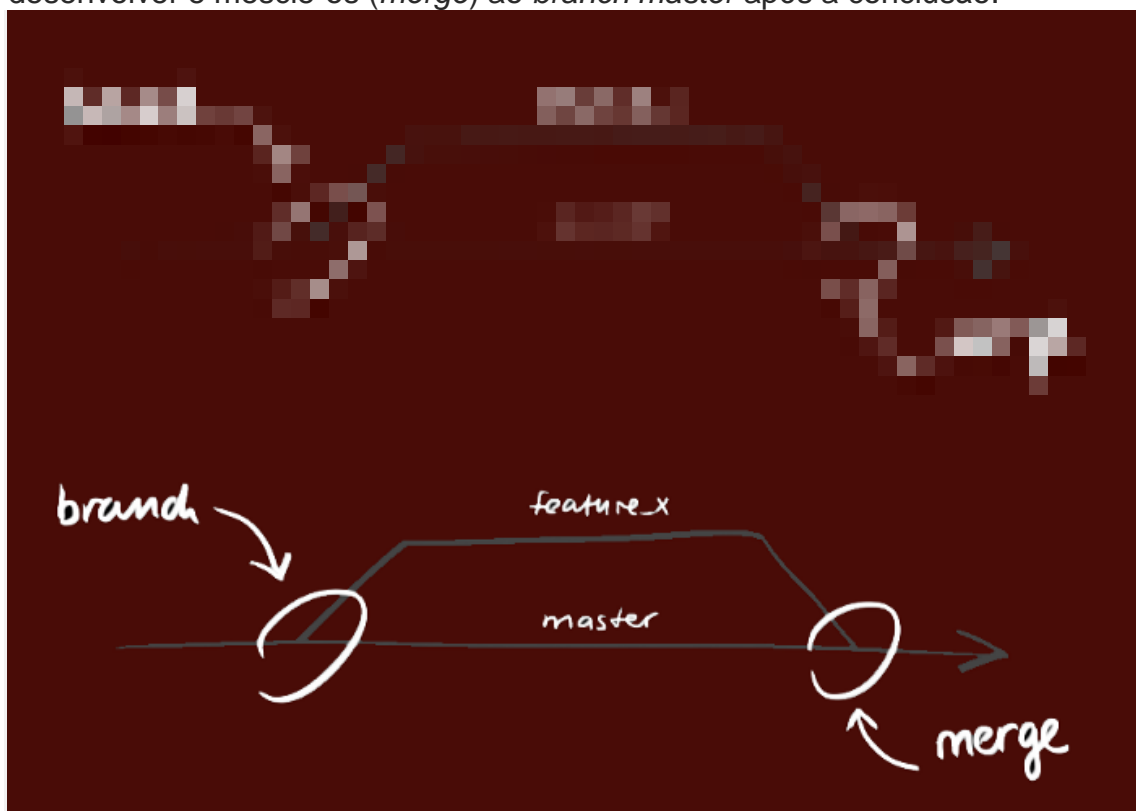




Como as alterações são posicionadas

RAMIFICAÇÕES

Branches (“ramos”) são utilizados para desenvolver funcionalidades isoladas umas das outras. O *branch master*, também chamado de tronco, é o *branch* “padrão” quando você cria um repositório. Use outros *branches* para desenvolver e mescle-os (*merge*) ao *branch master* após a conclusão.



Exemplo de ramificação

Crie um novo branch chamado “funcionalidade_x” e selecione-o pelo comando:

```
git checkout -b funcionalidade_x
```

Retorne para o master usando:

```
git checkout master
```

e remova o branch da seguinte forma:

```
git branch -d funcionalidade_x
```

Um branch *não está disponível a outros* a menos que você envie o branch para seu repositório remoto

```
git push origin <funcionalidade_x>
```

ATUALIZAR E MESCLAR ATUALIZAÇÕES

Para atualizar seu repositório local com a mais nova versão, execute o comando abaixo na sua pasta de trabalho para obter e fazer *merge* (mesclar) alterações remotas.

```
git pull
```

Para fazer merge de um outro branch ao seu branch ativo (exemplo *master*), execute o comando:

```
git merge <branch>
```

Ambos os casos o Git tentará fazer o *merge* das alterações automaticamente.

No entanto, nem sempre será possível e retornará em forma de conflitos. Você é responsável por fazer o *merge* destes *conflitos* manualmente editando os arquivos exibidos pelo Git. Depois de alterar, você precisa marcá-los como *merged* utilizando o comando abaixo.

```
git add <arquivo>
```

Antes de fazer o merge das alterações, você pode também pré-visualizar as diferenças usando o comando a seguir:

```
git diff <branch origem> <branch destino>
```

RÓTULO DE LIBERAÇÕES (*RELEASES*)

Uma prática muito utilizada e recomendada é criar rótulos para *releases* de *software*. Você pode criar um novo rótulo chamado *1.0.0* executando o comando a seguir:

```
git tag 1.0.0 1a2b3c4d2020
```

O rótulo 1a2b3c4d2020 representa os 12 primeiros caracteres do id de *commit* que você quer referenciar com seu rótulo. Você pode obter o id de commit com o comando abaixo.

```
git log
```

A única **restrição** para a identificação do rótulo é que ele precisa ser único.

SOBRESCREVER ALTERAÇÕES LOCAIS

Caso tenha errado algo e precise corrigir, você pode sobrescrever as alterações locais usando o comando abaixo.

```
git checkout -- <arquivo>
```

Isto substituirá as alterações na sua árvore de trabalho com o conteúdo mais recente no **HEAD**. Alterações já adicionadas ao **INDEX**, bem como novos arquivos serão mantidos.

Se ao invés disso você desejar remover todas as alterações e *commits* locais, recupere o histórico mais recente do servidor e aponte para seu *branch master* local usando os dois comandos abaixo.

```
git fetch origin
```

```
git reset --hard origin/master
```

RESOLVENDO PROBLEMAS

Quero desfazer tudo que eu fiz desde o último commit:

```
git clean -df
```

```
git checkout --
```

Preciso remover o último commit, porém mantendo os arquivos do jeito que estão:

```
git reset --soft HEAD~1
```

Preciso remover o último commit, inclusive as alterações nos arquivos:

```
git reset --hard HEAD~1
```

Preciso apagar o último commit no Github:

```
git push -f origin HEAD^:master
```

Quero mudar o meu repositório remoto “origin”:

```
git remote set-url origin [URL do repositório]
```

Entrei no VIM por engano. Como sair?

Tecla ESC, depois digite :q! e tecla ENTER

Quero alterar temporariamente os arquivos do projeto de modo a ficarem no estado do commit informado. ATENÇÃO: não podem haver modificações não commitadas no projeto:

```
git checkout <rótulo>
```

NOTA: para voltar ao último commit faça:

```
git checkout master
```

COMANDOS ÚTEIS E CURIOSIDADES

Interface gráfica padrão:

```
gitk
```

Usar saídas do Git coloridas:

```
git config color.ui true
```

Exibir log em apenas uma linha por commit:

```
git config format.pretty oneline
```

Fazer inclusões interativas:

```
git add -i
```