Case Study:
How We Migrated the Enlightenment Project to Git

Tom Hacohen
<tom@stosb.com>

Daniel Willmann
<daniel@totalueberwachung.de>

LinuxCon Europe 2013

What Made Us Go Through All the Trouble?

- We had some free time
- Already using Git-svn
- Backporting patches was a pain
- Our buildbot setup was unreliable
- Test-suite breaks usually went unnoticed

"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

Comparison of Git and SVN (2)

Git (2):

- \+ Fast
- \+ Supports multiple remotes
- \+ Back/forward-porting is easy
- \+ 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
- – 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
- 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)

"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 double-plus ungood
- 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/
        detorious/
...
```

Git

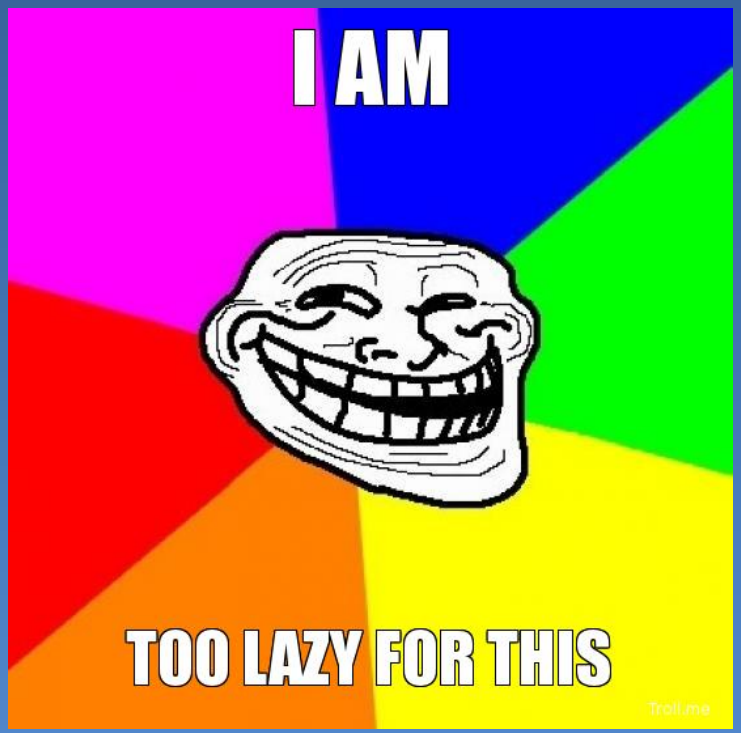
```
core/efl.git
core/enlightenment.git
bindings/python.git
bindings/cpp.git
legacy/evas.git
legacy/eina.git
legacy/edje.git
...
```

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
```

Clean Ups

- 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
```

- Add .mailmap

```
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:

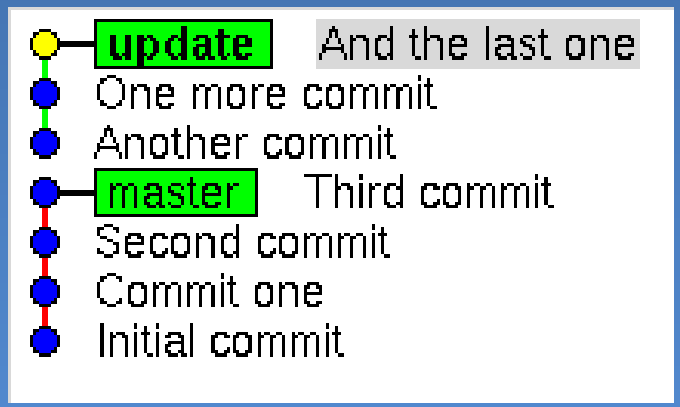
```
> git filter-branch -f --prune-empty ↩
 --index-filter path/remove_legacy_dup.sh ↩
 START_HASH..END_HASH
```

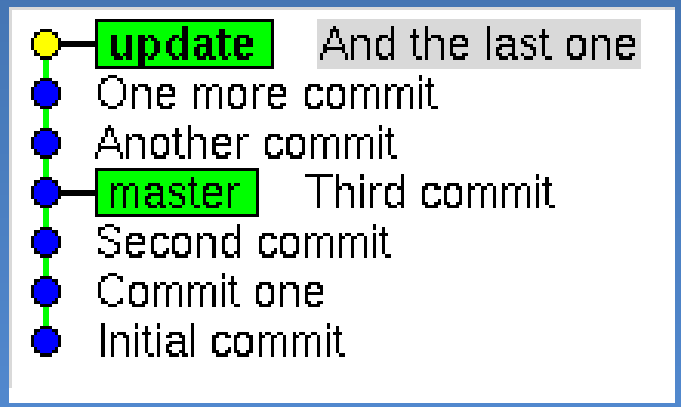
- The script contains:

```
> 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

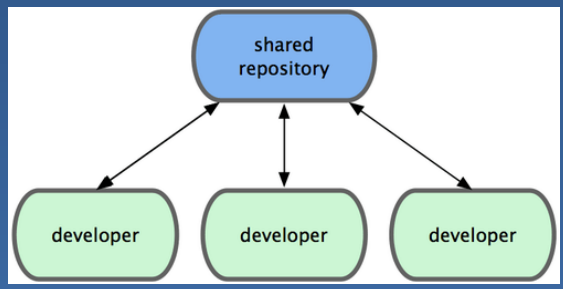
- Scripts ran for 3 days – with the optimizations
- `git filter-branch --tree-filter` was the slowest
  - `mv` relevant files to a new root directory, don't `rm -r`
  - Always used `mv`, never `cp`
  - Mount a filesystem on RAM and work from there
  - Was faster than `--index-filter` for large directory 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.' ' ' "

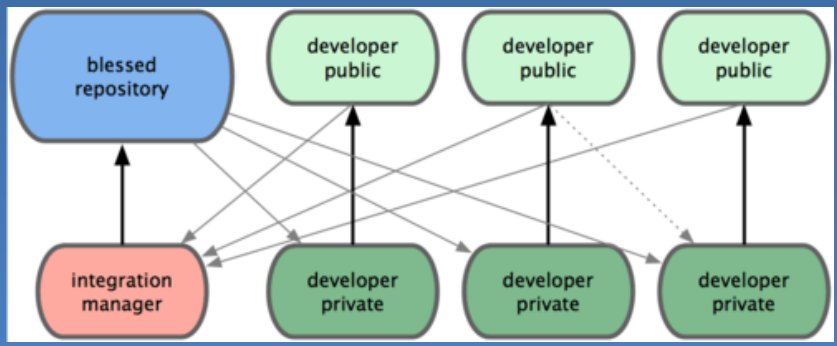
— Stefan Schmidt

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 Development 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`

"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

Conclusion

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

"Revert 'eina: fix CID 1106340: Logically dead code (DEADCODE) reported by coverity.' "

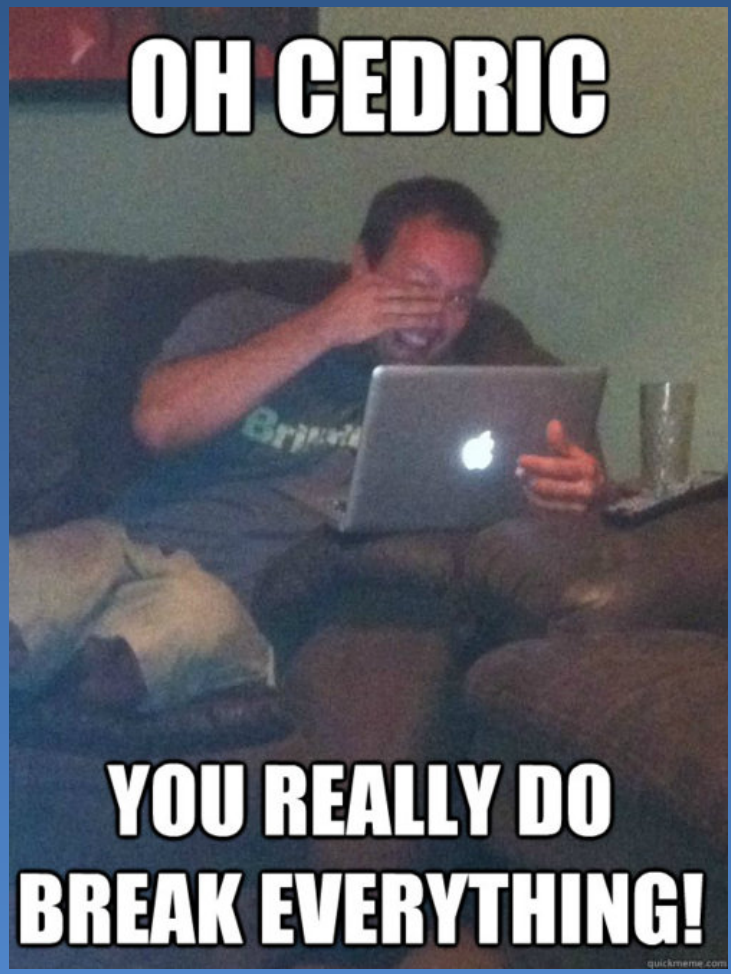
— Carsten Haitzler

Testing Is Good

- Everyone needs testing
- Things break when optimising, we optimise a lot

CEDRIC

- “fix cedric’s image property code... that broke load opt downscaling...”
- “CEDRIC! REVERT! this breaks text labels in e17 default theme when eselected - they all jump up to the top! :)”
- “finally found evas_map weirdness bug. CEDRIC code...! commit #74180.”



Unit/Regression Tests

- We don't have many, but some parts are well tested
- Many types of tests
  - Functions/data types
  - Rendering results
  - Animations (not always the same)
  - Different engines (Wayland, X11, Framebuffer, etcetera)
- I try to add a test case for every bug I fix
- We use libcheck
- Address sanitizer is awesome

Exactness – Pixel-Perfect Regression Testing

- Easy to use:
  1. Record (once): `exactness -r test_list.txt`
  2. Init (once): `exactness -i test_list.txt`
  3. Play (nightly): `exactness -p test_list.txt`
- Pixel-perfect regression testing
- Supports running tests in parallel
- Does not depend on X/Wayland

Checking Tests' Coverage

- Uses lcov - function, line, branch coverage
- Bad coverage means a lot, good coverage means a bit
- Eo is a good example (although someone broke it a bit recently)

lib/eo		93.0 %	1002 / 1077	93.4 %	99 / 106

- Evas as a bad example

lib/evas		4.4 %	3 / 68	9.1 %	1 / 11

Catching Issues in Uncommon Paths

- Clang `scan-build`, coverity, klockwork (licensing-issue)
- Compiler warnings (GCC and Clang)
  - `-Wall -Wextra -Wshadow -Wpointer-arith`
  - Some warnings are hard to maintain, e.g. unused parameters
- All of which found many bugs

"eio: Removed eio_map.c from build temporarily.
SPANK CEDRIC!"

— Daniel Juyung Seo

Reasons for Switching

Buildbot

- \+ Python
- \+ Already in operation
- \+ Very flexibile
- – Custom scripts for our setup
- – Not actively maintained
- – Many false-positives

Jenkins

- \+ Large userbase
- \+ User-facing interface was nicer
- \+ Many plugins available
- – Java (AbstractJobImplementationHandlerObserverFactory)
- – Not easily hackable

A Short Introduction

- One master, many slaves
- Slaves for load balancing, and testing different architectures
- Jobs are the building blocks
- Plugins extend Jenkins' functionality

Plugins (1)

Inheritance

- Reuse of job fragments
- Some plugins are not well supported
- Would be nice to have Matrix builds with inheritance

Git

- Integration is easy
- Tracking different branches for different jobs
- Polling for now (Could/should be changed)

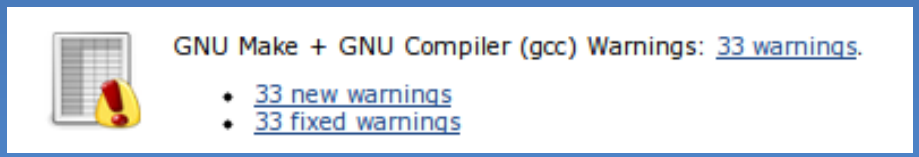
Plugins (2)

Scoreboard

- Gamification
- Some people are proud of their negative scores. . .
- Popular in the beginning, but not so much anymore

Warnings

- Record and summarize compiler warnings
- Trend graphs



Catching Issues Early

- Compile with `gcc` and `clang`
- x86, x86_64, x32 and mingw (windows) regularly tested
- Collect warnings
- Run test suites where available
- Enable address-sanitizer
- Nightly builds test more

Address Sanitizer

- Memory checker (compiled in with `-fsanitize=address`)
- Finds memory corruption like:
  - off-by-one issues
  - Writes to, reads from invalid memory
  - Use after `free()`
- Provides a backtrace where the error occured and where the memory region was allocated/freed

```
==9442== ERROR: AddressSanitizer ↩
 heap-use-after-free on address 0x7f7ddab8c084 at ↩
 pc 0x403c8c bp 0x7fff87fb82d0 sp 0x7fff87fb82c8
```

And more info...

Residual Failures

- `TMPDIR` runs out of disk space
- Jobs interdependencies (link to libraries from other jobs)
- Sometimes builds just hang (IRC plugin?), but overall pretty reliable

Conclusion

+ Found lots of bugs
+ Address sanitizer helps a lot
+ Quality of contributions has improved
– Still some issues with build failures
– Not enough test coverage

“moooooo changed rev to .10”

— Mandrake

Coming soon. . .

- Try new work-flows
- More refined access control
  - "Probie" access – Give access to everything but the official branches

Coming soon. . .

- Fix the exactness-elm tests
- Add exactness-enlightenment tests
- Write more tailored exactness test applications

Coming soon. . .

- Test your own branch: pushing to `devs/<devname>/jenkins` triggers a build
- Git notes to publish job results (planned)

“Thanks for listening, questions?”

— Daniel Willmann & Tom Hacohen

Resources Attributions

- Page 5, resources/git-logo.png
- Page 6, resources/subversion-logo.png
- Page 15, resources/i-am-too-lazy-for-this.jpg
- Page 27, resources/git_centralized.png
- Page 27, resources/git_integration_manager.png
- Page 31, resources/superhero_camp.png
- Page 36, resources/engineering_fail_camera.jpg
- Page 37, resources/tumblr_cedric_break_everything.jpg

We 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!

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

We 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:"
- Awful-hacks – spoofing "From:"
  - Hack v1.0: Keep the address, spoof the name (useful with sourceforge)
  - Hack v2.0: Spoof the address + name

We 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 presentation of branches with front-slashes in the name
  - Print author name instead of username
- Annoying timeout bug

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

CGit Web-Frontend

+ Written in C
+ 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)