Introduction to Git and Github

Sean Sall

May 9th, 2016

Objectives:

Today's objectives:

- Explain the basic function and purpose of version control
- Become familiar with the basic/fundamental git commands and what each is used for:
  1. fork
  2. clone
  3. add, commit
  4. push, pull
  5. pull request (less fundamental, but used frequently throughout the program)

Agenda

Today's plan:

1. Version control and `Git`/`Github` Overview
2. Intro to `Git` Commands

Why does this matter?

- Version Control:
  - Using some sort of version control system is nearly universal in the tech and data science world, and Git is one of the most common.
- `Git`/`Github`:
  - Allows you to take advantage of version control and everything that it offers.
  - `Github` (built on `Git`) will allow you to share your personal projects as well as collaborate on projects with others.

Version Control

- A **version control system** is a repository of files with monitored access
  - Files are primarily source code, but can be of other types
- Version control is useful primarily because every change to a repository is tracked, along with who made it, why they made it, and references to any problems fixed or enhancements made by the change.
  - Provides the ability to track changes over time, and the ability to reverse any of them if necessary
  - Allows for easy collaboration across teams

Version Control Systems

- Numerous version control systems exist, but four of the most popular are:
  1. `Git` (distributed)
  2. `Mercurial` (distributed)
  3. `Subversion` (centralized)
  4. `Concurrent Versions Systems` (centralized)
- With **centralized** systems, you have to be connected to the central repository at all times (e.g. you typically need an Internet connection), while with **distributed** systems there is more freedom to work locally.

Why Git?

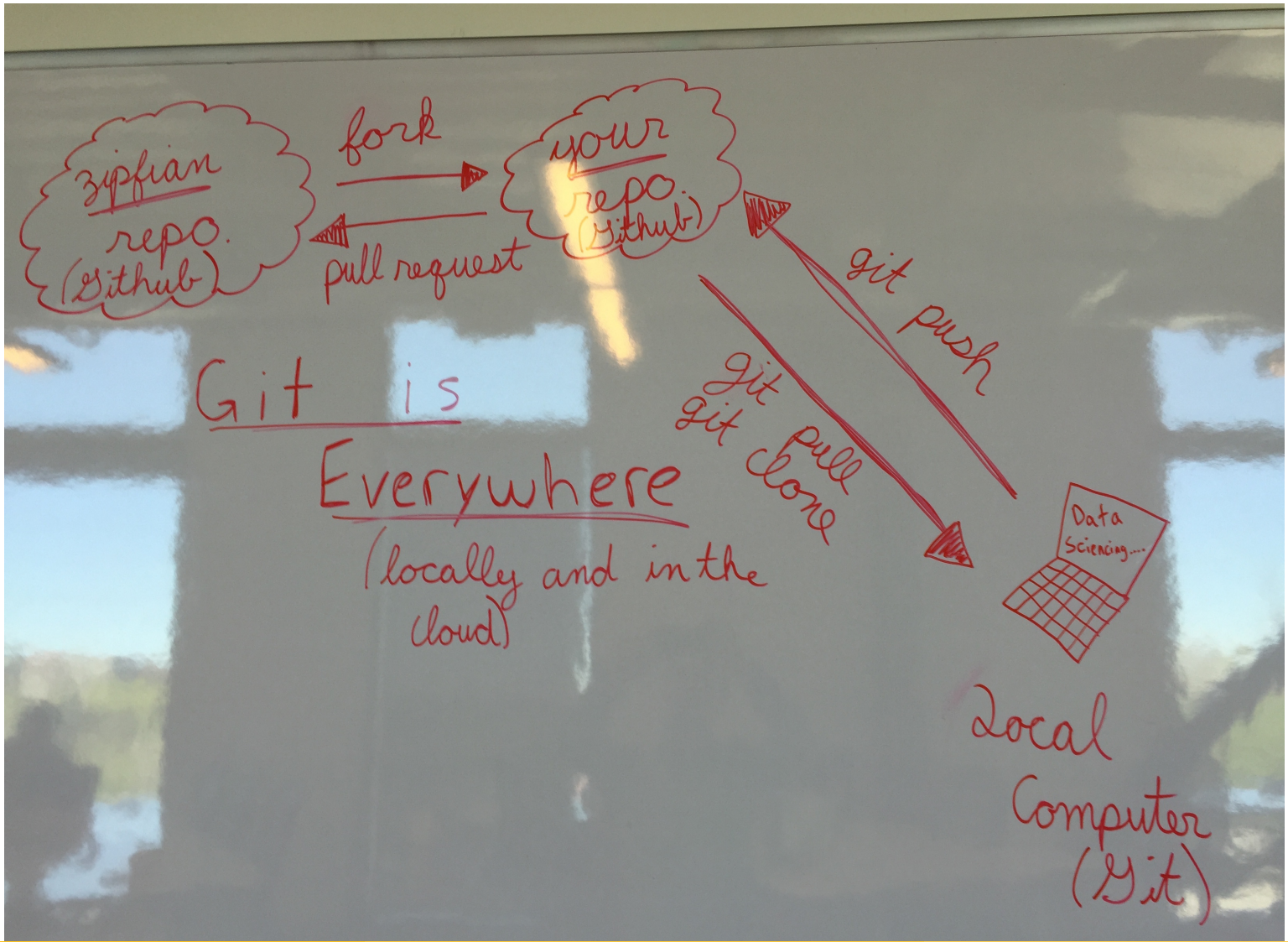
- Why use Git?
  - Distributed (allows more freedom to work locally)
  - Free, open source
  - Collection of hosting services for Git repositories (Bit Bucket, Github)
  - Arguably the most popular

Git v. Github

- What's the difference between Git v. Github?
  - Git is a **version control system**, while Github is a **web-based hosting service** for Git repositories (e.g. Github is "in the cloud", whereas Git works locally).
  - Git exists independently of Github, while the converse is not true.

Git v. Github Part Dos

- What if we visualize how Git and Github interact?



Intro `Git` Commands

- Major **`Git`** commands include:
  - `clone`: Makes a copy (clone) of a repository into a newly created directory, **with** a reference still pointing to the original repository.
  - `add`: Add one or more files to the index (e.g. tell `Git` to keep track of these files)
  - `commit`: Commit your changes, creating a "checkpoint" that can then be referenced or reverted back to later
  - `push`: Updates a remote copy of the repository with local changes
  - `pull`: Updates a local copy of the repository with remote changes

Intro `Github` Commands

- Major `Github` (issued **from the browser**) commands include:
  - `fork`: Makes a copy of a repository onto your personal Github account (it's like the `Github` version of cloning), but **without** a reference still pointing to the original repository.
  - `pull request`: Issued to try to update a repository with changes from another copy of the repository (a `fork` or `branch`)

Classroom Git/Github Workflow

- Let's `fork` the `assessment-day1` repository (**Note**: We're looking at the Zipfian copy of the repository, and just clicking that `Fork` button):

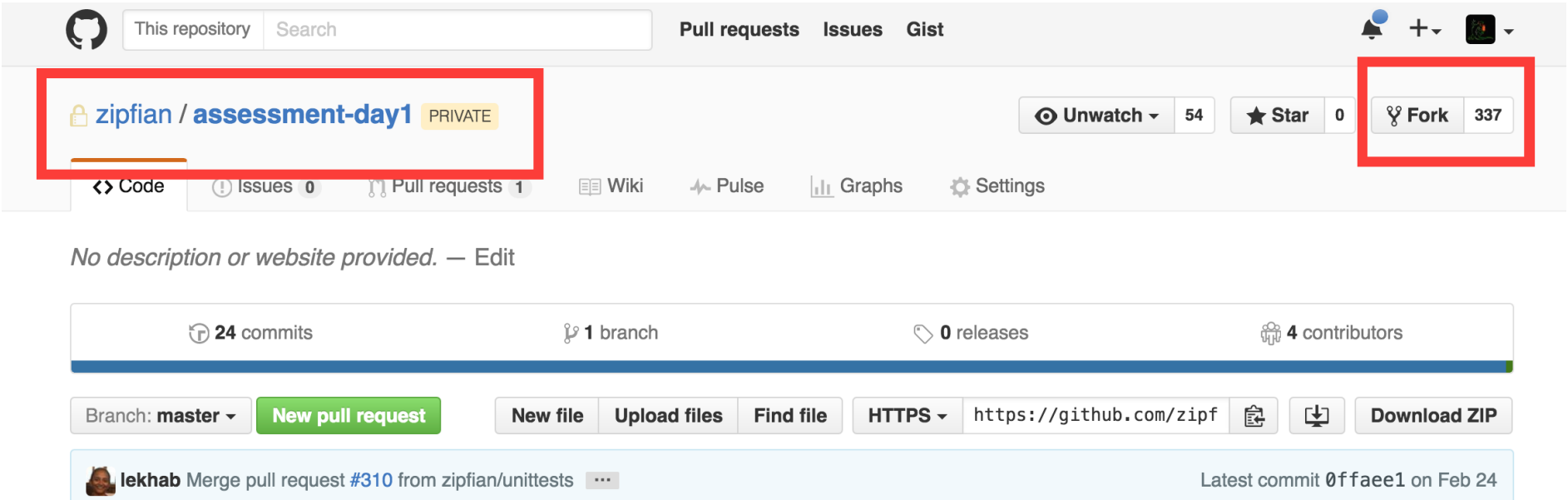


Figure 2:Forks!

Classroom `Git`/`Github` Workflow

- Now that we've `forked`, let's `clone` (**Note**: We're now looking at **our** personal copy of the repository, issuing this command from our terminal):

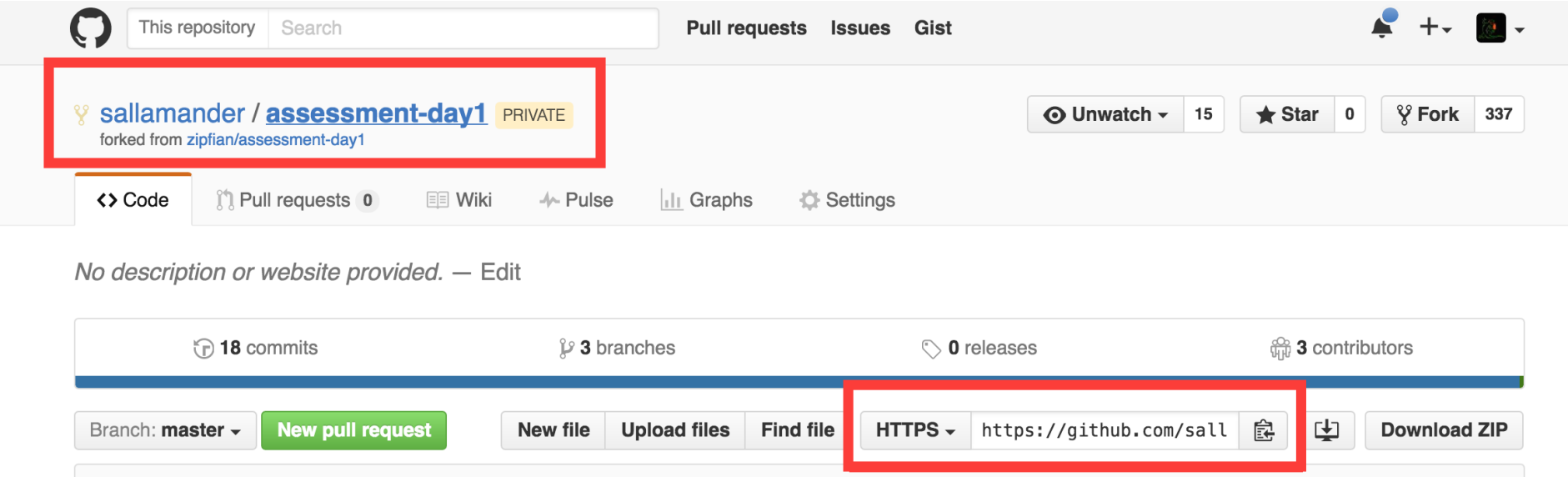


Figure 3:Cloning!

```
git clone https://github.com/sallamander/assessment-day1.git
```

Classroom Git/Github Workflow

- So, we issued a `fork` **in our browser** to get our own personal copy of some repository (here `assessment-day1`), and then we issue a `git clone` **in our terminal** to get that personal copy on our local machine.
- Now, you crush the assessment! Along the way, you'll `add`, `commit`, and `push` any changes you make. Let's say I just finished a part of the assessment in the `assessment.py` file, and I now want to tell `Git` to make sure that it registers all those changes...

```
git add assessment.py
git commit -m 'Crush part 1 of assessment.py'
git push
```

Classroom `Git`/`Github` Workflow

- Now that you've finished the assessment, we'd like to take a look at it. To do so, we're going to have you **ask** that the changes you've made to your copy of the repository (your **forked** copy) are used to update our copy (the **zipfian copy**). This is done via a **`pull request`**.

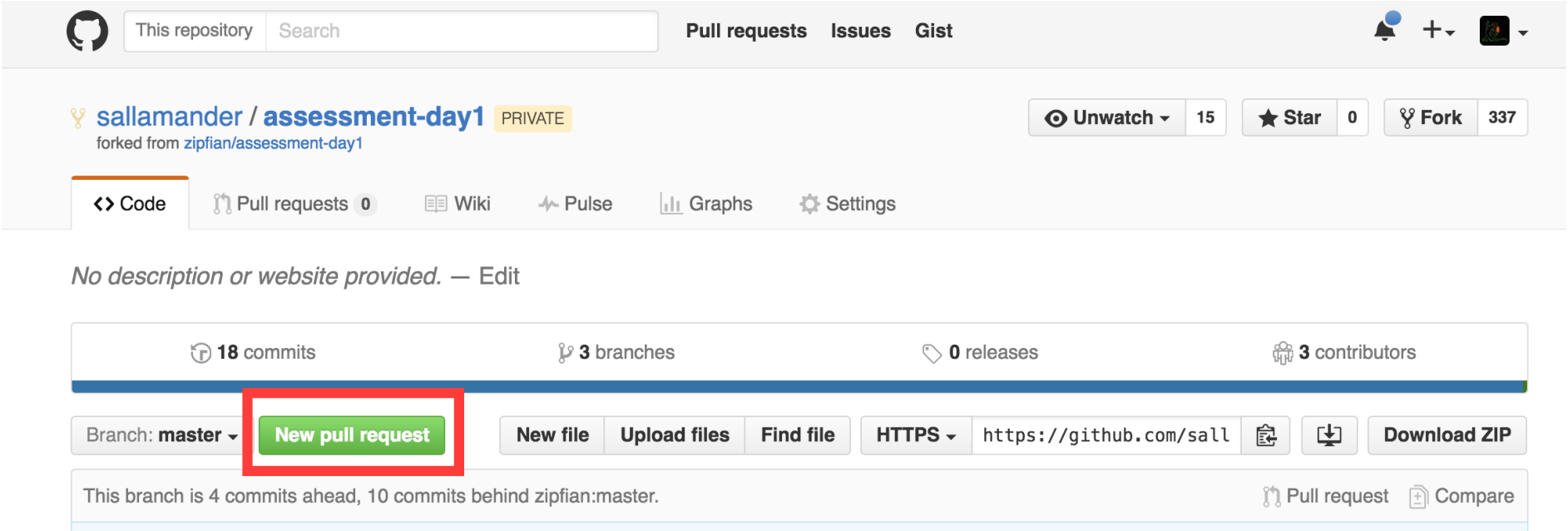


Figure 4:Pull Request!

Classroom Git/Github Workflow

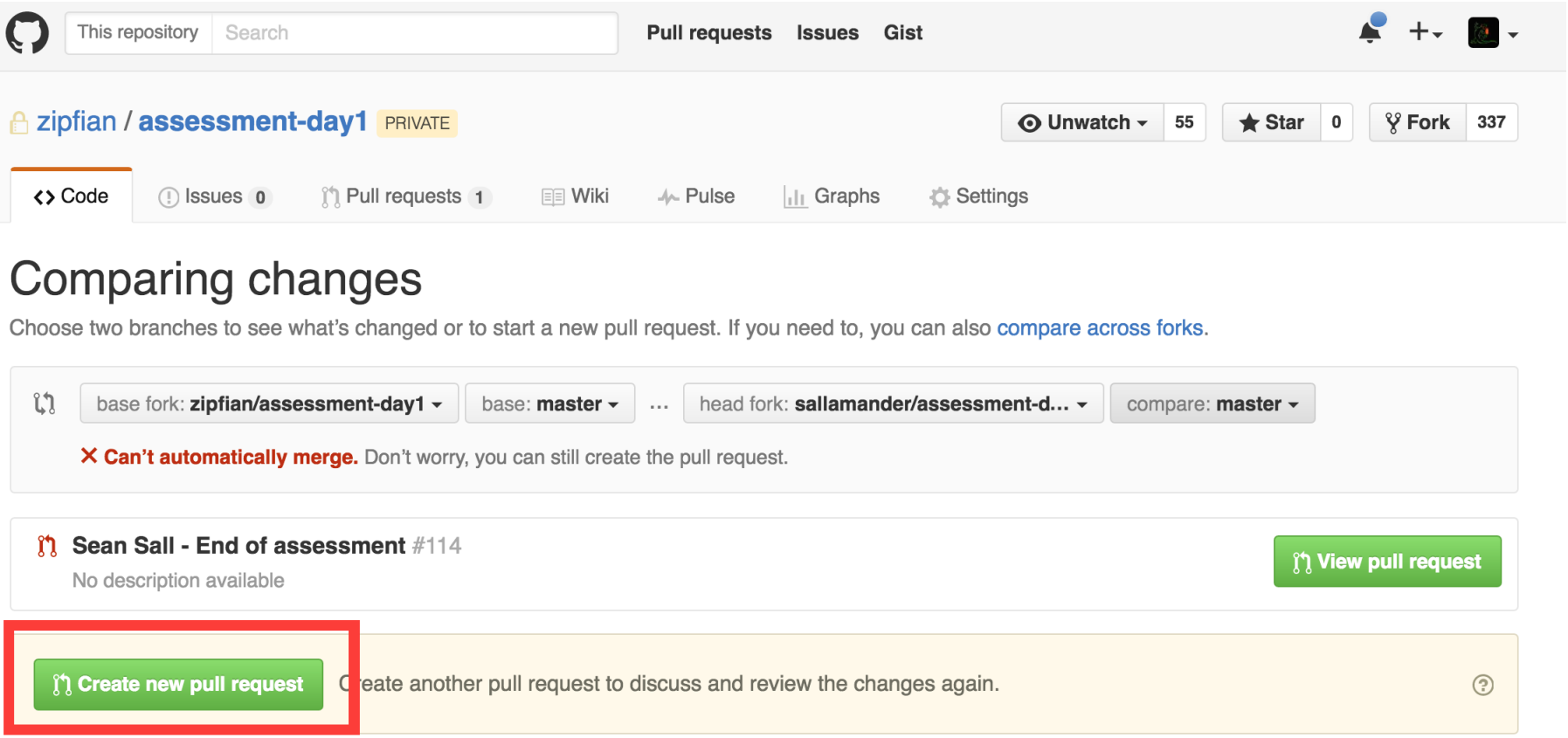


Figure 5:Pull Request!

Classroom Git/Github Workflow

- Once you've clicked the `Pull Request` button, just give the PR a title and a potential message:

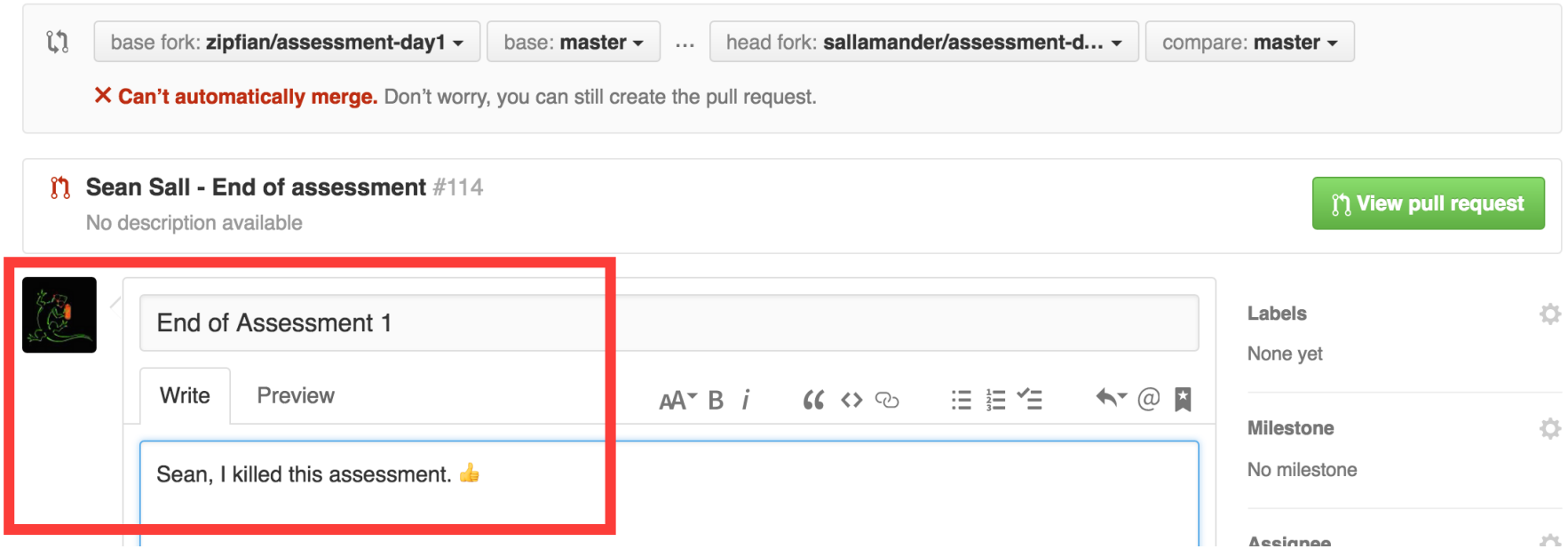


Figure 6:Pull Request!