

# Basics of version control

Data plumbers' corner, session 7

Arianna Masciolini

Språkbanken Text, University of Gothenburg

# 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

"FINAL".doc



FINAL.doc!



FINAL\_rev.2.doc



FINAL\_rev.6.COMMENTS.doc



FINAL\_rev.8.comments5.  
CORRECTIONS.doc



FINAL\_rev.18.comments7.  
corrections9.MORE.30.doc



FINAL\_rev.22.comments49.  
corrections.10.#@\$%WHYDID  
ICOMETOGRADSCHOOL?????.doc

WWW.PHDCOMICS.COM

# 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](https://youtube.com/watch?v=4XpnKHJAok8)

# Git: basic concepts



	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT  
MESSAGES GET LESS AND LESS INFORMATIVE.

image from [xkcd.com](http://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