



Basics of Version Control

Matthew Evans

<https://github.com/ml-evs/git-tutorial>

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/git-tutorial>

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

Why should I care?

An important meta-skill when programming:

→ **Version control**

→ **Writing good tests**

→ **Detecting “bad code smell”**

- ◆ Just because it works, doesn't mean it's good
- ◆ e.g. code golf or https://en.wikipedia.org/wiki/Esoteric_programming_language

→ **Knowing what to Google**

- ◆ And which bits to copy from Stack Overflow
- ◆ **Knowing what you don't know**

<https://github.com/ml-evs/git-tutorial>

Cutting corners to meet arbitrary management deadlines



Essential

Copying and Pasting
from Stack Overflow

O'REILLY*

The Practical Developer
@ThePracticalDev

Source: @ThePracticalDev

Why Git?

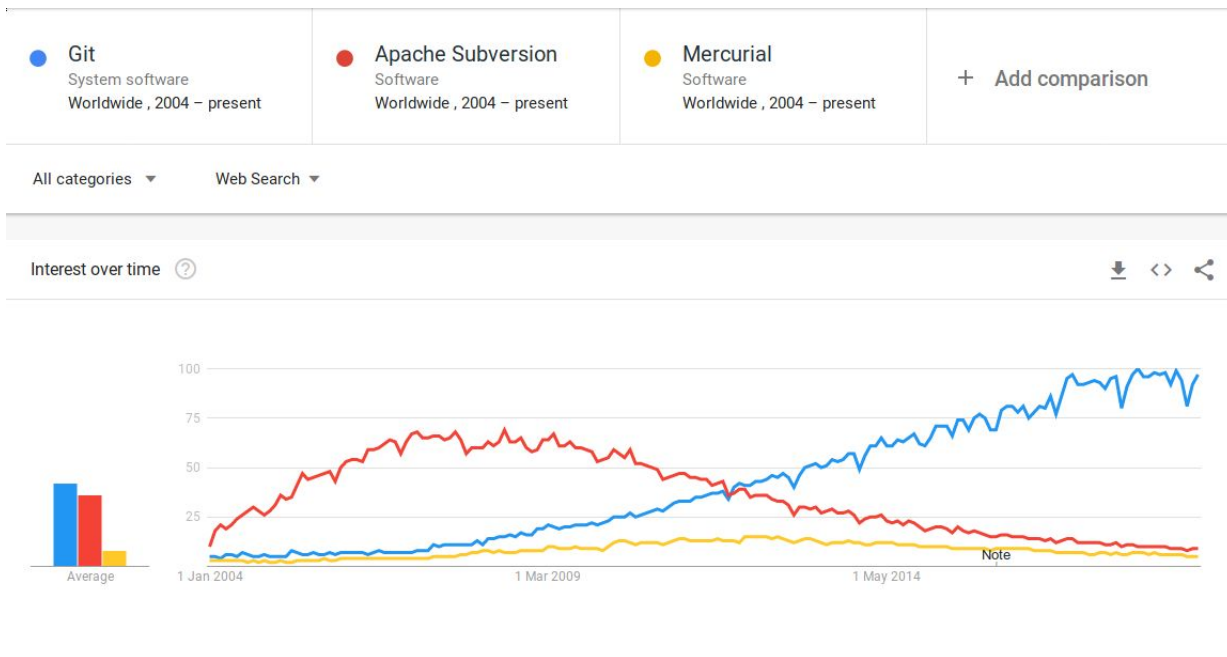
Git was spawned by **hate**

- 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)

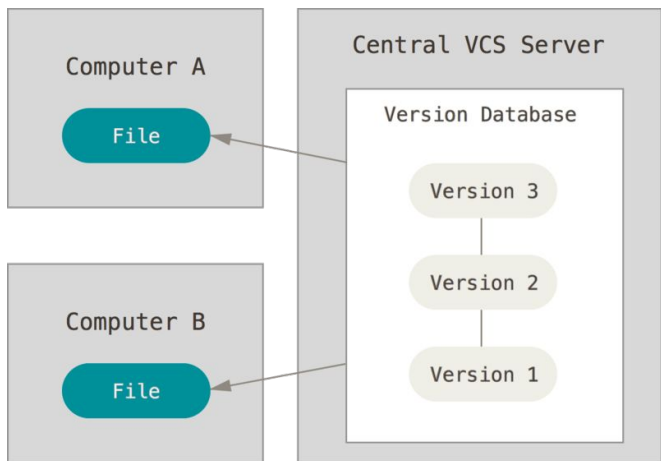
Why Git?



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

<https://github.com/ml-evs/git-tutorial>

Anatomy of Git: Distributed Version Control



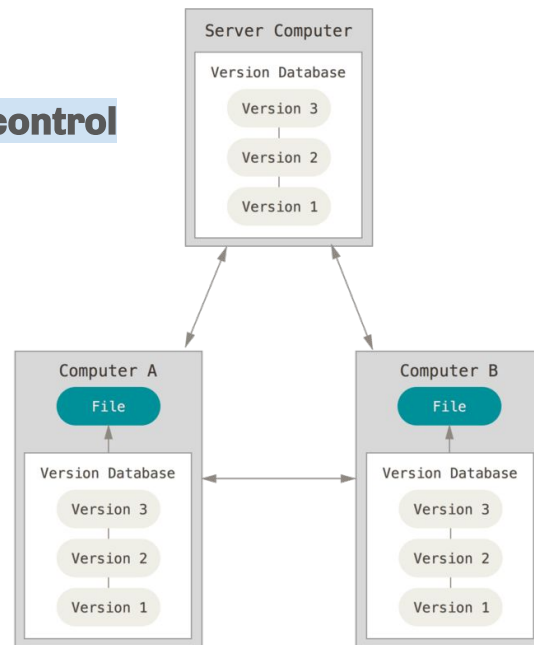
tired: **centralised version control**
e.g. Subversion

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



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

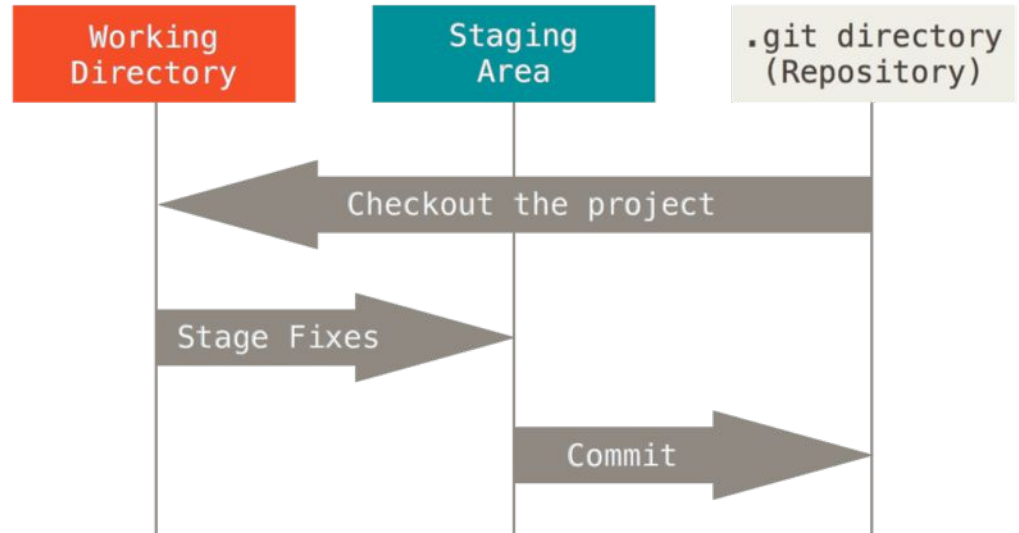


Image from Chapter 1.3 of Pro Git
<https://git-scm.com/book/en/v2>

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.

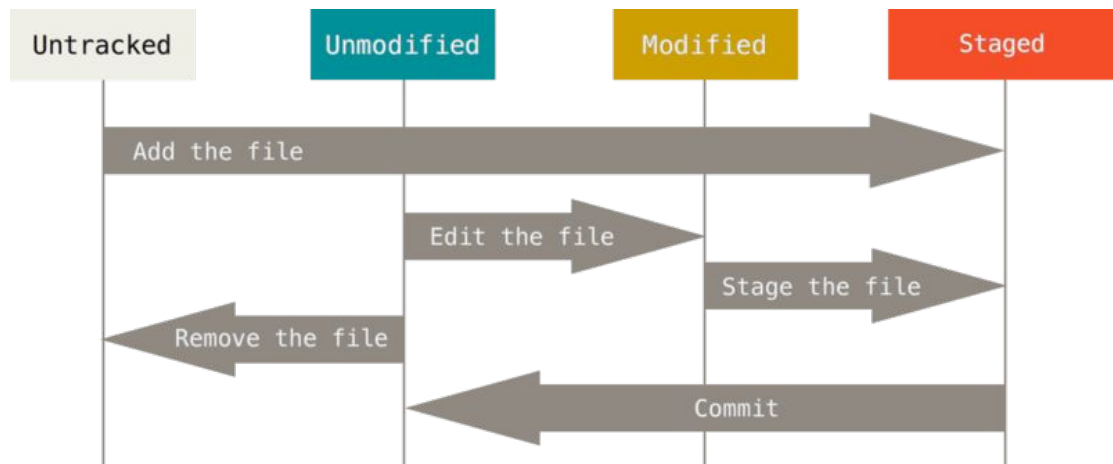


Image from Chapter 2.2 of Pro Git
<https://git-scm.com/book/en/v2>

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

	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
○	HAAAAAAAAANDS	2 HOURS AGO

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

<https://xkcd.com/1296/>

<https://github.com/ml-evs/git-tutorial>

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.

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>

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/git-tutorial>

Collaboration with Git: Branching & Merging

- Multi-developer projects always use **branches**, but they can be useful for solo devs too
- Allows developers to work on separate features without fear of **conflicting code**
- There are lots of possible branching strategies:
 - ◆ typically have a **master** branch that contains agreed upon changes
 - ◆ New features are “merged” in, either manually, or via **pull requests**

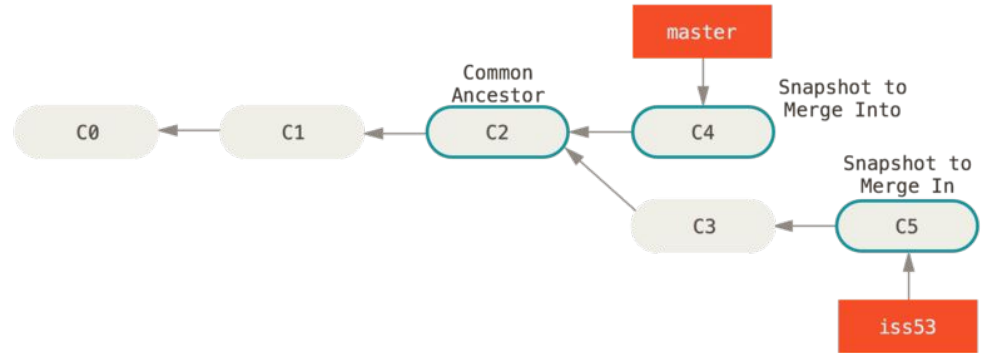


Image from Chapter 3.1 of Pro Git

Collaboration with Git: Issues & Pull Requests

- **Pull Requests** (also known as merge requests) and **Issues** are extra features implemented by online version control providers.
- **Issues** allow users to submit bug reports, ask for help, or request features.
- **Pull Requests** allow developers to review each others changes before merging into the main branch, and can become discussion points for new features.
 - ◆ PRs form an important part of open source etiquette

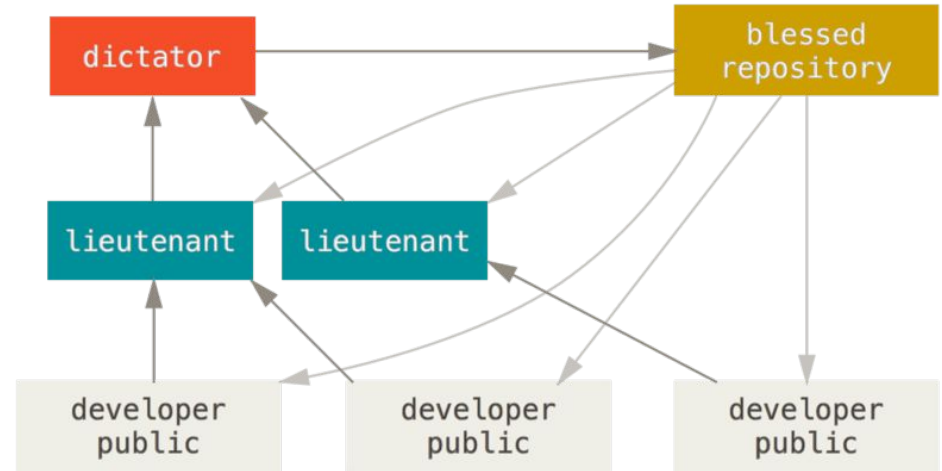


Image from Chapter 5.1 of Pro Git

Collaboration with Git: Continuous Integration (CI)

- Often development is **test-driven**:
 - ◆ When the input and desired output of a function is known ahead of time, write some test cases first!
 - ◆ Crucial when working with other people to avoid **unintended consequences** of new features.
- Commonplace to run a test suite for every “push”; can be automated through web services; this is called **Continuous Integration**.
 - ◆ Many services provided free for open source/academic software (e.g. Travis, Bitbucket Pipelines), which are closely integrated with e.g. GitHub, GitLab or BitBucket.
 - ◆ These often allow you to test in environments not available to you, e.g. on macOS/Windows/Linux all at once.

Git also has its own useful local testing feature: **git bisect**

Performs a binary search of commits to find which changes “broke the build”.

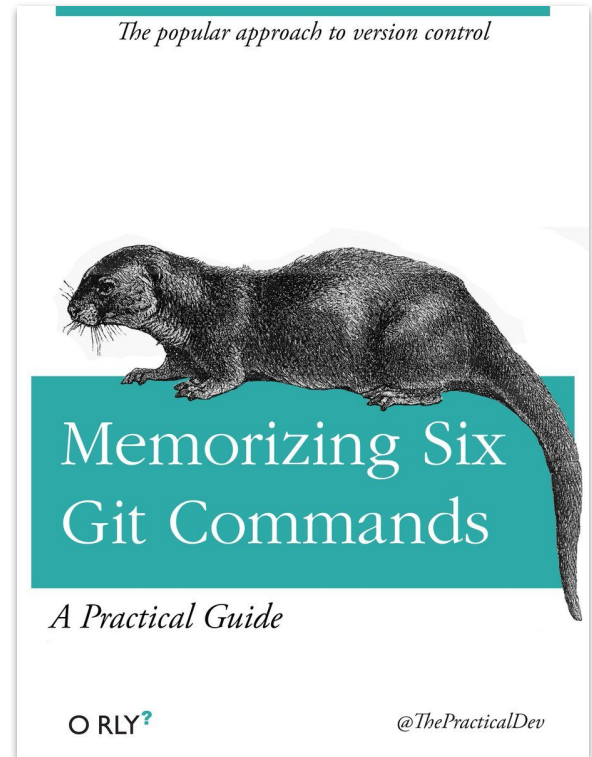

```
$ ./live_demo
```

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?

<https://github.com/ml-evs/git-tutorial>



Source: @ThePracticalDev