



**CNPq**



04 de Janeiro de 2018

# Introdução ao Git

Luana Nayara Pires Vilela

Laboratório Nacional de Luz Síncrotron, Campinas, Brazil

Neste tutorial são apresentados alguns aspectos básicos da utilização do sistema de controle de versões Git e da plataforma GitHub.

# 1. Sistema de controle de versões

Um sistema de controle de versões é um software utilizado para gerenciar diferentes versões de arquivos, mantendo o histórico de suas versões antigas e os registros de quem e quando manipulou estes arquivos.

Estes sistemas são utilizados principalmente para desenvolvimento de softwares, pois facilitam o compartilhamento e a recuperação de todas as versões dos arquivos, além de possibilitar que diversas pessoas trabalhem simultaneamente no mesmo conjunto de documentos.

## 2. Git e SVN

O Git e o Subversion (SVN) são dois exemplos de sistemas de controle de versões. Em ambos os sistemas os dados de todas as versões dos arquivos são salvos em banco de dados. O local onde este banco de dados fica armazenado é chamado de repositório.

A principal diferença entre os dois sistemas é que o SVN é um sistema servidor-cliente centralizado enquanto o Git é um sistema distribuído. Ou seja, no caso do SVN existe um repositório central localizado em um servidor a partir do qual os clientes podem copiar a versão desejada dos arquivos (para isso os computadores devem estar conectados a mesma rede). No caso do Git cada cópia local contém todas as versões dos arquivos daquele projeto, ou seja, cada cópia é um repositório completo. Assim, a maioria das ações do Git podem ser executadas offline enquanto que no SVN a maioria das ações depende da conexão com o servidor central, o que faz com que este último sistema tenha menor desempenho. Além disso, o Git é menos susceptível a falhas, pois se houver algum problema com o servidor, todos os clientes têm uma cópia completa do repositório que pode ser disponibilizada novamente.

## 3. GitHub

Outra vantagem de usar o Git é que existe uma plataforma de hospedagem de repositórios chamada [GitHub](#) que possibilita que qualquer pessoa conectada a internet possa ter acesso aos repositórios. Assim, os computadores não precisam estar conectados a uma mesma rede para que os arquivos sejam compartilhados. No entanto, como os repositórios no GitHub são públicos e qualquer pessoa pode ter acesso às informações contidas neles, esta ferramenta só deve ser utilizada para códigos abertos e informações não sigilosas.

O grupo de imãs possui uma página no GitHub chamada [lnls-ima](#) que atualmente contém repositórios para cada um dos imãs desenvolvidos para o Sirius, assim como alguns repositórios com código-fonte de softwares de medições magnéticas e análises de medidas.

O objetivo deste tutorial é facilitar o uso do Git e do GitHub para aqueles que não estão familiarizados com essas ferramentas para que todos os softwares de medidas e análises implementados pelo grupo de imãs sejam disponibilizados na página do grupo no GitHub.

Para conseguir criar e modificar repositórios do grupo de imãs é preciso criar uma conta no [GitHub](#), ser incluído na organização lnls-ima (isso deve ser feito por algum administrador da organização) e instalar o Git no seu computador.

## 4. Instalação do Git

### Instalação no Windows

Recomenda-se a instalação da interface gráfica do GitHub para Windows, encontrada [aqui](#).

Se você pretende usar apenas a interface gráfica não é necessário instalar o Git. No entanto, se você quiser usar a linha de comando precisará instalar a versão do Git para o Windows, que pode ser encontrada [aqui](#).

### Instalação no Linux

Se você usa uma distribuição baseada em Debian, pode instalar o Git usando o apt-get:

```
$ sudo apt-get install git-all
```

Para outras distribuições veja as instruções [aqui](#).

## 5. Configuração do Git

Após a instalação do Git é preciso configurar o seu nome de usuário e endereço de e-mail. Esta configuração pode ser feita utilizando a interface gráfica (GitHub Desktop) ou utilizando a linha de comando.

### Configuração com a interface gráfica

Ao abrir pela primeira vez a interface do GitHub Desktop clique em "Sign into GitHub.com". Entre com o seu usuário e senha do GitHub. Em seguida a tela de configuração aparecerá. Preencha o seu nome e e-mail e clique no botão "Continue".

### Configuração com a linha de comando

Para configurar o nome de usuário e e-mail utilize os seguintes comandos:

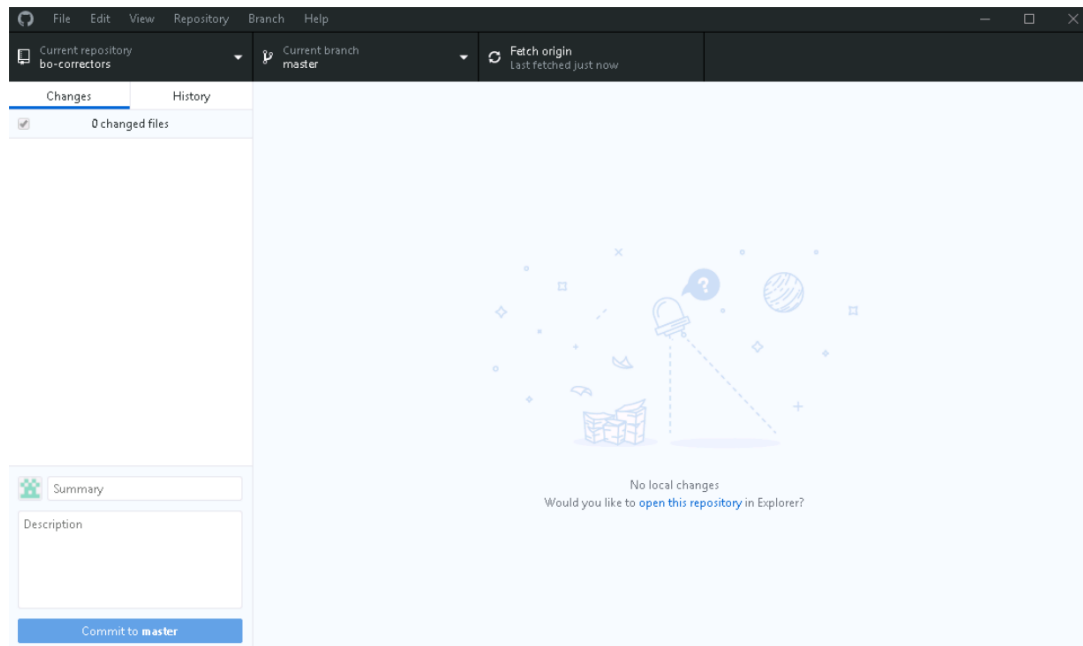
```
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```

## 6. Usando o Git

Nesta seção são apresentados alguns dos comandos básicos necessários para a utilização do Git. Para entender um pouco melhor como utilizar esta ferramenta pode ser feito o curso [Learn Git](#) disponível gratuitamente na plataforma Codecademy. Para aprofundar seu conhecimento sobre o assunto recomenda-se o livro [Pro Git](#), que contém uma descrição detalhada dos comandos e recursos disponíveis no Git.

### Interface gráfica

A interface do GitHub Desktop, mostrada na figura a seguir, contém as funcionalidades básicas do Git e possibilita o uso deste sistema sem a linha de comando. Na [seção seguinte](#) é apresentado um exemplo de utilização desta interface.



## Linha de comando

A seguir é apresentada uma breve descrição de alguns dos principais comandos do Git. Ao utilizar os comandos abaixo, o texto entre <> deve ser substituído pelos nomes apropriados.

### **git init**

Inicializa um novo repositório no local onde o comando foi executado.

### **git clone git@github.com:lnls-ima/<repository>.git**

Cria uma cópia do repositório do GitHub no local onde o comando foi executado.

### **git pull**

Traz as mudanças feitas no repositório remoto para o repositório local.

### **git status**

Mostra quais arquivos foram modificados desde a última vez que o banco de dados local foi atualizado.

### **git diff**

Mostra as mudanças feitas nos arquivos.

### **git add <filename>**

Seleciona o arquivo <filename> para que as mudanças feitas neste arquivo sejam incluídas no banco de dados local na próxima atualização.

### **git commit -m "<message>"**

Atualiza o banco de dados local. O termo <message> deve ser substituído por uma mensagem explicando quais as mudanças realizadas desde a última atualização.

### **git push**

Envia as mudanças feitas no repositório local para o repositório remoto.

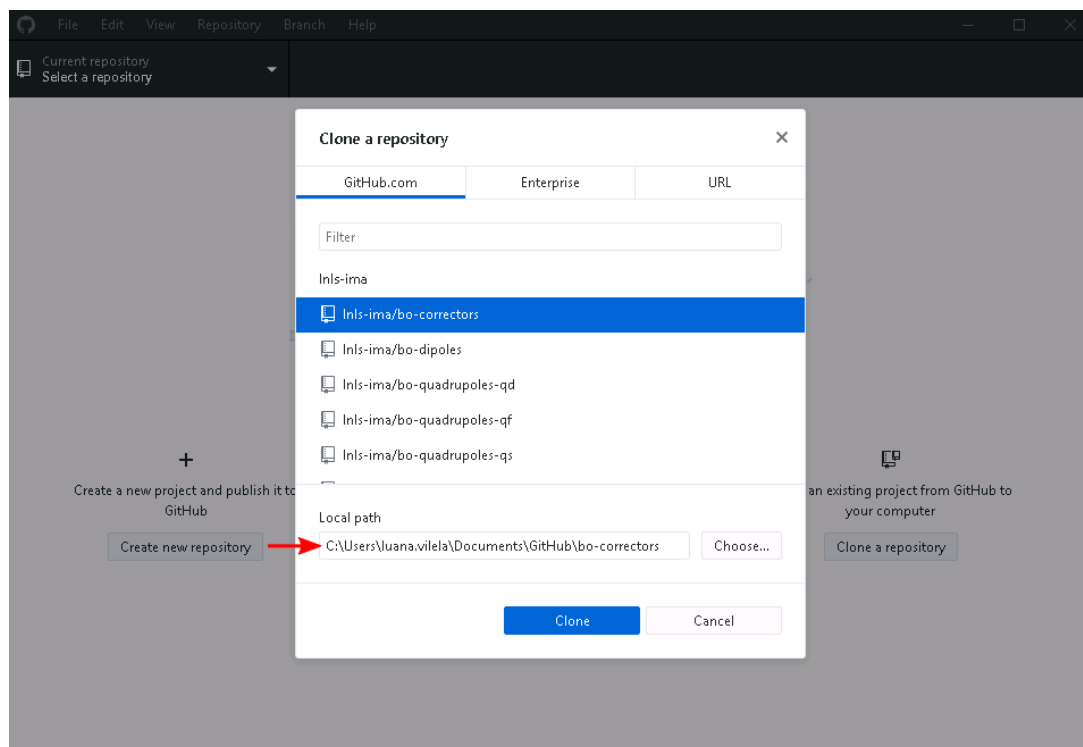
Um resumo de todos os comandos do Git pode ser encontrado [aqui](#).

## 7. Exemplo

Nesta seção é mostrado como modificar um arquivo de um repositório do GitHub utilizando a interface gráfica ou a linha de comando. Neste exemplo o arquivo *README.md* do repositório chamado *bo-correctors* será modificado.

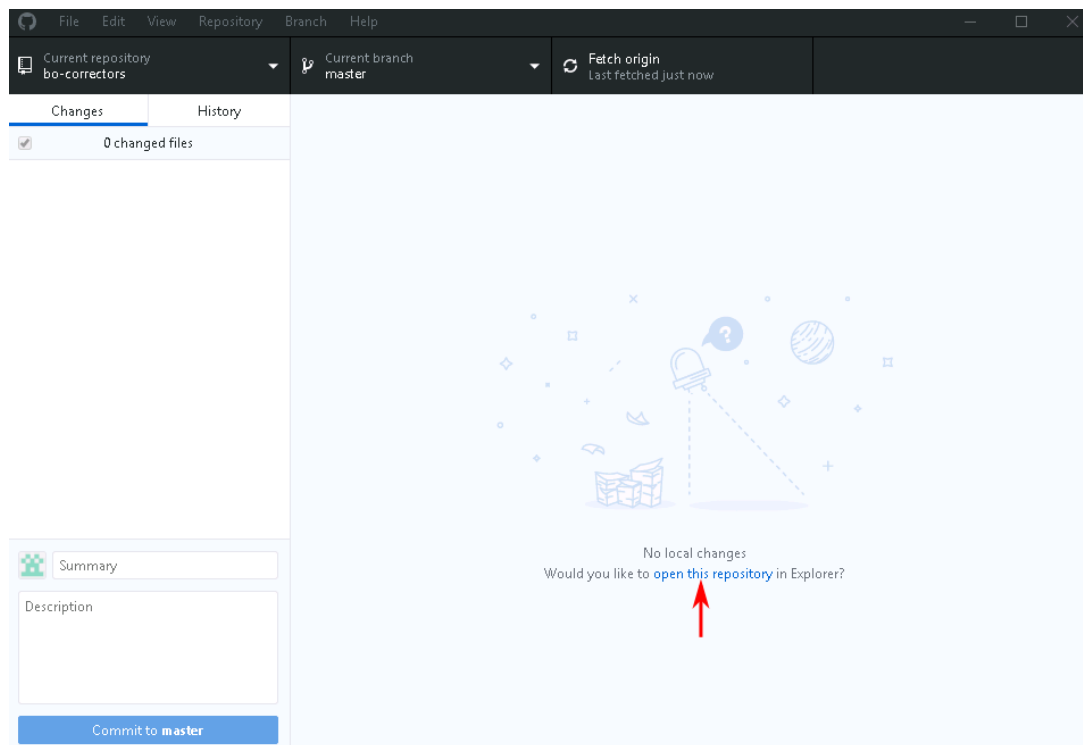
### Interface gráfica

Inicialmente é preciso clonar o repositório *bo-correctors* e assim ter cópia dos arquivos no seu computador. Isto pode ser feito clicando no botão "Clone a repository" da página inicial do GitHub Desktop ou a partir do menu "File" → "Clone repository". A seguinte tela será mostrada:

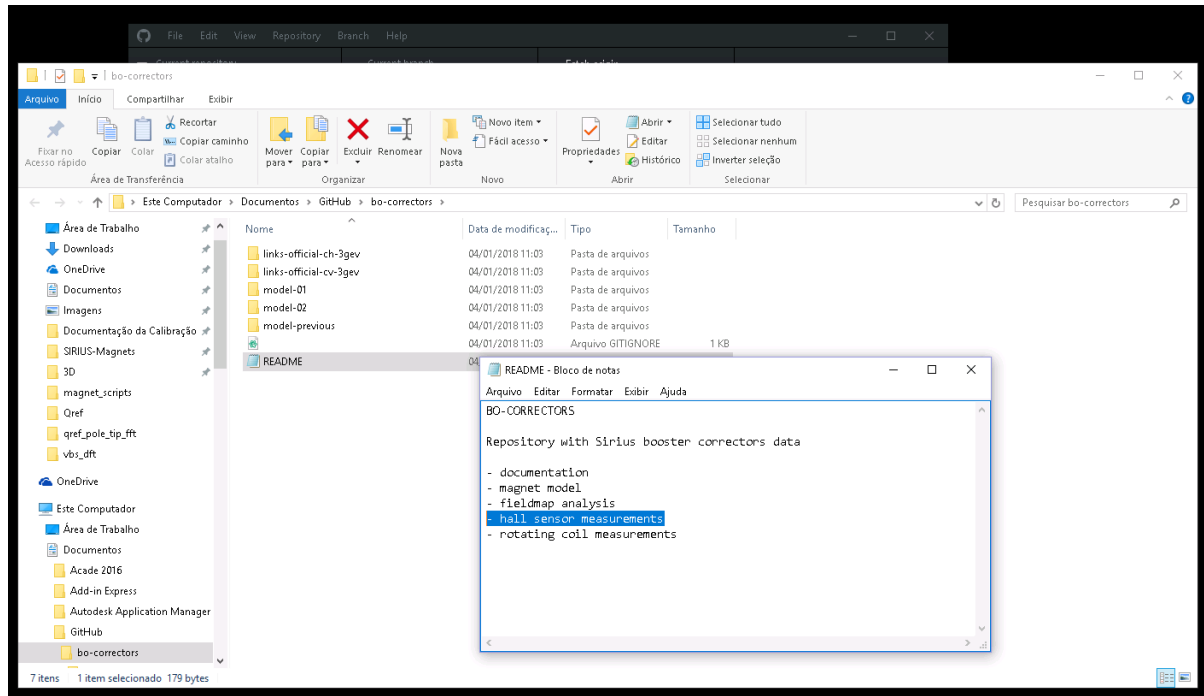


Selecione o repositório desejado e o local onde a cópia deste repositório será armazenada ("Local Path"). Em seguida clique em "Clone".

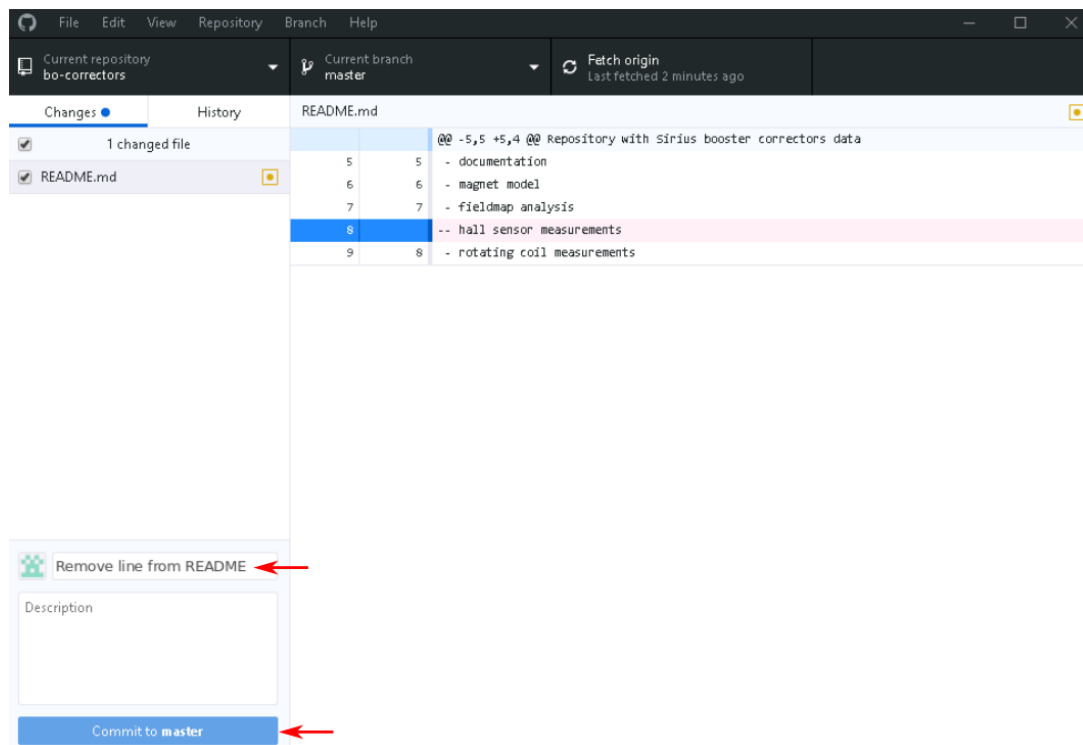
Inicialmente os arquivos do repositório local serão iguais aos arquivos do repositório remoto do GitHub. Para abrir a pasta que contém os arquivos clique em "open this repository", conforme indicado na figura abaixo:



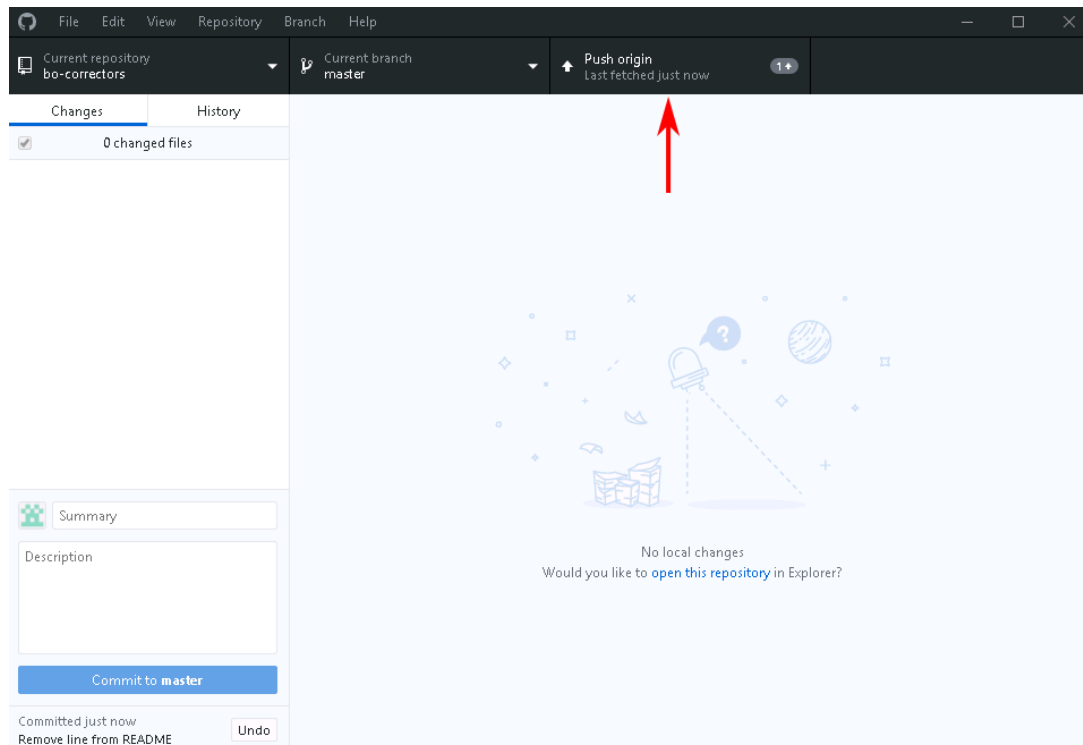
Faça as mudanças desejadas. Neste caso, uma das linhas do arquivo *README.md* será apagada, conforme mostrado na figura a seguir:



A interface do GitHub mostra a lista dos arquivos modificados e as mudanças feitas em cada arquivo. Para salvar a nova versão dos arquivos no banco de dados local adicione uma mensagem explicando as alterações no campo "Summary" e no campo "Description" (opcional) e em seguida clique em "Commit to master":

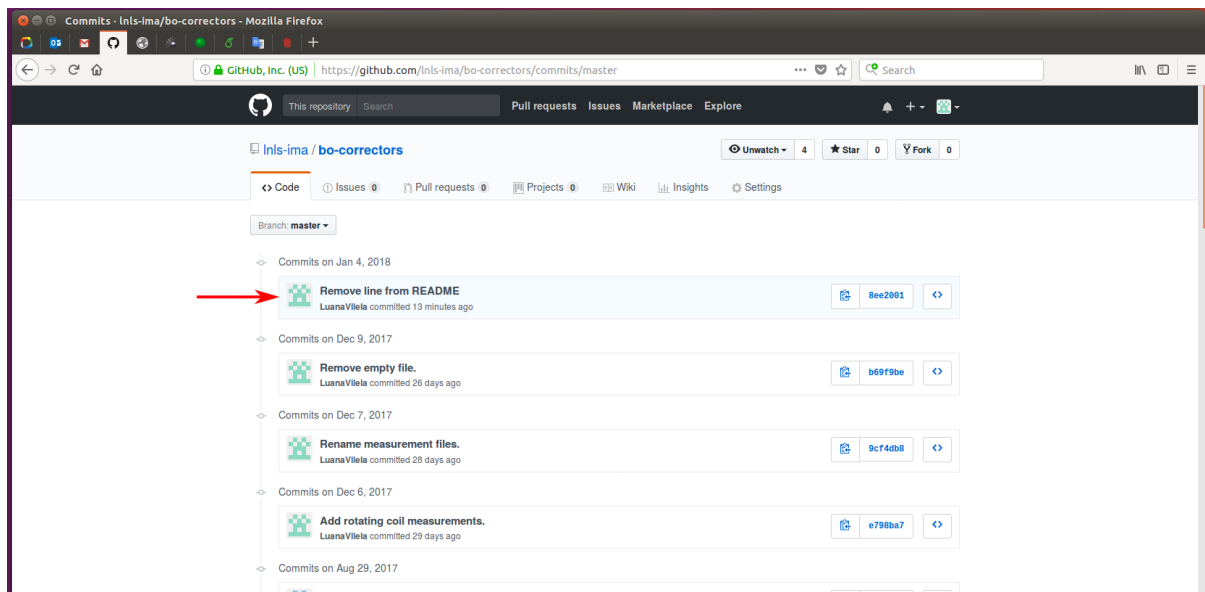


Agora que as mudanças estão salvas no repositório local elas podem ser enviadas para o repositório remoto. Para isso clique no botão "Push origin" mostrado abaixo:



Pronto! Agora todas as pessoas que utilizam este repositório poderão atualiza-lo clicando no botão "Fetch origin" e em seguida no botão "Pull origin" e o arquivo será

corrigido em todas as cópias. O commit realizado e a mensagem descritiva são mostrados no repositório do GitHub:



## Linha de comando

Para modificar o arquivo primeiramente é necessário obter uma cópia do repositório usando o comando **clone**:

```
$ git clone git@github.com:lnls-ima/bo-correctors.git
Cloning into 'bo-correctors'...
remote: Counting objects: 5470, done.
remote: Compressing objects: 100% (3143/3143), done.
remote: Total 5470 (delta 86), reused 3186 (delta 41), pack-reused 2200
Receiving objects: 100% (5470/5470), 65.10 MiB | 6.26 MiB/s, done.
Resolving deltas: 100% (213/213), done.
Checking connectivity... done.
```

Em seguida são feitas as mudanças desejadas nos arquivos. Para verificar quais arquivos foram modificados utiliza-se o comando **status**:

```
$ git status
On branch master
Your branch is up-to-date with 'origin/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:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
```

Para selecionar o arquivo para que as mudanças realizadas nele sejam incluídas no banco de dados local utiliza-se o comando **add**:

```
$ git add README.md
```

Em seguida utiliza-se o comando **commit** para atualizar o banco de dados local:



```
$ git commit -m "Remove line from README"
[master 6b31236] Remove line from README
1 file changed, 1 deletion(-)
```

Por fim, as mudanças feitas no repositório local são enviadas para o repositório remoto utilizando o comando **push**:

```
$ git push
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 302 bytes | 0 bytes/s, done.
Total 3 (delta 2), reused 1 (delta 1)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To git@github.com:lnls-ima/bo-correctors.git
54c32c7..6b31236 master -> master
```

## 8. Links uteis

[Git for Windows](#) Versão do Git para Windows.

[GitHub Desktop](#) Interface gráfica do GitHub para Windows.

[Git CheatSheet](#) Planilha com o resumo dos comandos do Git.

[Pro Git](#) Livro bem detalhado sobre o Git.

[Learn Git](#) Curso disponível no Codecademy.