An RDKit-centric intro to git and github

Gregory Landrum[1], Nadine Schneider[2]
[1] KNIME.com, T5 Informatics
[2] Novartis Institutes for BioMedical Research, Basel

2016 RDKit UGM

26 October 2016

Goal of this talk

- Showing one way of using Git to contribute to the RDKit

- Most important thing about Git: It is very flexible, everyone has to find her/his own way of using it and feeling comfortable

- What this talk isn’t meant to be
  - An in-depth general Git tutorial
  - A comparison of SVN or similar and Git

  →there is much of this in the web, books, etc.

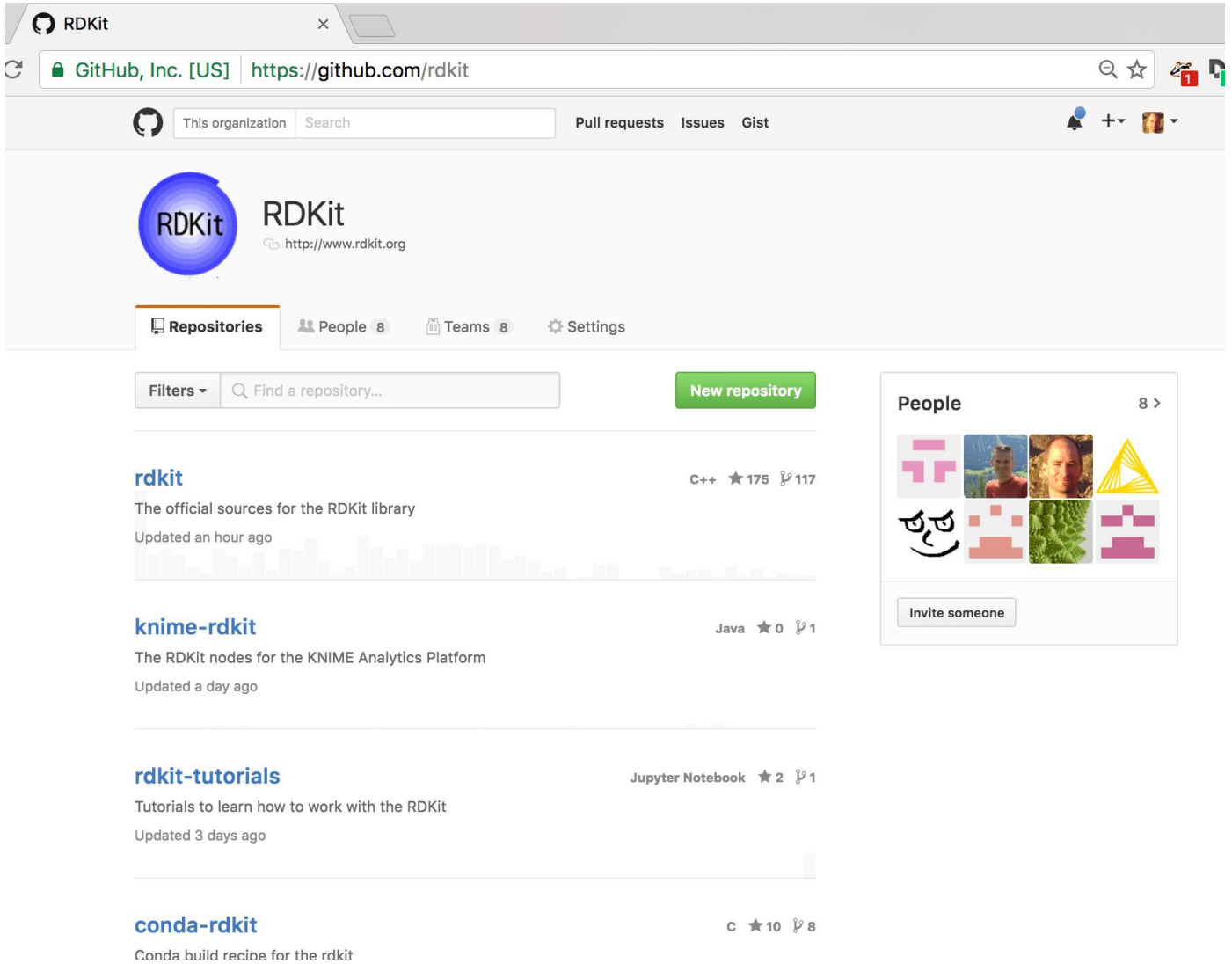
Starting point: what's the difference?

Git: an open-source distributed version control system

Github: a website that hosts git repositories
(among many other things)

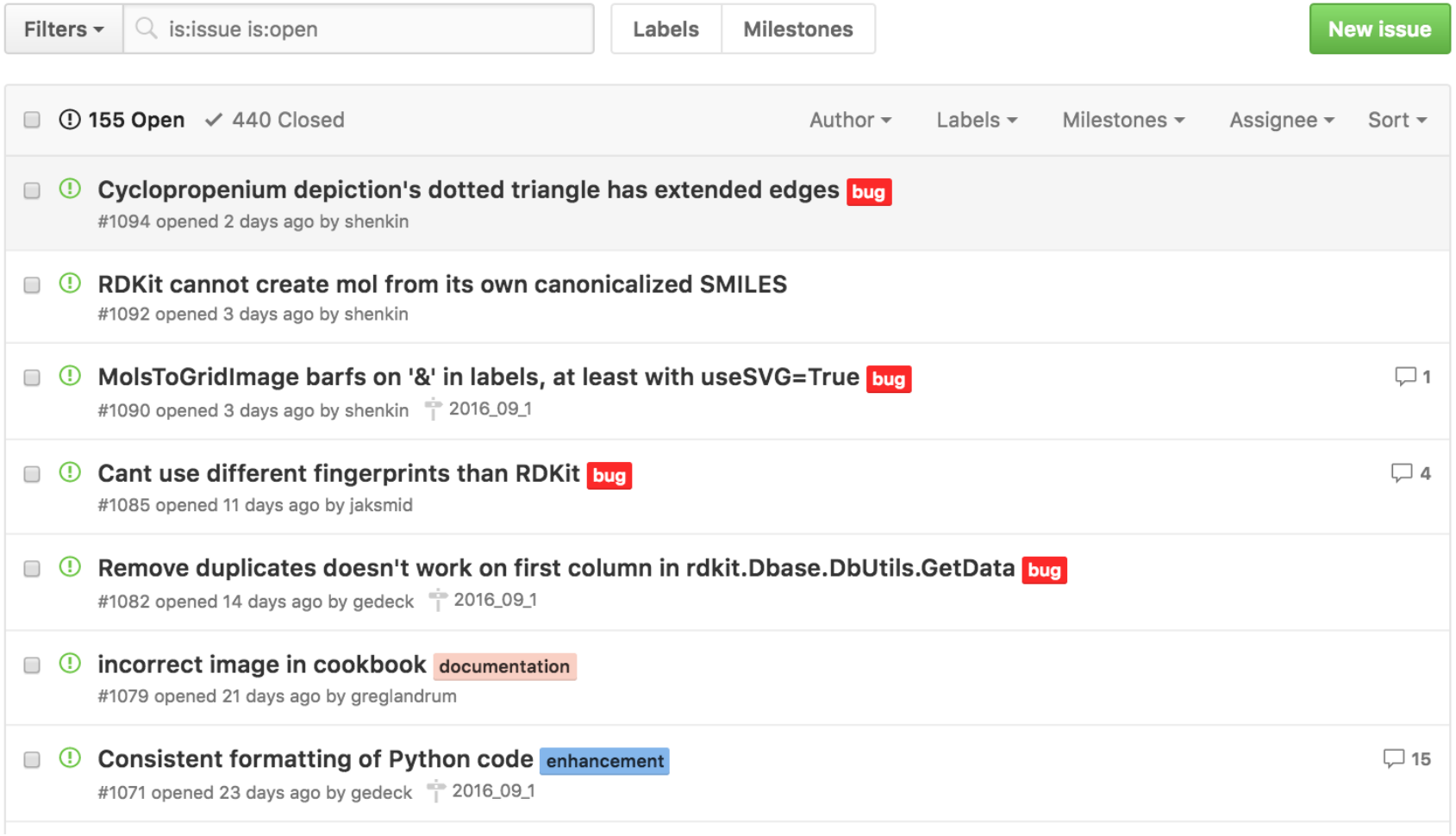
A tour of the RDKit github presence

The organization



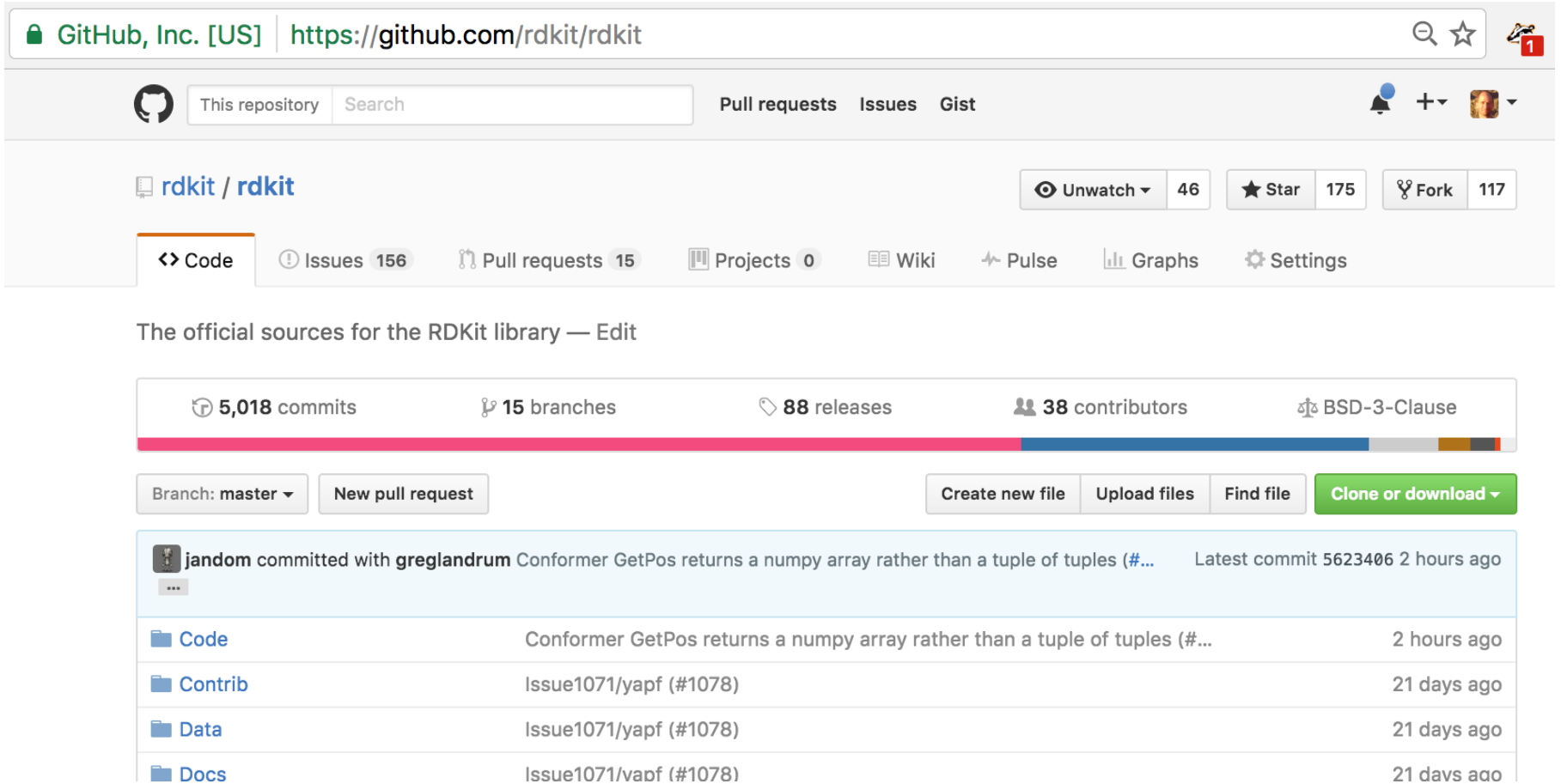
A tour of the RDKit github presence

Issues

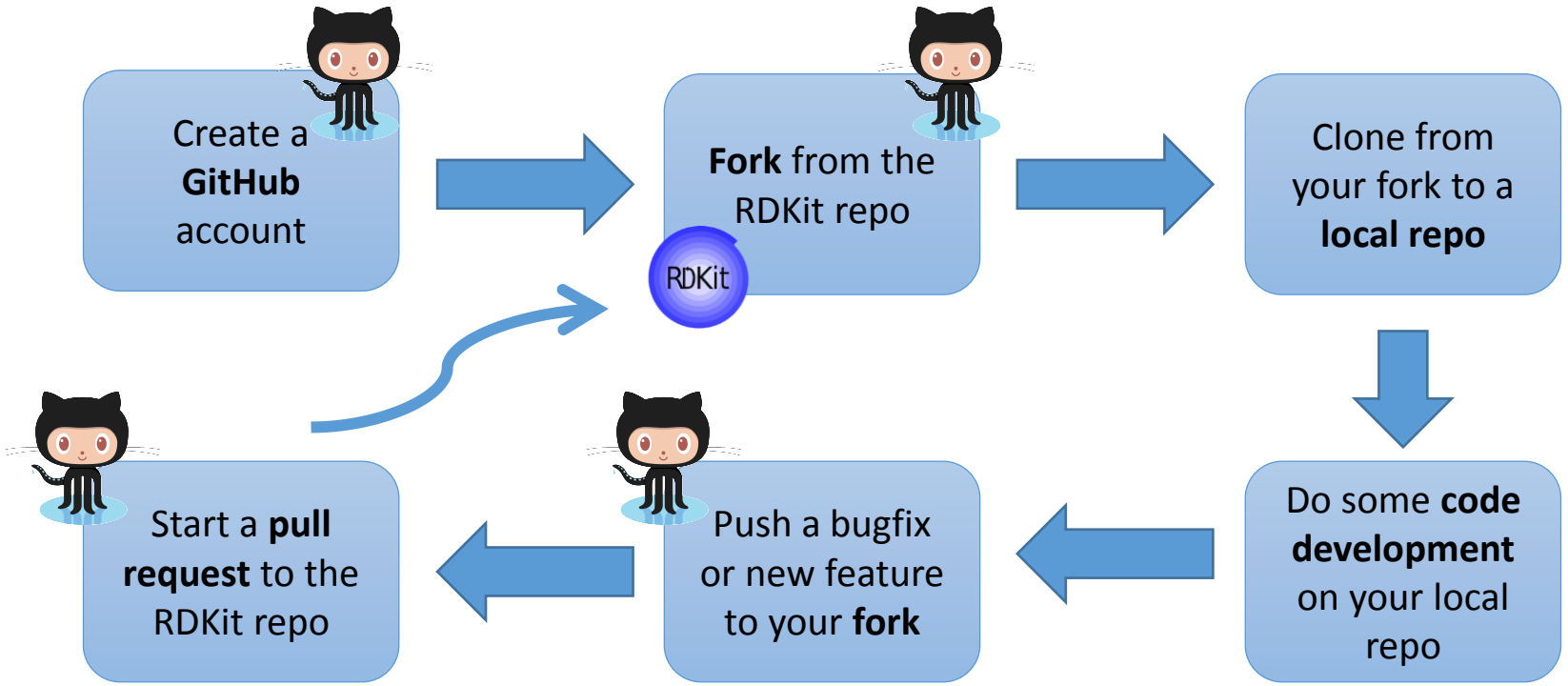


A tour of the RDKit github presence

The repo



The overall workflow



The ~~overall~~ realistic workflow

One cool thing about Git is that you can commit as much as you like in your **local repo**, no one will ever see it if you use **git squash** before pushing to your fork

Create a **GitHub** account

Fork from the RDKit repo

Clone from your fork to a **local repo**

Do some **code development** on your local repo

Google for "**git reset --hard**"

Google for "**what is git rebase**"

Google for "**and what the **** is git rebase -i**"

Push a bugfix or new feature to your **fork**

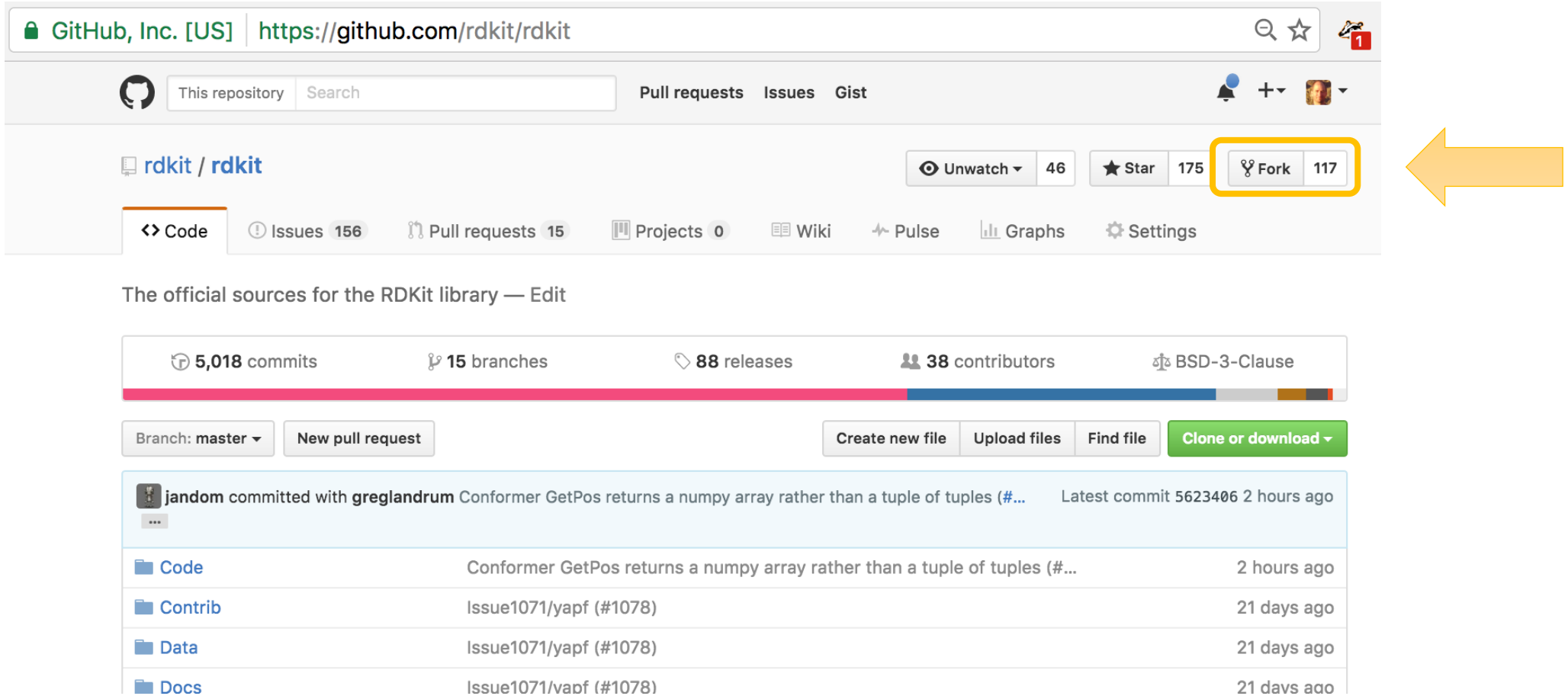
Start a **pull request** to the RDKit repo

Hope that all quality criteria are met and Greg is happy with it ;)

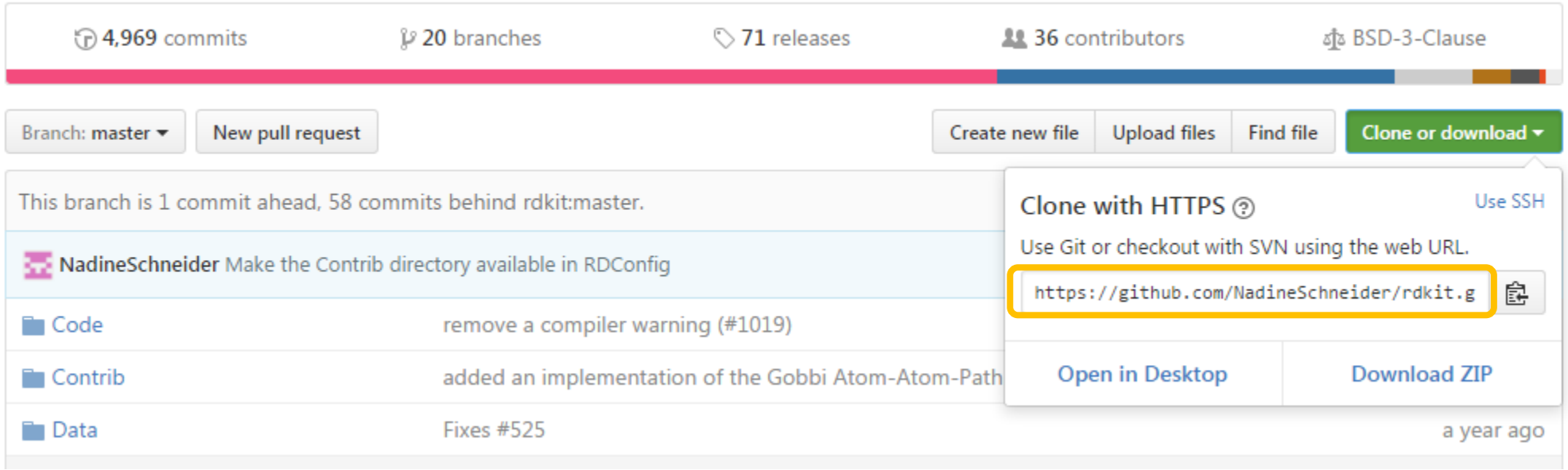
https://xkcd.com/1296/

A tour of the RDKit github presence

The repo



Clone from your fork to a local repo



Copy this to your console:

```
git clone https://github.com/YOUR_USERNAME/rdkit.git
```

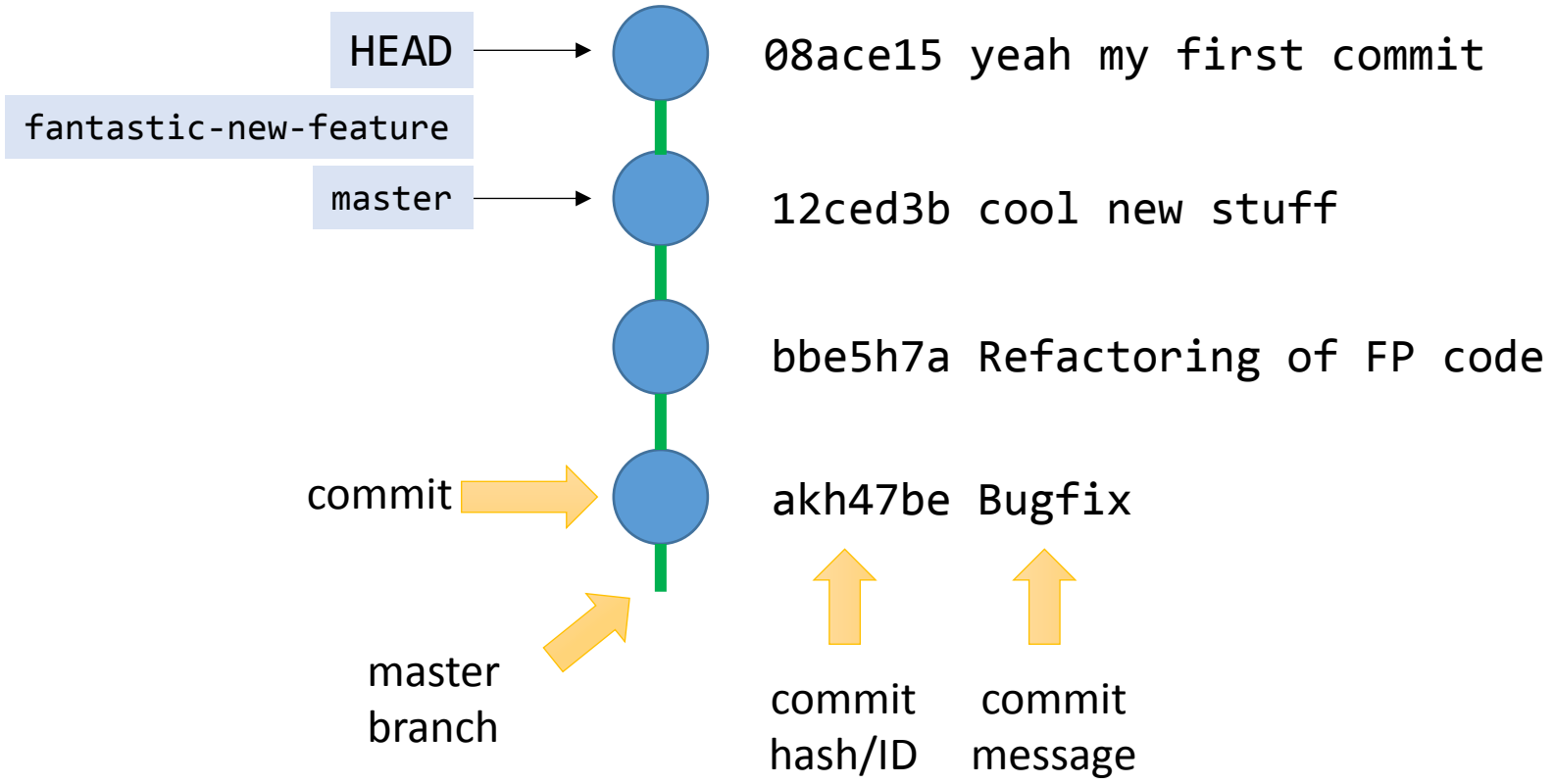
Now your contribution to RDKit can begin…

- Let's first checkout a new branch where we develop a new feature:

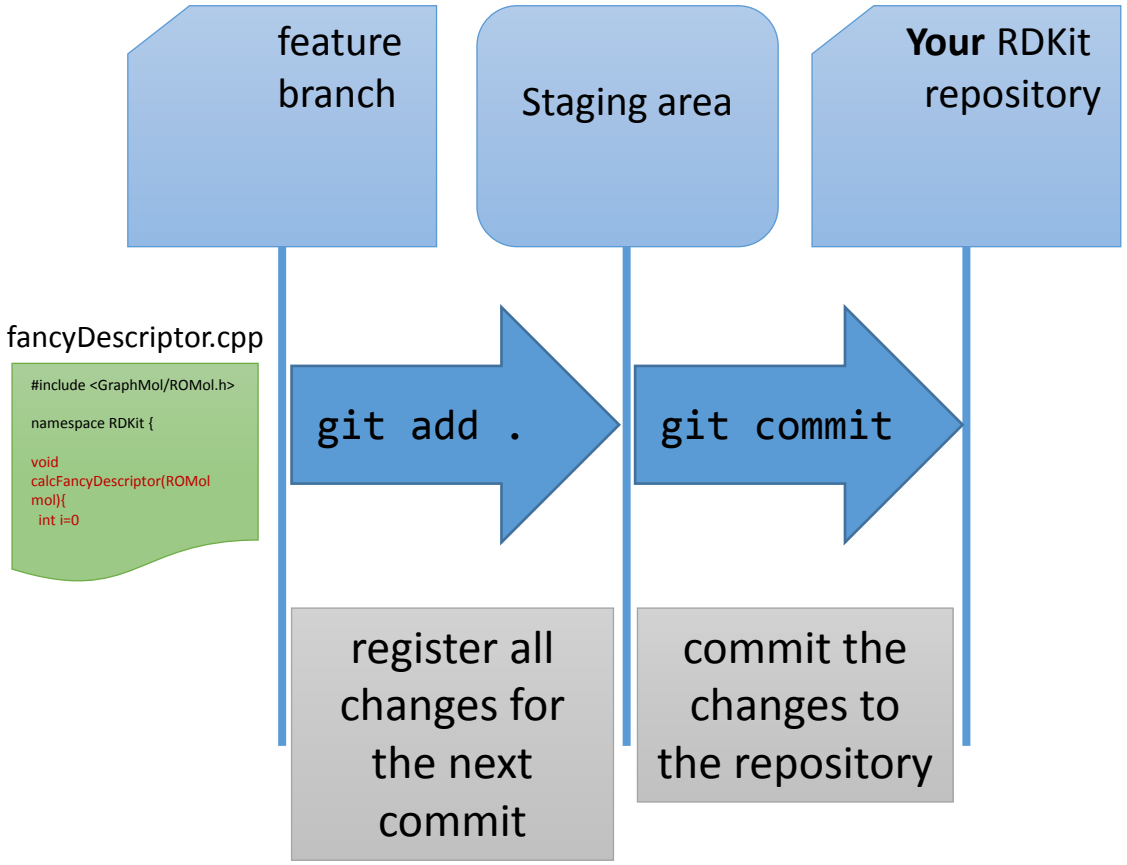
```
git checkout –b "fantastic-new-feature"
```

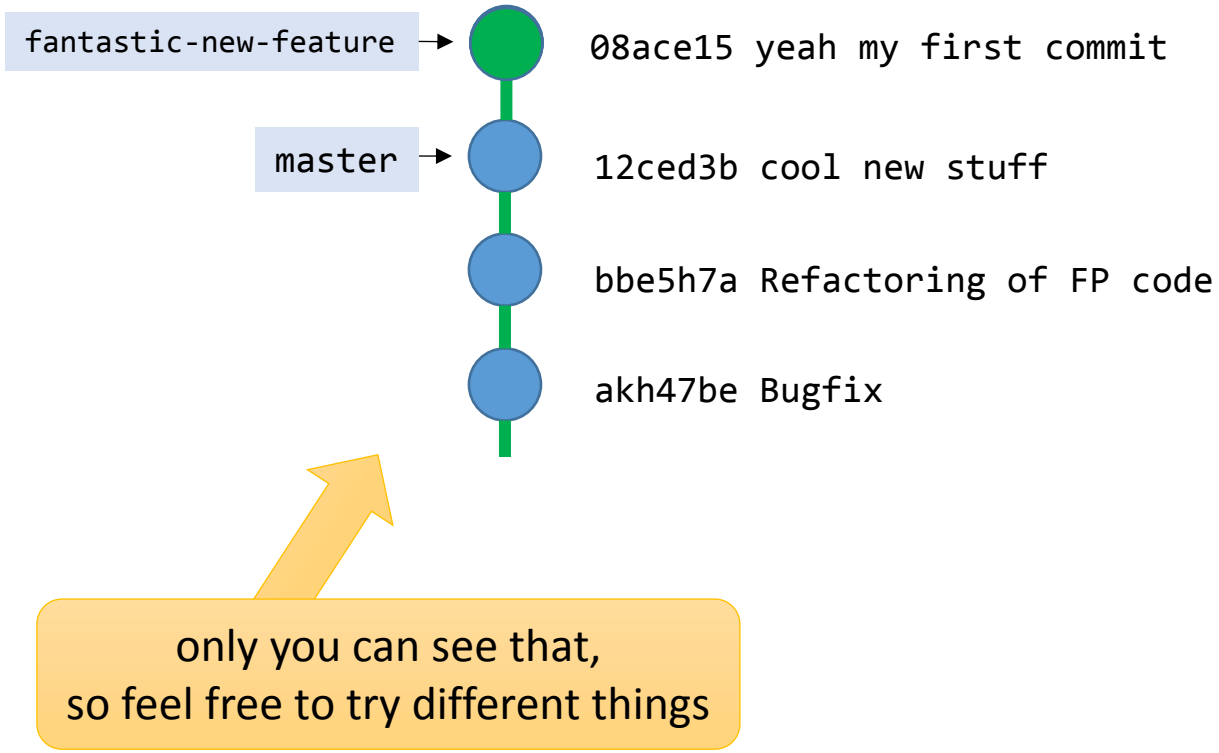
After a few lines of code, let's do a commit

```
git add .
git commit –m “yeah my first commit”
```

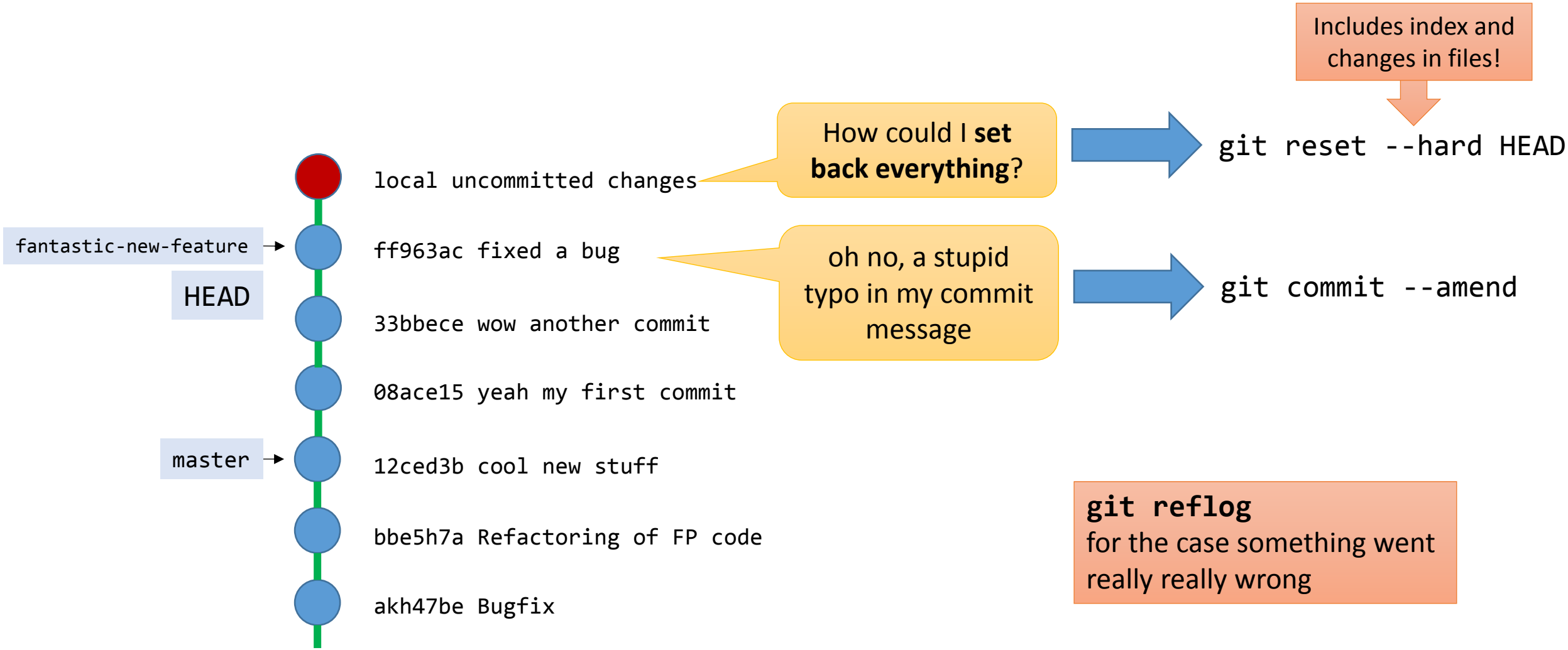


What happens in the back…





Let’s continue developing the new feature



Feature finished! Let’s put it back on the master

- But first, we have to update our master (RDKit never sleeps ☺)
- For this add a remote to your repository:

```
git remote -v
```

```
origin  https://github.com/YOUR_USERNAME/rdkit.git (fetch)
origin  https://github.com/YOUR_USERNAME/rdkit.git (push)
```

```
git remote add upstream https://github.com/rdkit/rdkit.git
git remote -v
```

```
origin  https://github.com/YOUR_USERNAME/rdkit.git (fetch)
origin  https://github.com/YOUR_USERNAME/rdkit.git (push)
upstream https://github.com/rdkit/rdkit.git (fetch)
upstream https://github.com/rdkit/rdkit.git (push)
```

Name of your remote. Choose whatever you like!

Feature finished! Let's put it back on the master

- Now let's get the update from the original RDKit repo

```
git fetch upstream
```

```
remote: Counting objects: 75, done.
remote: Compressing objects: 100% (53/53), done.
remote: Total 62 (delta 27), reused 44 (delta 9)
Unpacking objects: 100% (62/62), done.
From https://github.com/rdkit/rdkit
 * [new branch]      refactor_FPs  -> upstream/refactor_FPs
   12ced3b..eeffa12  master    -> upstream/master
```

Yeah we have an update!

- Before we continue, let's see where we are:

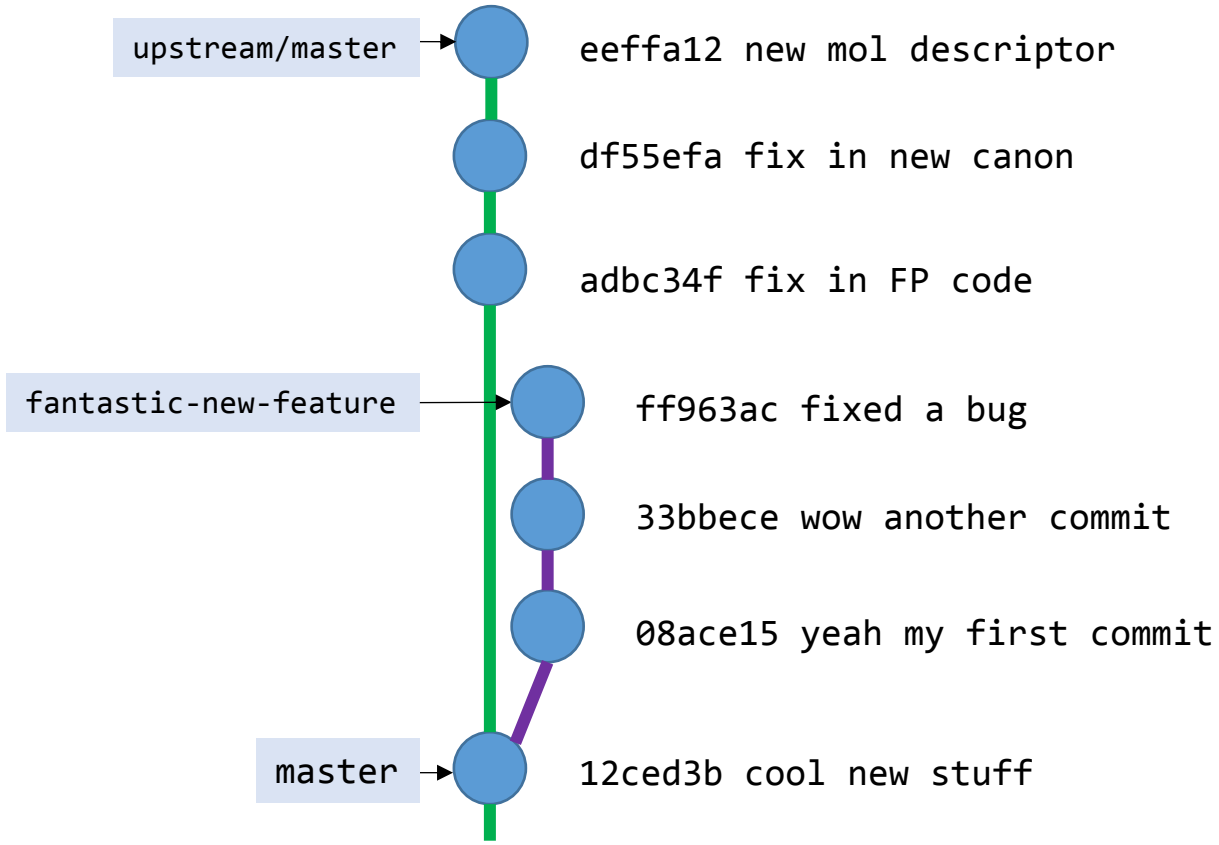
```
git branch
```

```
  master
* fantastic-new-feature
```

We are still on our feature branch

Feature finished! Let’s put it back on the master

How the repository looks at the moment



What we can do next:

1. `git merge upstream/master`

or

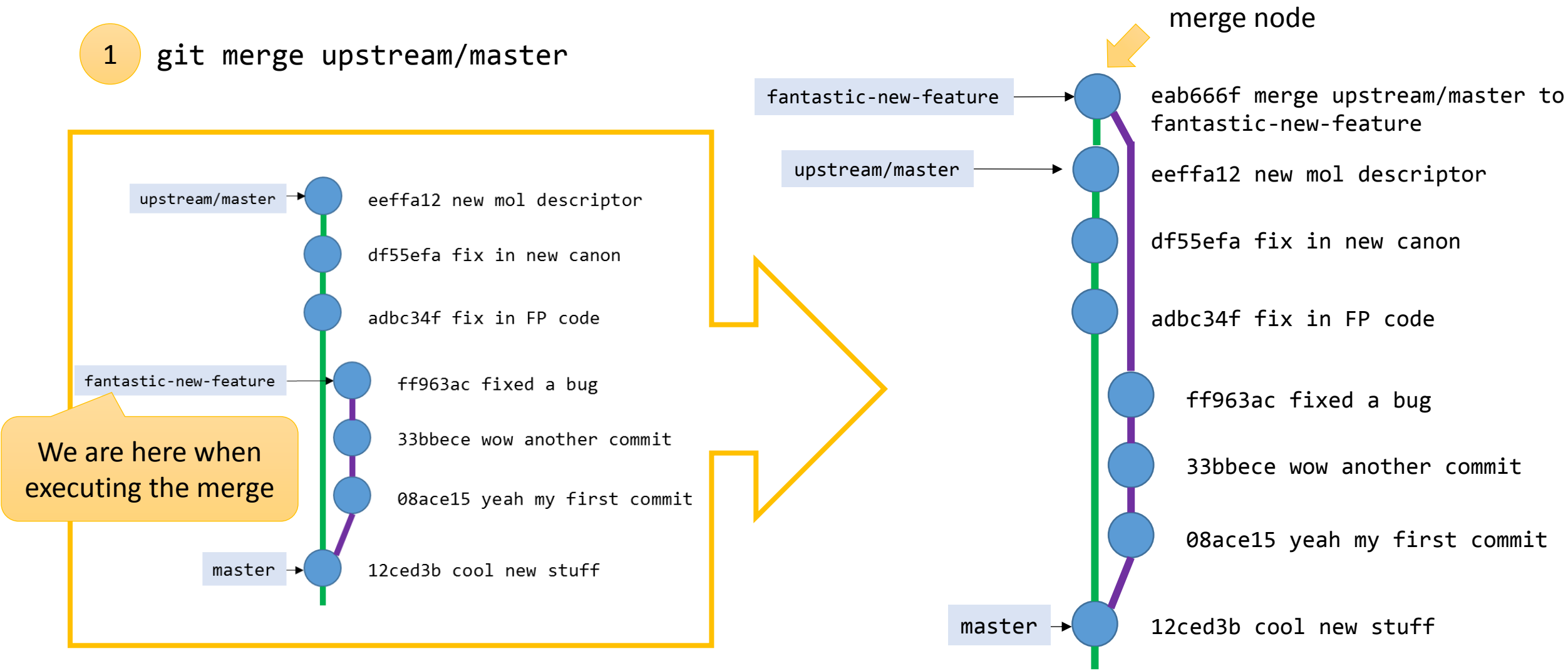
2. `git rebase upstream/master`

or

3.
```
git checkout –b new-master upstream/master
git cherry-pick 08ace15
git cherry-pick 33bbece
git cherry-pick ff963ac
```

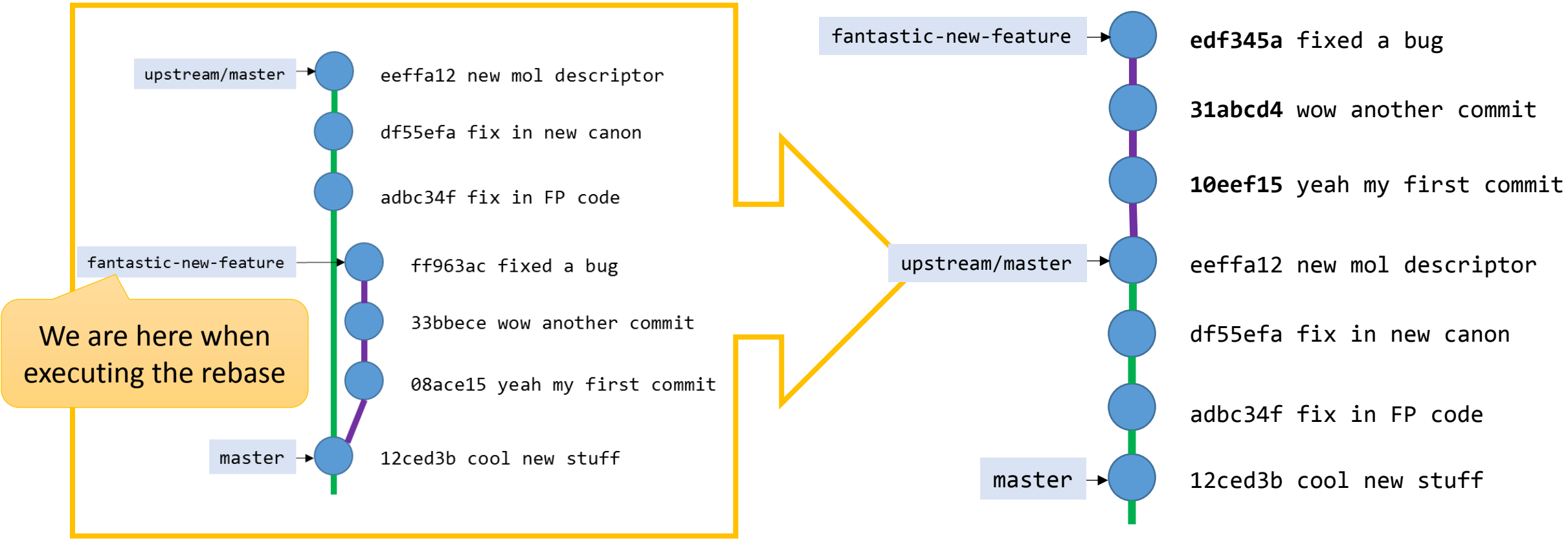
Feature finished! Let's put it back on the master

1 `git merge upstream/master`

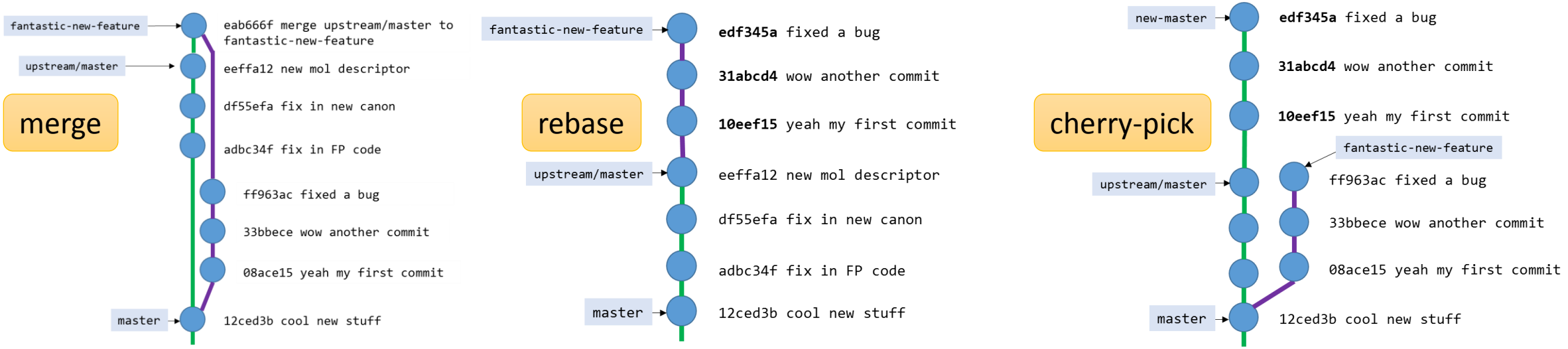


Feature finished! Let’s put it back on the master

2 `git rebase upstream/master`



We are ready to push our new feature!



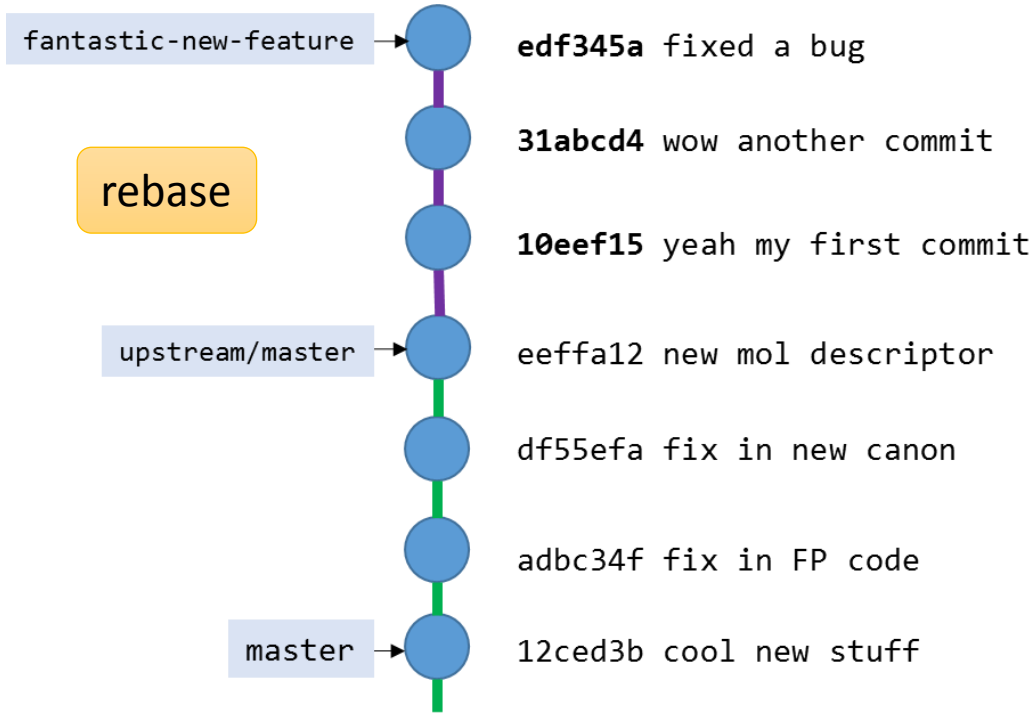
- Independent of your workflow the branch is now up to date and ready for a push on your github repo so that a pull request can be started

```
git push origin fantastic-new-feature:new-feature-for-rdkit
(new-master)
```

remote | local branch name | remote branch name

Done!!
Your changes are online. You can start a pull request!

One last word about squashing...



Would you really like to share all of your “experimental commits and messages” with the rest of the world?

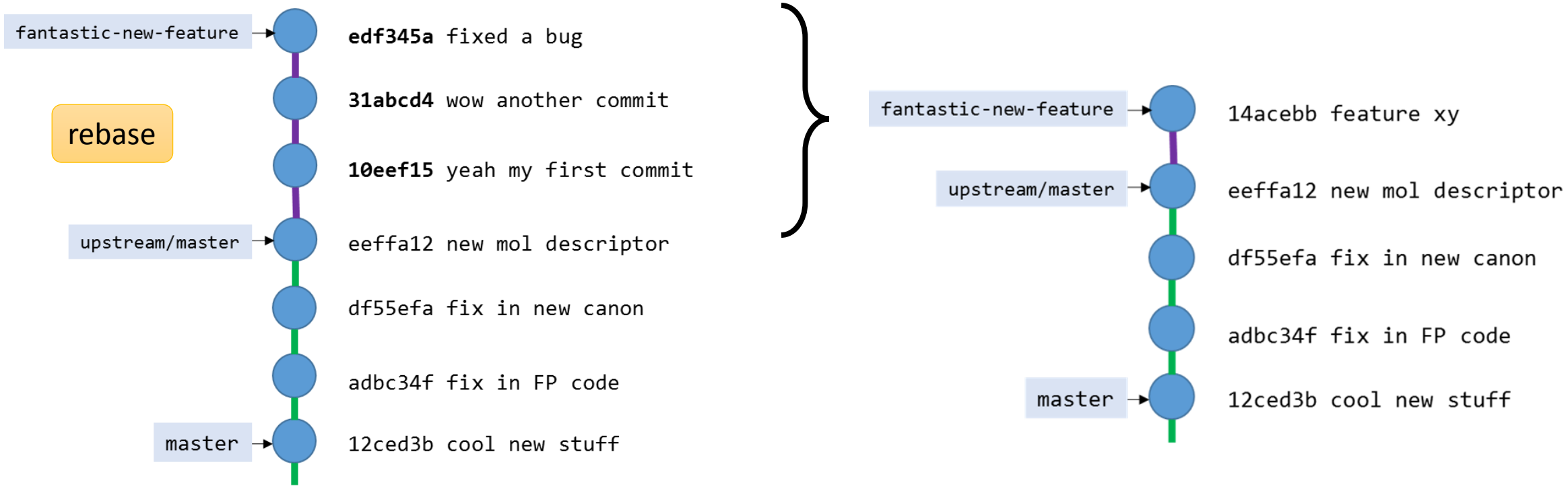
NOPE!

Let’s try

```
git rebase -i eeffa12
```

Allows you to consolidate single commits to one meaningful commit

One last word about squashing...give it a try!

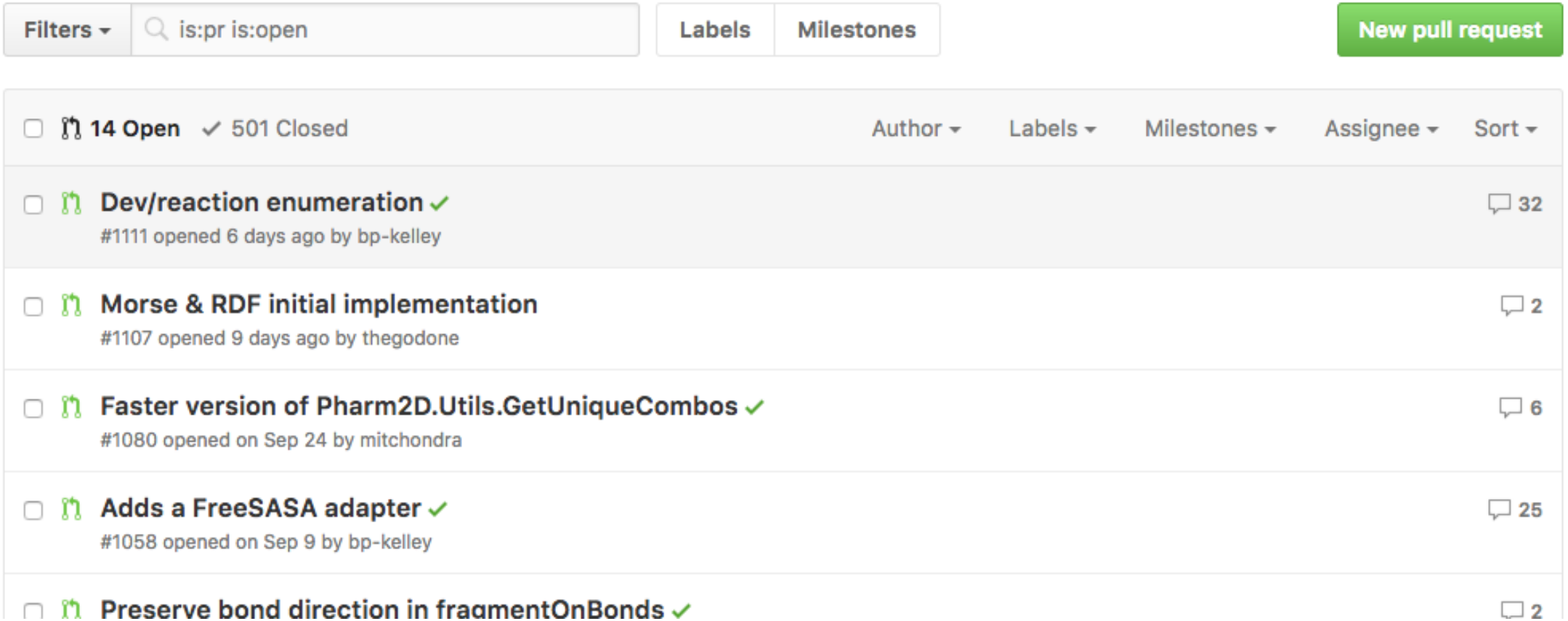


Result squashing using

```
git rebase -i eeffa12
```

A tour of the RDKit github presence

Pull requests

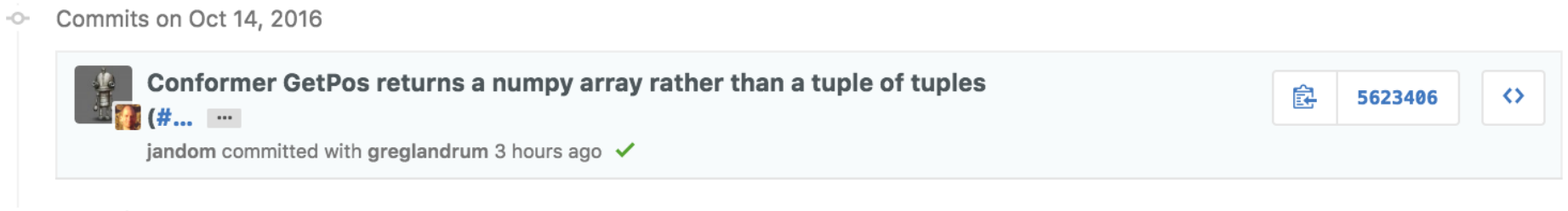


Reviews of pull requests

- We review all pull requests (PRs) for quality, compliance with the coding standards (such as they are), and to ensure that they make sense
- Reviews are open (in github) and anyone is welcome to participate
- We are starting to use the new (as of Sept 2016) formal review mechanism in github, but haven’t completely adopted that yet
- After the review, the PR is either merged by a maintainer or closed without merging it.

Why can’t I just email you my changes?

- Well… you can, if that’s ***really*** necessary
- But then *I* need to create the branch and do the PR
- A real benefit of using the standard route is that you get credit for the contribution:



(this will also show up correctly in the release notes)

Acknowledgements

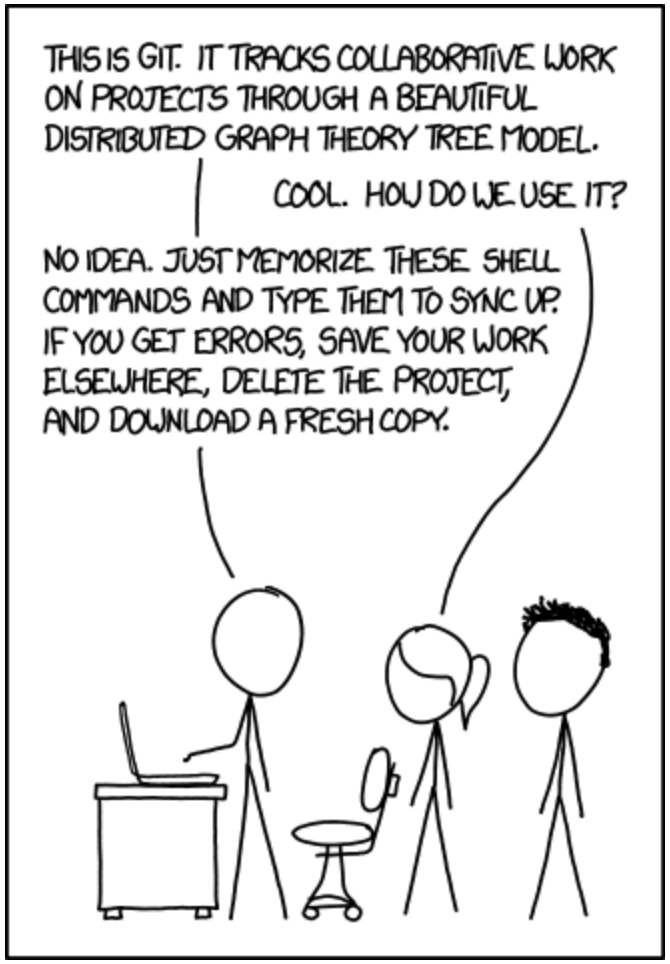
Brian Kelley

Nikolaus Stiefl

Nikolas Fechner

Nico Pulver

Anna Pelliccioli



http://xkcd.com/1597/

Any questions?

Still perplexed?

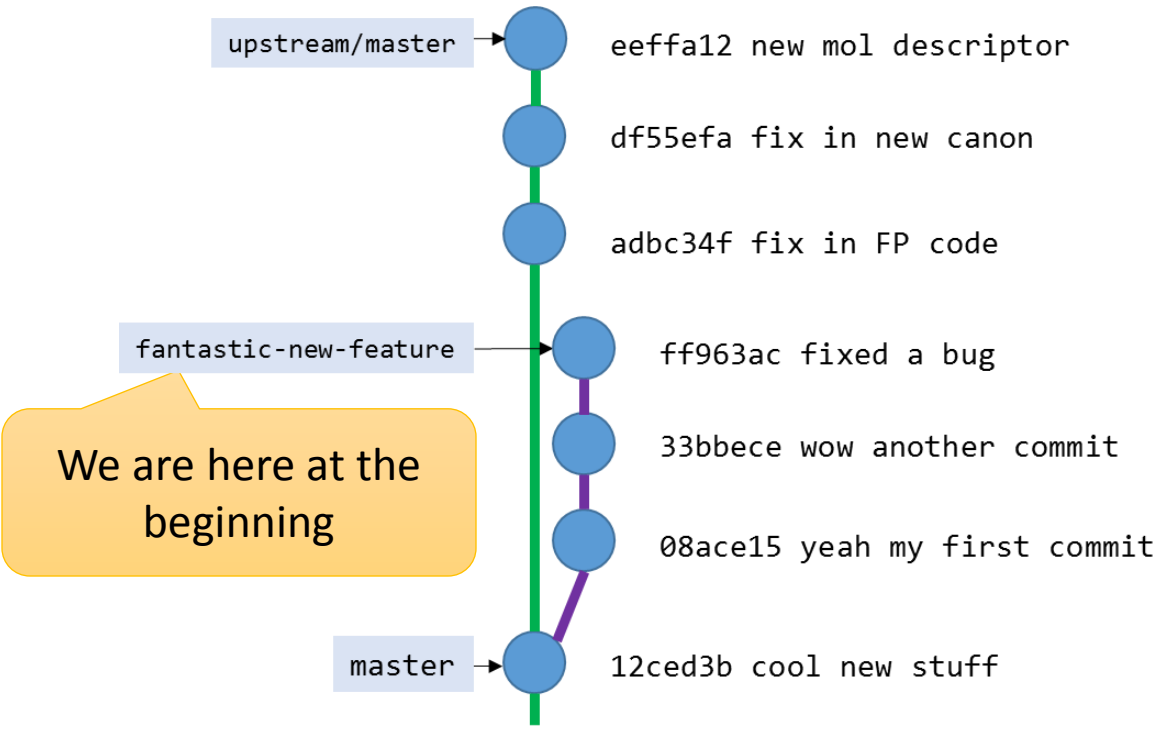
Some good cheat sheets:

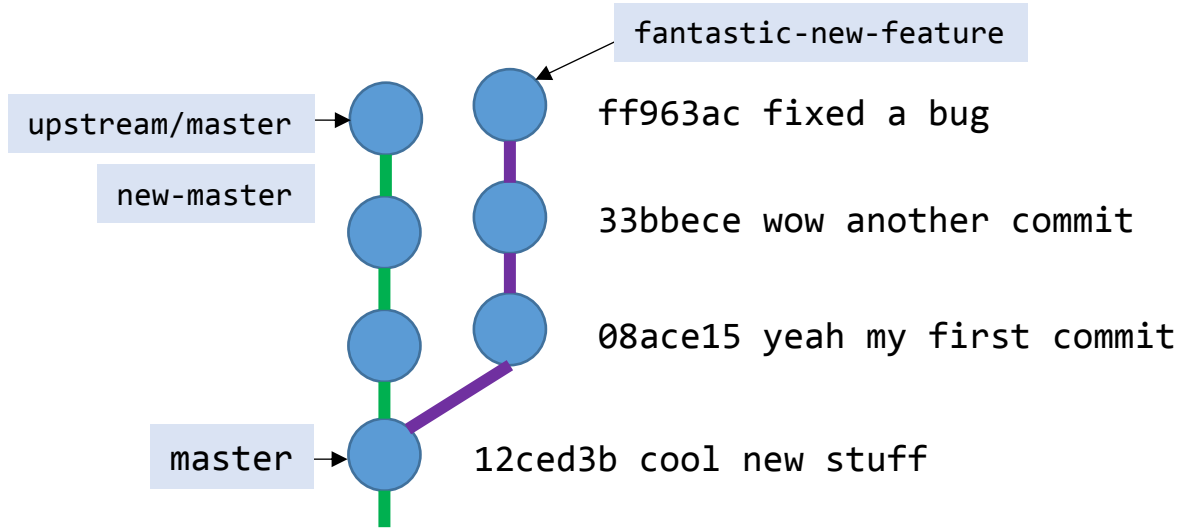
https://services.github.com/kit/downloads/github-git-cheat-sheet.pdf

https://zeroturnaround.com/wp-content/uploads/2016/05/Git-Cheat-Sheet-by-RebelLabs.png

Feature finished! Let's put it back on the master

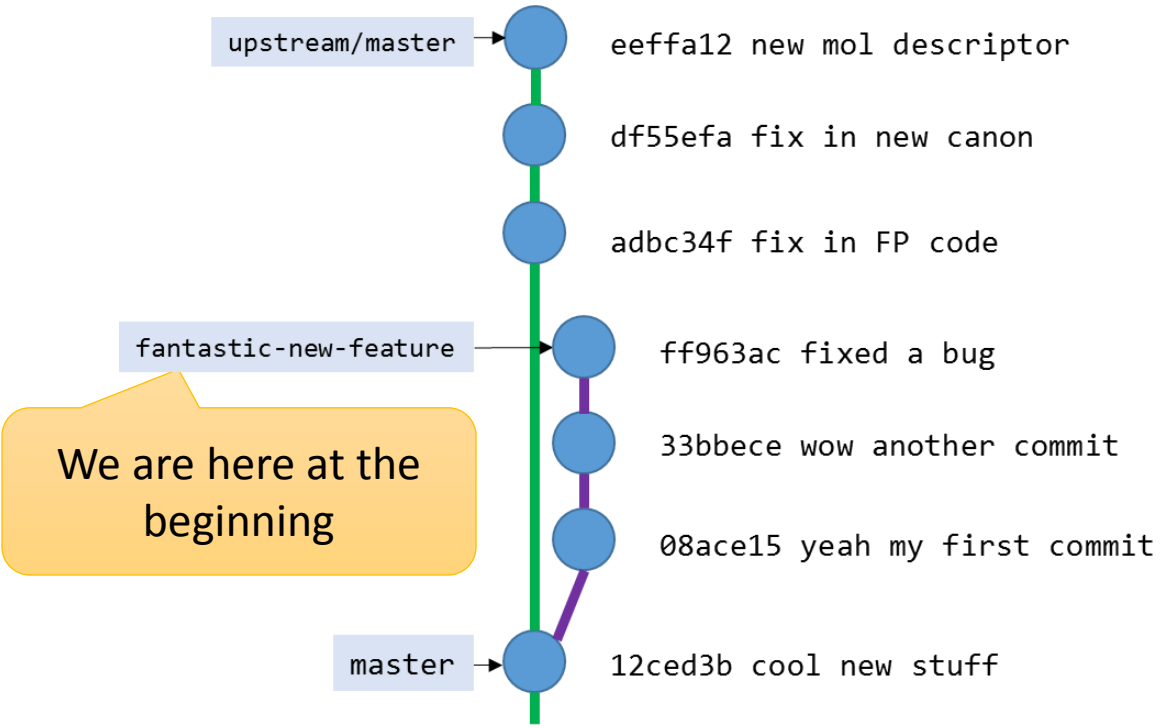
3 `git checkout -b new-master upstream/master`

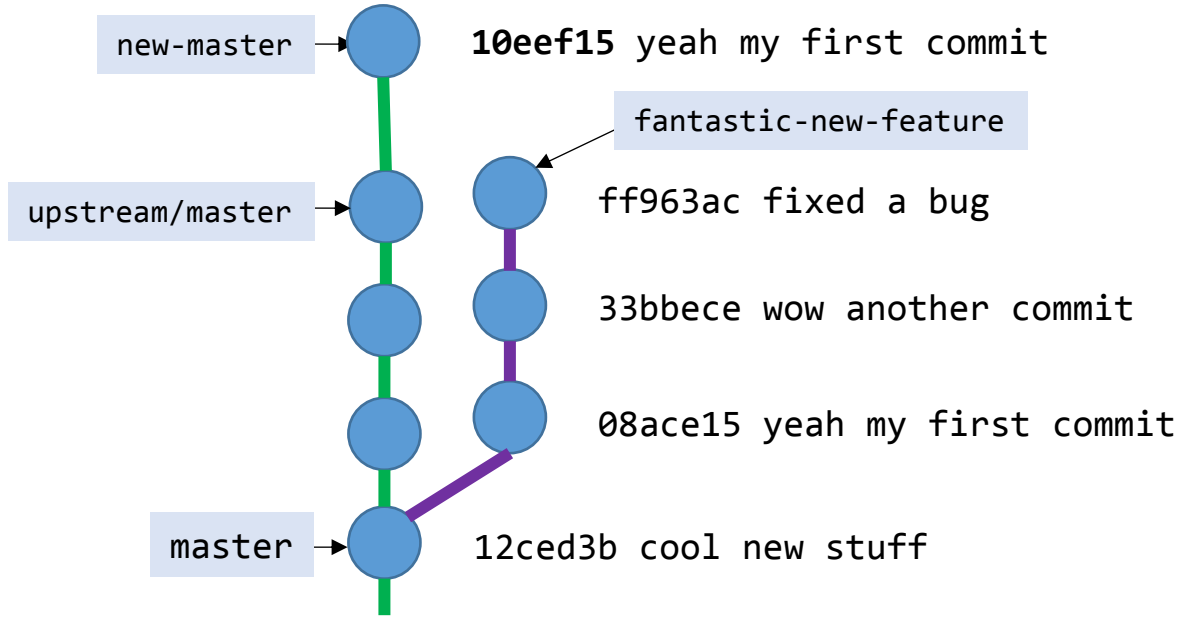




Feature finished! Let's put it back on the master

3 `git checkout –b new-master upstream/master`
`git cherry-pick 08ace15`

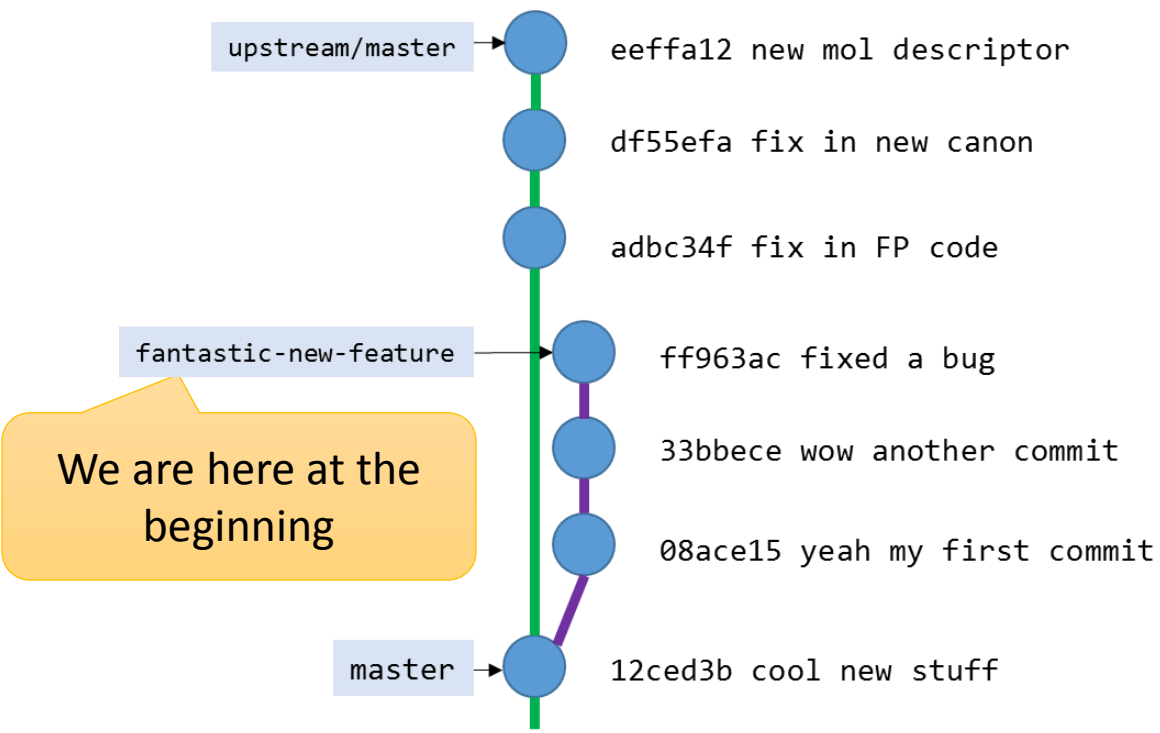


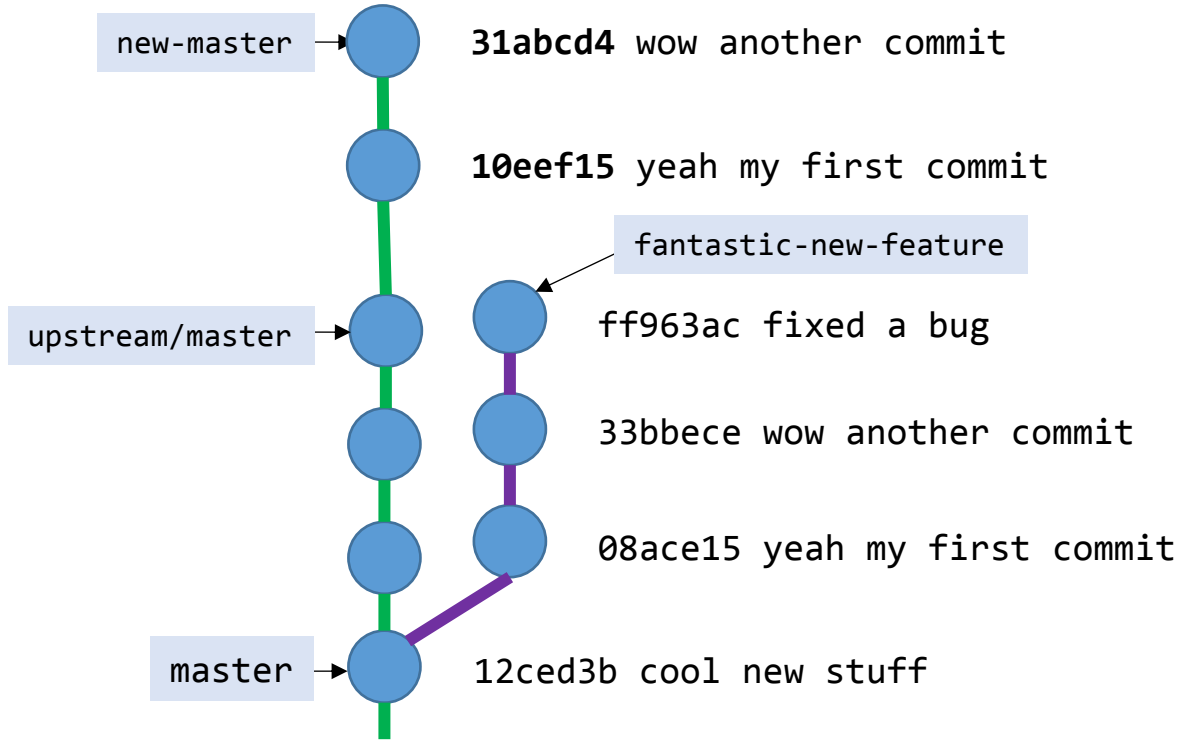


Feature finished! Let's put it back on the master

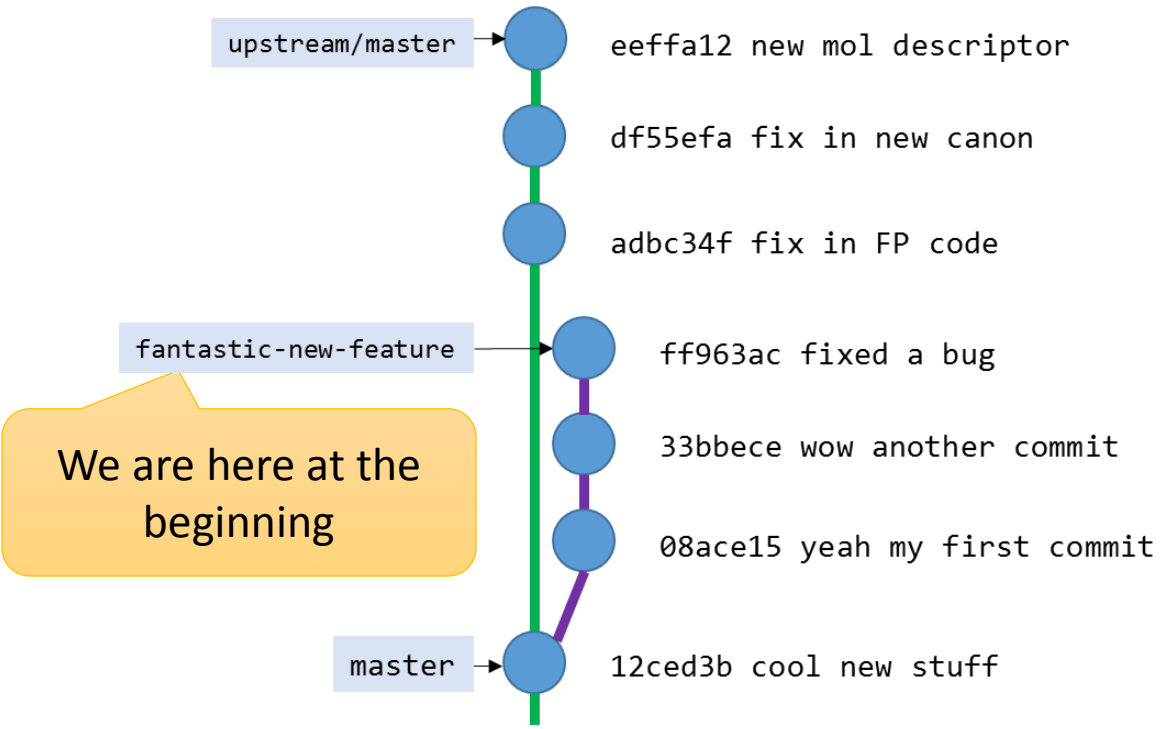
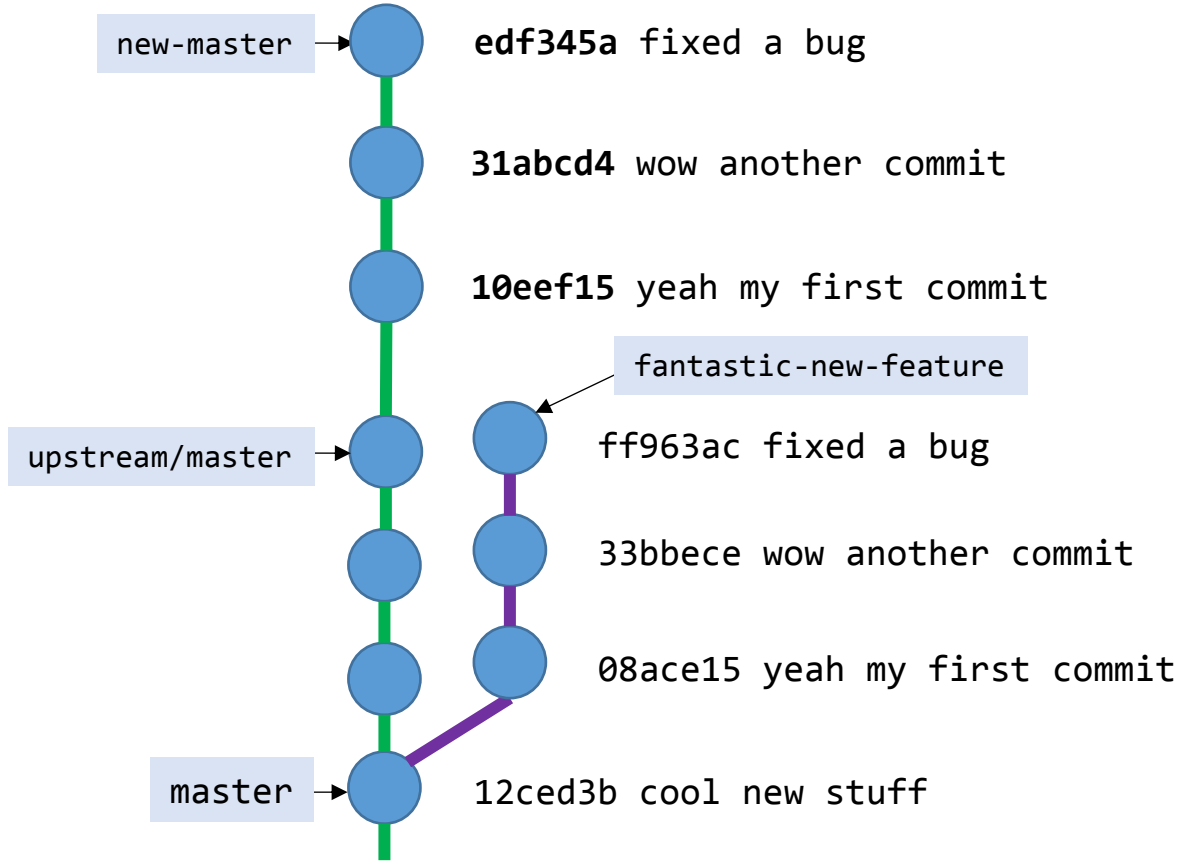
3

```
git checkout –b new-master upstream/master
git cherry-pick 08ace15
git cherry-pick 33bbece
```





Feature finished! Let's put it back on the master

3

```
git checkout –b new-master upstream/master
git cherry-pick 08ace15
git cherry-pick 33bbece
git cherry-pick ff963ac
```