Git for mathematicians

Pieter Belmans

December 4 2024

University of Luxembourg

Plan: become superheroes

1. understand *why* to use Git
2. ~~learn the entire commandline tool~~
3. getting started with Git
4. some fun more advanced things

Why Git?

What is Git?

we won't be talking about GIT, = Geometric Invariant Theory

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

why does a *mathematician* need "distributed version control"?

Tools for collaborative paper writing

maybe you haven't collaborated yet, but in case you have: *what did you use?*

- emailing files back and forth
- Overleaf
- Dropbox

any others?

have you used something beyond Overleaf for LaTeX editing?

What do you (or I) not like about them?

- lack of conflict resolution
- clutter of temporary files (with Dropbox)
- only with internet access (for Overleaf)
- paid accounts
- no real versioning: arXiv v1, submitted to journal, first revision, ...
  (unless you remember to save them at the time, or use Overleaf's basic history functionality)
- no diffs: when your coauthor edits things, there is no description of the changes, and no way to easily see what has changed

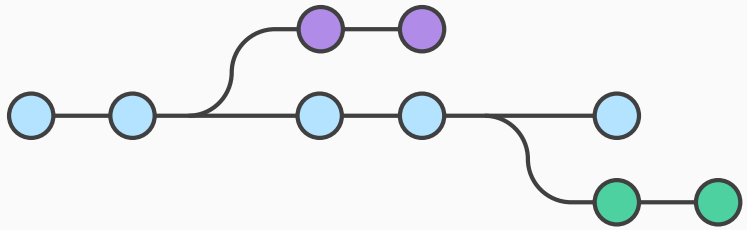
What is Git?

Think of it as a *much* better form of Dropbox (but it's not that at all to be honest):

1. never have `paper.tex`, `paperv2.tex`, `paperv3.tex`, `paperv3final.tex`, `paperv3submitted.tex`, ...(and the temporary files)
2. description of changes
3. good handling of conflicts: merge if possible, conflict markers if not

What is Git? (2)

4. diffs
5. decentralised (so without internet access)
6. ...(more advanced use cases later)



```
git diff
X$ is a regular
a squence of mo
a sequence of m
```

it has no interactive LaTeX editing like Overleaf though: you can use use your own preferred setup, or sync it up with Overleaf even

What is GitHub?

Git is the software

GitHub is a service (website) to make interactions with Git easier:

- central place for your projects
- many additional features

alternatives: use university-provided infrastructure (`gitlab.uni.lu`), GitLab, BitBucket, ..., or self-hosted

more benefits of GitHub:

- GitHub Codespaces: alternative to Overleaf (I don't use this, but you could!)
- GitHub Actions: partial alternative to Overleaf (I'm a big fan, see later)

Learning Git

How *not* to learn Git

- only use Git from the commandline, especially if you never used it before
- read `https://git-scm.com/book/en/v2` from front to cover, before starting a project
- first learn all about the history of CVS, Subversion, Mercurial, Git
- make sure to learn all about the “philosophy” and mathematics behind Git

instead:

- use a GUI (Graphical User Interface)
- just get started! mathematicians just need a few things

Getting started with Git

Workflow

0. write a paragraph or two for your awesome paper
1. stage the changes
2. write a commit message
3. commit

now repeat this process until your paper is ready

let's see it in action

- commit message is human-readable description of change
- Git also gives you a diff
- commits have a unique identifier

Try it yourself

- install GitHub Desktop
- create a repository
- create a text file
- commit the text file

now you are good to work on your own

or:

- `https://docs.github.com/en/get-started/getting-started-with-git/set-up-git`
- `https://docs.github.com/en/get-started/start-your-journey/hello-world`
- `https://skills.github.com/`

Collaborating

need a centralised location: GitHub

- push: move your changes to the central location
- pull: move changes from central location to your computer

you need to explicitly do this (unlike Dropbox or Overleaf)

I need a volunteer with GitHub account

- create and clone a repository on GitHub
- pull it
- create a commit
- push it

automatic merging whenever possible!

add collaborators in the settings of a GitHub repository

unlimited private repositories with unlimited collaborators

Conflicts

if you and collaborator edited the same location in a file: conflict

you have to resolve the conflict:

- choose between which changes to include
- conflicts are between markers:

```
>>>>>>>>>>>
first alternative
===========
second alternative
<<<<<<<<<<<
```

when Git no longer sees these symbols, the conflict is resolved and you commit the merge

More references

- https://idrissi.eu/post/git-1-preliminaries, https://idrissi.eu/post/git-2-theory, https://idrissi.eu/post/git-3-practice
- https://www.math.cmu.edu/~gautam/sj/blog/20130929-git-quickstart.html

Best practices in LaTeX for version control

use short-ish lines: up to 100 characters

- better diffs
- your LaTeX code looks like poetry
- insight into sentence structure

alternatively: 1 sentence per line

atomic commits

- 1 commit = 1 change
- makes it easy for your coauthors to keep track of why a change happened
- let's them ignore "irrelevant" commits (like typos) and tells them where to pay attention

Other things

you can commit bits and pieces

- stage pieces of a file
- keeps commits atomic

you can discard changes:

- if you don't like a change before you've committed it, just get rid of it
- be careful: this is destructive

Things not to worry about

reading about Git can be daunting: it has many features:
mathematicians can ignore almost all

- I rarely use branches
- I rarely use pull requests
  however, I do like to comment on GitHub: collaborator immediately sees what you are referring to!

Some highly recommended tricks

.gitignore

LaTeX creates all these pesky temporary files: nightmare in Dropbox

Git can easily ignore these: `.gitignore`

`https://github.com/github/gitignore/blob/main/TeX.gitignore`

latexdiff

create a pdf that highlights changes

integrates well with tags

- a tag is a human-readable label for a commit
- create tags for important versions: arXiv submissions, journal submissions, ...

Git LaTeX template

`https://github.com/pbelmans/latex-template`

- my standard layout (you can and should use your own!)
- best practices implemented
- also some advanced things (see later)

Some more advanced Git things

GitHub Issues

originally for bug reports

also ideal for mathematics discussion:

- no more chaotic and lengthy email discussions
- cross-references

GitHub Actions for LaTeX

Overleaf has pdf view, Git is for source code management...

GitHub Actions allow you to

- start a virtual machine
- install and run LaTeX inside it
- save the resulting pdf to the repository

sounds complicated, but you don't have to do anything!
`https://github.com/pbelmans/latex-template/blob/main/.github/workflows/pdf.yml`

Hooks

allow you to do something extra whenever Git does something

- post-commit hook with commit id
- `https://ctan.org/pkg/gitinfo2`
- footnote with current version

useful when

- sharing over email
- printing
- finer than just the date

GitHub Pages

you can host static websites using GitHub:

- static site generator like Jekyll
- perfect for academic websites
- but also for documentation of projects

GitHub Actions for software development

`https://github.com/QuiverTools/QuiverTools/tree/main/.github/workflows`

- automatic documentation builds
- automatic linting, = verifying code style
- automatic testing of code