Contributing to FreeBSD via Github

A guide to all the fussy bits

Warner Losh
Some History
The Github Pull Request Experiment

- Started just after git migration
- What to do about github pull requests?
- Core requested I figure out next steps for our workflow.
- How do we use git better?
- What can we do to improve our workflow?
Pull Request Ideals

- Easy path for mature changes
- Integrate worth while changes
- Make contribution easier
- Stop ignoring changes
- Improvements our process and culture
- Help recruit new talent
Boring Background
Overview

● Baseline assumptions
  ○ You know what git is and how to use it
  ○ You know what Github is
  ○ Vague familiarity with FreeBSD’s infrastructure

● Github
  ○ Will highlight the basic mechanics
  ○ Some GUI pictures

● FreeBSD Expectations
  ○ What we want
  ○ What we don’t want
  ○ How we want it packaged

● How to Help Out
Why Github?

- A familiar process to many
- An experiment to try to increase engagement
- Improve our developer pipeline
- Easy to publish to Github w/o being completely dependent on it
- A place to try new ways to improve our general development workflow
- Changes are more visible and discoverable
- Better fit for more people than Phabricator
Target Audience

- Casual user with small fix
  - Man page improvements
  - Minor tweaks (new devices, build fix, etc)
- Casual developers with small to medium sized change
  - Fix to broken behavior
  - Optimizations with measured speed ups
  - Minor new feature
- People Wanting to Contribute
  - Improve the FreeBSD experience
  - Improve the patch acceptance workflow
- Vendors with drivers for their hardware
- Focuses on base system, with callouts for docs and ports
Only One Part of the Story

- Bugzilla still for bugs
- Project private resources guard the source of truth repo
- Phabricator for developer review
- Complements CI efforts
- Mailing Lists
Why Not Phabricator?

- Have to create a new account
  - Most people already have github account
- Hard to discover changes
  - Phabricator’s interface makes it hard to find things to commit
- Hard to extract information
  - Phabricator tracks less information than git, and creating a commit message with proper credit from that is hard
- Is End of Life
  - Although things like Phorge are replacing it, they aren’t materially fixing these defects
- Phabricator is a developer tool to communicate with other developers
  - Not a friendly place to contribute a change
Getting Started – Basic Flow
Basic Flow of Commits

- FreeBSD cgit
- Github fork of freebsd-src repo
- Developer repo
- Github freebsd freebsd-src repo
- FreeBSD cgit src repo
- Staging Area (also developer repo)
- Pull Request
- Main mirroring
- Fork

Developer downloads Pull Request
Main commits
Create a new fork

A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks.]

*Required fields are marked with an asterisk (*).*

**Owner** *

- **Repository name** *

  - **freebsd**/freebsd-src

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

**Description** (optional)

The FreeBSD src tree publish-only repository. Experimenting with 'simple' pull requests....

**Copy the main branch only**

Contribute back to freebsd/freebsd-src by adding your own branch. [Learn more.]

You are creating a fork in your personal account.

Create fork
Grabbing the URL

https://github.com/qemu-bsd-user/freebsd-src
Clone your new repo

```
10:46am rebo: [20061]> git clone -o github \
   https://github.com/qemu-bsd-user/freebsd-src
Cloning into 'freebsd-src'...
remote: Enumerating objects: 3287614, done.
remote: Counting objects: 100% (993/993), done.
remote: Compressing objects: 100% (585/585), done.
remote: Total 3287614 (delta 412), reused 815 (delta 397), pack-reused 3286621
Receiving objects: 100% (3287614/3287614), 2.44 GiB | 22.06 MiB/s, done.
Resolving deltas: 100% (2414925/2414925), done.
Updating files: 100% (100972/100972), done.
10:50am rebo: [20062]> cd freebsd-src
```
Make your changes

10:50am rebo:[20063]> git checkout -b bsdcan-demo
Switched to a new branch 'bsdcan-demo'

10:51am rebo:[20064]> vi mumble-foo

10:55am rebo:[20065]> git commit -a

- Rince, Lather, Repeat
Push Changes to Your Repo

11:15am rebo: [20066]> git push github
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 64 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 301 bytes | 3.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote:
remote: Create a pull request for 'bsdcan-demo' on GitHub by visiting:
remote: https://github.com/bsdimp/freebsd/pull/new/bsdcan-demo
remote:
To github.com:bsdimp/freebsd.git
  * [new branch]                bsdcan-demo -> bsdcan-demo
Open a pull request
Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks. Learn more about diff comparisons here.

Able to merge. These branches can be automatically merged.

Add a title
Remove cat -v

Add a description
Cat -v is considered harmful. Remove it.
Suggested by: Rob Pike

Create pull request
More changes, repush

10:55am rebo:[20065]> git push github --force-with-lease
THIS COUPON GOOD FOR:
ONE (1) FREE
DISAPPOINTMENT
WHILE SUPPLIES LAST
Evaluating the Changes
First Steps

- Does it pass the CI steps on github
- Are there complaints from our ‘checker’ scripts
Do We Want It?

- Is it important enough
  - Typo fixes generally not wanted
- Automated Checkers
  - Signed errors
  - Theoretical problems
- Does it fix a real problem?
- Does it solve an interesting problem?
- Generally, cleanup and style changes generally not wanted
- But… on going work in an area may warrant an exception
Is it correct

- Is the code correct?
- Does it solve it in a desirable way?
- Does it integrate well into FreeBSD systems?
FreeBSD Style

- Style checker in Github
- But there’s FreeBSD architecture considerations
- Integrates to the FreeBSD specific things
Right Size

- Large enough to do something interesting, desirable and useful
- Small enough to be able to review
- < 10 commits
- < 200 lines changed
- (though those are just guidelines, not hard limits)
- Exceptions do apply
Right Subject

- Change in user-visible behavior
- Change that has consensus
- Change that is worth volunteer time to review, test and commit
Changes Are Mature Enough

- Some changes during review are inevitable
- Major rewrites
  - Must be tested
  - Must have been socialized
  - Must have little to no dissent in the community
- Changes are stable
  - Experimental changes that crash are not welcome
Vendor Driver Changes

- Can be large
- Assume vendor tested
- But we build everywhere (vendors don’t always)
- Sanity checks
- Possible alternative to “vendor commit bit”
Some Bad Examples

● Typos in comments
  ○ Nobody cares, unless you have real bug fixes too

● Theoretical Bugs
  ○ The kind found by “scanners” that look for patterns
  ○ That won’t change behavior
  ○ Though some exceptions may apply

● Changes that don’t even compile
  ○ Changes should pass the github testing jobs

● Changes that fix one thing but break other things
How to Help
Github Actions

● Moving the process along
  ○ “Thank you for your submission”
  ○ “Thanks for your update” + tag

● Better checking
  ○ Commit message checking
  ○ Merge commit filtering
Tooling

- Staging
- CI testing
- Pushing with rebase
- Other sanity checkers
  - Run igor on man page changes
  - Run lua checker on lua changes
- Context Sensitive Checking
  - No need to rebuild world for man page changes
  - But need to for man page addition (and install too!)
Questions?
Warner Losh
imp@bsdimp.com