git-series: Tracking the History of History

Josh Triplett
josh@joshtriplett.org

Linux Plumbers Conference 2016
RFC: feature
RFC: feature
Development
git commit
git format-patch -3
git format-patch -3

[PATCH 1/3] Cleanup and yak shaving
[PATCH 2/3] Implement feature
[PATCH 3/3] Use feature
git request-pull ...
git request-pull ...
git hub pull new ...
Feedback

- Split cleanup and yak shaving
Feedback

- Split cleanup and yak shaving
- Additional use of feature
Feedback

- Split cleanup and yak shaving
- Additional use of feature
- Add benchmark data
Feedback

- Split cleanup and yak shaving
- Additional use of feature
- Add benchmark data
- Add tests
Feedback

- Split cleanup and yak shaving
- Additional use of feature
- Add benchmark data
- Add tests
- Fix typo
Rewriting history
git commit --amend
git rebase -i
“fast-forward” vs “non-fast-forward”
git format-patch -v2 -6
git format-patch -v2 -6

[PATCH v2 1/6] Cleanup
[PATCH v2 2/6] Yak shaving
[PATCH v2 3/6] Implement feature
[PATCH v2 5/6] Use feature
[PATCH v2 6/6] Use feature elsewhere
What about v1?
git reflog
mutt -f =Sent
Public mailing list archives
Git tracks history
We rewrite history
We need the history of history
git submodule
git submodule --fthagn
Two common solutions
Two common solutions

Pull one of the histories out of git
Pull the patches out of git
quilt patch files
debian/patches/*
git rebase -i
Pull the history of the patches out of git
Versioned branch names
feature-v1
feature-v1
feature-v2
feature-v1
feature-v2
feature-v3-typofix
feature-v1
feature-v2
feature-v3-typofix
feature-v8-rebased-4.6-alice-fix
feature-v1
feature-v2
feature-v3-typofix
feature-v8-rebased-4.6-alice-fix

feature-v8-rebased-4.6-alice-fix.pptx
We have a version control system!
Cover letter
Cover letter

[PATCH v2 0/5] feature: summary of new idea
Base
git format-patch -3

git format-patch -v2 -6
Base

git format-patch -3

git format-patch -v2 -6

git rebase -i ...

Base

git format-patch -3

git format-patch -v2 -6

git rebase -i ...

git log
Collaboration
“Never rewrite published history”
How can you collaborate on it?
Patch series
Patch series

Feature backport
Patch series
Feature backport
Distribution package
git-series
Tracks the history of a patch series
git-series

- Tracks the history of a patch series
- Handles non-fast-forwarding changes
git-series

- Tracks the history of a patch series
- Handles non-fast-forwarding changes
- Tracks a cover letter
git-series

- Tracks the history of a patch series
- Handles non-fast-forwarding changes
- Tracks a cover letter
- Tracks the base of the series
Internals
Review of git internals
Review of git internals

- blob (file)
Review of git internals

- blob (file)
- tree (directory)
Review of git internals

• blob (file)
• tree (directory)
• commit (tree, message, parents)
Review of git internals

- blob (file)
- tree (directory)
- commit (tree, message, parents)
- tag
Review of git internals

- blob (file)
- tree (directory)
- commit (tree, message, parents)
- tag
- ref
trees can refer to commits
trees can refer to commits

“gitlink”
Requirement: Every object must remain reachable by git.
Requirement: Every object must remain reachable by git.

- Required to keep git from pruning objects
- Required for push/pull of series
- git doesn’t follow gitlinks for reachability or push/pull
- git doesn’t follow gitlinks for reachability or push/pull
- Have to also include “series” as a parent
• git doesn’t follow gitlinks for reachability or push/pull
• Have to also include “series” as a parent
• git-series ignores that parent when traversing
HEAD  →  Current branch
HEAD $\rightarrow$ Current branch

refs/SHEAD $\rightarrow$ Current series
Working and staged (as seen in status)?
Working and staged (as seen in status)?

refs/git-series-internals/working/feature

refs/git-series-internals/staged/feature
Avoiding errors
Long and complex error messages
Long and complex error messages suggest a design flaw.
Long and complex error messages suggest a design flaw

Redesign to make the error impossible
Detach from a series or check out a new series with uncommitted changes to the series (series, base, cover)
Detach from a series or check out a new series with uncommitted changes to the series (series, base, cover)

Every series has its own independent working and staged versions
Detach from a series or check out a new series without making any commits
Detach from a series or check out a new series without making any commits

git series start makes working/staged
Detach from a series or check out a new series without making any commits

git series start makes working/staged "(new, no commits yet)"
Detach from a series or check out a new series without making any commits

`git series start` makes working/staged
   "(new, no commits yet)"
`git series checkout` works
Long and complex error messages suggest a design flaw

Redesign to make the error impossible
git series rebase
git series rebase

git rebase --continue
Rust and libgit2
https://github.com/git-series/git-series
https://github.com/git-series/git-series

Questions?