

Version Control With Git

Mark Andrews

Psychology Department, Nottingham Trent University

What is Git?

- ▶ Version control software keeps track of all the versions or revisions to a set of files in an efficient and orderly manner.
- ▶ Git is version control software, initially developed for version control of the Linux operating system kernel.
- ▶ It is now extremely widely used for almost all kinds of software development projects.
- ▶ Git is characterized by the following features or goals:
 - ▶ Nonlinear development.
 - ▶ Distributed and decentralized.
 - ▶ Speed and efficiency.
 - ▶ Open source.

How Git works internally (very briefly)

- ▶ Git works on whole “projects”, which are simply directories, not individual files.
- ▶ Each project is completely self-contained, and completely independent of every other.
- ▶ It keeps a “snapshot” of the state of the project as it develops. These are *commits*.
- ▶ Git copies every version of every file in every commit. Calculates its sha-1 hash. Compresses it. Names it by its hash. These are known as *blobs*.
- ▶ It creates a hash tree (Merkle tree) from the files in each commit (the *tree object*).
- ▶ Each commit stores the tree hash, parent commit, author info, log message, etc, and calculate it sha-1 hash.

Distributed and decentralized

- ▶ The Git project is completely self-contained. All the meta data and history of the project is contained in `.git`.
- ▶ There is no centralized server. No synchronization. No edit locks.
- ▶ Collaboration is via cloning and re-emerging.
- ▶ E.g. a project can be *cloned*, developed independently, and possibly re-merged.
- ▶ Usually developers use a *remote* host with a “bare” repository, clone it, develop locally, *commit* and then *push* back to and *pull* from the remote host. The *push* and *pull* are merges.
- ▶ GitHub is one of the most widely used hosting sites (but there are others, e.g. BitBucket; and running your own git hosting server is simple and inexpensive).

What should we (academics, researchers, scientists) care?

- ▶ Git (and a Git server) provides a means of sharing your *research compendium* and its updates, and this can be done from the beginning of the research and continually throughout its entire development.
- ▶ Sharing can be with the public or just with collaborators
- ▶ But it's not just another "dropbox": It is a decentralized version control system.
 - ▶ It allows open, distributed, collaborative development of research and data analysis code, like the open source development model for software generally.
 - ▶ Even for small teams, it provides a means for efficient and organized development of source code, including (and especially) for writing the source code for our articles, slides, etc (no more of this: http://phdcomics.com/comics/archive_print.php?comiciid=1531).
 - ▶ Even for individuals, it allows for efficient and organized development.
 - ▶ And it is (almost) endlessly fault tolerant. So long as there is one clone somewhere, the entire project and its version history is preserved entirely.

What should we (academics, researchers, scientists) care?

- ▶ The minimal requirements for project development for individuals or small teams.
 - ▶ One *single source of truth*.
 - ▶ All past versions are stored; allowing for indefinite undoing.
 - ▶ Parallel development.
- ▶ In addition, Git provides
 - ▶ A development log
 - ▶ Nonlinear workflows.
 - ▶ Robustness and fault toleration.
 - ▶ Multiple remote server options.
 - ▶ Free and open source and cross platform.
 - ▶ The *de facto* standard for version control.

Git: Tiny tutorial

- ▶ Start by cloning a remote repository:

```
git clone https://github.com/yihui/knitr.git
cd knitr
git log # Read all the commit logs
```

- ▶ Work as normal, i.e. edit files, create new files, delete files.
- ▶ You now *stage* your changes, e.g.

```
git add foo.file.1 foo.file.2 # for edits or new files
git rm foo.file.3 # for removed files
```

- ▶ You then *commit* these:

```
git commit # Editor opens for your log msg
```

Git: Tiny tutorial (2)

- ▶ Pull down any recent changes by others from the remote:

```
git pull  
git log # If new changes, read their logs
```

- ▶ Now, push your own changes to the remote

```
git push # requires permissions
```

- ▶ Undo changes:

```
git reset a381f2f # move back "head"  
git revert a381f2f # applies new change to revert
```


References