Case Study:
How We Migrated the Enlightenment Project to Git

Tom Hacohen <tom.hacohen@samsung.com>
http://stosb.com
@TomHacohen

SAMSUNG
Open Source Group

What Made Us Go Through All the Trouble?

- We had some free time
- Already using Git-svn
- Back-porting patches was a pain
- Our development process was annoying and inefficient
- It was hard for downstream (e.g. Tizen) projects to stay in sync

"quick - evas scalecache **put this in svn do i dont lose my patch.** i'll revert and work on gettign the leak fixed."

— Carsten Haitzler

Comparison of Git and SVN

Git:

+ Leads to better commits (and messages) compared to SVN
  - Reorganise your commits after you are done (`git rebase -i`)
  - Split commits while working
+ Can work off-line seamlessly
+ Branches and tags are cheap
+ Attribution is built-in – Includes name and email
+ Supports multiple remotes

Comparison of Git and SVN (2)

Git (2):

+ Popular – Many already use it (thanks to Github)
+ Fast
+ Back/forward-porting is easy
+ De facto industry standard
+ AWESOME
– Some inconsistent commands

Comparison of Git and SVN (3)

SVN:

- \+ Linear revision numbers
  - In Git: `git rev-list --count HEAD`
- \+ Makes it harder for people to do drive-by/spray commits
  - In Git: just add a rule to Gitolite to disallow it
- − No diffs for binary files
- − Load on the server

"revert test commit. SVN e/trunk/efl is not locked?"

— Daniel Juyung Seo

Access Control for Git, through Git

Very fine-grained control

- Users and repos can be grouped
- Limit access based on users/groups
- Prevent operations (e.g. branch creation)
- Limit access to branches
- Limit access based on file/directory
- Deploys hooks
- Controls per-repo configuration

- Branches `devs/devname/*` are user-owned
- Can't rewind `<reponame>-version` and `master`

Gitolite (2)

- Secure
  - SSH login – restricted to `gitolite-shell`
  - Forces usage of public key authentication
  - One low privilege user
  - Managed using Git – access is cryptographically hashed
- Integrates with CGit (web viewer) and git-daemon for public repos
- Developer-owned repositories and branches (playground)

Managing Gitolite Users (Our Setup)

`admin/devs.git` repository:

- Remnant from our very early days – still used for access control
- Includes extra developer information (e.g. name and email)
- Different directories for different access levels
  - Three access levels: Probie, Developer and Maintainer

Gitolite Tips

- Make sure you use Gitolite 3.
- You can split group definitions to multiple lines!

```
@some_group = repo1 repo2 repo3
@some_group = repo4 repo5
```

- Gitolite supports macros and includes
- Read about VREFs
- Avoid Git hooks, VREFs can be used instead

"Revert previous commit. Damn svn, didn't mean to delete the whole dir."

— Tom Hacohen

Migration Goals

- Retain all the history since the dawn of time
  - Except when history is doubleplusungood
- Use Git's own branches and tags
- Make Git commands work as expected
- Fix attribution

Repository Layout

SVN

```
trunk/
edje/
eina/
enlightenment/
evas/
BINDINGS/
python-efl/
efl_cpp/
OLD/
THEMES/
dark/
...
```

Git

```
core/efl.git
core/enlightenment.git
bindings/python.git
bindings/cpp.git
legacy/evas.git
legacy/eina.git
legacy/edje.git
...
```

Git Migration Tools

- `git fast-import`
- `git svn`
- `git filter-branch`
- Git grafts

Why not `git fast-import`?

- `git svn` already does most of the work
- Would have to rewrite SVN commit parsing

Starting with the Migration

- We have a nice README
- Clone the repository:
  ```
  > git svn clone --use-log-author http://svnrepo/trunk
  ```
- Limit the repo to a subdirectory (optional):
  ```
  > git filter-branch --prune-empty -f --subdirectory-filter ↩
  subdirectory
  ```

Fixing Attribution

- Transform the authors:

```
> git filter-branch -f --env-filter 'eval ↩
 $(/path/rename_authors.sh)'HEAD
```

- rename_authors.sh's output:

```
> export GIT_AUTHOR_NAME=...
> export GIT_AUTHOR_EMAIL=...
> export GIT_COMMITTER_NAME=...
> export GIT_COMMITTER_EMAIL=...
```

- Check authors look OK:

```
> git shortlog -nse
```

Log Cleanups

- Beautify the SVN revisions in the log:
  ```
  > git filter-branch -f --msg-filter 'sed -e "s/git-svn-id: ↩
   [^@]*@\([0-9]*\).*/SVN revision: \1/"'HEAD
  ```
  Old: `"git-svn-id: svn+ssh://server/var/svn/e@2940 ..."`
  New: `"SVN revision: 2940"`
- Add .mailmap (optional)
  ```
  John Doe <jd@gmail.com> John D <jd@hotmail.com>
  ```

Following Repository Layout Changes

- Choose a unique file and run:
  ```
  > git log --follow --name-only -- path/to/file
  ```
- Get the list of the locations:
  ```
  > git log --name-only --format=format: --follow path/to/file | ↩
    sort | uniq
  ```

```
trunk/elementary/configure.ac
trunk/TMP/elementary/configure.ac
trunk/tmp/elementary/configure.ac
trunk/PROTO/elm/configure.ac
trunk/edje/configure.ac
```

Following Repository Layout Changes

- Choose a unique file and run:
  ```
  > git log --follow --name-only -- path/to/file
  ```
- Get the list of the locations:
  ```
  > git log --name-only --format=format: --follow path/to/file | ↩
    sort | uniq
  ```

```
trunk/elementary/configure.ac
trunk/TMP/elementary/configure.ac
trunk/tmp/elementary/configure.ac
trunk/PROTO/elm/configure.ac
trunk/edje/configure.ac
```

Following Repository Layout Changes (2)

- Create filter-branch.sh according to the list
- Run:

```
> git filter-branch -f -d /tmp/ram/filter --prune-empty ↩
 --tree-filter /path/to/filter-branch.sh HEAD
```

```
mv trunk/elementary newroot
mv trunk/TMP/elementary newroot
mv trunk/tmp/elementary newroot
mv trunk/PROTO/elm newroot
# trunk/edje/configure.ac -- We don't want that
```

Following Repository Layout Changes (2)

- Create filter-branch.sh according to the list
- Run:

```
> git filter-branch -f -d /tmp/ram/filter --prune-empty ↩
 --tree-filter /path/to/filter-branch.sh HEAD
```

```
mv trunk/elementary newroot
mv trunk/TMP/elementary newroot
mv trunk/tmp/elementary newroot
mv trunk/PROTO/elm newroot
# trunk/edje/configure.ac -- We don't want that
```

Duplicate Files – Our Very Own Hell

- Run:

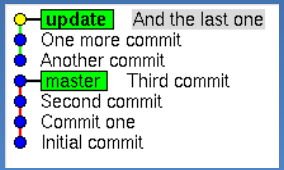
```
> git filter-branch -f --prune-empty --index-filter ↩
 path/remove_legacy_dup.sh START_HASH..END_HASH
```

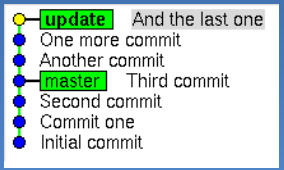
- The script contains:

```
> git log -C -C -M -M --name-status ↩
    $GIT_COMMIT^..$GIT_COMMIT | while read dir
...
> git rm --cached --ignore-unmatch -q
```

Git and an actively used SVN

- Get a clone with (just) the new commits:
  ```
  > git svn clone -r 83370:HEAD --use-log-author http://svnrepo/trunk
  ```
- Run all of the scripts as described
- Use grafts to stitch the trees together:
  - Set the graft points:
    ```
    > echo commit parent1 > .git/info/grafts
    ```
  - Make the graft permanent:
    ```
    > git filter-branch --prune-empty -f origin/master..HEAD
    > rm .git/info/grafts
    ```





Speeding Things Up

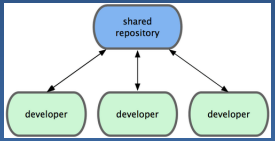
- Scripts ran for 3 days – with the optimizations
- `git filter-branch --tree-filter` was the slowest
  - `mv` relevant files to a new root directory, don't `rm -r`
  - Always used `mv`, never `cp`
  - Mount a filesystem on RAM and work from there
  - Was faster than `--index-filter` for large directory removals
- We sorted the `rename_authors.sh` list by commits per author
- Always choose the most suitable mode for `filter-branch`
- Lots of room for improvement, but this got us far

"Revert 'Revert 'Revert 'eina: use Eina_Spinlock for Eina_Chained_Mempool.' ' ' "

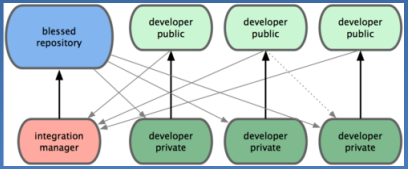
— Stefan Schmidt

Lets Us Experiment with Different Work-Flows

- Centralized work-flow (SVN – what we do now)



- Integration-manager work-flow



- Rebase and merge with no fast-forward

Makes Collaboration Easier

- Easy to fork
  - Work on features alone
  - Internal company clone
  - Easy to maintain local patches
- Works well with email-based development
  - `git format-patch`
  - `git send-email`
  - `git request-pull`
- Having an online patch review system helps contributions
  - Though the one we use (Phabricator) is lacking

Makes Development Easier

- Easy to back-port fixes – `git cherry-pick`
- Simple to work on multiple features at once – branches
- Possible to clean-up commits after the fact – `git rebase -i`
  - You can create progress snapshots while working
  - You can merge, split, reword and re-shuffle commits
- Have a cover-letter for a set of commits – `git merge --no-ff`

Makes Releasing Versions Easier

- Generate news from shortlog
  - We tag commits (in the message) with @fix, @feature, CID and etc.
  - We use the tagged commits + their shortlog for the NEWS file
- Getting logs between versions/master is very easy
- Can easily verify if we forgot to backport anything (grep for @fix tag).
- Can have longer freeze periods and shorter merge windows thanks to branches

"SCREW YOU GIT!... here is my fix for jack daniels leak!"

— Carsten Haitzler

Using Git

- Commit Messages
  - Funny commit messages in SVN → Pro-commit messages in Git
  - Git works best with a certain commit format
- Git dev-training
  - Tech restrictions
  - Education
- The Git book: `http://git-scm.com/book`
- Git - SVN crash course: `http://git.or.cz/course/svn.html`
- EFL developer Git practices: link

Using Git Like a Pro(ish)

- Use private dev repos and branches instead of creating dirs in repos
- `master` is stable – not a playground (Jenkins)
- Use their newly acquired powers (rebase -i, branches and etc)

"@68591: multiselect here is intentional to allow for theme overlays **try asking in irc or mailing list** before randomly changing things like this."

— Mike Blumenkrantz

Tools We Used

- Mailing-list – `git_multimail.py`
- IRC bot – Irker
- CGit

"efl/doc: Remove all .svn entries from the generated file. This is now longer needed now that we switched to git."

— Stefan Schmidt

Reviewing the Changes One Year After...

- Haters gonna hate, trolls will be trolls...
- Though early haters grew to like it
- The community considers the switch as a very good thing
- People are starting to utilize Git's abilities.
  - People develop in feature branches
  - Like the refined access control
  - Back-port using cherry-pick
  - and more...

Conclusions

+ Helped our development work flow
+ Social and technical changes are both needed
+ Flexible for the future
– Learning curve involved

“mooooo changed rev to .10”

— Mandrake

Coming soon. Maybe...

- Try new work-flows
- Use pre-receive lint, style checkers and etc.
- More refined access control
  - Setting maintainers – Extra access the rest do not have

"Thanks for listening, questions?"

— Tom Hacohen
tom.hacohen@samsung.com
http://stosb.com
@TomHacohen

Resources Attributions

- Page 5, resources/git-logo.png
- Page 6, resources/subversion-logo.png
- Page 8, resources/keep_gate_closed.jpg
- Page 16, resources/i-am-too-lazy-for-this.jpg
- Page 28, resources/git_centralized.png
- Page 28, resources/git_integration_manager.png
- Page 34, resources/superhero_camp.png
- Page 38, resources/haters_gonna_hate.jpg

*This presentation is based on a talk Daniel Wilmann and myself gave at LinuxCon EU 2013.

I didn't want to talk about it...

Phabricator*

- It's PHP! So can be easily hacked on by any C programmer
- Nice, shiny, we like to use it
- Lets you connect with Facebook, Twitter and Google!
- Makes contributions easy.

* The opinions reflected in this slide may not accurately represent the opinions of the presenter.

I didn't want to talk about it...

Mailing-lists – tweaked `git_multimail.py`

- Built-in support for gitolite
- Very customizable, but we had to tweak it anyway...
  - Removed cover-letter
  - Allow empty announcement addresses without it complaining
  - Change "From:" (since solved in upstream)
- Awful-hacks – spoofing "From:"
  - Hack v1.0: Keep the address, spoof the name (useful with sourceforge)
  - Hack v2.0: Spoof the address + name (self-hosted ML)

I didn't want to talk about it...

IRC bot – Irker

+ Simple and easy to deploy
+ We fixed issues, and tweaked (simple code)
  - Fixed Unicode support (fixed in upstream)
  - Fixed presentation of branches with front-slashes in the name
  - Print author name instead of username
- Annoying timeout bug (fixed in upstream)

I didn't want to talk about it...

CGit Web-Frontend

+ Looks nicer and faster than gitweb
+ Integrates with Gitolite
  - Repository owner/description is controlled via Gitolite
  - Dev repo description is controlled by users
+ Category set via Gitolite
− Sort is only alphabetical (dev repositories)