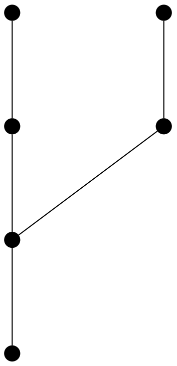
Git & GitHub
Hands-On

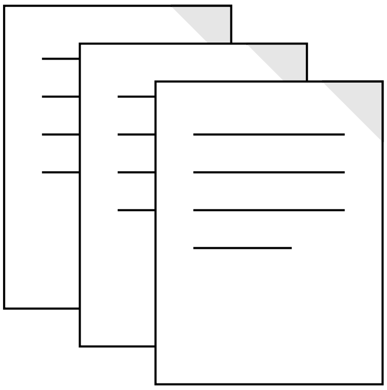
An opinionated guide

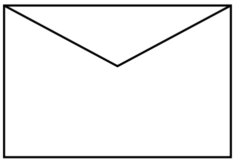
Nicolás Aguado
nico@nico.eus

November 3, 2024



1. Motivation





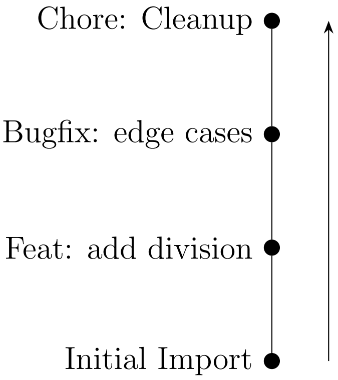
Git [1] [2]
Global **I**nformation **T**racker
Goddamn **I**diotic **T**ruckload of shit

Source code version control - Based on *commits* - Scope: Only text files

[1] `git-scm.com`

[2] Explorer Association, Notepad as Default Editor

2. Working Tree (Checkout, Commit)
   `nicoGHO/workingtree` [3] [4] [5] [6] [7]



[3] `git log`
[4] `tail -f <file>`
[5] `git checkout <hash|master>`
[6] `git add <file|pattern>`
[7] `git commit -m <msg>`

3. Git Graphical Clients
   Environment Setup

   Setup Environment Variables
   Microsoft VSCode [8], GitKraken [9]

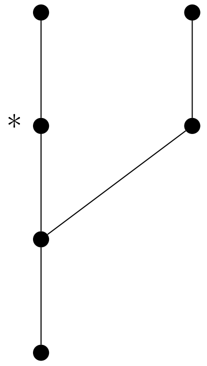
   `nicoGHO/workingtree`

[8] `code.visualstudio.com`

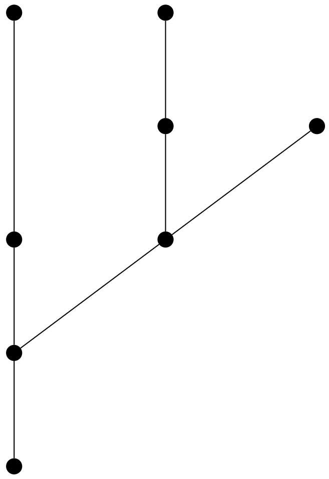
[9] `gitkraken.com`

4. Good practices with projects
   - Use descriptive commit messages (feat, bugfix, chore)
   - Use .gitignore (Minimal per folder, granularity)
   - Do relatively small commits (Max. 100-200 lines)
   - Dot (.) files are usually configuration files - do not touch them

5. Branches [10]
   Tags



Merge, Merge Conflicts
`nicoGHO/branches`



[10] Cuidado en VSCode con el modo *detached*!

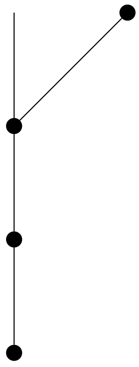
6. Good practices with branches
   - Protected **stable** master branch
   - One branch per feature / person / organization
   - Merges done with affected coder(s) present

Rebase, Cherry-Pick

7. Git in the Cloud (GitHub) [11] [12]
   Origin (Clone, Push, Pull) [13]
   Permissions for GUIs (Private Clone, Contribute)

   `nicoGHO/pushpull`

   Conflicts when pushing and pulling - People working at same time on **same branch**



[11] `git config --global user.name "John Doe"`

[12] `git config --global user.email johndoe@example.com`

[13] En Linux: `git config --global credential.credentialStore cache`

8. Visualization (Code, README)
   Network Graph
   Issues
   Projects

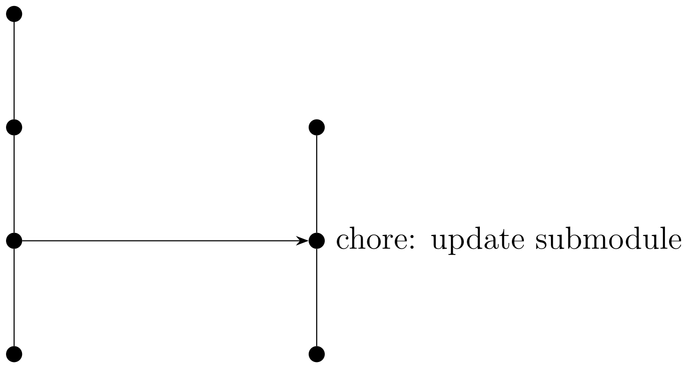
   `nextcloud/server`

9. Pull Requests
   Permission-regulated

   `nicoGHO/pullrequests`

10. Submodules
    Adding [14], Updating [15] (*Commit-based* `.gitmodules`)
    Cloning [16]
    Easier if inmutable

    `nicoGHO/submodules`



[14] `git submodule add <repository-url> <path>`

[15] `git submodule update --init --recursive`

[16] `git clone --recursive <project url>`

11. Summary

To start:

```
git clone --recursive <project url>
```

To edit:

- Create branch or checkout one
- Edit, commit
- ¿Push?
- Edit, commit
- Submodule has updated? Run commands
- Edit, commit
- Issue arises and not your responsability? Document it
- Edit, commit
- Pull request for merging into staging
- Ask your local wizard for merge
- Repeat

If something breaks:

Do not be afraid to use the terminal!

¡Use git and **learn by fighting against it**!

These are some extra topics left as an exercise to the reader

- Git LFS
- Multi-Remote Work
- Revert Pushed Commits
- Several Staging Branches (prod, pre, dev)
- Automatic Deployments (CI/CD)