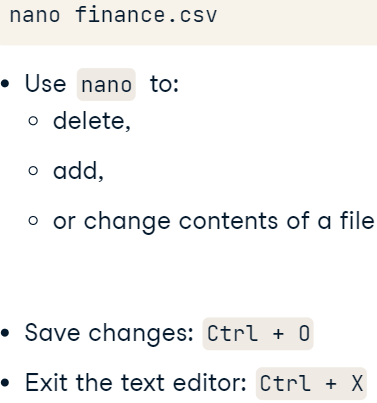
# GIT

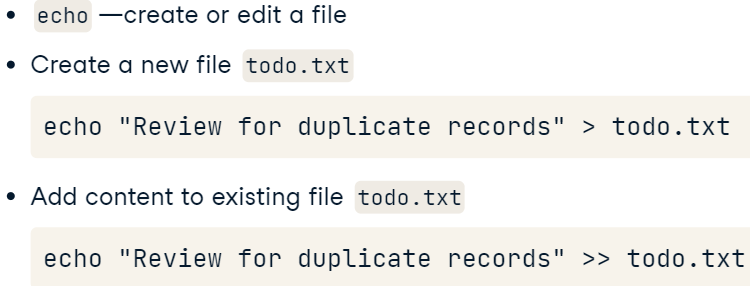
## Introduction to Git

### Introduction to Git

#### Introduction to version control with Git

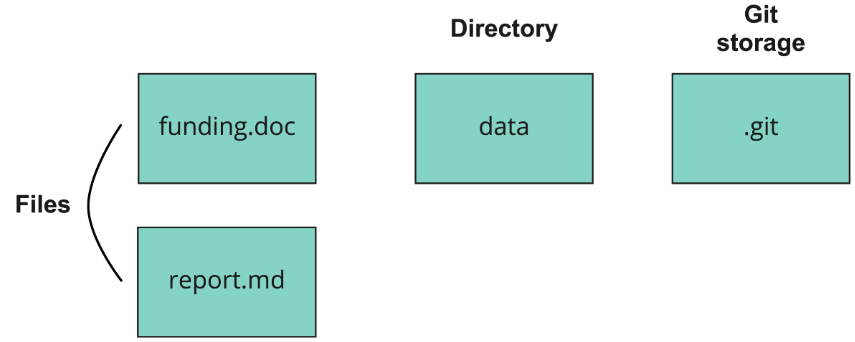
* **Pwd** – directory;
* **Ls** – files in directory;
* **Cd “directory”** – moving deeper in directory;
* **Editing a file**:



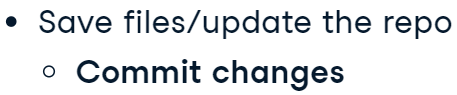
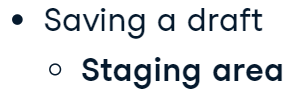


#### Saving files

* A repository:

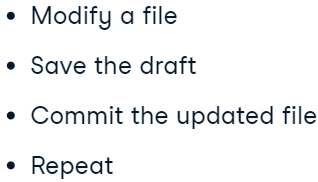


* Staging and making a commit:



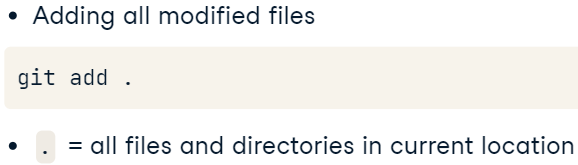


* **Ls –a** – hidden files in directory;
* **Nano file** – open a text editor;
* **git status** – check the status of files.
* Git workflow:

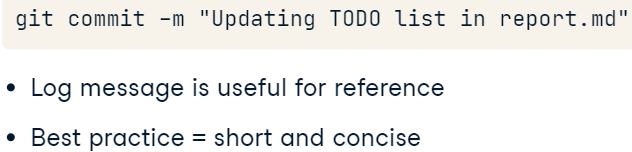


* Saving files:



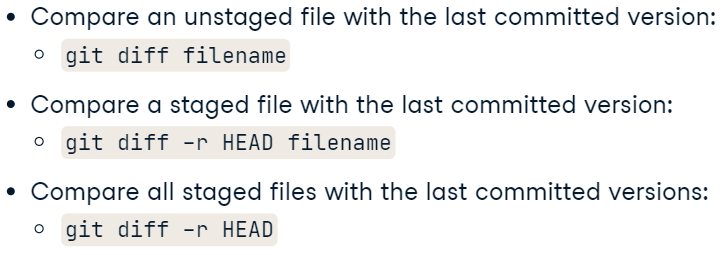


* Make a commit:



#### Comparing files

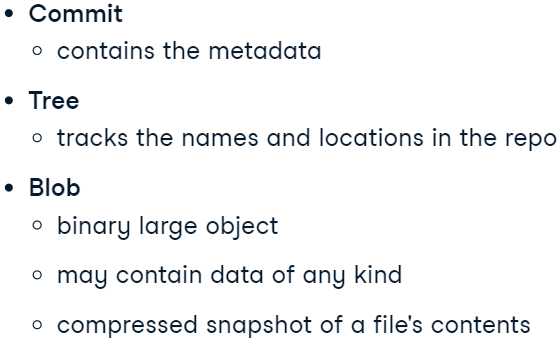
* Compare files:



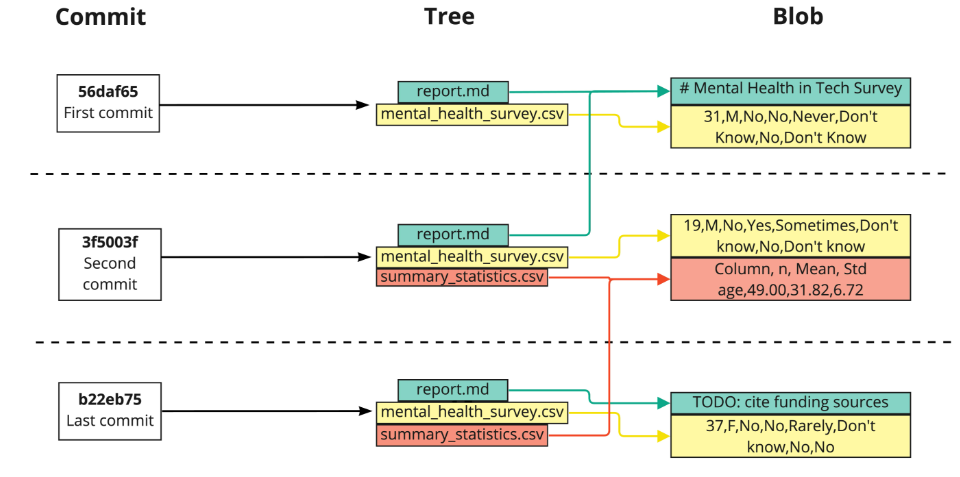
### Making changes

#### Storing data with Git

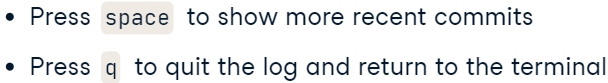
* The commit structure:



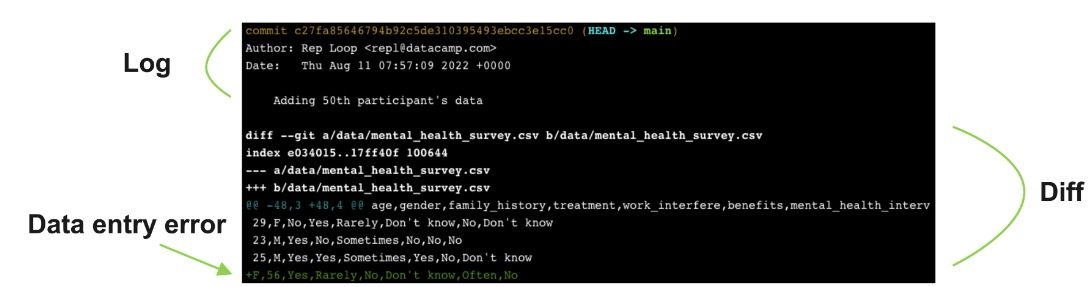
* Commit structure:



* **git log** – commit information using the git log command, which will display all commits made to the repo in chronological order.

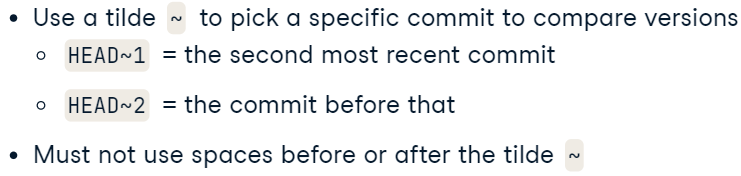


* **git show hash** – finds changes in a particular commit.



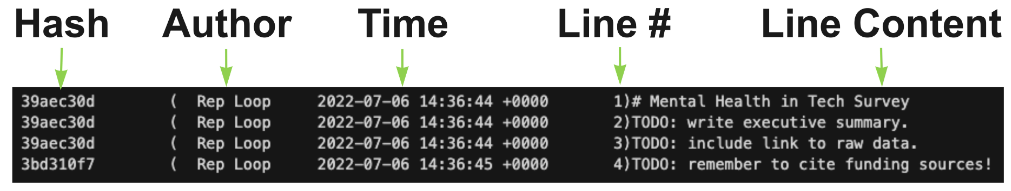
#### Viewing changes

* **HEAD~number**:



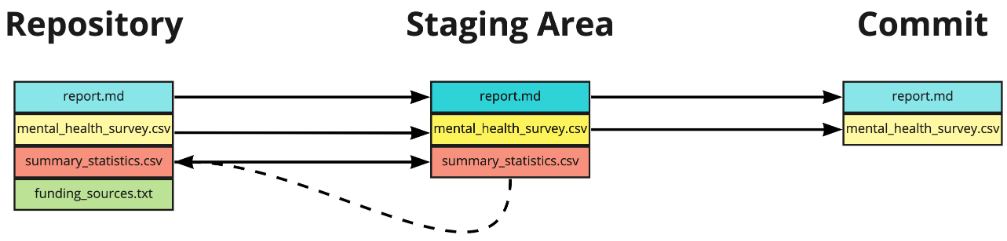
* **git show HEAD~number** – show the changes made in that specific commit.
* **git diff hash (HEAD~numb) hash (HEAD~numb)** – show the difference between commits.
* **git annotate filename** – who and when made changes to the file.

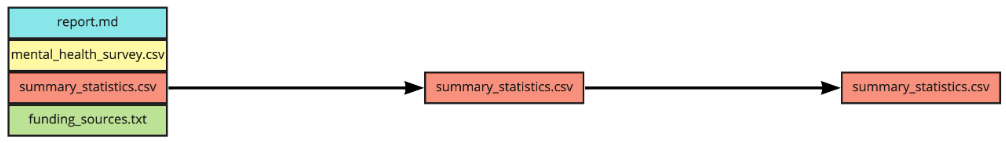
Result:



#### Undoing changes before committing

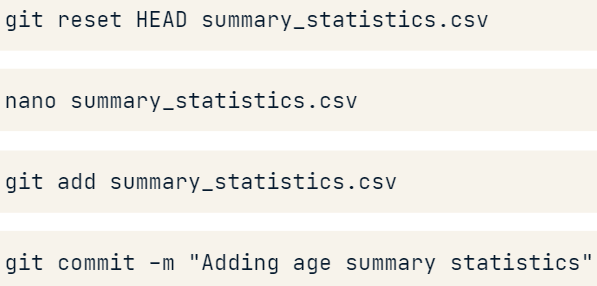
* Unstaging process:



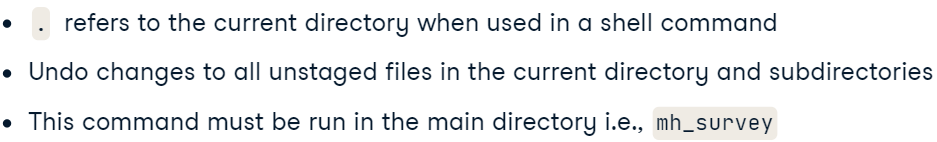


* **git reset HEAD filename** – unstaging file.

Process: unstage, edit file, add to the stage, commit it



* + **git reset HEAD** – unstage all files.
* **git checkout –– filename** – undo changes to a file in a rep (reverse file to the stage of last commit).
  + **git checkout .** – undo change to all files in a rep.
  + process:



* Unstage, undo, stage, commit:



#### Restoring and reverting

* **git log –number** – shows us “number“ of most recent commits.
  + **git log –number filename** – shows us “number“ of most recent commits a of particular file.
* **git log –since=’month dd yyyy’** – restrict log by date.
  + commits between days:

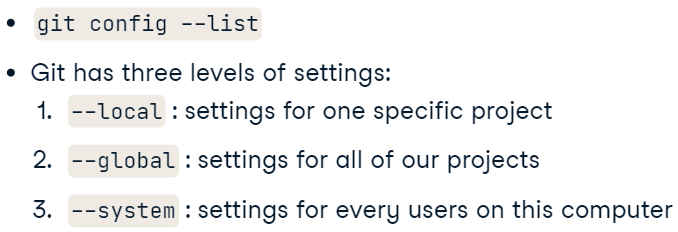


* **git checkout hash (HEAD~number) filename** – reverts a file to a specific commit.
* **git clean -n** – a list of untracked files.
  + **git clean -f** – deletes all untracked files.

### Git workflows

#### Configuring Git

* Level of settings:



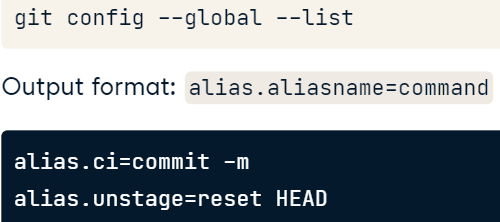
* Changing settings:



* Using an alias:



* Tracking aliases:



* Ignoring specific files:

Data analysis often produces temporary or intermediate files that you don't want to save.

You can tell it to stop paying attention to files you don't care about by creating a file in the root directory of your repository called .gitignore and storing a list of wildcard patterns that specify the files you don't want Git to pay attention to.

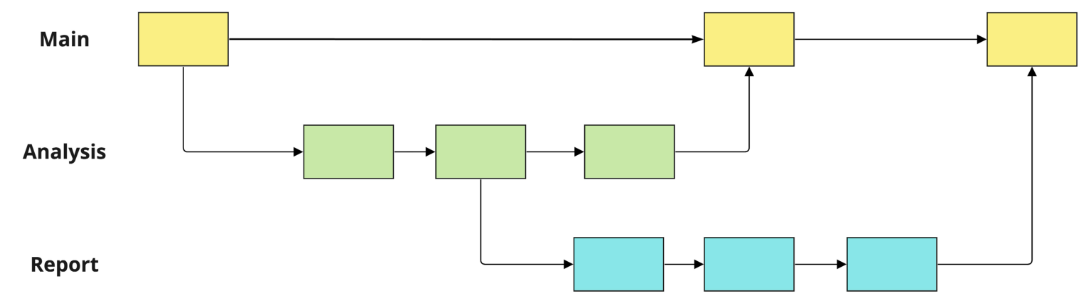




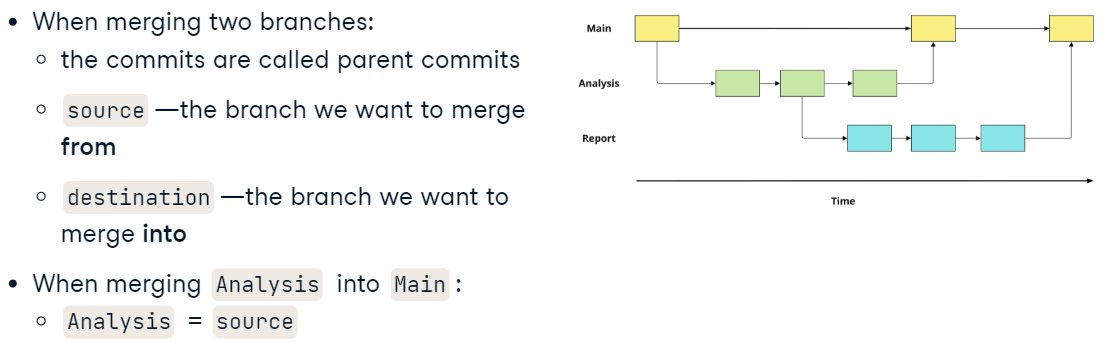


#### Branches

* Merge report into main:



* Source and destination:



* Identify branches:



* Create a new branch:

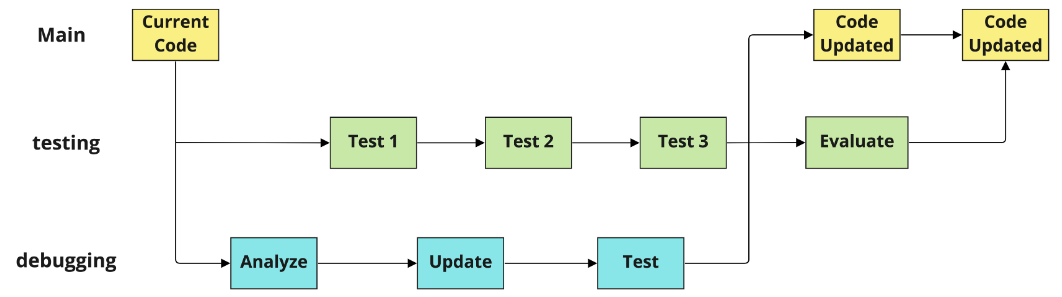


* Difference between branches:



#### Working with branches

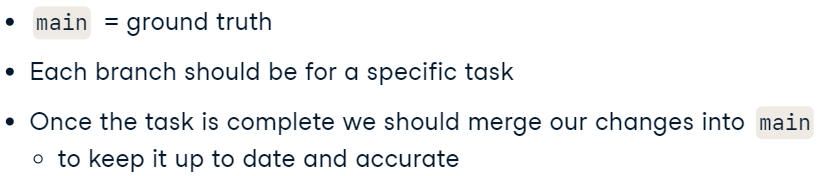
* Example of use switching branches:



* switch branch:



* Working with branches:



* Merging branches:

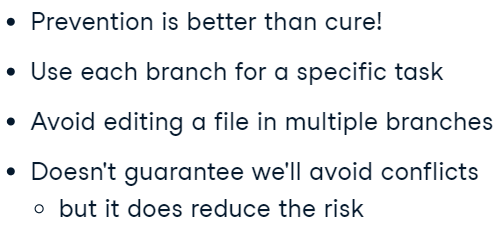


* Update branch content:



#### Handling conflict

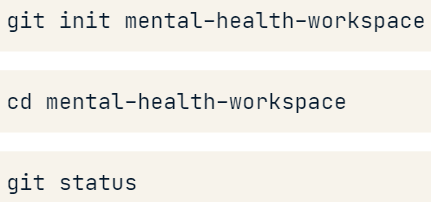
* **Conflict Indicators**: You learned to recognize Git's conflict syntax, which includes <<<<<<< HEAD to mark the beginning of conflicting changes in the current branch, ======= as the divider, and >>>>>>> [other branch name] to indicate the end of the conflicting section from the other branch.



### Collaborating with Git

#### Creating repos

* Creating a repo and check it’s status:



* Coverting a project:



#### Working with remotes

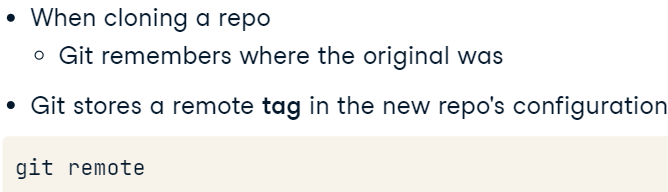
* Clone project in a repo (with the name at the end):



* Clone a remote on local version:

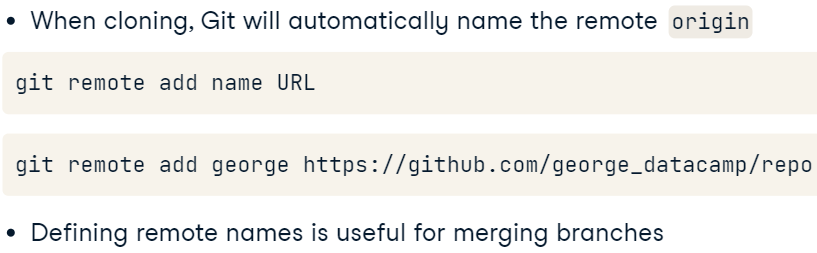


* Identifing a remote:



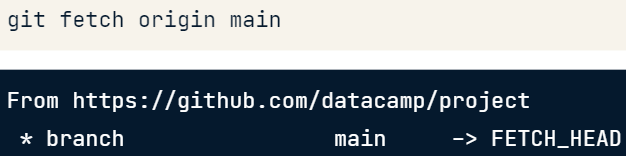
 - with URL’s.

* Create a remote:



#### Gathering from a remote

* Comparing remote against the contents of a local repo (from remote to local):
  + Fetch from the remote:



* + Fetch it into a branch:



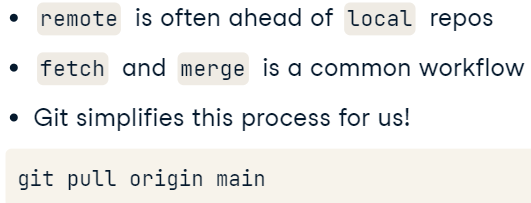
* + Synchronize the contents (show changes):



* + Comapre contents:

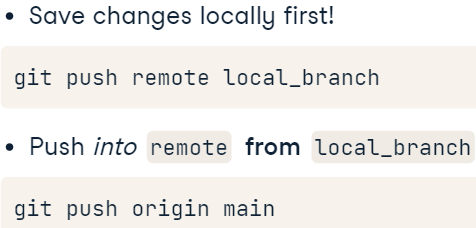
git diff rep\_1 rep\_1

* Pull from the remote (fetch and merge in a one command):



#### Pushing to a remote

* Files from local to remote:



* Resolve a conflict of not rightly working with remote and local repos:

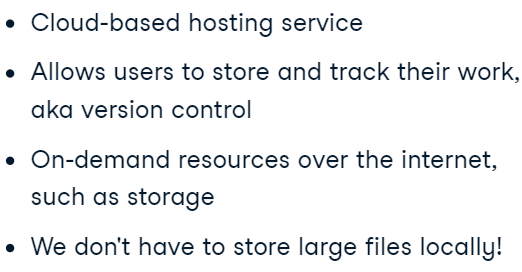


## GitHub Concepts

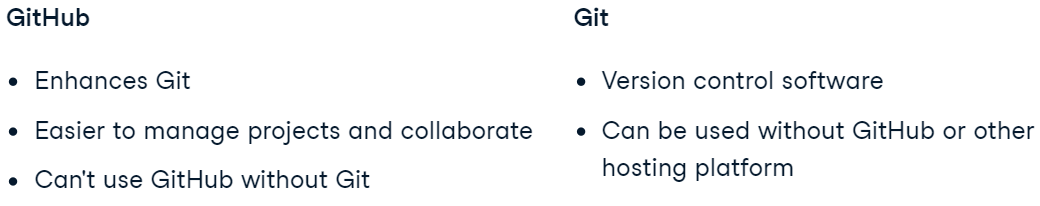
### Introduction to GitHub

#### What is GitHub?

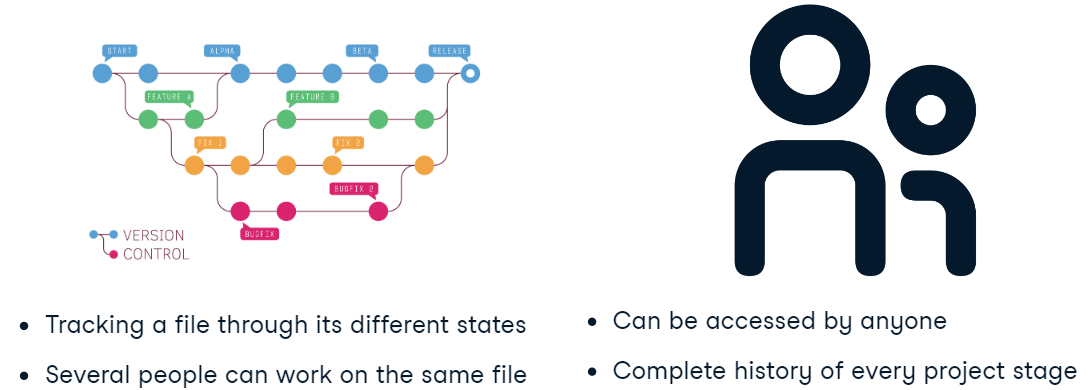
* What is it:



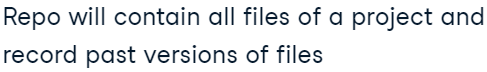
* GitHub vs Git:



* Collab on GitHub:

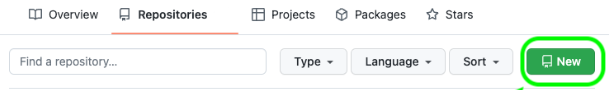


* What is a repo:



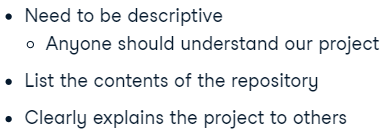
#### Setting up a repo

* Creater a repo:

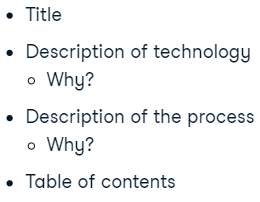


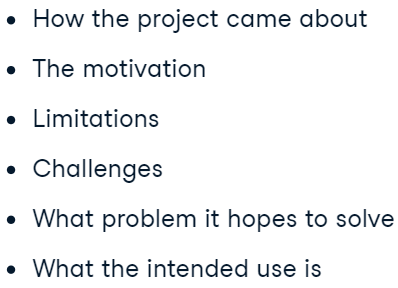
#### Creating a README

* What README should do:

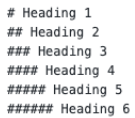


* + What does it contain:





* Headings:



* Text formatting:



* Hyperlinks:



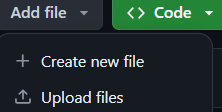
* Images:



### Working with Repos

#### Modifying a repo structure

* Create a file or upload one:



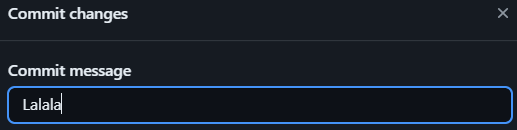
* + naming it:



* Saving file (save on main branch or on a new one):



* Committing massage:



* Creating file path:

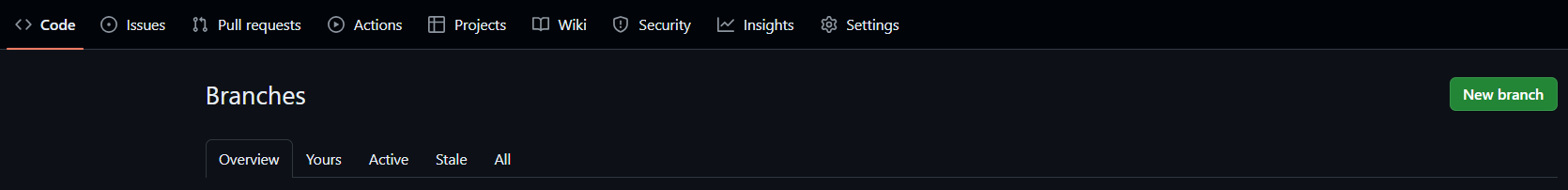


* Process of creating a directory:

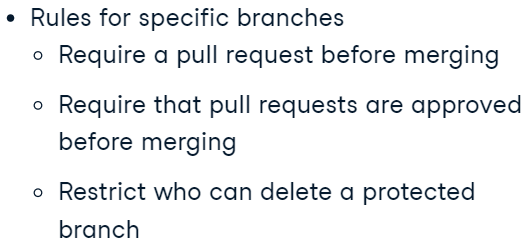
Adding a new directory in GitHub requires the creation of a file inside the new directory, so it's common to create a README.md file during this process.

#### Working with branches

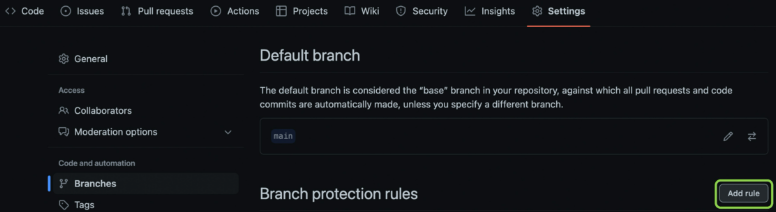
* Create a branch:



* Branch protection rules:
  + why we need it:



* + where can we change it:

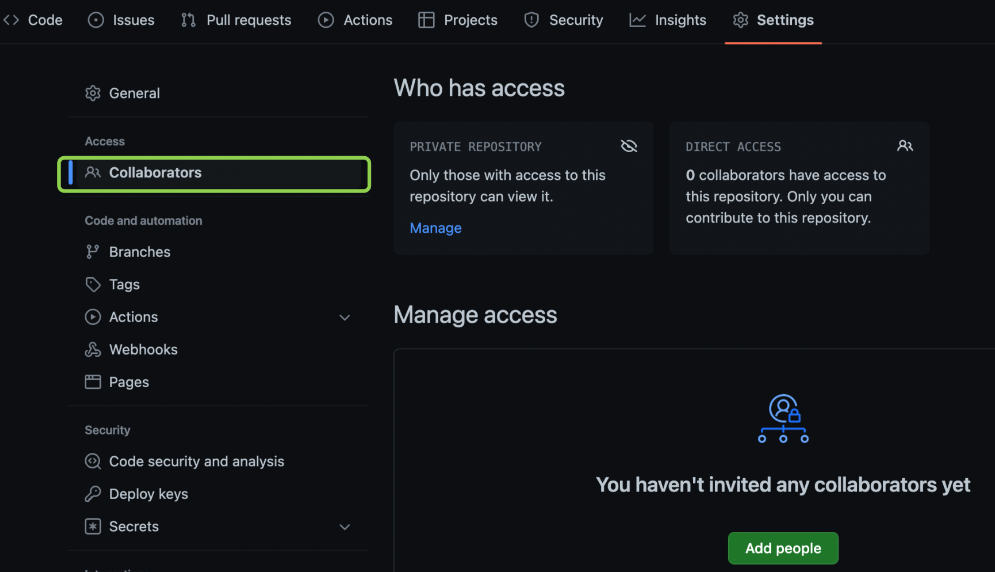


#### Repo access

* Creating private repo:

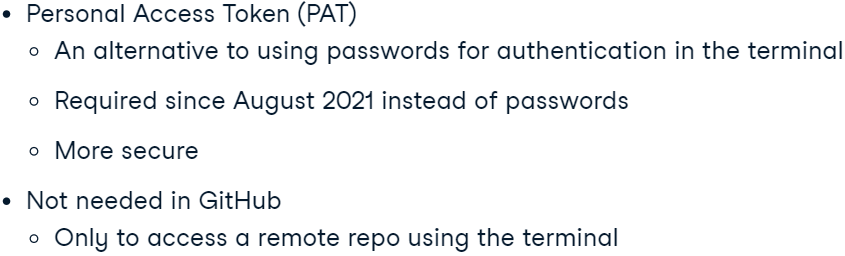


* Adding people to project:



#### Personal Access Tokens (PAT)

* What is it:



Generally, working directly on GitHub doesn't require a PAT, while performing some tasks to interact with remote repos via the terminal will require you to set up a PAT.

* Create a PAT:

profile -> settings -> developer settings -> PAT -> generate token.

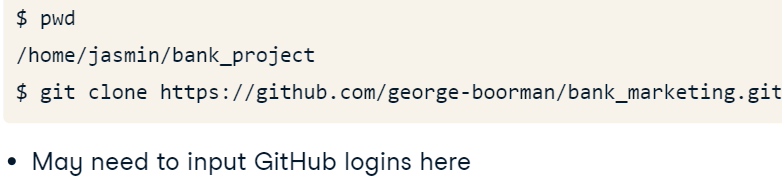
* From git to local version:



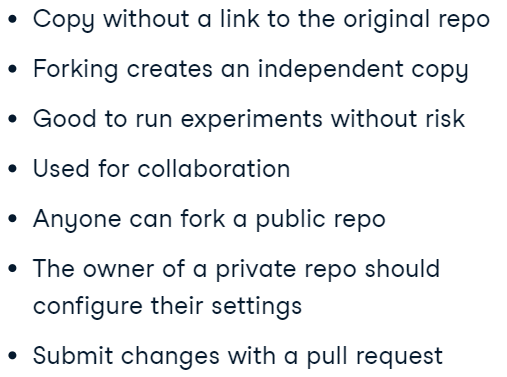
### Collaboration with GitHub

#### Using other repos

* Cloning a repo:



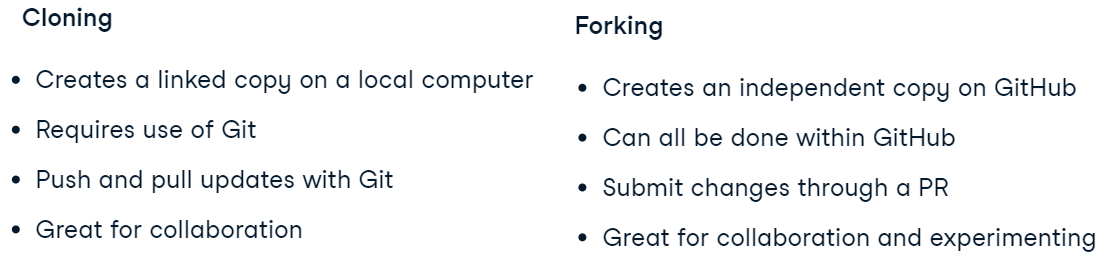
* Forking:



* + where to access:

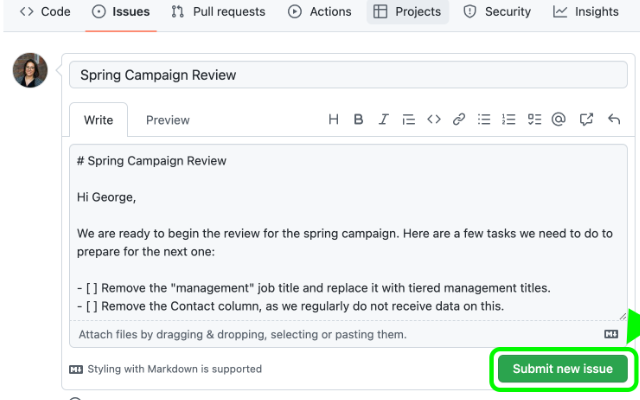


* Fork vs clone:

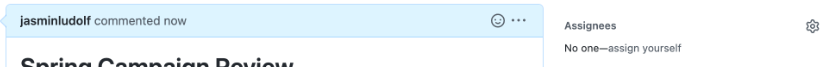


#### GitHub issues

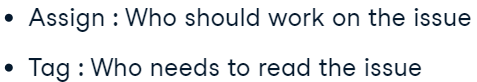
* Create an issue:



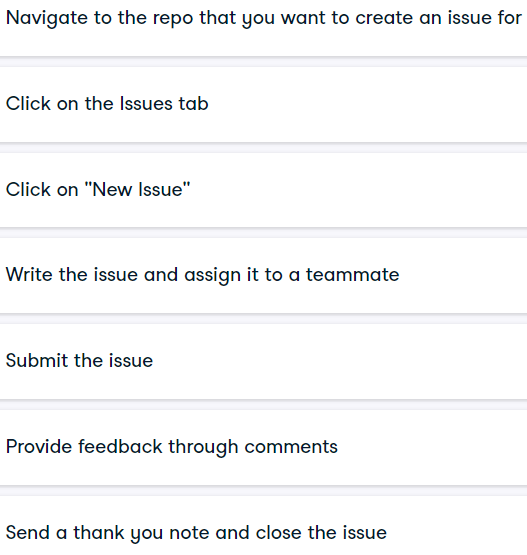
* Which person has to see it:



* Tagging vs assigning:

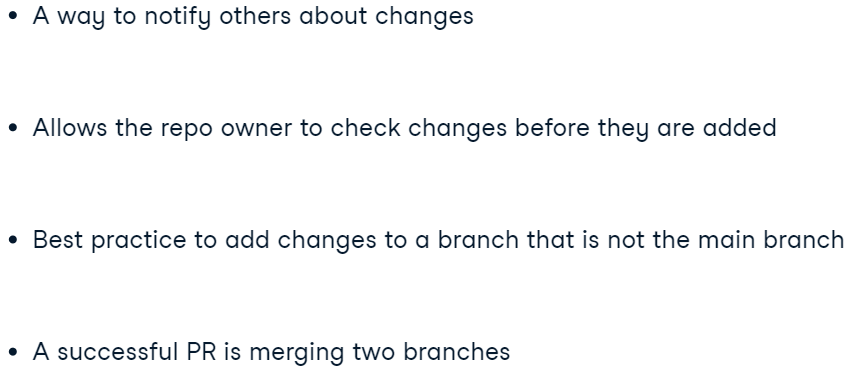


* The process:

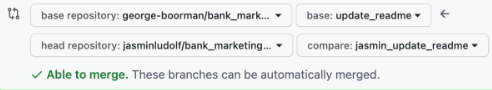


#### Pull requests

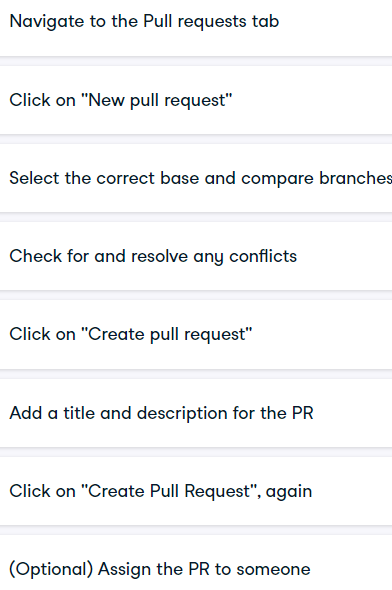
* Why we need PR:



* Pull Request -> create pr -> : -> add a comment

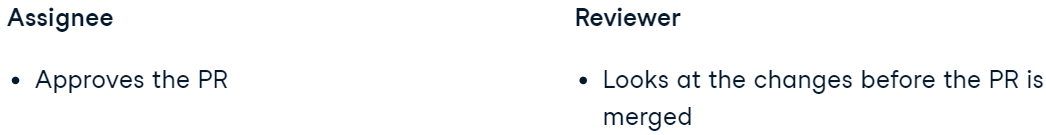


* Process:

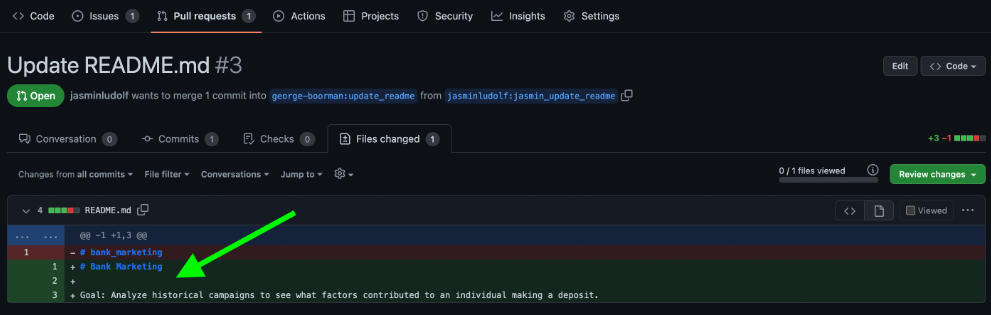


#### Reviewing pull requests

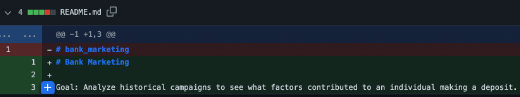
* Assignee vs reviewer:



* To see changes as a reviewer (then click “review changes”):



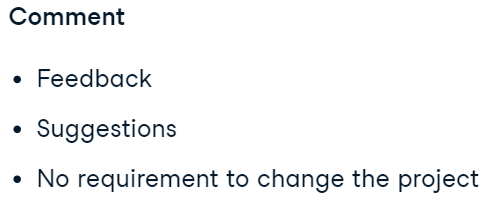
* To clirify detail use “+“:



* Options to request:



* + Comment vs feedback:





* + Approve – when all is good.
* When the changes are implemented (re-request review):

