Readings on Git

COMP 150 - Applied Functional Programming

October 9, 2012

The readings below are intended as fuel for the discussion on Monday, October 15. Please also bring the filesystem semantics paper.

Readings on how git works underneath

Required reading:

- Four students suggested the ever-popular *Git from the Bottom Up* from http://ftp.newartisans.com/pub/git.from.bottom.up.pdf. This happens to be one of my own favorites.

  Key sections:

  - *Introduction*, pages 3 and 4
  - *Repository*, pages 5–19

  The second on the index is less critical, but in order to participate fully in the discussion, you will want to digest at least the first two pages (20–21). At the moment, I don't see any value for us in pages 22–

  (If you find the blob/tree/commit discussion hard to follow, you might prefer the treatment in the Git book at http://git-scm.com/book/en/Git-Internals-Git-Objects. But this is optional.)

- "Git for the confused", the section on Merging only. http://www.gelato.unsw.edu.au/archives/git/0512/13748.html

Recommended by multiple students:

- *Understanding Git Conceptually* at http://www.sbf5.com/~cduan/technical/git/ is neither fish nor fowl: it is mostly intended for people who are going to *use* Git, but there is a little information about what is happening underneath.

Not recommended:

- Despite the title, I find little of value in "Git for Computer Scientists." But it has pictures.

Understanding rebase

Required reading:

- The rebasing section in *Understanding Git Conceptually*, http://www.sbf5.com/~cduan/technical/git/git-5.shtml, contains this intriguing description of rebase:

  > When [rebase is] run, Git performs the following steps:
  >
  > 1. Identifies [a sequence of commits].
  > 2. Determines what changed for each of those commits, and puts those changes aside.
  > 3. [Moves] the current head.
  > 4. For each of the changes set aside, replays that change onto the current head and creates a new commit.

  This is as close as I've been able to find of a real description of how rebase works underneath

- In the Git book section on branching and rebasing http://git-scm.com/book/ch3-6.html, after Figure 3-30 it says

  > Now, the snapshot pointed to by C3' is exactly the same as the one that was pointed to by C5 in the merge example.

  This text suggests that rebase and merge use the same algorithm.

Recommended reading (short):

- "Git is inconsistent" http://r6.ca/blog/20110416T204742Z.html says that Git's merge algorithm violates natural algebraic laws and shows an example. The same example is presented in a slightly more readable way at http://rjsteinert.com/content/git-breaking-%E2%80%9Cmerge-associativity-law%E2%80%9D

Recommended only if you didn't follow the discussion in *Git from the Bottom Up*:

- A condescending explanation of rebase complete with a the "golden" rule "don't rebase a published branch." http://blog.experimentalworks.net/2009/03/merge-vs-rebase-a-deep-dive-into-the-mysteries-of-revision-control/
- Another article on rebase vs merge at http://www.jarrodspillers.com/2009/08/19/git-merge-vs-git-rebase-avoiding-rebase-hell/ contains the intriguing claim

  > After it unwinds all your hard work it then stuffs the new commits from the remote branch onto the stack and then plays your changes back on top of them

- If you rebase a published branch, you might not be totally hosed; the man page for `git rebase` has a section on recovering from upstream rebase.

Git in pictures

- http://marklodato.github.com/visual-git-guide/index-en.html

Readings on how git should be used (completely optional)

Rebase in particular

- https://help.github.com/articles/interactive-rebase is a short form on how to use interactive rebase:
- http://darwinweb.net/articles/the-case-for-git-rebase is a thoughtful post about why, when, and how to use rebase instead of merge. I especially like the comment that says "rebase onto feature branches; merge onto the main branch."

Long-form tutorials

If you want to use git and haven't, try one of these long tutorials

- *Git Magic* tutorial by Ben Lynn (NR recommended)

  http://www-cs-students.stanford.edu/~blynn/gitmagic/
- *Git Immersion* tutorial in lab/walkthrough style (JC recommended)

  http://gitimmersion.com/index.html

War stories and polemics (totally optional)

Polemic against rebase:

- http://paul.stadig.name/2010/12/thou-shalt-not-lie-git-rebase-ammend.html