Fastcampus Data Science SCHOOL

Linux && git

Introduce

최우영

- Co-founder, CTO(disceptio)
- Solution Architect, Web Developer, Instructor
- Skills: Python, Golang, Julia, Node.js, Google tag manager ...

blog: https://blog.ulgoon.com/

github: https://github.com/ulgoon/

email: me@ulgoon.com

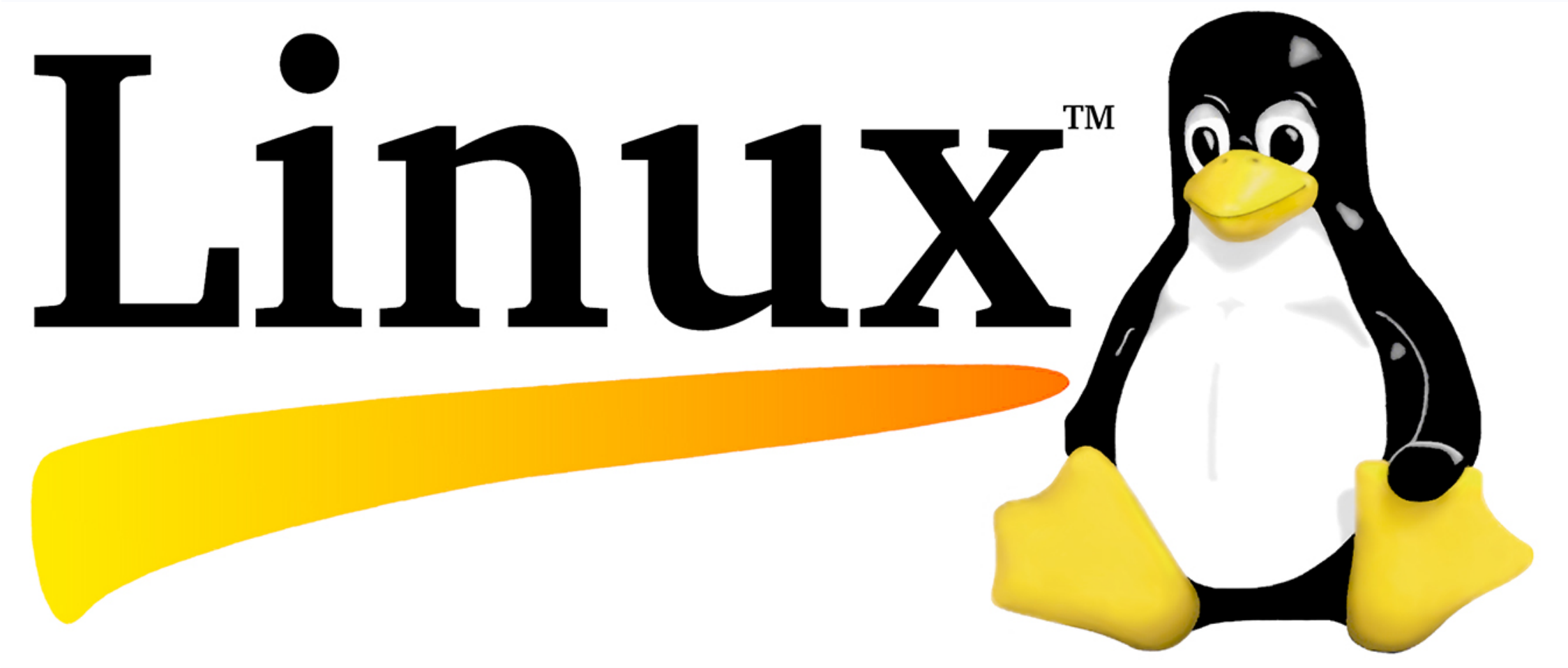
Goal

- Linux의 역사를 이해한다
- CLI에 대한 공포를 극복하고 Shell과 친구가 된다
- Linux Shell 커맨드를 학습하여 능숙하게 이를 활용할 수 있다
- Vim 텍스트 에디터를 통해 파일을 작성하고 매크로를 만들 수 있다

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

Linux



Before Linux

- 1965년 데니스 리치, 켄 톰슨 외 x명이 AT&T Bell 연구소에서 PDP-7 기반 어셈블리어로 작성한 UNIX를 개발

Before Linux

- 1973년 데니스 리치와 켄 톰슨이 C를 개발한 뒤, C 기반 UNIX 재작성

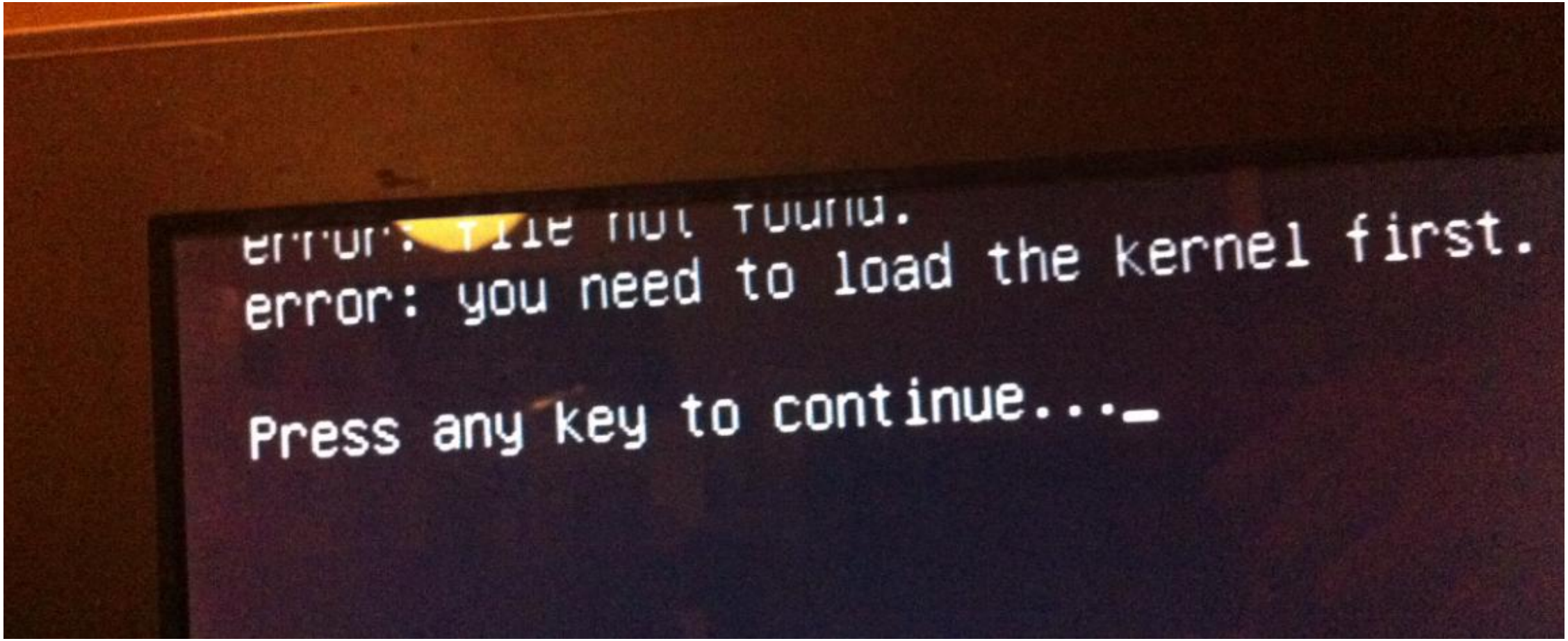
Before Linux

- 1984년 리차드 스톨먼이 오픈 소프트웨어 자유성 확보를 위한 GNU 프로젝트 돌입

Meaning of GNU

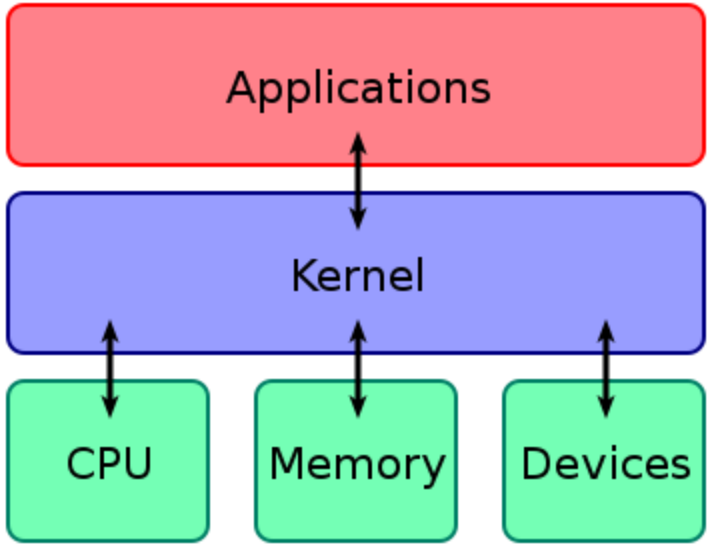
GNU == `G` NU is `N` ot `U` nix

Before Linux



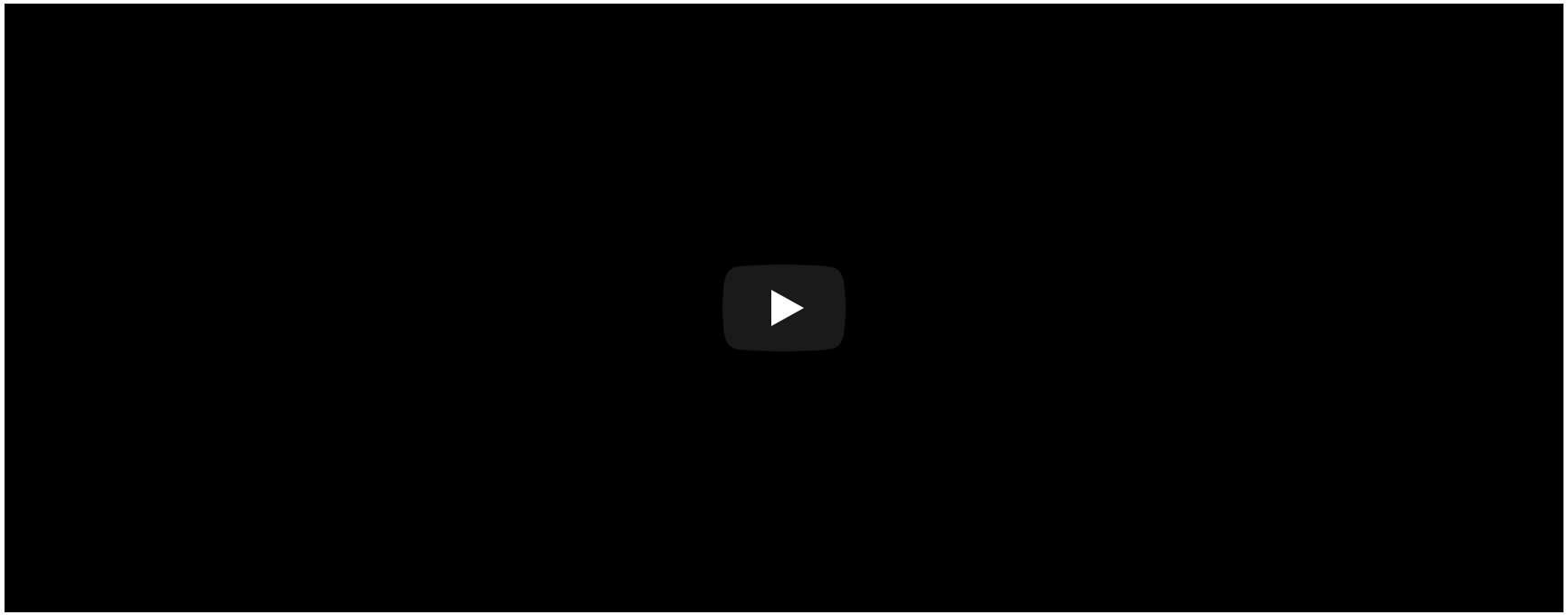
- But, GNU 프로젝트에는 커널이 없었고..

Kernel



- 하드웨어와 응용프로그램을 이어주는 운영체제의 핵심 시스템소프트웨어

Linus Torvalds



https://www.youtube.com/embed/IVpOyKCNZYw

- 헬싱키 대학생이던 리누스 토발즈는 앤디 타넨바움의 MINIX를 개조한 Linux를 발표
- 0.1 - bash(GNU Bourne Again SHell), gcc(UNIX 기반 C 컴파일러)

Linux

- 리누스 토발즈가 작성한 커널 혹은 GNU 프로젝트의 라이브러리와 도구가 포함된 운영 체제
- PC와 모바일, 서버, 임베디드 시스템 등 다양한 분야에서 활용
- Redhat, Debian, Ubuntu, Android 등 다양한 배포판이 존재

Shell

- 운영체제의 커널과 사용자를 이어주는 소프트웨어
- sh(Bourne Shell): AT&T Bell 연구소의 Steve Bourne이 작성한 유닉스 쉘
- csh: 버클리의 Bill Joy가 작성한 유닉스 쉘(C언어랑 비슷한 모양)
- bash(Bourne Again Shell): Brian Fox가 작성한 유닉스 쉘
    - 다양한 운영체제에서 기본 쉘로 채택
- zsh: Paul Falstad가 작성한 유닉스 쉘
    - sh 확장형 쉘
    - 현재까지 가장 완벽한 쉘

Let's learn bash

Shell Command Basic

```
$ cd documents

$ mkdir python - make directory python
$ cd python - change directory
$ cd .. - up to

$ ls
$ ls -al

$ touch hello.py - create hello.py
$ exit - terminate shell
```

chmod

> 파일의 권한을 설정할 때 사용

`drwxr-xr-x`

`d` or `-` : directory or file

(user)(group)(other)

`r` : read

`w` : write

`x` : execute

`-` : no permission

chmod

```
$ chmod [옵션] (8진수) (파일명)
```

8진수

0: 000

1: 001

2: 010

3: 011

4: 100

5: 101

6: 110

7: 111

Shell Command Basic

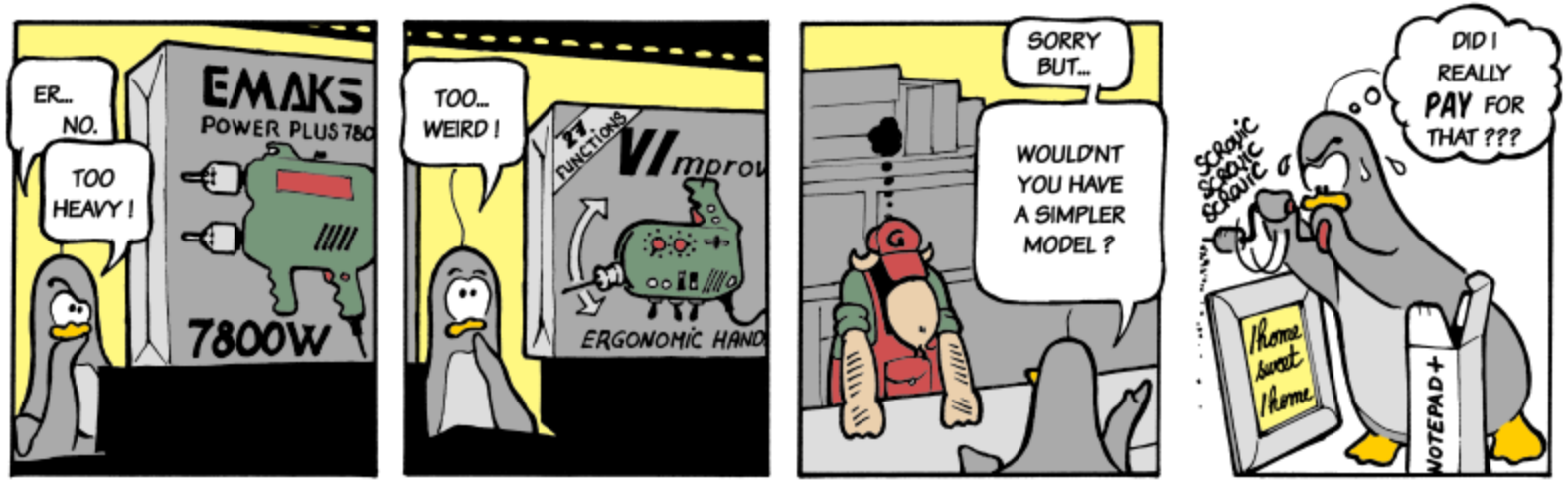
```
$ mv hello.py python
$ cp hello.py python

$ rm hello.py
$ rm -rf python/


$ python --version
$ python --help
```

Vim

Vim



- Vi improved Text Editor

Vim Basic

Command

```
h,j,k,l - move cursor
i - insert mode
v - visual mode
d - delete
y - yank
p - paste
u - undo
r - replace
$ - move end of line
^ - move start of line

:q - quit
:q! - quit w/o write(no warning)
:wq - write and quit

:{number} - move to {number}th line
```

write `hello.py` with Vim

`$ vim`

`$ vim hello.py`

`i`

`-- insert --`

type `print("hello python!")`

press `esc` to escape

`:wq`

`$ python hello.py`

copy & paste

`$ vim hello.py`

`v`

`-- visual --`

블록지정 후 `y`

`p`

press `esc` to escape

`:wq`

`$ python hello.py`

Use macro with Vim

`$ vim hello.py`

`qa` - a라는 매크로를 생성

`--recording--` 이 보이면 매크로 작성

`q` - 매크로 작성 종료

`@a` - a 매크로 실행

`10@a` - a 매크로 10회 실행


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 git 이라는 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

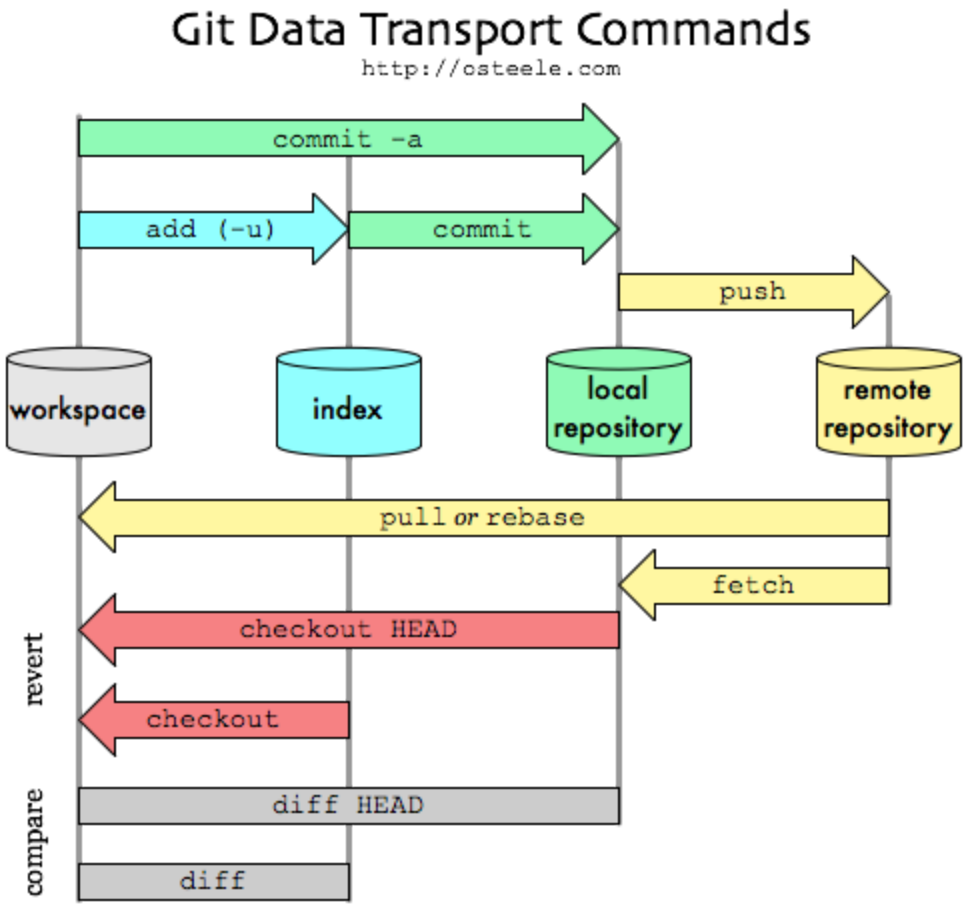
https://github.com/python/cpython
https://github.com/tensorflow/tensorflow

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

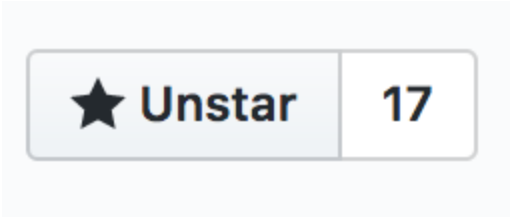
https://github.com/

important!!

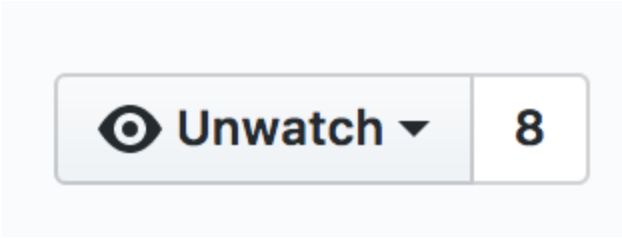
- 가입할 `email`과 `username`은 멋지게
- private repo를 원한다면 $7/month

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 origin master
```

start project with clone

- github에서 repo를 생성합니다.

```
$ git clone {repo address}
$ git add .
$ git commit
$ git push
```

How to move files

```
$ mv style.css static/css
```

-> deleted:

-> untracked files:

```
$ git mv style.css static/css
```

-> renamed:

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 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?

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
```

continuous pull

continuous pull

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

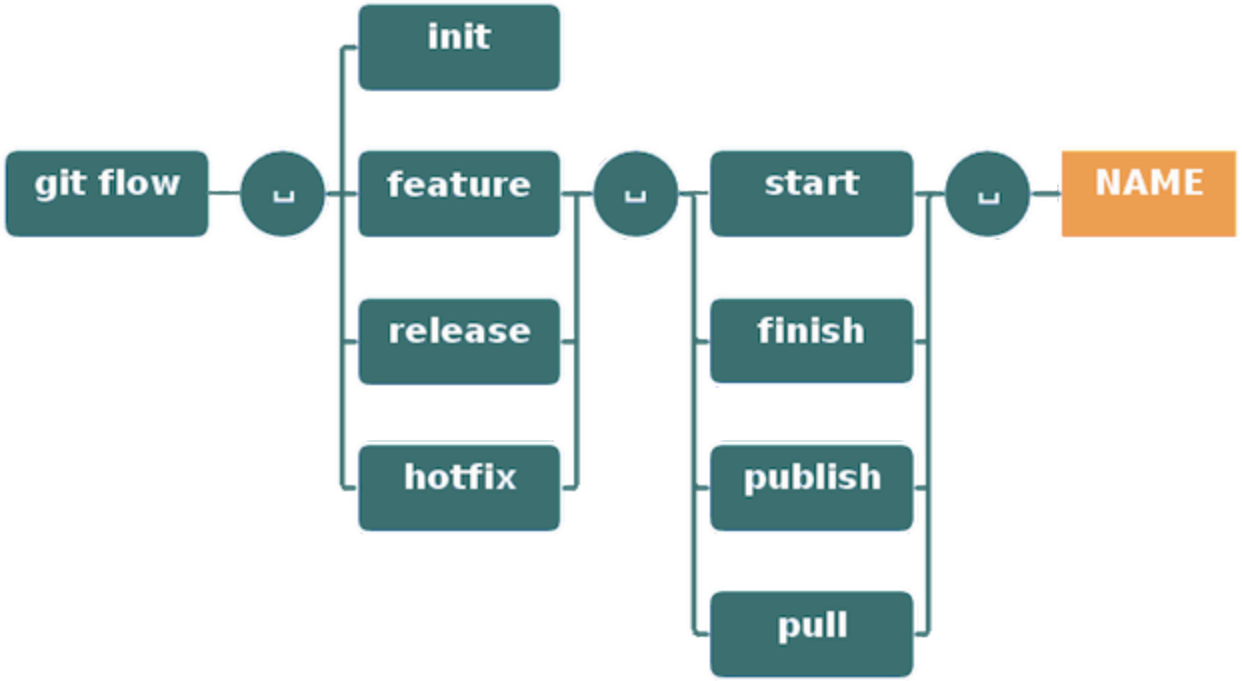
```
$ git fetch upstream
```

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

git flow strategy

use git flow easily!

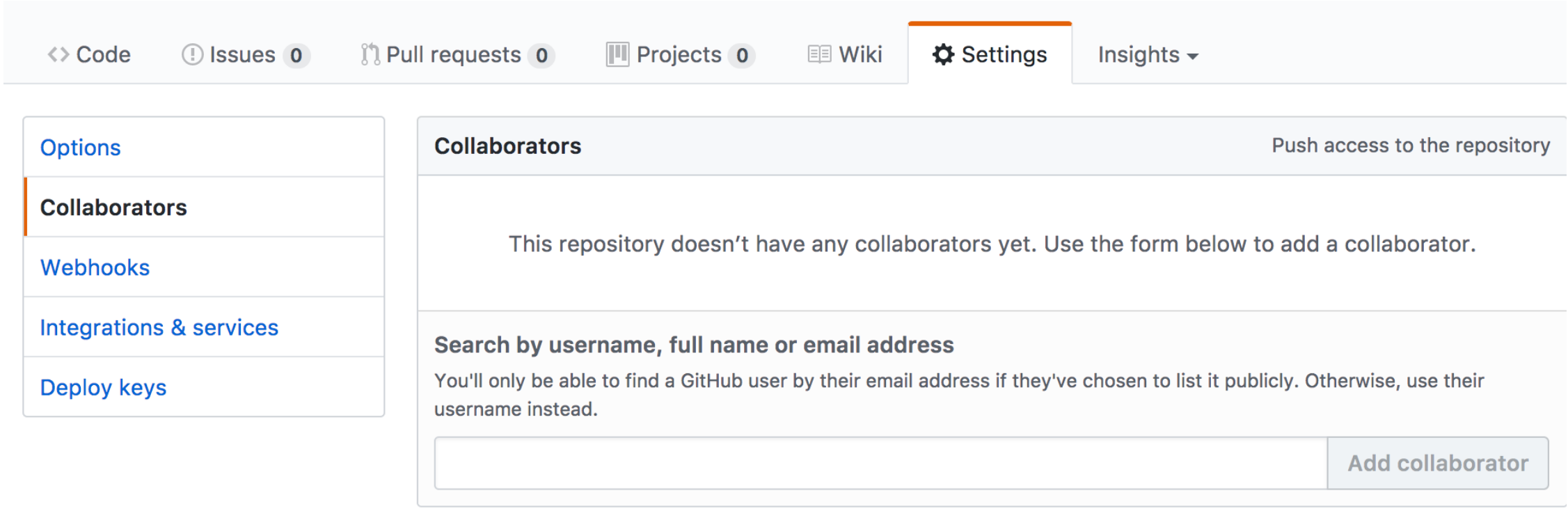
Link



Collaborate with your Co-worker

Method 1: Collaboration

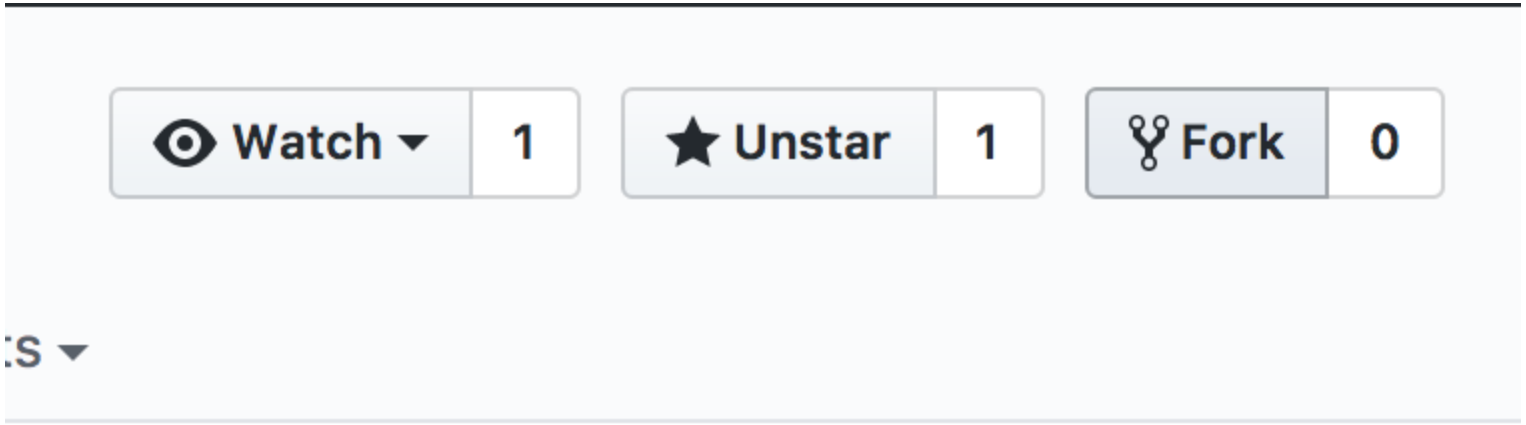
Add Collaborator



Collaboration

Add, Commit and Push like you own it.

Method 2: Fork and Merge

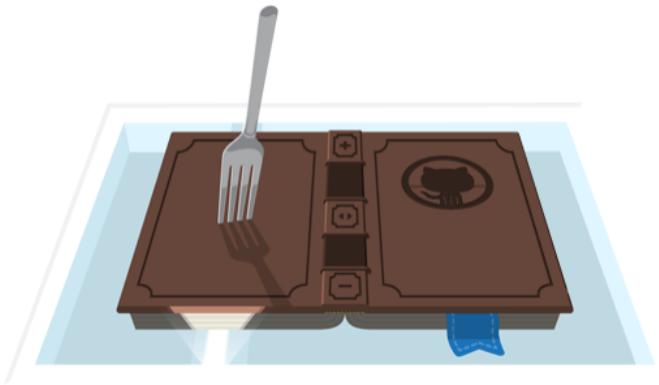


Fork and Merge

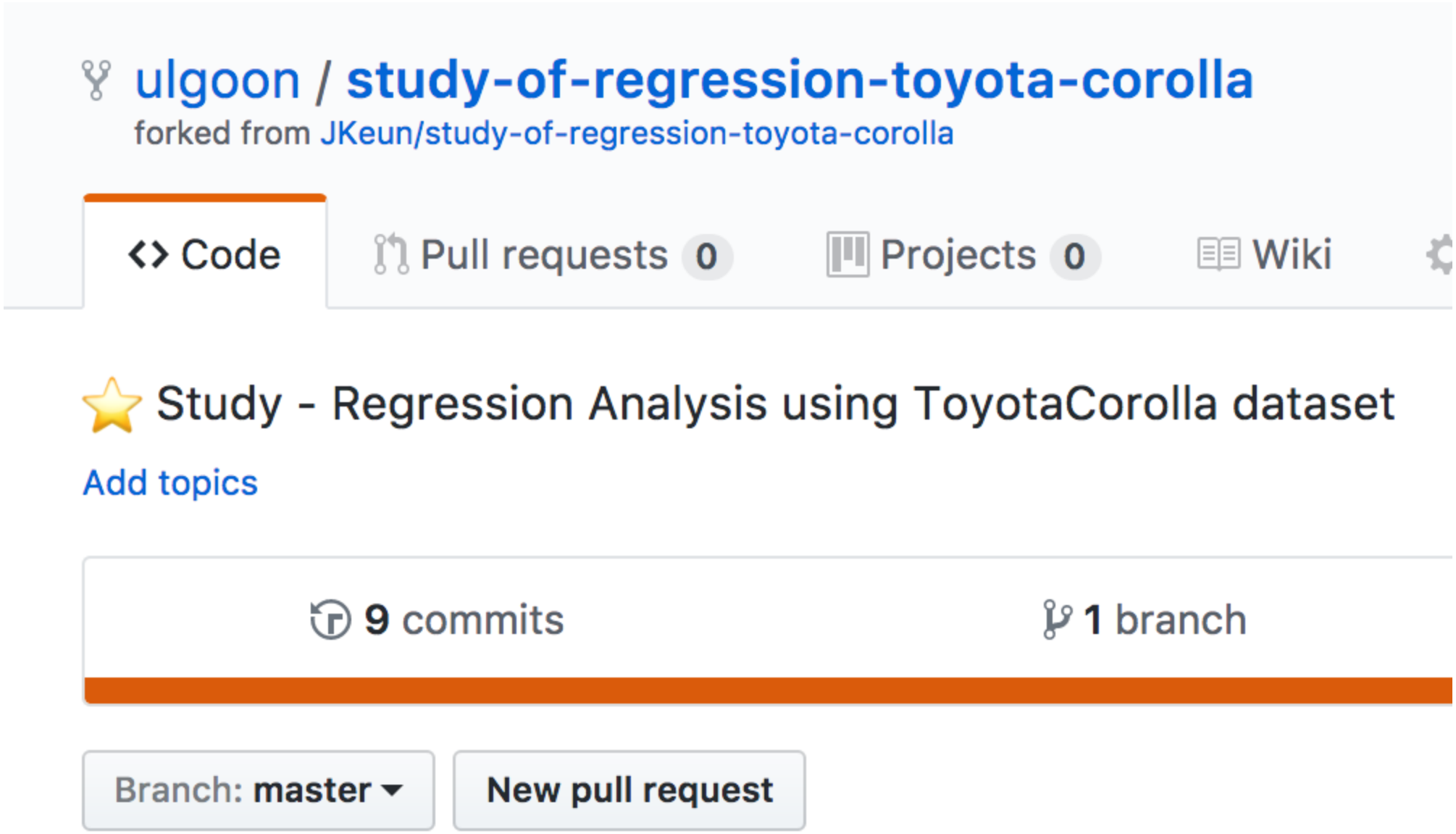
Forking JKeun/study-of-regression-toyota-corolla

It should only take a few seconds.

Refresh



Fork and Merge



Fork and Merge

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

Fork and Merge

```
$ git branch -a
```

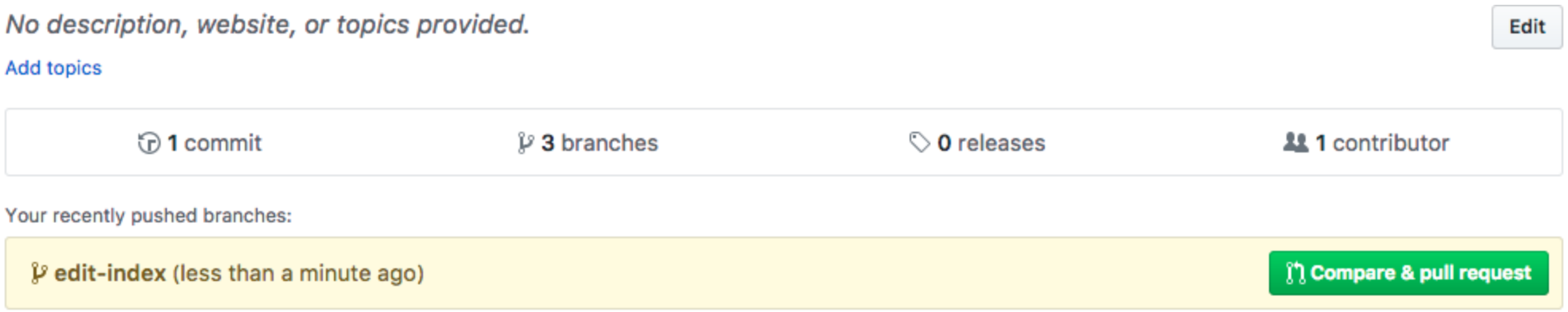
```
$ git checkout -b new-feature
```

Fork and Merge

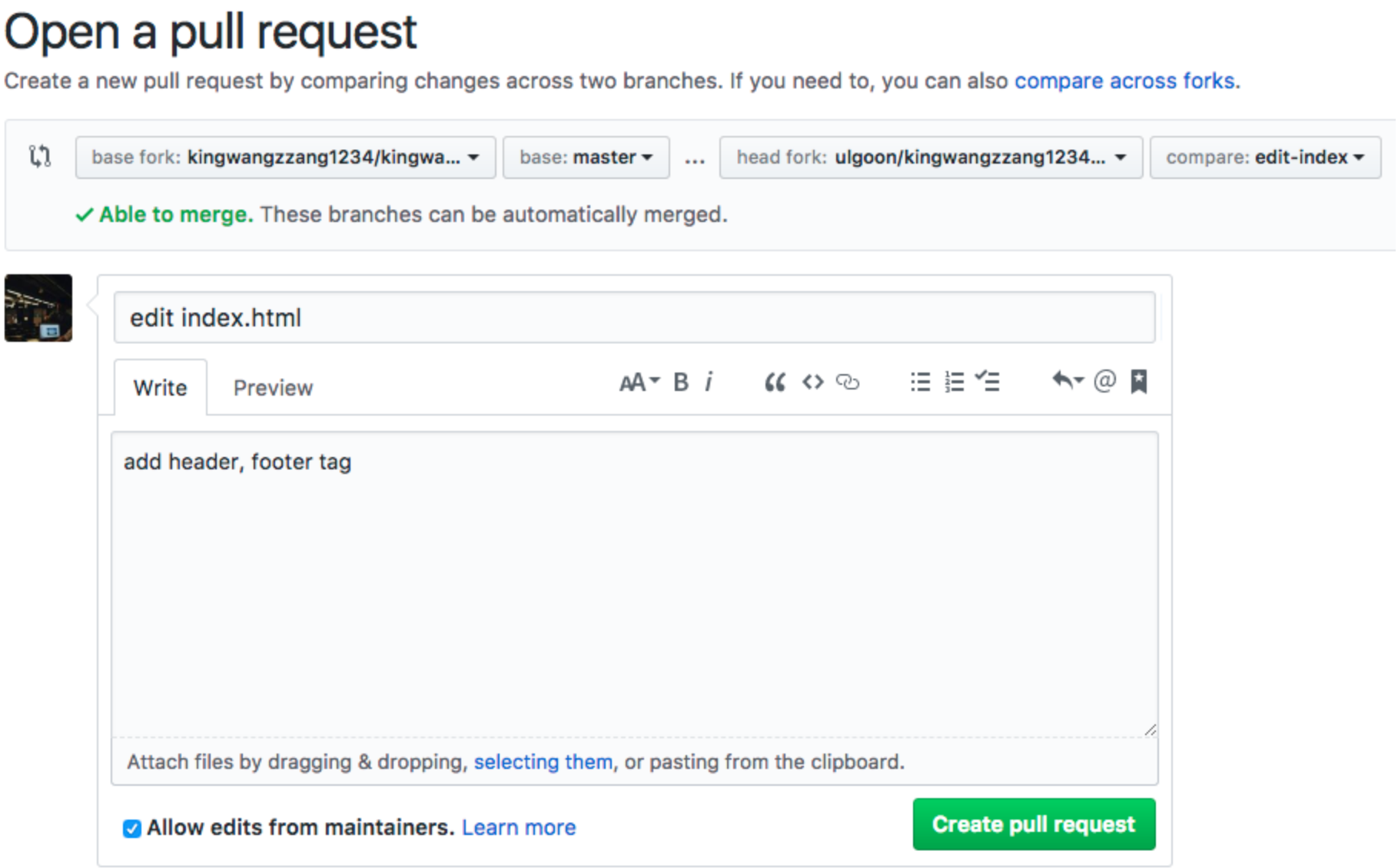
Make some change

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

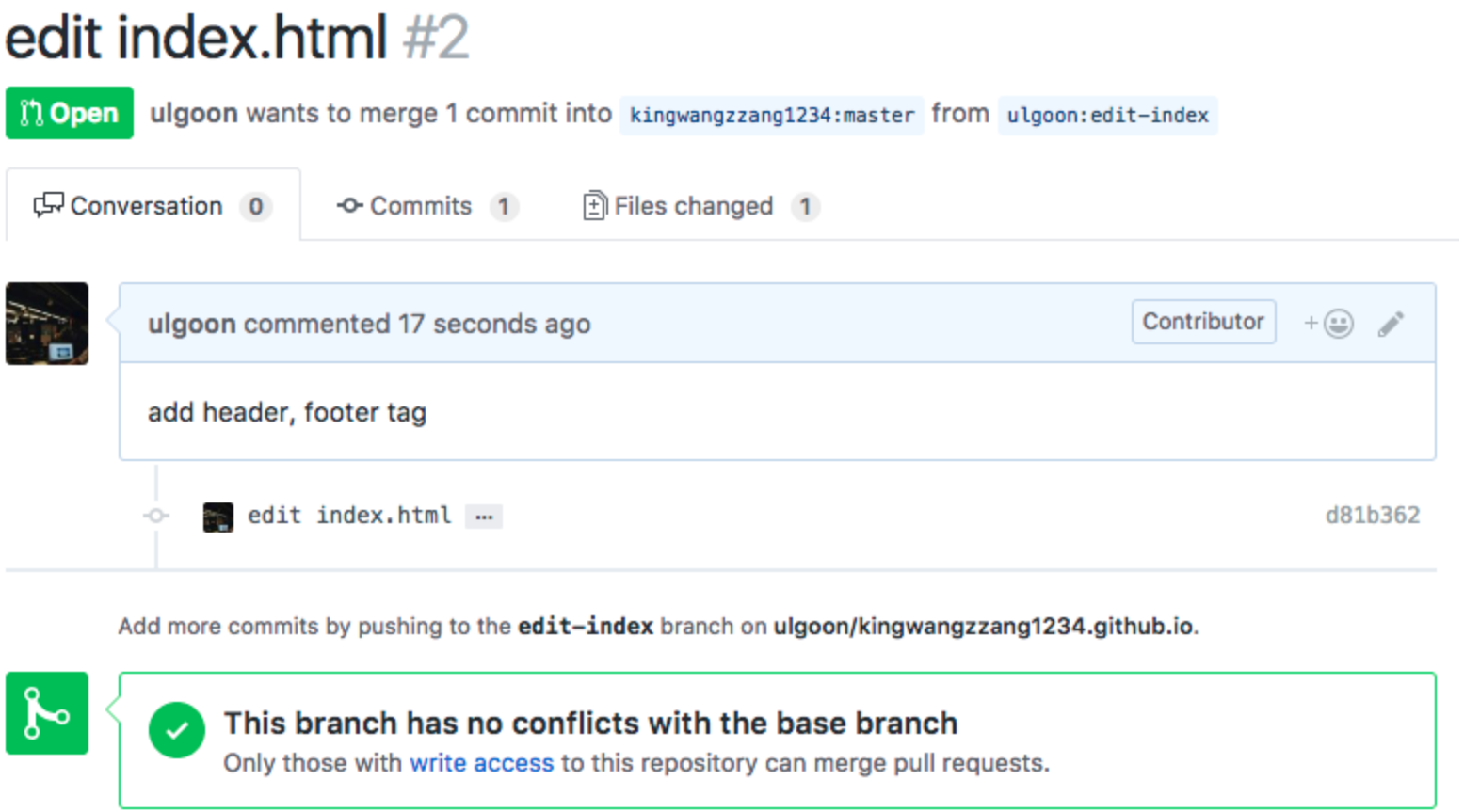
Fork and Merge



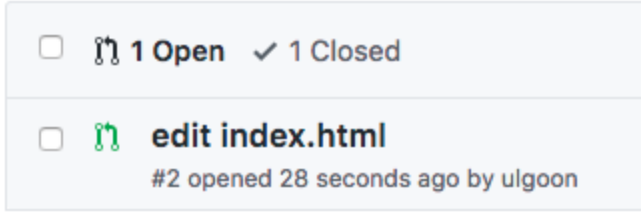
Fork and Merge



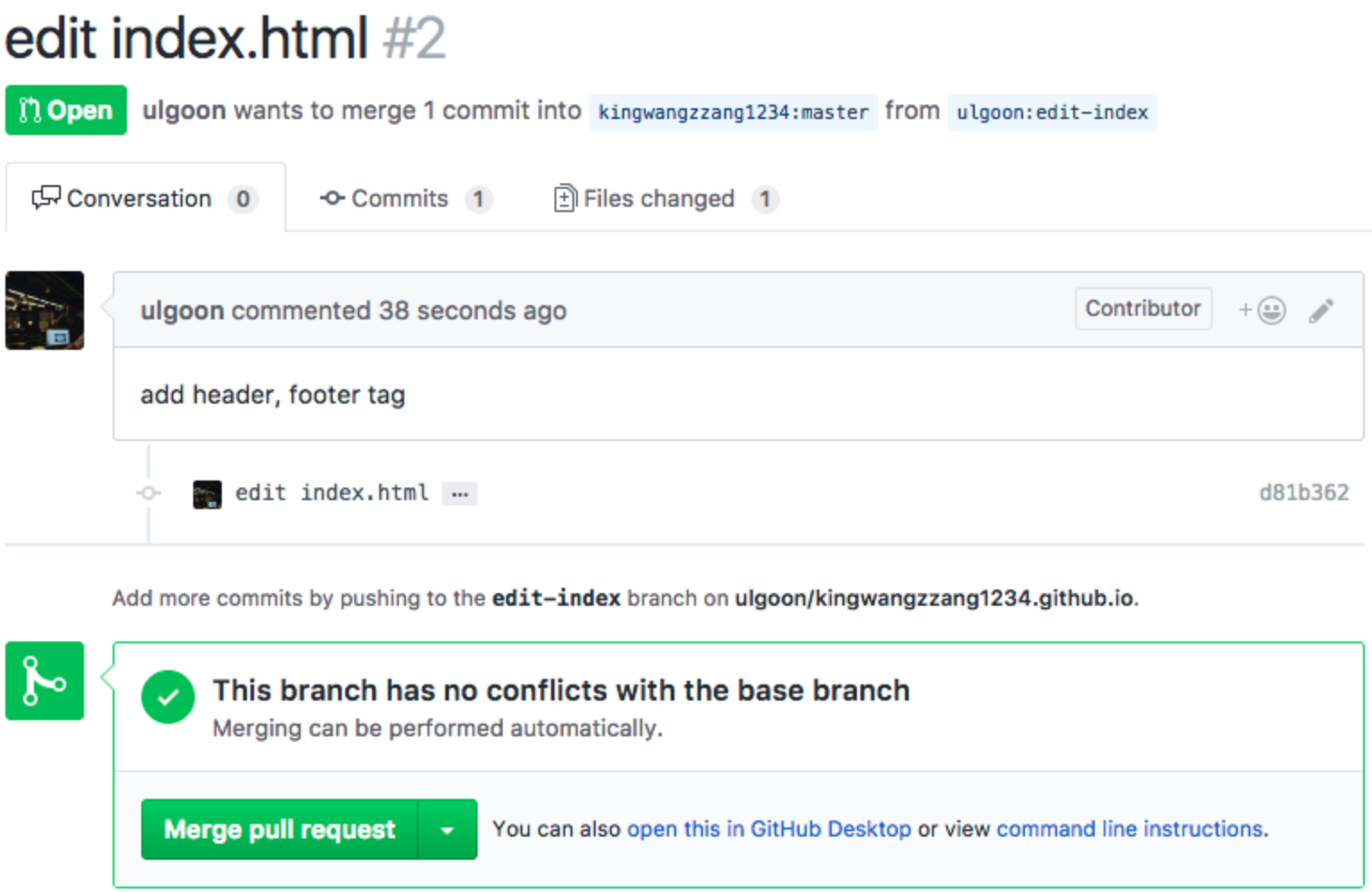
Fork and Merge



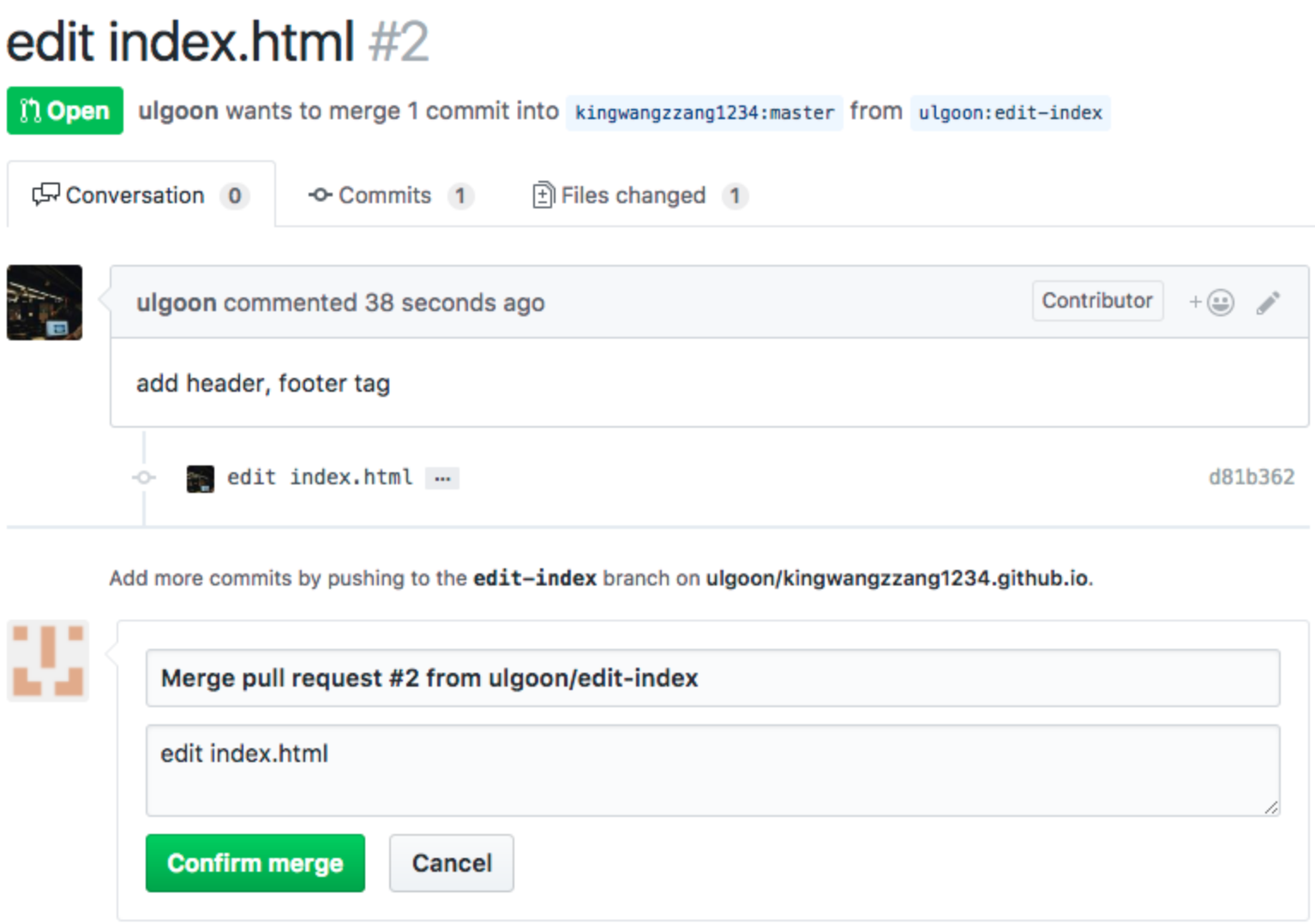
Fork and Merge



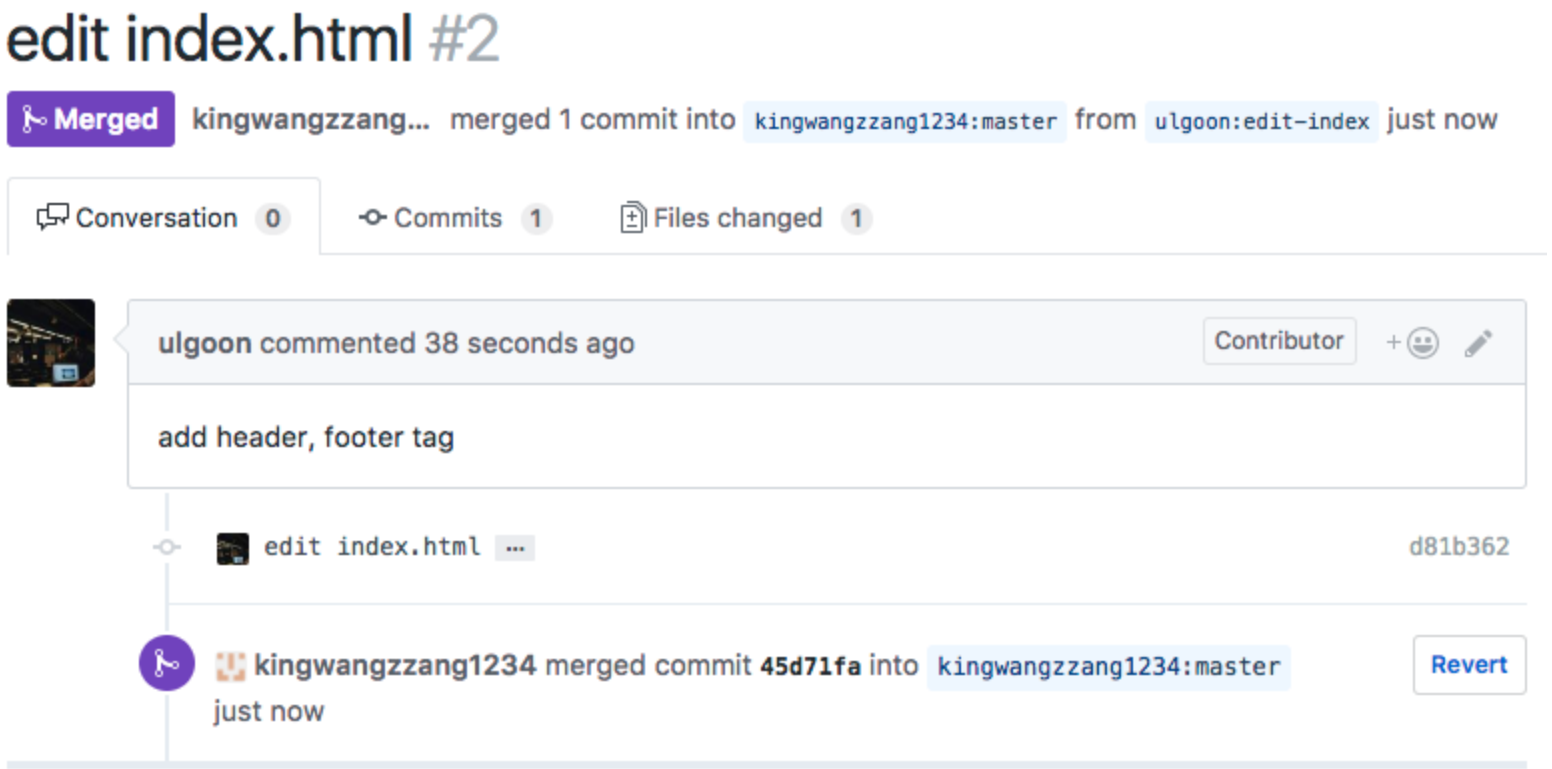
Fork and Merge



Fork and Merge



Fork and Merge



Process

PM's job
$ git clone [PM's repo addr]
$ git branch develop
$ git checkout develop
Do some works..(include add, commit, push on develop)

Dev's job
after PM's works, fork and clone into your local system.
(사실은 이 사이에 $ git remote add rmorigin {PM's repo addr})
$ git branch develop
$ git checkout develop
fetch, merge rmorigin/develop to your develop
Do some works..(include add, commit, push on develop)
create pull request on github.com/{you}/{cloned_repo}

PM's job
After receiving pull request mail,
talk about this works
merge into your repo

Do your project with co-worker

TODO

1. 각 분단의 한 가로줄이 한 팀입니다.
2. 각 팀의 센터에 있는 분이 PM입니다.
3. PM은 프로젝트 repo를 생성합니다. (git flow init 포함)
4. feature/project-init 브랜치 생성 후 index.html을 생성하여 develop 브랜치에 merge 합니다.
5. 나머지 동료는 repo fork, clone 후 git flow를 활용하여 작업을 실시한 뒤 pull request 를 생성합니다.
6. PM은 pull request시 코드 리뷰 후 main repo에 merge 합니다.

issue managing

do team-work one more time with issue