FreeBSD and Git

Ed Maste - FreeBSD Vendor Summit 2018
Purpose

- History and Context - ensure we’re starting from the same reference
- Identify next steps for more effective use / integration with Git / GitHub
- Understand what needs to be resolved for any future decision on Git as the primary repository
Version control history

- CVS
  - 1993-2012

- Subversion
  - src/ May 31 2008, r179447
  - doc/www May 19, 2012 r38821
  - ports July 14, 2012 r300894

- Perforce
  - 2001-2018

- Hg Mirror

- Git Mirror
  - 2011-
Subversion Repositories

repo ➔ svnsync ➔ svn
Subversion & Git Repositories today

repo -> svnsync -> svn -> svn2git -> git -> git push -> github
Repositories Today

repo / svn

Freebsd github

downstream github
Repositories Today

repo / svn

Freebsd

github

fork

Downstream
github
“Git is not a Version Control System”

phk@ missive, reproduced at
https://blog.feld.me/posts/2018/01/git-is-not-revision-control/

Subversion vs. Git: Myths and Facts
https://svnvsgit.com/

“Git has a number of advantages in the popularity race, none of which are really to do with the technology”

10 things I hate about Git
https://stevebennett.me/2012/02/24/10-things-i-hate-about-git
Git popularity

Nobody uses Subversion anymore

False. A myth.

Despite all the marketing buzz related to Git, such notable open source projects as FreeBSD and LLVM continue to use Subversion as the main version control system. About 47% of other open source projects use Subversion too (while only 38% are on Git).

(2016)

https://svnvsgit.com/
Git popularity (2018)
Git UI/UX

Yes, it’s a mess.
Repository size (svn)

% svn checkout https://svn.freebsd.org/base/head freebsd-svn
494s

% du -sh freebsd-svn
3.3G freebsd-svn

% du -sh freebsd-svn/.svn
1.6G freebsd-svn/.svn

(svn 1.10.12)
Repository size (git)

% git clone https://github.com/freebsd/freebsd freebsd-git
872s

% du -sh freebsd-git
3.9G   freebsd-git

% du -sh freebsd-git/.git
2.3G
Repository update (no changes)

% time svn up
Updating ".":
At revision 339336.
  4.80 real   1.79 user   2.06 sys

% time git pull
Already up to date.
  0.45 real   0.09 user   0.09 sys
Monotonically increasing version numbers

A change was made in revision X, I have a checkout at revision Y, do I have the change?

Change: r339181

Checkout: r339296
Monotonically increasing version numbers

A change was made in revision X, I have a checkout at revision Y, do I have the change?

Change: r339181

Checkout: r339296

No - it’s a trick question. r339296 is stable/11 and r339181 is HEAD. Change MFC’d in r339300.
Monotonically increasing version numbers

“Do I have the change” depends on (branch, revision number) tuple

Git has no monotonically-increasing revision number, but can give commit counts:

% git rev-list HEAD --count
255139

Not inherent to the VCS, and not instantaneous as with SVN. Takes 3s on my desktop.

Could be manufactured by a pre-commit hook and inserted as a note (and shown via uname)
Partial tree checkouts

Previously not possible, but as of git v2.19:

% git clone --filter=blob:none

(Another example of “great” UX)
Shallow clones (without history)

% git clone --depth 1
Mutability of History

SVN

- By default, commit message may be modified, content may not
- In FreeBSD, a commit hook disallows commit message changes

Git

- Local history may be modified at will (git rebase -i)
- A commit hook disallows history modification in the definitive repository
Keywords

- $\text{FreeBSD}$ etc.
- Unsupported by git
- Can we do without them?
File rename/move/copy tracking

- Git infers renames and copies based on similarity
  - And often gets it wrong (e.g. new files with trivial content dwarfed by common header text)
- However, no concern over tree state fidelity
  - Tree content will match what’s expected
- History may be obscured
- Not aware of any work in progress on this in Git
Scalability

- Concerns with Git’s scalability to very large repositories
  - FreeBSD is not “very large”
- FreeBSD .git/ dir size ≈ Linux .git/ dir size
Binary artifacts / Lock-Modify-Unlock

- SVN supports Copy-Modify-Merge and Lock-Modify-Unlock
- Git does not natively support file locking (due to distributed nature)
  - Git LFS (large file storage) supports locking in v2.0
- Not a concern in FreeBSD
Serialization

- Typical configuration requires change(s) to be rebased at current head revision in designated repository
- Losing the race with another committer requires local update and retry
  - Similar issue exists with MFCs in Subversion today
- Commit time delta over the last year:
  - 0-1 min 955 9.8%
  - 1-5 min 1663 17%
  - 5-10 min 1068 11%
  - 10 min-1 hour 3608 37%
  - 1 hour-1 day 2429 25%
  - > 1 day 1 0.01%
FreeBSD’s path

1. Project provides Subversion only (any mirrors provided by others)
2. Subversion definitive repo with
   a. Best effort Git mirror (status quo)
   b. Supported and maintained Git mirror
3. Switch to Git as definitive repo
   a. Without Subversion mirror
   b. With Subversion access via GitHub
4. Something else
   a. Fossil
   b. Hg
   c. ???
FreeBSD Git hosting

- Git vs SVN is separate from self-hosted vs GitHub vs GitLab etc.
- Mirror on GitHub serves advocacy/marketing purposes and may be independent of a self-hosted server or other location
- Many options
  - GitHub
  - GitLab (self hosted or gitlab.com)
  - Gogs
  - Phabricator
Git in FreeBSD today

- Internal conversion (2 ways), not public
- GitHub mirror, in freebsd org
- 1539 GitHub forks
- 140 closed, 30 open pull requests
- Downstreams and developers depend on Git / GitHub mirror
DragonFlyBSD

- Since 2003-06-17, “Initial import from FreeBSD RELENG_4”
- Self hosted, GitHub mirror
- 174 unique authors, 21 --since=1year
- 38558 commits, 1863 --since=1year
- 75 forks
- 5 pull requests closed (as unsupported process)
Potential option 1

- Subversion definitive repository (source of truth)
- Promote Git conversion to fully supported
  - monitoring, SLA
- Formalize guarantee about changing hashes
Potential option 1a

- Continue with GitHub as the primary / only access to the mirror
- Formalize handling of pull requests and issues
- Develop stronger integration with Project resources, Subversion repo
  - CI integration (automatic testing of pull requests)?
  - Automatic pull request to Phabricator conversion?
  - Allow @freebsd.org users to tag a pull request for automatic SVN import?
Potential option 1b

- Provide our own infrastructure as the primary access to the mirror
  - Self-hosted GitLab?
  - Default Git server?
- Maintain GitHub mirror for existing users / marketing
- Still need to formalize handling of pull requests and issues
Potential option 2

- Resolve outstanding mirror metadata issues/concerns and technical debt as first step
- Make Git repository the definitive one
- Provide self-hosted access to the repository (GitLab, other)
- Maintain GitHub mirror
- Subversion mirror (1 or 2 way)
  - Not provided
  - Provided by GitHub
  - Developed internally
Open Questions
License

- git GPLv2
- libgit2 LGPLv2 w/ Linking Exception
- JGit EDL (new-style BSD) - Java
- dulwich Apache 2.0 - Python
- js-git MIT - Javascript
- go-git Apache 2.0 - Go

What does the license mean for FreeBSD? Server? Client?

Develop a small BSDL checkout-only tool (a la Portsnap)?
MFCs

Git mirror:

- Must be performed in Subversion
- Requires a committer

Git definitive:

- `git cherry-pick`
- Metadata can be associated with the commit (mergeinfo equivalent)
  - Not aware of an equivalent to "svn mergeinfo --show-revs eligible", but straightforward to develop own tooling
Vendor imports

Git mirror:
- Must be performed in Subversion, not feasible to stage / test in Git
- Must be performed by a committer

Git definitive:
- Migrate process to one of submodule, subtree, subrepo
- Imports and updates can be done by anyone
Git submodule

- Earliest approach, git built-in
- Additional git repo located at some subdirectory
- Main and submodule repos remain distinct
- Some awkward interaction with updates, branch switching, etc.
- General advice: avoid submodule
Git subtree

- Originally separate project, built-in as of May 2011
- Contrib repo grafted into main repo at specified location
- Contrib changes can be extracted for upstream development
- History is included
- Developers elsewhere in the tree do not need to be aware of subtree
- Research / experimentation required
Git subrepo

- WIP, multiple projects named subrepo
- History condensed to a single commit on import
- Individual subrepo commits may be pushed upstream
- Research / experimentation / testing / development required
Release Engineering

- Branching, tagging same process migration as other developers
- Freezes can be accommodated with pre-commit hooks, as today
- Freeze effectiveness is a separate topic, out of scope here
Security Team

- Git enables perfect-fidelity staging of commit(s) for security advisories
- Impact on advisories as previously discussed (monotonic change number)
Community

- Git maps well to Linux-style lieutenants
- SVN maps well to our flat model
- Committer / contributor distinction in hybrid model?
Next Steps (near term)

- Fix SVN-Git mirror to be reproducible
- Document transition plan for hash discontinuity
- Implement reliable monitoring (both availability and content)
- Formalize pull request handling
Next Steps (mid term)

- Prototype Git-primary project (downstream derivative)
- Incorporate some work-in-progress (e.g. Clang for all architectures)
- Allow experimentation with vendor branch strategies
- Investigate CI integration (proposed change testing)