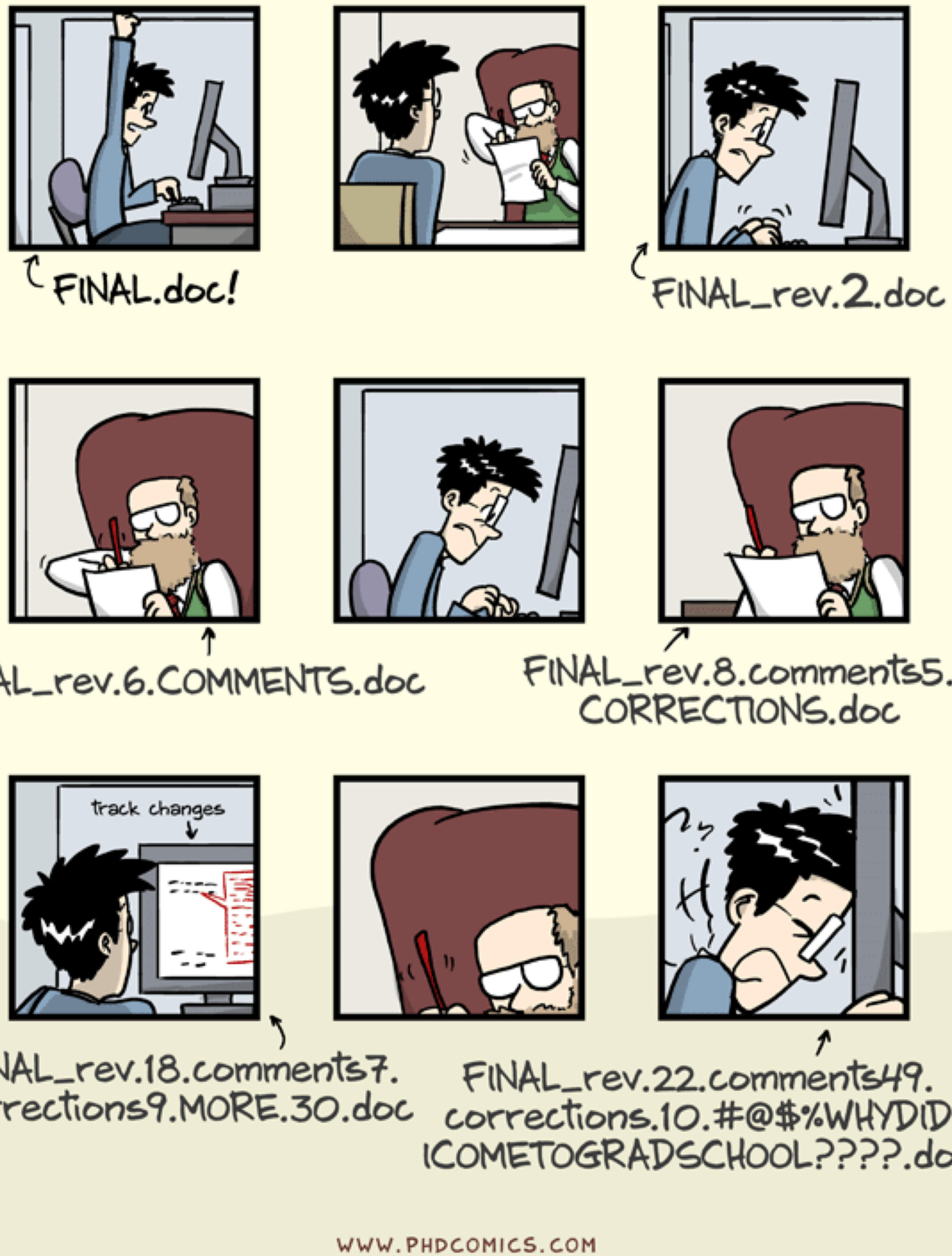
THE WHY AND HOW OF VERSION CONTROL

Juulia Suvilehto
D.Sc.(Tech)
2020

"FINAL".doc



WHAT IS VERSION CONTROL

- A software tool which is used to manage changes (typically in code)
- Tracks all changes to code (including who did it, when it was done, and what the other files looked like at that point in time) in a special database

Version control is a very powerful backup tool

(-> turbocharged undo-button)

WHY WOULD SHOULD YOU USE IT (ESPECIALLY IN CODING PROJECTS)

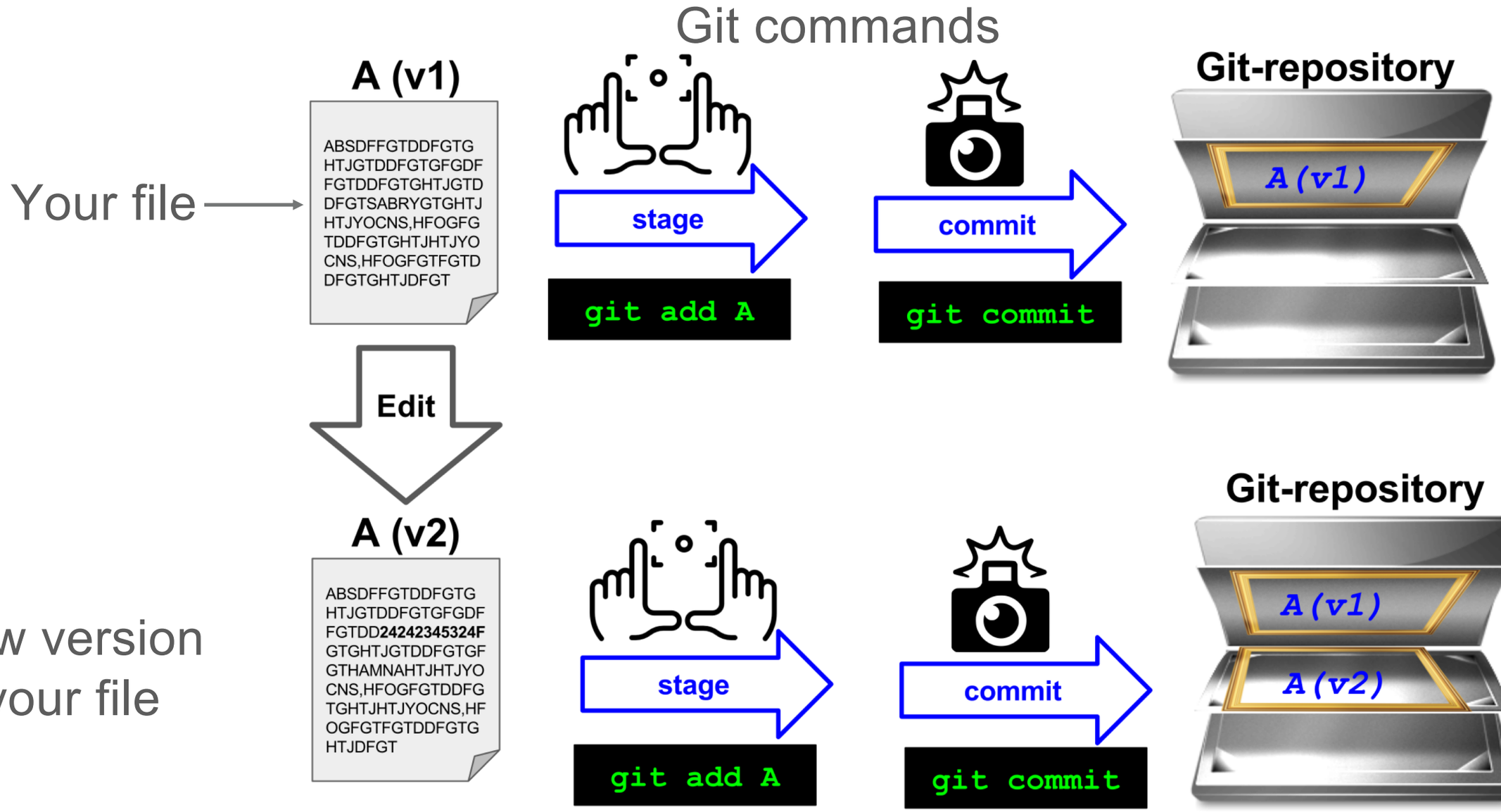
- Everyone has the latest version of the code all of the time
- Multiple people can work on the same project (and even the same file!) simultaneously
- Track changes and ownership
- Have snapshots of whole project at a given point in time, not just a single file
- Create sandboxes with branching

GIT EXPLAINED

WHAT IS GIT

- The most popular version control tool
- A small program which runs on your own computer
- You can use a separate server (a **remote**) to store a copy of your code. This enables
  - Sharing code with collaborators
  - Having a back-up of your code even if your own computer gets destroyed
- Popular online hosting services for repositories are GitHub and GitLab

HOW GIT WORKS & SOME VOCABULARY



Quick vocabulary

stage: focus the “camera”

commit: take the snapshot

GITHUB AND OTHER REMOTES

REMOTES

Your own computer

Out there, in the big internet

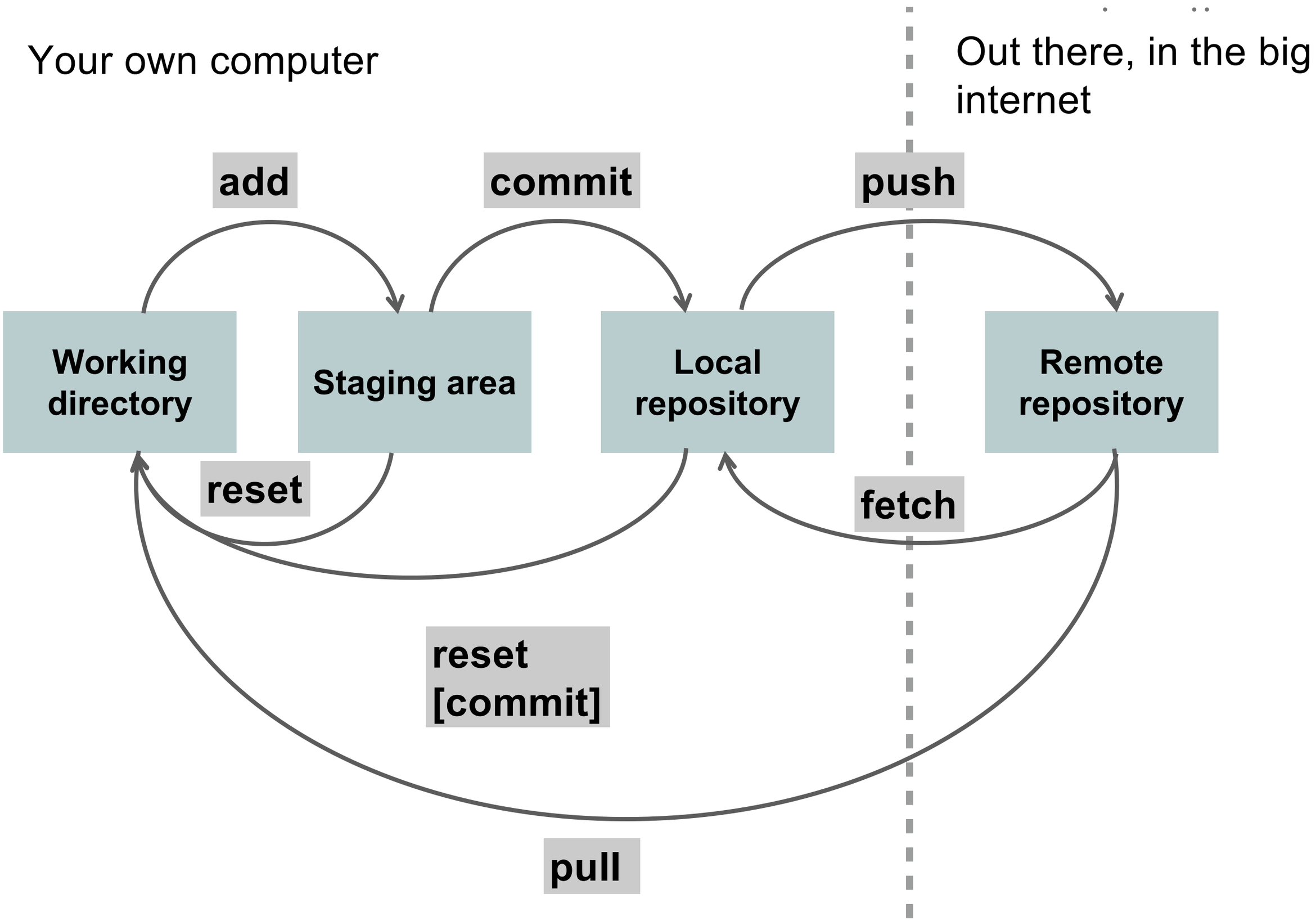
Working directory

Staging area

Local repository

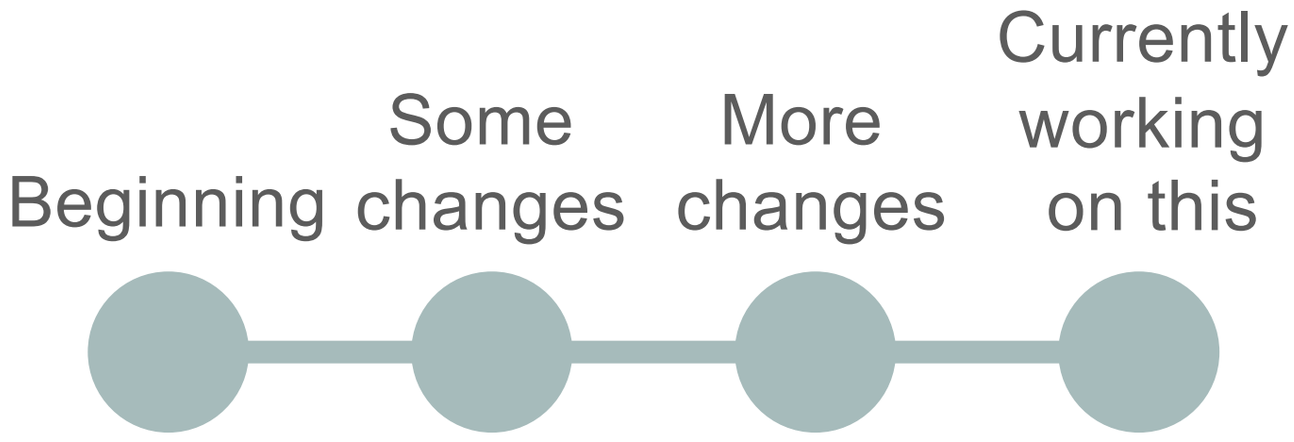
Remote repository

REMOTES


Your own computer
Out there, in the big internet
add
commit
push
Working directory
Staging area
Local repository
Remote repository
reset
fetch
reset [commit]
pull

SOME WORKFLOWS

BASIC WORKFLOW



Basic workflow

All changes happen on the trunk

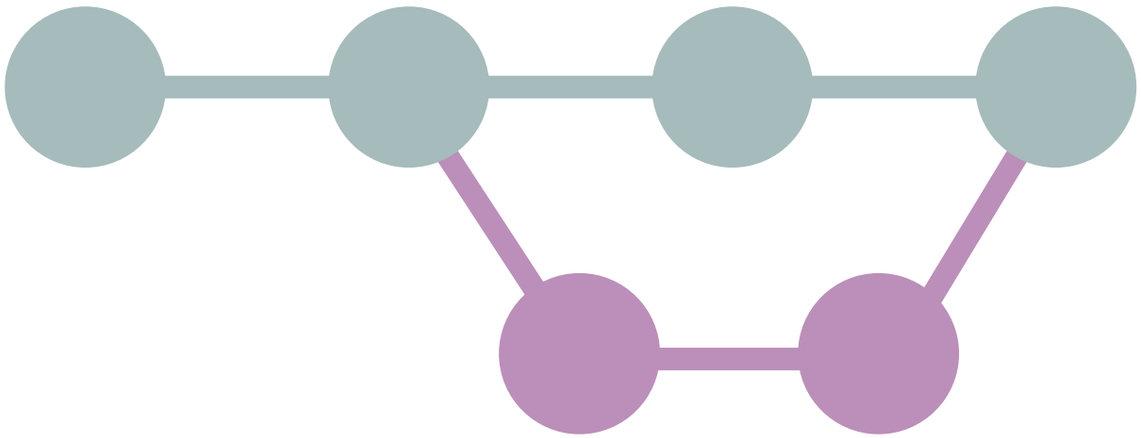
Works well for projects in the very beginning and solo projects

Quick vocabulary

trunk / main / master: The primary location for the code

branch: separate copy of the folder where you can work without disrupting the main

FEATURE BRANCH



Feature branch

New features get developed in their own branches, then merged back to master

Keeps code on master functional at all times, even if feature takes a while to develop

Makes collaboration easier

Quick vocabulary

trunk / main / master: The primary location for the code

branch: separate copy of the folder where you can work without disrupting the main

OTHER WORKFLOW OPTIONS

- Most companies & teams have their own preferred workflow, which depends on what type of projects they do
- The following two are too heavy for small projects, but you might encounter them if you end up working on an open source project or a software company
- Gitflow
  - Strict branching model for managing large projects
  - Similar to Feature Branch, but more rules for which branches to create and which branches things can be merged to
- Fork & Merge
  - Norm for open source development projects
  - Separate centrally managed repository and your own working repository
  - Enables strict control over whose code gets to the centrally managed repository, keeps the central code very tidy

QUICK DO'S AND DON'TS

DO

- Commit often
- Check status often
- Make commit messages good enough that you still know what was done in > 1 month
- Push/pull from the remote frequently, you never know what will happen to your local computer

DON'T

- Add any data to version control, ever
- Add any non-git(hub/lab) credentials to git
- Be afraid to try things - there are very few actions in git that are not undo-able. The system will warn you if you're about to do one of those

WORKING ON THE COMMAND LINE: USEFUL COMMANDS

- pwd ↵ (display path of current working directory)
- cd /some/directory ↵ (change directory to /some/directory)
- cd .. ↵ (move to parent directory
- ls ↵ (list contents of current directory)
- ls -la ↵ (list contents of current directory with more detail, including hidden files)

  ↵ stands for enter key

MORE RESOURCES FOR THE INTERESTED

- https://coderefinery.github.io/git-intro/
- http://swcarpentry.github.io/git-novice/
- https://happygitwithr.com/
- http://rogerdudler.github.io/git-guide/
- http://try.github.io/

- Most of these have fantastic tutorials as well, feel free to work through them if you’re interested in getting deeper into git