Git

Source code management the UNIX way

Jesper Louis Andersen Alexander Færøy

October 24, 2009

Table of Contents

The History of Git

- Initially designed by Linus Torvalds.
- Git became self-hosted on April 7, 2005.
- The Linux Kernel project moved to Git 9 days later.
- Today, thousands of projects.

What is?

- Storage system (Persistence).
- Revision control on top.
- UNIX philosophy in the tool-set.
- Key advantage: Flexibility, speed.
- Key disadvantage: Relatively steep learning curve.

What can it do?

- Projects *evolve*.
- Git, like SVN, Darcs, CVS, Hg and Bazaar, manages project. data, tracks history, facilitates collaboration etc.

Concept: Persistence

- Databases.
- Functional languages.
- Accounting/Finance.
- *Never* overwrite old data.
- Git breaks the rules in a few places.

Term: Blob

- A Blob stores *content*, ie what is in a file.
- Compressed, for saving storage and disk reads.
- Identified by an SHA1 checksum.
- Note that SHA1 is 2nd preimage resistant (still).

Term: Tree

- A Tree contains a list of references paired with meta-data.
- References points to either blobs or other trees.
- This is used to map (among other things) the directory structure.
- Identified by an SHA1 checksum.

Term: Commit

- A Commit references a Tree and some parent commits.
- Identified by an SHA1 checksum.
- Consistency: Use the SHA1-sums.

Term: Tag

- A Tag is a reference to a Commit.
- These beasts can be cryptographically signed.

For functional programmers

```
type meta;
datatype blob = BLOB of (Word8.word vector)

datatype tree = TREE of { name : string,
                          content : tree_content } list
and tree_content = TC_Tree of tree * meta
                 | TC_Blob of blob * meta

datatype commit = COMMIT of { parents : commit list,
                              author : string,
                              date : Date.date,
                              message : string,
                              t : tree }
```

Concept: Storage principles

- Reuse existing subtrees (dedup).
- Always write a new object.

Concept: Packs

- A pack compresses a set of objects into a single file.
- Delta Compression: Order objects by (type, basename, desc. size) lexicographically.
- Delta Compression: Run a sliding window on the order, search for delta-coding opportunities.
- Storage: Recency ordered in the pack from the HEAD.
- Storage: Good locality.
- Only "destructive operation" – fsync().
- Ideas the same as garbage collection.

Concept: Index

Conceptual Index location:

$$\textit{WorkingDir} \leftrightarrow \textit{Index} \leftrightarrow \textit{Storage}$$

Concept: Index(2)

- Index is *mutable*, storage is *immutable*.
- UI View: Staging area for the next commit.
- Backend View: Directory cache for speeding up operations.
- Git owes much of its speed to the index.

Term: Refs

- A *Ref* is a stick-it-note we can place on a commit.
- Refs are used for *branches*, different paths in development.
- Refs are used for certain magic markers in the storage tree.
- Some refs are *local*, other are *remote*.

Concept: Distribution

- Distribution happens by cloning the repository – copying all its contents.
- The storage model makes this approach feasible.
- Fetches: Copy the difference, track where the refs moved to.
- Pushes: Copy the difference, track where the refs moved to.

How to start

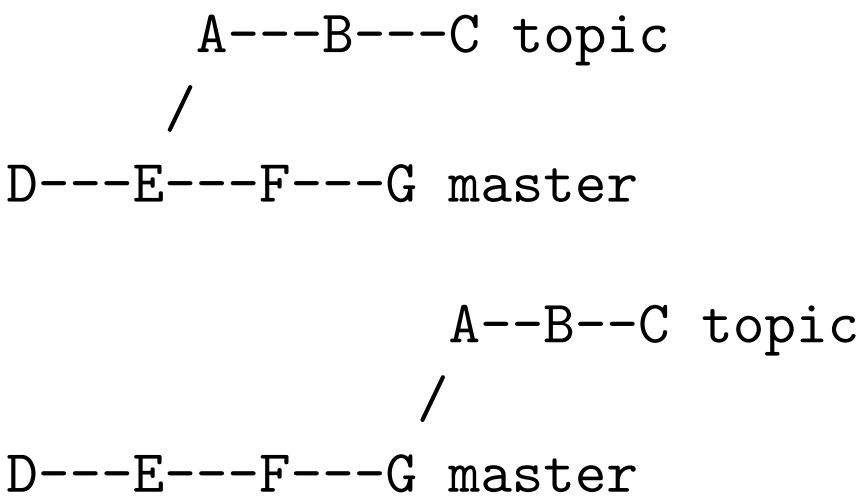
- Grab a tutorial.
- We think you gained a lot with the knowledge about the storage.

git add

- When you run `git add`, you add things to the index.
- `git commit` snapshots what is in the index alone.

merge/rebase

There is a difference between merge and rebase. Usually you want to rebase in git:



merge/rebase (2)

- Note: Patches already in upstream are skipped.
- Keeps a linear development path – easier to track for other people.
- As long as history is *local* it can be rewritten! (Persistence break!)
- Do not rewrite public branches, or people can't track it.

Moving files

- Controversy: git does not track that a file has been moved. It tracks it as a content change.
- A heuristic decides if the file is a move when browsing.
- Usually gets it right in most cases.

Publishing your changes

- One simple way is `github.com` or `gitorius.org`.
- Provides an outlet for publishing your tree.
- For internal hosting at companies, either pay Github, or
- Use a web-interface on top of a repository, or
- Use no central repository system at all!

Editor integration etc.

- Emacs: magit-mode.
- Vim: There is git-vim, or just use the command line.
- gitk – interface to git via Tk.

Question Time

- Questions from the audience?
- `http://github.com/ahf/sslug-workshop-git/`