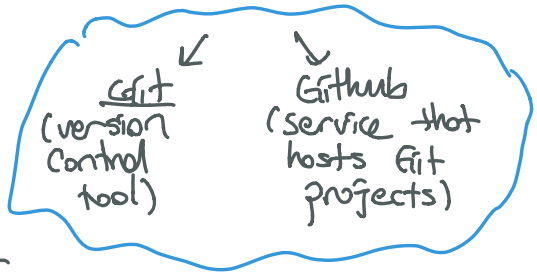
What is version control → control different versions of your project.

- Git
- Subversion
- Mercurial

- Centralized (a powerful center comp.)
- Distributed (each developer has complete copy of the project on their computer.) → you can work offline.



- Version control in our lives ⇒ "undo" (ctrl-z)
- What else ⇒ the ability to label a change.
  " " " " give a detailed explanation why the change was made.
  " " " change A, make edit B, then get back change A, without effecting edit B.

Terms

- Source Code Manager = version control system
- Commit = save the state your project in Git
- Repo = directory which contains your project work
- Working Directory = the files that you see in your computer's file system.
- Checkout = repo → working directory.
- Staging Area / staging Index / Index = prep table where Git will take the next commit.
- SHA = id number for each commit
- Branch = a line of development that diverges from the main line of development.

Git Bash Here

After downloading → right click on empty space and press "git bash here"

cd : Moves shell to name directory.
start . : Open this directory.
mv : rename the file

git config --global user.name "..." → sets up Git with your name
" " " " .email
" " " " color.ui
" " " " merge.conflictstyle diff3
git config --list

git config --global core.editor "atom --wait"

Create A Repo from Scratch

"git init" → to create repo.
- ls: to list files
- mkdir: create a new directory
- cd : change directories
- rm: remove files and directories
- pwd: print working directory

① To create directory and move into that directory:
mkdir -p ___/___ && cd $_

② To create repo: git init
After that in the command line there is "master"

Cloning

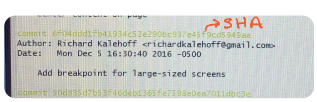
"git clone" (before that: make sure you are located in correct directory. Don't clone in git repo.

git clone _____ project-name you can change its name by this way.

Git status

"git status" → ① on branch master → which branch?
② your branch is up-to-date with 'origin/master' → copied from another computer and it is sync with the original one.
③ nothing to commit → there are no pending changes.

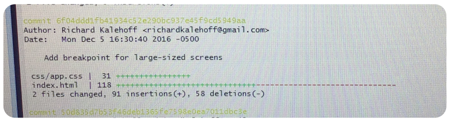
Git Log → log of commits



- SHA: id number → we may not want the see all content
- Author
- Date
- Message ✓

Alter how log displays → git log --oneline (no author/no date/first part of SHA

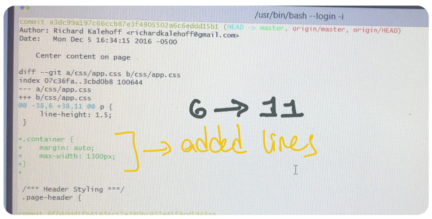
git log --stat : display the files that have been changed.



short-cut for log.

↑ ↓ : one line at a time
u, d : move half the page
b, f : move whole the page

We can see actual content change by git log -p.



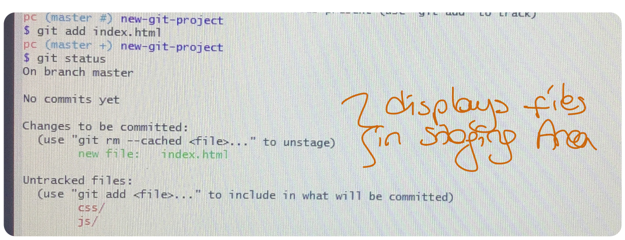
To see specific SHA git log -p fd... (but it will also show prior ones

git show --- → exact one

Git Add

When we have some new files that we want Git to start tracking
First we need to put them on repository. In order to put on repo:
it first needed to be Staging index.

Git add: move files from Working Directory to Staging Index.



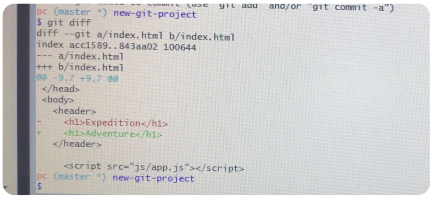
- git add . → stage all files!

Git Commit

Staging Index → Repo

Opens text editor →

Git diff : To see changes that have been made but haven't been committed yet.



Looks very same ase git log -p

.gitignore ⇒ Thanks to this file we ignore specific files which we dont want to see on git status (we untrack those files)

touch .gitignore

Git Tag

see the tag.

git tag → v1.0

↳ Let's say that this is special commit and we want to give a tag name like v.1.0 or beta.

git tag -a v1.0

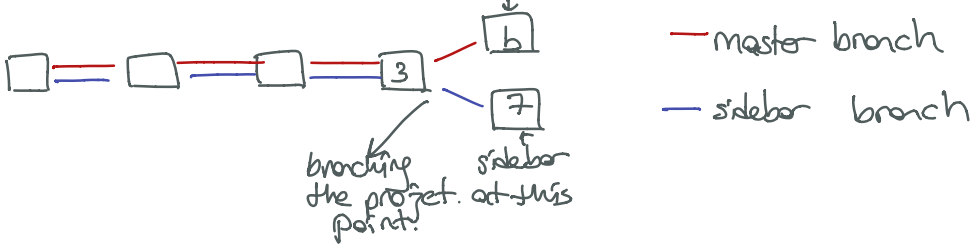
↳ this one gives tag name the last commit.

git tag -a v1.0 a88082 → it gives tag name to past commit

git tag -d v1.0 a88082 : this deletes tag name

Git Branch

head → can be changed by checkout



git branch → list all branch names in the directory

git branch sidebar → create sidebar branch at the last commit

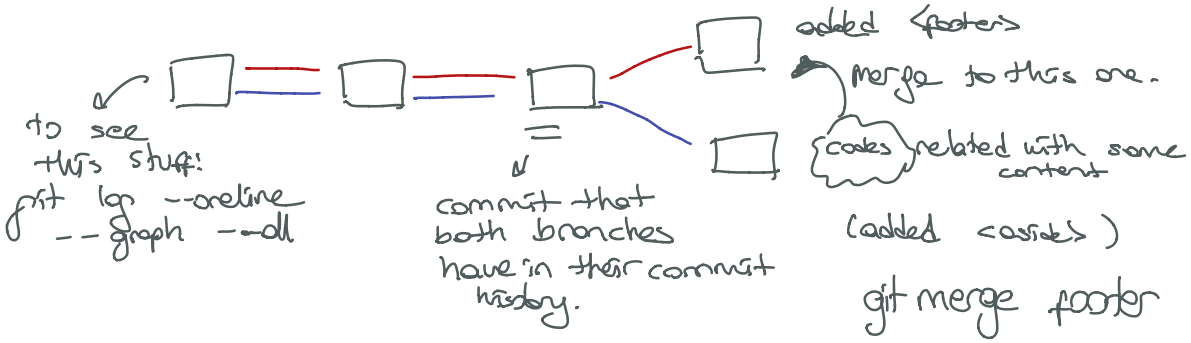
git checkout sidebar → switch (moves head to) that branch (remove all files and copies from active branch)

git branch sidebar 3 ↳ branches from this SHA

git branch -d sidebar → delete branch (-D for forcing)

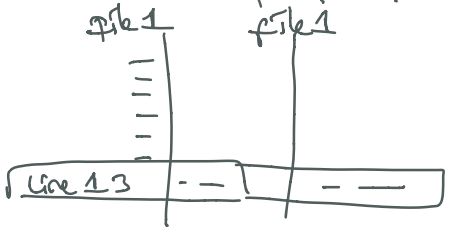
Merging = Combining branches

git merge <name-of-branch-to-merge-in>



git merge footer

When does it fail? ⇒ merge conflict.



→ There are two different things at the exact same location. So what should be header line?

Run git status

Undoing Changes

① git commit --amend : you can alter the most recent commit (code editor will be opened)

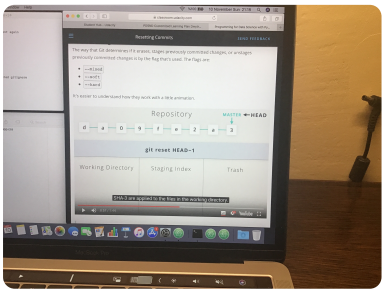
Moreover, if you forget something in your code:

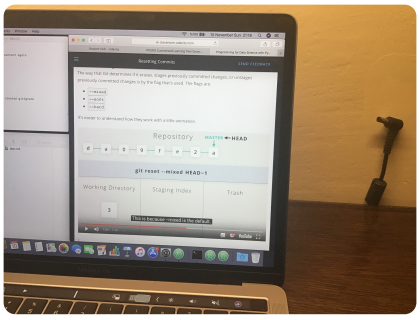
- edit the code
- save the code
- stage it (git add)
- git commit --amend

② git revert: does the exact opposite of that commit.
(add → delete)
(delete → add)

git revert <SHA-of-commit-to-revert>

③ git resets: erases the commits





① → go 1 before of header

reference point

git reset --mixed HEAD~1 (3 → WD)

git reset --soft HEAD~1 (3 → SI)

git reset --hard HEAD~1 (3 → trash)

before you delete anything → use git branch backup

Working with Remotes

① Add a remote Repository:

- You can create remote repositories by Github
- git remote add origin <url> (connection from my local repository to the remote repo)

② git push send all of the commits from local to remote.

git push <remote-shortname> <branch>

origin master

After you push,

"origin/master" → tracking branch. = remote + branch

remote origin has the master branch.

③ git pull <remote_shortname> <branch>:

After pull, then does the merging.

④ fetch: If you don't want to automatically merge the local branch with the tracking branch. There are commits on the repo that you don't have but there are also commits on the local repo that the remote one does not have either

```
git fetch origin master
git merge origin/master
```

Working On Another Developer's Repository

① ~~git fork~~ split a repository into an identical copy.

Fork

↓

you can do it on github not on git.

It is different than cloning ⇒ clone = local machine
fork = remote

- My account does not have permission to edit her repo directly, so I'll fork the repo to my own account.
- After forking, you can clone it on your own computer.

```
git clone <url>
```

Reviewing Existing Work

① Group by Commit Author : git shortlog

= just numbers / order by number: git shortlog -s -n

② Filter by Author

```
git log --author=Sumra
git log --author='Paul Lewis'
```

③ filter commits by search

```
git log --grep=bug
git log --grep='unit tests'
```