Lucas Ferreira Rodrigues 122272

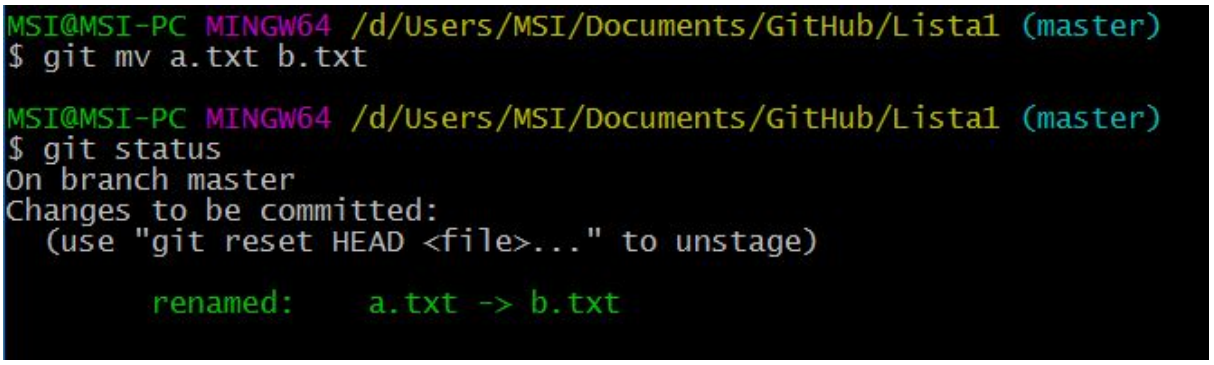
1- O comando para receber a versao do git é **git --version**

```
MSI@MSI-PC MINGW64 /d/Users/MSI/Documents/GitHub
$ git --version
git version 2.8.3.windows.1
```

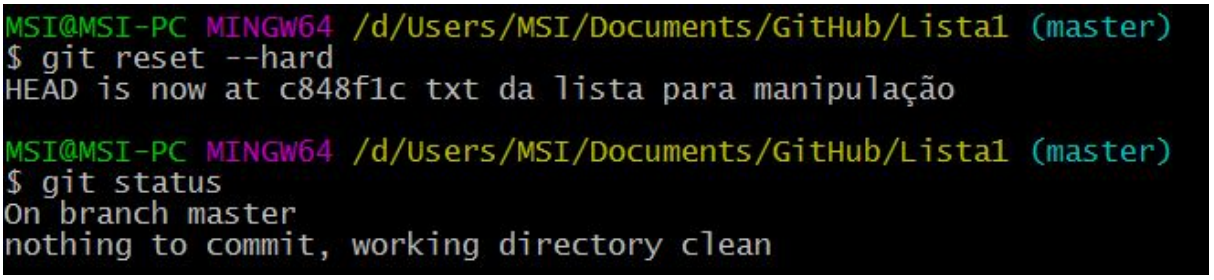
2a- o comando **git config -l** exibe as configurações do git

```
MSI@MSI-PC MINGW64 /d/Users/MSI/Documents/GitHub
$ git config -l
core.symlinks=false
core.autocrlf=true
core.fscache=true
color.diff=auto
color.status=auto
color.branch=auto
color.interactive=true
help.format=html
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/ce
rts/ca-bundle.crt
diff.astextplain.textconv=astextplain
rebase.autosquash=true
credential.helper=manager
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.required=true
user.name=LucasFerreiraRodrigue
user.email=lucas_f_rodrigues@hotmail.com
```

2b- o comando **git mv a.txt b.txt** renomeia o arquivo a.txt para b.txt



2c- o comando **git reset --hard** reverte tudo feito antes do ultimo commit



2d- o comando **git log -27** de acordo com a documentação exibe 27 logs de commits

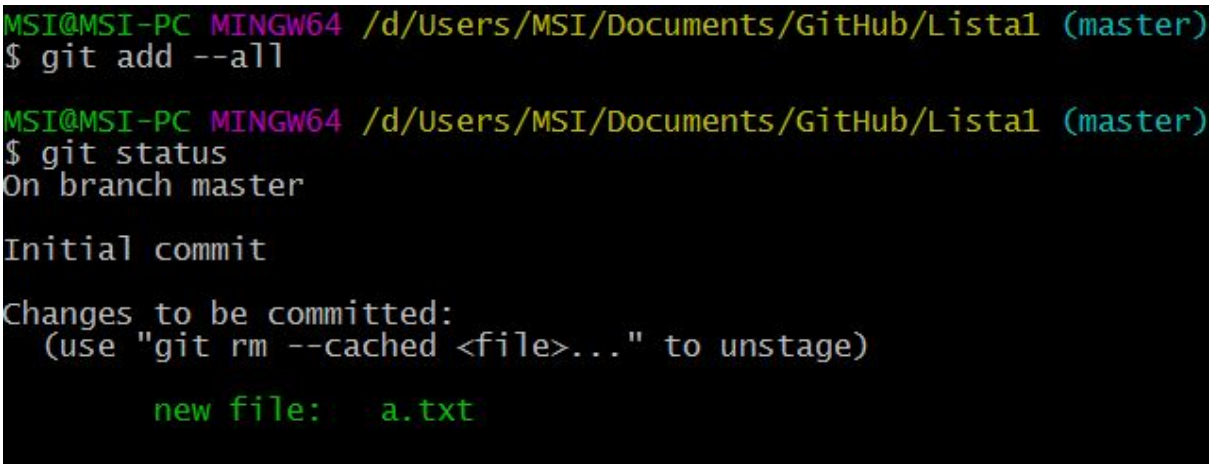
2e- o comando **git help** exibe

```
MSI@MSI-PC MINGW64 /d/Users/MSI/Documents/GitHub/Lista1 (master)
$ git help
usage: git [--version] [--help] [-C <path>] [-c name=value]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path
]
           [-p | --paginate | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone      Clone a repository into a new directory
   init       Create an empty Git repository or reinitialize an existing
one

work on the current change (see also: git help everyday)
   add        Add file contents to the index
   mv         Move or rename a file, a directory, or a symlink
   reset      Reset current HEAD to the specified state
   rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
   bisect     Use binary search to find the commit that introduced a bug
   grep       Print lines matching a pattern
   log        Show commit logs
   show       Show various types of objects
   status     Show the working tree status

grow, mark and tweak your common history
   branch     List, create, or delete branches
   checkout   Switch branches or restore working tree files
   commit     Record changes to the repository
   diff       Show changes between commits, commit and working tree, etc
   merge      Join two or more development histories together
   rebase     Reapply commits on top of another base tip
   tag        Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   fetch      Download objects and refs from another repository
   pull       Fetch from and integrate with another repository or a local
 branch
   push       Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
```

2f- o comando **git help reset** abre o arquivo git-reset.html

```
MSI@MSI-PC MINGW64 /d/Users/MSI/Documents/GitHub/Lista1 (master)
$ git help reset
Launching default browser to display HTML ...
```

2g- o comando **git add --all** adiciona todos os arquivo modificados para o commit



2h- o comando -git add -u adiciona todos os arquivos modificados porém ignora arquivos “untracked”

3- os comandos seriam **git add <file name>/-all** e depois git commit -m “comentário do commit”

4- **git status**

5- **git status**

6- **git commit -m “comentario”**

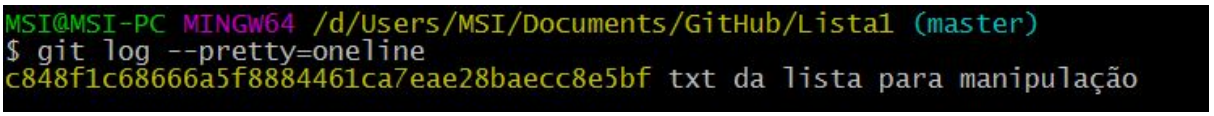
7- **git checkout --test.txt**

8- para que um determinado diretório seja ignorado basta ir no arquivo .**gitgnore.txt** e adicionar em uma linha em branco **“diretório/”** por exemplo caso queria que a pasta build seja ignorada basta colocar **build/**

9- Em caso de exclusão de arquivos não desejados basta dar **git pull --rebase**

10- **git clone <url>**

11- **git log -pretty=oneline**



12- **.git/config**

13- **git init <nome>**

```
MSI@MSI-PC MINGW64 /d/Users/MSI/Documents/GitHub
$ git init Lista1
Initialized empty Git repository in D:/Users/MSI/Documen
ts/GitHub/Lista1/.git/
```

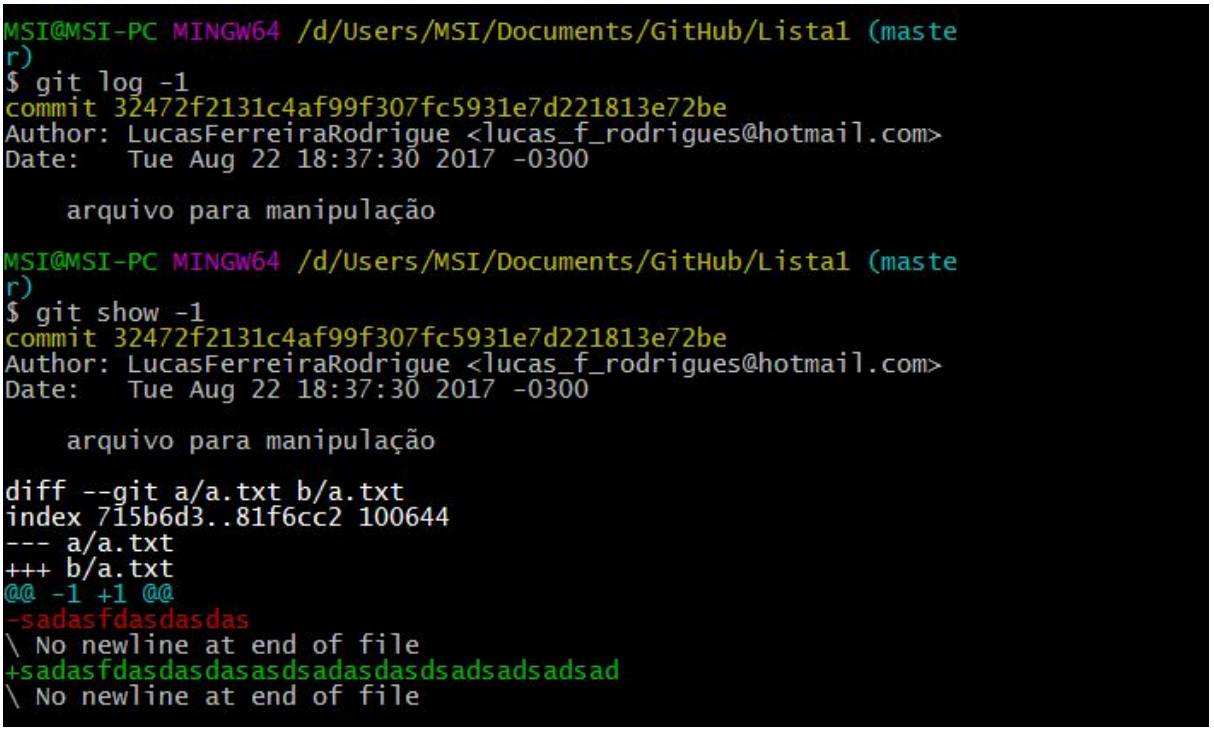
14- ao realizar o comando **git init** ele cria o diretório **.git** caso o mesmo já exista ele recria o diretório **.git**

15- **git add -u**

```
MSI@MSI-PC MINGW64 /d/Users/MSI/Documents/GitHub/Lista1 (master)
$ git add -u

MSI@MSI-PC MINGW64 /d/Users/MSI/Documents/GitHub/Lista1 (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   Novo Documento de Texto (2).txt
        modified:   Novo Documento de Texto (3).txt
        modified:   Novo Documento de Texto.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        aaaa.txt
        asdasdasdasd.txt
```

16- SHA1(Secure Hash Algorithm 1) é um método de criptografía de dados, o propósito de métodos de criptografia em geral é garantir a integridade do arquivo bem como sua segurança

17- Na hora de passar o parâmetro para o commit deseja o parâmetro **1** sempre retorna o último commit



18- Não pois o comando **git add -u** ignora arquivos “untracked”

19- O comando **git reset --soft HEAD~1** não modifica os arquivos no index ou na árvore mas reseta o HEAD para o números de commits especificados, já o comando **git reset --hard** volta todos os arquivos para o estado em que estavam no ultimo commit

20- **git clean -f**

21- **.gitgnore**

22- No arquivo **.gitgnore** em uma linha vazia, em branco, preencha-a com **.class** assim todos os arquivos no repositório da extensão .class vao ser ignorados

23- **git clone** https://github.com/jquery/jquery.git

24- O comando **git shortlog -sne** mostra e ordena pela quantidade de commits e os usuários/email de quem os fez

```
MSI@MSI-PC MINGW64 /d/Users/MSI/Documents/GitHub/jquery (master)
$ git shortlog -sne
  1714  John Resig <jeresig@gmail.com>
   657  Timmy Willison <4timmywil@gmail.com>
   579  Dave Methvin <dave.methvin@gmail.com>
   336  Jörn Zaefferer <joern.zaefferer@gmail.com>
   332  Julian Aubourg <aubourg.julian@gmail.com>
   315  Rick Waldron <waldron.rick@gmail.com>
   267  Oleg Gaidarenko <markelog@gmail.com>
   262  Richard Gibson <richard.gibson@gmail.com>
   250  Brandon Aaron <brandon.aaron@gmail.com>
   230  Michał Gołębiowski-Owczarek <m.goleb@gmail.com>
   200  Ariel Flesler <aflesler@gmail.com>
    85  Mike Sherov <mike.sherov@gmail.com>
    71  Colin Snover <github.com@zetafleet.com>
    67  David Serduke <davidserduke@gmail.com>
    59  Yehuda Katz <wycats@gmail.com>
    55  Corey Frang <gnarf37@gmail.com>
    47  Louis-Rémi Babé <lrbabe@gmail.com>
    35  Anton Matzneller <obhvsbypqghgc@gmail.com>
    34  Scott González <scott.gonzalez@gmail.com>
```

25- O comando **git remote -v**

```
MSI@MSI-PC MINGW64 /d/Users/MSI/Documents/GitHub/jquery (master)
$ git remote -v
origin  https://github.com/jquery/jquery.git (fetch)
origin  https://github.com/jquery/jquery.git (push)
```

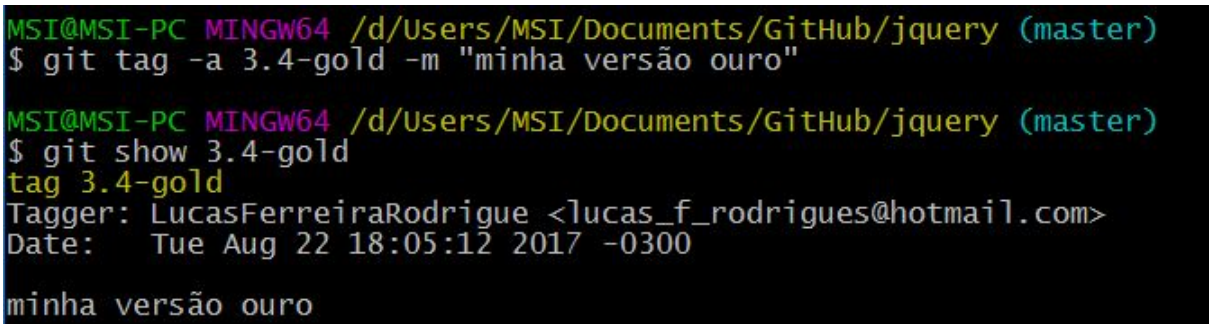
26- **git tag**
git tag -l/--list

27- O comando **git tag --list 2.0.***

```
MSI@MSI-PC MINGW64 /d/Users/MSI/Documents/GitHub/jquery (master)
$ git tag --list 2.0.*
2.0.0
2.0.0-beta3
2.0.0b1
2.0.0b2
2.0.1
2.0.2
2.0.3
```

28- O comando **git tag -a 3.4-gold -m "minha versão ouro"** cria a tag 3.4-gold o parâmetro -m faz com que essa tag tenha uma mensagem

29-



30- O comando **git push origin 3.4-gold** tem como efeito o push da tag ja que o comando push por si só sem parâmetros não envia as tags, caso queira enviar todas o uso do comando **git push --tags** é indicado

31- O comando **git commit --amend** reutiliza o corpo do último commit na realização de um novo

32- O arquivo x.txt volta a ser untracked

33- O comando **git checkout --a.txt** pega o arquivo a.txt do repositório remoto sobrescrevendo o atual a.txt

34- O comando **git reset HEAD x.txt** faz com que o x.txt retorne ao estado em que o txt estava no último commit e o torna untracked, já o comando **git checkout --a.txt** faz com que o txt seja sobrescrito com a versão remota podendo ter alterações não esperadas