

# Git

THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.

COOL. HOW DO WE USE IT?

NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.





**@tabqwerty**

chi wai lau

"git gets easier once you get the basic idea that branches are homeomorphic endofunctors mapping submanifolds of a Hilbert space."

9 Mar via web   Unfavorite   Undo Retweet   Reply

**A MONAD IS JUST A MONOID IN THE  
CATEGORY OF ENDOFUNCTORS**

**WHATS THE PROBLEM?**

# Git?

"I'm an egotistical <censored>, and I name all my projects after myself. First 'Linux', now 'git'."

- Linus Torvalds



# Background

- `git add`
- `git pull`
- `git push`

# Let's start at the very beginning

A very good place to start.

# Let's start at the very beginning

- git init
  - meh
    - description -> GitWeb
    - config -> project-wide config
    - info -> personal excludes file
    - hooks -> scripts
  - not meh
    - objects
    - HEAD
    - refs
    - index -> you can't see this right now, but you soon will

# objects

- objects folder is the database
- meat of the version control
- 4 different types of objects



# objects

- **pack**
  - compressed objects in a single file
- **info**
  - metadata about the object store

# Time for some theory!

1. Object model
2. The 3 States

# Object model: object types

1. **blob** -> particular version of one file
2. **tree** -> directory, contains blobs and trees
3. **commit** -> reference to top-level tree + parent commit
4. **tag** -> meh

Everything is immutable!

# 3 states

1. Working directory -> *modified*
2. Staging area -> *staged*
3. Object database -> *committed*

## 3 states: object database

- All the important stuff
- The stuff inside your objects/ folder
- Contains, well, objects
  - commits, blobs, trees, (tags)

## 3 states: working directory

- Current branch
- Stuff gets pulled out (uncompressed) out of the objects/ folder
- ls

## 3 states: index

- Just a file
- What will go into your next commit
- Intermediate step
- `git add <filename>`
- File now staged! Let's see what it looks like...

# Blob

1. Check out blob in objects folder
2. A blob is a snapshot
3. Hash of contents of version of file
  - a. SHA1 hash of contents -> zlib compression after hash.
  - b. Can create a new file with same contents and add it, won't create new object
4. `git hash-object <filename>`
5. We can make changes to our file!
6. New blobs. Blobs everywhere.



# git commit

- Takes index, commits file into object database
- Adds commit + tree
- Let's commit!
- Index changed

# Let's play around

- Create new folder and file
- Check new blob added
- Commit. Check new tree and new commit, with old reference to original file.

# refs

- Just pointers to commits
- Check out the refs folder
- Let's create a new branch, and check out what happens when we make a commit

# Resources

- <https://pcottle.github.io/learnGitBranching/>
- <https://git-scm.com/docs>