Fastcampus Data Science Extension SCHOOL

Shell Command, Vim

Index

- git
- collaboration
- get source easily

Goal

- git을 이해하고, git과 github이 다름을 인지한다
- git을 활용하여 나의 소스코드를 관리할 수 있다
- 데이터 사이언티스트의 커리어를 스웩할 나만의 멋진블로그를 만들 수 있다
- git의 branch model을 활용해 능숙하게 코드관리할 수 있다
- git으로 타인과 협업하며, 다른 프로젝트에 기여할 수 있다


git

VCS (Version Control System)

== SCM (Source Code Management)

< SCM (Software Configuration Management: 형상관리)

chronicle of git

chronicle of git

- Linux Kernal을 만들기 위해 Subversion을 쓰다 화가 난 리누스 토발즈는 2주만에 git이라는 버전관리 시스템을 만듦
  git official repo

Characteristics of git

- 빠른속도, 단순한 구조
- 분산형 저장소 지원
- 비선형적 개발(수천개의 브랜치) 가능

데이터 사이언티스트가 git을 잘 써야 하는 이유?

Pros of git

- **중간-발표자료_최종_진짜최종_15-4(교수님이 맘에들어함)_언제까지??_이걸로갑시다.ppt**
- 소스코드 주고받기 없이 동시작업이 가능해져 생산성이 증가
- 수정내용은 **commit** 단위로 관리, 배포 뿐 아니라 원하는 시점으로 **Checkout** 가능
- 새로운 기능 추가는 **Branch**로 개발하여 편안한 실험이 가능하며, 성공적으로 개발이 완료되면 **Merge**하여 반영
- 인터넷이 연결되지 않아도 개발할 수 있음

Open-source project

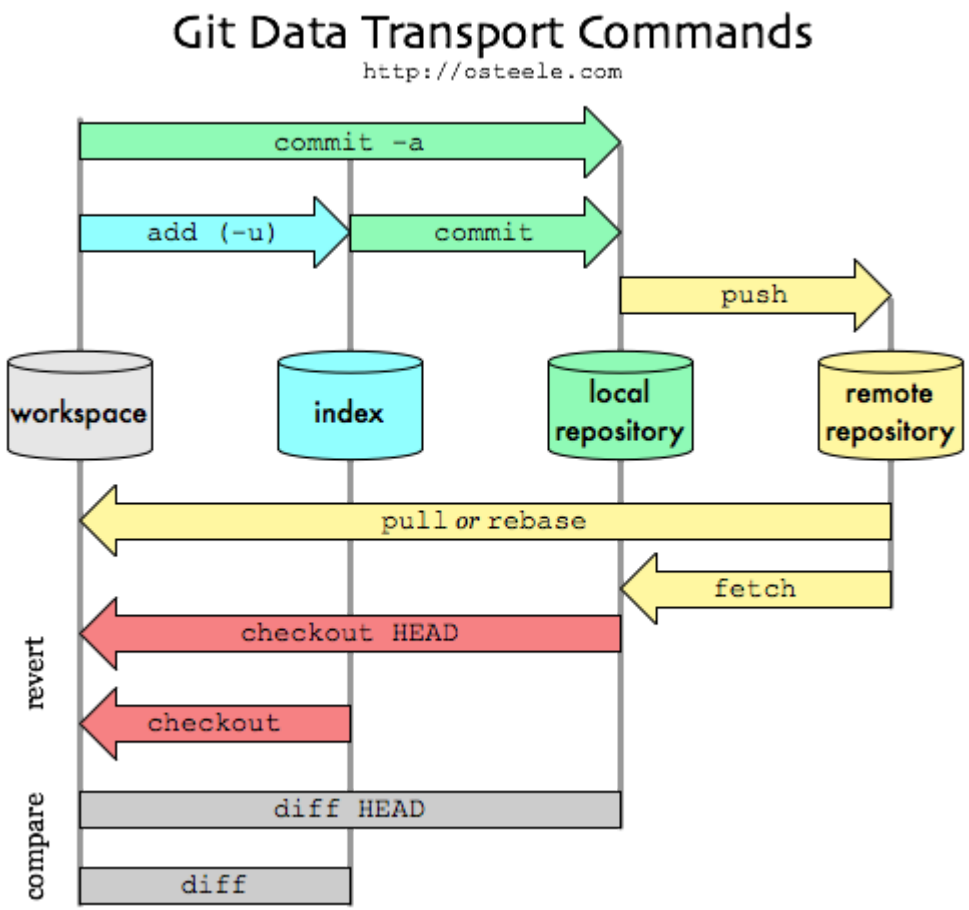
https://github.com/python/cpython
https://github.com/tensorflow/tensorflow

https://github.com/JuliaLang/julia
https://github.com/golang/go

git inside

- Blob: 모든 파일이 Blob이라는 단위로 구성
- Tree: Blob(tree)들을 모은 것
- Commit: 파일에 대한 정보들을 모은 것

git Process and Command



Useful manager for mac

http://brew.sh/index_ko.html

install git

https://git-scm.com/

```
// MacOS
$ brew install git
// Linux
$ sudo apt-get install git
```

- Windows: install git bash

`$ git --version` 으로 정상적으로 설치되었는지를 확인

git is not equal to github

sign up github

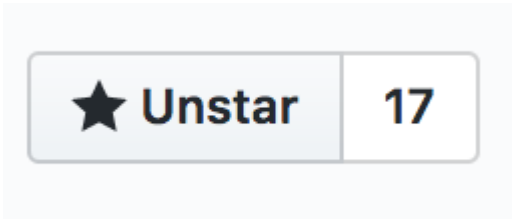
https://github.com/

important!!

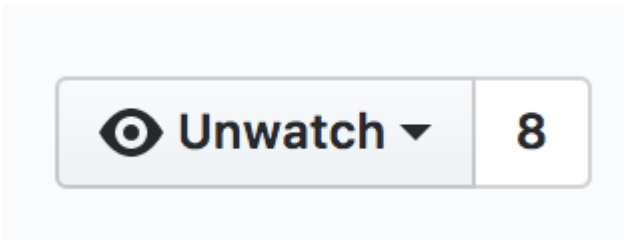
- 가입할 `email`과 `username`은 멋지게
- private repo를 원한다면 $7/month

Important github User Interface

Star



watch



Set configuration

terminal

```
$ git config --global user.name "username"
$ git config --global user.email "github email address"
$ git config --global core.editor "vim"
$ git config --list
```

My First Repo

Let's make your first repo with github

My First Repo

```
$ git init
$ git add .
$ git commit -m "some commit"
```

After create new repo through github,

```
$ git remote add origin https://github.com/username/repo.git
$ git push -u origin master
```

My First Github Pages

github 저장소를 활용해 정적인 사이트 호스팅이 가능

`username`.github.io
http://tech.kakao.com/
https://spoqa.github.io/

sample index page

After create new repo throuch github,

```
$ git clone https://github.com/username/username.github.io.git
```

Create New file `index.html`

```
$ git add .
$ git commit -m "first page"
$ git push -u origin master
```

sample index page

```
<!doctype html>
<html>
 <head>
  <meta charset="utf-8">
  <title>My first gh page</title>
 </head>
 <body>
  <h1>Home</h1>
  <p>Hello, there!</p>
 </body>
</html>
```

Static Site Generator

- Jekyll: Ruby 기반 정적인 블로그 생성기
  - 설치와 사용이 쉬움
  - 사용자가 많았음
- Hugo: Golang 기반 정적인 블로그 생성기
  - 빠른 속도로 사이트를 생성
  - 사용자 증가 중
- Hexo: Node.js 기반 정적인 블로그 생성기
  - Node.js를 안다면 커스터마이즈가 쉬움
  - 빠른 속도로 사용자 증가 중

Recommand

`Jekyll` > `Hugo` > `Hexo`

Let's use Hexo

Requirements

1. git
2. node.js(https://nodejs.org/en/)

```
$ npm install -g hexo-cli
```

Init hexo project

```
$ hexo init <folder>
$ cd <folder>
$ npm install
```

clean && generate static files

```
$ hexo clean && hexo generate
```

Run hexo server

```
$ hexo server
```

deploy

```
$ npm install hexo-deployer-git --save
```

```
deploy:
  type: git
  repo: <repository url>  branch: [branch] #published
  message:
```

What is branch?

What is branch?

What is branch?

분기점을 생성하고 독립적으로 코드를 변경할 수 있도록 도와주는 모델

ex)

master branch

```
print('hello world!')
```

another branch

```
for i in range(1,10):
    print('hello world for the %s times!' % i)
```

Branch

Show available local branch

```
$ git branch
```

Show available remote branch

```
$ git branch -r
```

Show available All branch

```
$ git branch -a
```

Branch

Create branch

```
$ git branch stem
```

Checkout branch

```
$ git checkout stem
```

Create & Checkout branch

```
$ git checkout -b new-stem
```

make changes inside readme.md

```
$ git commit -a -m 'edit readme.md'
```

```
$ git checkout master
```

merge branch

```
$ git merge stem
```

Branch

delete branch

```
$ git branch -D stem
```

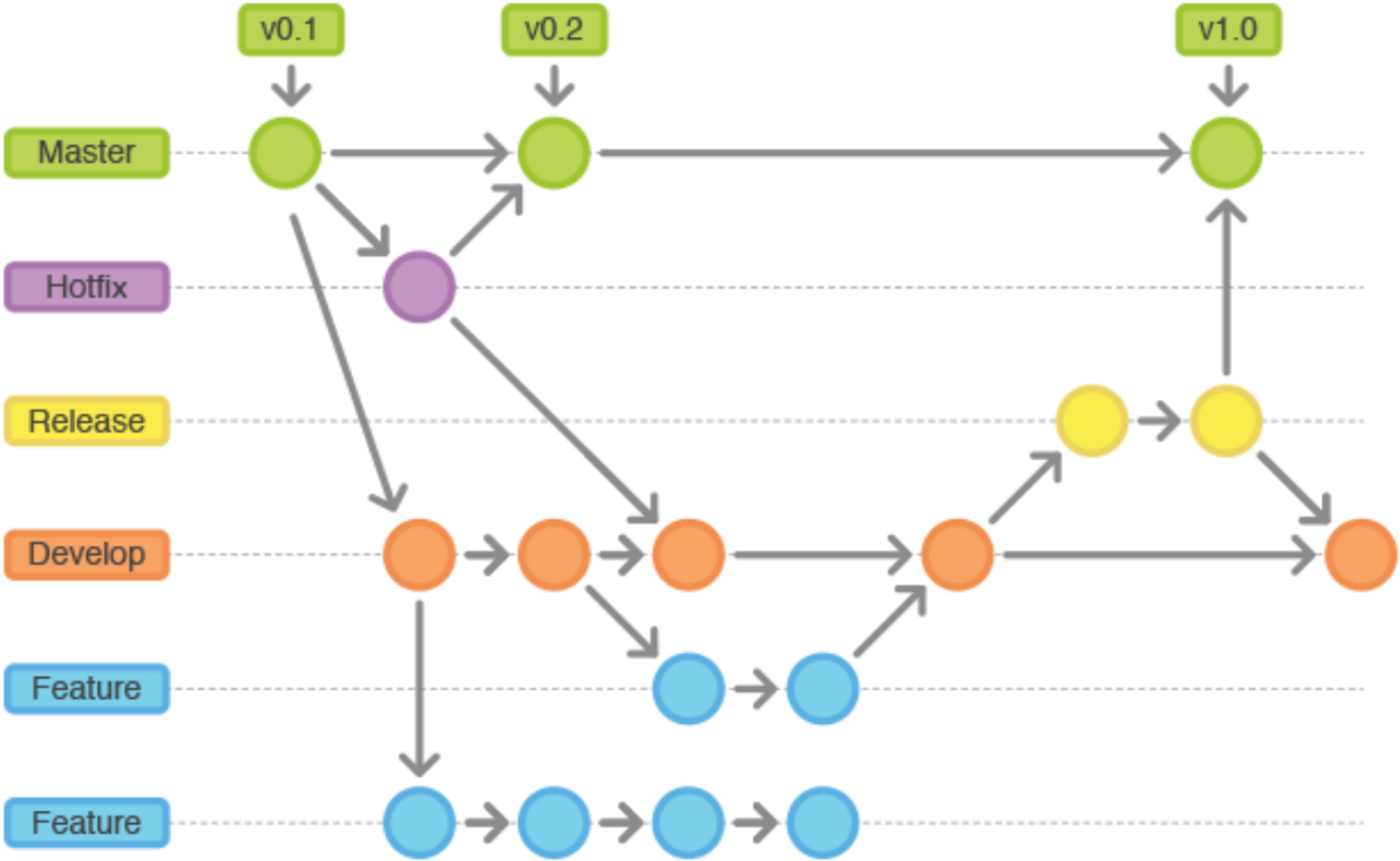
push with specified remote branch

```
$ git push origin stem
```

see the difference between two branches

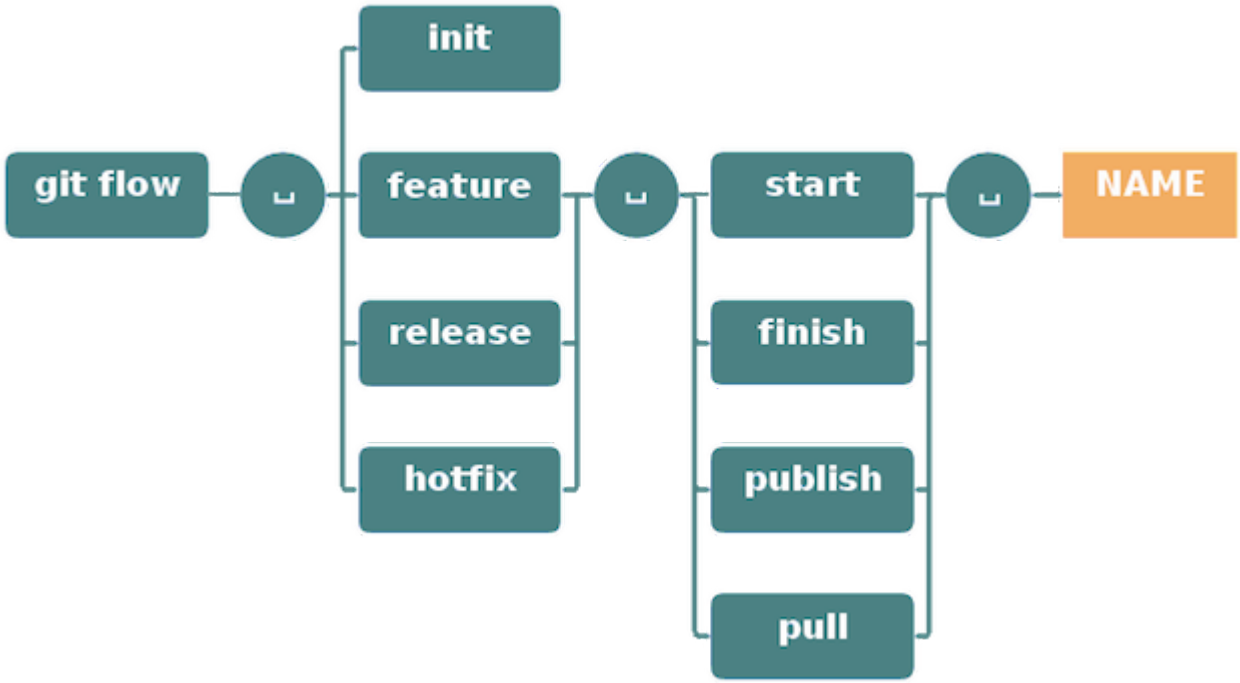
```
$ git diff master stem
```

git flow strategy



use git flow easily!

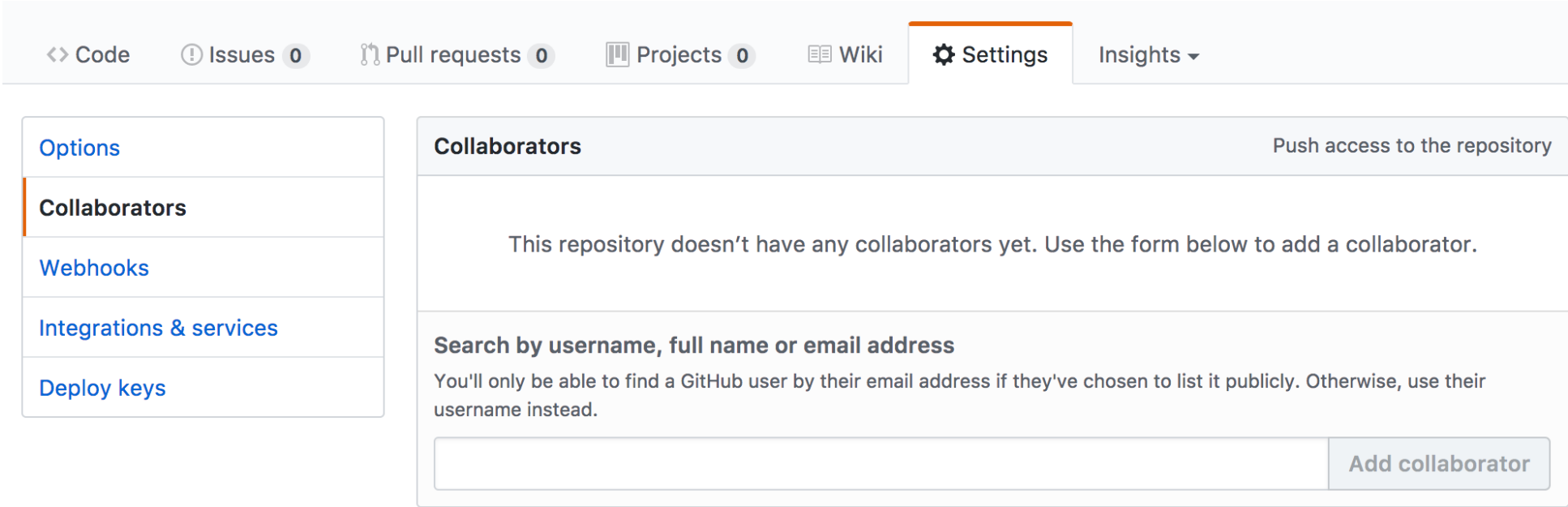
Link



Collaborate with your Co-worker

Method 1: Collaboration

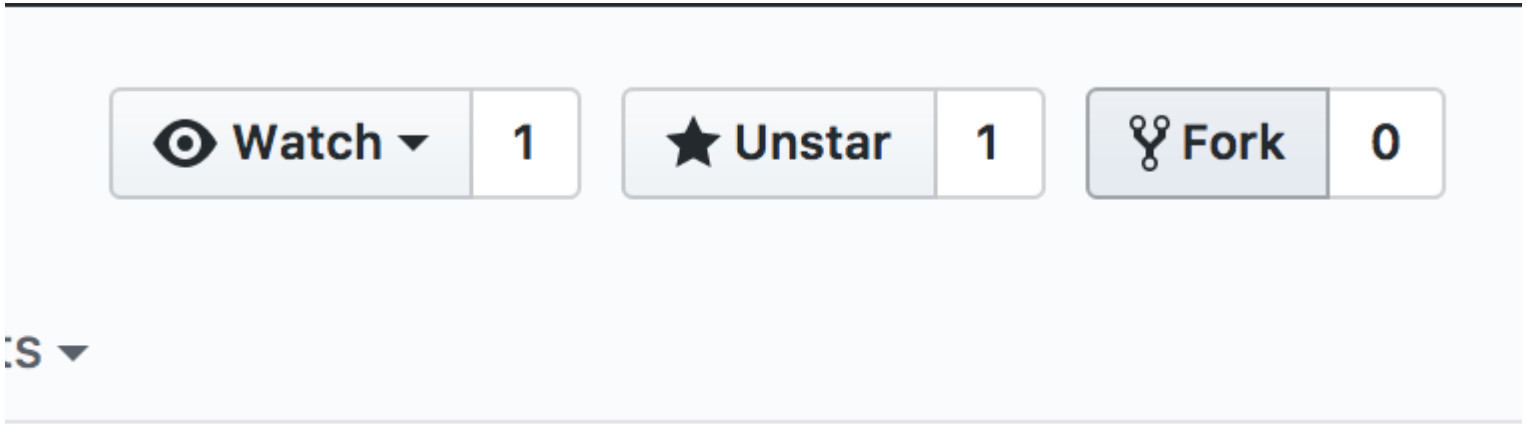
Add Collaborator



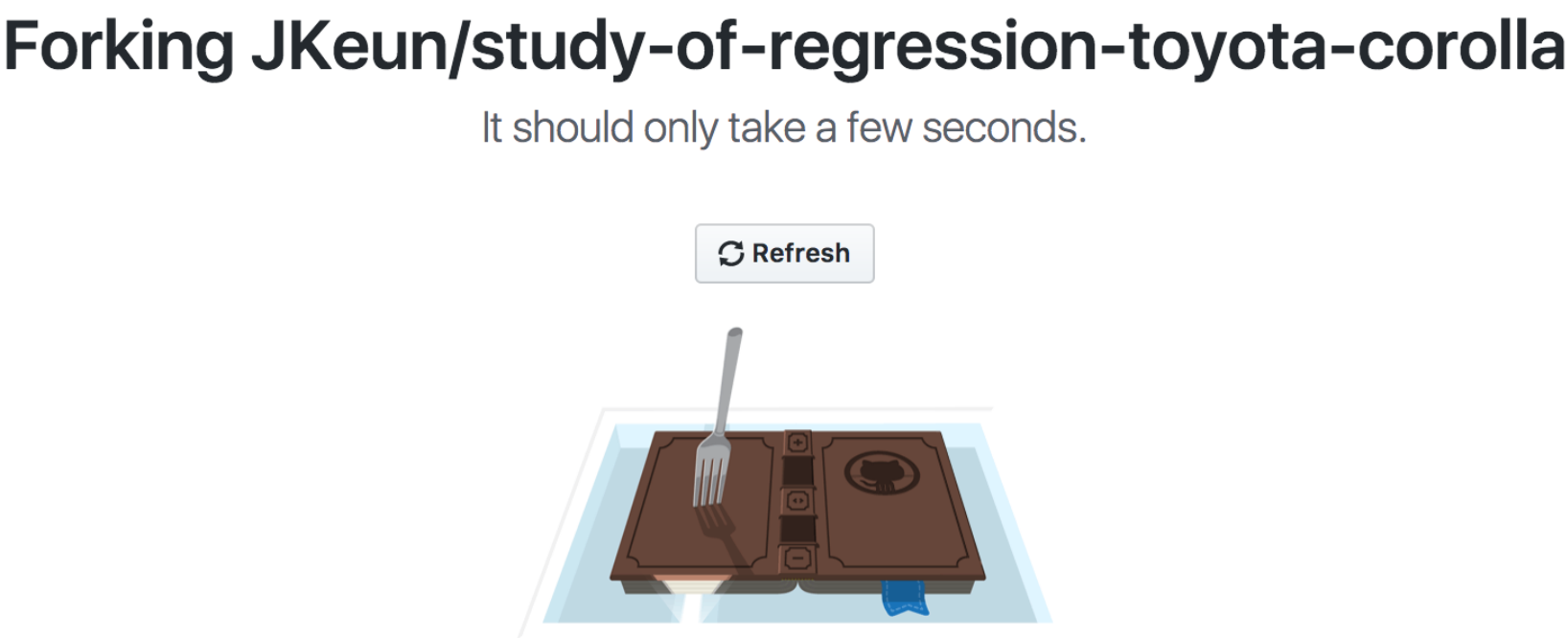
Collaboration

Add, Commit and Push like you own it.

Method 2: Fork and Merge



Fork and Merge



Fork and Merge

Fork and Merge

```
$ git clone https://github.com/username/forked-repo.git
```

Fork and Merge

```
$ git branch -a
```

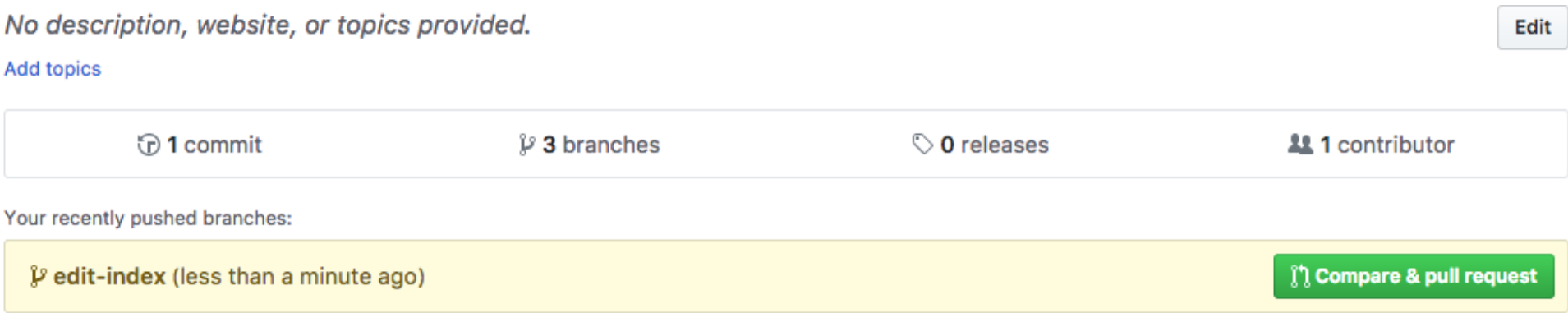
```
$ git checkout -b new-feature
```

Fork and Merge

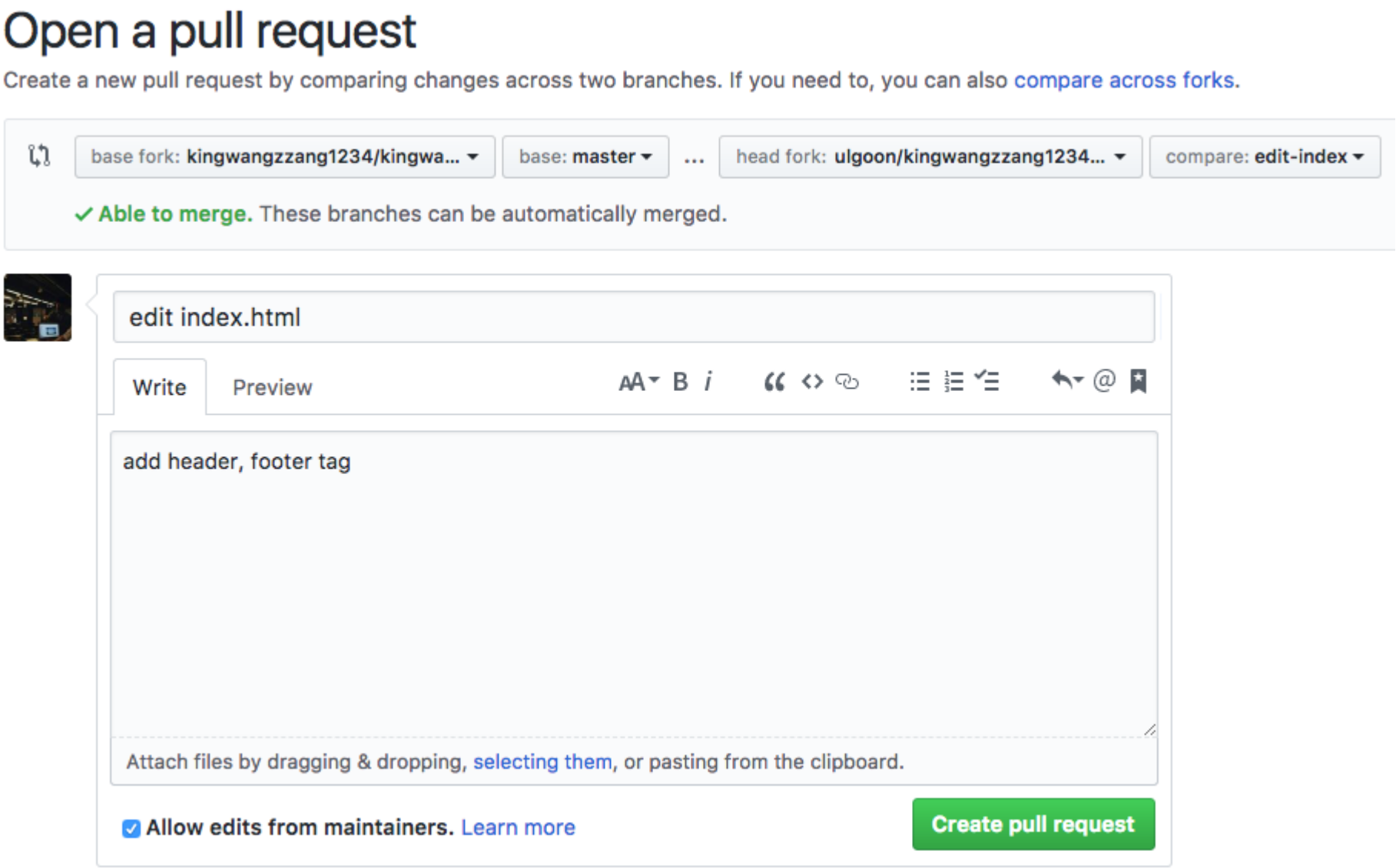
Make some change

```
$ git add file
$ git commit -m "commit message"
$ git push origin new-feature
```

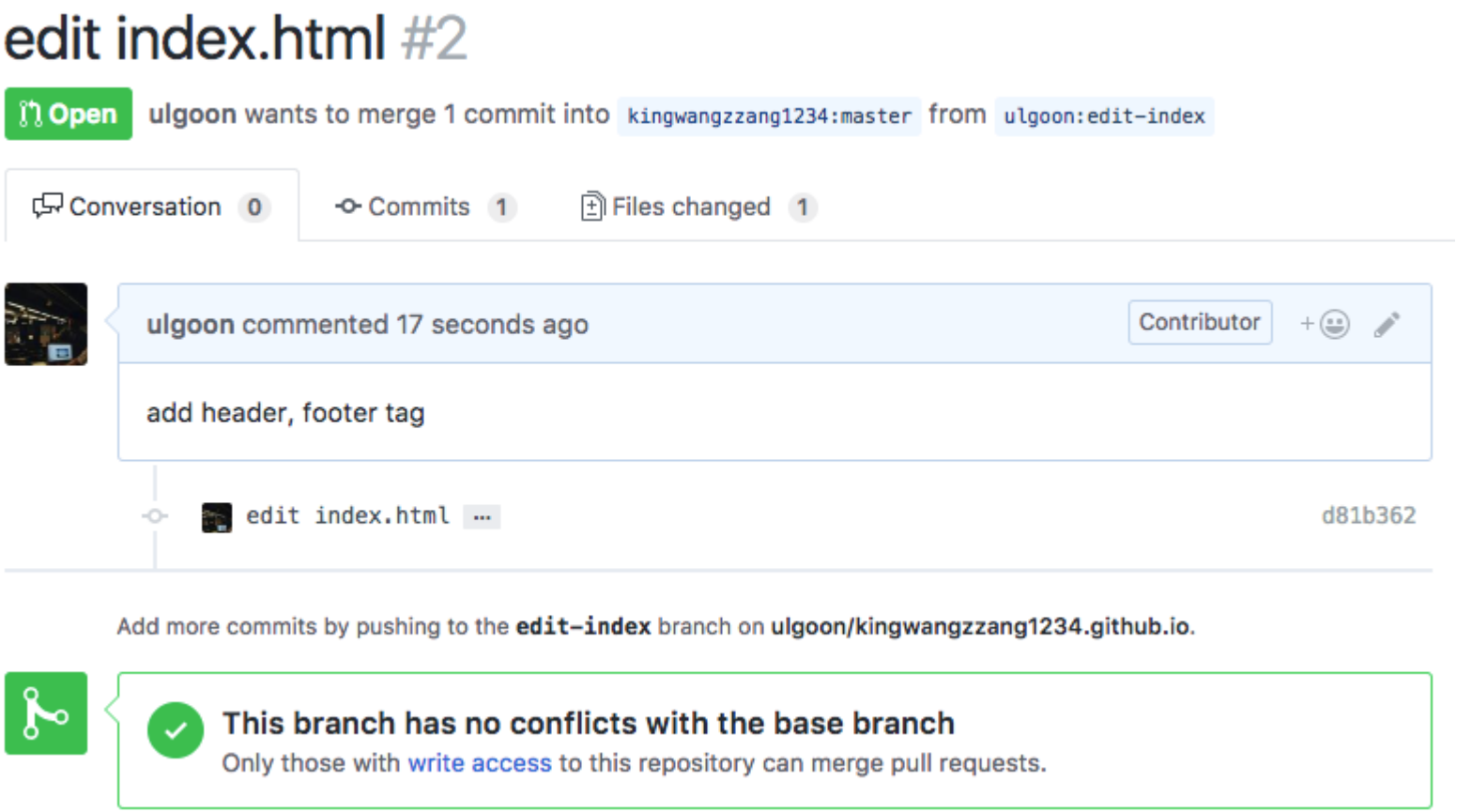
Fork and Merge



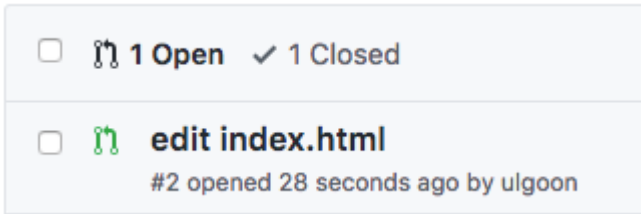
Fork and Merge



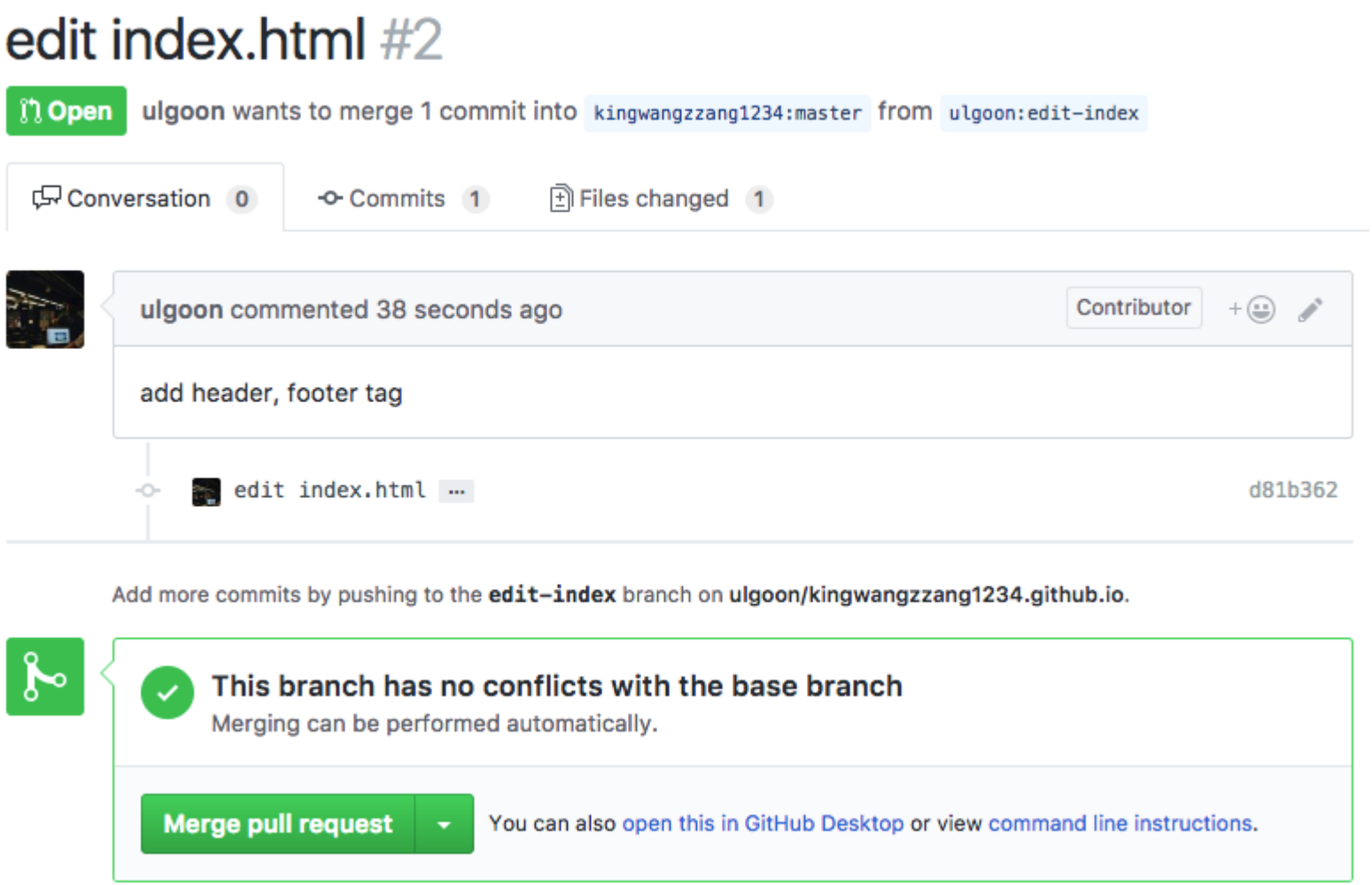
Fork and Merge



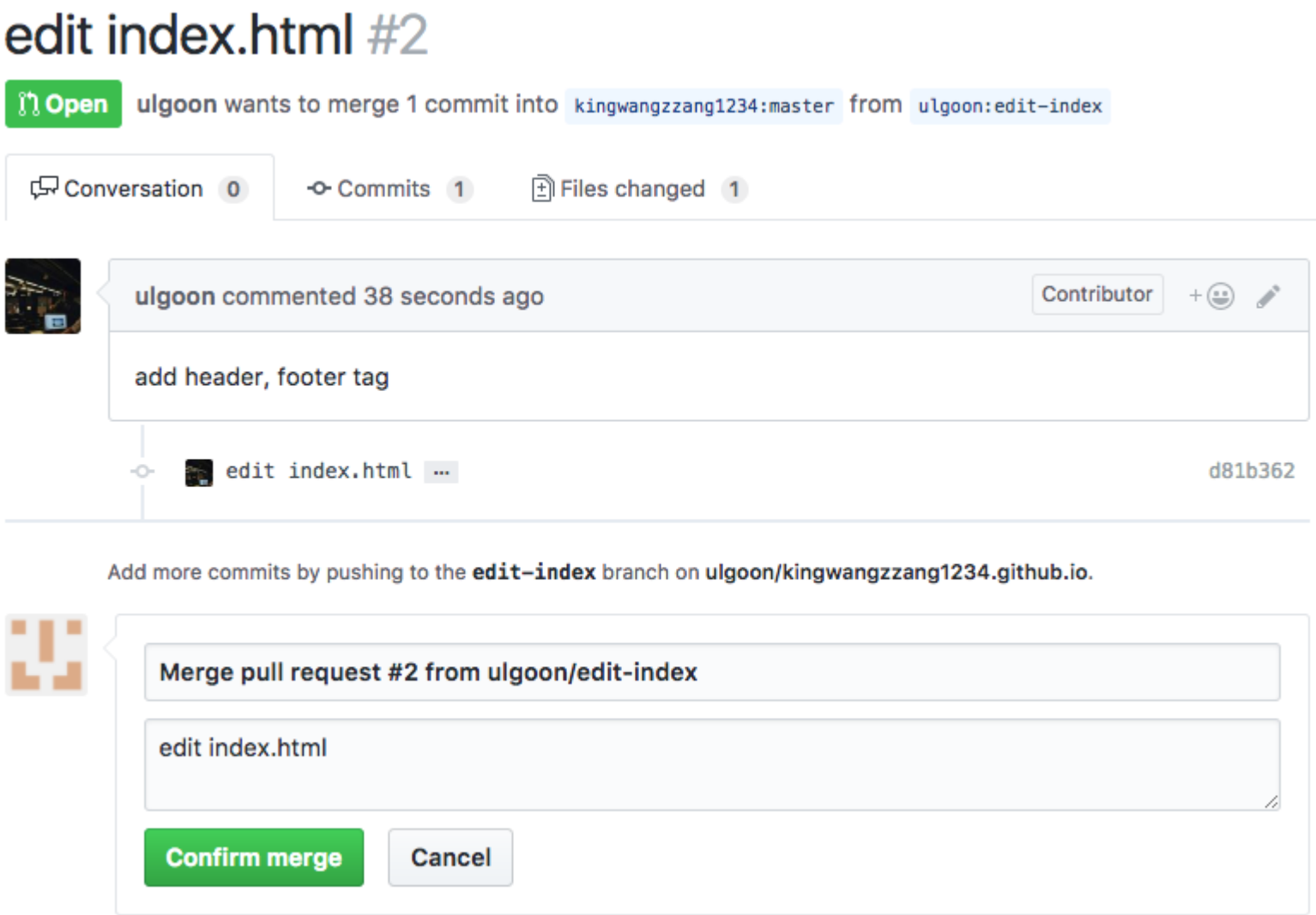
Fork and Merge



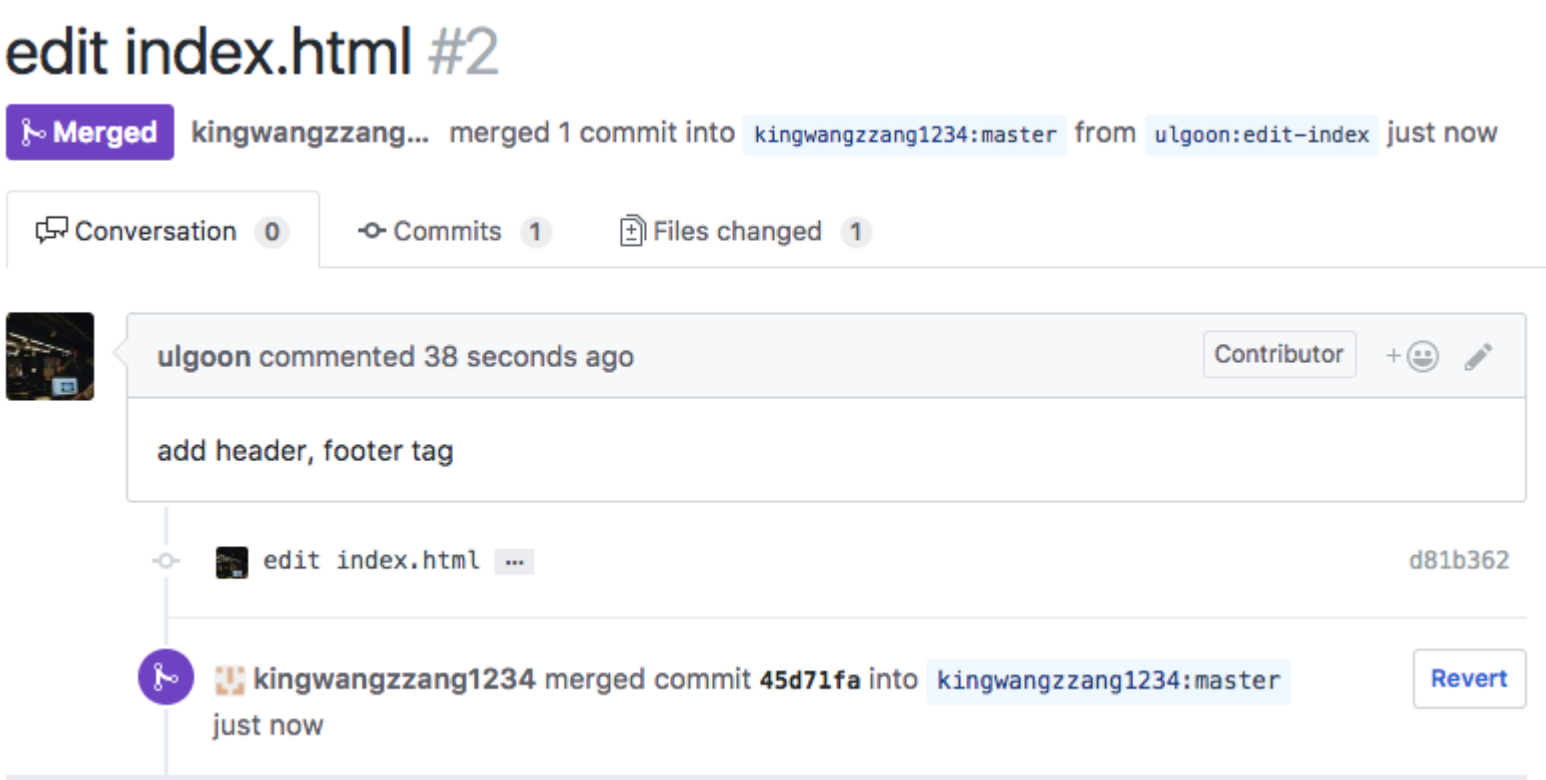
Fork and Merge



Fork and Merge



Fork and Merge



continuous pull

continuous pull

```
$ git remote add upstream
https://github.com/anotheruser/original-repo.git
```

```
$ git fetch upstream
$ git merge upstream/master
```