GIT PATCH STACK

HELPING YOU FREE YOUR MIND OF THE "FEATURE"

GIT BEST PRACTICES

THERE ARE THREE BEST PRACTICES AROUND GIT THAT HAVE BEEN A GIVEN IN THE COMMUNITY FOR QUITE SOME TIME.

- short lived branches/trunk based dev
- small pull requests
- buildable/testable commits

SHORT LIVED BRANCHES/TRUNK BASED DEV

Reduce heavy rework costs and promote earlier integration

SMALL PULL REQUESTS

Support valuable peer reviews

- easier to focus on the changes and provide valuable feedback without being overwhelmed

BUILDABLE & TESTABLE COMMITS

- support continuous integration
- support tooling that facilitate isolating bugs (git bisect, etc.)
- facilitates better architecture & code

ALL THESE ARE

THE PUSH & PULL

AN UNDERLYING TENSION IN TRYING TO ACHIEVE ALL THESE BEST PRACTICES


PONDERING
AND PONDERING
AND PONDERING
AND PONDERING...

INTENT IS *everything*

WHAT CAN WE LEARN ABOUT GIT


WHO CREATED IT?

WHY WAS IT CREATED?

As a replacement for a proprietary distributed source control system the Linux Kernel Team legally could no longer use.

KERNEL TEAM REVIEW WORKFLOW

- make small localized changes that are buildable & testable
- Have a Show Work principle with small building changes with good commit messages to provide context of intent.
- Take changes and create a patch file and email it to the appropriate mailing list for review
- People that receive patches have to maintain stacks of patches that get introduced into upstream over time. `quilt`

GIT PATCHES?

- Patches come from Unix `diff` & `patch`
- Git Patches are the same concept but include additional information like a message, authors, etc. `git format-patch`, `git apply`, `git send-email`, `git am`
- a commit is similar, but it has a references to parents in the tree and patches don’t
- Patches are floating diffs until applied

EMAIL SUCKS RIGHT?

Actually it isn't as bad as you might think. It has built in ability to support per line of code comments, etc.

It is a bit old school though and has a negative connotation in today's world.

Beyond that dealing with the overhead of generating patches, managing them locally, and emailing a mailing list for review seems less than ideal.

GIT PATCHES PROS

- independently reviewable
- bite size (Show your work)
- buildable & testable

GIT PATCHES CONS

- patch creation overhead
- email for code review
- patch management on consumption side

I WANT IT ALL THOUGH

- follow best practices
- bite size (Show your work)
- buildable & testable
- earlier code review of pieces of my overall effort
- no email
- good & current code review tools

THE PATCH STACK WORKFLOW

different mental model for doing local development


NO FEATURE BRANCHES
WORK DIRECTLY ON master

A CONCEPTUAL PATCH STACK

Think of `master` being a conceptual stack of patches (a.k.a. commits) on top of `origin/master`

REBASE ON PULL & WHENEVER

Instead of managing branches. You think of changes just being individual patches that you use `rebase` to squash, reorder, amend, edit, etc. as you continue to develop.

SIMPLE

CAN'T I JUST USE

```
pull.rebase =
    true
```

HOW DO YOU LIST YOUR STACK OF PATCHES?

git ps ls

HOW DO YOU REQUEST REVIEW OF A PATCH?

```
git ps rr <patch-index>
```

HOW DO YOU RE-REQUEST REVIEW OF A PATCH?

```
git ps rr <patch-index>
```

HOW DO YOU REBASE YOUR PATCH STACK?

git ps rebase

HOW DO YOU FETCH UPSTREAM ORIGIN/MASTER & REBASE YOUR PATCH STACK ON IT?

```
git ps pull
```

HOW DO YOU PUBLISH A PATCH UPSTREAM?

```
git ps pub
<patch-index>
```


START STACKING
PATCHES
AND

NO LONGER HAVE THE NEED FOR YOUR PR TO GET REVIEWED AND MERGED IN IMMEDIATELY

NEVER WORRY ABOUT DEPENDENT BRANCHES AGAIN

FOLLOW ALL THE BEST PRACTICES

- short lived branches/trunk based dev
  - small pull requests
  - buildable/testable commits
  - logical units of work
- good commit messages (what, why, and contextual details around how)

WORK SEAMLESSLY WITH OTHER GIT WORKFLOWS

(GIT-FLOW, ENVIRONMENT BASED BRANCHES, ETC.)

Flourish IN PERFECT HARMONY WITH OUTSIDE-IN DEVELOPMENT

STOP THINKING ABOUT BRANCHES AND

START THINKING ABOUT PATCHES

GET YOUR PATCH STACK ON AT HTTPS://GITHUB.COM/UPTECH/GIT-PS