A Rebasing Workflow

Emma Jane Hogbin Westby

www.gitforteams.com
@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.
When should I rebase?

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

```bash
$ git rebase -i edit
$ git rebase -i squash
$ git 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
git log --graph --oneline

@emmajanehw
revision numbers for commits with branch depth, not commit hashes

branches shown as summaries by default, click to open and show individual commits
Revision numbers for commits with branches shown as such.

Revision: 6596 revid: pqm@pqm.ubuntu.com - 20140507222027 -

@emmajanehw
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
Drawing the history graph before running a command will help you choose the right command.
before merge with no fast forward

updating a feature branch with merge, no fast forward

closing a feature branch with merge, no fast forward
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
@emmajanehw