# Basics of version control

**Data plumbers' corner, session 7**

Arianna Masciolini
Språkbanken Text, University of Gothenburg

24 January 2025

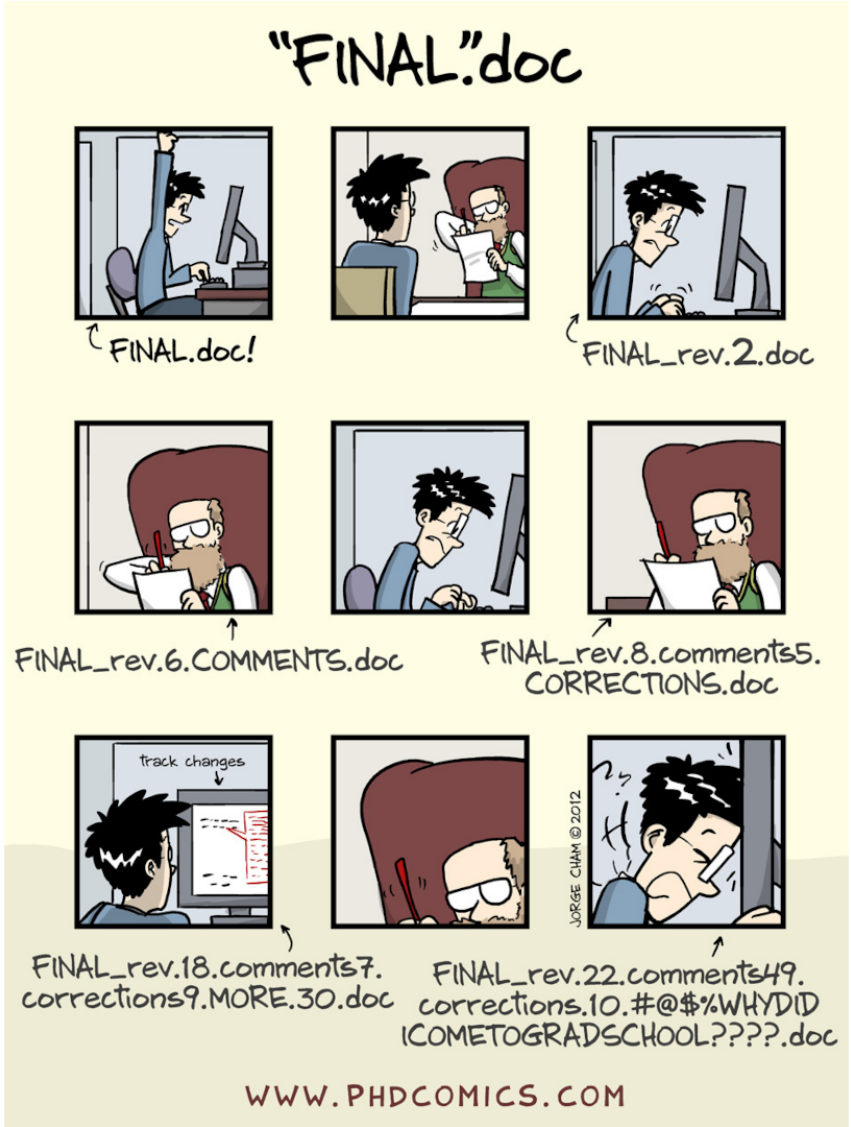
# Today's session

1. intro to version control and Git (again)
2. hands-on Git(Hub) tutorial

# What you will need

- a Unix-like shell
- the basic Git CLI program
- a GitHub account
- (if you use VSCode, I can recommend an extension called Git Graph)

# Manual versioning



# Version Control Systems

1. need to save disk space: **patch-based VCS** (e.g. RCS)
2. need for collaborative development: **centralized VCSs** (e.g. SVN)
3. need to work offline and prevent the server from being the "single point of failure": **distributed VCSs** (e.g. Git, the current *de facto* standard)

# Behind Git

Linus Torvalds, the creator of both Linux and Git

recommended: youtube.com/watch?v=4XpnKHJAok8

# Git: basic concepts


image from xkcd.com

- (named) filesystem snapshots rather than patches
- version history as a directed graph
- distributed, supports easy branching
- several interfaces and “hubs”

# Hands-on Git tutorial

1. **local usage** (simple versioning)
2. **setting up a remote GitHub repository** (versioning + backup)
3. collaborative usage with GitHub

# Basic commands overview

| `git` command | meaning |
|---|---|
| `init` | initialize a new repository in the current folder |
| `status` | see what files have been modified/added/removed |
| `add` | add file to staging area to include it in the next commit |
| `commit` | create a named snapshot |
| `stash` | saves current state on a stack of unfinished changes |
| `log` | see commit history |
| `checkout` | move to a different branch/commit |
| `push` | send changes to a remote repo |
| `pull` | download changes from a remote repo |