A Rebasing Workflow

Emma Jane Hogbin Westby - Trillium Consultancy Ltd.
www.gitforteams.com
https://speakerdeck.com/emmajane/a-rebasing-workflow-for-git
@emmajanehw
A Rebasing Workflow

Emma Jane Hogbin Westby - Trillium Consultancy Ltd.
www.gitforteams.com
https://speakerdeck.com/emmajane/a-rebasing-workflow-for-git
@emmajanehw
A Rebasing Workflow

Emma Jane Hogbin Westby - Trillium Consultancy Ltd.

www.gitforteams.com

https://speakerdeck.com/emmajane/a-rebasing-workflow-for-git

@emmajanehw
Agenda

• the tl;dl
• Rant: why I don't like rebasing
• Rave: when / why I use rebase
• Wrap-up
• Q&A
the tl;dl

too long; didn't listen
Why should I rebase?

• Because it makes your commit history easier to read, and manipulate.

• “Because Joe told me to.”
Rebasing is like a malleable sports re-play.

**Step 1.** Rewind the history of your branch.

**Step 2.** Apply new commits from an outside source.

**Step 3.** Then re-apply your work.
Rebasing is like a malleable sports re-play.

**Step 1.** Rewind the history of your branch.

**Step 2.** Apply new commits from an outside source.

**Step 3.** Then re-apply your work.
Rebasing is like a malleable sports re-play.

**Step 1.** Rewind the history of your branch.

**Step 2.** Apply new commits from an outside source.

**Step 3.** Then re-apply your work.
Rebasing is like a malleable sports re-play.

**Step 1.** Rewind the history of your branch.

**Step 2.** Apply new commits from an outside source.

**Step 3.** Then re-apply your work.
When should I rebase?
When should I rebase?

• Before sharing proposed work.
When should I rebase?

• Before sharing proposed work.
When should I rebase?

• Before sharing proposed work.
• To update your local working branch.
When should I rebase?

- Before sharing proposed work.
- To update your local working branch.
When should I rebase?

- Before sharing proposed work.
- To update your local working branch.
- To split a commit into two.
When should I rebase?

- Before sharing proposed work.
- To update your local working branch.
- To split a commit into two.
When should I rebase?

- Before sharing proposed work.
- To update your local working branch.
- To split a commit into two.

$ git rebase -i edit
When should I rebase?

- Before sharing proposed work.
- To update your local working branch.
- To split a commit into two.

$ git rebase -i
eedit
rebase -i
squash
When should I rebase?

- Before sharing proposed work.
- To update your local working branch.
- To split a commit into two.

$ git rebase -i edit
rebase -i squash
rebase <branch_name>
How do I rebase?

• Squash a bunch of tiny commits into one:
  
  $ git rebase --interactive

• Bring your local, working branch up-to-date

  $ git pull master

  $ git checkout my_local_branch

  $ git rebase master
Groundhog Day Conflict?!  

• Resolve the conflict and then *immediately* save your recorded solution with:

  
  $ git rerere
  

• Proactively turn on REuse REcorded REsolution:

  
  $ git config --global rerere.enabled true

• Bad merge? Git will remember the wrong resolution. Immediately forget how you resolved it with:

  
  $ git rerere forget
How do I stop rebasing?

- If things go right, it will stop on its own.
- If there are tears, you can abort.

$ git rebase --abort
let the rants begin!
Rant

Why I don’t like rebasing.
Historical Revisionism
Merge pull request #15 from tsphethean/atom-date
- Make ATOM feed updated date RFC3339 valid

Copyright date

Merge pull request #14 from mavimo/patch-2
- Add atom.xml definition in pages

Merge pull request #12 from resmo/patch-1
- Added sitemap.xml

First pass at an Atom file. Thanks to @xejgi closes #4

Updated depts.

No need for layout anymore.

Merge pull request #11 from stevector/master--implements-spelling
- Spelling fix in 2013-02-04-highlight.md
- Merge pull request #10 from stevector/master--fixing-typo-in-read
- Spelling fix in README
- Merge pull request #8 from kenjis/fix_url
- Fix category and tag names in URL which are not url encoded
- Merge pull request #9 from kenjis/fix._typo
- Fix typo in title attr

Post navigation.

Merge pull request #7 from chiarand/feature/add-fenced-code-blocks--
- Add missing link target in sample post
- Add a fenced code block example in sample posts

Migrate fully to sculpin.phar

More robust errors.

Download phar; no more installer.

No more assets:install

Additional flag

Migrate to phar.

Merge pull request #6 from Pawka/master
- Throw help message if incorrect parameter is provided.
- Fixed forced --dry-run for s3 deployment.
- Load s3.conf file on publish script.

Updated dependencies.

Merge pull request #5 from beryllium/s3-publish
- Removing assets:install reference
- Fixing indentation
e4d406f Merge pull request #12 from resmo/master

f48e672 added sitemap.xml

ac6b40e First pass at an

edcd486 Updated deps.

05af892 No need for layout anymore.

eb85b12 Merge pull request #11 from stevevector/master -- implements spelling

086b01e Spelling fix in 2013-02-04-highlight.md

58e67f4 Spelling fix in README

8ebb37d Merge pull request #8 from kenjjs/fix_url

107f62d Fix category and tag names in URL which are not URL encoded

0399081 Merge pull request #9 from kenjjs/fix_typo

a345de0 Fix typo in title attr

fedb68a Post navigation.

ff584da Merge pull request #7 from cigarand/feature/add-fenced-code-blocks-exam...
Revision numbers for commits with:

- 6598: (vila) Bzr config should save the changes explicitly when added.
- 6597: (richard-wilbur) Jelmer: Don't pass blob to file.write()
- 6596: (richard-wilbur) Also honor $XDG_CONFIG_HOME specification on windows.
- 6591.2.1: Also honor $XDG_CONFIG_HOME specification on windows.
- 6595: (vila) Fix command line override handling for access rights.
- 6589.3.3: Fix typo.
- 6589.3.2: Fix typo.
- 6589.3.1: bug #1249732 Fix command line override handling for access rights.
- 6594: (richard-wilbur) Fix bug LP: #1123460, download.
- 6593: (vila) Fix python-2.7.6 test failures. (Vincent Lamy)
- 6592: (vila) Use LooseVersion from distutils to check.
- 6590: (vila) Fix test failure on recent trusty kernels (Ubuntu 14.04)
- 6589: (vila) Stricter on formats for .py.
- 6588: (vila) Make .netrc 0600 in tests so python-2.7.5-8's can read.
Rave

Why I love the effects of rebasing.
Making Whole Thoughts
Resources for Good Commit Messages

gitforteams.com/resources/commit-granularity.html
Converting Conversations to Conclusions
Patches vs. Pull Requests
Merge vs. Rebase
Often drawing the history graph BEFORE running a command will help you choose the right command.
two branches before merging

after merge with fast forward

after true merge (no fast forwarding)
Choosing a Strategy

- Pull = Fetch + Merge
- Merge -> no fast forward (“true merge”)
- Merge -> with fast forward
- Merge with --squash -> all into one
- Rebase -> rewind + replay
- Cherry Pick -> copy and paste commits
In Summary

• The benefits of rebasing are most apparent to projects with multiple branches and multiple committers.

• Rebasing allows you to reshape commit history so that you are storing conclusions, not conversations.

• Rebasing can be used in place of merge to update a branch and results in a simplified graph of your repository history.

• Rebasing can be used interactively to reshape a series of commits.
www.gitforteams.com

Emma Jane Hogbin Westby
https://speakerdeck.com/emmajane/a-rebasing-workflow-for-git
@emmajanehw