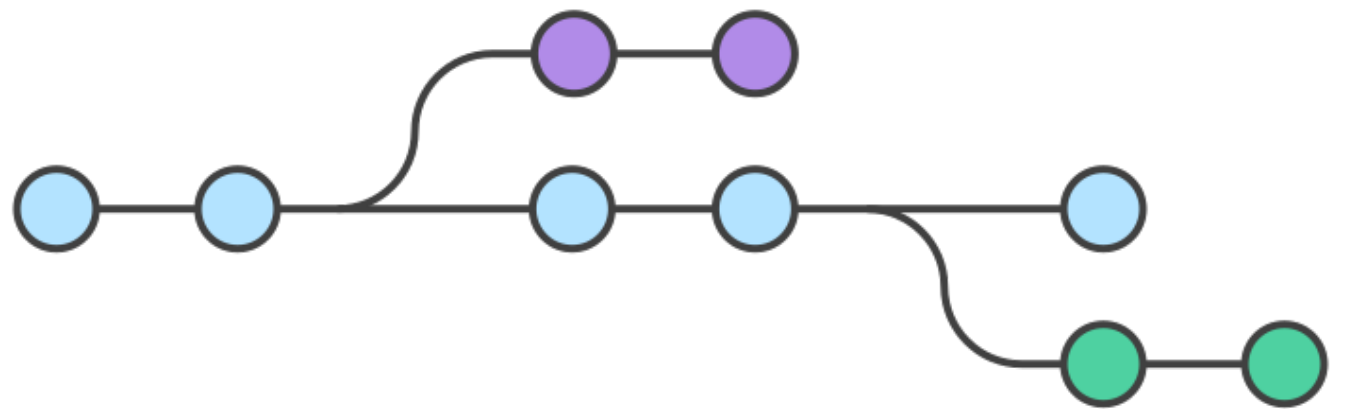


An introduction to version control with git



What is version control?

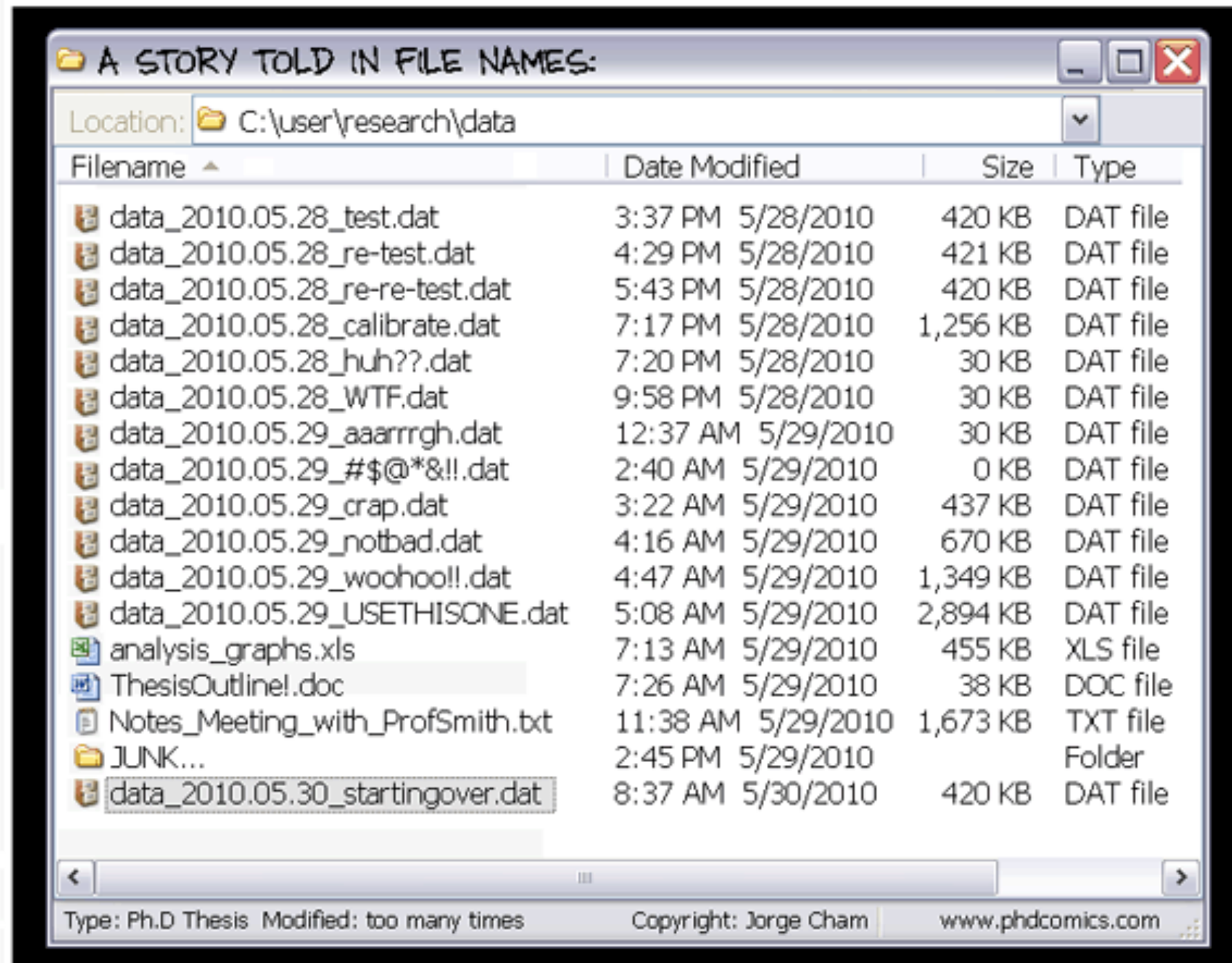
- Basic idea: Keep track of how information changes.
- Origins and main application: Software engineering.



-
- Inherent to Linux & Unix systems (e.g., macOS).
 - *git* is a command line tool.
 - Various front-ends for offline and online use (e.g., Github, Gitlab).

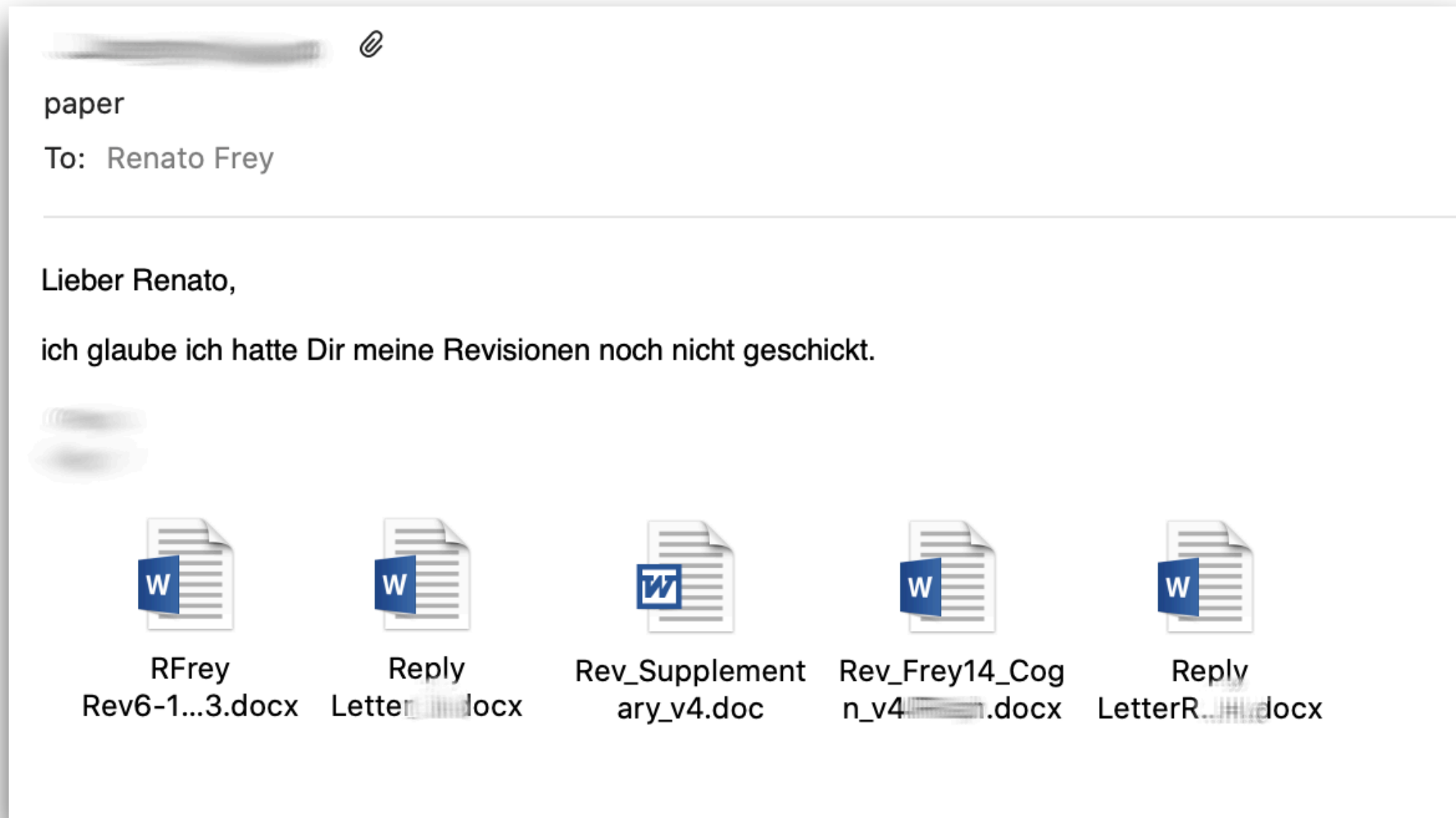
Why version control?

If you want to avoid this...



Why version control?

... or this ...



Why version control?

... and get that instead.

The screenshot shows a version control interface for a project named `risksr_latex`. The interface is divided into several sections:

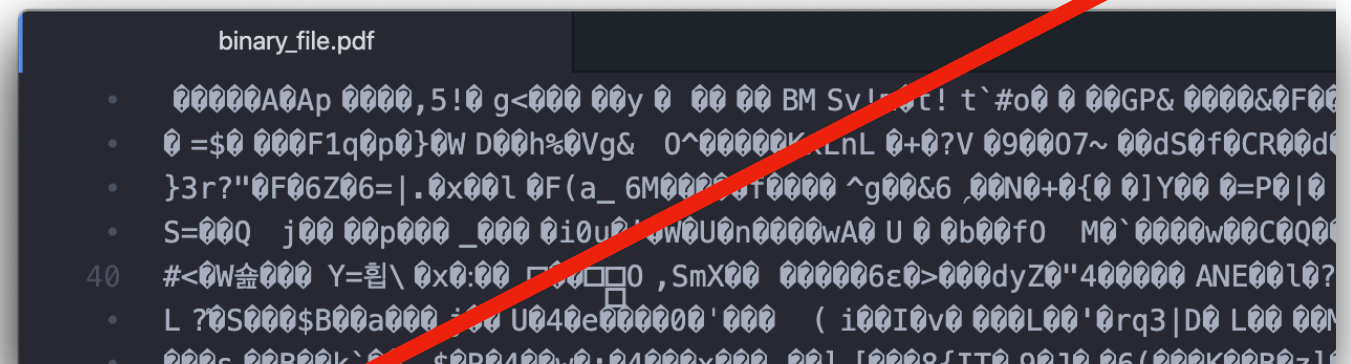
- Left Sidebar:** Contains navigation options: `Local Changes (4)`, `All Commits`, `Branches` (with `master` selected), `Remotes` (with `github` and `master` listed), `Tags`, `Stashes`, and `Submodules`.
- Commit History:** A list of commits with their authors, commit hashes, and dates. The commits are ordered chronologically, with the most recent at the top.

Author	Commit Hash	Date
mdsteiner	3852b1c	14 Aug 2020 at 16:13
mdsteiner	49ff2c5	14 Aug 2020 at 16:07
mdsteiner	4d0c290	14 Aug 2020 at 15:51
Renato Frey	945b4ca	14 Aug 2020 at 14:41
mdsteiner	d6484bf	14 Aug 2020 at 11:59
mdsteiner	c3714ae	14 Aug 2020 at 10:29
Renato Frey	c715924	13 Aug 2020 at 14:13
mdsteiner	02bd249	12 Aug 2020 at 19:54
Renato Frey	12be165	12 Aug 2020 at 18:13
mdsteiner	53b849a	12 Aug 2020 at 15:53
Renato Frey	20ba646	12 Aug 2020 at 09:53
mdsteiner	30103da	11 Aug 2020 at 13:38
- Changes:** A section showing the changes in the selected commit. It lists the commit hash, author, and date. The changes are categorized by file type (e.g., `main.tex`).
- Diff View:** A view showing the differences between the selected commit and the previous one. It displays the code changes in a diff format, with lines of code and their corresponding line numbers. The diff shows changes in the `main.tex` file, including a new section titled "Corresponding author: Markus D. Steiner, tract{" and a paragraph discussing risk preference and self-reports.

What can I track with *git*?

Anything that is text-based!

- **Code** that you write to **program your studies**: Python, PHP, ...
- **Data**, as long as it is in a raw format: .csv, .txt, ...
- **Code** that you write to **analyze your data**: R-scripts, ...
- **Text**, as long as it is in a raw format: Markdown, Latex, ...
- Rule of thumb: If the content of a file does not look like entirely cryptic, you are good to go!



Fine, but do I really need *git*?

Yes, absolutely! *git* helps you...

- **Structuring your work flow:**

Break down your work into small steps (e.g., “revise the abstract”) and track progress, instead of “working a little bit here and there”.

- **Getting rid of unnecessary stuff:**

Fearlessly delete sections of your work that you potentially no longer need — you could always go back, but usually will not!

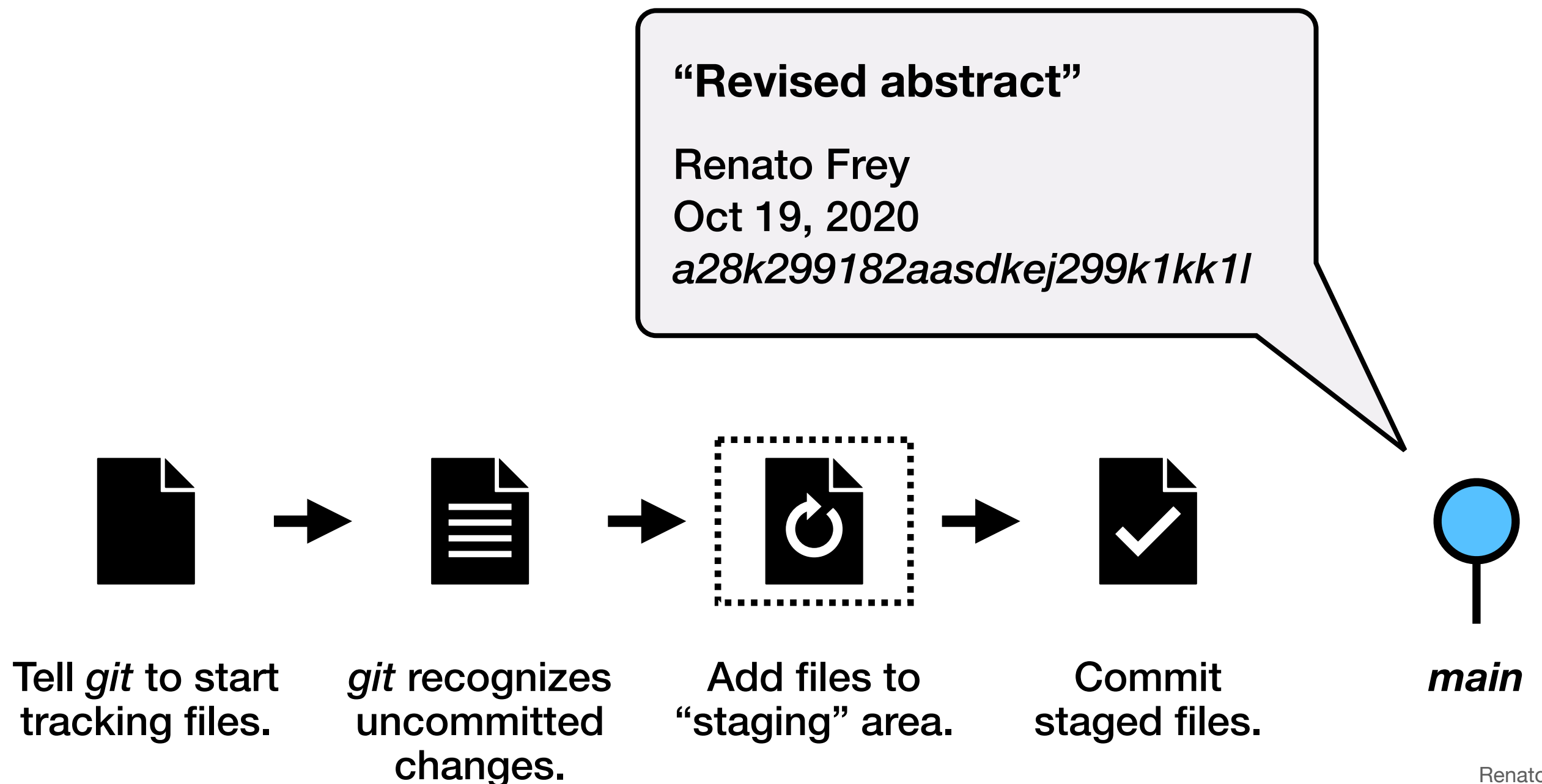
- **Fixing mistakes and reverting changes:**

Quickly fix errors and revert selected changes, such as when needing to switch back to a previous layout of your manuscript.

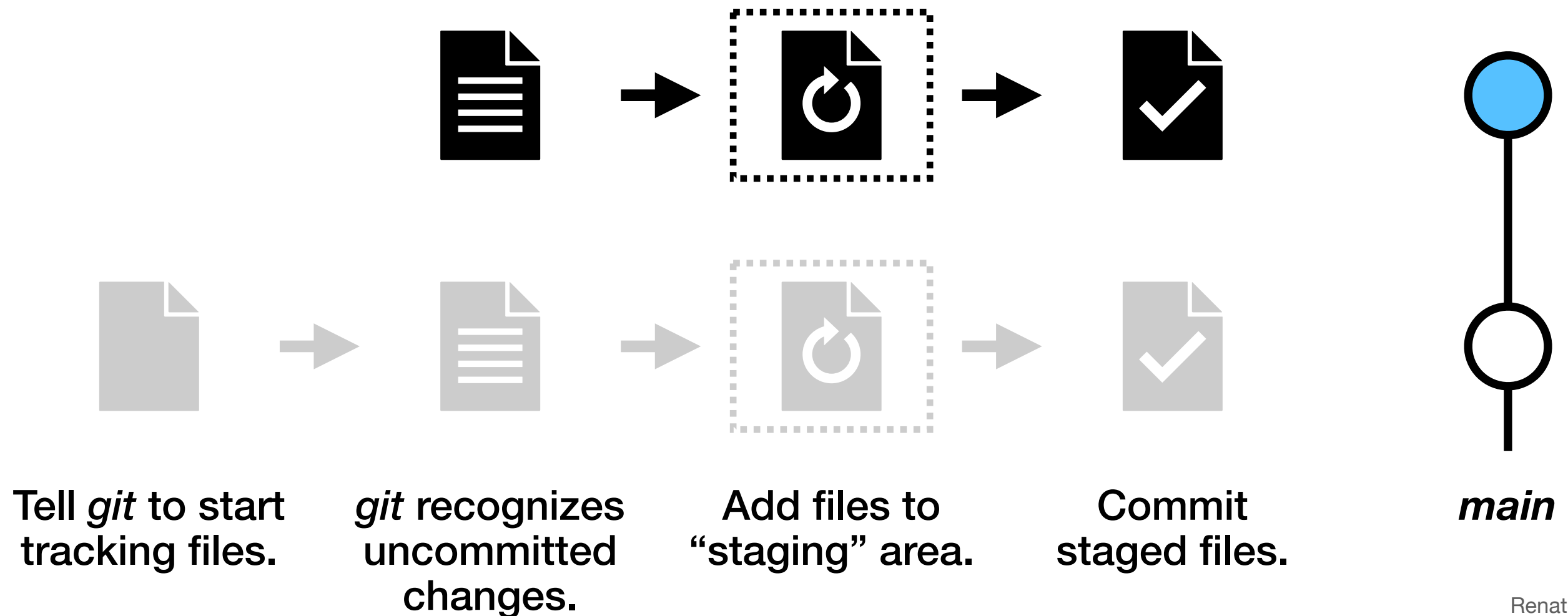
- **Boosting transparency and collaboration:**

Document who changed what and why, and keep everything in sync across computers and collaborators.

Basic work flow

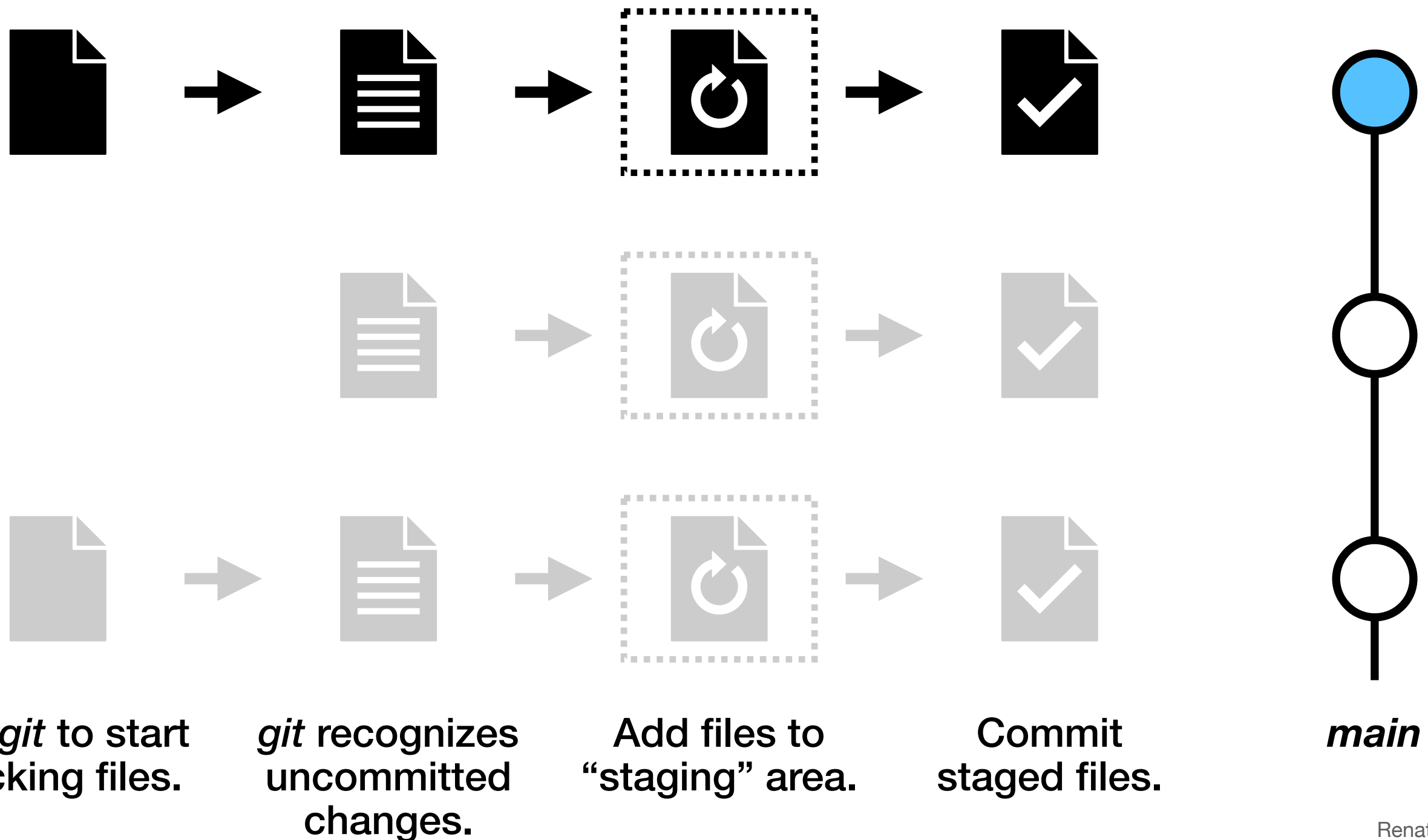


Basic work flow



Basic work flow

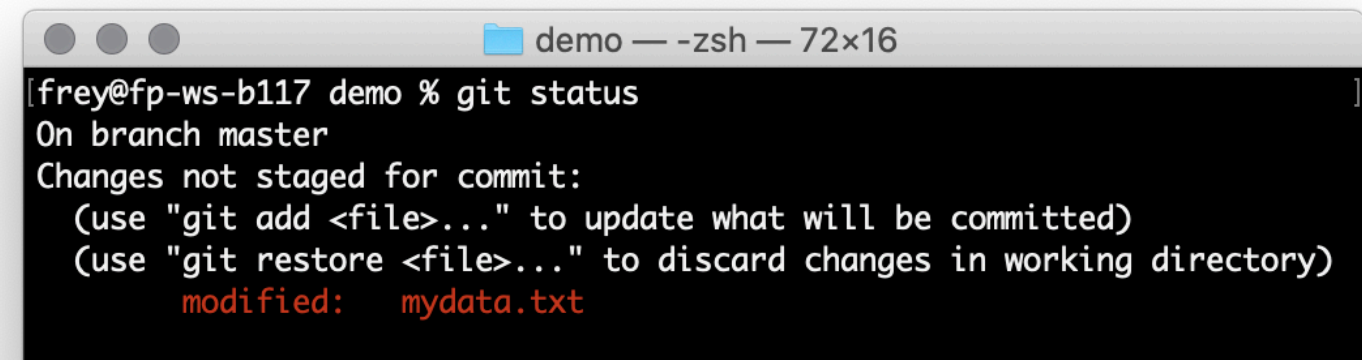
Commit (and push) frequently!



How to get started

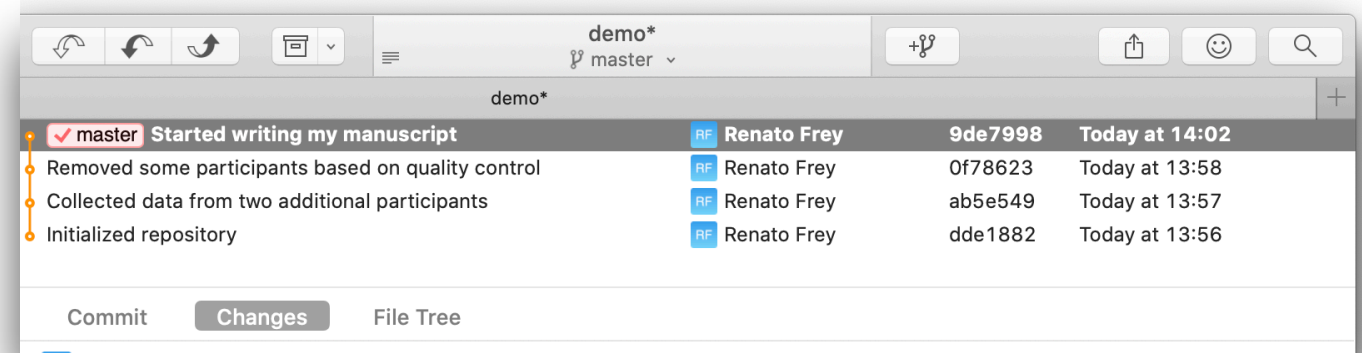
Learning by doing...

Step 1: Play around with some basic commands to get a feeling and intuition of how *git* works.

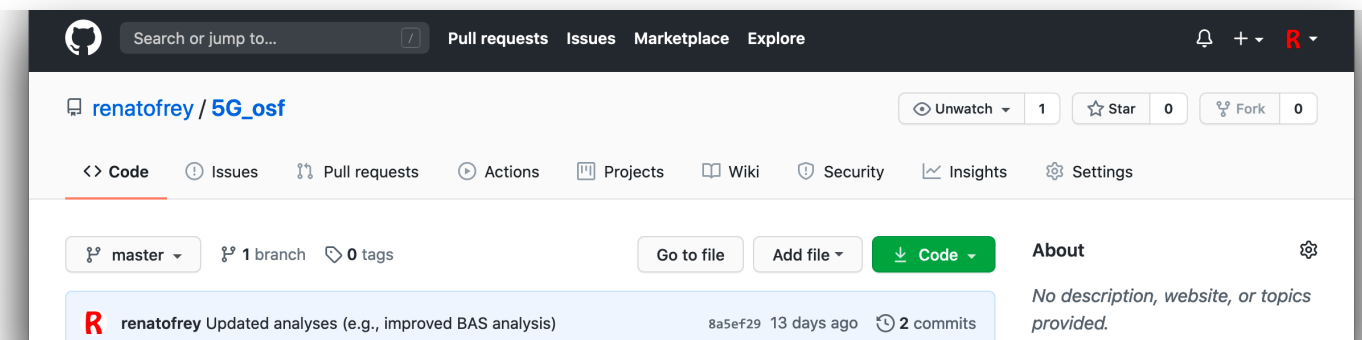


```
demo — -zsh — 72x16
[fre@fp-ws-b117 demo % git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   mydata.txt
```

Step 2: For everyday use, familiarize yourself with an offline front-end, e.g. git-fork.com or desktop.github.com.



Step 3: Start collaborating online using Github, Gitlab, or Overleaf.



Step 1

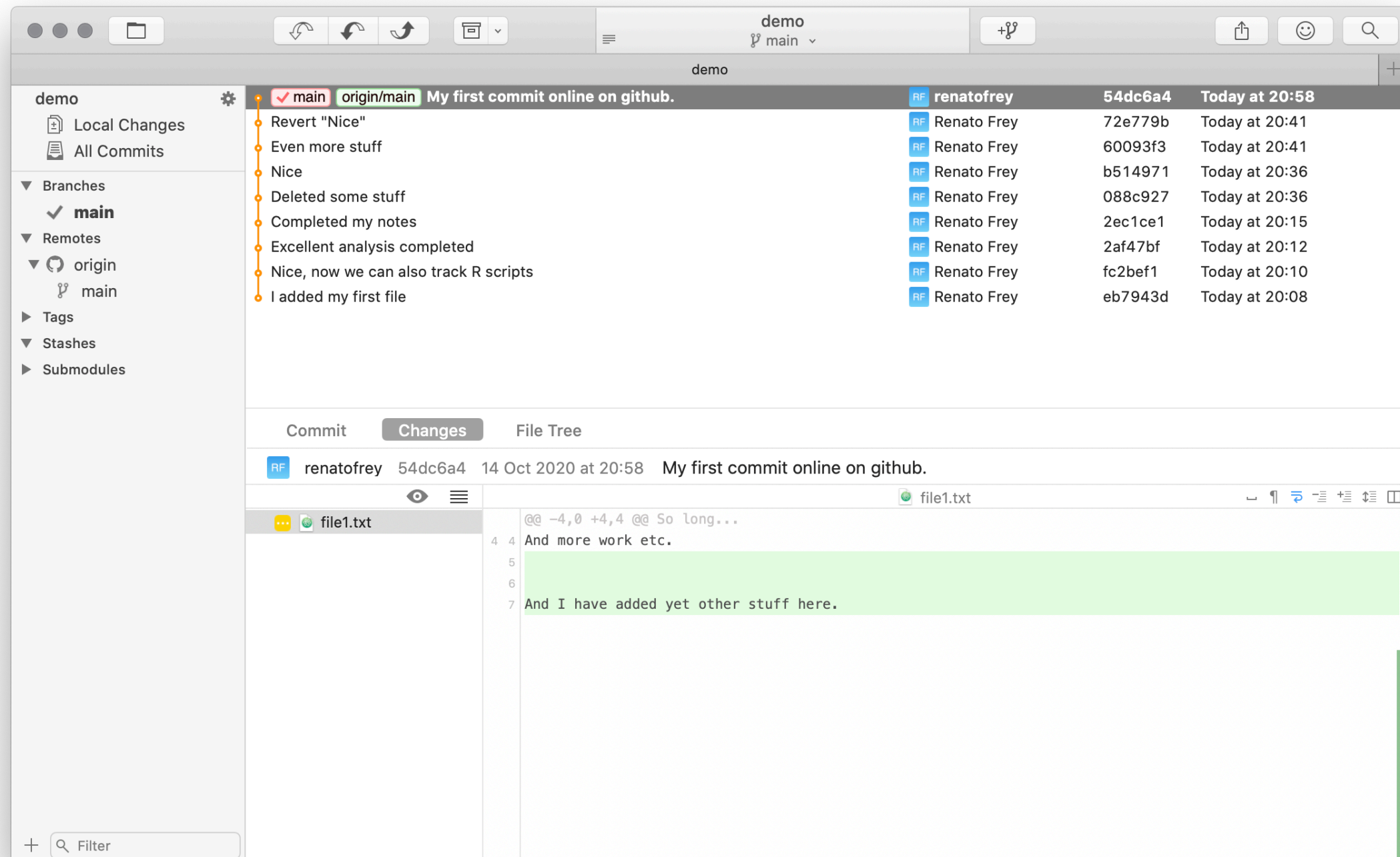
Learn some basic commands.

- Initialize project: `git init`
- Start tracking a file [all files]
and / or add modified files to
the staging area: `git add file1.txt` [`git add .`]
- Commit changes: `git commit -m "Edited intro."`
- Display the current status: `git status`
- Display what has changed: `git diff`
- Display commit history: `git log`

Step 2

Familiarize yourself with an offline front-end.

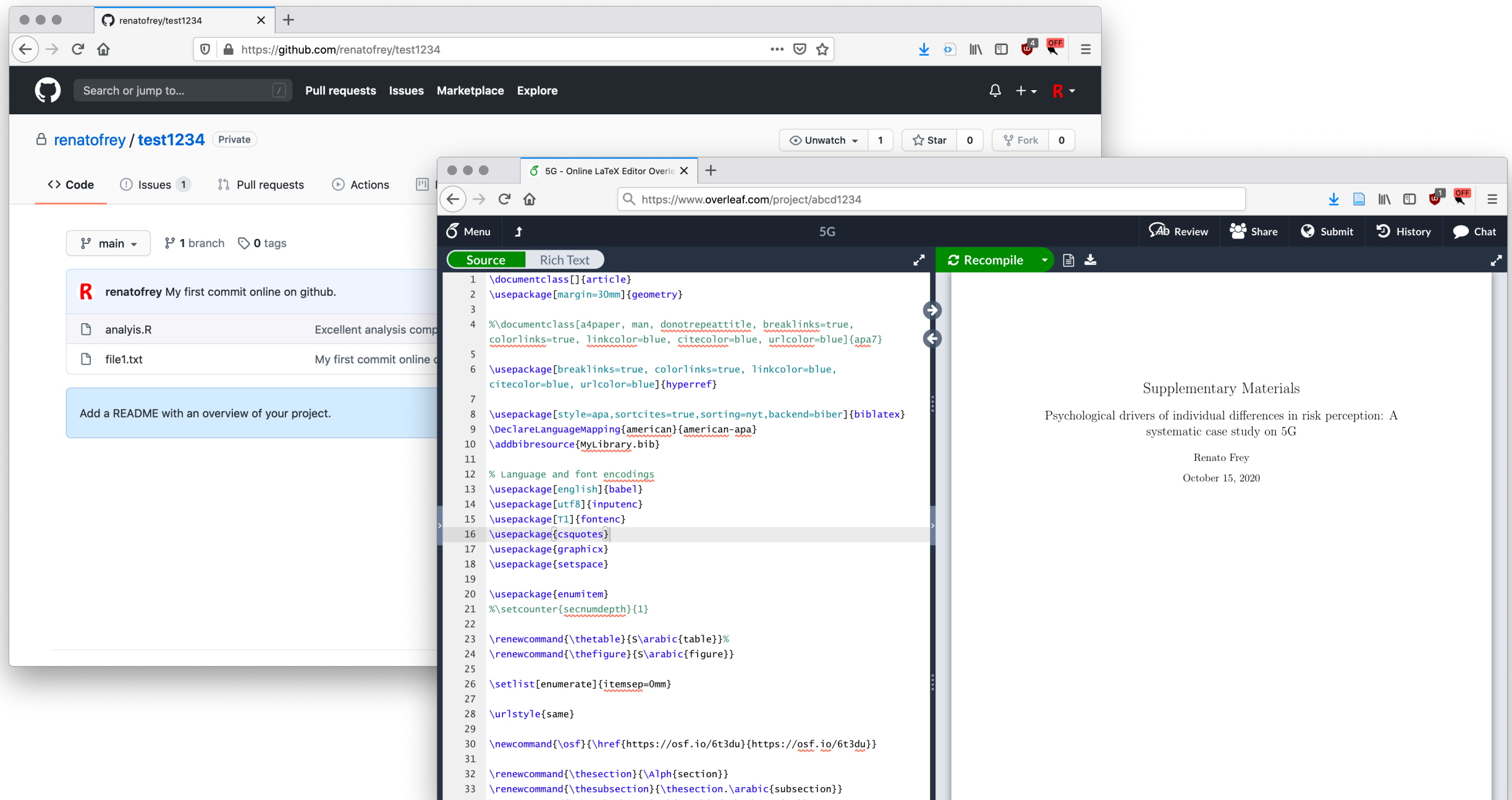
E.g., git-fork.com or desktop.github.com.



Step 3

Start pushing to and pulling from remotes.

E.g., github.com, git.scicore.unibas.ch, and overleaf.com.



Useful resources

Tutorials:

- git-scm.com
- atlassian.com/git

Remote services:

- github.com
- gitlab.com
- overleaf.com

Front-ends:

- git-fork.com
- desktop.github.com
- git-scm.com/downloads/guis
- en.wikipedia.org/wiki/Comparison_of_Git_GUIs