

# Usando Comandos do Git

## Git config

```
MINGW64/C:/Users/Aluno/AtividadeProa
$ git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager-core
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
user.name=SaraLopes7
user.email=sara.r.a.lopes07@gmail.com
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.sshcommand=ssh
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager-core
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
user.name=SaraLopes7
user.email=sara.r.a.lopes07@gmail.com
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true
core.symlinks=false
core.ignorecase=true
```

## Git init

## Git status

gist.github.com/leocomelli/2545add34e4fec21ec16

- Geral: Normalmente armazenado no diretório do usuário do Sistema Operacional. O arquivo que possui a lista dos arquivos/diretórios a serem ignorados por todos os repositórios deverá ser declarado conforme citado acima. O arquivo não precisa ter o nome de .gitignore.
- Por repositório: Deve ser armazenado no diretório do repositório e deve conter a lista dos arquivos/diretórios que devem ser ignorados apenas para o repositório específico.

### Repositório Local

**Criar novo repositório**

```
git init
```

**Verificar estado do repositório**

```
git status
```

**Adicionar arquivos**

Adicionar um arquivo em específico

```
git add meu_arquivo.txt
```

Adicionar um diretório em específico

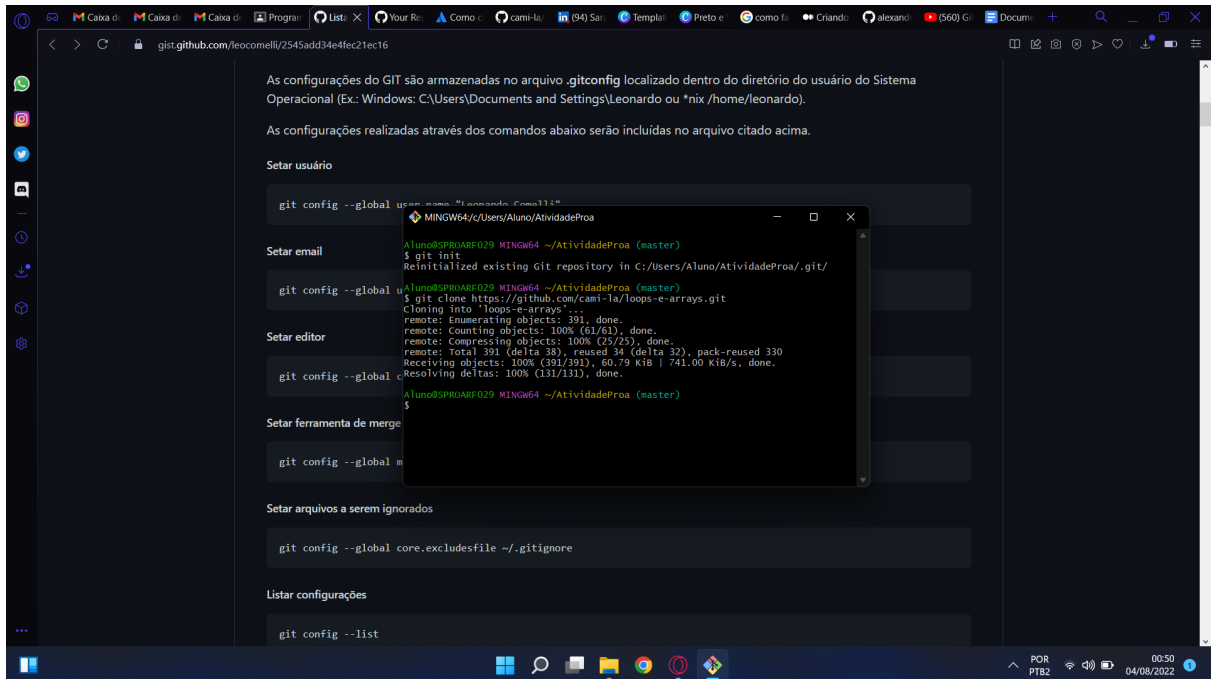
```
git add meu_diretorio
```

```
MINGW64/C:/Users/Aluno/AtividadeProa
$ git init
Initialized empty Git repository in C:/Users/Aluno/AtividadeProa/.git/
$ git status
On branch master

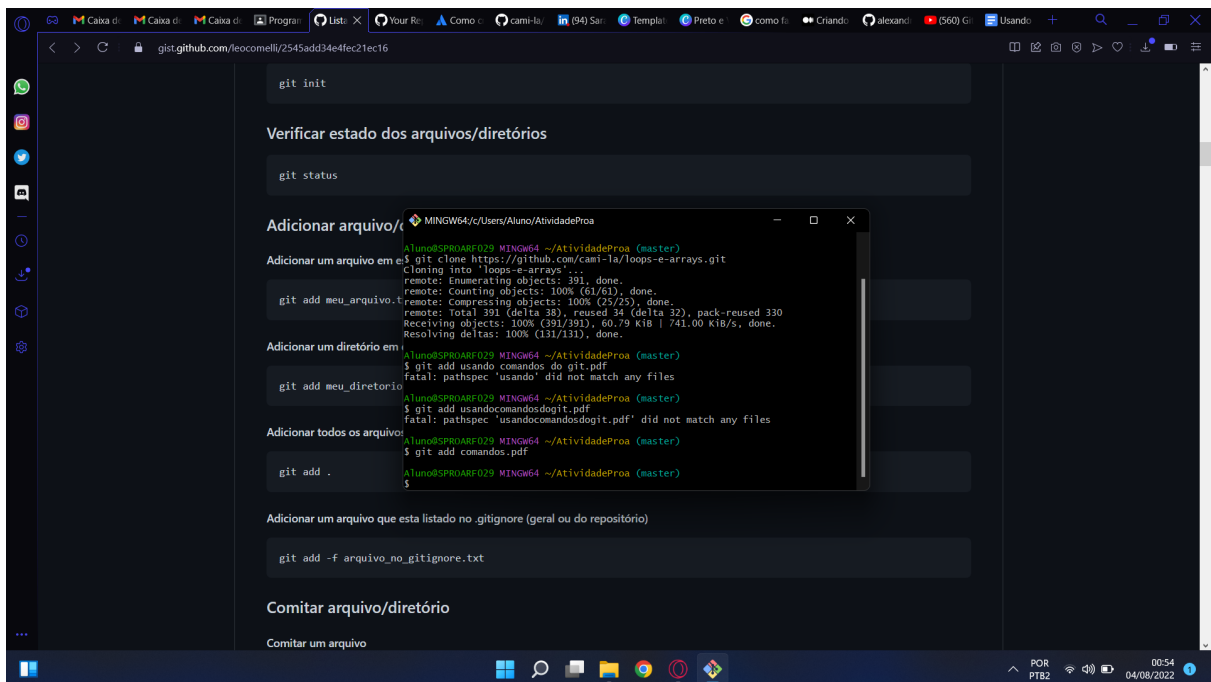
No commits yet

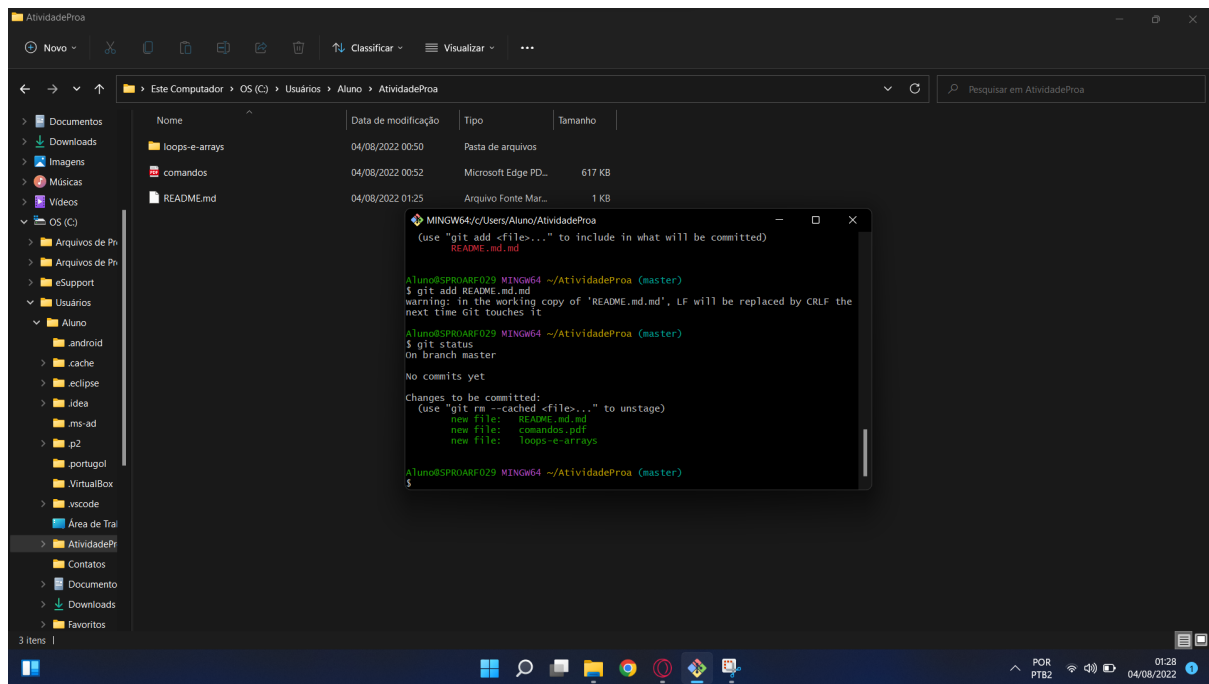
nothing to commit (create/copy files and use "git add" to track)
```

## Git clone

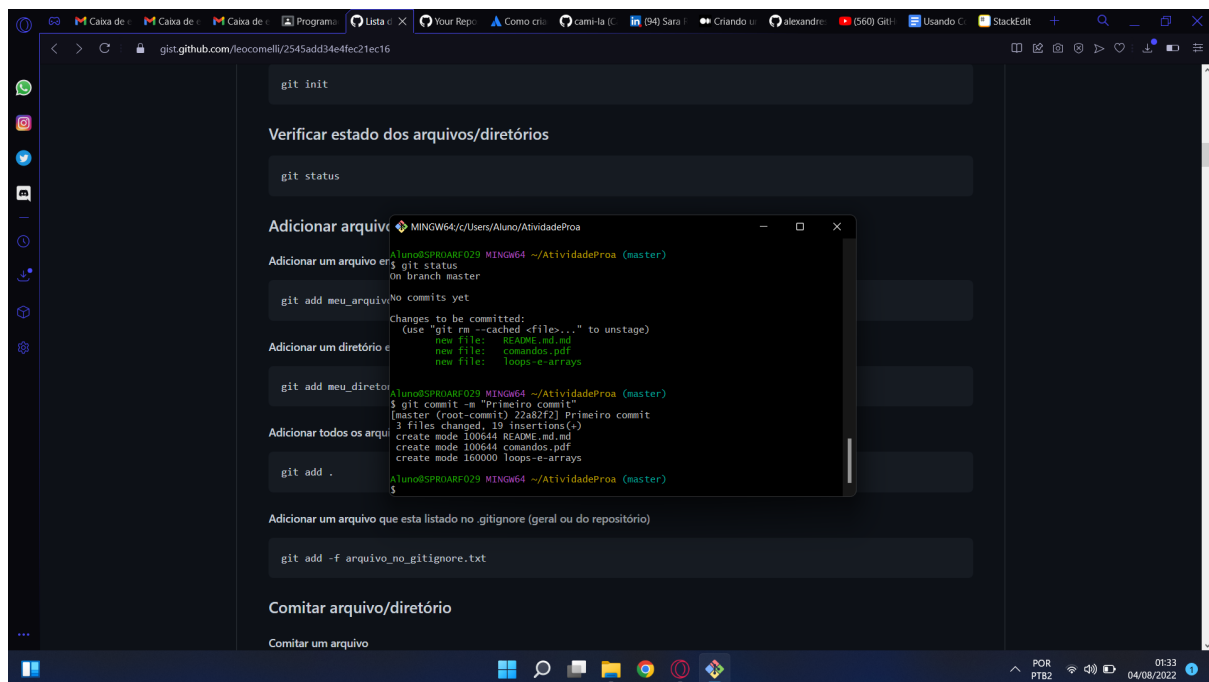


## Git add

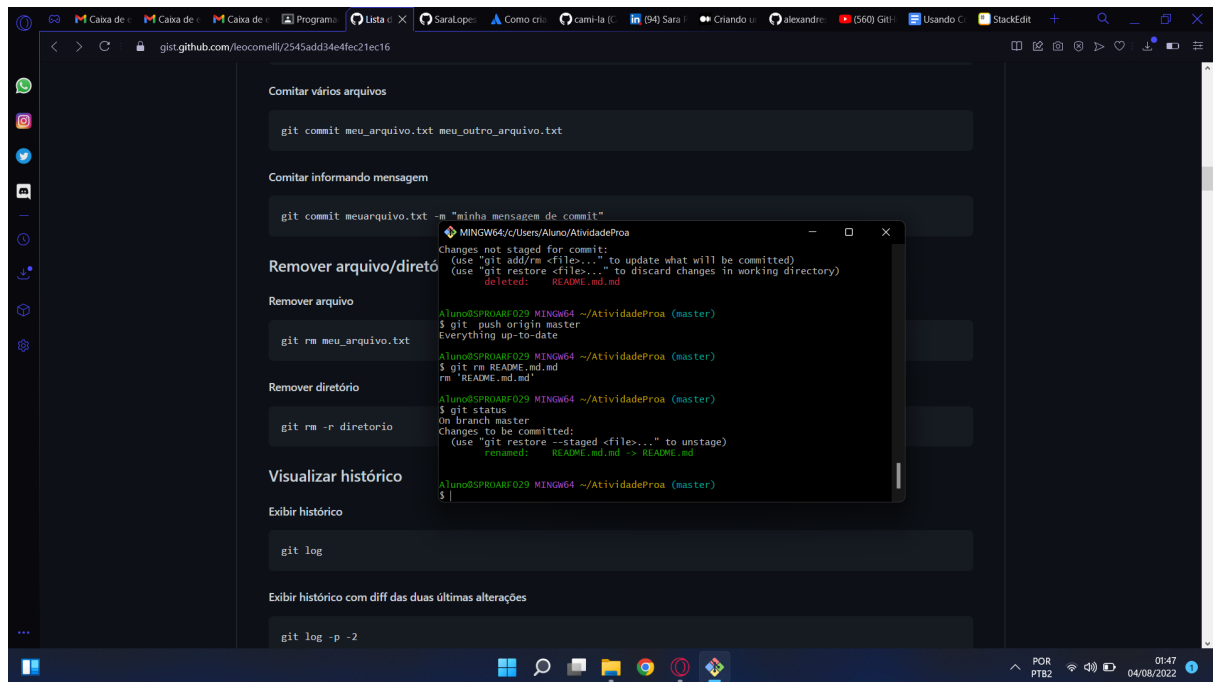




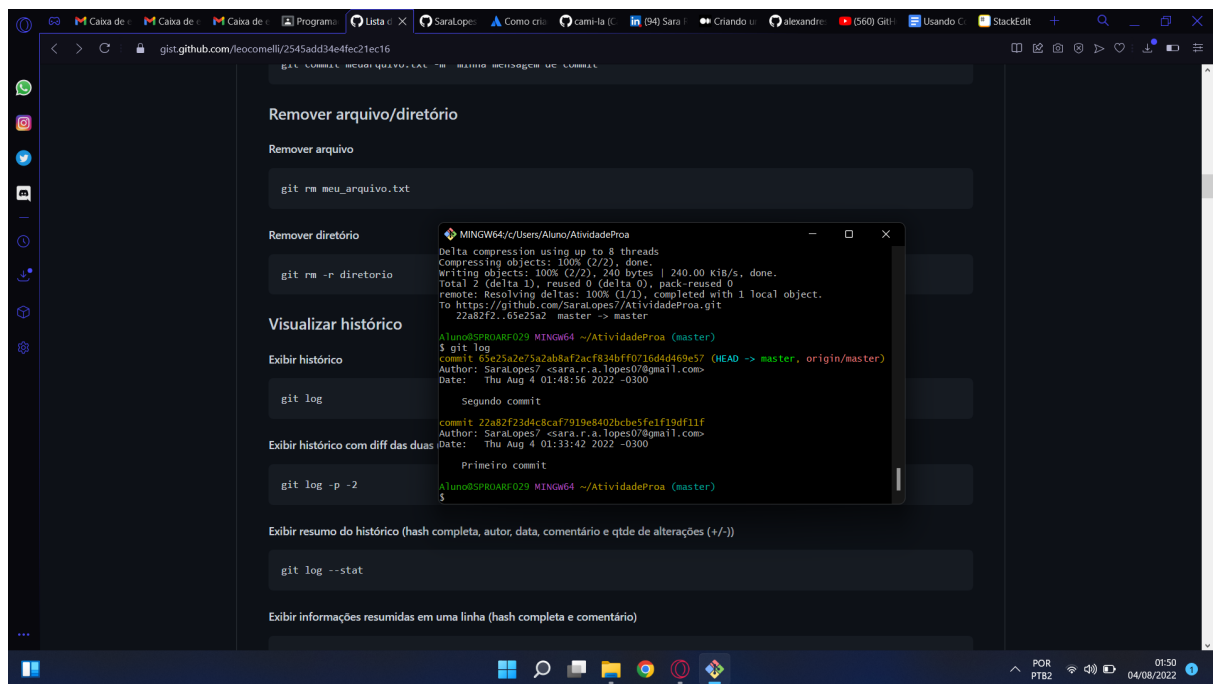
## Git commit



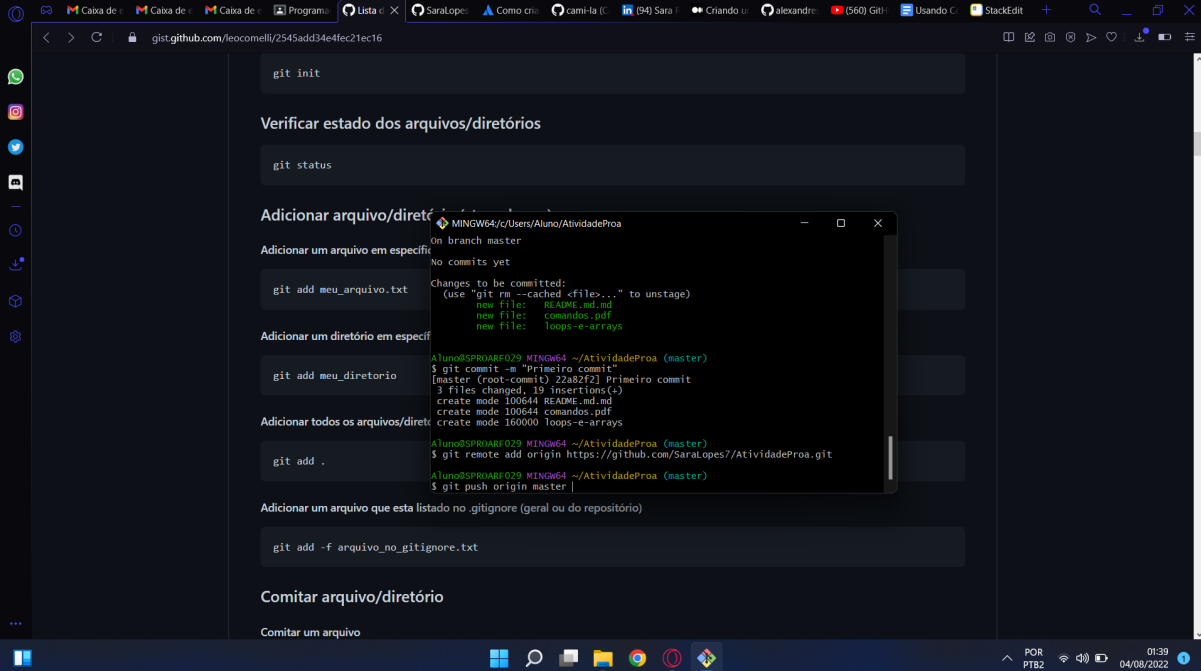
## Git rm



## Git log



## Git remote e push



The screenshot shows a GitHub tutorial page titled "Git remote e push" with a dark theme. The page lists several commands for initializing a repository, adding files, and pushing to a remote. A terminal window is overlaid on the page, showing the execution of these commands in a Windows command prompt.

**Verificar estado dos arquivos/diretórios**

```
git status
```

**Adicionar um arquivo/diretório**

Adicionar um arquivo em específico

```
git add meu_arquivo.txt
```

Adicionar um diretório em específico

```
git add meu_diretorio
```

Adicionar todos os arquivos/diretórios

```
git add .
```

Adicionar um arquivo que esta listado no .gitignore (geral ou do repositório)

```
git add -f arquivo_no_gitignore.txt
```

**Comitar arquivo/diretório**

```
git commit -m "Primeiro commit"
```

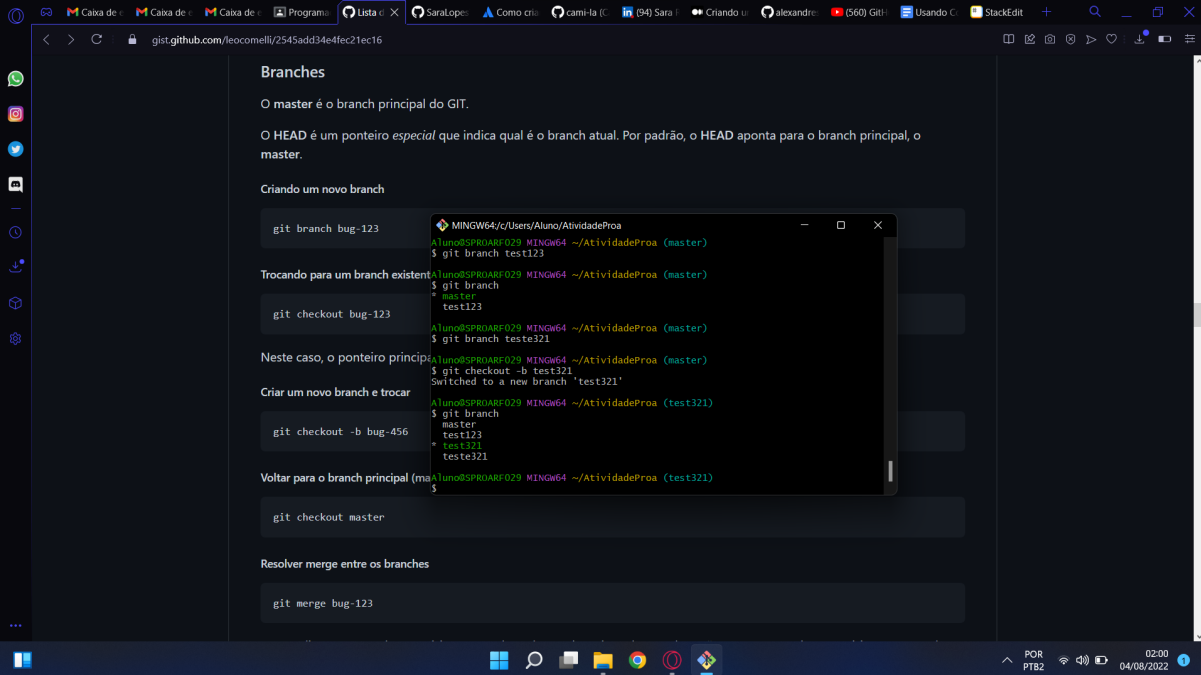
**Comitar um arquivo**

```
git commit -m "Primeiro commit"
```

The terminal window shows the following output:

```
Aluno@SPROARF029 MINGW64 ~/AtividadeProa (master)
$ git commit -m "Primeiro commit"
[master (root-commit) 22a82f2] Primeiro commit
3 files changed, 19 insertions(+)
create mode 100644 README.md
create mode 100644 comandos.pdf
create mode 160000 loops-e-arrays
```

## Git branch



The screenshot shows a GitHub tutorial page titled "Git branch" with a dark theme. The page explains the concept of branches and provides commands for creating, switching, and merging branches. A terminal window is overlaid on the page, showing the execution of these commands in a Windows command prompt.

**Branches**

O master é o branch principal do GIT.

O HEAD é um ponteiro *especial* que indica qual é o branch atual. Por padrão, o HEAD aponta para o branch principal, o master.

**Criando um novo branch**

```
git branch bug-123
```

**Trocando para um branch existente**

```
git checkout bug-123
```

Neste caso, o ponteiro principal HEAD aponta para o branch bug-123.

**Criar um novo branch e trocar**

```
git checkout -b bug-456
```

**Voltar para o branch principal (master)**

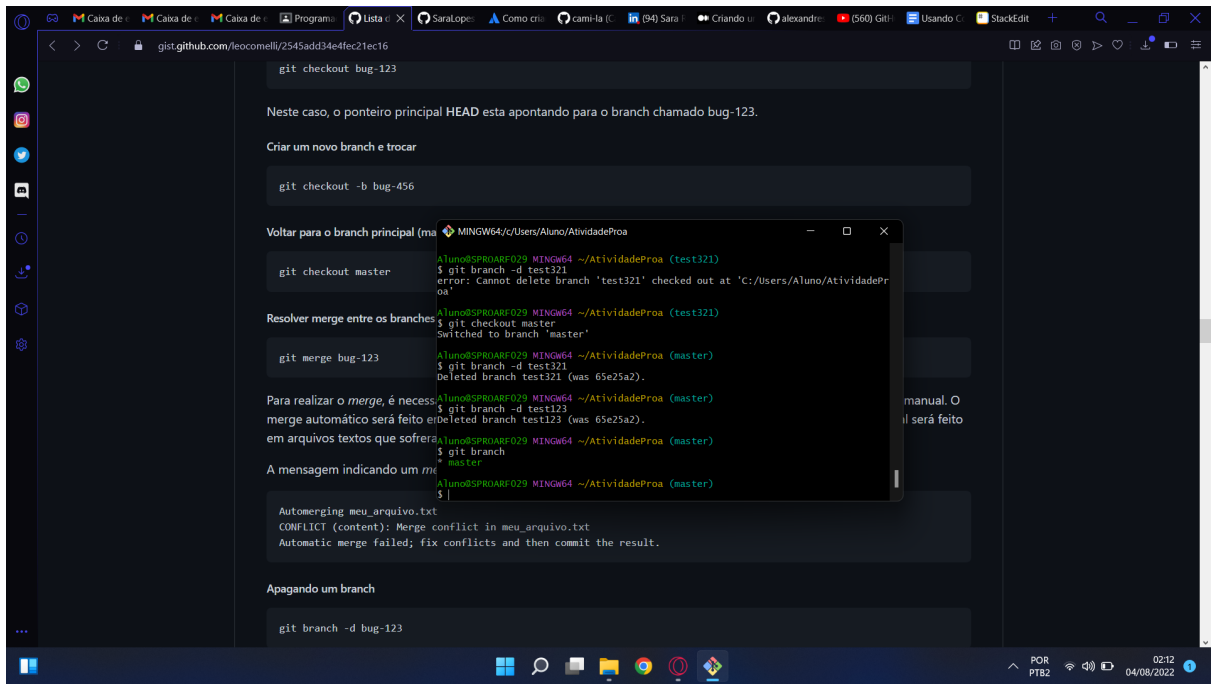
```
git checkout master
```

**Resolver merge entre os branches**

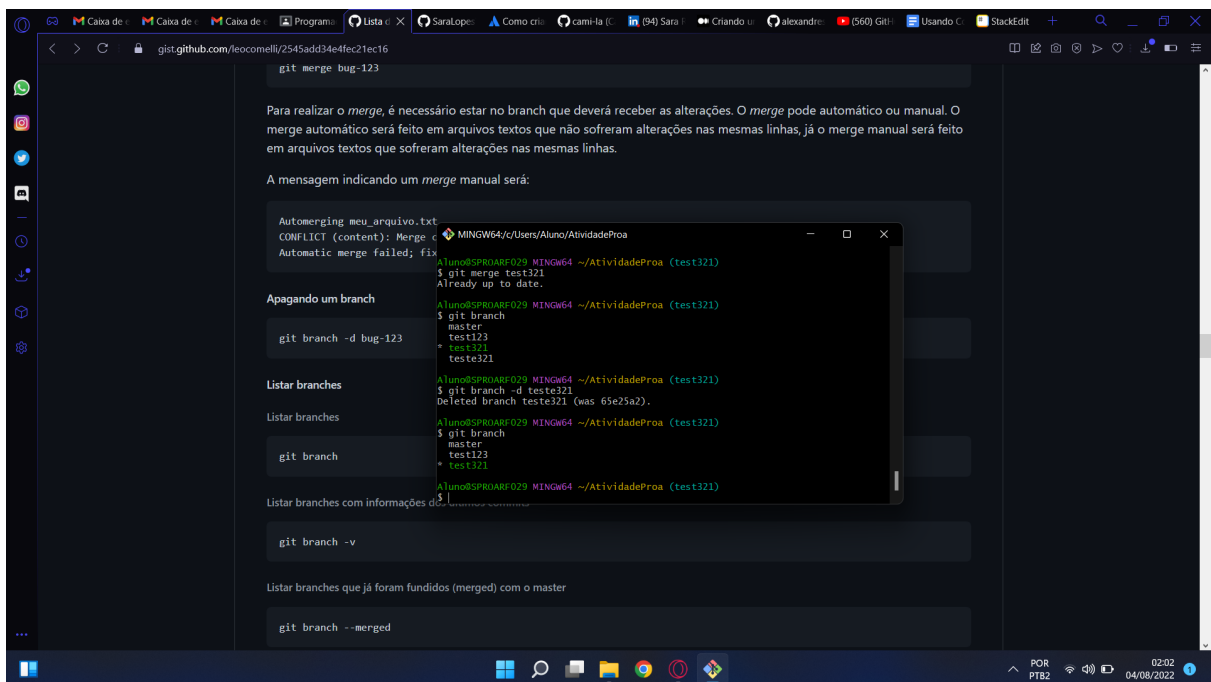
```
git merge bug-123
```

The terminal window shows the following output:

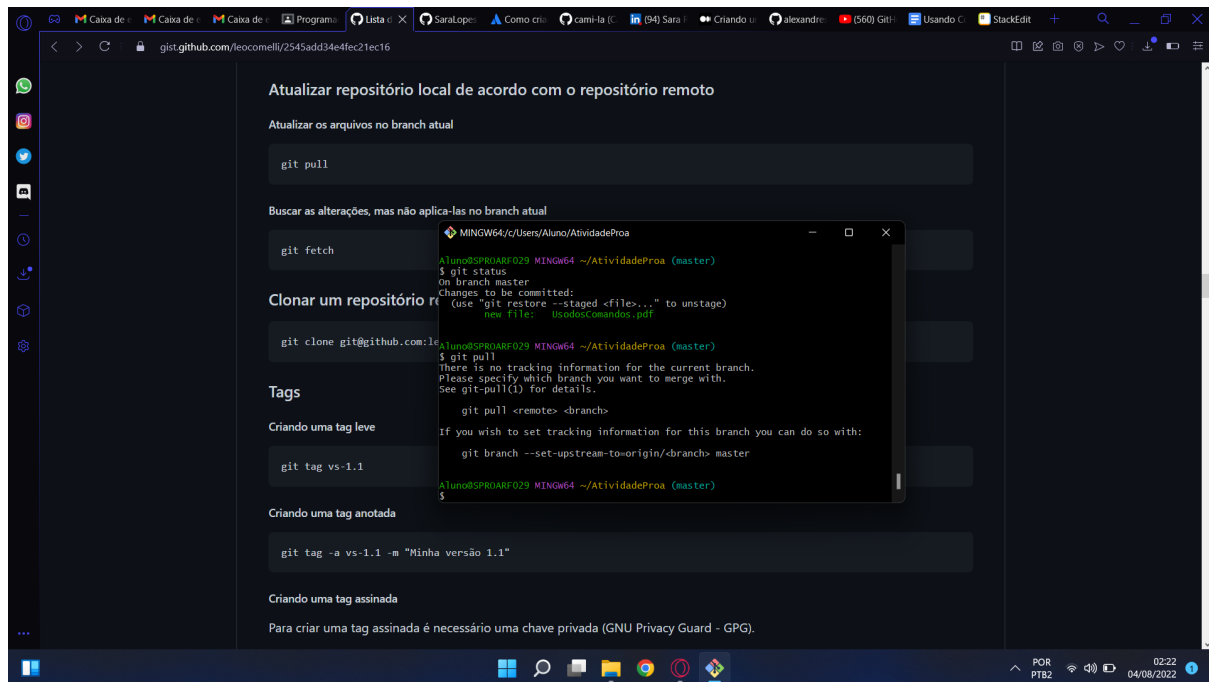
```
Aluno@SPROARF029 MINGW64 ~/AtividadeProa (master)
$ git branch bug-123
Aluno@SPROARF029 MINGW64 ~/AtividadeProa (master)
$ git checkout bug-123
Switched to a new branch 'bug-123'
Aluno@SPROARF029 MINGW64 ~/AtividadeProa (bug-123)
$ git checkout -b bug-456
Switched to a new branch 'bug-456'
Aluno@SPROARF029 MINGW64 ~/AtividadeProa (bug-456)
$ git checkout master
Switched to branch 'master'
Aluno@SPROARF029 MINGW64 ~/AtividadeProa (master)
$ git merge bug-123
```



## Git Merge



## Git pull



## Git diff

