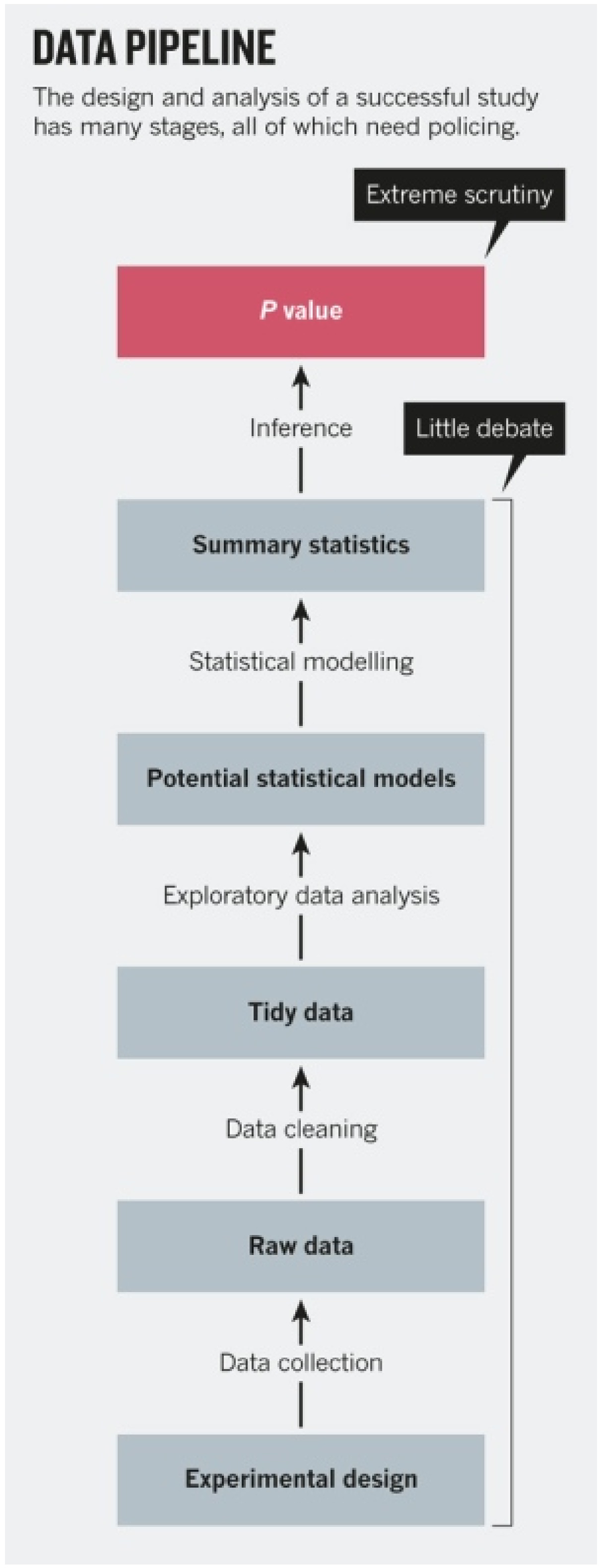
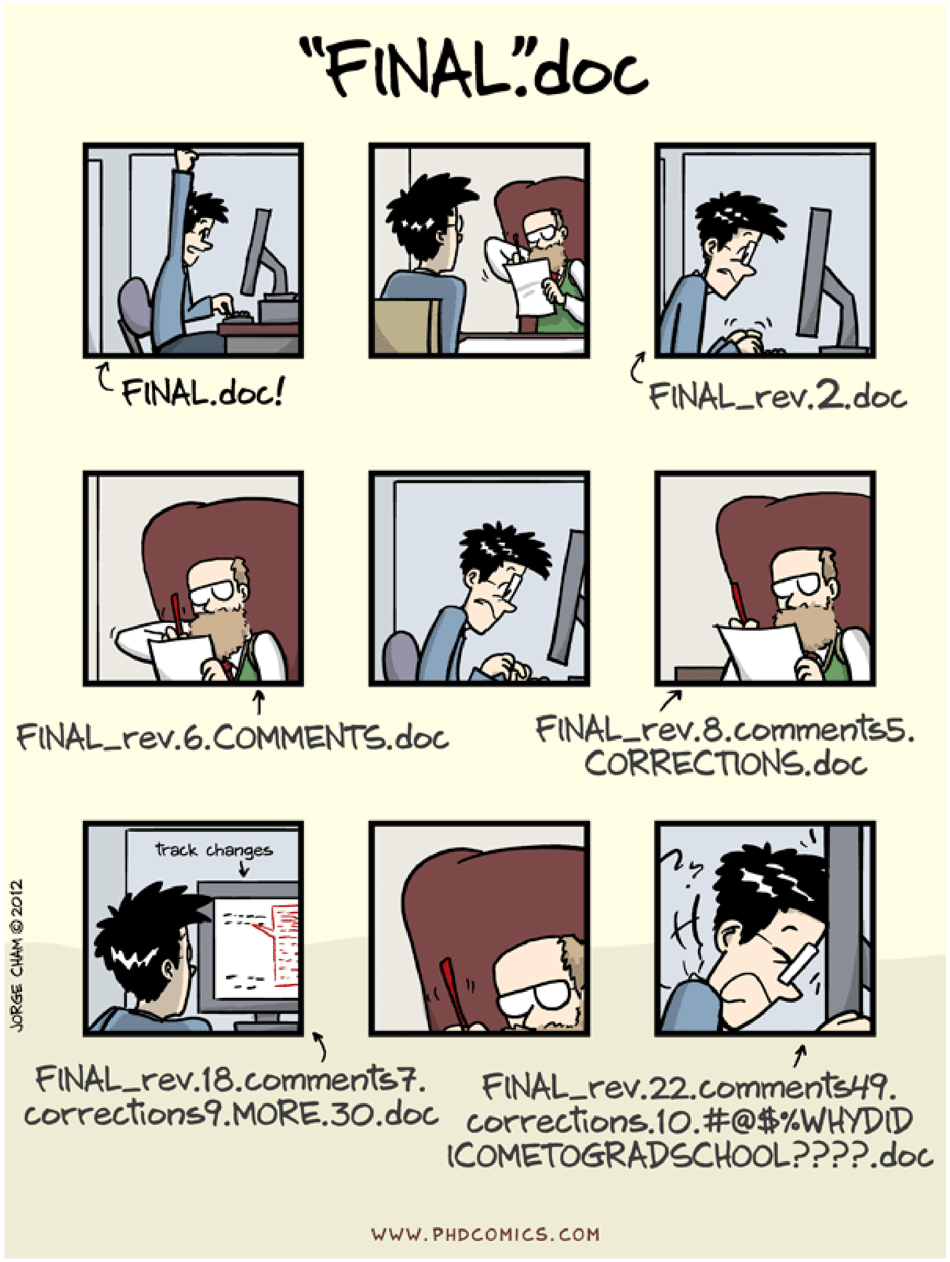


Reproducible Research

Leek & Peng (2015)
Nature

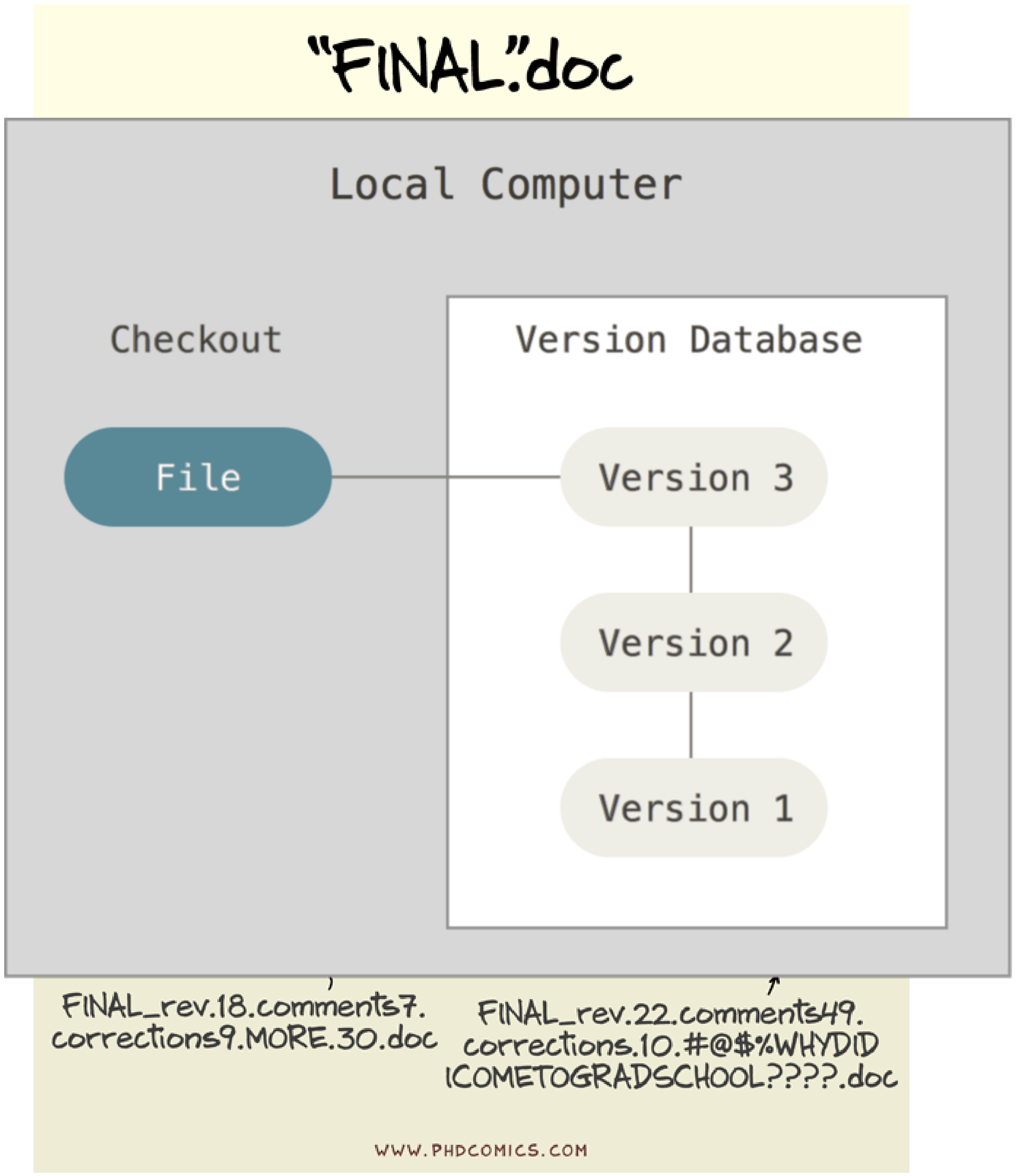
- Keep track of how results were produced
- Version control all custom scripts (git)
- Document your code (markdown)
- Provide public access to scripts and, data and results (github)
- Use open software when possible (R)
  - Compatible with multiple platforms

Version Control



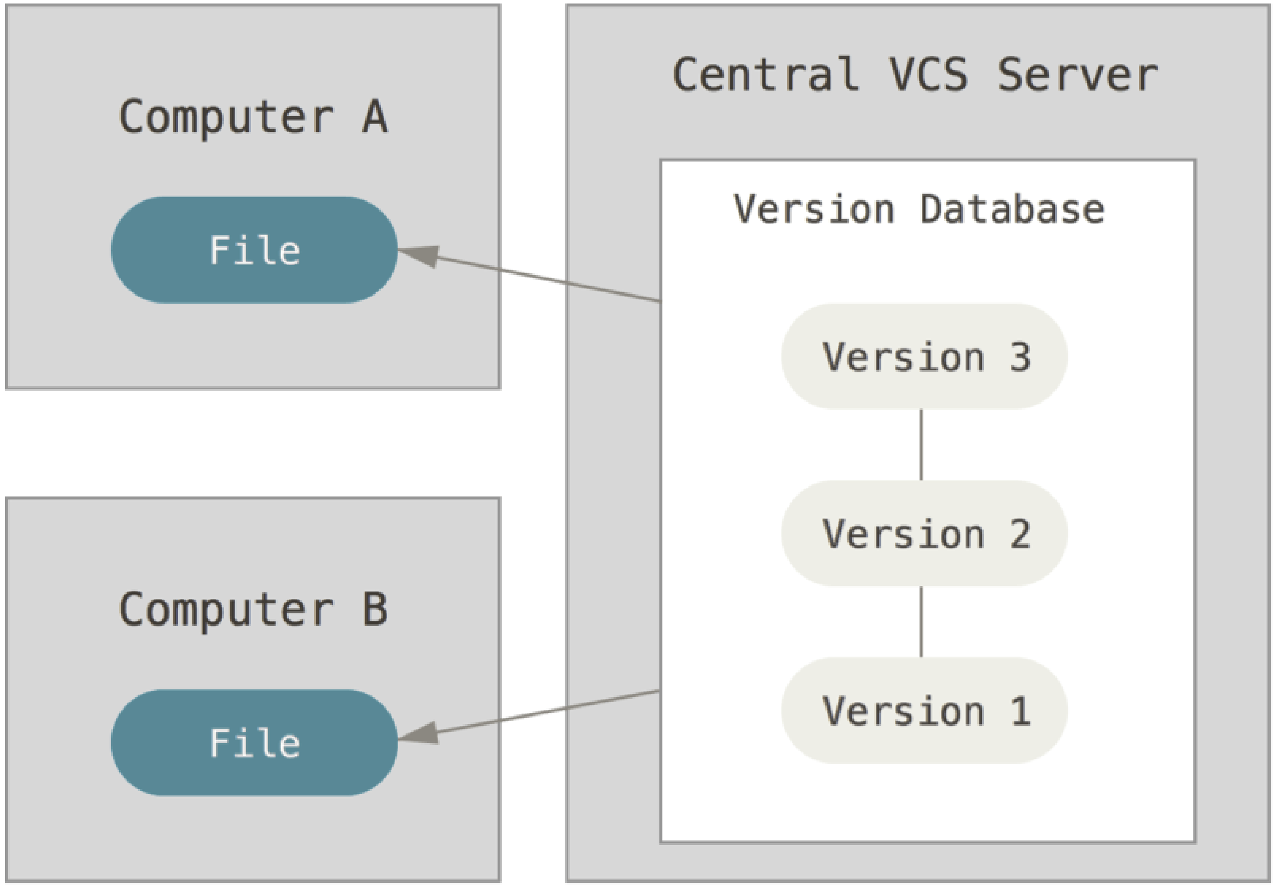
- Records changes to a file or set of files over time
  - Recall specific versions later
  - Revert files back to a previous state
  - Revert the entire project
  - Compare changes over time
  - See who last modified something that might be causing a problem
  - who introduced an issue and when, and more.

(Local) Version Control


"FINAL".doc
Local Computer
Checkout
File
Version Database
Version 3
Version 2
Version 1
FINAL_rev.18.comments7.corrections9.MORE.30.doc
FINAL_rev.22.comments49.corrections.10.#@$%WHYDIDICOMETOGRADSCHOOL????.doc
WWW.PHDCOMICS.COM

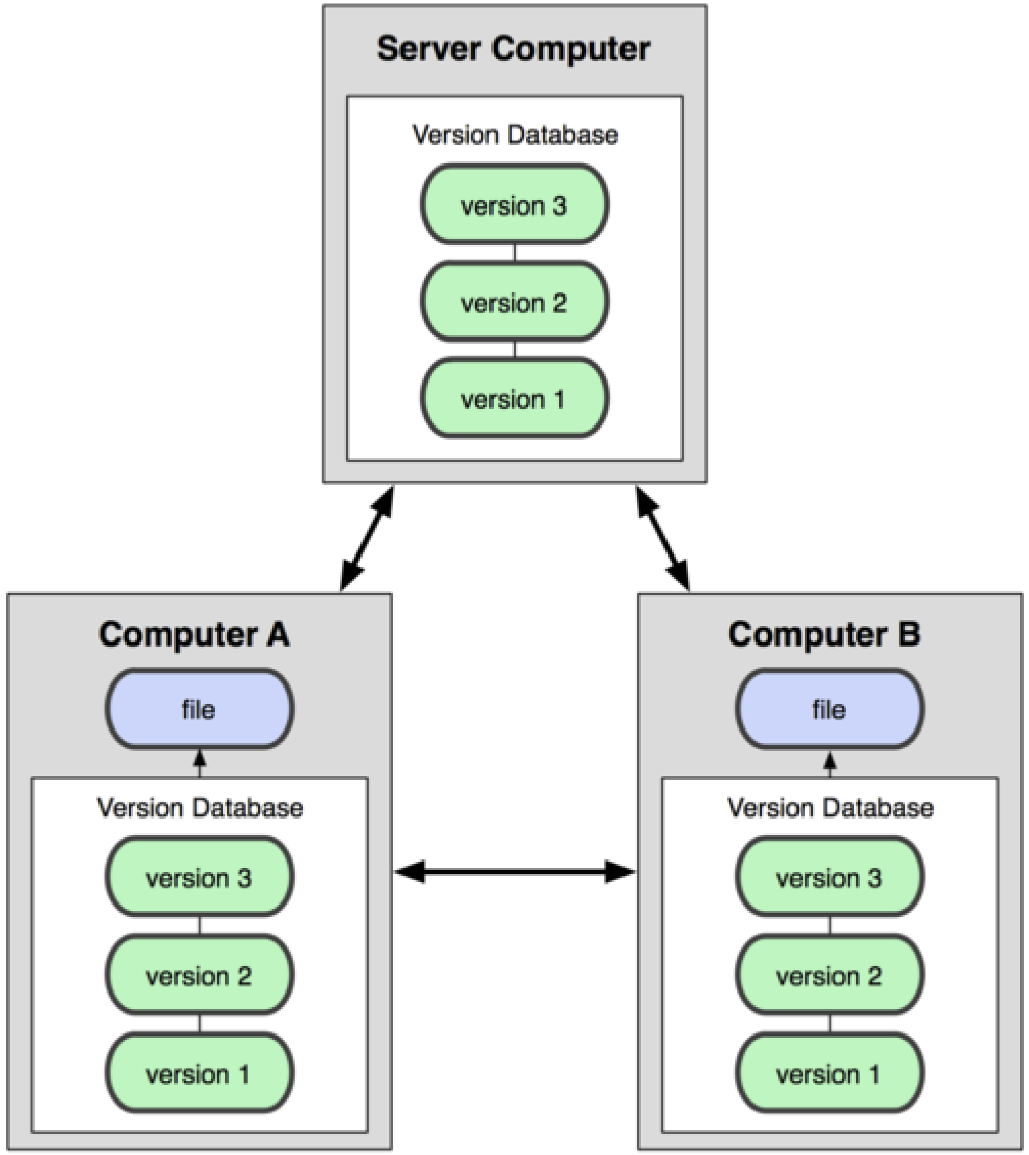
- Many people's version-control method of choice is to copy files into another directory
- Use specific software (Revision Control Software -RCS-)
- Not useful to collaborate with other people

(Centralized) Version Control



- Allowed collaboration
- All the versions are in the server
- If the server goes down no one can save changes
- If the hard disk the central database is on becomes corrupted you lose everything!

(Distributed) Version Control



- Allows collaboration
- Every client have all the versions
- If the server dies any client can restore the repository in the server
- Every clone is really a full backup of all the data.

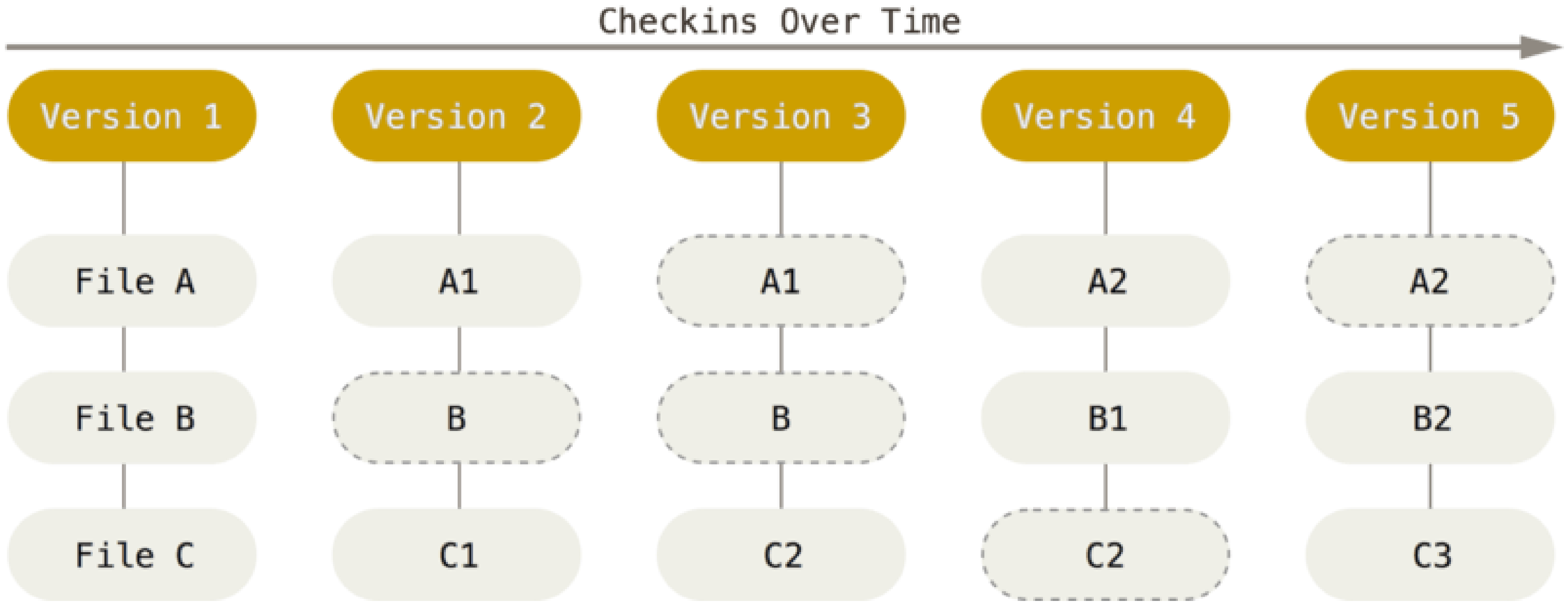
Git: version control tool developed by the Linux community

- Linux based systems are open source and therefore have a huge development community.
  - Thousands of people working on the same program at the same time

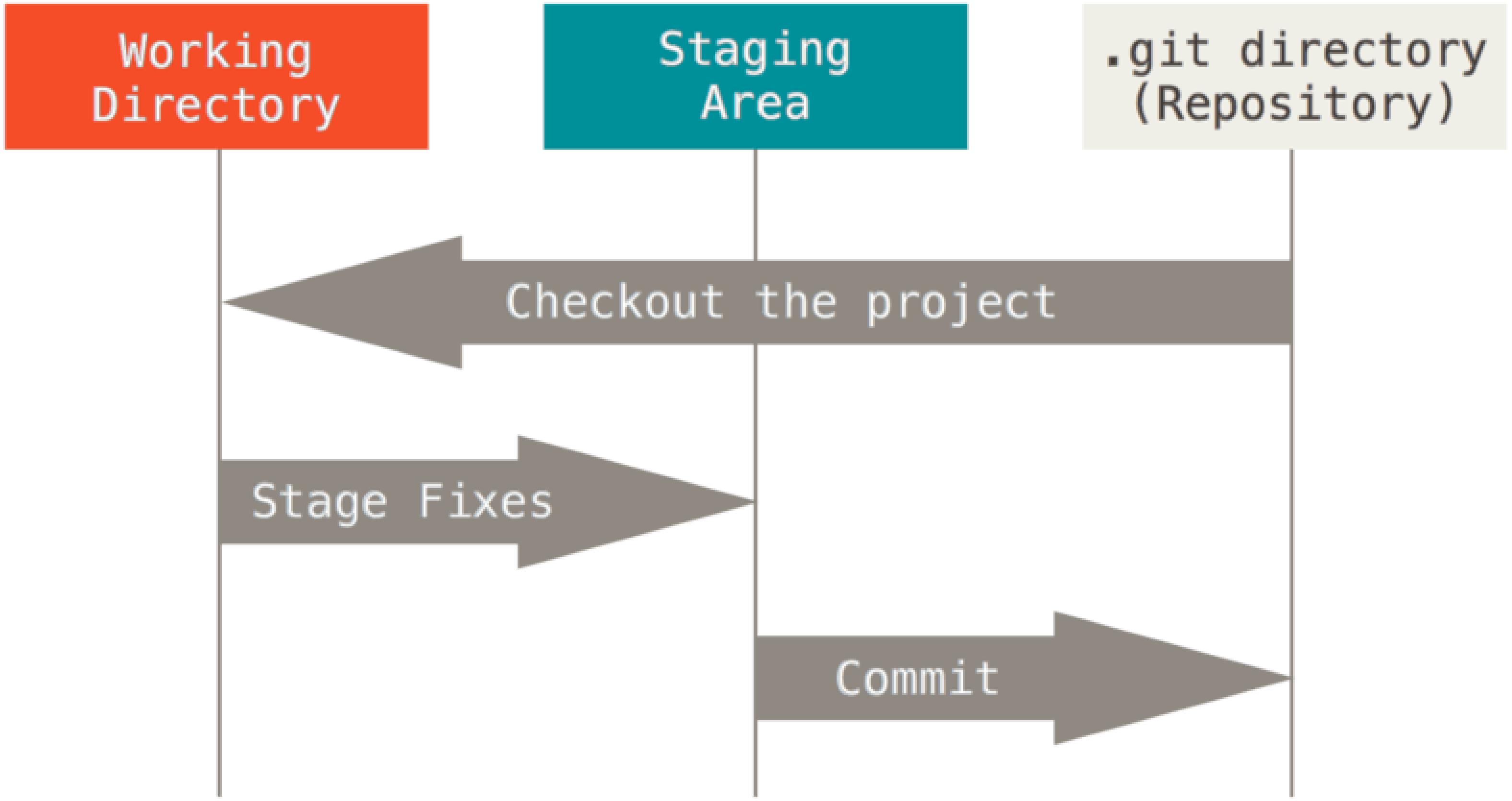
- They developed in 2005 a version control system with the goals:
  - **Speed**
  - **Simple** design
  - Strong support for **non-linear** development (thousands of parallel branches)
  - Able to handle large projects **efficiently** (speed and data size)

- But Git is not the only option

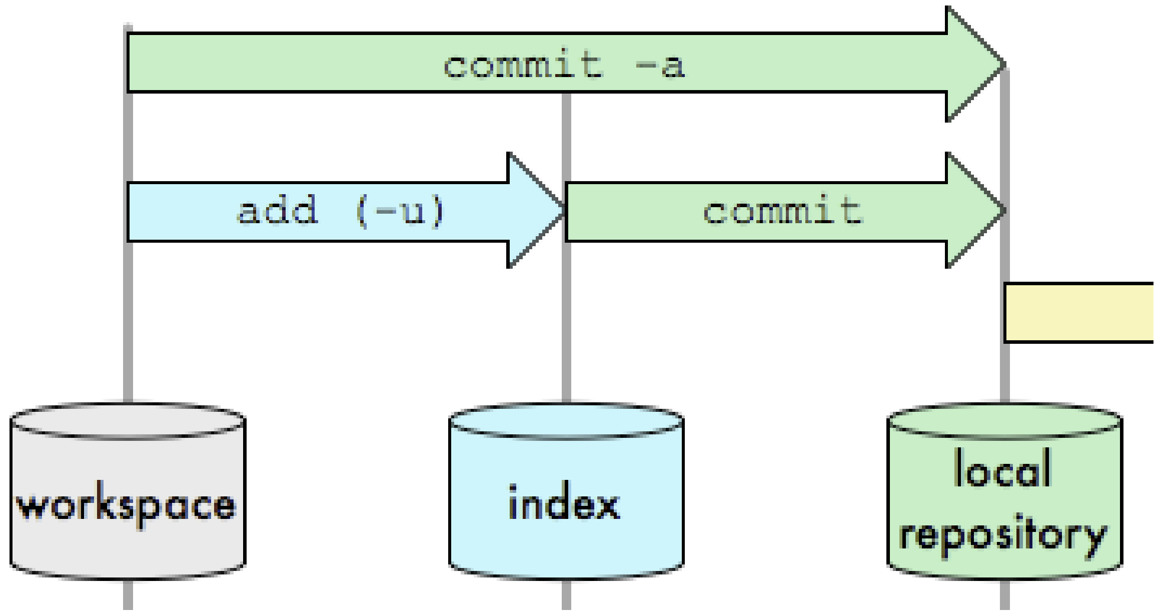
Git is based on “Snapshots”



Git has three main states that your files can reside in: committed, modified, and staged



(Local) Git work-flow



Or Working directory

1. Modify files

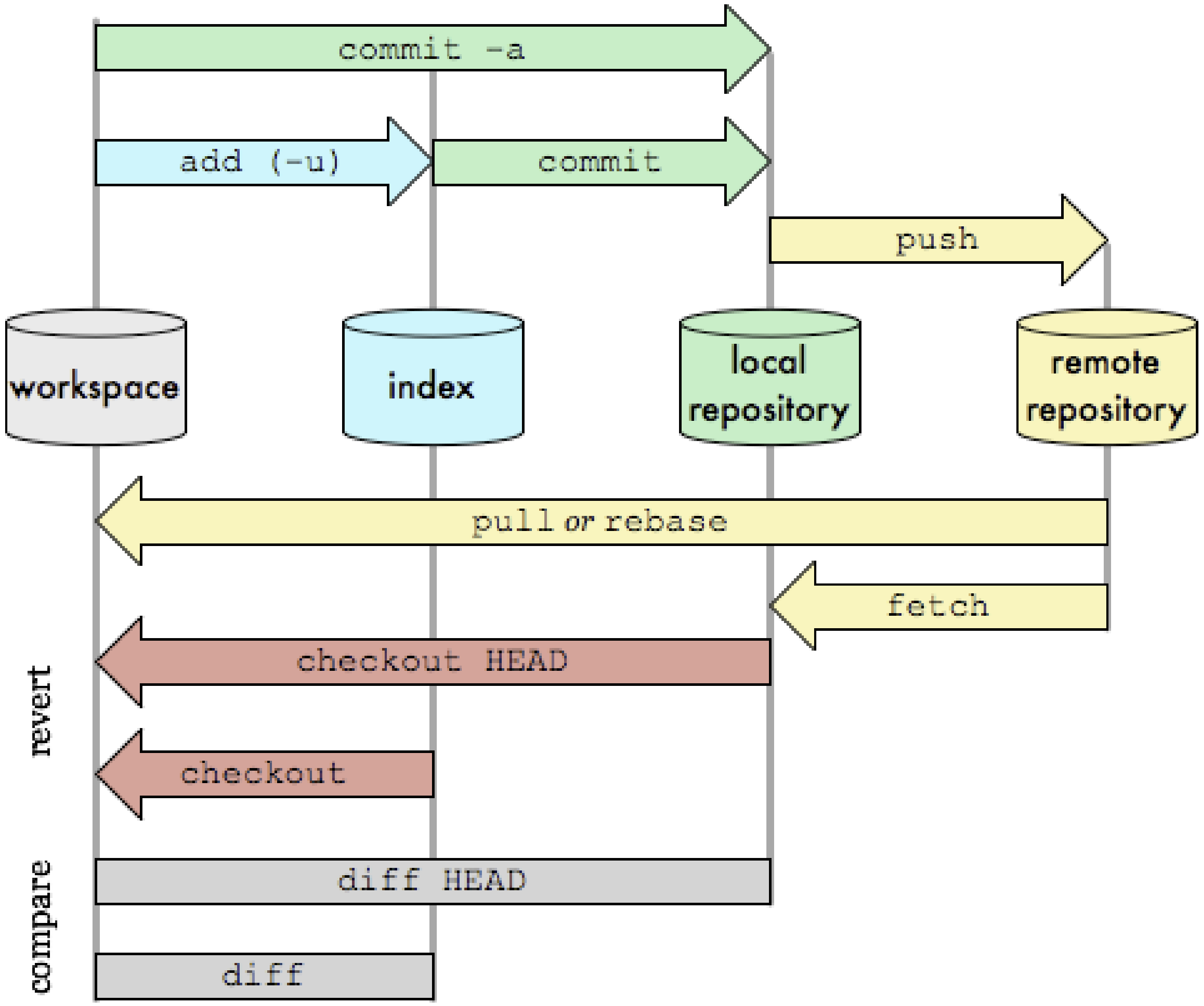
Or Staging area

2. You stage the files, adding snapshots of them

Or Git directory (all metadata)

3. do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory

(Remote) Git work-flow



Getting Started

- There are many GUIs, but we are using the Command Line.

1. Install GIT:

- Linux: `$ sudo yum install git-all` or `$ sudo apt-get install git-all`
- Mac:
  - Install the Xcode Command Line Tools (The Command Line Tool package gives terminal users many commonly used tools, utilities, and compilers, including make, GCC, clang, perl, svn, git, size, strip, strings, libtool, cpp, what, and many other useful commands that are usually found in default linux).
  - On Mavericks (10.9) or above try to run git from the Terminal. If you don't have it installed already, it will prompt you to install it.
  - Or get the binary installation file: http://git-scm.com/download/mac
- Windows:
  - Get the binary file: https://git-for-windows.github.io/
  - Or install installing GitHub for Windows http://windows.github.com

2. First-Time Git Setup

2.1 Your Identity: set your user name and email address. Every Git commit uses this information!

$ git config --global user.name "John Doe"

$ git config --global user.email johndoe@example.com

- Use the option --**global** to configure all the projects for your user. You will only need to do this once in each computer you are using.
- If you want to override this with a different name or email address for specific projects, you can run the command without the --global option when you're in that project.

2.2 Your Text Editor: you can configure the default text editor that will be used. If not configured, Git uses your system's default.

- Unix: $ git config --global core.editor emacs
- Windows:
  - x86 system: $ git config --global core.editor "'C:/Program Files/Notepad++/notepad++.exe' -multiInst -nosession"
  - x64 system: $ git config --global core.editor "'C:/Program Files (x86)/Notepad++/notepad++.ex

Help!!!

- $ git help <verb>
- $ git <verb> --help
- $ man git-<verb>

- $ git help **config**
- $ git **config** --help
- $ man git-**config**

- Look for resources online

3. Getting a Git Repository

Starting from scratch?

1) Take an existing project or directory and imports it into Git (creates a local git repository)

Using someone else repository?

2) Clone an existing Git repository from another server.

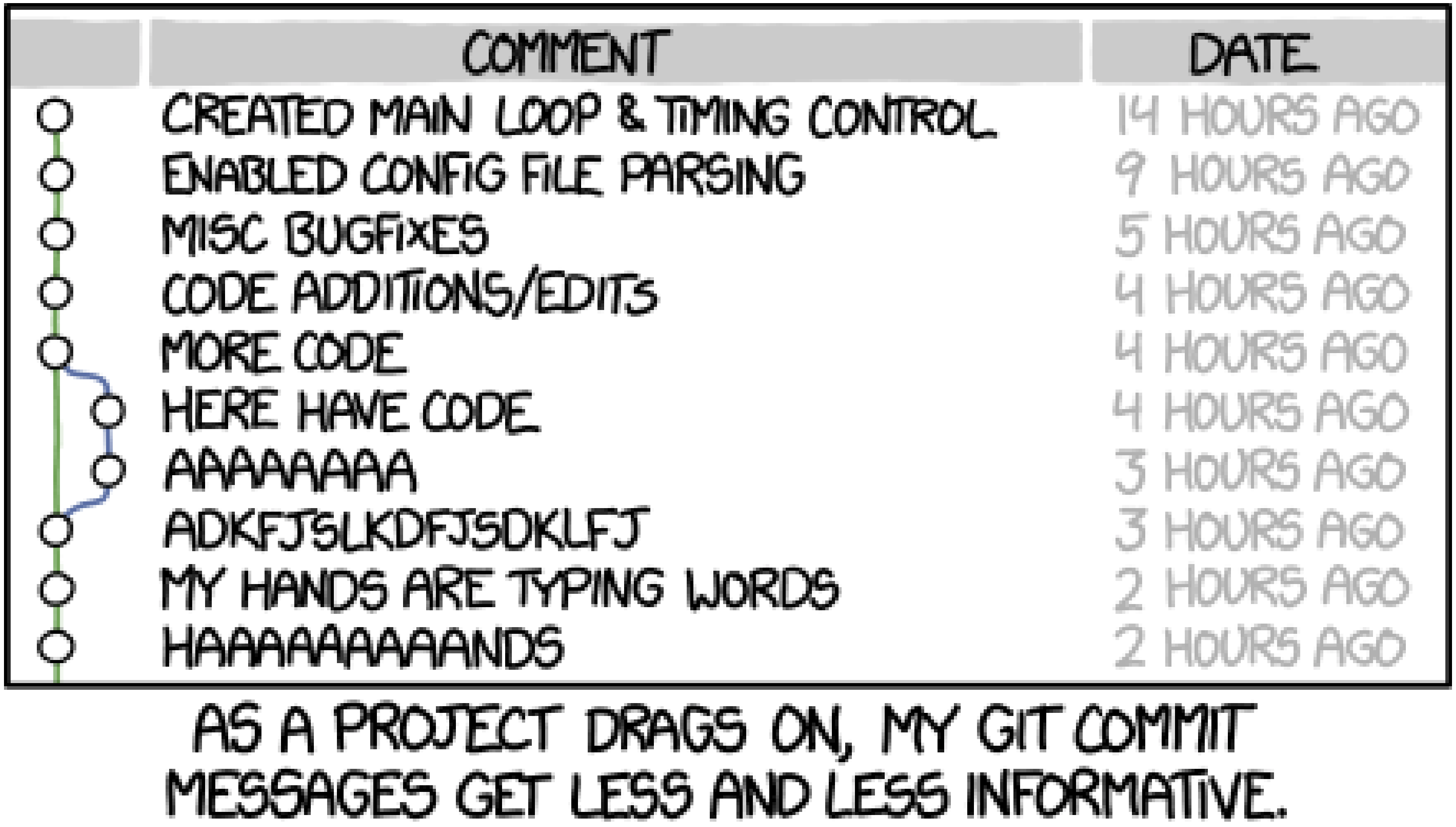
3.1 Initializing a Repository in an Existing Directory

- Go to the directory you want to track and create you repository skeleton:
  - $ cd /home/Ana/MyProject
  - $ git init

- Add the files you want to start tracking and make a initial commit:
  - $ git add ...
    - $ git add *.c
    - $ git add -A
  - $ git commit ...
    - $ git commit
    - $ git commit -a
    - $ git commit -m 'initial project version'

This message is important!

Commit comments should make sense!

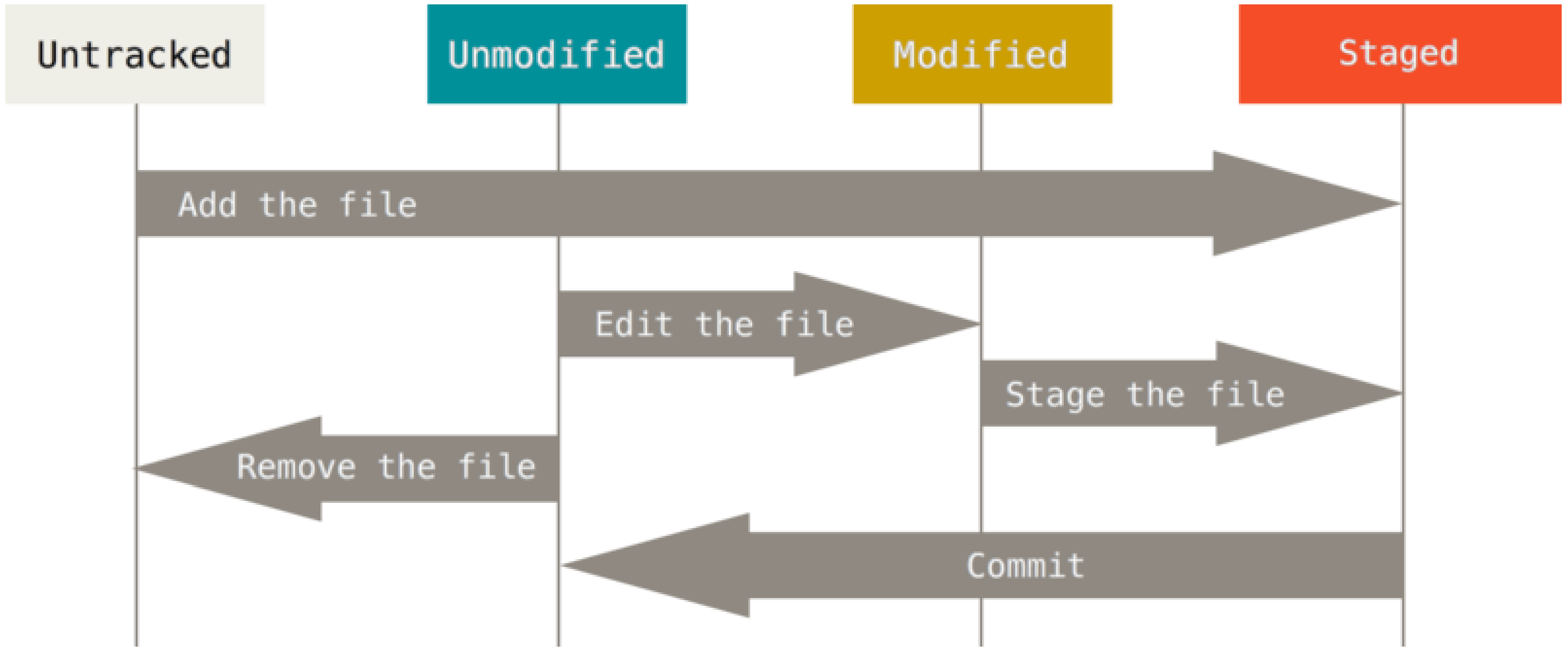


	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS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

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

Possible status of files in your repository

- $ git status # list all the files and their status
- $git add FILENAME # start tracking or stage a file
- Create a .gitignore file!!!!



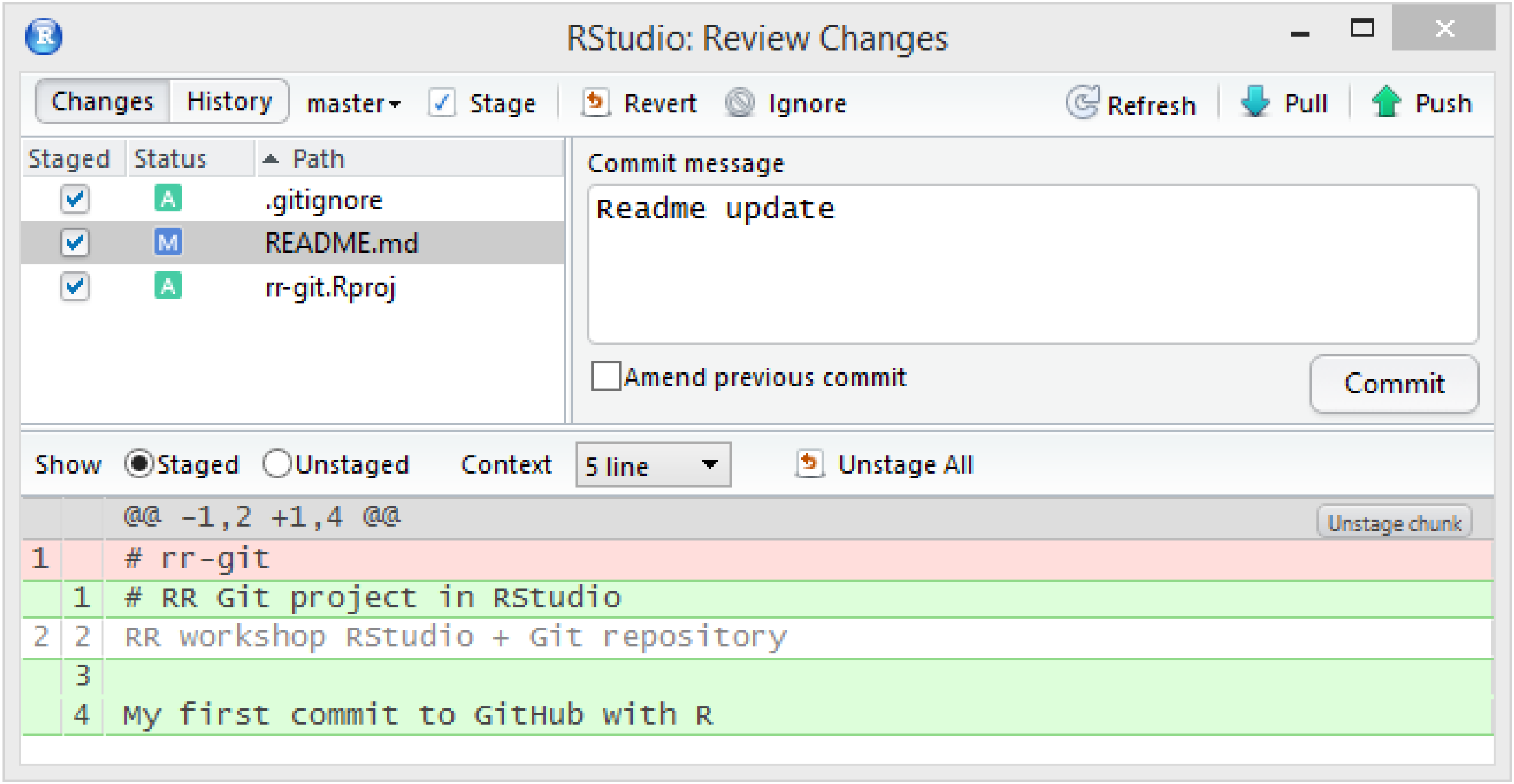
Now you can play...

- Project history, changes...
  - $ git log
  - $ git log -p
  - $ git diff
  - $ git diff MyScript.R
  - $ git status
  - $ git blame <filename>
- How to revert files to previous states?

Integration with other programs

- Git + Rstudio
  - https://support.rstudio.com/hc/en-us/articles/200532077-Version-Control-with-Git-and-SVN

    https://jennybc.github.io/2014-05-12-ubc/ubc-r/session03_git.html



Staged	Status	Path
✓	A	.gitignore
✓	M	README.md
✓	A	rr-git.Rproj

Commit message

Readme update

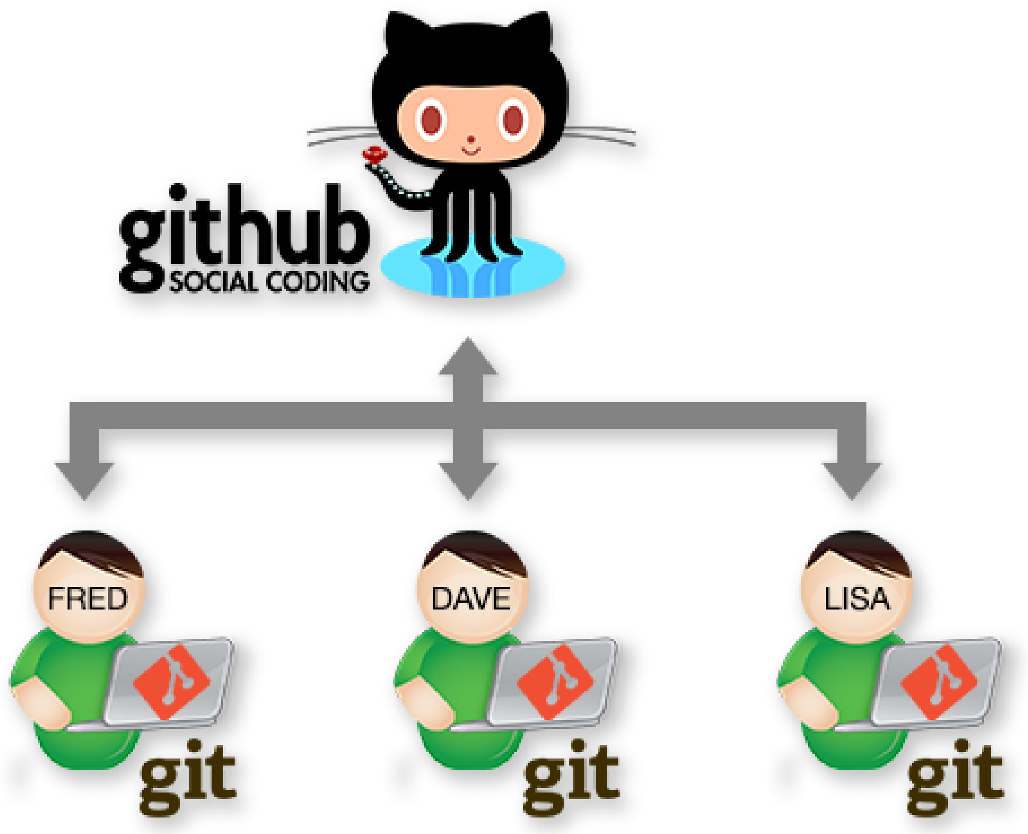
☐ Amend previous commit

Show [● Staged] ○ Unstaged Context 5 line Unstage All Unstage chunk

```
@@ -1,2 +1,4 @@
1     # rr-git
   1  # RR Git project in RStudio
2  2  RR workshop RStudio + Git repository
   3
   4  My first commit to GitHub with R
```

Integration with other programs

- Git + GitHub



3.2 Cloning an Existing Repository

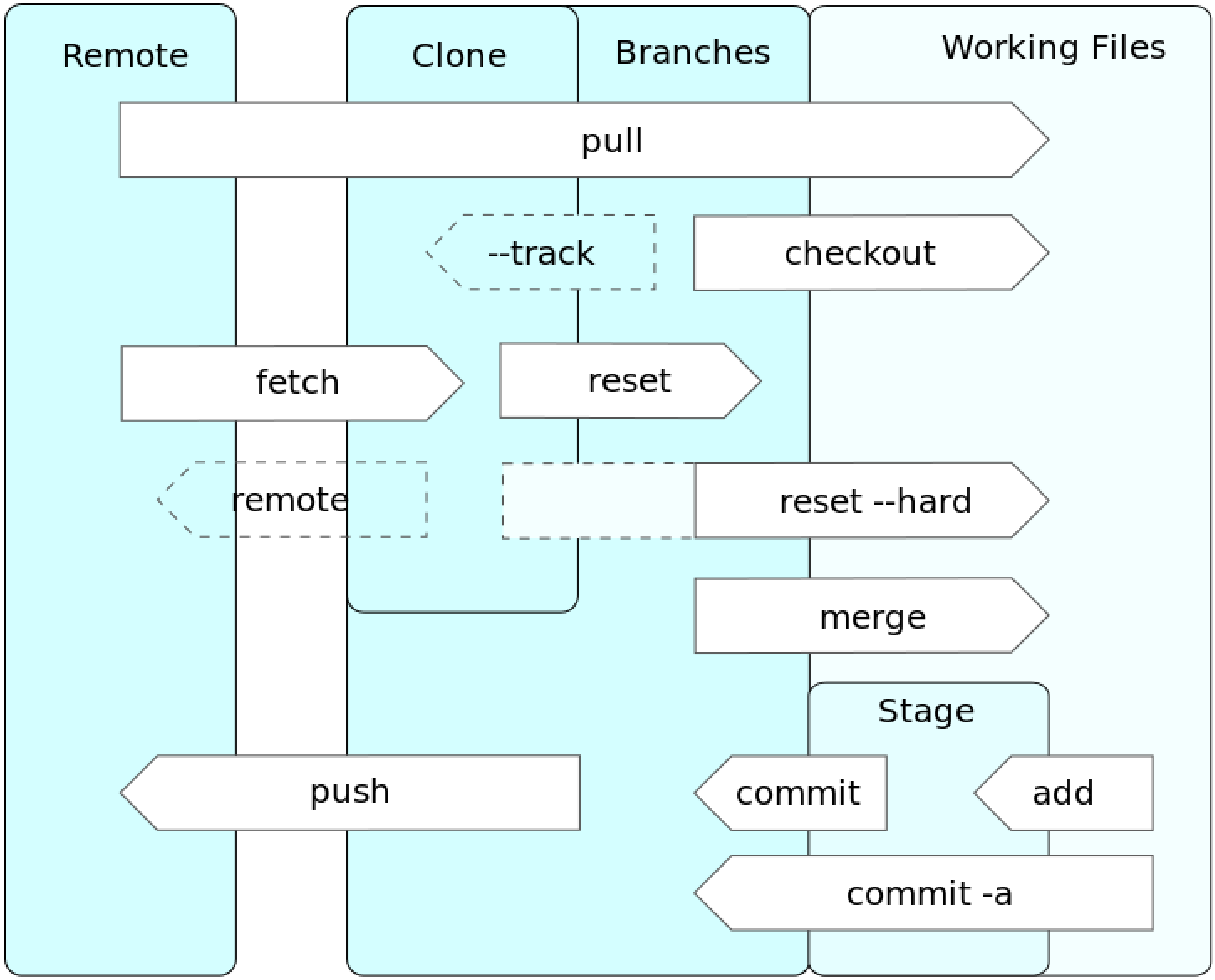
If you want to get a copy of an existing Git repository (a project you'd like to contribute to)

- $ git clone [url]
- $ git clone https://github.com/AnaPalacio/Slides

  (Not sure if you need a github account to clone, open one??)
- $ git clone https://github.com/AnaPalacio/Slides /home/user/MyCopyOfTheSlides

  This creates a directory named "MyCopyOfTheSlides", initializes a .git directory inside it, **pulls** down all the data for that repository, and **checks out** a working copy of the latest version

Git + GitHub


Remote
Clone
Branches
Working Files
pull
--track
checkout
fetch
reset
remote
reset --hard
merge
Stage
push
commit
add
commit -a

More commands to play...

- Project history, changes...
  - $ git push
  - $ git push origin master
  - $ git push origin branch_name
  - $ git pull

- Origin: original remote repository
- Branches: Used to develop features isolated from each other. You can later merge them (git merge)

Everything I have told you is here:

https://git-scm.com/book/en/v2