COMMON COMMIT

> git commit -m "Message..."

DETAILED COMMIT

> git commit → opens the default editor (VS Code here) on the file COMMIT_EDITMSG

Edit COMMIT_EDITMSG

↓

First line gets the short description.

In

[redacted] line gets the long description.

↓

Save

↓

Close editor

↓

Back in terminal

↓

Basic statistics are shown ← commited!

TO KNOW WHAT IS IN THE STAGING AREA

> git status → general view

> git ls-files → shows only the staging area.

TO REMOVE A FILE FROM BOTH WORKING DIRECTORY AND THE STAGING AREA

> git rm file.txt

MOVING AND RENAMING FILES

> mv file1.txt main.js → generates two changes for git

> git status → 1 deleted (file1.txt)
1 new (main.js) (untracked.

> git add file1.txt → stages the deletion.

> git add main.js → stages the "new" file.

> git status → will indicate "renamed"!

ALL THESE IN ONE STEP

> git mv main.js file1.txt → renames and stages.

> git status → renamed!

> git commit -m "Names changed."

.GITIGNORE

> mKdir logs
> echo hello > logs/dev.log
> git status → shows new untracked /logs/ directory.

Let's ignore it.

> echo logs/ > .gitignore

> code .gitignore → opens it for edition.
-or-

> echo main.log >> .gitignore

> echo *.log >> .gitignore

> git status → shows only the new file .gitignore, because now it is ignoring the /logs/ directory.

> git add .gitignore

> git commit -m "Add .gitignore"

IGNORING FILES THAT WERE ARLREADY COMMITED.

> mkdir bin → Supposedly the directory with my compiled code.

> echo hello > bin/app.bin → just to have a file there

> git status → shows we have this new untracked directory.

> git add . → adding all the changes to the staging area.

> git commit -m "accidental add bin".

= Now we have a problem =

> code .gitignore =or=
> echo bin/ >> .gitignore
> git status → shows we have modified the .gitignore
> git add .
> git commit -m "Include bin/ in .gitignore."

1 file changed
2 insertions
Git is not ignoring the bin/ directory because it is already tracking it.

= Let's modify our bin file =

> echo BAD? > bin/app.bin

> git status → git says that the app.bin file is modified this is not what we want.

— So solve this problem We have to remove this file from the staging area =

> git ls-files → shows files in the staging area and bin/app.bin is there.

> git rm -h → shows help for git rm

> git rm --cached bin/ → removes bin/ only from the staging area but not recursively. Will fail.

> git rm --cached -r bin/

↳ Removes /bin from staging area only and recursively. So that one will work.

> git ls-files → shows the staging area without the /bin/app.bin file.

> git status → one change ready to be commited:
deleted: bin/app.bin

> git commit -m "Remove the bin directory that was accidentally commited."

= From now on git is not going to track this directory anymore =

> echo test > bin/app.bin
> git status → nothing on staging area, nothing to commit, working tree clean.

GIT IGNORE TEMPLATES

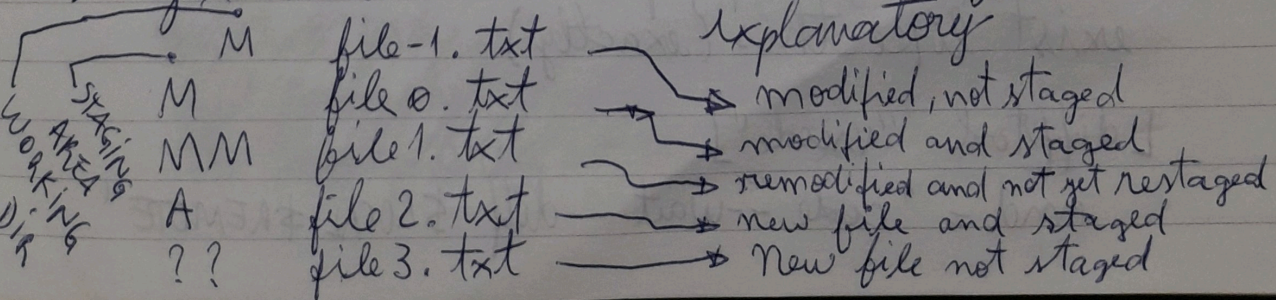
https://github.com/github/gitignore

SHORT STATUS OUTPUT

= More comprehensive =

> git status → verbose

> git status -s → short but yet more explanatory



= COMPARE REPO TO STAGING AREA =

> git diff --staged

= COMPARE W. DIR TO STAGING AREA =

> git diff

= VISUAL DIFF TOOLS =

- KDIFF3
- P4MERGE
- WINMERGE (Windows only)
- VSCODE

= TO SET VSCODE AS THE DEFAULT VISUAL DIFF TOOL =

> git config --global diff.tool vscode

= TELLING GIT HOW TO LAUNCH VSCODE =

> git config --global difftool.vscode.cmd ^
"code --wait --diff $LOCAL $REMOTE"

In ~~the~~ .gitconfig file 2 lines must exist like this (exactly):

```
[difftool "vscode"]
    cmd = "code --wait --diff $LOCAL $REMOTE"
```

Remember that the command to open the .gitconfig file in VScode is:

> git config --global -e

— TO LAUNCH THE VISUAL DIFFTOOL =

> git difftool

> git difftool --staged

== VIEWING THE HISTORY ==

> git log → long descriptive history.
"space" for next page
"q" to quit

> git log --oneline → a short summary of the commits.

> git log --oneline --reverse → the same but in reverse order.

== TO UNDO THE STAGING OF A FILE OR DIRECTORY ==

> git restore --staged { file1.txt / *.py } ← undo the add command

~~This takes the file1.txt from the last snapshot committed and puts it on the staging area~~

— LINK LOCAL TO REMOTE REPO —

> git remote add origin LINK

— TO GET ALL FILES
FROM main BRANCH
IN THE REMOTE REPOSITORY
TO THE LOCAL REPOSITORY —

> git pull origin main
master
develop

— TO SEND THE LAST
COMMIT TO THE REMOTE
REPOSITORY (TO THE main BRANCH).
FROM THE LOCAL REPOSITORY —

> git push -u origin main
master
develop

— CRIAR UMA NOVA BRANCH —

> git branch novabranch

— COMEÇAR A TRABALHAR NA
NOVA BRANCH —

> git checkout novabranch

= ADD THE NEW BRANCH TO THE REMOTE REPOSITORY =

```
> # make changes, do things #
> git add <files>
> git commit -m "message"
> git push -u origin novabranch
```