COLLABORATION BEST PRACTICES

WITH GIT & GITHUB
AGENDA

- Spruce up these commit messages
- Force pushing is a time bomb
- 2 everyday kinds of rebase
- Pull Requests FTW!
- Issues & PR triaging
- Lower the barrier to entry
- Wrap git with hub
HI, I'M CHRISTOPHE

github.com/tdd
@porteneuve
(Twitter, Medium)
SPRUCE UP THESE COMMIT MESSAGES
SPRUCE UP THESE COMMIT MESSAGES

LAZY MESSAGES ARE LIKE SPITTING IN PEOPLE'S FACES

▸ Commit message = quick doc for others + for the future
▸ Critical to a useful usable history
SPRUCE UP THESE COMMIT MESSAGES

OTHERS ARE... YOU, 3 MONTHS DOWN THE LINE
GREAT COMMIT MESSAGE: RULE #1

- 1st line ≤ 50 chars, present imperative, no final stop.

❌ Fixing incorrect UTF16-LE padding, as we are all slaves to !@# MS Excel.

✔ Fix incorrect UTF16-LE padding
SPRUCE UP THESE COMMIT MESSAGES

GREAT COMMIT MESSAGE: RULE #1 PROTIP

- Standardized emojis as starter? (see Atom)
  - 🔥 Remove newlines_after_classes rule
  - 🎨🔨 Wording
GREAT COMMIT MESSAGE: RULE #2

- Blank line before details,
- Details lines ≤ 72 chars, hard-wrapped
  - nontrivial decisions, especially merge arbitrations (keep the conflict lines, btw)
  - algorithm details
- etc.
SPRUCE UP THESE COMMIT MESSAGES

GREAT COMMIT MESSAGE: RULE #2 PROTIPS

- `git merge --log/merge.log` configuration setting

- `git cherry-pick -x`

---

```
commit 4b7af9e
Merge: 7f028a2 3516c99
Author: Christophe Porteneuve <tdd@tdsworld.com>
Date: Thu Nov 12 10:33:44 2015 +0100

Merge branch 'ga-tracking' into HEAD

* ga-tracking:
  Tracking lecture homepage via GA
  Début GA async
```

```
commit 331941d
Author: Christophe Porteneuve <tdd@tdsworld.com>
Date: Tue Jul 28 18:11:37 2015 +0200

Added CSS
(cherry picked from commit 71cf2134b5ef787f46ec1f059d36962c3af84d82)
```
SPRUCE UP THESE COMMIT MESSAGES

GREAT COMMIT MESSAGE: RULE #3

▸ **GFM** everywhere: speak it!
  ▸ Italics, bold
  ▸ Auto-linking
  ▸ Explicit links
  ▸ ...and more!

▸ Auto-close issues when appropriate, e.g. **Fixes #42**.
FORCE PUSHING

IS A TIME BOMB
FORCE PUSHING IS A TIME BOMB

INVALIDATE ALL THE SHAS!... FOR NO GOOD REASON?

- Force-pushing usually means you rewrote some history
- Think about downstream users: QA, partners, deploys, subtrees...
- For public-facing code, legitimate only when faced with legal/security risk (usually because you versioned sensitive data, especially third-party credentials)
- Otherwise, just git rm --cached or git revert, depending on the use-case.
FORCE PUSHING IS A TIME BOMB

PROTECT AGAINST ABUSIVE FORCE PUSHING

- Protected branches
- Per-branch rights management
- Doug tends to force push:
2 EVERYDAY KINDS OF REBASE
THE DIRTY LITTLE SECRET OF PULL’S DEFAULT BEHAVIOR

- `pull = fetch + merge`
- Terrible, terrible idea. Means collaborators on the same branch merge it all the time “upon itself.” Messy graph.
- So why? Because, well, legacy.
SETTING PULL.REBASE TO PRESERVE

- What we really want is for our local work to rebase on top of the updated remote branch. **Single line per branch = cleaner graph.**

- `git pull --rebase=preserve `/pull.rebase=preserve` configuration setting

- “preserve” needed to avoid inlining local merges. 1.8.5+

- Also helps transparently pull rewritten histories
Cardinal rule of Git: commit often, push when needed

Not pushing early gets you leeway to clean up your log

Pre-push reflex:

- `git log --oneline --graph --decorate @{push}..`
- `git rebase -i`

Cleaner logs = more usable logs = more productivity

* or `@{upstream}/@{u}` on older versions, that don't have separate suffixes for pull/push
PULL REQUESTS

FTW!
PULL REQUESTS FTW!

NOT JUST FOR FORKS

▶ At the heart of the GitHub Workflow
▶ PR = “Please, can I get some code review on this?”
▶ Don't commit directly into master
RESPECT THE REVIEWERS’ TIME

- Don’t just re-use your latest commit message. Take the time to write a descriptive, useful PR text. Make your reviewing needs explicit.
- You’re asking others to dedicate time to you. Play nice.
- Leverage GFM whenever relevant (tables, images, etc.)
PULL REQUESTS FTW!

SAVE ON YOUR OWN TIME

- **Reply by e-mail.** Use standard signature prefixes.

- If you target specific reviewers, @-mention them directly in your PR text. Think about **team mentions**.

- Later on, bring relevant people in by @-mentioning them.
RESPECT THE AUTHOR’S TIME

▸ Use line-specific comments

▸ Finalize your first pass with a “cover letter” summary comment in the main discussion thread
CONFLICTING PULL REQUESTS

- Impossible to auto-merge? Grab the PR locally, merge manually, push. Will auto-merge/close the PR.

1. `git fetch origin refs/pull/23/head`
2. `git checkout master`
3. `git merge FETCH_HEAD`
4. Manually solve, add and commit...
5. `git push origin master`
PULL REQUESTS FTW!

A NOTE ON PROTECTED BRANCHES

- Use that for branches that need **extra special care**.
  - No force-pushing
  - No deletion
  - No merge allowed until **all status checks pass** (CI, etc.)
ISSUES & PR

TRIAGING
LABEL LIKE A BOSS

- Issues and PRs
- Use the classics; **color-code** for quicker scanning
- **Filter** listings and **save common combos**
ISSUES & PR TRIAGING

ASSIGNEES

- Assign **responsibility** ASAP
- Re-assignment always possible
- **Common filters** available
- Global, **cross-repo listings** in the top bar
ISSUES & PR TRIAGING

MILESTONES & PROGRESS TRACKING

- Can **group** issues and PRs
- **Due date** possible (e-mail warnings a couple days ahead)

![Milestones and Progress Tracking Dashboard](image-url)
PUSHING THE ENVELOPE WITH THE ECOSYSTEM

- Project management on top of GitHub
  - Huboard, Waffle...
- Advanced PR review workflows
  - ReviewNinja, Reviewable, GitColony...
- Statuses API for advanced checks on PR mergability
- ...and more!
LOWER THE BARRIER TO ENTRY
A NOTE ABOUT LICENSES

- No license ≠ “do what you want”
  - Actually the opposite: “all rights reserved”
  - With the exception of GitHub’s ToS
- Learn more at choosealicense.com and mass-apply with addalicense.com 😊
WHAT'S IN A GREAT README?

▸ Synopsis (elevator pitch)
▸ Code example (if library)
▸ Motivation / Rationale / How it differs from alternatives
▸ Installation
▸ Getting started
▸ API reference (or link to it) (if library)
▸ Cross-links to contribution guide / wiki / license(s)
CONTRIBUTION GUIDELINES

- Typically **CONTRIBUTING.md** ([here’s a great one](#))
- **Auto-detected** by creation forms for issues and PRs
- Include **Code of Conduct** for anyone involved! ([start here](#))
LOWERING THE BARRIER TO ENTRY

WIKIS FOR EXTENDED DOCS

▸ Ideal for more extensive docs than README / contrib.
  ▸ Extensive API documentation
  ▸ How-To’s
  ▸ Jumpboard for external links to prominent articles, tutorials, screencasts, etc.
LOWERING THE BARRIER TO ENTRY

GITHUB PAGES FOR “MARKETING” SITES

- Now on github.io
- Based on gh-pages branch (for projects), special repos (for users and organizations)
- Custom domains are a snap
- Super easy to get started (generator), then can be customized (Jekyll-based)
- Great for demos, too: HTTPS means availability of numerous web technos, e.g. ServiceWorker.
5MINFORK OR MORE

- Temporary **static hosting** of a branch from a GitHub repo
- Entirely URL-based
- Extensions and bookmarklets for convenience
- Neat little **free** service: 5minfork.com
- Demo'ing your code needs more involved hosting? Check out Heroku Buttons.
CONCISE AND FAST

- `git clone cool-project` (from my user account)
- `git clone ramda/ramda` (from another account)
- `git fetch sam,tobie` (adds remotes and fetches them)
- `git checkout https://github.com/github/hub/pull/134`
- `git compare v0.9..v1.0`
- `git browse resque issues`
FORKING AND PR’ING IN A SNAP

- `git fork` (inside a clone of the upstream)
- `git pull-request`
  - `git pull-request -b sam:master`
  - `git pull-request -F prepped-pr.txt -h d3-experiment -o`
  - `git pull-request -i 123`
- And **so much more!**
THANK YOU!

HTTP://BIT.LY/COLLAB-BP

ALSO SEE
HTTP://BIT.LY/GIT-PROTIPS-VIDEO
HTTP://MEDIUM.COM/@PORTENEUVE
IMAGE CREDITS

- **Barn raising** © 2010 Vuong Ta, U.S. Navy
- **Suitcase bomb** © 2014 Rafael Mizrahi
- **Cleaning products** © 2014 Dave O
- **Wikipedia Higher Education Summit Boston** © 2011 Sage Ross
- **Organized** © 2006 Uwe Hermann
- **NodeSchool Paris #4** © 2015 Christophe Porteneuve
- Octocat icon from the **Octicons** font © GitHub
- **Road ahead** © Jay Mantri