latex input: mmd-article-header Title: Git Notes Author: Ethan C. Petuchowski Base Header Level: 1 latex mode: memoir Keywords: Git, Version Control, Command Line, Terminal, Syntax CSS: http://fletcherpenney.net/css/document.css xhtml header: copyright: 2014 Ethan C. Petuchowski latex input: mmd-natbib-plain latex input: mmd-article-begin-doc latex footer: mmd-memoir-footer

Refs

- O'Reilly's *Git Pocket Guide,* by Richard E. Silverman
- *Ry's Git Tutorial,* by Ryan Hodson.

Useful

These are each tutorials that I was able to use to really *GET IT*

- Difference between HEAD^ and HEAD~
- Traverse forward and backward in history
- `git reset --soft vs (default or --mixed) svs --hard`

Intermediate commands

8/10/15

```
# set head at commit number $1
$ gitj() { git checkout `git rev-list master | tail -n$1 | head -n1`

# now we have
$ gitj 3
Checked out the 3rd commit on master

# show sha, author, date, and message of every commit until HEAD
$ git log
$ git rev-list --pretty HEAD     # same as `git log`

## show commit graph (in color!)
$ git rev-list --graph HEAD             # just hashes
$ git rev-list --graph --pretty HEAD    # full info

# list commit objects in reverse [or forwards] chronological order
$ git rev-list [--reverse]

# show diff between current snapshot `k` snapshots ago
$ git diff HEAD~k..HEAD
```

Basic Tutorial

11/12/14

```
# git fetch && git merge
$ git pull

# read `man` file for git command
$ git --help rm

# Show current branch
$ git branch
* master

# Show all local and remote branches
$ git branch -a
* master
  remotes/origin/HEAD -> origin/master
  remotes/origin/master
  remotes/origin/release-1.1.3

# Switch working directory's state to a particular tag, branch, comm
$ git checkout ( tag_name | branch_name | commit_sha )
You are in 'detached HEAD' state...

# Show `diff`erence between your working tree and the *index* (stag
$ git diff

# Show `diff`erence between your *index* (staging area)
# and the most recent ("current") commit
$ git diff --staged

# add versions of files from the working tree to the index
# (no files means all files)
$ git add [file1 [file2 ...]]

# Make the *index* (staging area) *become* the newest committed "sna
# Physically, this just adds a pointer from it to the previous "pare
$ git commit

# Merge branch `refactor` into `master`.
#   1. applies the diffs
#   2. asks you to resolve conflicts
#   3. commits the result
$ git checkout master   # switch to master branch
$ git merge refactor

# Add only *some* of the changes you've made
#    Starts an interactive loop that lets you select
#    which "hunks" of (all) changes you want to index.
# Use "`?`" during the interactive session to see the commands
$ git add -p

# Remove file from your index/"staging" **and from the working tree
#       WARNING: removal from "working tree" means this
#                 file disappears from your file-system!
$ git rm [filename]

# Reset (empty) the staging area
#   Nonobviously: Your changes will still be there on your local fi
$ git reset
```