```
################################ Instalar git ###################################

# Debian, Ubuntu, Mint
        $ sudo apt-get install git

# Fedora
        $ sudo dnf install git

# ArchLinux
        $ sudo pacman -S git

# Windows
        https://git-scm.com/download/

################################### Termos ######################################

* Nota - Sempre que num comando vejam (< >) é para retirar.
* Nota - Quando num comando vejam aspas ("") ou pelicas (’’) é para as manter.

* git - Ferramenta de controlo de versões.
* GitHub - Serviço de alojamento de repositórios remotos, utiliza a ferramenta
git.
* repositório - Onde são guardados os ficheiros e o histórico de suas alterações.
Organizado em branches (ramos). Termo ’repo’ é usado como abreviatura.
* repositório local - Repositório existente nas nossas maquinas.
* repositório remoto - Repositório existente no GitHub.
* origin - Designação normalmente dada ao repositório remoto.
* master -  Branch principal de um repositório.
* branch - Excluindo o master, é uma cópia de outro branch que segue que
um histórico independente. Cada branch tem o seu histórico de alterações
relativos aos ficheiros que contem. Os mesmos ficheiros podem estar associados
a diferentes branches.
* branch remoto - Branch de um repositório remoto.
* branch local - Branch de um repositório local.
* tracking - Capacidade de um branch local seguir um branch local.
* up stream tracking - Capacidade de um branch local seguir um branch remoto.
* stage area - Pode-se pensar nisto como um buffer, cache, onde são adicionados
os ficheiros alterados.
* commit - Salva o estado dos ficheiros, anteriormente adicionados à stage area,
na diretoria git, criando um ponto de restauro no histórico do branch.
* .gitignore - onde adicionamos o caminho dos ficheiros ou pastas que não
queremos que o git monitorize.

################################ Configurar git #################################

# Para seguir estas instruções é necessária uma conta no GitHub.
# Definir nome de utilizador.
        $ git config --global user.name "nome_utilizador"

# Definir email, deve ser o mesmo que está associado à conta do GitHub.
        $ git config --global user.email "email_do_utilizador"
```

```
# Opcional, estes comandos servem para que a password seja
# guardada durante, por exemplo, 3600 segundos (1h). -> MANTER AS PELICAS <-
    # LINUX
        $ git config --global credential.helper 'cache --timeout <segundos>'
    # MAC & Windows
        https://confluence.atlassian.com/bitbucketserver/
        permanently-authenticating-with-git-repositories-776639846.html


############################## Criar repositório novo ##############################

# Iniciar um repositório local
        $ git init

# Associar o repositório local a um repositório remoto <endereço> (sem < >).
# <nome> (sem < >) indica o nome que vamos dar a este repositório remoto,
# normalmente é origin.
# Exemplo: git remote add origin https://github.com/jfp52843/Projeto_LTI.git
        $ git remote add <nome> <endereço>

###################### Clonar um repositório já existente ######################

# Clonar repo já existente em <endereço>.
# Exemplo: git clone https://github.com/jfp52843/Projeto_LTI.git
        $ git clone <endereço>

############## Adicionar ficheiros à stage area do branch atual ################

# Adicionar ficheiro alterado à stage area do branch atual.
        $ git add <nome-do-ficheiro>

# Adicionar todos os ficheiros alterados, novos, e removidos
# à stage area do branch atual.
        $ git add -A

# Adicionar todos os ficheiros alterados ou removidos, não adiciona novos.
        $ git add -u

# Adicionar todos os ficheiros alterados ou novos, não os removidos
        $ git add .

# Adicionar todos os ficheiros da presente diretoria à stage area.
        $ git add *

############################ Gravar o estado do projeto ############################

# Submeter os ficheiros na stage area para a diretoria git.
        $ git commit -m "mensagem que descreve as alterações feitas"

######################### Enviar para o repo remoto ############################

# Enviar o estado de um branch especifico do nosso repo local, <branch>,
# para o repo remoto, origin.
```

```
# A opção -u cria uma ligação de up stream tracking entre o branch do repo local
# e o repo remoto.
# Exemplo: git push -u origin master
        $ git push -u origin <branch>

##################### Descarregar alterações feitas por outros ##################

# Descarrega o conteúdo do branch remoto do qual o branch local
# faz up stream tracking, opção -u do push apresentado anteriormente.
# Junta o conteúdo, merge.
        $ git pull

# Este comando pode ser substituído por estes dois, evita a necessidade
# da existência de up stream tracking entre os branches local e remoto.
        $ git fetch origin <branch>
        $ git merge origin/<branch>

########################### Navegar em branches locais ##########################

# Criar novo branch no repo local.
        $ git branch <nome-do-branch>

# Mudar para outro branch.
        $ git checkout <nome-do-branch>

# Cria e mudar para branch <nome-do-branch> com base em <branch-base>.
        $ git checkout -b <nome-do-branch> <branch-base>

# Ver todos os branches, locais e remotos, e suas ligações de rastreio.
        $ git branch -vva

# Juntar dois branches.
        $ git checkout <nome-do-branch-que-desejamos-manter>
        $ git merge <nome-do-branch-que-queremos-juntar>

# Eliminar branch.
        $ git branch -d <nome-do-branch>

# Eliminar branch com alterações não submetidas.
        $ git branch -D <nome-do-branch>

############################ Definir Tracking (Rastreio) ########################

# Para definir up stream tracking sobre um branch remoto <nome-branch>.
        $ git branch -u origin/<nome-branch>

# Para definir tracking entre dois branches locais, na criação do branch.
        $ git checkout --track -b <nome-do-branch> <branch-base>

################################# Ver estado ####################################

# Ver o estado de um branch relativamente ao branch definido
# como branch de up stream tracking ou tracking.
```

```
        $ git status

# Quando queremos ver o estado do branch local relativamente a um branch remoto,
# devemos primeiro atualizar o estado do repo remoto.
        $ git remote update

# Ver o histórico de commits do branch local atual.
        $ git log

######################### Reverter para um estado anterior ######################

# Obter o id do commit que representa o estado para o qual desejamos voltar.
        $ git log

# Fazer reset com, no mínimo, os 7 primeiros carateres do id do commit.
# Exemplo: git reset --soft c14809fa
        $ git reset --soft <id-do-commit>
        $ git commit -m "mensagem que descreva este reset"

############################### Remover ficheiro ###################################

# Remover ficheiro <nome-do-ficheiro>.
        $ git rm <nome-do-ficheiro>
        $ git commit -m "eliminei o ficheiro <nome-do-ficheiro>"

############################## Remover repo remoto #################################

# Remover branch no repositório remoto (GitHub)
        $ git push origin --delete <nome-do-branch>

########################### Ignorar ficheiros ou pastas ##########################

# Indicar ao git os ficheiros e pastas que devem ser ignorados
# Editar o ficheiro .gitignore, adicionar o caminho para tais pastas
# ou ficheiros. Caminho esse relativamente ao nosso repositório local.
# Exemplo:
        # caminho absoluto do ficheiro
            /home/user/MEGA/java_projects/Projeto_LTI/Files/GIT_tutorial.pdf
        # caminho do repositório local
            /home/user/MEGA/java_projects/Projeto_LTI/
        # caminho relativo
            /Files/GIT_tutorial.pdf

############################### Lista de comandos úteis ##########################

$ git config --global user.name "nome_utilizador"

$ git config --global user.email "email_do_utilizador"

$ git config --global credential.helper cache

$ git config --global credential.helper 'cache --timeout <tempo-em-segundos>'
```

```
# Cria atalho para um comando.
$ git config --global alias.<alias-name> <git-command>

# Muda editor de texto definido pelo git
$ git config --system core.editor <editor>

$ git init

$ git remote add <nome> <endereço>

$ git clone <endereço>

$ git add <nomes-dos-ficheiros>

$ git add -A

$ git add -u

$ git add .

$ git commit -m "mensagem que descreve as alterações feitas"

$ git push -u origin <branch>

$ git push origin --delete <nome-do-branch>

$ git pull

$ git fetch origin <branch>

$ git branch <nome-do-branch>

$ git branch -vva

$ git branch -u origin/<nome-branch>

$ git checkout <nome-do-branch>

$ git checkout -b <nome-do-branch> <branch-base>

$ git checkout --track -b <nome-do-branch> <branch-base>

$ git merge <nome-do-branch-que-queremos-juntar>

$ git branch -d <nome-do-branch>

$ git branch -D <nome-do-branch>

$ git remote update

$ git status

$ git log
```

```
$ git reset --soft <id-do-commit>

# Reset a eliminar todos os commit feitos depois
# do ponto para o qual queremos voltar. (perigoso)
$ git reset --hard <id-do-commit>

# Quando queremos trocar de branch mas só temos meio trabalho feito e portanto
# não queremos fazer um commit para guardar as alterações antes de
# trocar de branch.
$ git stash

$ git rm <nome-do-ficheiro>
```