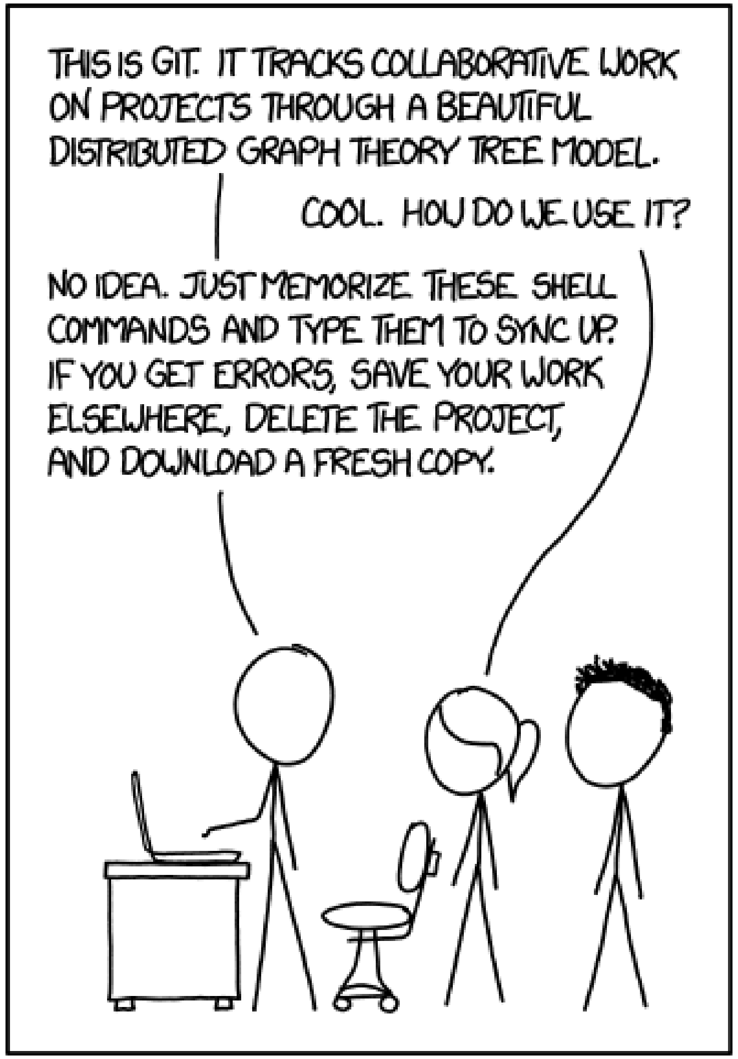
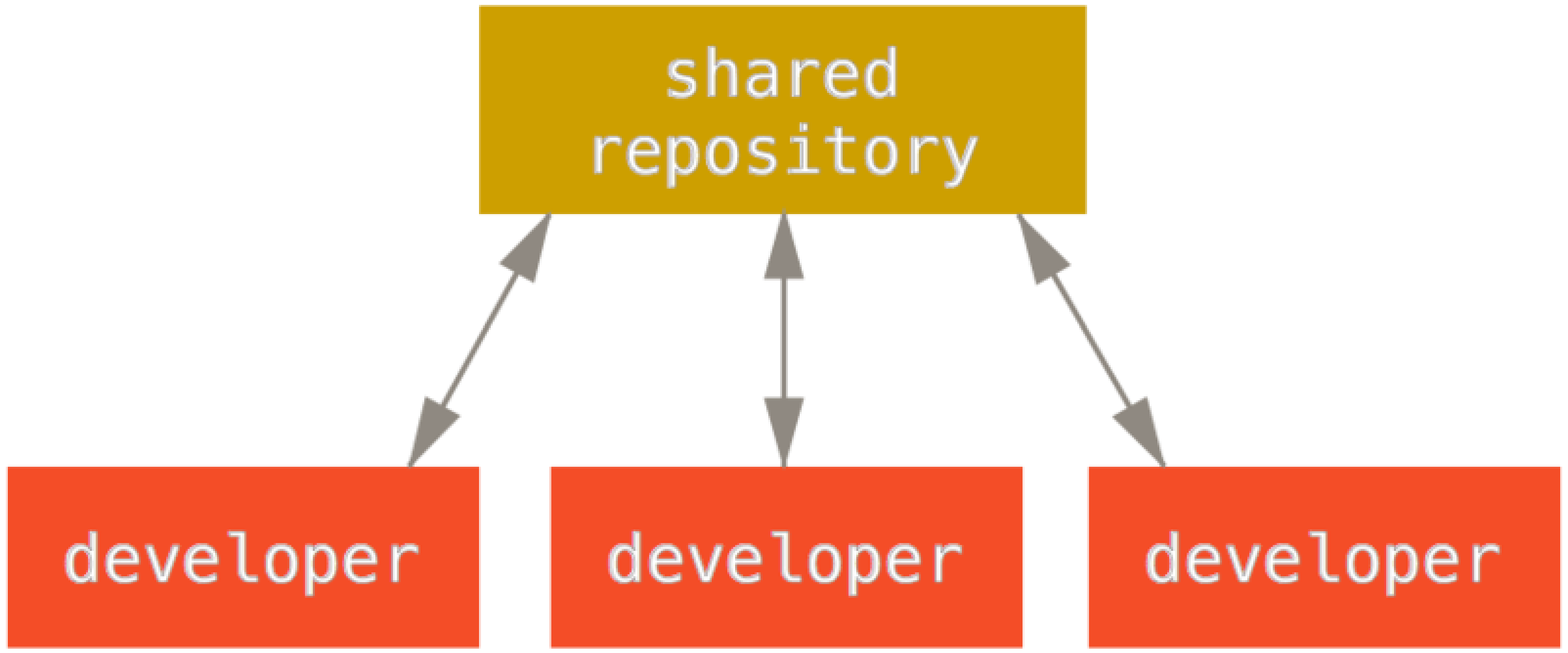
Lecture 8

Integration-Manager Workflow and Rebasing


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

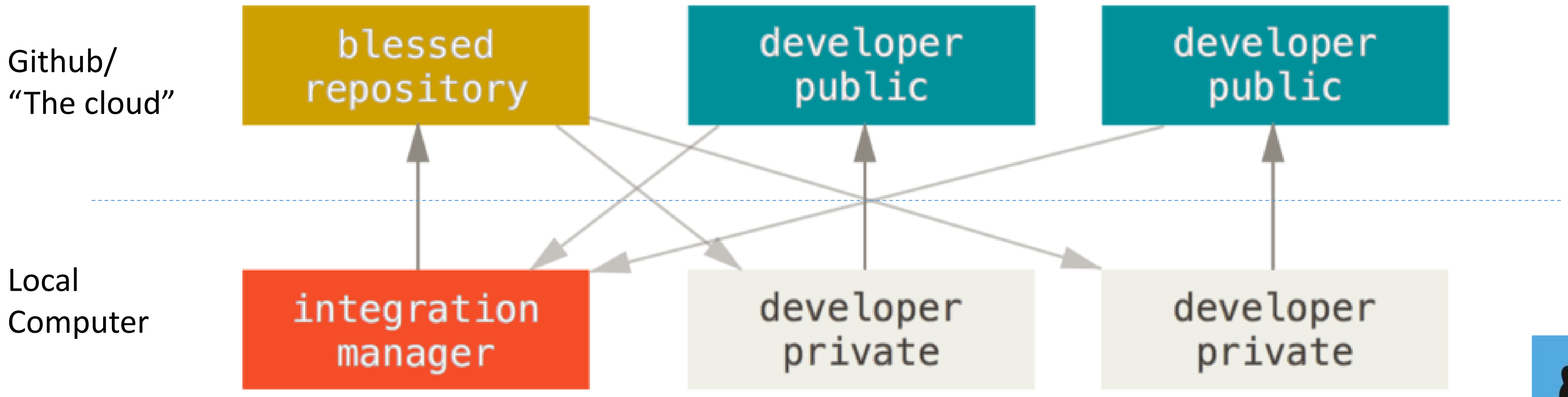
Sign in on the attendance sheet!

Remember the Centralized Workflow?



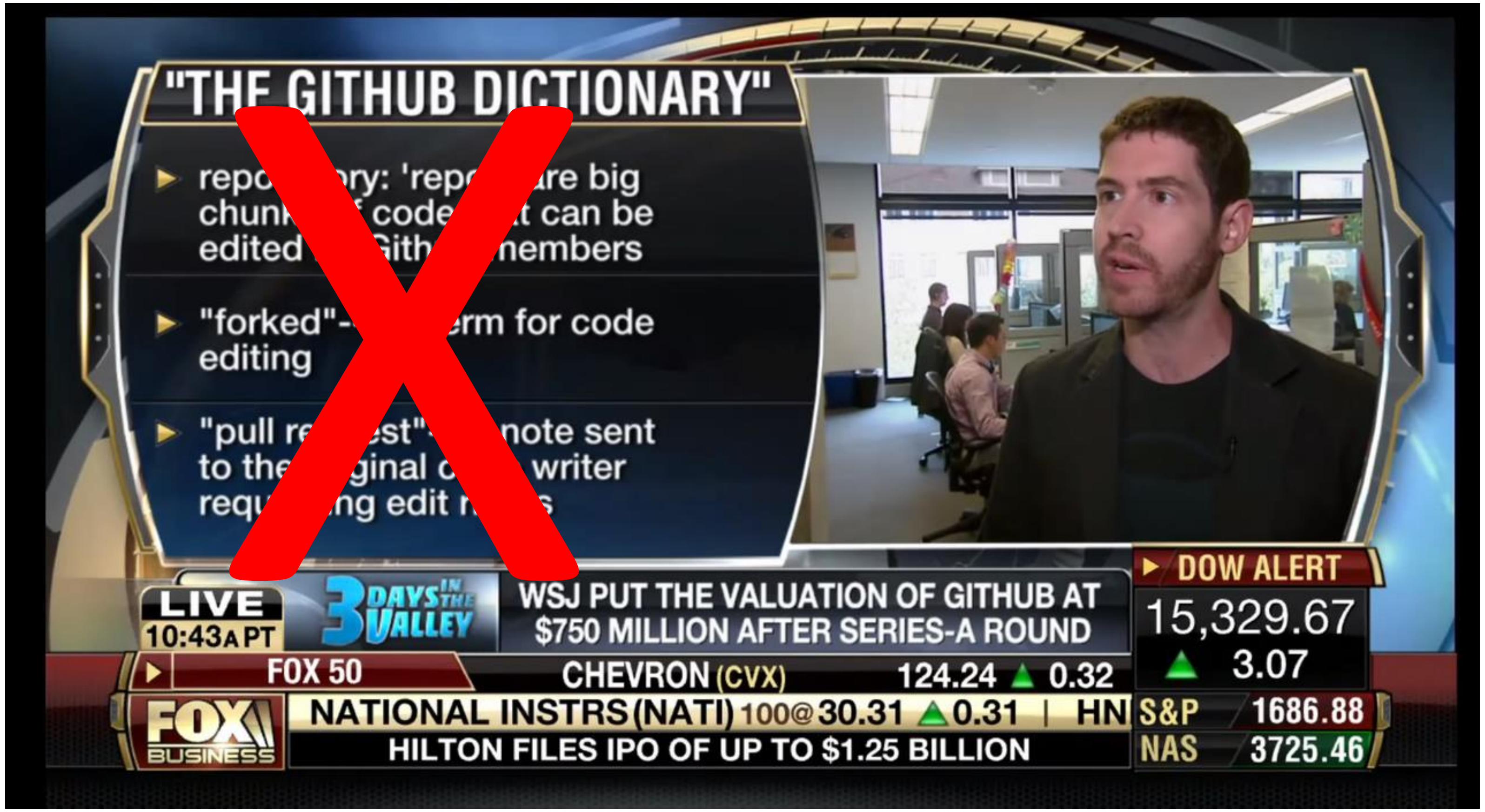
Problem: Every developer needs **push access** to the shared repository!

Integration-Manager Workflow



Step 1. **Fork** the public repository

(make your own public copy)

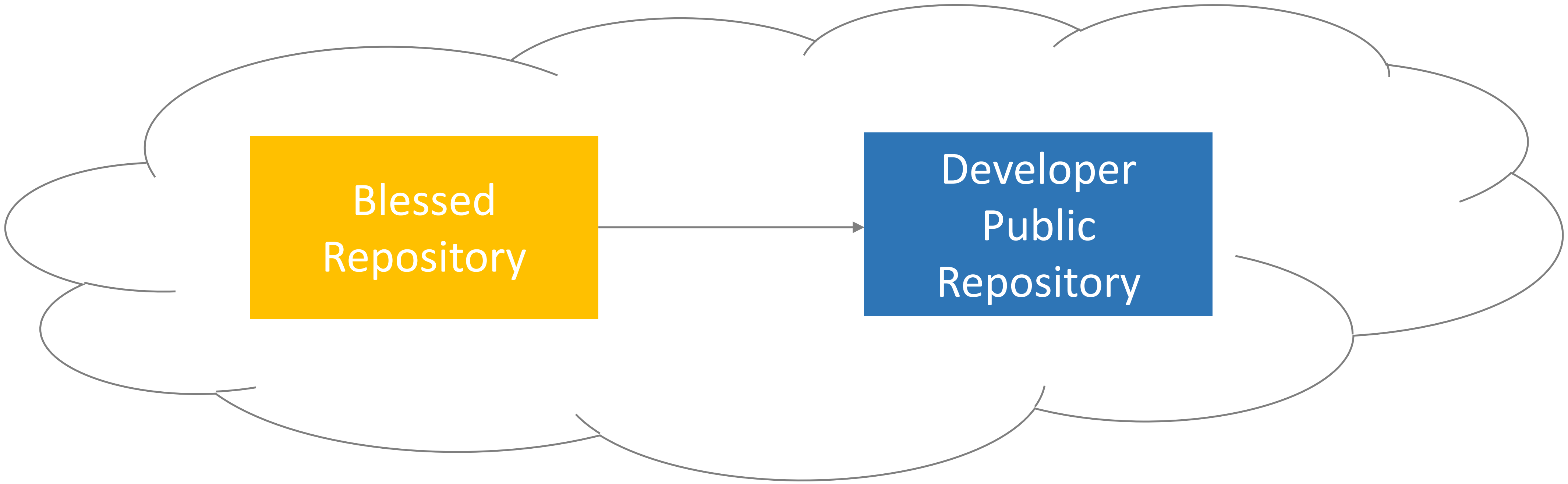


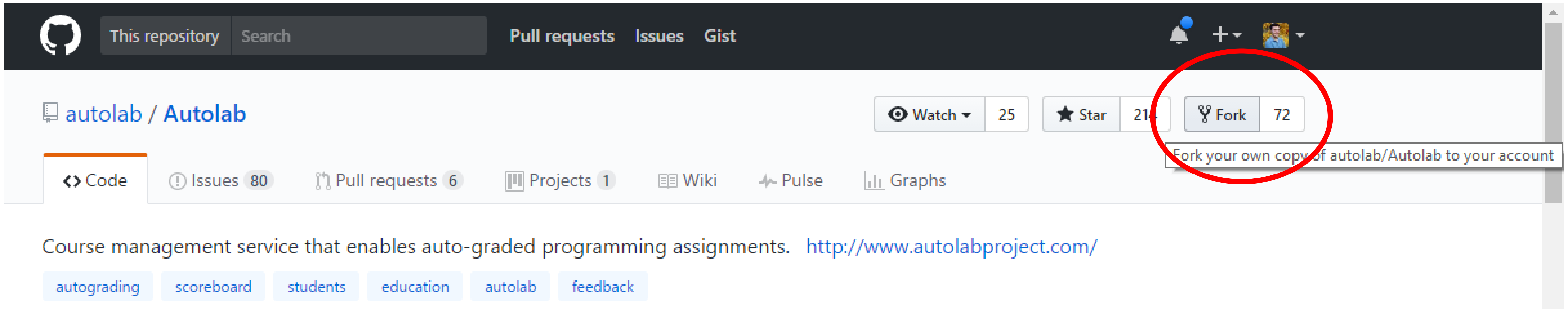
"THE GITHUB DICTIONARY"

- repo ry: 'rep re big chun f code t can be edited Gith nembers
- "forked"- rm for code editing
- "pull re est"- note sent to the ginal c writer requ ng edit r s

LIVE 10:43A PT
3 DAYS IN THE VALLEY
WSJ PUT THE VALUATION OF GITHUB AT $750 MILLION AFTER SERIES-A ROUND
DOW ALERT
15,329.67
3.07
FOX 50
CHEVRON (CVX) 124.24 0.32
NATIONAL INSTRS (NATI) 100@30.31 0.31 | HN
HILTON FILES IPO OF UP TO $1.25 BILLION
S&P 1686.88
NAS 3725.46
FOX BUSINESS

Step 1. **Fork** the public repository



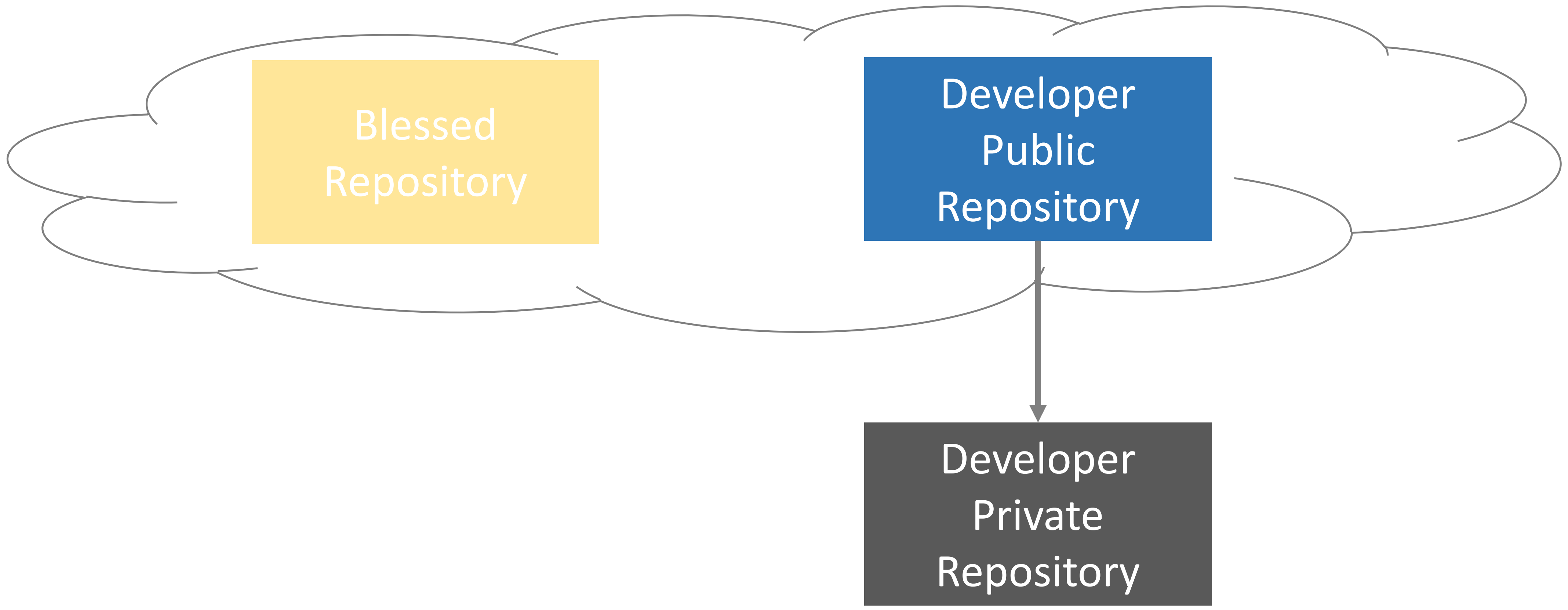


Course management service that enables auto-graded programming assignments. http://www.autolabproject.com/

autograding scoreboard students education autolab feedback

Step 2. Clone your public repository

```
$ git clone https://github.com/aperley/Autolab.git
```

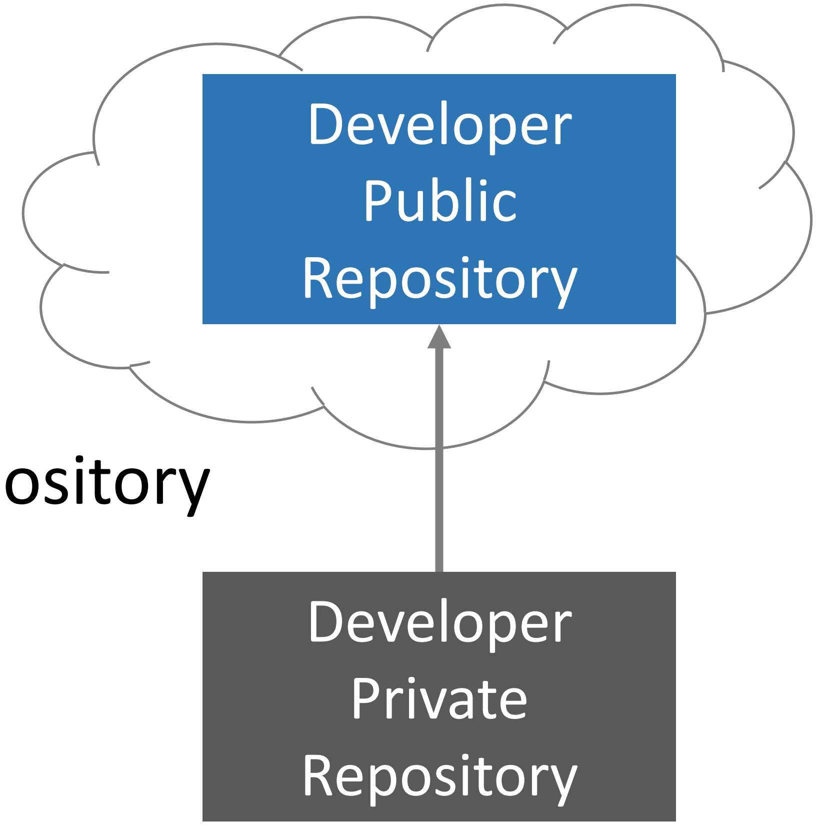


Step 3. Create a **feature branch** and make some commits

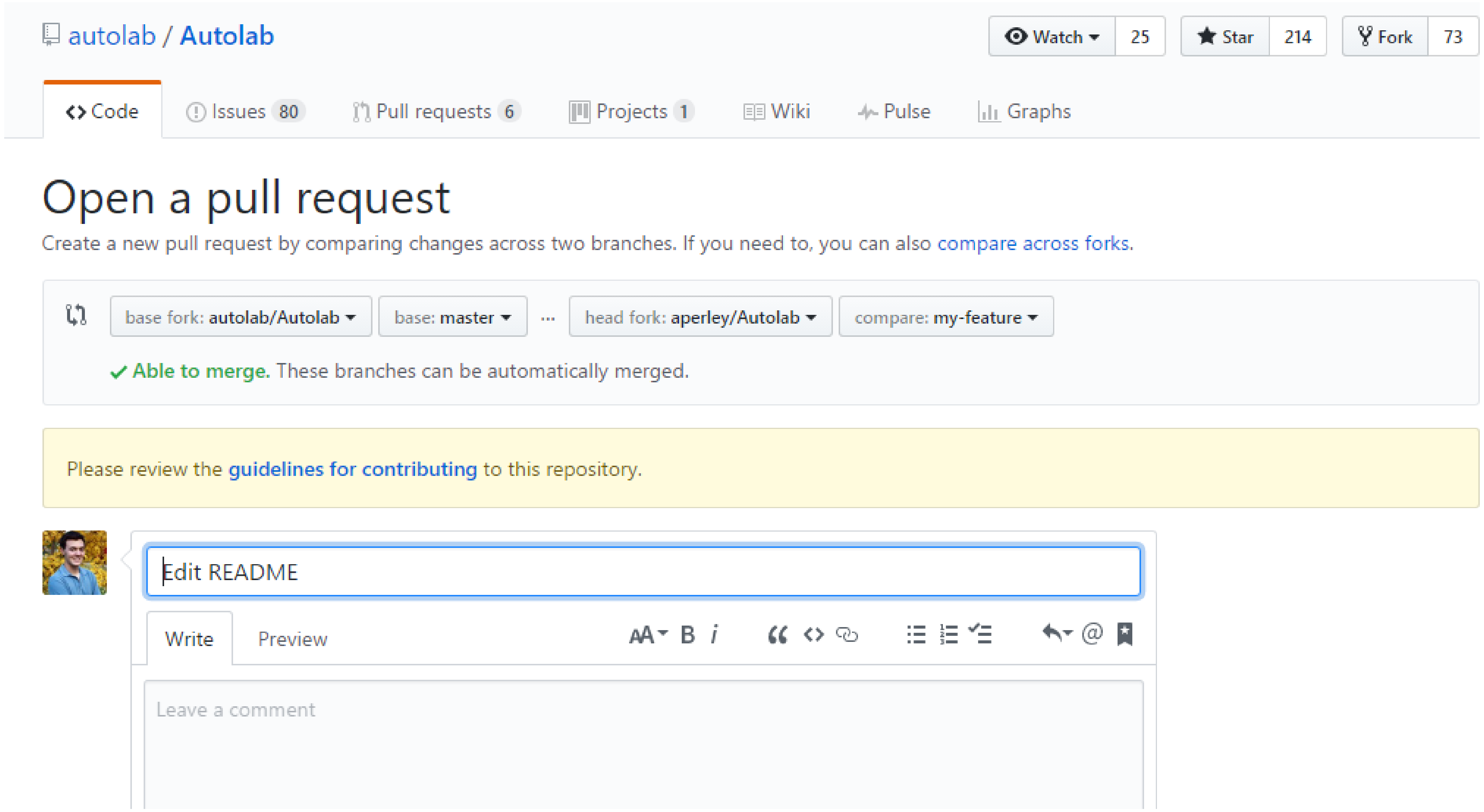
```
$ git checkout -b my-feature
$ <do some work>
$ git commit -am "add my feature"
```

Then **push** your feature branch to your public repository

```
$ git push origin my-feature
```



Step 4. Create a **pull request**



autolab / Autolab

Watch 25 | Star 214 | Fork 73

Code | Issues 80 | Pull requests 6 | Projects 1 | Wiki | Pulse | Graphs

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

base fork: autolab/Autolab | base: master | ... | head fork: aperley/Autolab | compare: my-feature

✓ Able to merge. These branches can be automatically merged.

Please review the guidelines for contributing to this repository.

Edit README

Write | Preview

Leave a comment

The integration manager can inspect and **pull in** your changes

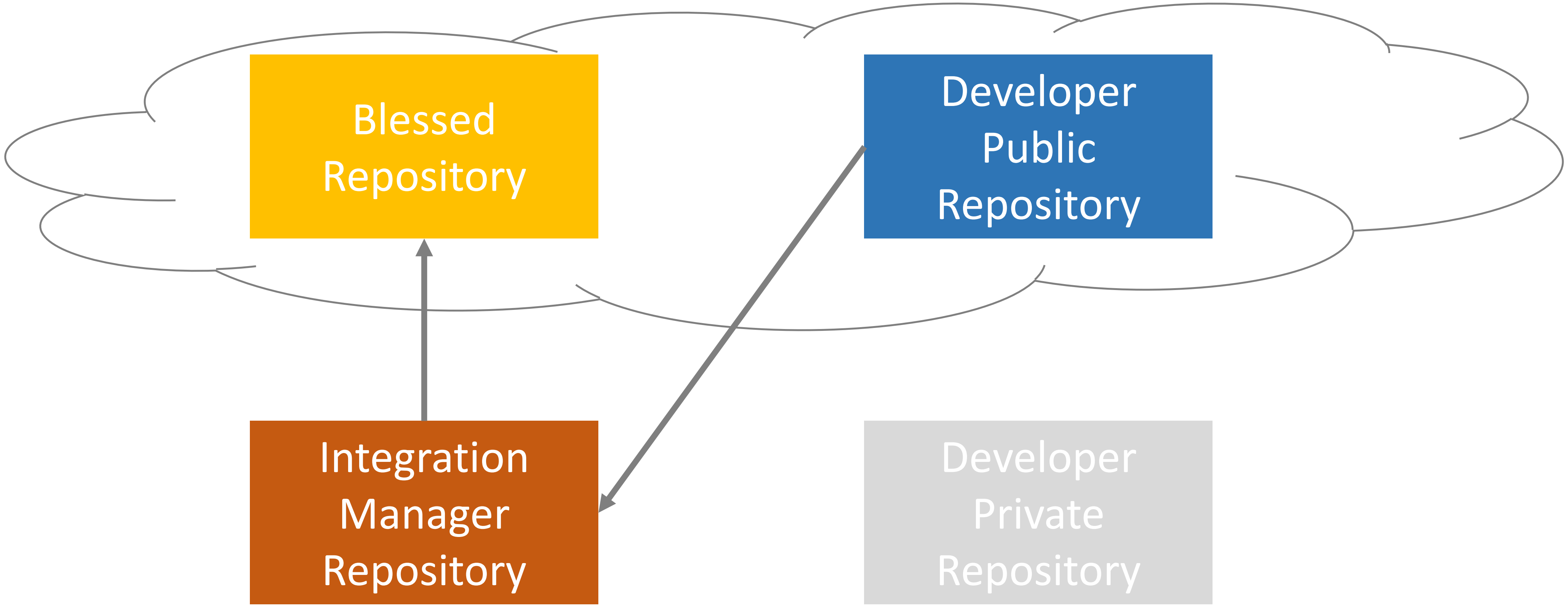
As the integration manager:

```
$ git remote add aperleys-fork
https://github.com/aperley/Autolab.git
$ git checkout aperleys-fork/my-feature
```

If it looks good:

```
$ git checkout master
$ git merge aperleys-fork/my-feature
$ git push origin master
```

The integration manager can inspect and **pull in** your changes

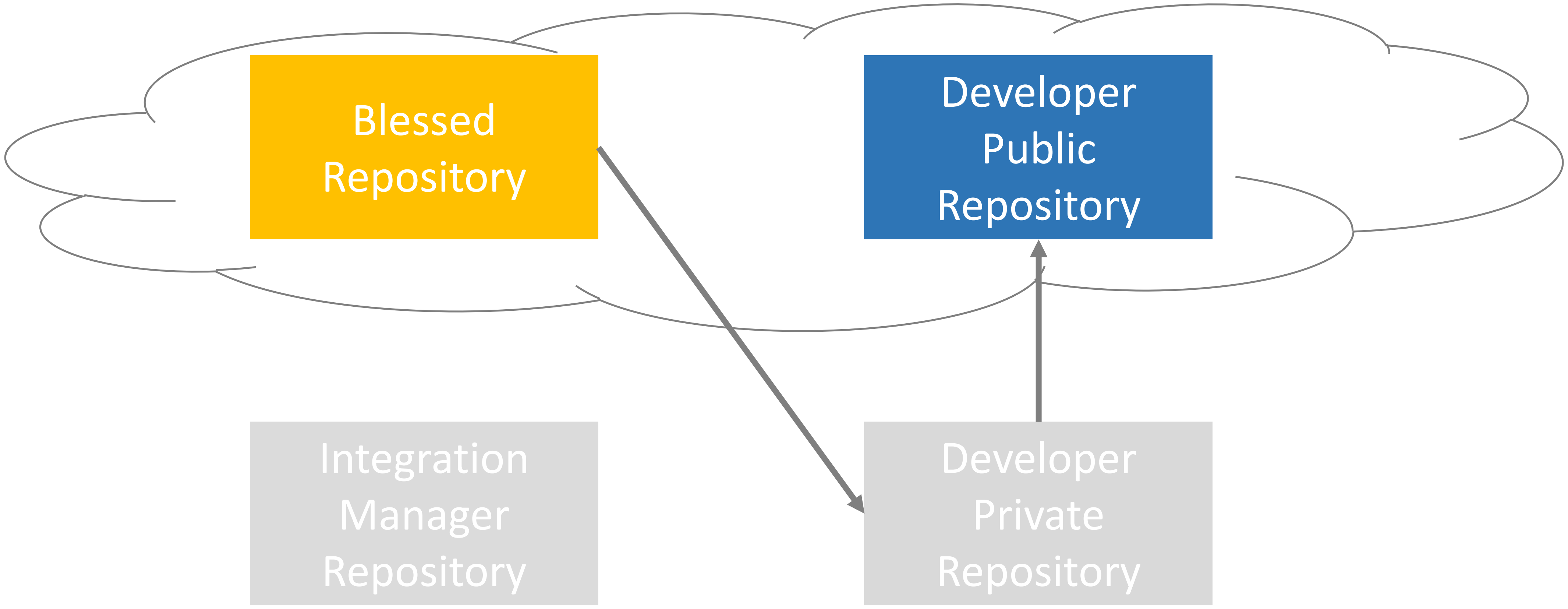

Blessed Repository
Developer Public Repository
Integration Manager Repository
Developer Private Repository

You need to keep your fork up to date

In the private developer repo

```
$ git remote add upstream
https://github.com/autolab/Autolab.git
$ git fetch upstream
$ git checkout master
$ git merge upstream/master
$ git push origin master
```

You need to keep your fork up to date


Blessed Repository
Developer Public Repository
Integration Manager Repository
Developer Private Repository

Git Rebase: Squashing Commits

X

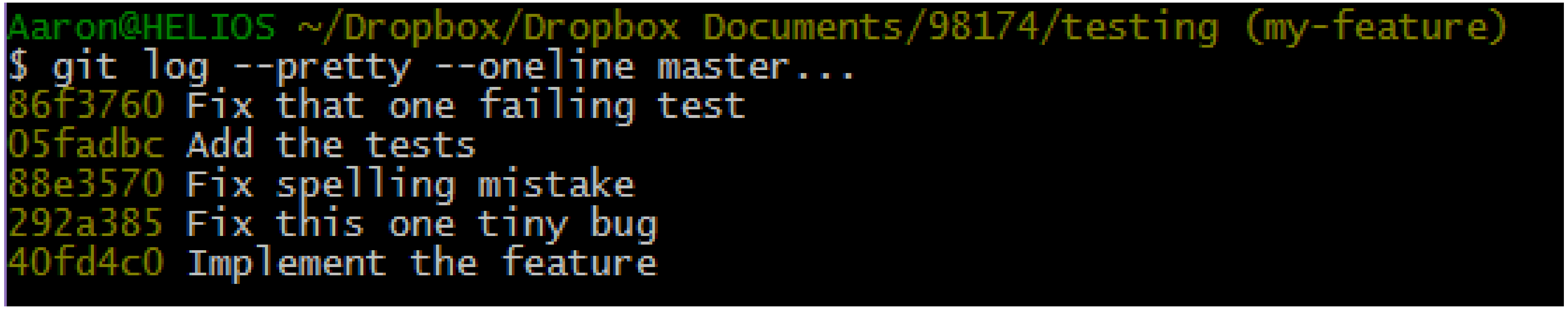
X

✓

Squashing Commits

Scenario:

Made some commits on a feature branch but want to “clean it up” before making a pull request or merging to master



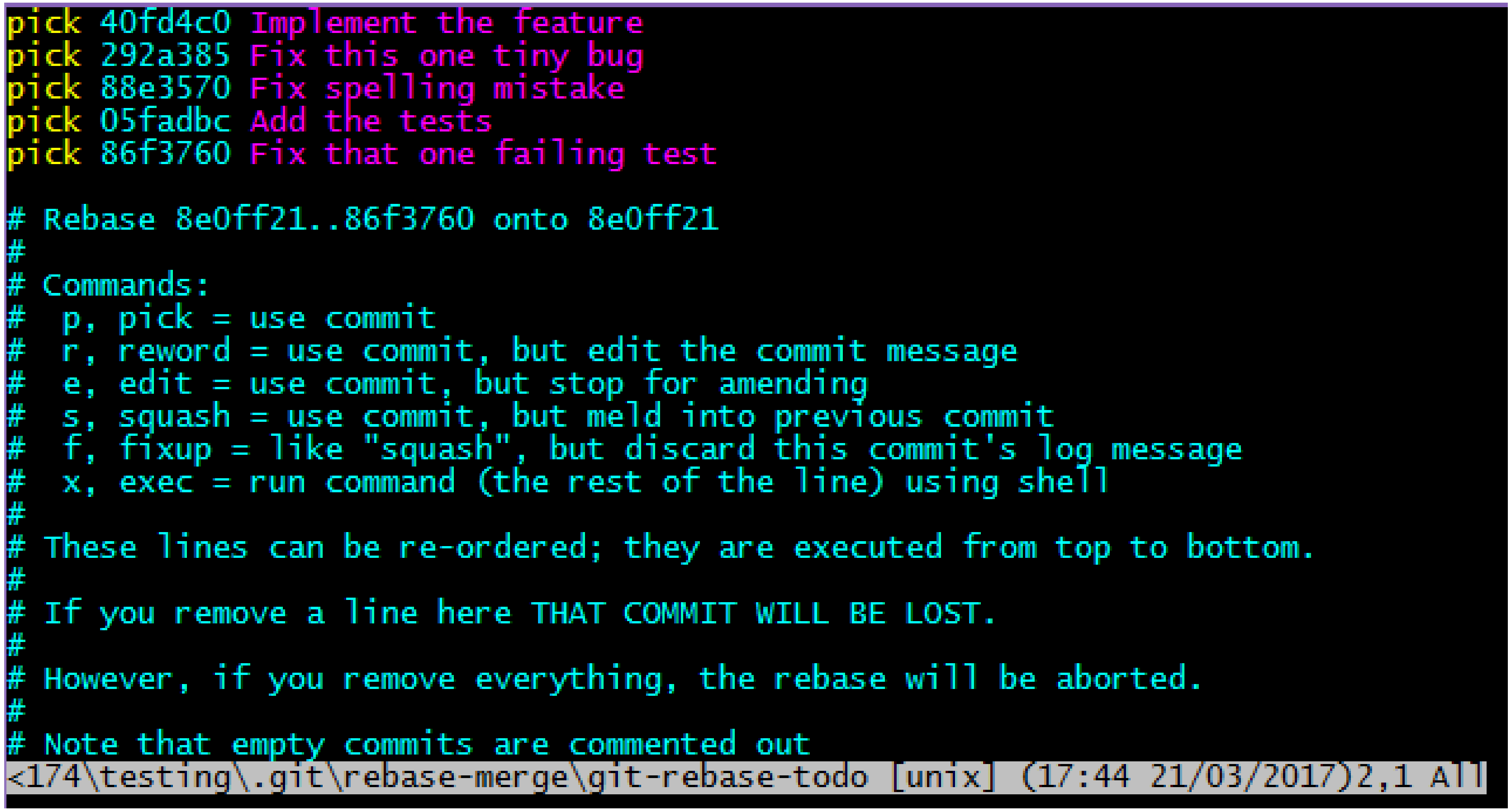
```
Aaron@HELIOS ~/Dropbox/Dropbox Documents/98174/testing (my-feature)
$ git log --pretty --oneline master...
86f3760 Fix that one failing test
05fadbc Add the tests
88e3570 Fix spelling mistake
292a385 Fix this one tiny bug
40fd4c0 Implement the feature
```

Squashing Commits

```
$ git rebase -i master
```

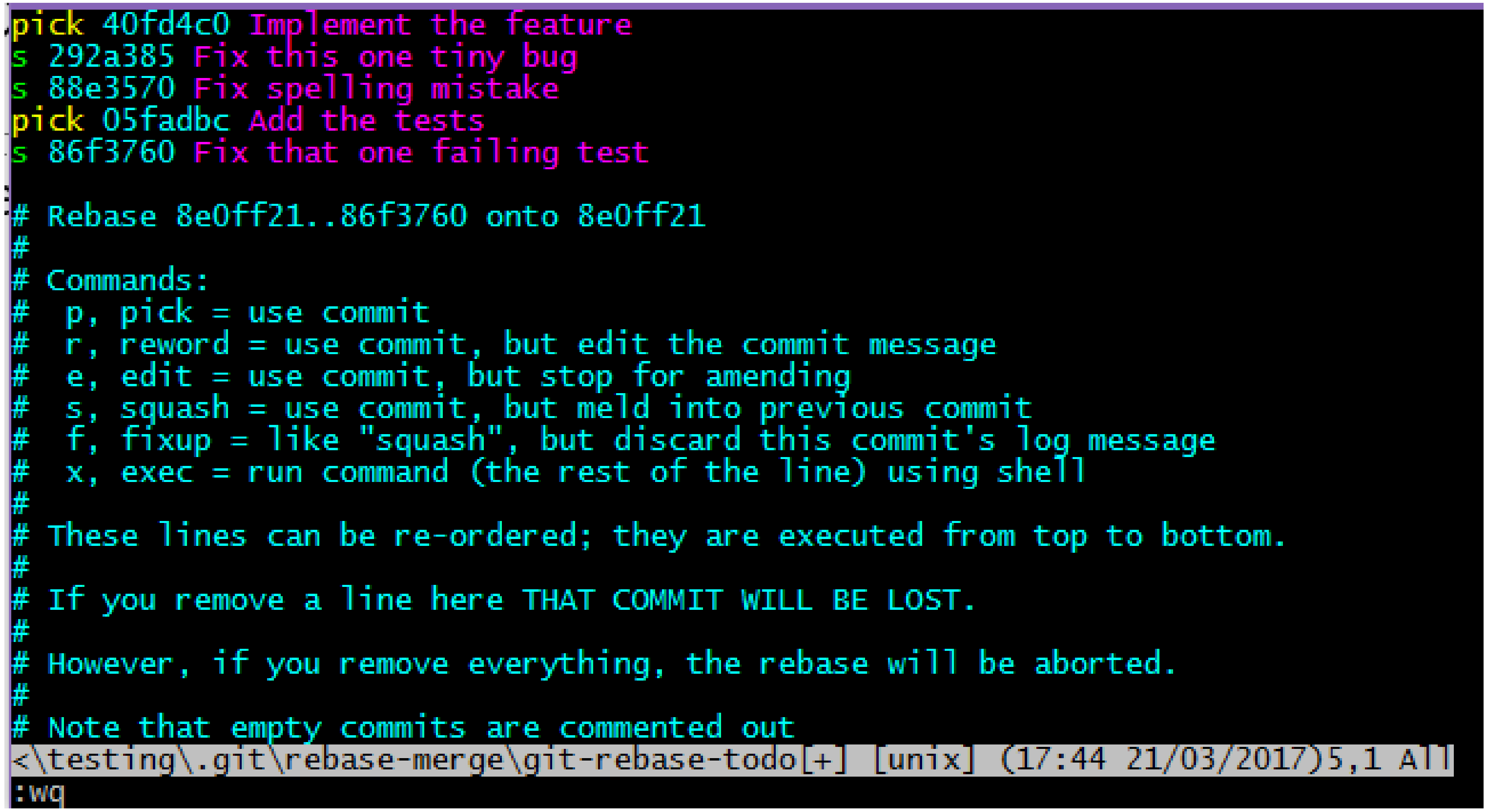
Begins an interactive rebase of all of the commits since the branch split off of master.

Interactive Rebase



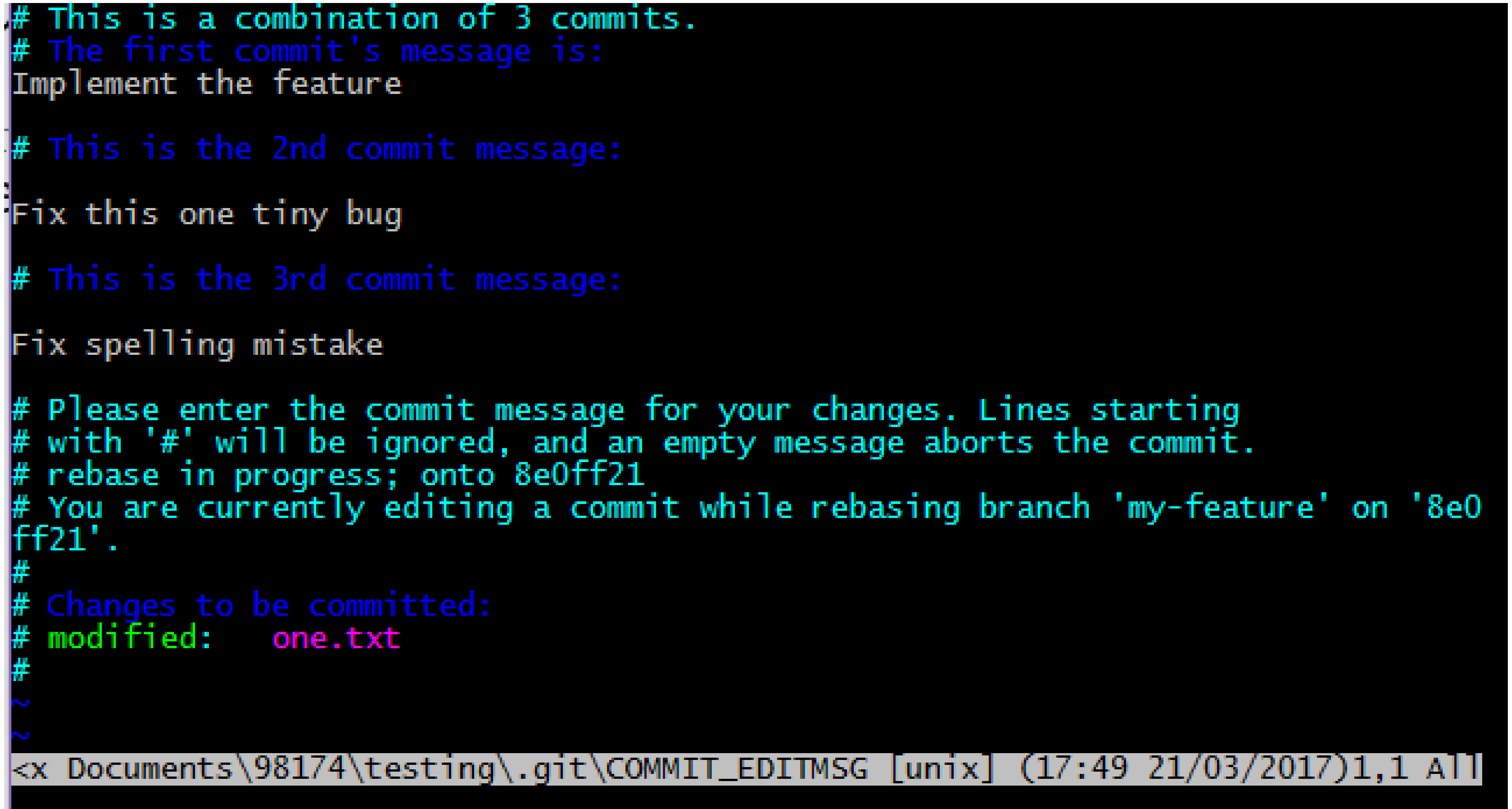
```
pick 40fd4c0 Implement the feature
pick 292a385 Fix this one tiny bug
pick 88e3570 Fix spelling mistake
pick 05fadbc Add the tests
pick 86f3760 Fix that one failing test

# Rebase 8e0ff21..86f3760 onto 8e0ff21
#
# Commands:
#  p, pick = use commit
#  r, reword = use commit, but edit the commit message
#  e, edit = use commit, but stop for amending
#  s, squash = use commit, but meld into previous commit
#  f, fixup = like "squash", but discard this commit's log message
#  x, exec = run command (the rest of the line) using shell
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
#
# However, if you remove everything, the rebase will be aborted.
#
# Note that empty commits are commented out
<174\testing\.git\rebase-merge\git-rebase-todo [unix] (17:44 21/03/2017)2,1 All
```

Decide what you want to squash onto the commit above (before)



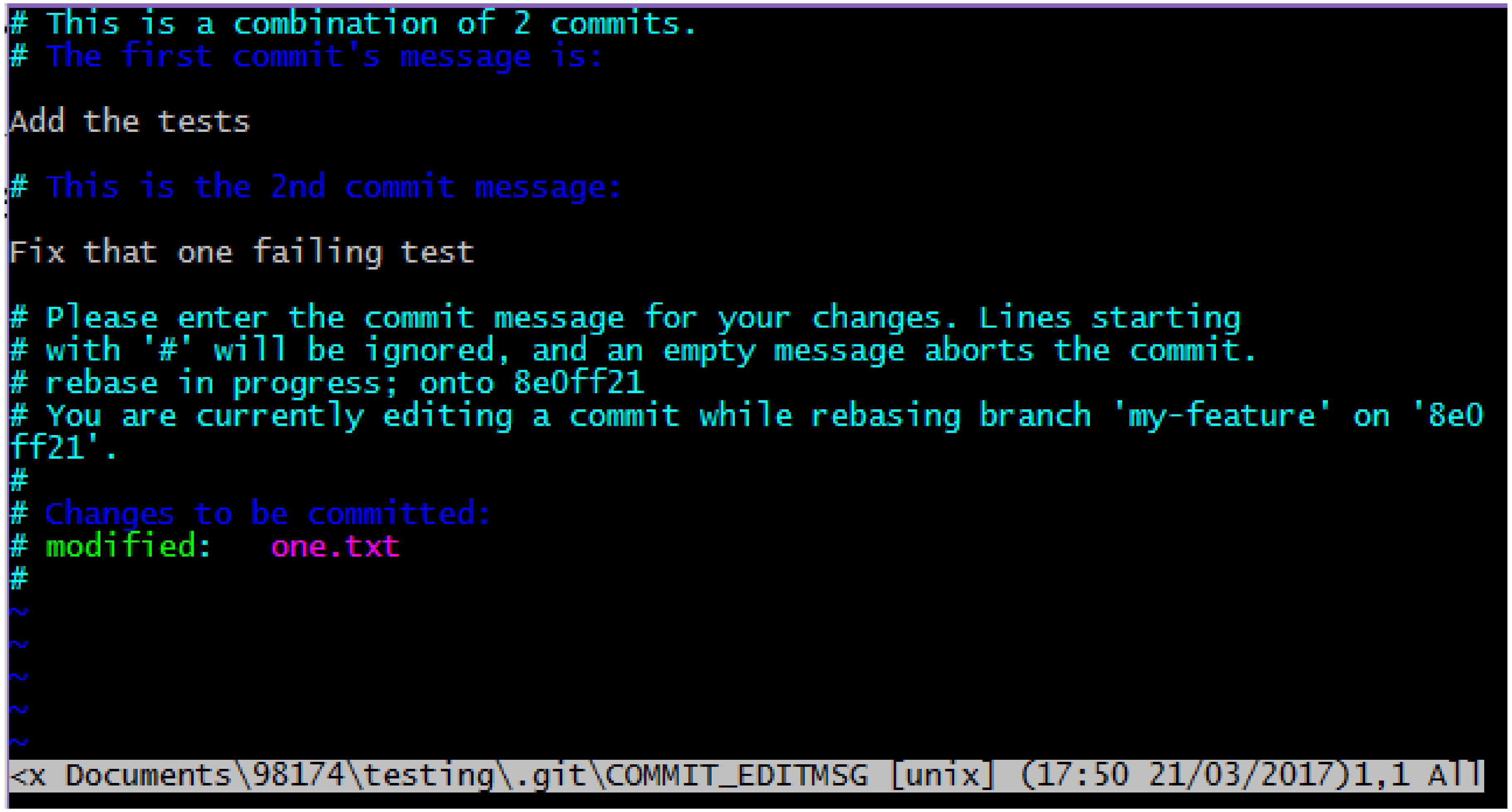
```
pick 40fd4c0 Implement the feature
s 292a385 Fix this one tiny bug
s 88e3570 Fix spelling mistake
pick 05fadbc Add the tests
s 86f3760 Fix that one failing test

# Rebase 8e0ff21..86f3760 onto 8e0ff21
#
# Commands:
#  p, pick = use commit
#  r, reword = use commit, but edit the commit message
#  e, edit = use commit, but stop for amending
#  s, squash = use commit, but meld into previous commit
#  f, fixup = like "squash", but discard this commit's log message
#  x, exec = run command (the rest of the line) using shell
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
#
# However, if you remove everything, the rebase will be aborted.
#
# Note that empty commits are commented out
<\testing\.git\rebase-merge\git-rebase-todo[+] [unix] (17:44 21/03/2017)5,1 All
:wq
```

Now we need to edit the commit message for the first squash group



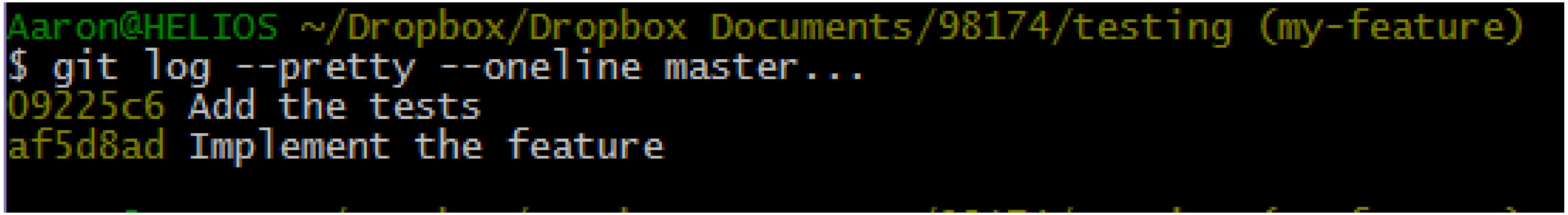
```
# This is a combination of 3 commits.
# The first commit's message is:
Implement the feature

# This is the 2nd commit message:

Fix this one tiny bug

# This is the 3rd commit message:

Fix spelling mistake

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# rebase in progress; onto 8e0ff21
# You are currently editing a commit while rebasing branch 'my-feature' on '8e0
ff21'.
#
# Changes to be committed:
# modified:   one.txt
#
~
~
<x Documents\98174\testing\.git\COMMIT_EDITMSG [unix] (17:49 21/03/2017)1,1 All
```

Same for the second squash group



```
# This is a combination of 2 commits.
# The first commit's message is:

Add the tests

# This is the 2nd commit message:

Fix that one failing test

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# rebase in progress; onto 8e0ff21
# You are currently editing a commit while rebasing branch 'my-feature' on '8e0
ff21'.
#
# Changes to be committed:
# modified:   one.txt
#
~
~
~
~
~
<x Documents\98174\testing\.git\COMMIT_EDITMSG [unix] (17:50 21/03/2017)1,1 All
```

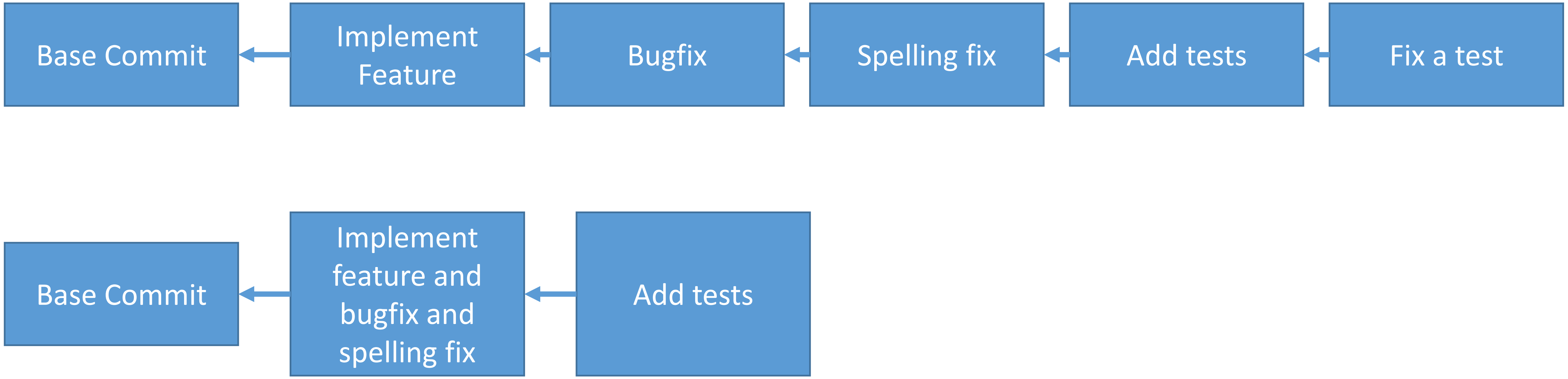
We are left with a new commit history



```
Aaron@HELIOS ~/Dropbox/Dropbox Documents/98174/testing (my-feature)
$ git log --pretty --oneline master...
09225c6 Add the tests
af5d8ad Implement the feature
```

What happened?

We **rewrote history** by replaying the patches for each commit


Base Commit
Implement Feature
Bugfix
Spelling fix
Add tests
Fix a test
Base Commit
Implement feature and bugfix and spelling fix
Add tests