Workshop: Introduction to Git and GitHub

Part 1: Getting started

Philippe Joly

Freie Universität Berlin

March 9, 2021

Reference

- This workshop draws extensively on Scott Chacon and Ben Straub (2021), *ProGit*, Version 2.1.295, 2021-02-26.
- Like the book, this workshop carries the CC BY-NC-SA 3.0 license.

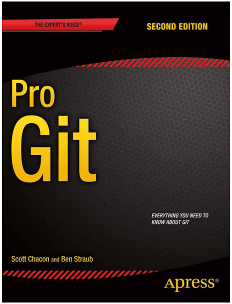
THE EXPERT'S VOICE®
SECOND EDITION
Pro Git
EVERYTHING YOU NEED TO KNOW ABOUT GIT
Scott Chacon and Ben Straub
Apress®

Figure 1

Introduction

Objectives

- Introduction to Git and GitHub
  - Git: a free and open source **distributed version control system** used to track changes in files
  - GitHub: the largest host for Git repositories

Figure 2

Figure 3

What you should be able to do after this workshop

- Record the version history of a project locally
- Synchronize your version history with an online repository
- Work on parallel lines of development using branches
- Collaborate in a project using branches
- Coordinate and discuss the integration of multiple lines of development on GitHub

Why learn Git and GitHub?

Git...

- is safe.
- is fast.
- works offline.

With Git you can...

- revert to a previous state.
- compare changes over time.
- see who introduced what, when, and why.

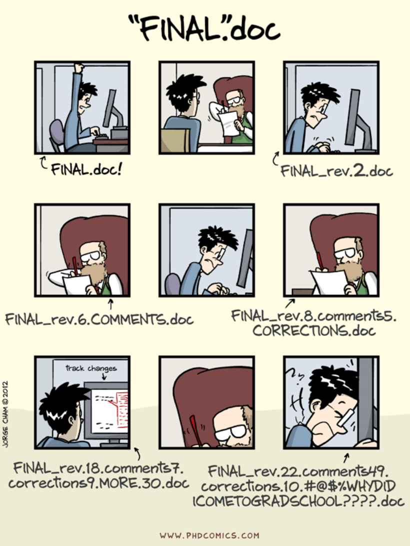
"FINAL".doc
FINAL.doc!
FINAL_rev.2.doc
FINAL_rev.6.COMMENTS.doc
FINAL_rev.8.comments5.CORRECTIONS.doc
track changes
FINAL_rev.18.comments7.corrections9.MORE.30.doc
FINAL_rev.22.comments49.corrections.10.#@$%WHYDID ICOMETOGRADSCHOOL????.doc
JORGE CHAM © 2012
WWW.PHDCOMICS.COM

Figure 4: *Source*: "Piled Higher and Deeper" by Jorge Cham www.phdcomics.com

Why using Git as a social scientist?

- Three pillars of open reproducible research
  1. Version control (Git)
  2. Open source data analysis software (R/Python)
  3. Markup languages (LaTeX, Markdown, HTML)
- Safely save manuscripts (e.g. your thesis)
- Safely save scripts
- Collaborate with others in a more stuctured way
- Contribute to open source projects
- Make your work more visible

A brief history of Git

- Linux Kernel: the most important open-source project in history
- 1991-2002: Maintenance passed around as patches and archived files
- 2002-2005: BitKeeper
- 2005: the Linux development community creates Git

Figure 5: Linus Torvalds *Source*: Krd/Von Sprat. License: CC BY-SA 4.0. https://commons.wikimedia.org

What is version control?

Local version control

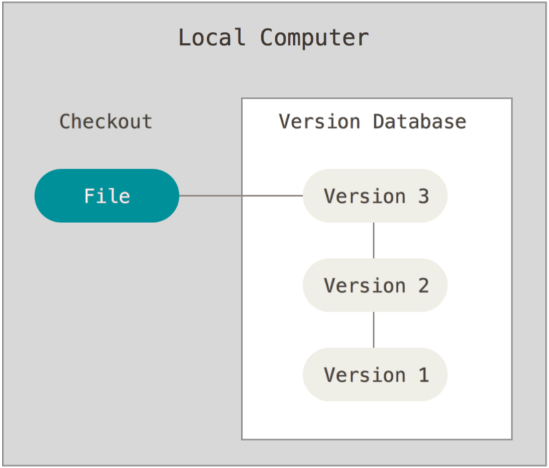

Figure 6: Local version control *Source*: Chacon & Straub (2021), Figure 1.

Centralized version control

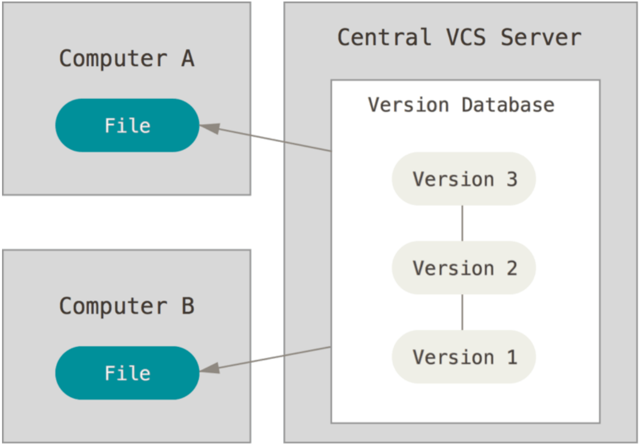

Figure 7: Centralized version control system (CVCS) *Source*: Chacon & Straub (2021), Figure 2.

Distributed version control

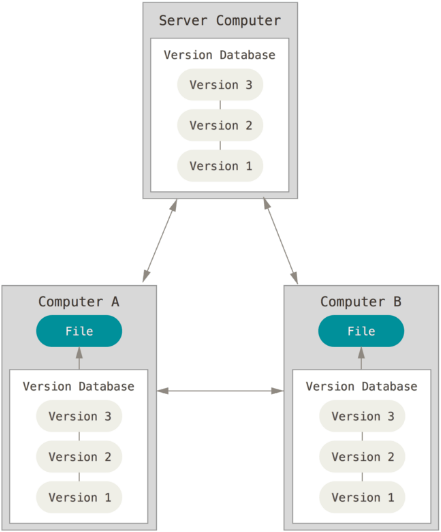

Figure 8: Distributed version control system (DVCS) *Source*: Chacon & Straub (2021), Figure 3.

What is Git?

Snapshots

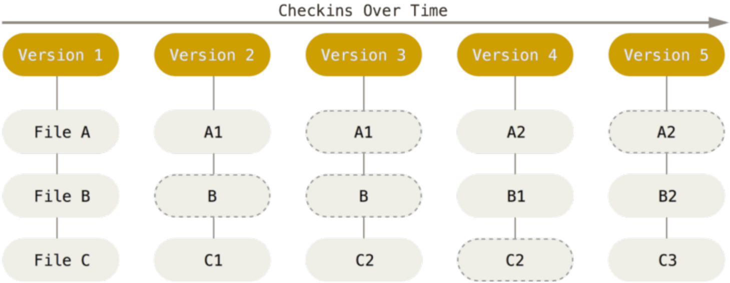

Figure 9: Git stores the version history of a project as a stream of snapshots over time. *Source*: Chacon & Straub (2021), Figure 5.

The three states

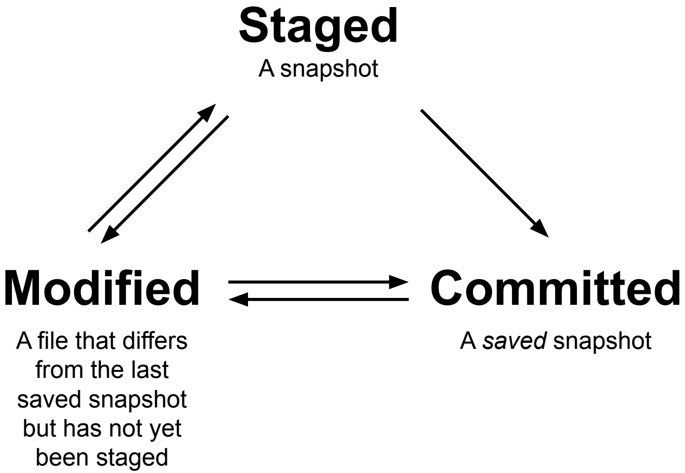
Staged
A snapshot
Modified
A file that differs from the last saved snapshot but has not yet been staged
Committed
A *saved* snapshot

Figure 10: In a git repository, a file can reside in three states: modified, staged, and committed

The Git workflow

The basic Git workflow goes something like this:

1. You **modify** files in your working directory.
2. You selectively **stage** just those changes you want to be part of your next commit.
3. You do a **commit**, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.

Adapted from Chacon & Straub (2021), Section 1.3.

Git Setup

Why using Git from the command line?

- The only way to really learn the mechanics of Git.
- Access to **all** Git functions.
- Easier to get help

Your identity

Set your global user name and email address

```
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```

Note: Your email address is an integral part of your version history in Git. If you make your repository available, everyone will see your email address. Choose wisely!

Other settings

By default, Git uses **Vim** as editor for certain operations. If you have installed **atom**, I would recommend using it as your default editor instead.

Change your default editor

```
$ git config --global core.editor "atom --wait"
```

The default branch name on Git is `master`. I would recommend changing the default name to `main` (only possible from `git --version` 2.28 onwards).

Change your default branch name

```
$ git config --global init.defaultBranch main
```

Viewing all your settings

```
$ git config --list
```

```
user.email=philippe.joly@posteo.net
user.name=Philippe Joly
init.defaultbranch=main
core.editor=atom --wait
```

Getting help

Three equivalent ways to get the comprehensive manual page

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

More concise “help” output

```
$ git <verb> -h
```

What we have learned in this part of the workshop

- The concepts behind a distributed version control system
- The three states a file can reside in inside Git
- The basic Git workflow
- Setting up your identity
- Other settings
- Getting help

Exercises

1. Find the version of Git you are using with `git --version`.
2. Set your global `user.name`.
3. Set your global `user.email`.
4. If you want, change your default `core.editor`.
5. Set your default branch name to `main` (Git version $\geqslant$ 2.28).
6. Examine all your settings.
7. Examine the concise help output for the `git config` command.