Basics of Version Control

Part II Computational Physics
Lent 2019
Matthew Evans

https://github.com/ml-evs/part2-computing-git-tutorial / https://bit.ly/2Gmh8NW

What is version control?

"A system that tracks and manages changes to a set of files (e.g. source code)."

- ➔ **Reversibility**
  - ◆ Ability to revert to previous state when (not if) things go wrong
- ➔ **History**
  - ◆ Ability to record explanations and intentions of changes
- ➔ **Concurrency**
  - ◆ Ability to work with others, rather than against them

https://www.gnu.org/software/emacs/manual/html_node/emacs/Introduction-to-VC.html

https://github.com/ml-evs/part2-computing-git-tutorial / https://bit.ly/2Gmh8NW

Why should I care?

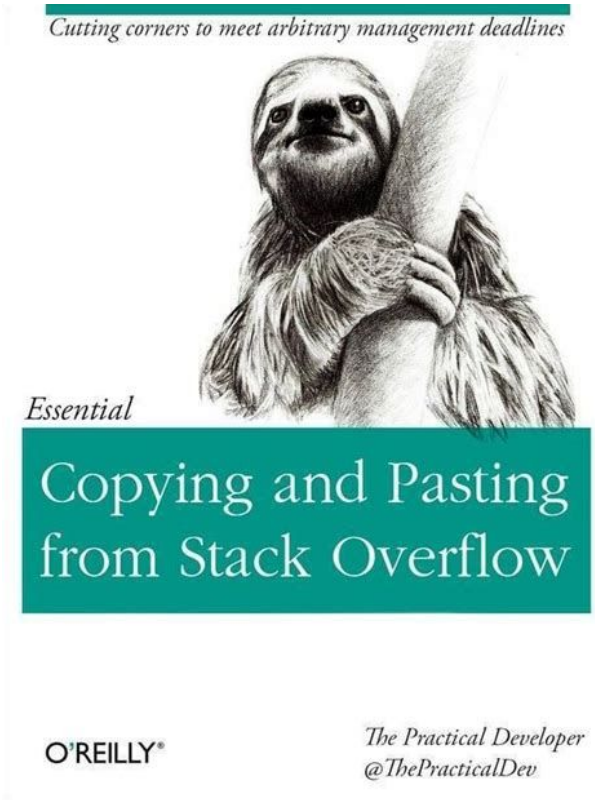
- Avoids **horror scenario** of
  `exercise1.py`,
  `exercise1_broken.py`,
  `exercise1_maybefixed.py`,
  `exercise1_nostillbroken.py`,
  `exercise1_final.py`,
  `exercise1_finalfinal.py`,
  `exercise1_submitted.py`,
  `exercise1_resubmitted.py`…
- **Revert** code to its state at any other time, i.e. when it was working.
- Much more of a **reliable workflow** for the messy, nonlinear software development process than e.g. Dropbox, Google Drive or even *Facebook* version control.
- Enforces personal **discipline** and can drastically affect the way you code.
- Absolutely vital when working with **multiple interdependent files**,
  - e.g. consider changing a low-level function signature

https://github.com/ml-evs/part2-computing-git-tutorial / https://bit.ly/2Gmh8NW

Why should I care?

An important meta-skill when programming:

- ➔ **Version control**
- ➔ **Writing good tests**
- ➔ **Detecting “bad code smell”**
- ➔ **Knowing what to Google**
  - And which bits to copy from Stack Overflow
  - **Knowing what you don’t know**



Source: @ThePracticalDev

https://github.com/ml-evs/part2-computing-git-tutorial / https://bit.ly/2Gmh8NW

Why Git?

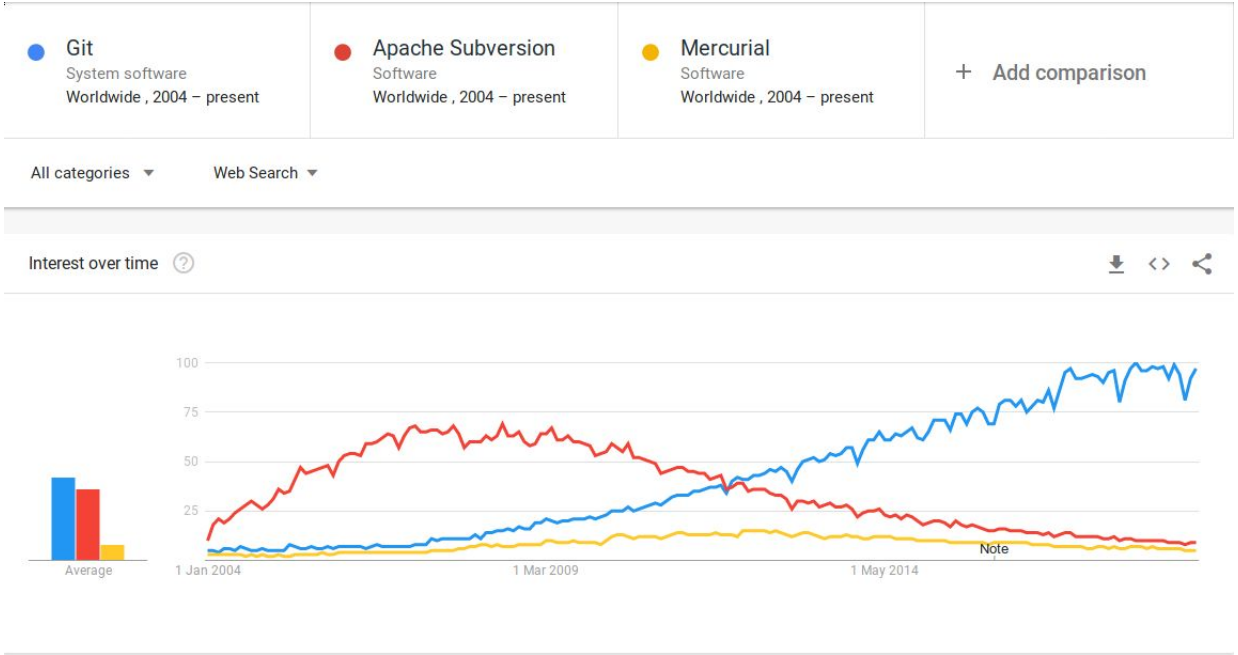
Git was spawned by **hate**

- ➔ Fast and scalable in project size
  - ◆ both lines of code and number of developers
- ➔ Distributed
- ➔ Secure
- ➔ “Simple” to learn
- ➔ Easily the most popular, as of 2019

Linus Torvalds
(image from Wikipedia)

https://github.com/ml-evs/part2-computing-git-tutorial / https://bit.ly/2Gmh8NW

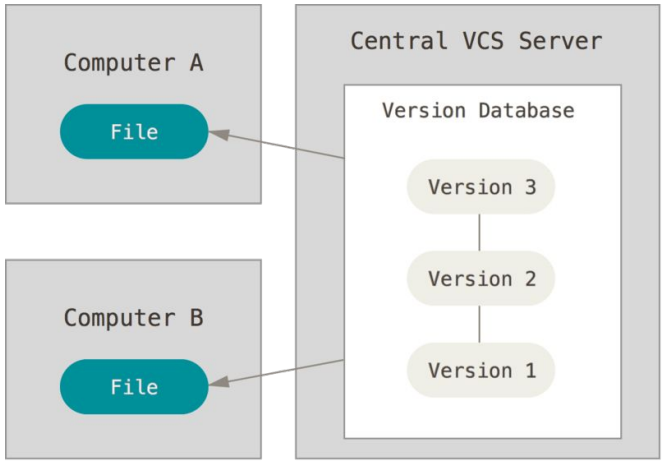
Why Git?



Data from Google trends: https://bit.ly/2DBqUZ5

https://github.com/ml-evs/part2-computing-git-tutorial / https://bit.ly/2Gmh8NW

Anatomy of Git: Distributed Version Control



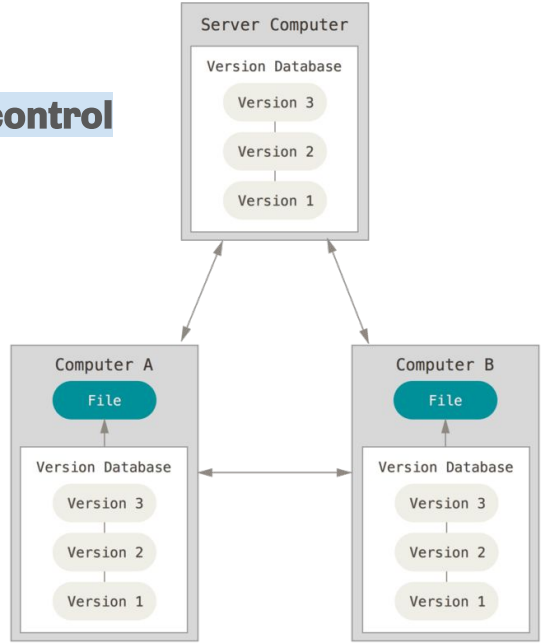
tired: **centralised version control**
e.g. Subversion

Images from Chapter 1.1 of Pro Git
https://git-scm.com/book/en/v2

wired: **distributed version control**
e.g. Git, Mercurial

Advantages:

- ➔ Redundancy: every local repository has all the history
- ➔ Don't need to be online
- ➔ More flexible hierarchy



https://github.com/ml-evs/part2-computing-git-tutorial / https://bit.ly/2Gmh8NW

Anatomy of Git: Repositories

- Any top-level directory that is version controlled is called a **repository**.
- The VC magic happens inside the `.git` folder.
- Git **blobs** all objects, computes an SHA-1 hash and compresses
  - See Chapter 10 of Pro Git

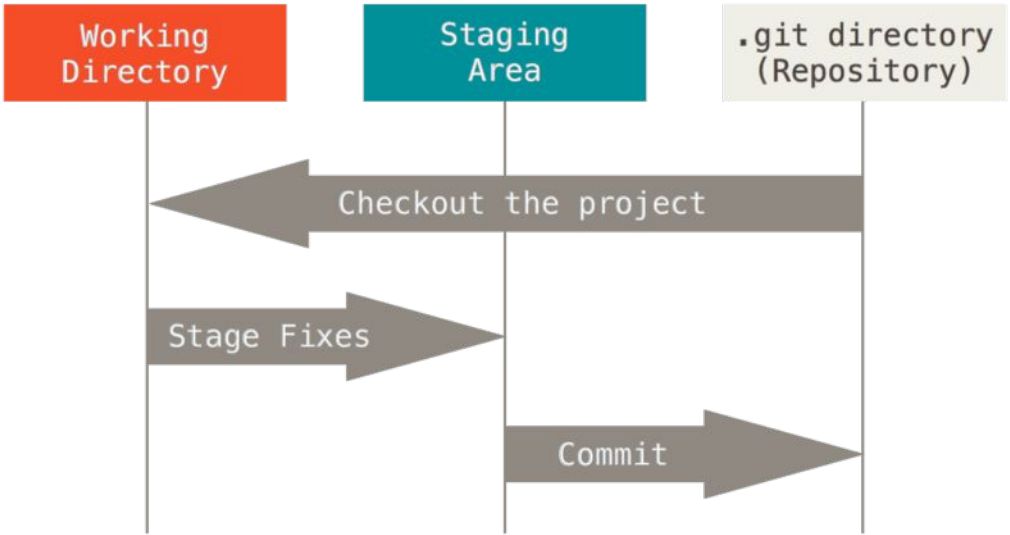


Image from Chapter 1.3 of Pro Git
https://git-scm.com/book/en/v2

https://github.com/ml-evs/part2-computing-git-tutorial / https://bit.ly/2Gmh8NW

Anatomy of Git: Commits

- ➔ Changes to files are tracked in the repository via **commits**.
- ➔ A set of **thematically linked changes** given a descriptive message.
- ➔ Each commit defines a **whole snapshot** of the repository.

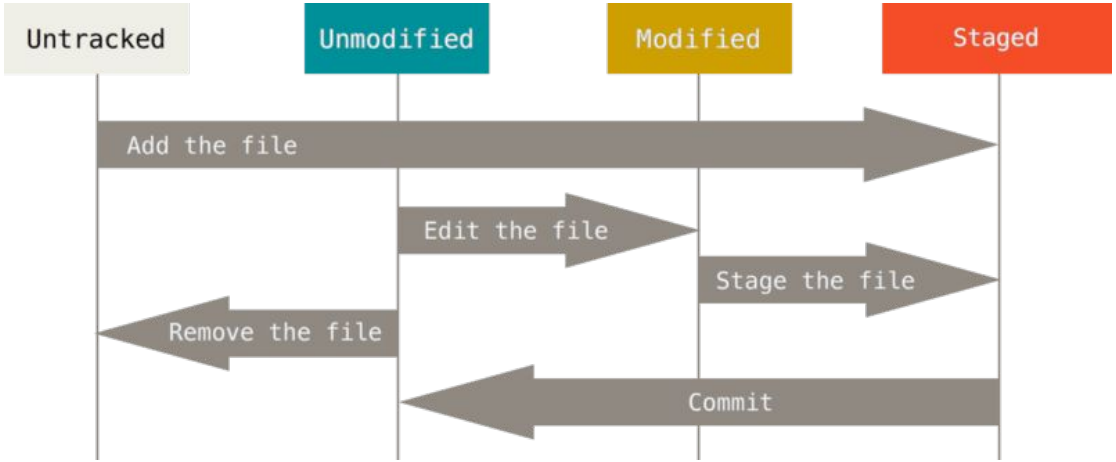


Image from Chapter 2.2 of Pro Git
https://git-scm.com/book/en/v2

https://github.com/ml-evs/part2-computing-git-tutorial / https://bit.ly/2Gmh8NW

Anatomy of Git: Commits

- ➔ Commits **stack** (in the computing sense) on top of each other.
- ➔ In this sense, commits cannot be undone, but can be **reverted to**.
- ➔ The granularity of **your commits** is up to personal preference
  - ◆ or is agreed upon for a particular project

https://xkcd.com/1296/

https://github.com/ml-evs/part2-computing-git-tutorial / https://bit.ly/2Gmh8NW

Anatomy of Git: User Interface

- Cross-platform command-line program `git` with several subcommands, each with their own options
  - e.g. `git commit --help` or `git clone --help`.
- Sheer number of commands gives it a reputation for being hard to use, but can get away with only using a small subset regularly:
  - `add/commit/push/pull`.
- GUIs also exist, such as GitKraken. A more complete list can be found at https://git-scm.com/downloads/guis/
- Our examples will use the command line, which should be installed on the MCS already.

https://github.com/ml-evs/part2-computing-git-tutorial / https://bit.ly/2Gmh8NW

Online version control providers

- ➔ Allow you to add a mirror of your git repository on a reliable server and provide a place to **distribute your code** (see `git clone`).
- ➔ Big three:
  - GitHub https://github.com
  - BitBucket https://bitbucket.org
  - GitLab https://gitlab.com
- ➔ All offer **free plans** for students/academics/open source, your choice which to use (see “Useful Links” in the notes)
- ➔ Now exist software journals let you submit your code repository for review, e.g. Journal of Open Source Software: http://joss.theoj.org

https://github.com/ml-evs/part2-computing-git-tutorial / https://bit.ly/2Gmh8NW

git <3 open source

- **Scientific software is powered by open source**
- The majority of open source software projects use Git...
  - Often open source software is developed by many remote collaborators (see e.g. Linux https://github.com/torvalds/linux)
  - but companies also host their stuff (e.g. Google-developed programming language Go https://github.com/golang/go).
  - Have a look for the source of NumPy or even CPython itself
- **Anyone can contribute!**
  - Many projects have “good first issues” tags
- Most are hosted on GitHub.
  - Brands itself as a “social platform for software”.
  - Recently acquired by Microsoft...

https://github.com/ml-evs/part2-computing-git-tutorial / https://bit.ly/2Gmh8NW

$ ./live_demo

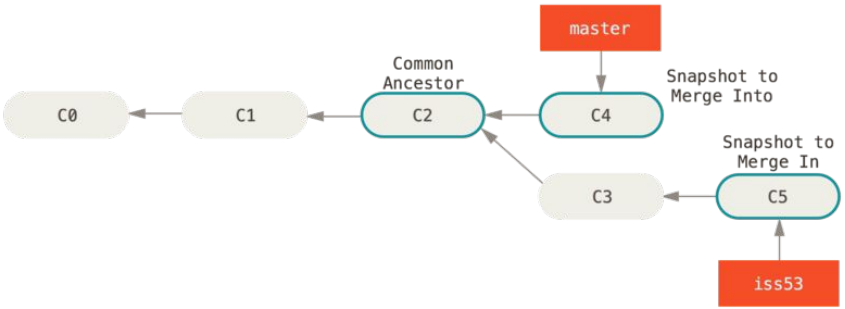
Follows Example 1.2
“Remote version control”
in the notes

https://github.com/ml-evs/part2-computing-git-tutorial https://bit.ly/2Gmh8NW

Advanced Usage

Branching & Merging

- ➔ Multi-developer projects always use branches, but they can be useful for solo devs too
- ➔ See Chapter 3.1 of Pro Git (source of image below) for more



Testing and Continuous Integration (CI)

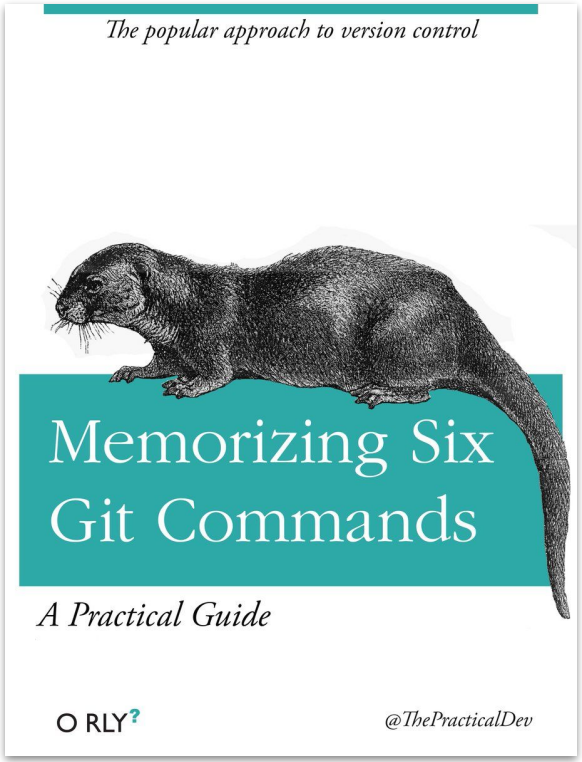
- ➔ Commonplace to run a test suite for every “push”; can be automated through web services such as Travis, Jenkins or Bitbucket Pipelines.
  - ◆ Very useful when “deploying” a product.
- ➔ Git also has its own useful local testing feature: `git bisect`
  - ◆ Binary search of commits to find which changes “broke the build”.

https://github.com/ml-evs/part2-computing-git-tutorial / https://bit.ly/2Gmh8NW

Conclusions

- ➔ Version control is a useful tool for protecting yourself against your own stupidity and that of others
- ➔ Git is the *de facto* standard for version control throughout industry and academia
- ➔ Have a go at Example 1 from the GitHub repo for yourself, and if you're sold you can try putting your exercise solutions under VCS.

Thank you for listening, any questions?



Source: @ThePracticalDev

https://github.com/ml-evs/part2-computing-git-tutorial / https://bit.ly/2Gmh8NW