Git/GitHub

Git / GitHub Training

Instructor

송태웅 (Taeung Song, https://github.com/taeung)

- NIPA KOSS(Korean Open Source Software) Lab. Software Engineer
- Linux Kernel 프로젝트 Contributor

Git / GitHub Training

Training Mode

우리가 스마트폰을 (메뉴얼 없이) 사용하면서 이해하듯

Git 이란 프로그램도 일단 써보면서 이해 해보자 (실습위주 교육)

Training Mode

우리가 스마트폰을 (메뉴얼 없이) 사용하면서 이해하듯

Git 이란 프로그램도 일단 써보면서 이해 해보자 (실습위주 교육)

백문이불여일행 (百聞不如一行)

Git / GitHub Training

Training Mode

add commit push pull

show status log shortlog diff

clone checkout merge fetch rebase

reset stash blame cherry-pick submodule

Git 학습

"Deduction"

Git 의 이해

"Induction"

commit(역사 한단위) 만들기 commit 에 서명 포함하기

Github 에 밀어넣기 변화분 확인하기 브랜치 만들기

commit 수정하기 commit 을 메인테이너에게 전송하기

상황별 적용

야전경험
(시나리오 따라하며)

Git / GitHub Training

Table of Contents

Contents

Chapter 1. Git 소개

CHAPTER 01

Git 이란 ?

Git == History 관리 도구

CHAPTER 01

Git 이란 ?

Git 개발과정, 소스파일 등을 관리하는 도구

Histroy 관리가 되어 개발되어온 과정, 역사를 볼 수 있고,

특정시점으로 복구 가능

CHAPTER 01

Git 이란 ?

Ctrl+c, v 나 Alzip 압축파일 관리법

Source code management **tool**

그 사이에 뭐가 바뀌었는지
차이 (Diff) 를 알 수 없다 .

과제 1_ 최종 _2016_02_28.zip

과제 1_ 진짜최종 _2016_02_29.zip

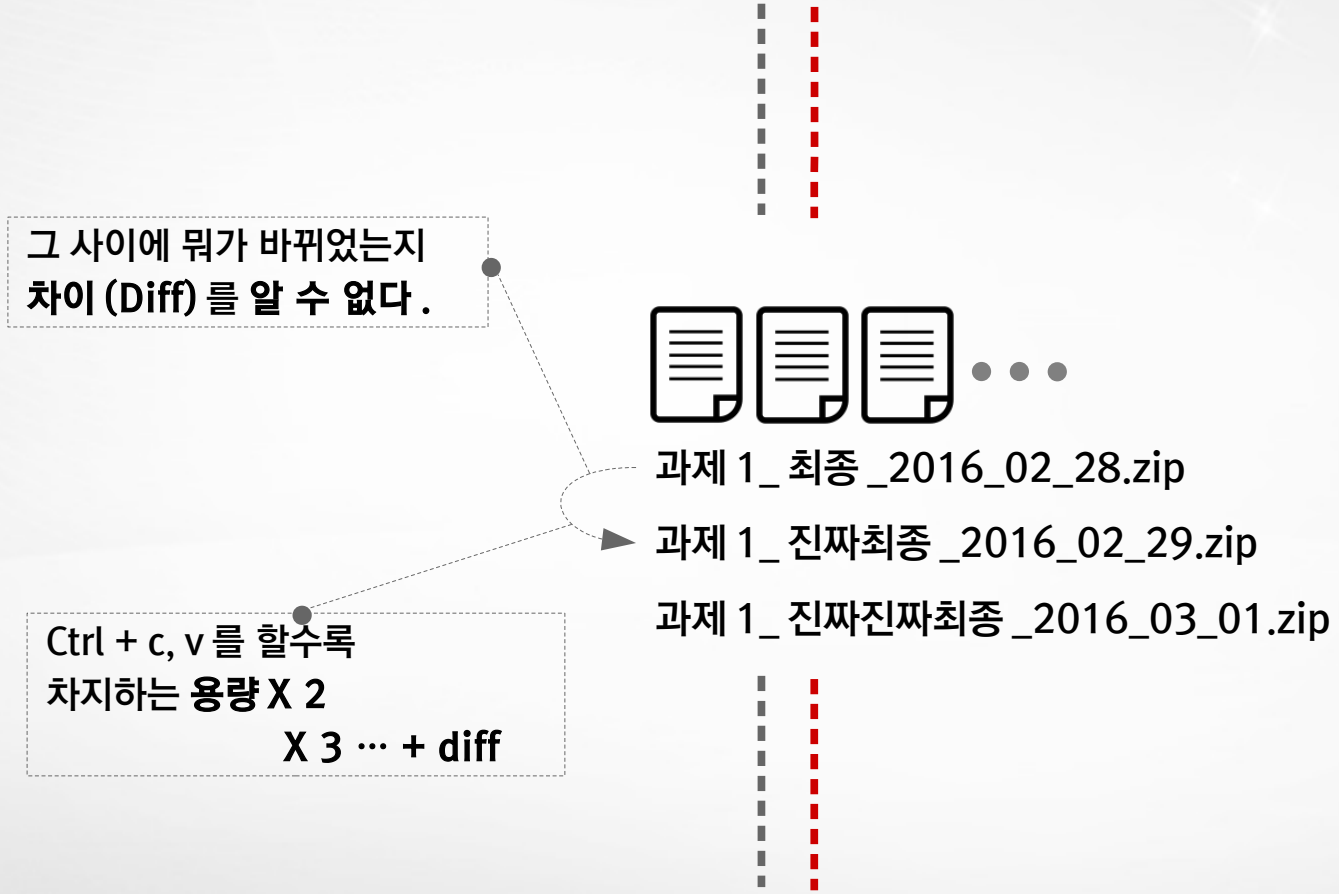
과제 1_ 진짜진짜최종 _2016_03_01.zip

CHAPTER 01

Git 이란 ?

Ctrl+c, v 나 Alzip 압축파일 관리법

Source code management **tool**

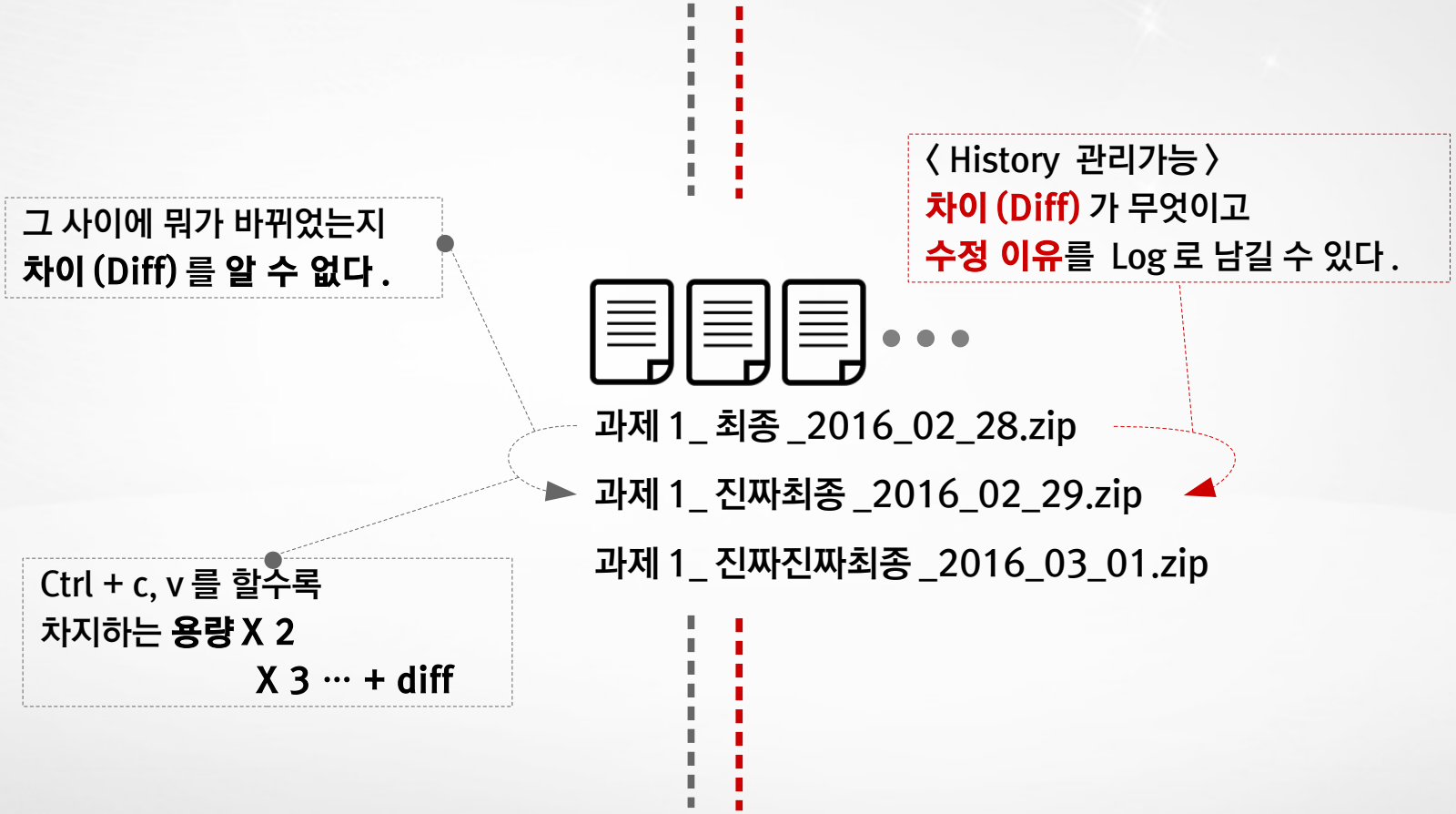


CHAPTER 01

Git 이란 ?

Ctrl+c, v 나 Alzip 압축파일 관리법

Source code management **tool**



CHAPTER 01

Git 이란 ?

Ctrl+c, v 나 Alzip 압축파일 관리법

Source code management **tool**

그 사이에 뭐가 바뀌었는지
차이 (Diff) 를 알 수 없다 .

〈 History 관리가능 〉
차이 (Diff) 가 무엇이고
수정 이유를 Log 로 남길 수 있다 .

과제 1_ 최종 _2016_02_28.zip

과제 1_ 진짜최종 _2016_02_29.zip

과제 1_ 진짜진짜최종 _2016_03_01.zip

Ctrl + c, v 를 할수록
차지하는 **용량 X 2**
X 3 … + diff

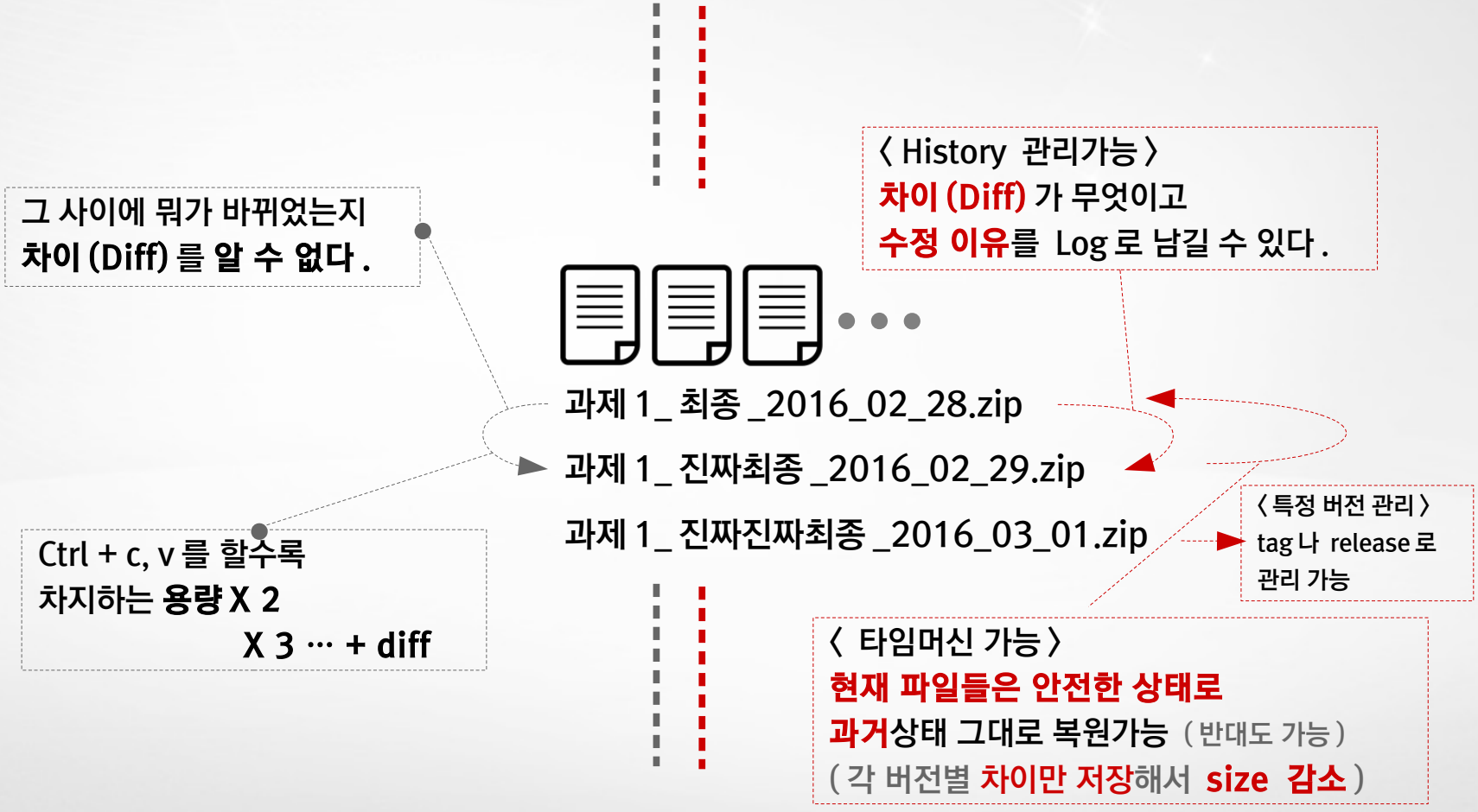
〈 타임머신 가능 〉
현재 파일들은 안전한 상태로
과거상태 그대로 복원가능 (반대도 가능)
(각 버전별 **차이만 저장**해서 **size 감소**)

CHAPTER 01

Git 이란 ?

Ctrl+c, v 나 Alzip 압축파일 관리법

Source code management **tool**



CHAPTER 01

Git 이란 ?

Ctrl+c, v 나 Alzip 압축파일 관리법

Git 배우는데 시간 소비하느니
Code 한줄이라도 더 개발 ..)

과제 1_ 최종 _2016_02_28.zip

과제 1_ 진짜최종 _2016_02_29.zip

과제 1_ 진짜진짜최종 _2016_03_01.zip

Source code management tool

좋은건 알겠는데 ..
Git 을 쓸 이유가 부족 ..

Wants 는 맞지만 Needs 는 아니야

Git / GitHub Training

How to use git

- Git 간단한 정의 / 기능
- Git 기본 실습 (How)
- Git 이해하기 (Why, What)
- Git 고급 (Advanced)
- Opensource (with Git)

Git / GitHub Training

Git 실습 (Basic)

Don't think about git,

just do git

시나리오대로 단계 단계 똑같이 따라 해보자 Git

CHAPTER 01

Git 실습 준비단계

예제소스 다운받기

- git-training-ex-v2.zip 압축풀고 폴더 열어두기
  https://www.dropbox.com/sh/9q2emkhxmyckoj6/AAA_H55BVhfRvGHOs9j7l9N2a?dl=1&pv=1

Git 설치하기

- Window 사용자라면 Git-bash 설치 https://git-scm.com/downloads
- Mac 사용자는 git-osx-installer 를 통한 설치 https://git-scm.com/downloads
- Linux 사용자는 # sudo apt-get install git

Editor 다운받기

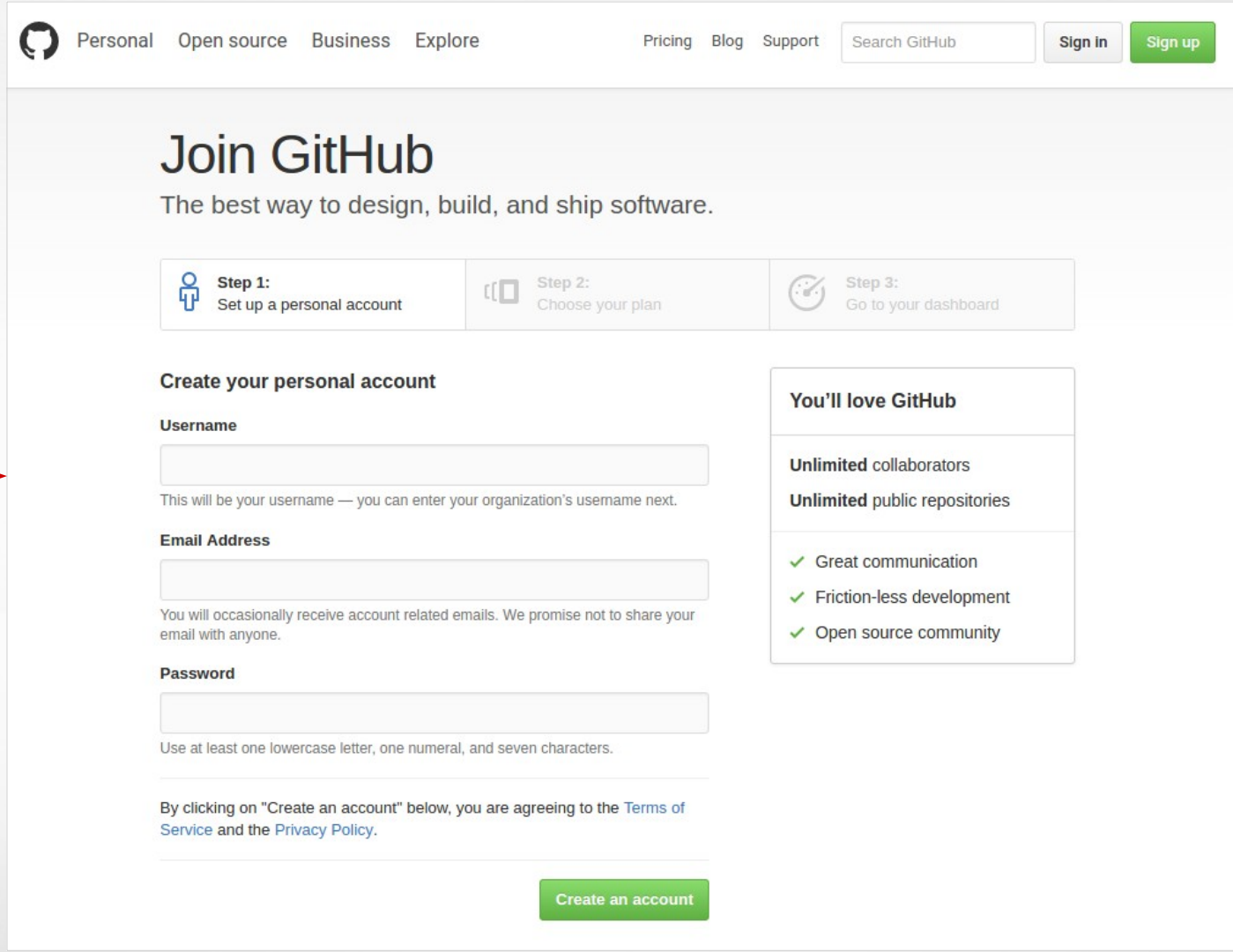
- 이미 사용중인 Editor 가 있다면 그대로 이용
- 사용중인 Editor 가 없다면 Atom 써보기 https://atom.io/

Github 회원가입

- 회원가입 https://github.com/join
- 메일 인증하기

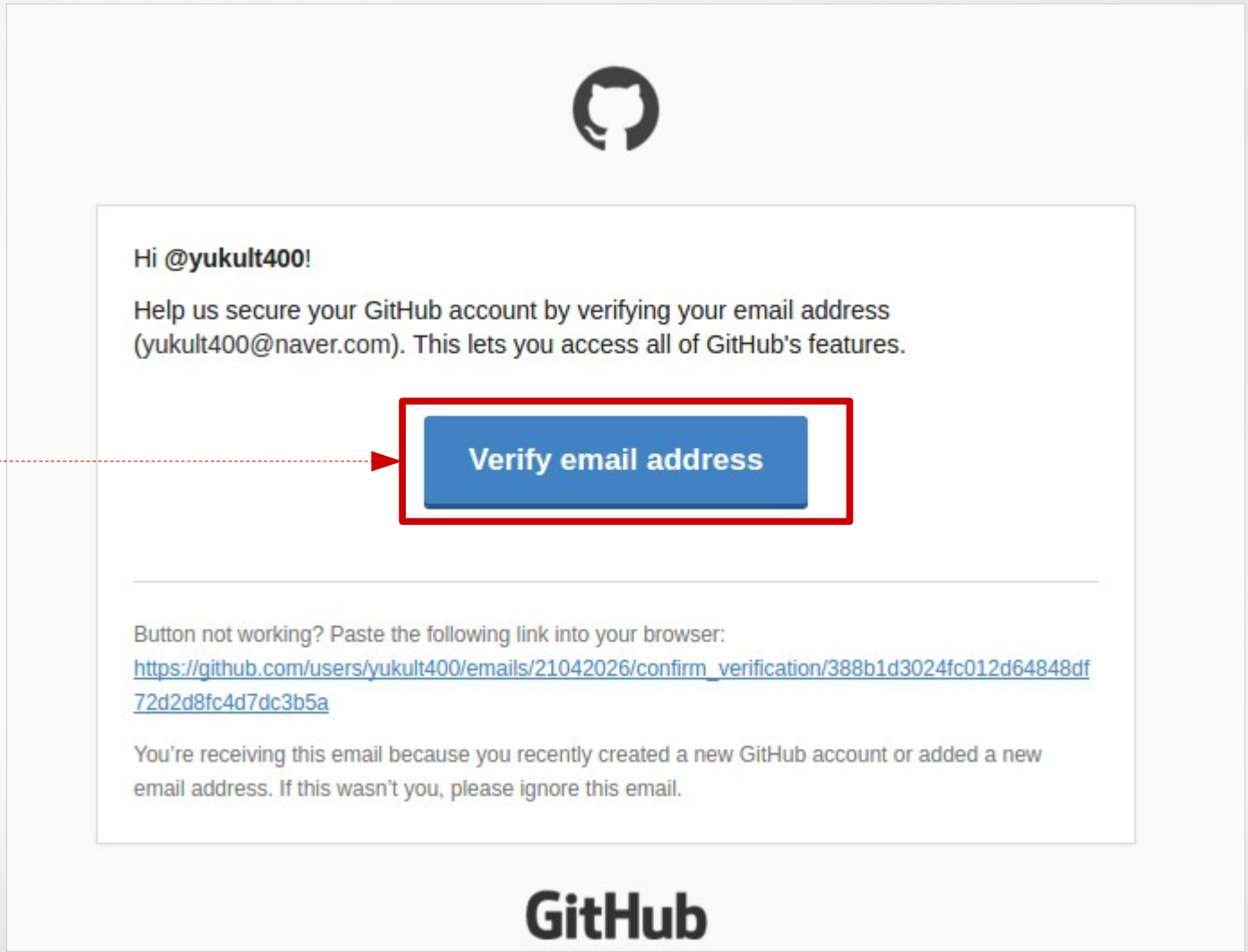
Git 실습 준비단계

가입정보 기입



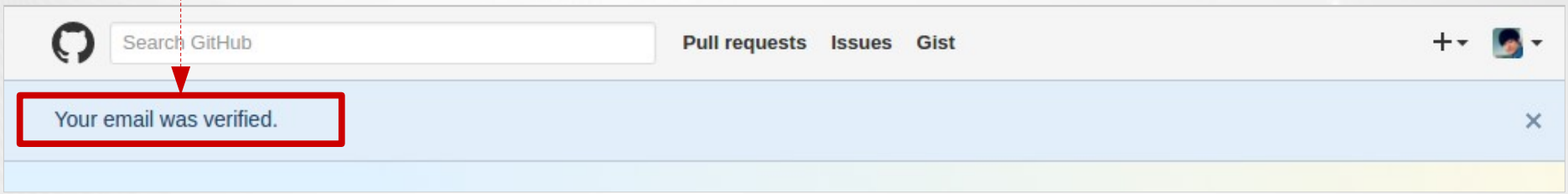
Git 실습 준비단계

본인 메일 확인해서 버튼 누르기



Git 실습 준비단계

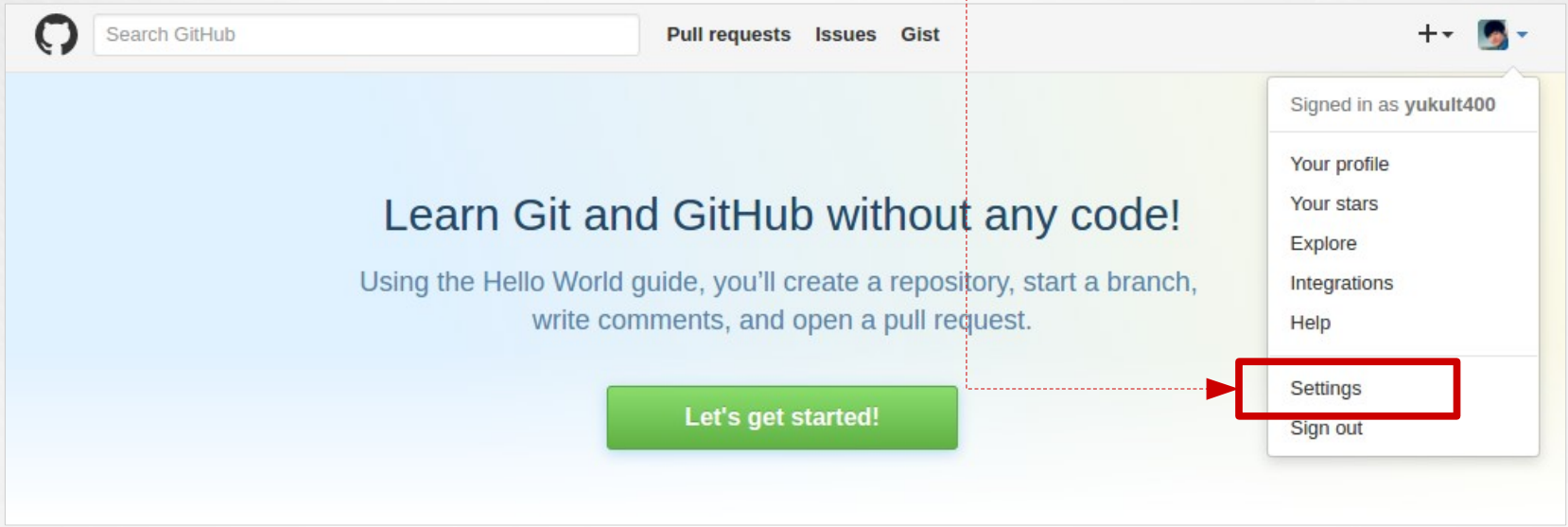
다시 Github 접속해서 메일인증 확인하기



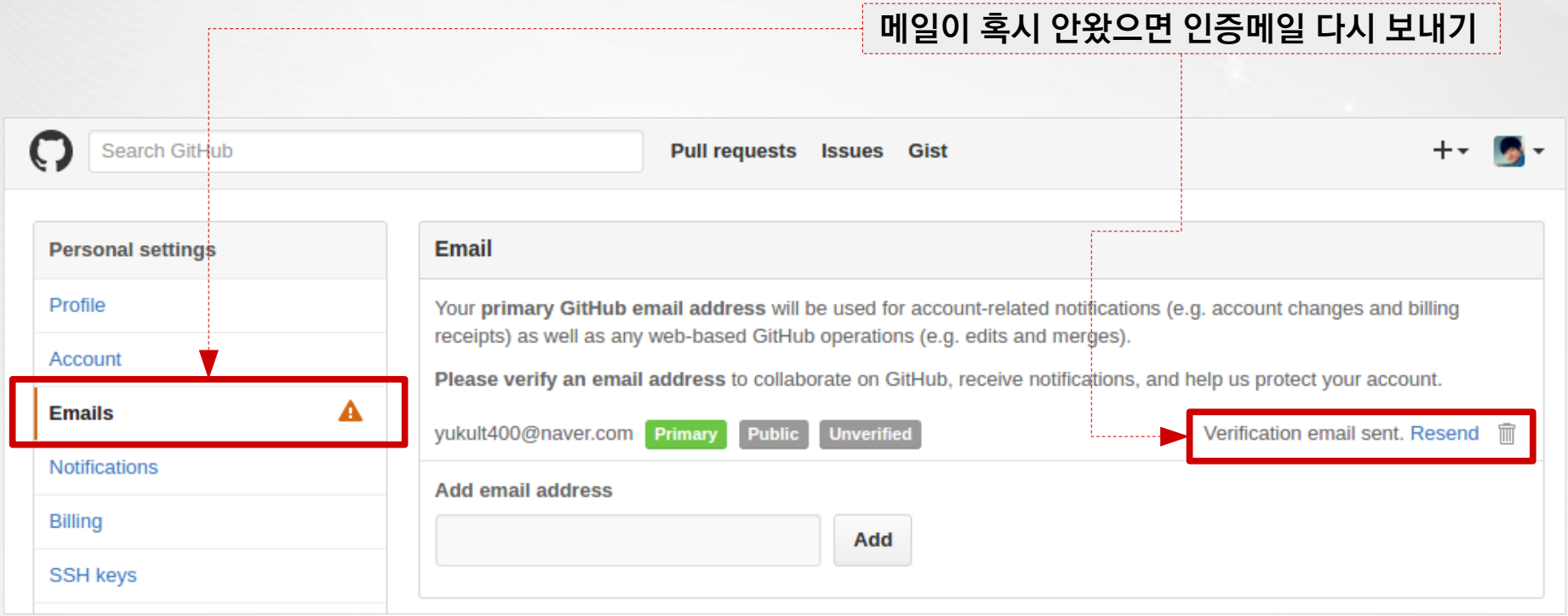
http://github.com

Git 실습 준비단계

기다려도 메일이 혹시 안왔으면설정버튼



Git 실습 준비단계



CHAPTER 01

Git 실습 준비단계

Git 실습 방법은 ?

직접 C 프로그래밍을 짜면서 Git 을 사용한다는 시나리오

report card (성적 출력하기 문제)

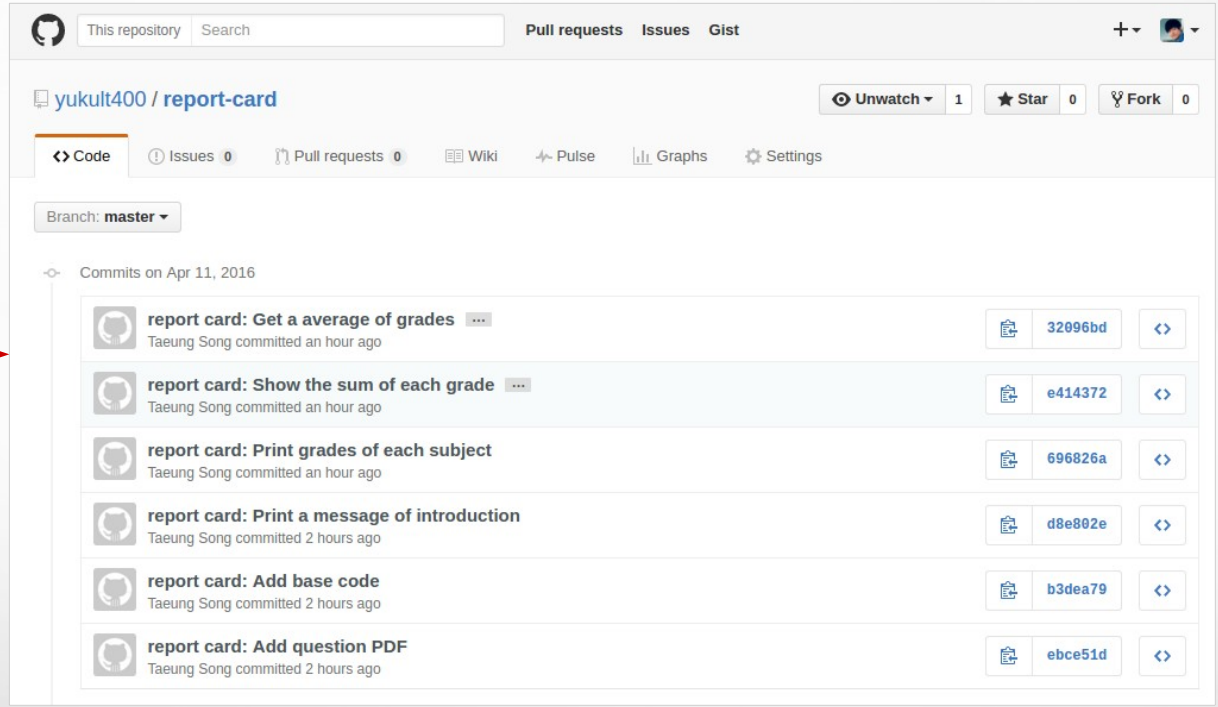
1) 코딩은 Ctrl-c, v 로 하고
2) Git 은 직접 명령어 쳐서 (git-bash)

Git 실습 준비단계

Stage 해결하기 위해서 수단과 방법을 가리지말자 !

본인의 Github 계정에 본인의 프로젝트에 commit 기록이 다음과 똑같아 지면 된다 .

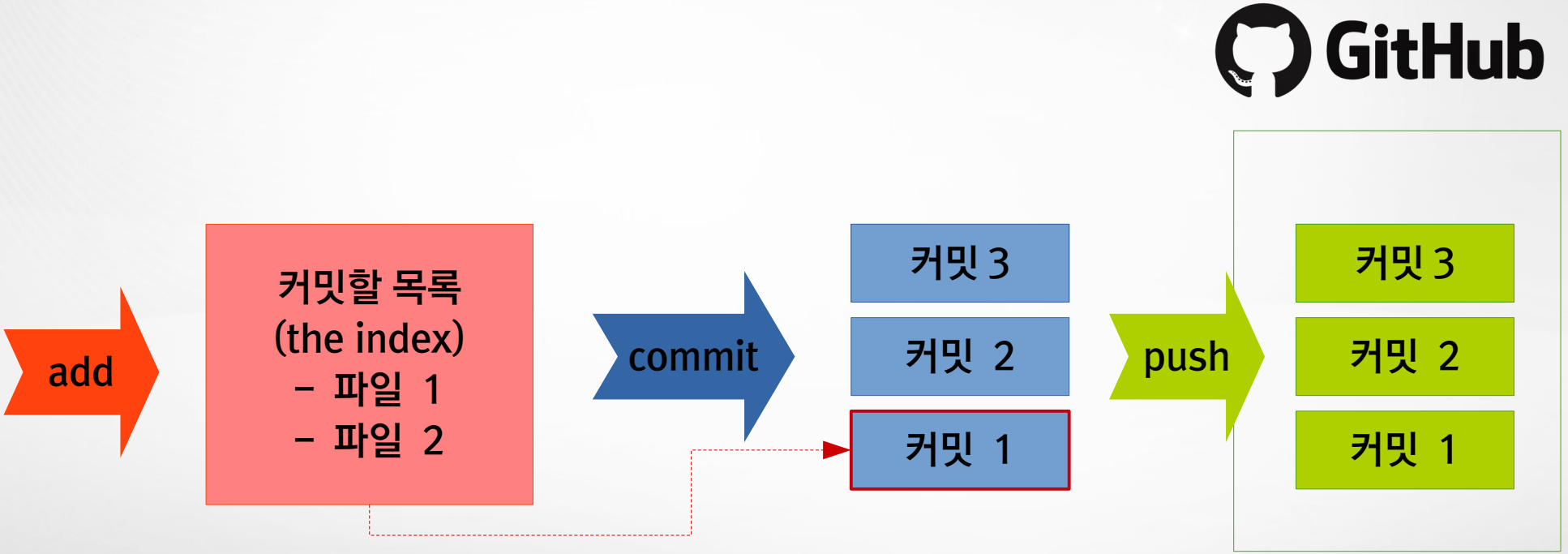
https://github.com/yukult400/report-card/commits/master



Git 실습에 필요한 기초개념

〈 Git 필수 명령 〉

add : 커밋할 목록에 추가
commit : 커밋 (히스토리의 한단위) 만들기
push: 현재까지 역사 (commits) Github 에 밀어넣기



Git 실습 준비

Prompt, 프롬프트

Arguments, 인자 (1 개)

```
# git add report_card.pdf
```

Git 의 Sub-command 중 하나

Arguments, 인자 (2 개)

```
# git config --global user.name “Taeung Song”
```

-- 로 시작하면 보통은 Long Name 옵션

```
# git commit -s
```

- 로 시작하면 보통은 Short Name 옵션

CHAPTER 01

Git 실습 목표

- 본 강의는 실습 90% 이론 10% (스파르타)
- 나중에 하는건 없다 무조건 오늘 목표는 이루고 가자
- 질문은 모두를 이롭게 한다 (taeung@kosslab.kr 메일로 대체)
- 내 손으로 명령어를 직접 입력해보자
- 터미널 환경 경험하자
- GUI(Graphic User Interface) 대신 CLI(Command Line Interface) 경험하자
- vi 에디터를 사용해보자

Chapter 2. Git 초기화 및 첫 commit

Stage 1. 초기화 및 첫 commit

1) Git bash 를 실행 (명령어 칠 준비), 폴더 생성하기

```
# mkdir report-card
```

2) 경로 이동

```
# cd report-card
```

3) 해당 폴더를 git 초기화

```
# git init
```

4) 프로그램 문제 PDF 파일 추가 (커밋할 목록에 추가 add) (commit1 폴더내 파일 활용)

```
# git add report_card.pdf
```

5) 첫 commit 하기 (역사 한단위 만들기)

```
# git commit -m "report card: Add question PDF"
```

Git 상태확인 명령어
(중간중간 치면서 수시로 확인하자)

```
# git show
# git log
# git shortlog
# git diff
# git status
```

Stage 1. 초기화 및 첫 commit

몰라도 좋으니, 일단 따라해보자.

6) 소스코드 추가하기 (커밋할 목록에 추가 add) (commit2 폴더내 파일 활용)

```
# git add report_card.c
```

7) commit 하기 (역사 한단위 만들기)

```
# git commit -m "report card: Add base code"
```

Git 상태확인 명령어
(중간중간 치면서 수시로 확인하자)

```
# git show
# git log
# git shortlog
# git diff
# git status
```

Stage 2. diff 사용과 추가 commit

몰라도 좋으니, 일단 따라해보자.

1) 상태를 확인한다

```
# git status
```

2) commit3 폴더에 있는 report_card.c 소스 파일로 수정, 덮어쓰기 후 확인

```
# git diff
```

3) diff 를 통해서 변화분을 확인했다면 add 진행

```
# git add report_card.c
```

4) 준비된 소스파일을 commit 한다

```
# git commit -m "report card: Print a message of introduction"
```

5) 지금까지한 3 개의 commit 들을 확인하자

```
# git log
```

CHAPTER 02

Stage 2. diff 사용과 추가 commit

몰라도 좋으니, 일단 따라해보자.

6) commit4 폴더에 있는 report_card.c 소스 파일로 수정, 덮어쓰기 후 확인

```
# git diff
```

7) diff 를 통해서 변화분을 확인했다면 add 진행

```
# git add report_card.c
```

8) 준비된 소스파일을 commit 한다

```
# git commit -m "report card: Print grades of each subject"
```

9) 지금까지한 4 개의 commit 들을 확인하자

```
# git log
```

Stage 3. commit message 에 본인서명 포함하기

1) 나의 Github 계정 이메일과 이름을 적자

```
# git config user.email “ 본인메일적으세요 @gmail.com”
```

```
# git config user.name “ 본인이름적으세요 Taeung Song”
```

2) commit5 폴더에 있는 report_card.c 소스 파일로 수정 , 덮어쓰기 후 확인

```
# git diff
```

3) diff 를 통해서 변화분을 확인했다면 add 진행

```
# git add report_card.c
```

4) 서명과함께 commit 한다 (-s 옵션으로 서명을 포함한다 .)

```
# git commit -sm "report card: Show the sum of each grade"
```

Stage 3. commit message 에 본인서명 포함하기

5) commit6 폴더에 있는 report_card.c 소스 파일로 수정, 덮어쓰기 후 확인

```
# git diff
```

6) diff 를 통해서 변화분을 확인했다면 add 진행

```
# git add report_card.c
```

7) 서명과함께 commit 한다

```
# git commit -sm "report card: Get a average of grades"
```

Contents

Chapter 3. Github 에 Push 하기
Commit 수정 , 삭제 및 Add 취소

Stage 4. Github 에 Push 하기

몰라도 좋으니, 일단 따라해보자.

1) 상태를 확인하고 현재 브랜치명 master 를 확인하자

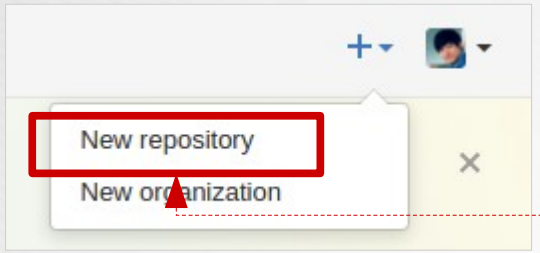
```
# git status
```

2) 지금까지한 commit 들을 확인하자 (6 개가 아니면 다시 확인하자)

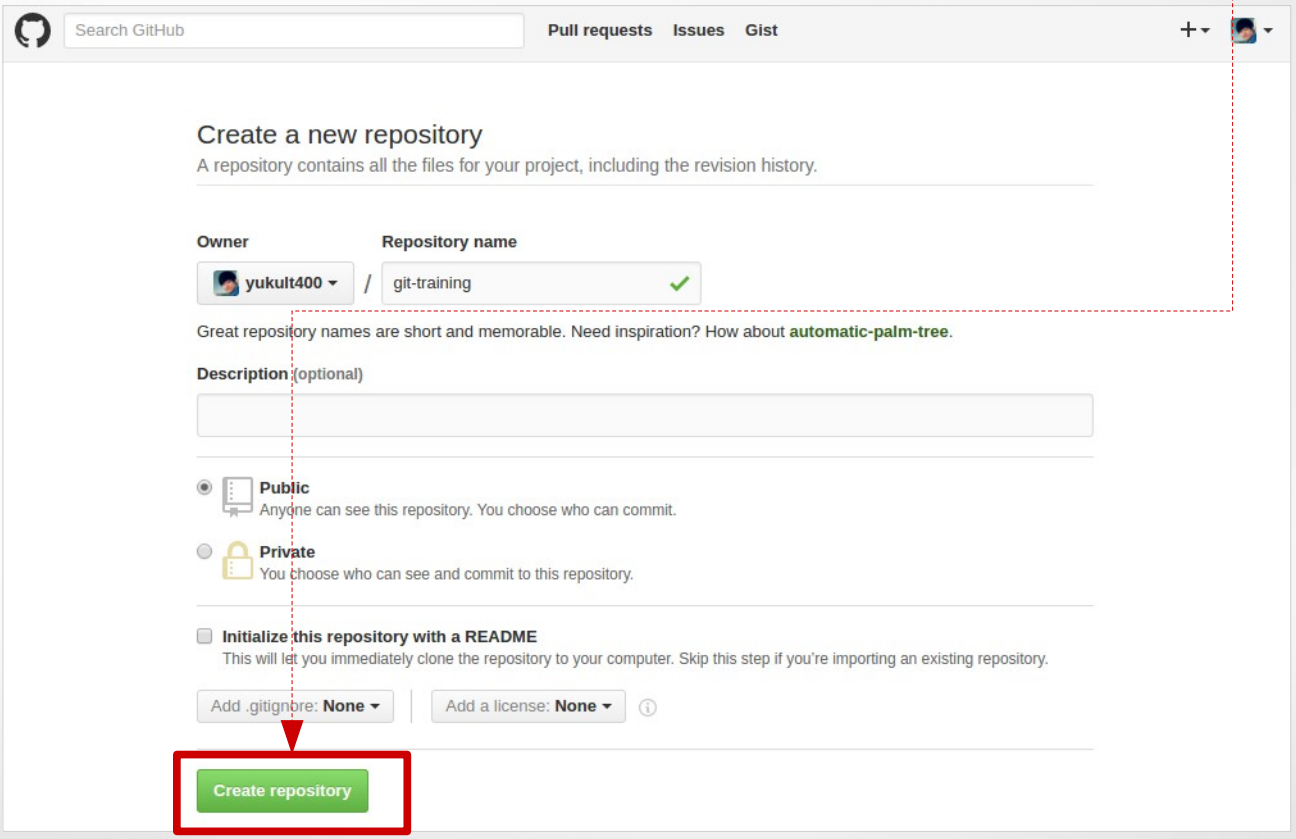
```
# git shortlog
```

3) Github 원격저장소 URL 를 등록하자
(잠깐 멈추고 http://github.com 를 켜고 repository 새로 생성하자)

Stage 4. Github 에 원격저장소 만들기



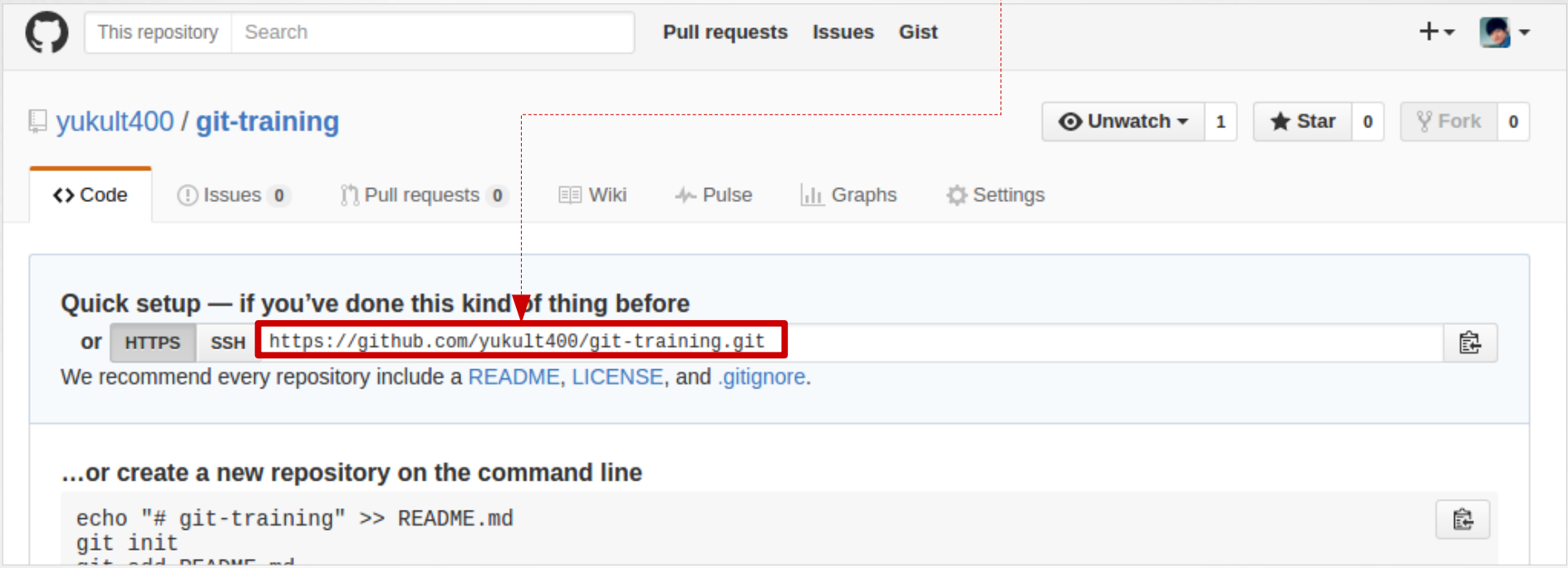
새로운 원격 저장소를 생성하자 (클릭 , 프로젝트명 자유)



CHAPTER 03

Stage 4. Github 에 원격저장소 만들기

해당 URL 복사해서 Stage5 의 4) 으로 이어서 진행



Stage 4. Github 에 Push 하기

4) 아까복사한 URL 로 Github 원격저장소 등록하자

```
# git remote add origin <아까복사한 URL>
```

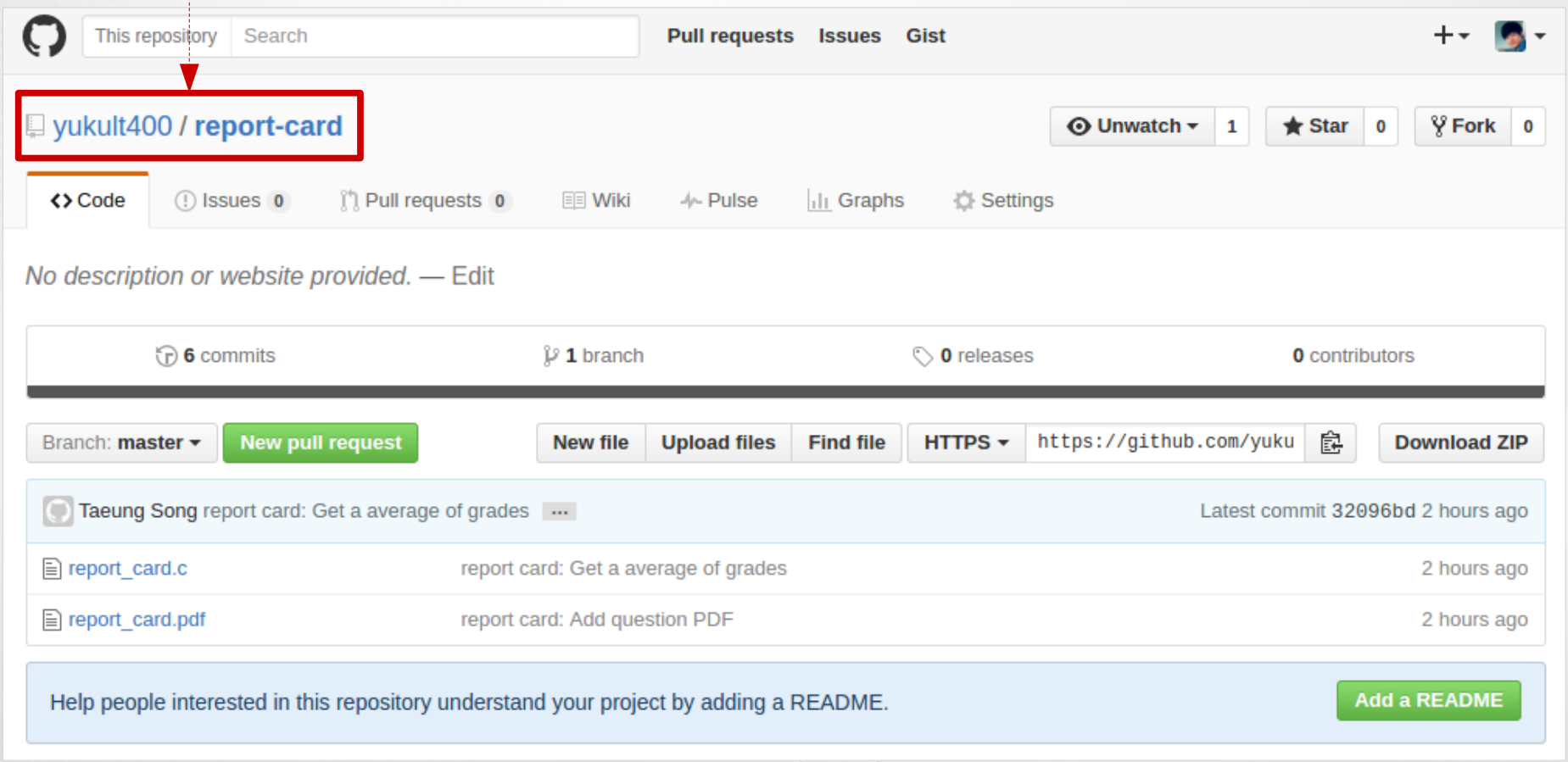
5) Github 원격저장소 (origin) 에 밀어 넣자

```
# git push origin master
```

6) Github 웹페이지 열고 확인하자

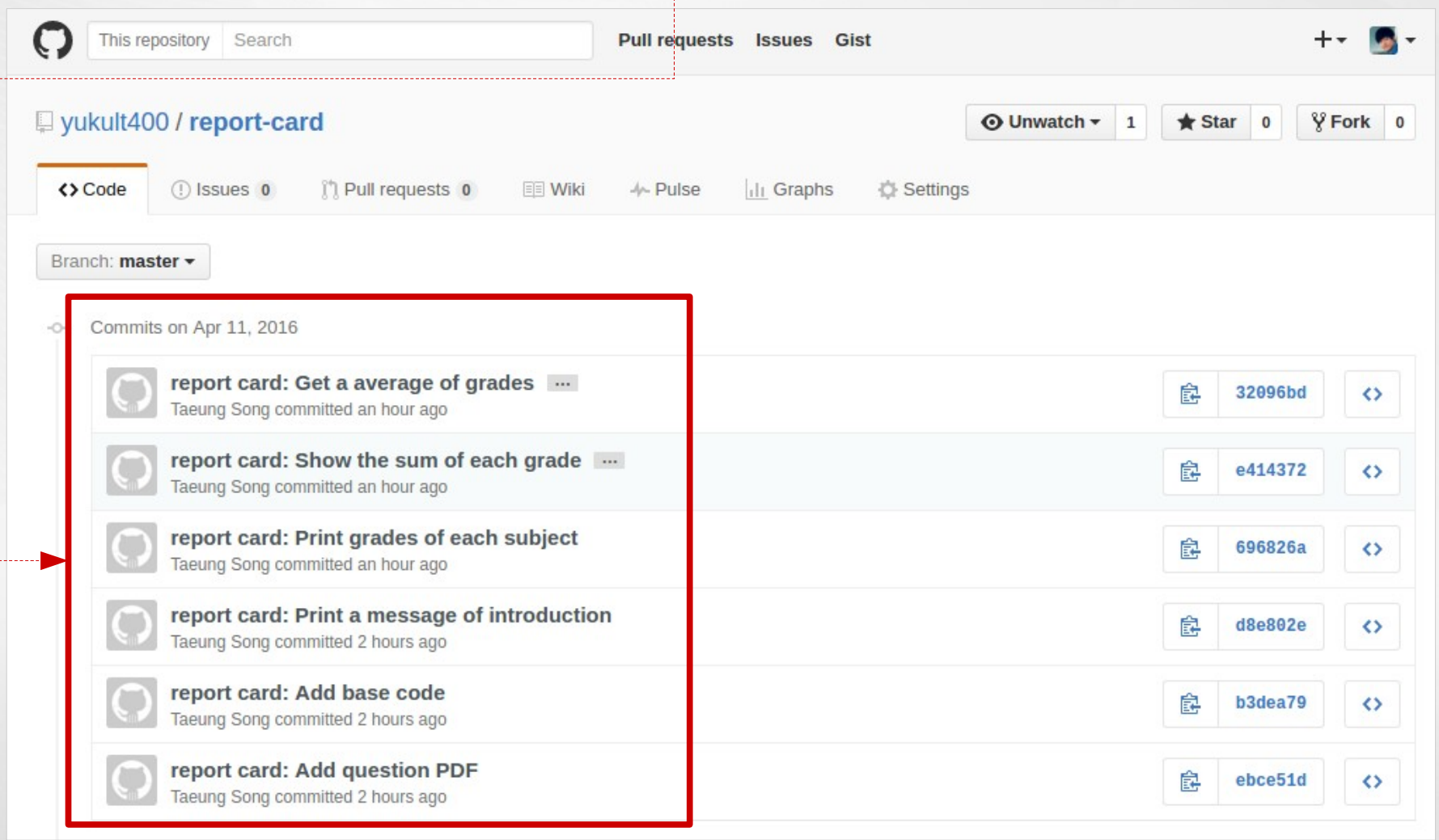
Stage 4. Github 에 Push 결과확인

Github 들어가서 본인 프로젝트의 commit 기록 눌러보자



Stage 4. Github 에 Push 결과확인

본인이 추가한 commit 들이 나오는걸 확인하자 (본인 Github)



Stage 5. Commit 수정

1) report_card.c 소스내의 'Mean' 변수명을 'Average'로 바꾼다고 가정하자
(commit6-1 폴더내 소스 활용)

```
# git diff
```

2) diff를 통해서 변화분을 확인했다면 add 진행

```
# git add report_card.c
```

3) 가장 위에 있는 commit을 수정하자

```
# git commit --amend
```

Stage 5. Remote 의 Commit 수정

4) 바로 push 해보자 (충돌 **오류발생**)

```
# git push origin master
```

5) 강제로 push 해서 수정하자

```
# git push origin master -f
```

6) 다시 Github 가서 제대로 변경되었는지 확인해보자

Stage 6. Commit 삭제

1) 아까 test 파일 여전히 존재하는지 확인 (지웠으면 다시 만들기)

```
# git status
```

2) 임의로 실수의 commit 을 만들어 낸다 (; 로 명령어들을 연속적 실행가능)

```
# git add test; git commit -sm “test”
```

3) 그리고 push 까지 해서 Github 에 있는 tree 까지 실수 commit 을 넣어버린다

```
# git push origin master
```

4) 그리고 가장 최근 commit 을 지운다

```
# git reset HEAD~1
```

5) 강제로 Github 에 있는 tree 도 밀어넣어서 수정한다

```
# git push origin master -f
```

Stage 7. Add 취소

1) touch 로 빈파일 생성하고 add 하자 (; 로 명령어들을 연속적 실행 가능)

```
# touch test; git add test
```

2) 현재상태 확인해보고

```
# git status
```

3) reset 으로 add 한거 취소해보자

```
# git reset
```

4) 현재 상태 다시 한번 확인해본다

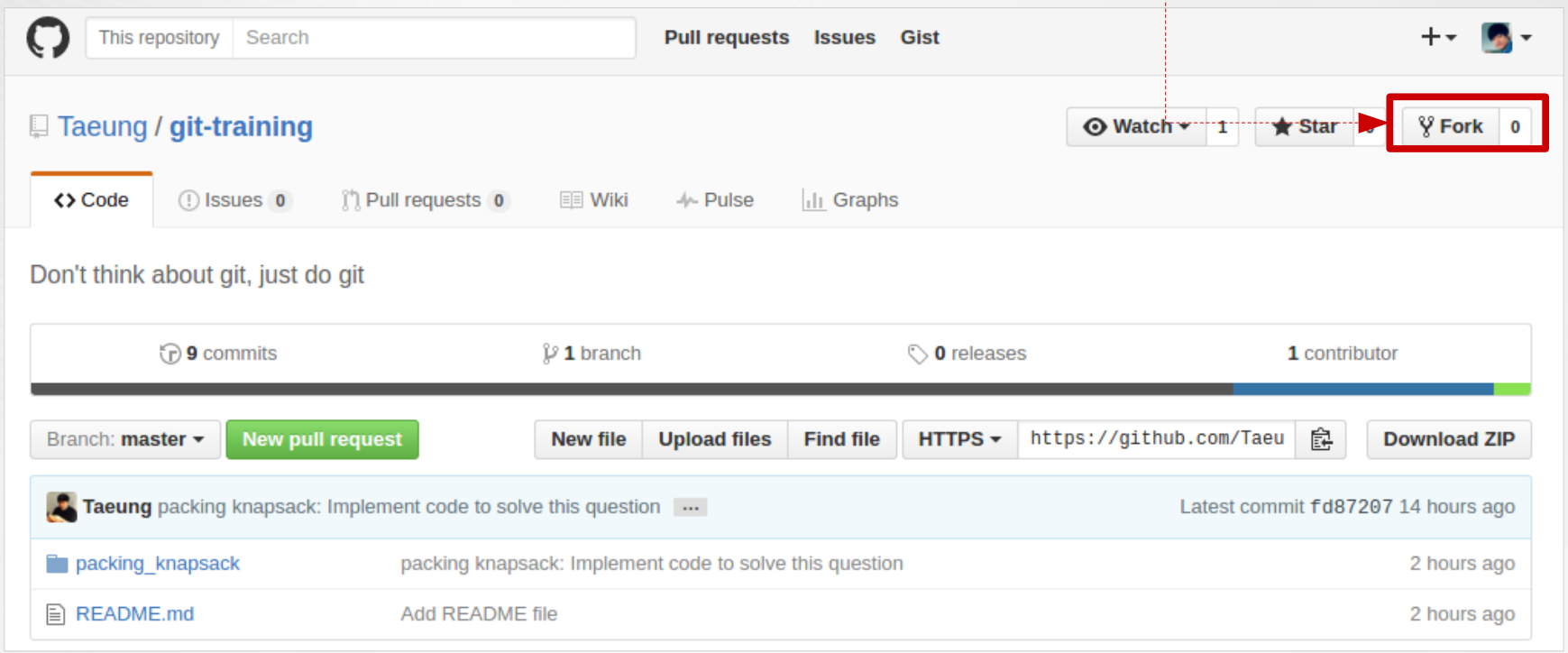
```
# git status
```

Chapter 4. Pull-request 하기

CHAPTER 04

Github 에서 Fork 하기

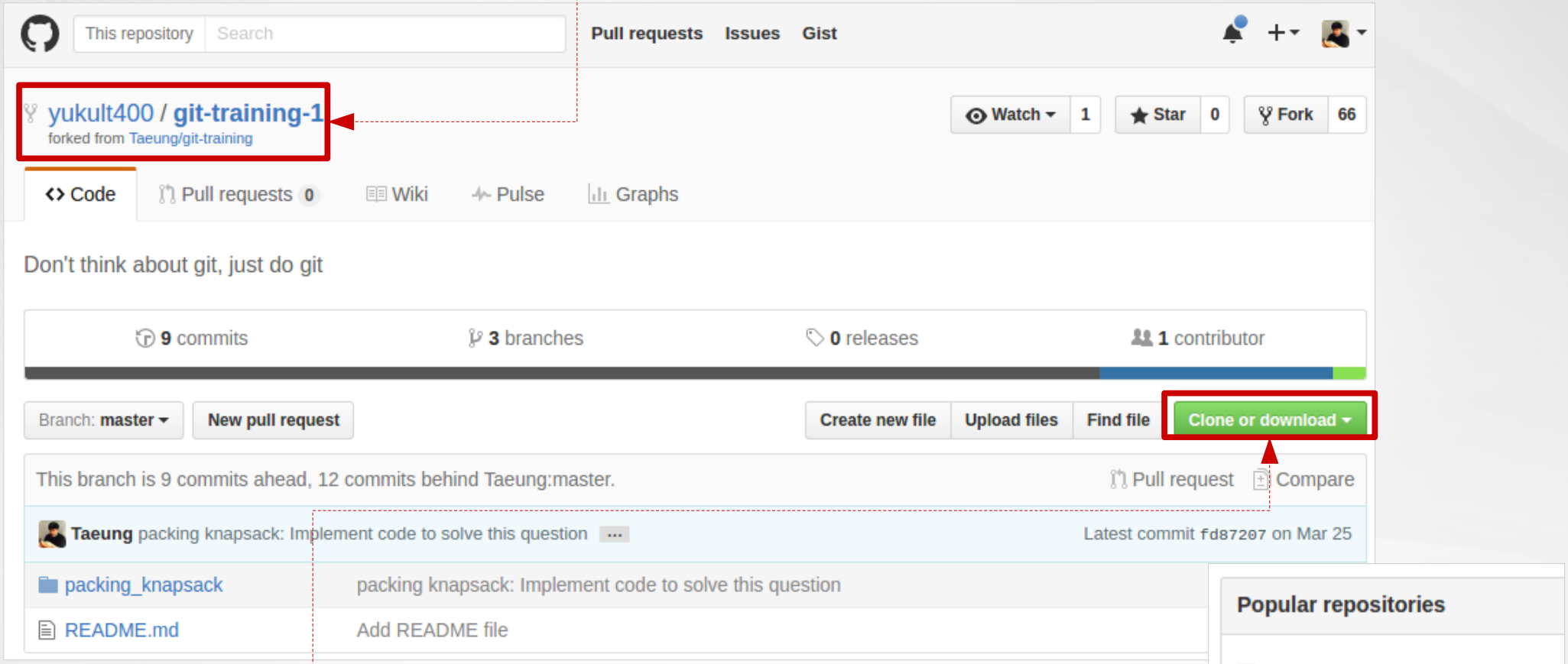
http://github.com/taeung/git-training 가서 fork 버튼을 누르자



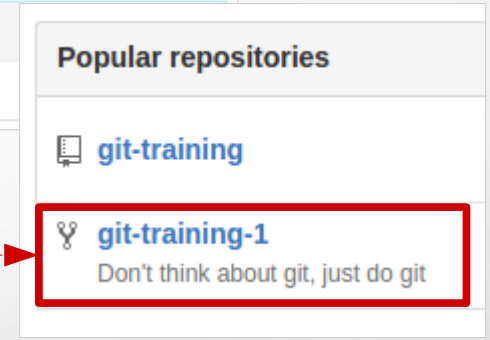
CHAPTER 04

Github 에서 Fork 하기

fork 가 되면 내 원격저장소가 추가 된다



fork 해서 만들어진 repo 의 URL 복사



나의 profile 페이지에서도 확인 가능 (Github)

Stage 8. Pull-request 하기

1) clone 으로 fork 한 repo 받아오기 (주의 : .git 이 생성된 폴더내에서 clone 하지말것)

```
# git clone 〈아까 fork 한 repo 에서 복사한 URL〉
```

2) pull-request 작업할 브랜치 (develop) 따로 만들기

```
# git checkout -b develop
```

3) pull_request_test 폴더로 이동해서

```
# cd pull_request_test
```

4) 내 이름으로된 폴더 만들고

```
# mkdir taeung
```

5) 내 이름으로 된 폴더에 작업한 소스 및 임의의 파일을 넣고나서

Stage 8. Pull-request 하기

6) 추가한 폴더 (내가작업한 소스 내용) 통째로 add

```
# git add <나의 소스작업폴더>
```

7) 준비된 파일들 commit

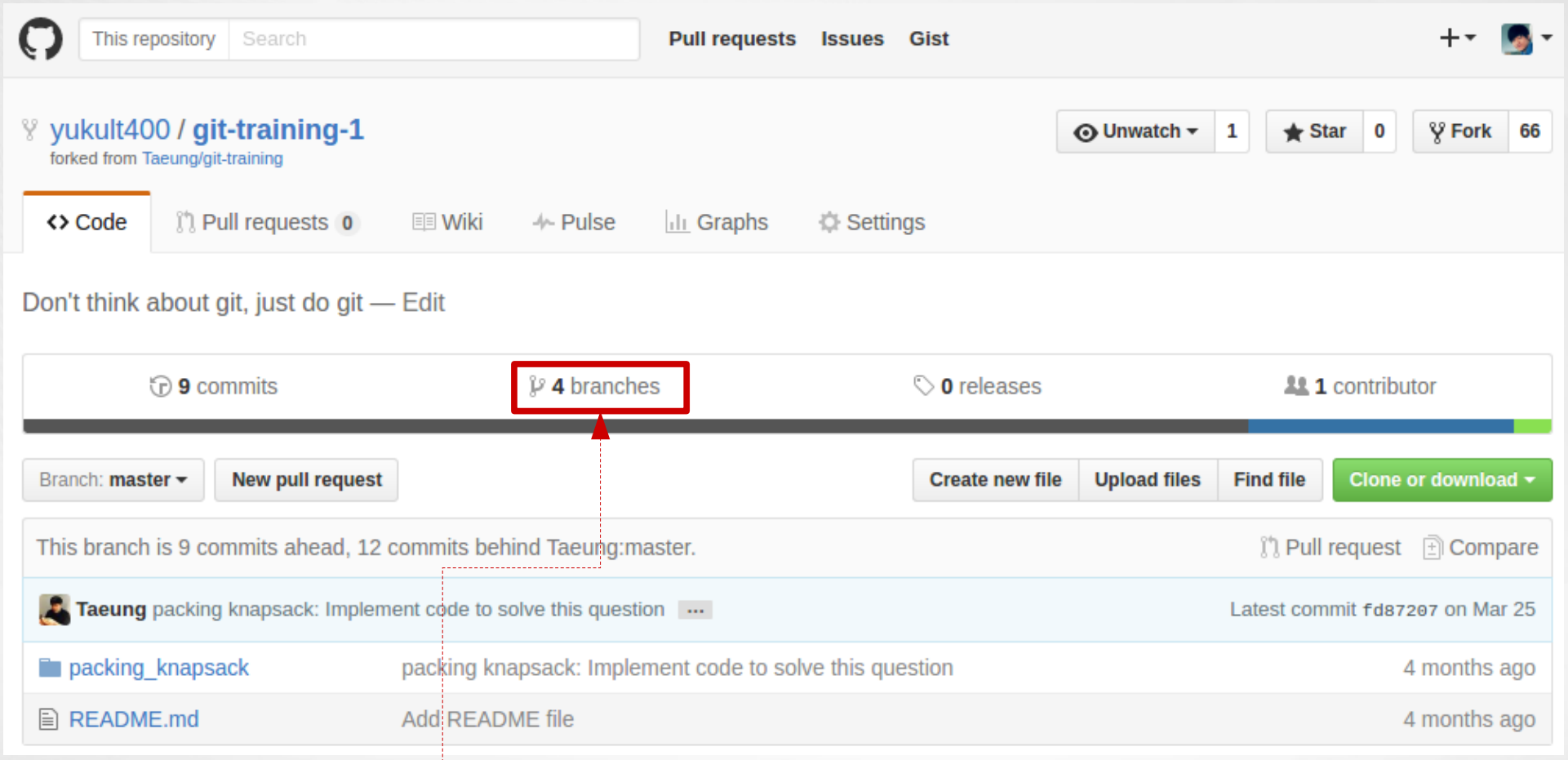
```
# git commit -sm “test pull request”
```

8) 내가 fork 한 repo 의 develop 브랜치로 push (주의 : master 아님)

```
#  git push origin develop
```

CHAPTER 04

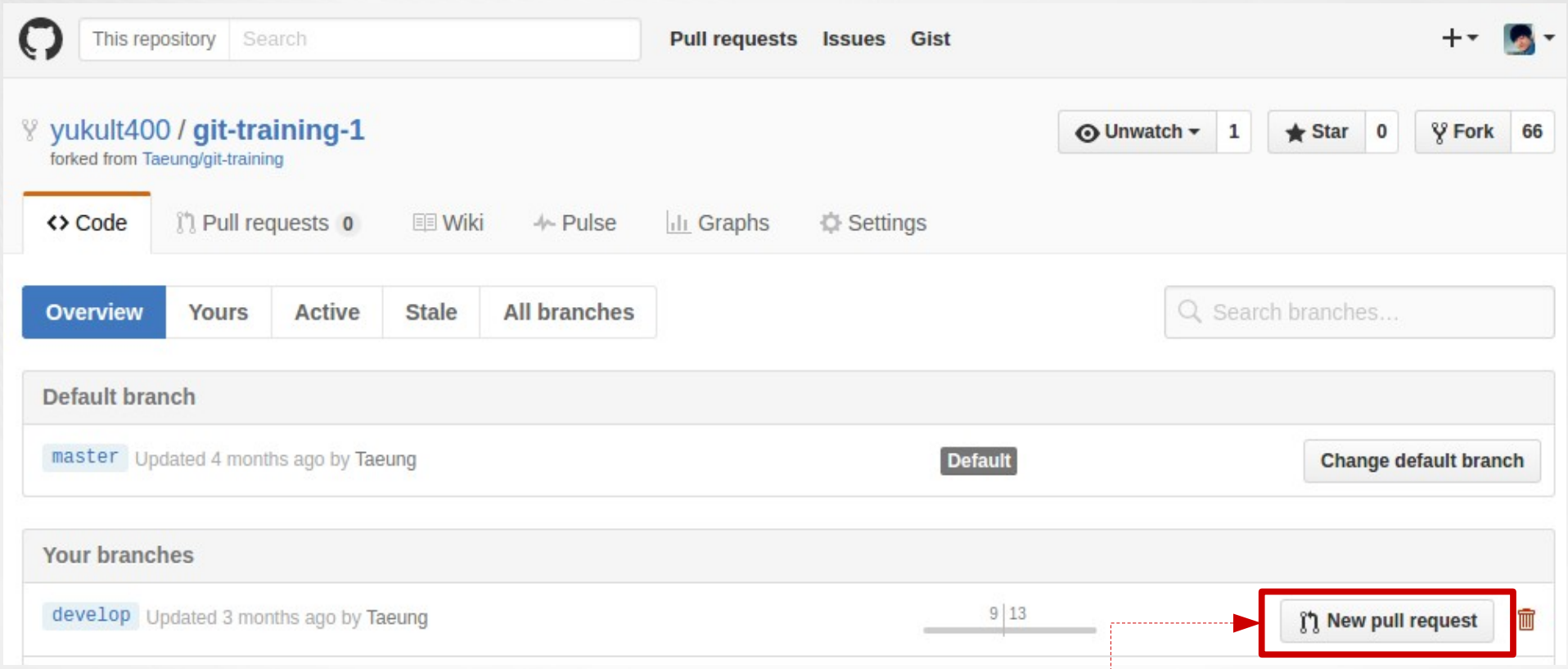
Stage 8. Pull-request 하기



방금 Push 했던 브랜치를 확인하기 위해서 Branch 탭 클릭

CHAPTER 04

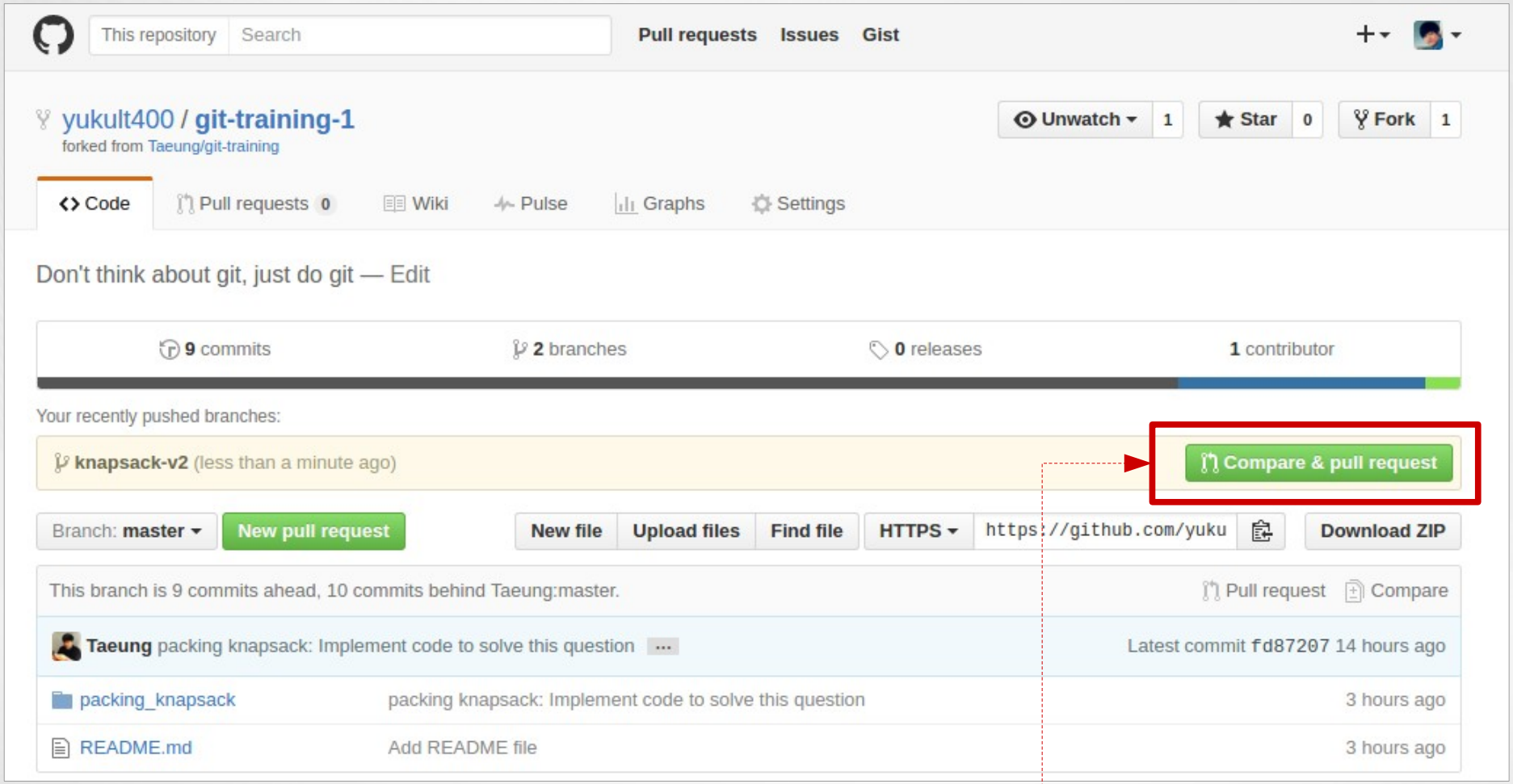
Stage 8. Pull-request 하기



Pull-request 하려는 브랜치에서 New pull-request 버튼을 클릭

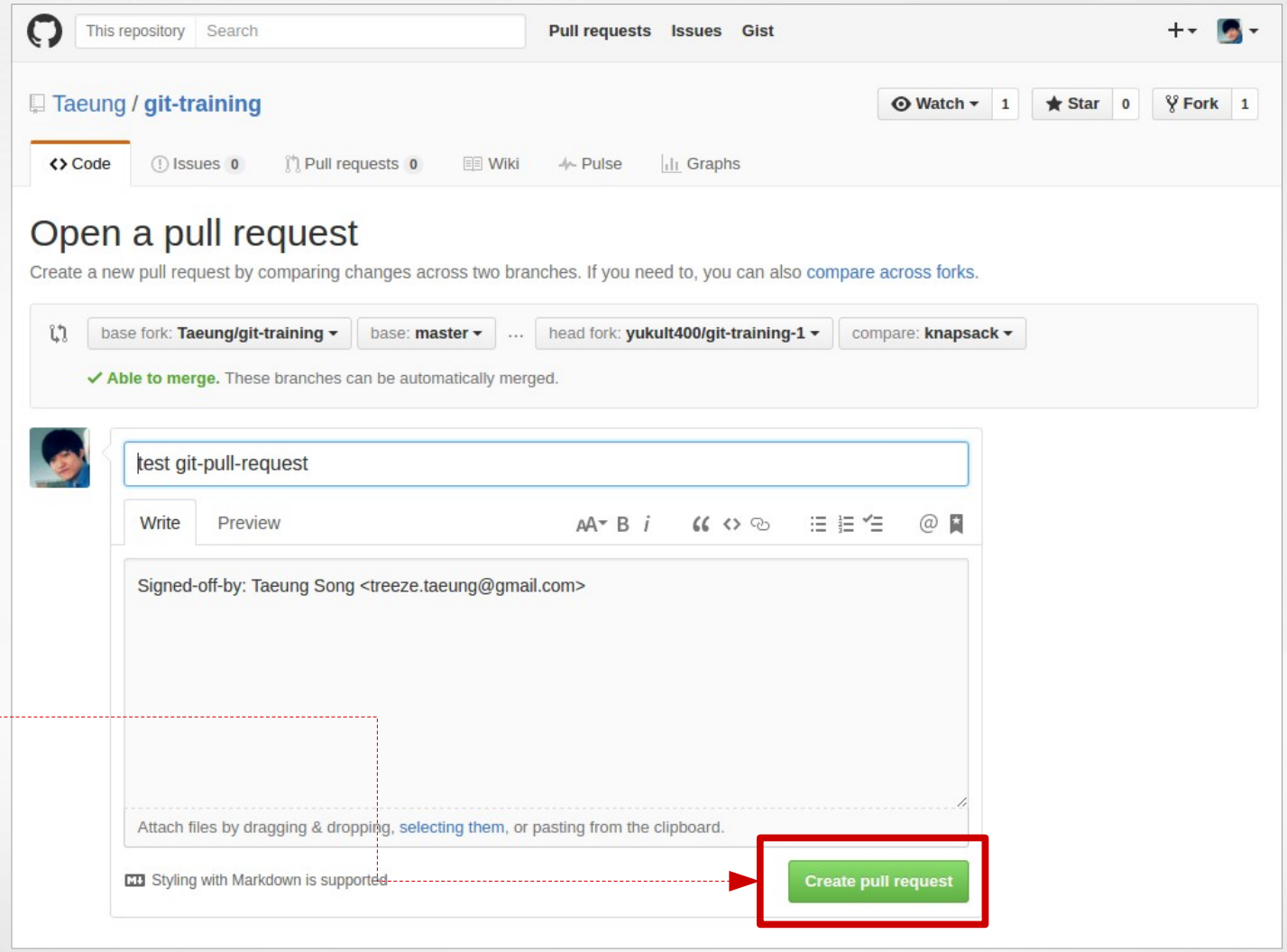
CHAPTER 04

Stage 8. Pull-request 하기



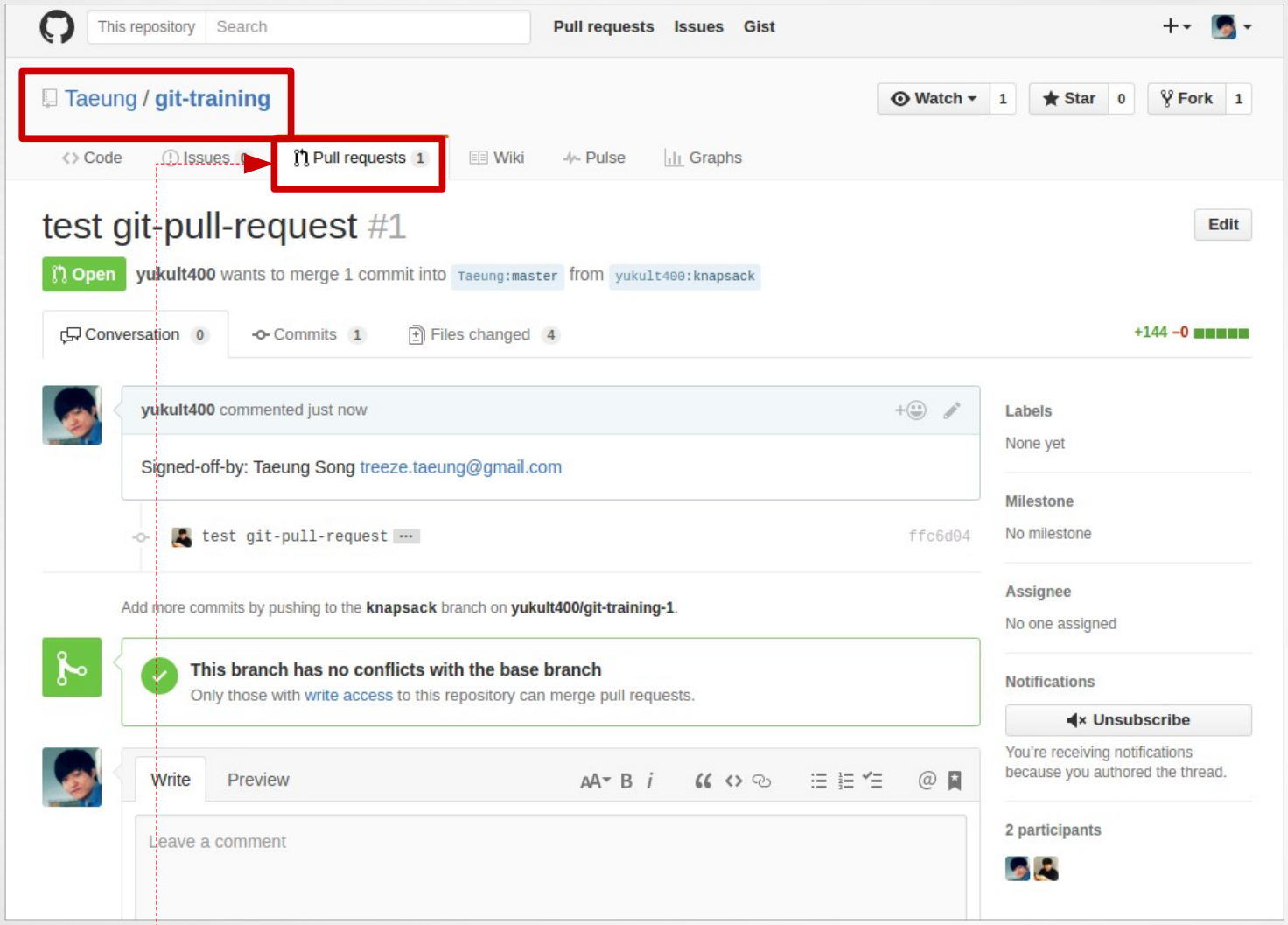
다음과 같이 방금 Push 한 내용이 Pull-request 할 수 있도록 버튼이 만들어지기도 한다 .

Stage 8. Pull-request 하기



Pull-request 보내기

Stage 8. Pull-request 확인



만들어진 pull-request 확인하기

Git / GitHub Training

How to use git

- Git 간단한 정의 / 기능
- Git 기본 실습 (How)
- Git 이해하기 (Why, What)
- Git 고급 (Advanced)
- Opensource (with Git)

Chapter 5. Git 을 사용하는 진짜 이유

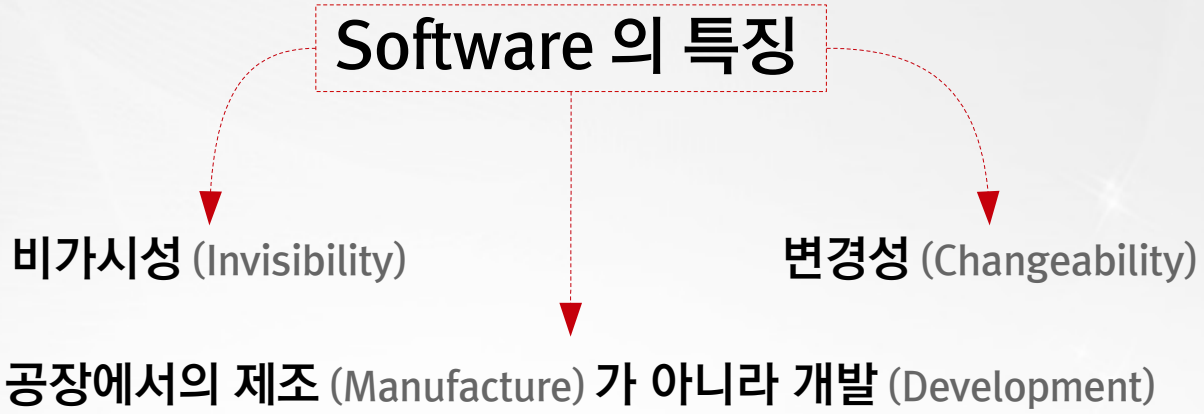
Git 을 쓰는 이유

협업 때문에 Git 을 쓴다 (집단지성의 극대화)

현대적인 , 선진화된 **소스코드 개발** 과정의 **필수도구**로 Git 을 쓴다 (Needs)

https://geektimes.ru

Software 의 특징과 문제점



Commit 단위개발 Idea

Commit 단위로 코딩하고 리뷰하고 토론하고 적용한다 . (집단지성의 극대화)

소프트웨어의 취약점을 극복하는 전략

Commit 단위개발의 전제

혼자가면 빨리가지만 (짧은다리로 빨리 달리기)

함께가면 멀리간다 (긴다리로 천천히 가기)

Git 을 통한 현명한 협업

Software 의 특징

- 비가시성 (Invisibility)
- 변경성 (Changeability)

공장에서의 제조 (Manufacture) 가 아니라 개발 (Development)

Git 과 Commit 단위 개발을 하면서 극복

Coding Style 또는 Coding Convention 은 기본

Commits(변화분) 단위 관리를 통해 (코드를 수정한 이유가 명확)

이 변화분을 통한 Review 와 Discussion 그리고 적용 (다수의 버그 / 오류 미리 차단)

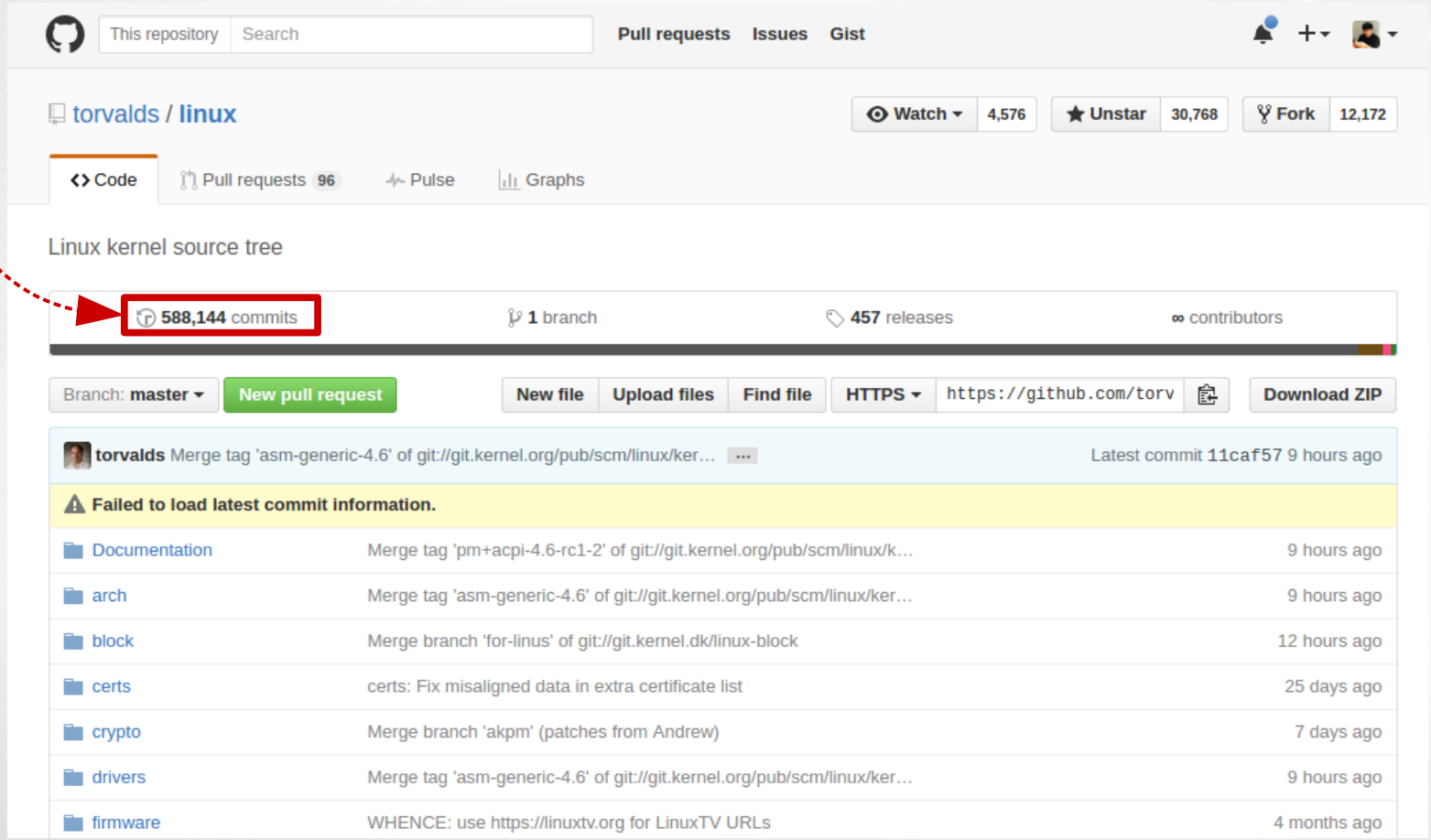
개발과정 가시화 + 협업용이 + 유지보수 + 추적 용이 + 소스코드 품질 상승

CHAPTER 05

Commit 단위 관리 예시

* 참고 : Linux kernel 은 Github 에서 mirror 까지는 되지만 다음 Repository 가 공식
http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/

Commits(변화분) 단위 관리를 통해 (코드를 수정한 이유가 명확)



https://github.com/torvalds/linux

Commit 단위 관리 예시

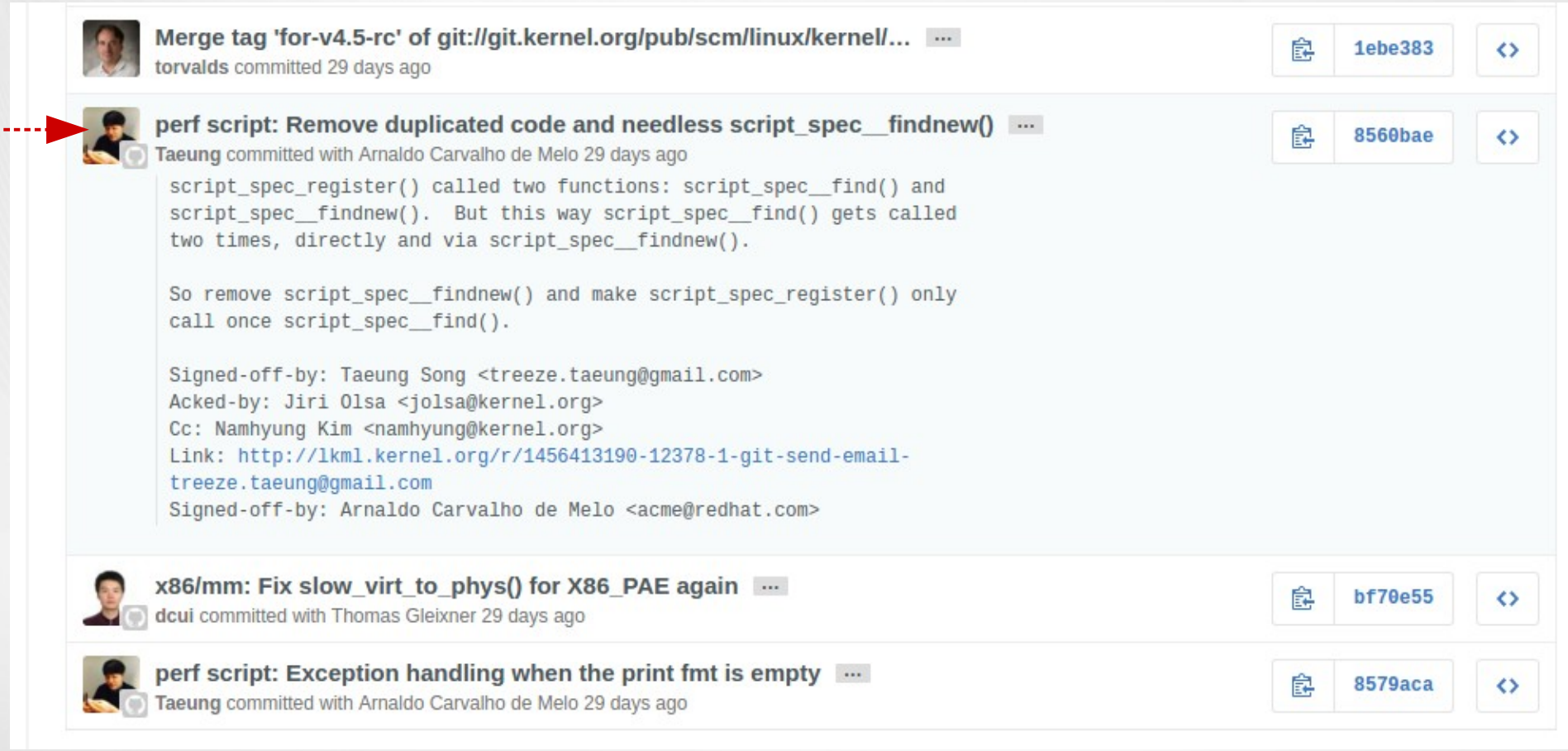
Commits(변화분) 단위 관리를 통해 (코드를 수정한 이유가 명확)

https://github.com/torvalds/linux

CHAPTER 05

Commit 단위 관리 예시

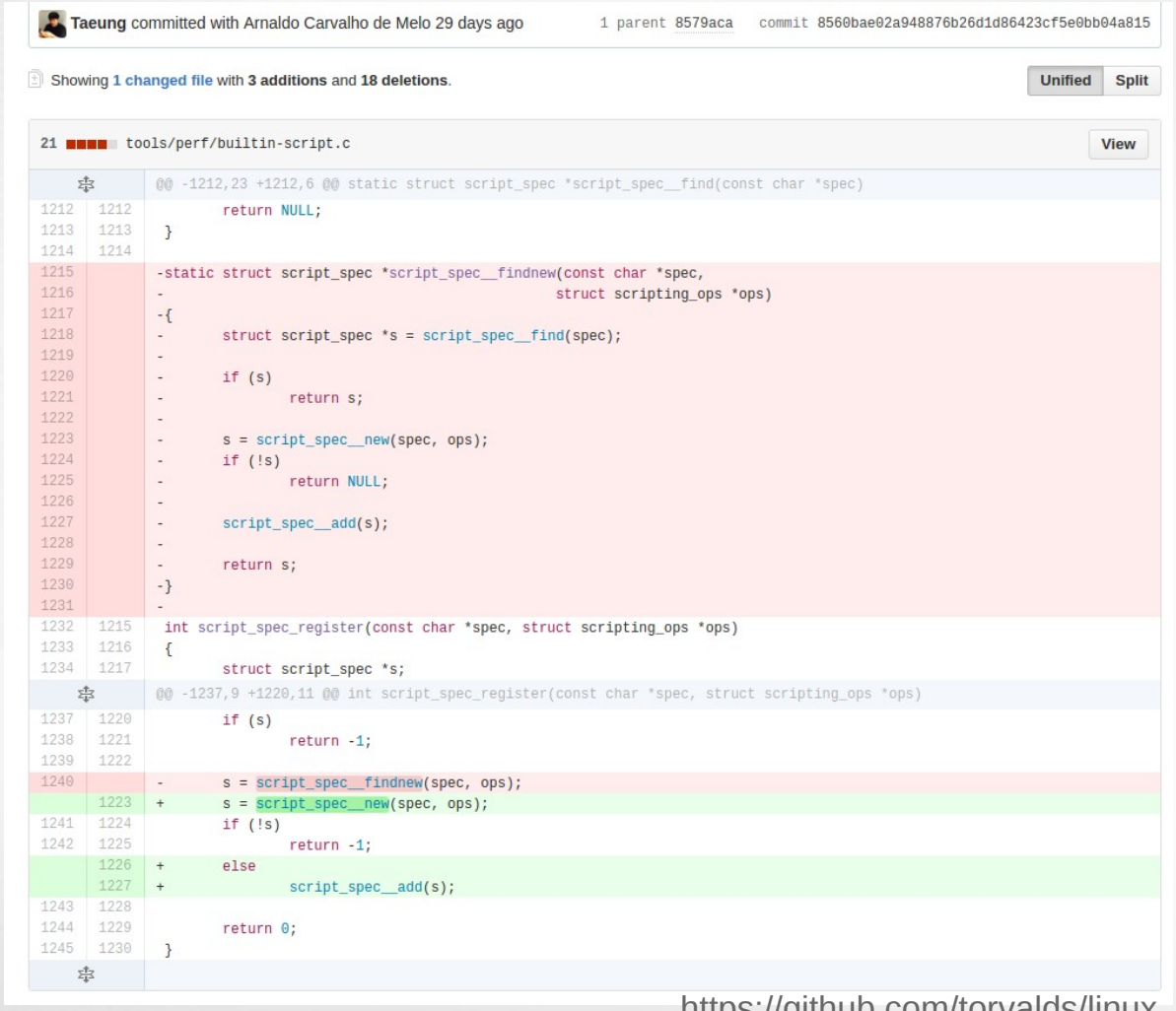
Commits(변화분) 단위 관리를 통해 (코드를 수정한 이유가 명확)



https://github.com/torvalds/linux

Commit 단위 관리 예시

Commits(변화분) 단위 관리를 통해 (코드를 수정한 이유가 명확)



Taeung committed with Arnaldo Carvalho de Melo 29 days ago 1 parent 8579aca commit 8560bae02a948876b26d1d86423cf5e0bb04a815

Showing **1 changed file** with **3 additions** and **18 deletions**. Unified | Split

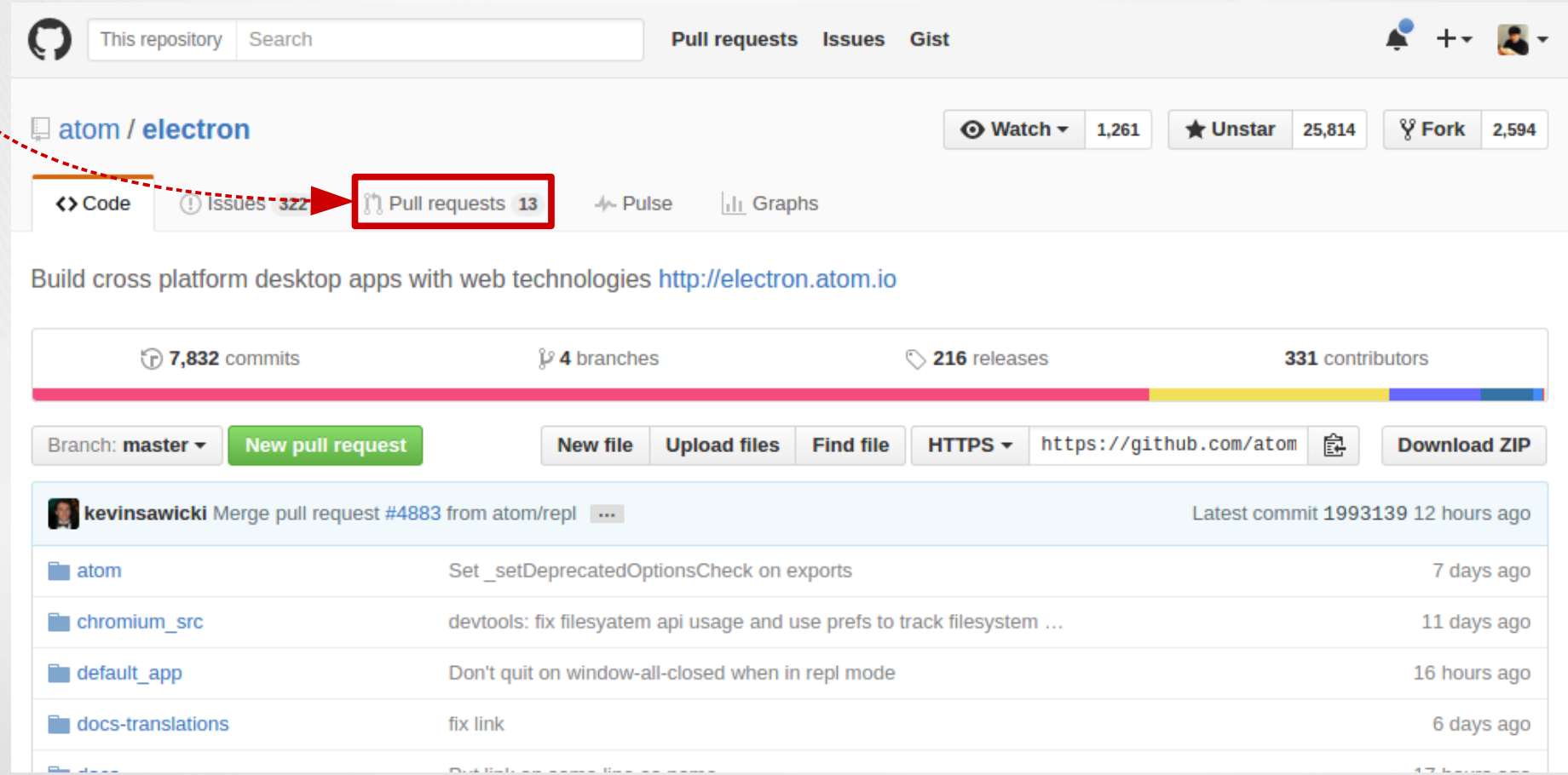
21 tools/perf/builtin-script.c View

```
@@ -1212,23 +1212,6 @@ static struct script_spec *script_spec__find(const char *spec)
        return NULL;
 }

-static struct script_spec *script_spec__findnew(const char *spec,
-                                                struct scripting_ops *ops)
-{
-       struct script_spec *s = script_spec__find(spec);
-
-       if (s)
-               return s;
-
-       s = script_spec__new(spec, ops);
-       if (!s)
-               return NULL;
-
-       script_spec__add(s);
-
-       return s;
-}
-
 int script_spec_register(const char *spec, struct scripting_ops *ops)
 {
        struct script_spec *s;
@@ -1237,9 +1220,11 @@ int script_spec_register(const char *spec, struct scripting_ops *ops)
        if (s)
                return -1;

-       s = script_spec__findnew(spec, ops);
+       s = script_spec__new(spec, ops);
        if (!s)
                return -1;
+       else
+               script_spec__add(s);

        return 0;
 }
```

https://github.com/torvalds/linux

Commit 단위 Review, Discussion 예시

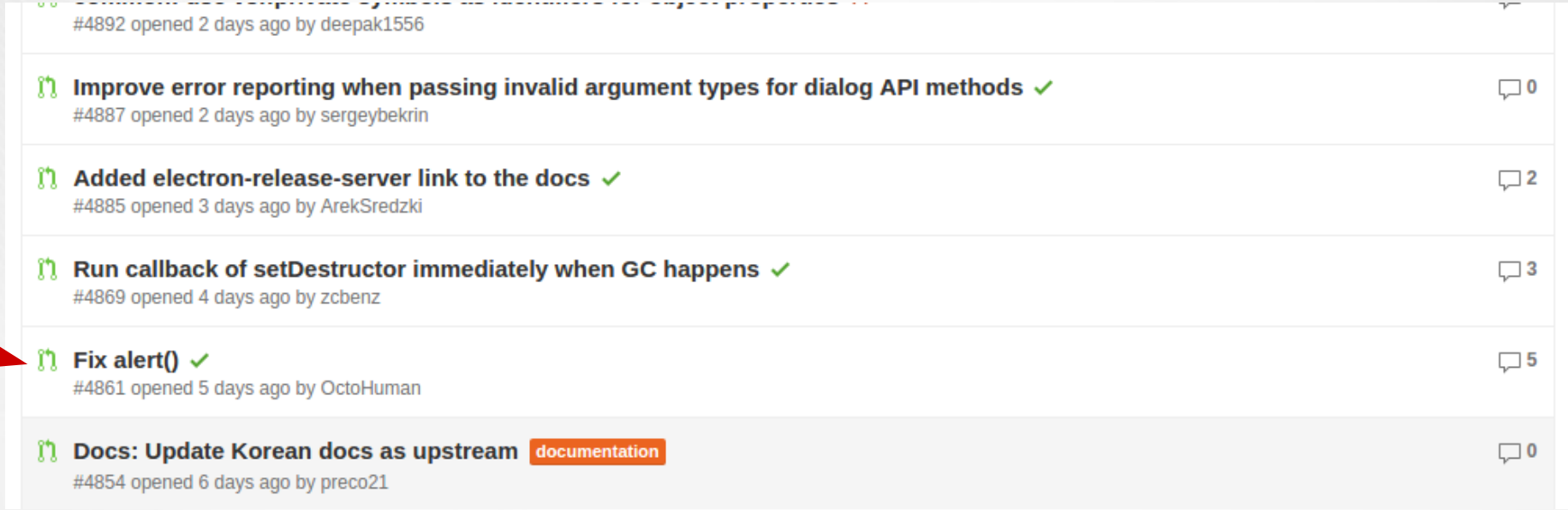
이 변화분 (commits) 을 통한 Review 와 Discussion 그리고 적용 (다수의 버그 / 오류 미리 차단)



CHAPTER 05

Commit 단위 Review, Discussion 예시

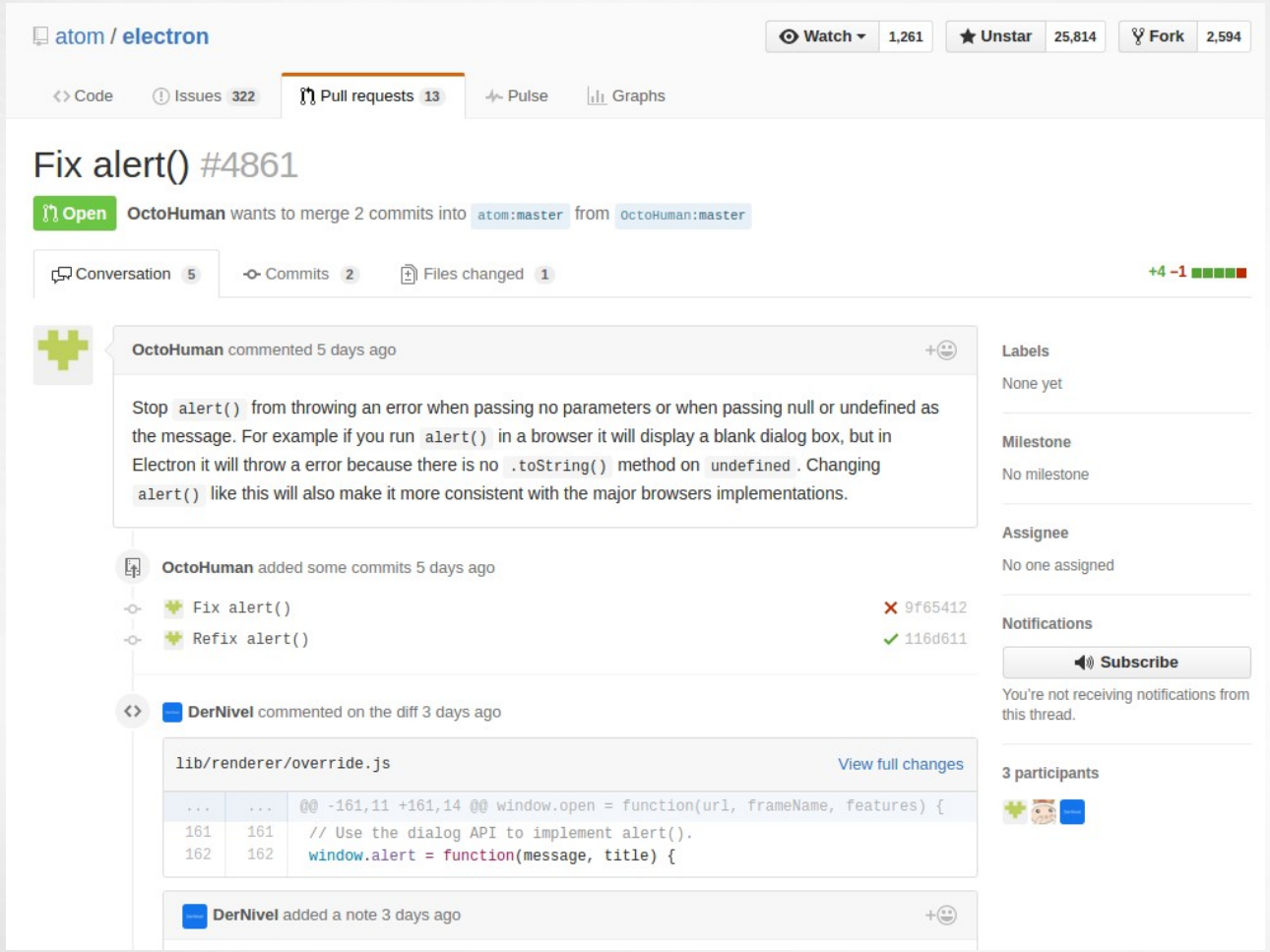
이 변화분 (commits) 을 통한 Review 와 Discussion 그리고 적용 (다수의 버그 / 오류 미리 차단)



CHAPTER 05

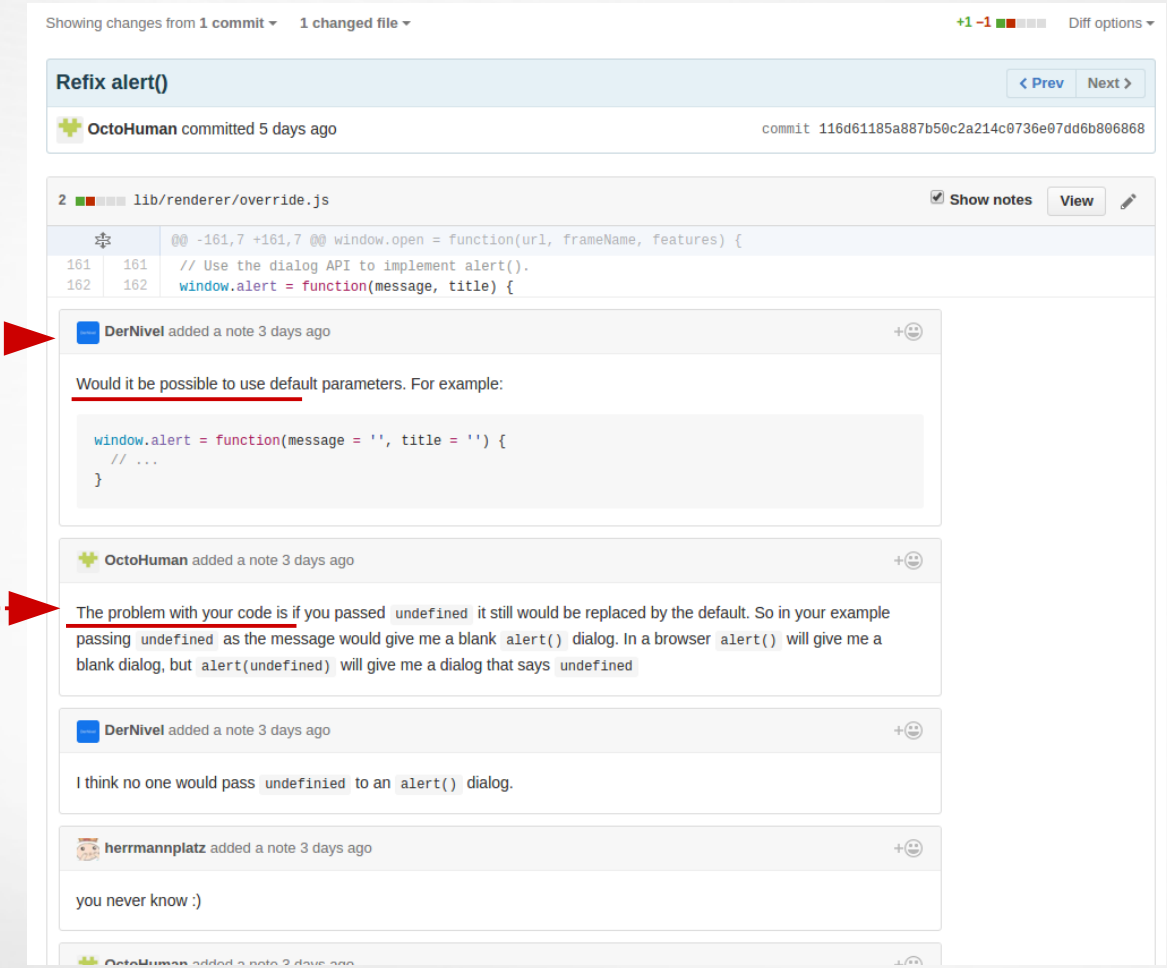
Commit 단위 Review, Discussion 예시

이 변화분 (commits) 을 통한 Review 와 Discussion 그리고 적용 (다수의 버그 / 오류 미리 차단)



Commit 단위 Review, Discussion 예시

이 변화분 (commits) **을 통한 Review 와 Discussion 그리고 적용** (다수의 버그 / 오류 미리 차단)



Commit 단위 Review, Discussion 예시

이 변화분 (commits) 을 통한 Review 와 Discussion 그리고 적용 (다수의 버그 / 오류 미리 차단)

Git 을 쓰는 이유

Commits(변화분) 단위 관리를 통해 (코드를 수정한 이유가 명확)

→ 개발과정 가시화 + 협업용이 + 유지보수 + 추적 용이 + 소스코드 품질 상승

Git 을 활용하는 오픈소스가 대표적인 증명 (IT 기업들의 높은 의존 , 소프트웨어 역사를 이끄는)

→ 카카오 , NHN 엔터테이먼트 등 SW 기업들의 pull-request 방식도입

Git 을 쓰는 이유

Junior 개발자 교육 효과 (프로개발자의 코드를 볼수 있는 기회)

Commits(변화분) 단위 관리를 통해 (코드를 수정한 이유가 명확)

개발과정 가시화 + 협업용이 + 유지보수 + 추적 용이 + 소스코드 품질 상승

Git 을 활용하는 오픈소스가 대표적인 증명 (IT 기업들의 높은 의존 , 소프트웨어 역사를 이끄는)

카카오 , NHN 엔터테이먼트 등 SW 기업들의 pull-request 방식도입

Contents

Chapter 6. Git 과 Github 통합구조

Github 란 ?

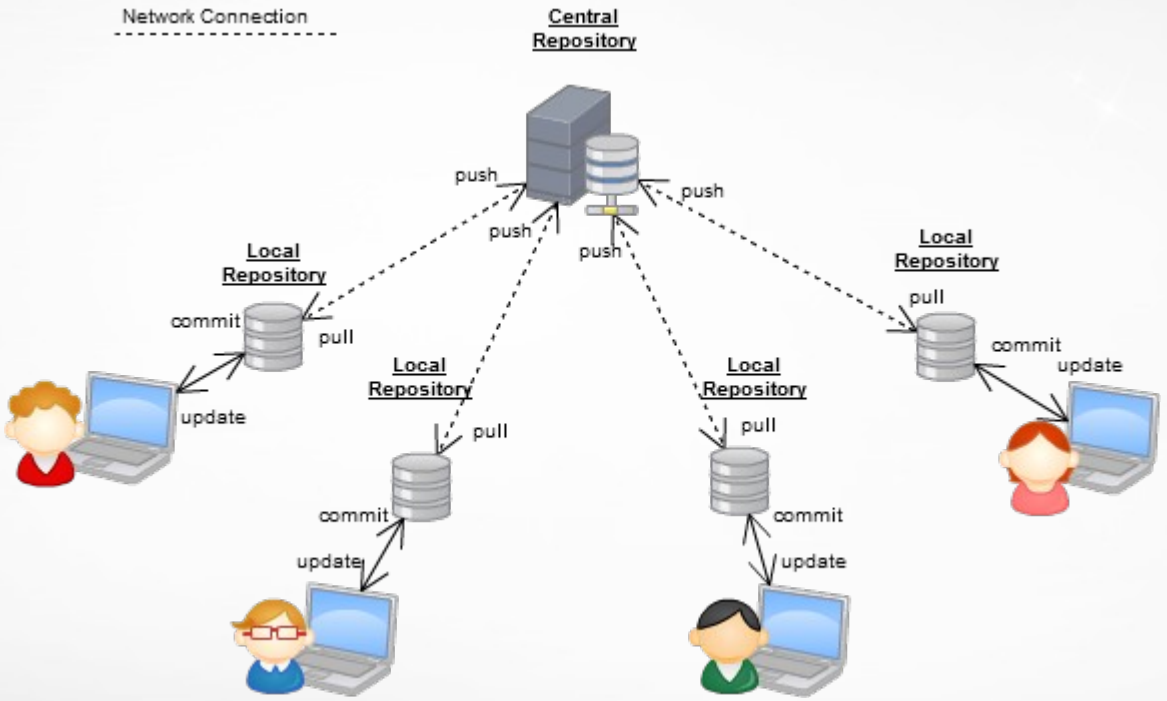
GitHub

Git 이라는 도구를 응용한 사이트

각종 Remote repository (원격저장소) 들의 집합소

Git 을 통한 작업의 흐름

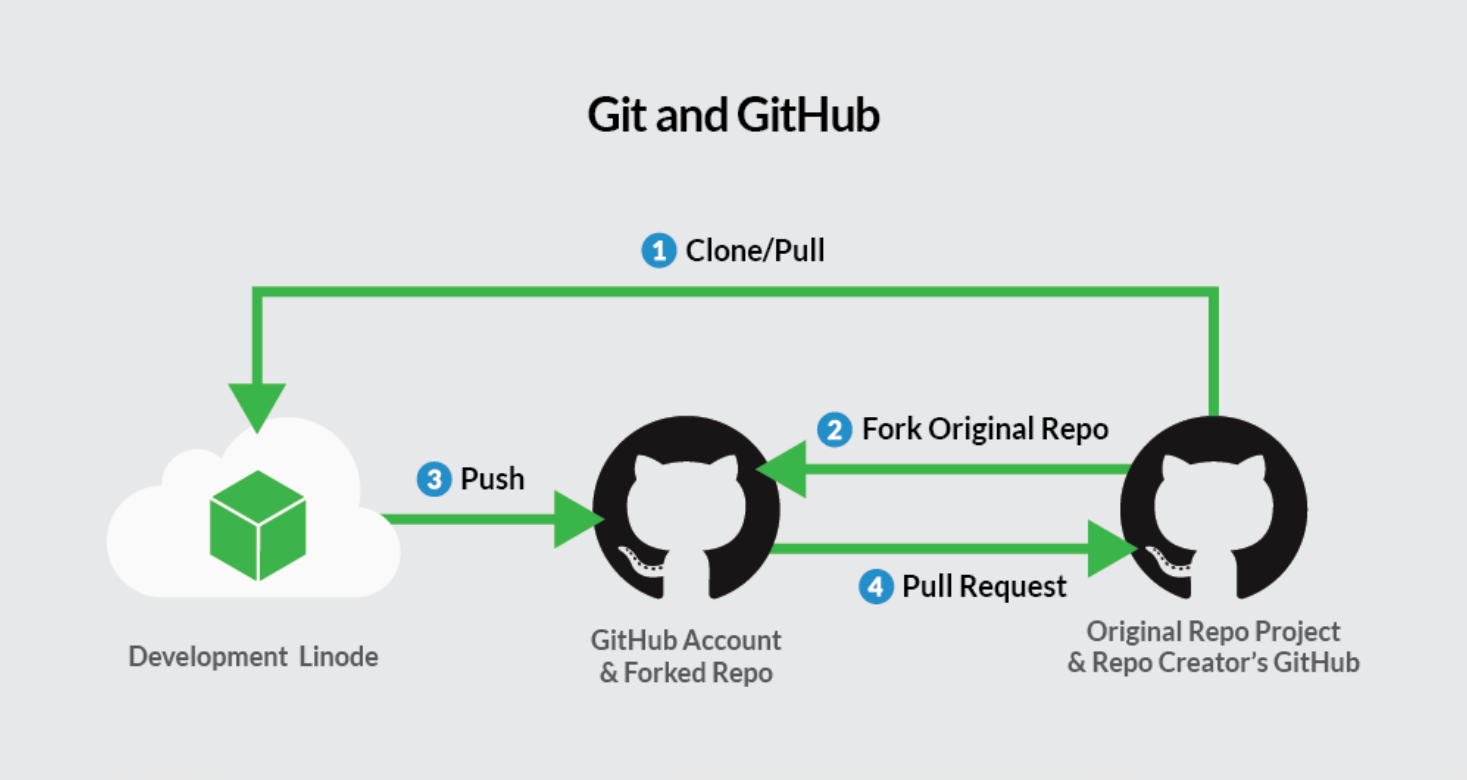
이제는 이 그림을 이해할 수 있다 .



http://jordankasper.com

Git pull-request 를 통한 작업의 흐름

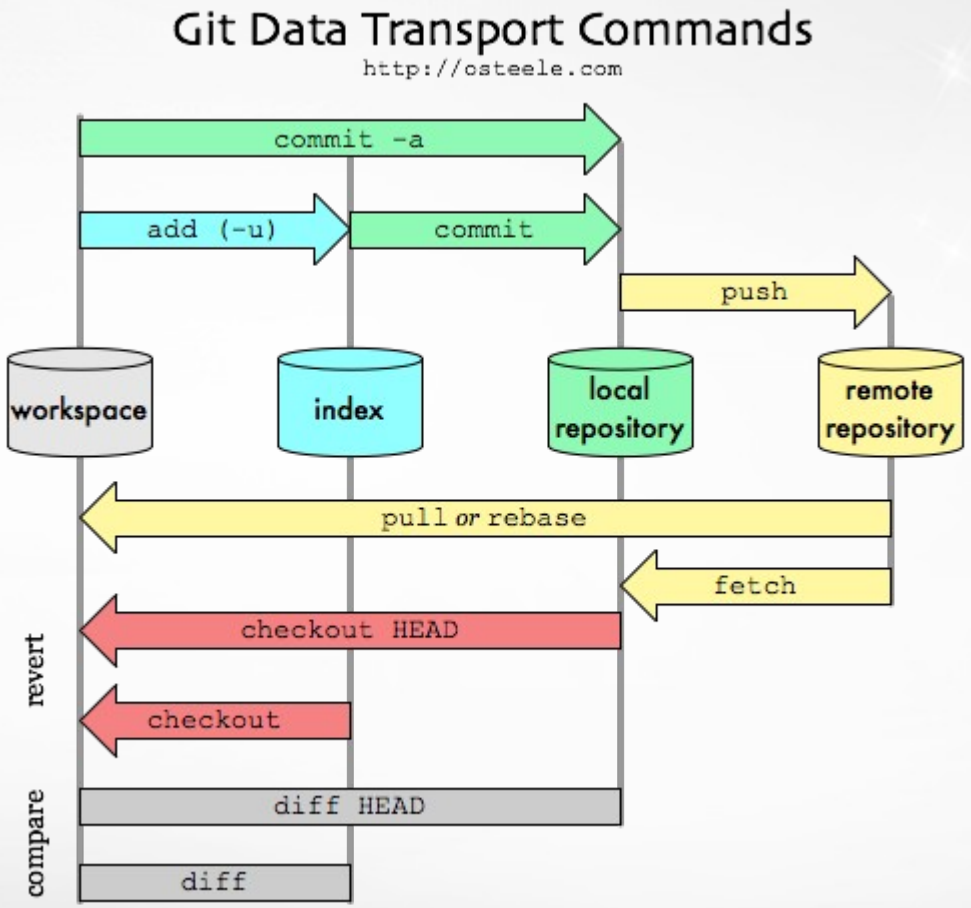
이제는 이 그림도 이해할 수 있다 .



https://www.linode.com/docs/applications/development/developing-git-github

Git 을 통한 작업의 흐름

이제는 이 그림도 이해할 수 있다 .



Opensource – Github / Not Github

Not Github

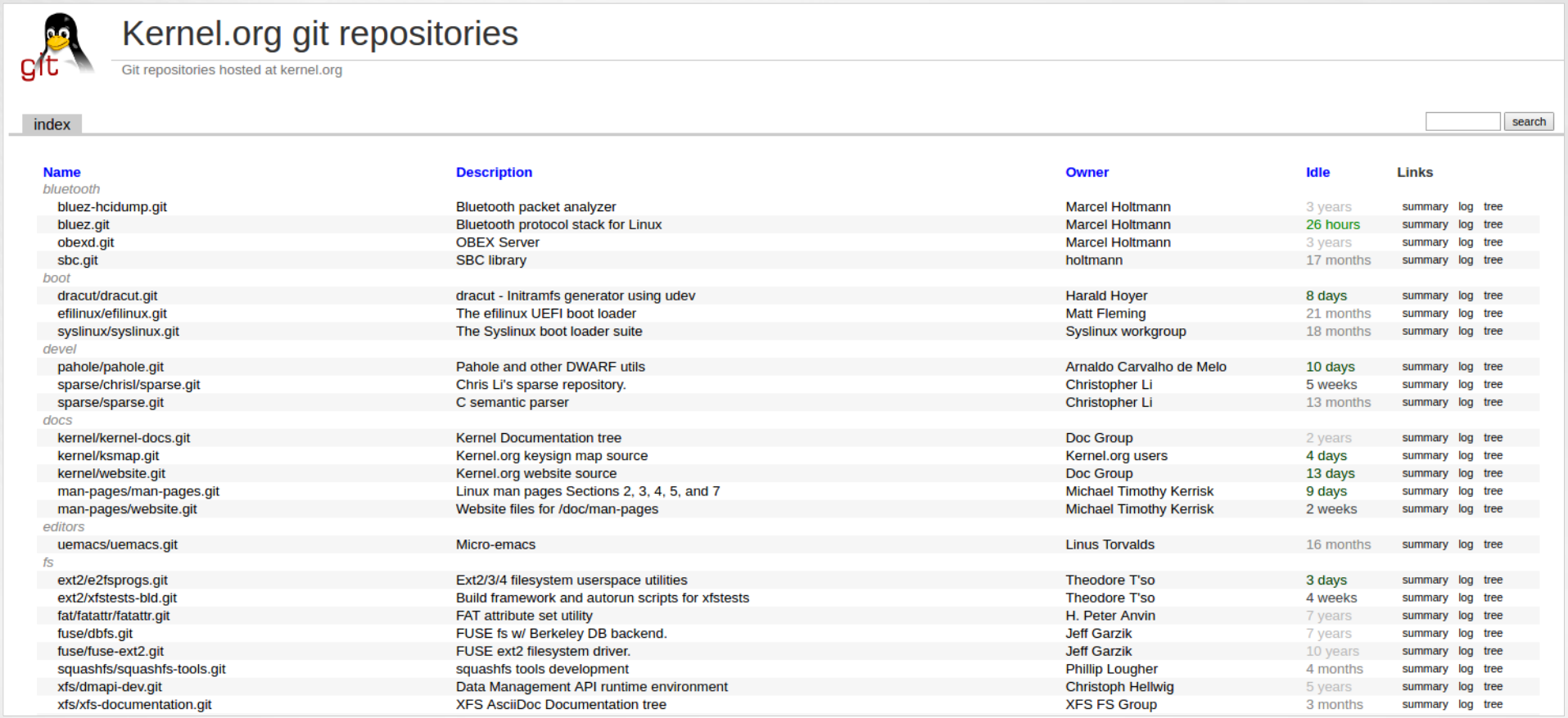
- Apache (http://git.apache.org)
- Linux kernel (http://git.kernel.org)
- GNU (http://git.savannah.gnu.org/cgit/)
- Webkit (git://git.webkit.org/WebKit.git)

...

Github

- Node.js (https://github.com/nodejs/node)
- Angular.js (https://github.com/angular/angular.js)
- Spring-boot (https://github.com/spring-projects/spring-boot)
- Rust (https://github.com/rust-lang/rust)
- Redis (https://github.com/antirez/redis)
- Flask (https://github.com/mitsuhiko/flask)

Not Github – Linux Kernel git repo



Kernel.org git repositories

Git repositories hosted at kernel.org

index

search

Name	Description	Owner	Idle	Links
bluetooth				
bluez-hcidump.git	Bluetooth packet analyzer	Marcel Holtmann	3 years	summary log tree
bluez.git	Bluetooth protocol stack for Linux	Marcel Holtmann	26 hours	summary log tree
obexd.git	OBEX Server	Marcel Holtmann	3 years	summary log tree
sbc.git	SBC library	holtmann	17 months	summary log tree
boot				
dracut/dracut.git	dracut - Initramfs generator using udev	Harald Hoyer	8 days	summary log tree
efilinux/efilinux.git	The efilinux UEFI boot loader	Matt Fleming	21 months	summary log tree
syslinux/syslinux.git	The Syslinux boot loader suite	Syslinux workgroup	18 months	summary log tree
devel				
pahole/pahole.git	Pahole and other DWARF utils	Arnaldo Carvalho de Melo	10 days	summary log tree
sparse/chrisl/sparse.git	Chris Li's sparse repository.	Christopher Li	5 weeks	summary log tree
sparse/sparse.git	C semantic parser	Christopher Li	13 months	summary log tree
docs				
kernel/kernel-docs.git	Kernel Documentation tree	Doc Group	2 years	summary log tree
kernel/ksmap.git	Kernel.org keysign map source	Kernel.org users	4 days	summary log tree
kernel/website.git	Kernel.org website source	Doc Group	13 days	summary log tree
man-pages/man-pages.git	Linux man pages Sections 2, 3, 4, 5, and 7	Michael Timothy Kerrisk	9 days	summary log tree
man-pages/website.git	Website files for /doc/man-pages	Michael Timothy Kerrisk	2 weeks	summary log tree
editors				
uemacs/uemacs.git	Micro-emacs	Linus Torvalds	16 months	summary log tree
fs				
ext2/e2fsprogs.git	Ext2/3/4 filesystem userspace utilities	Theodore T'so	3 days	summary log tree
ext2/xfstests-bld.git	Build framework and autorun scripts for xfstests	Theodore T'so	4 weeks	summary log tree
fat/fatattr/fatattr.git	FAT attribute set utility	H. Peter Anvin	7 years	summary log tree
fuse/dbfs.git	FUSE fs w/ Berkeley DB backend.	Jeff Garzik	7 years	summary log tree
fuse/fuse-ext2.git	FUSE ext2 filesystem driver.	Jeff Garzik	10 years	summary log tree
squashfs/squashfs-tools.git	squashfs tools development	Phillip Lougher	4 months	summary log tree
xfs/dmapi-dev.git	Data Management API runtime environment	Christoph Hellwig	5 years	summary log tree
xfs/xfs-documentation.git	XFS AsciiDoc Documentation tree	XFS FS Group	3 months	summary log tree

https://git.kernel.org/cgit

PATCH mail 개발 방식 소개

이 변화분 (commits) 을 통한 Review 와 Discussion 가능 (pull-request, PATCH mail)

Arnaldo Carvalho de Melo November 23, 2015 at 11:17 PM
To: Taeung Song Cc: Namhyung Kim, linux-kernel@vger.kernel.org, Ingo Molnar, Jiri Olsa
Re: [PATCH v11 RESEND 02/24] perf tools: Add perf-config document

Em Sun, Nov 22, 2015 at 07:11:56PM +0900, Taeung Song escreveu:
> Add perf-config document to describe the perf configuration
> and a subcommand 'list'

Worked this time, applied, will continue processing the other patches soon.

- Arnaldo

See More

tip-bot for Taeung Song Today at 5:20 PM
To: linux-tip-commits@vger.kernel.org Cc: Ingo Molnar, and 7 more...
[tip:perf/core] perf config: Add initial man page

Commit-ID: 7d6852432acb3b09fc3ec45dd65421d34eebe3b5
Gitweb: http://git.kernel.org/tip/7d6852432acb3b09fc3ec45dd65421d34eebe3b5
Author: Taeung Song <treeze.taeung@gmail.com>
AuthorDate: Sun, 22 Nov 2015 19:11:56 +0900
Committer: Arnaldo Carvalho de Melo <acme@redhat.com>
CommitDate: Mon, 23 Nov 2015 18:31:25 -0300

perf config: Add initial man page

Add perf-config document to describe the perf configuration and a 'list' subcommand.

Signed-off-by: Taeung Song <treeze.taeung@gmail.com>
Cc: Jiri Olsa <jolsa@redhat.com>
Cc: Namhyung Kim <namhyung@kernel.org>
Link: http://lkml.kernel.org/r/63AD9B57-7B8C-46F8-8F18-0FFEB9A6A1BC@gmail.com
Signed-off-by: Arnaldo Carvalho de Melo <acme@redhat.com>
---
tools/perf/Documentation/perf-config.txt | 103 +++++++++++++++++++++++++++++++++
1 file changed, 103 insertions(+)

tip-bot for Taeung Song Today at 5:20 PM
To: linux-tip-commits@vger.kernel.org
Cc: tglx@linutronix.de, Jiri Olsa, linux-kernel@vger.kernel.org, Ingo Molnar, hpa@zytor.com, Taeung Song, Namhyung Kim, acme@redhat.com hide
[tip:perf/core] perf tools: Add 'perf config' command

Commit-ID: 30862f2c5725c46afcfab5af710fdf5163bf0f81
Gitweb: http://git.kernel.org/tip/30862f2c5725c46afcfab5af710fdf5163bf0f81
Author: Taeung Song <treeze.taeung@gmail.com>
AuthorDate: Tue, 17 Nov 2015 22:53:21 +0900
Committer: Arnaldo Carvalho de Melo <acme@redhat.com>
CommitDate: Mon, 23 Nov 2015 18:31:24 -0300

perf tools: Add 'perf config' command

The perf configuration file contains many variables to change various aspects of each of its tools, including output, disk usage, etc.

But looking at the state of configuration is difficult and there's no documentation about config variables except for the variables in perfconfig.example exist.

So this patch adds a 'perf-config' command with a '--list' option.

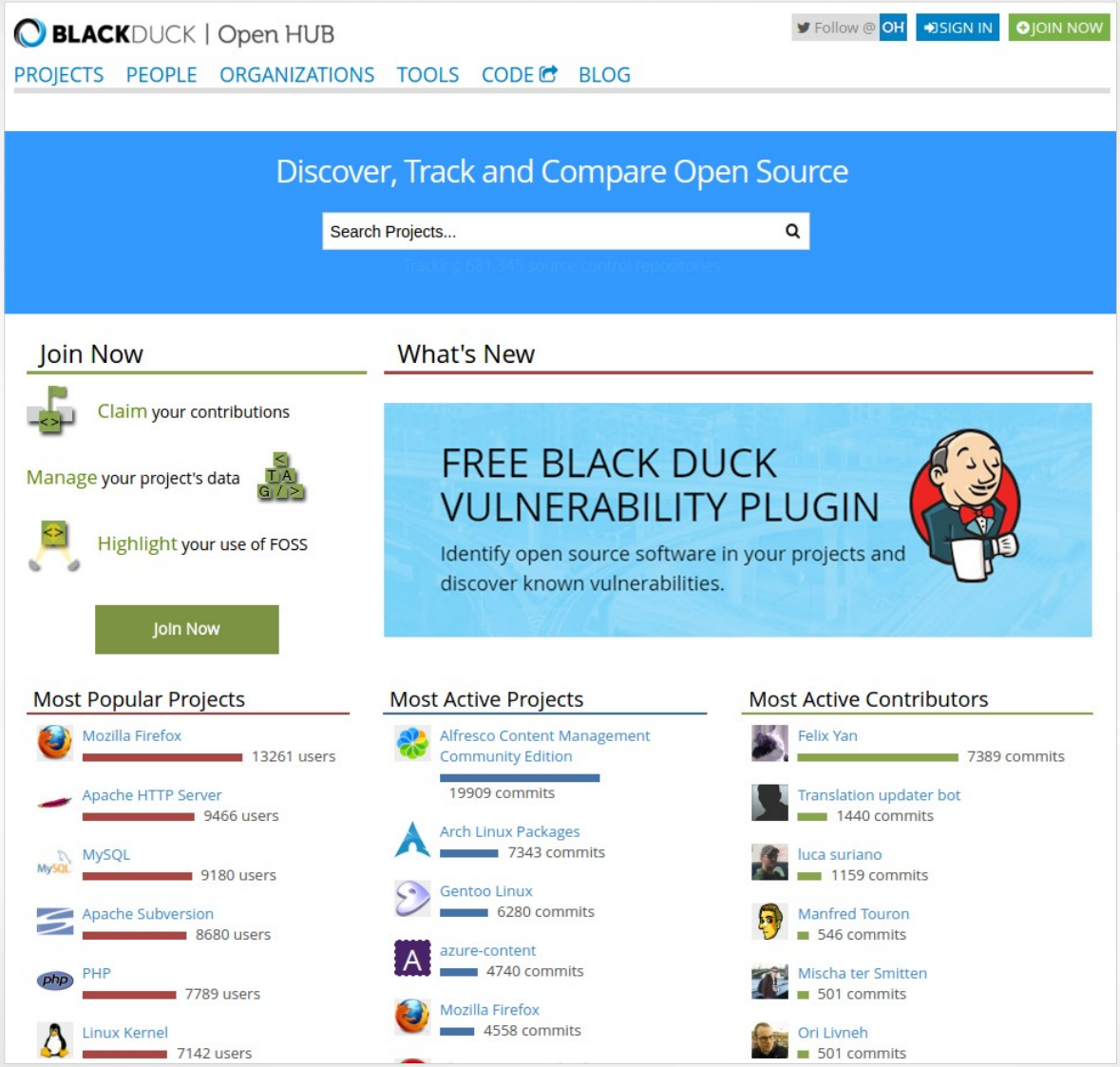
perf config [options]

display current perf config variables.
perf config -l | --list

Signed-off-by: Taeung Song <treeze.taeung@gmail.com>
Acked-by: Namhyung Kim <namhyung@kernel.org>
Cc: Jiri Olsa <jolsa@redhat.com>
Link: http://lkml.kernel.org/r/1447768424-17327-1-git-send-email-treeze.taeung@gmail.com
Signed-off-by: Arnaldo Carvalho de Melo <acme@redhat.com>
---
tools/perf/Build | 1 +
tools/perf/builtin-config.c | 66 +++
tools/perf/builtin.h | 1 +
tools/perf/command-list.txt | 1 +
tools/perf/perf.c | 1 +
5 files changed, 70 insertions(+)

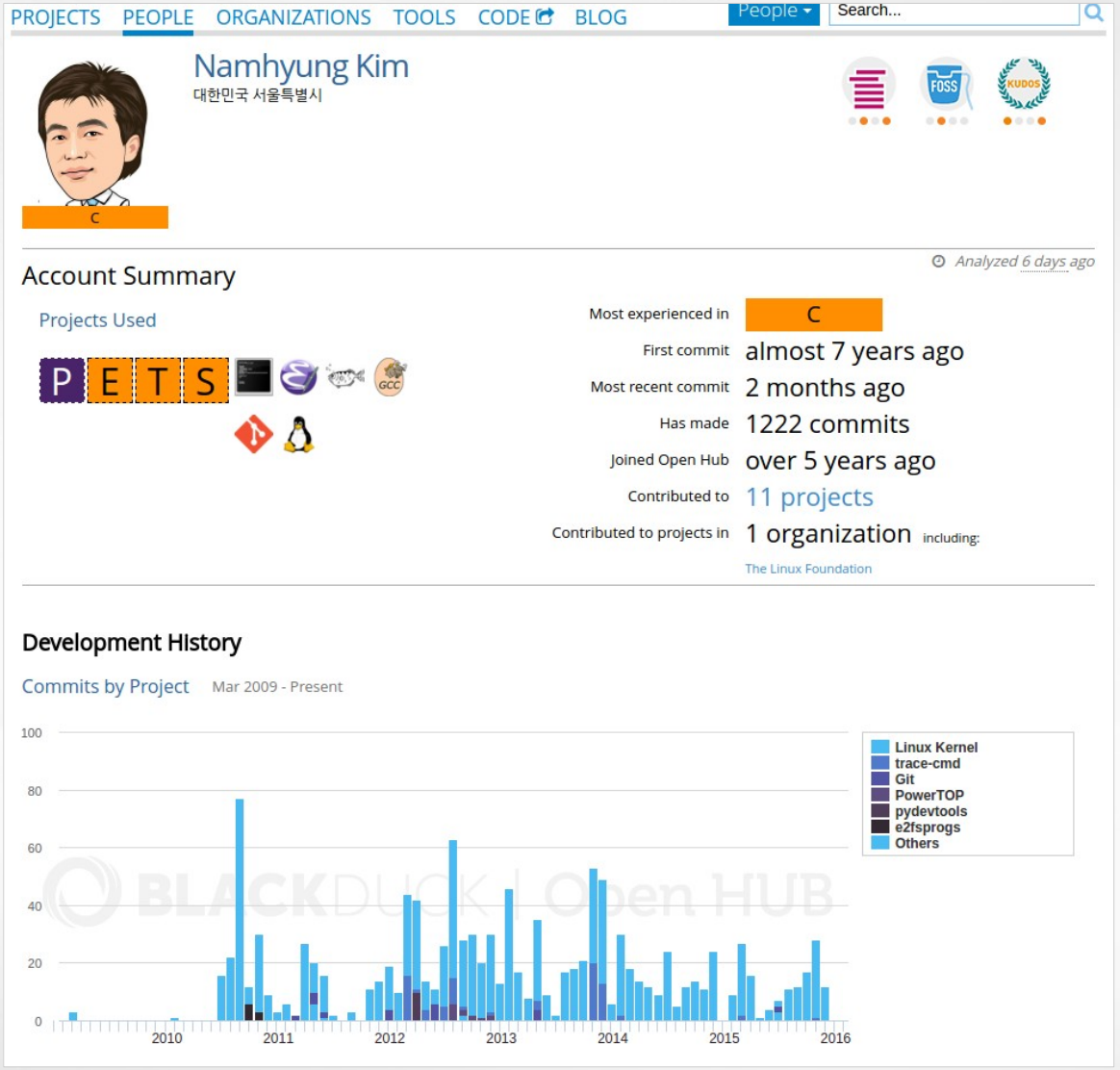
SK 주식회사 C&C

Opensource 각종 오픈소스



전세계 각종 오픈소스 통계 사이트 (https://www.openhub.net/)

Opensource 각종 오픈소스에 기여 통계



오픈소스 커미터 개인별 통계 뷰 (https://www.openhub.net/accounts/namhyung)

Opensource – Contributor, Committer, Maintainer

오픈소스 컨트리뷰션의 효과

실력 , 프로개발자 , 코어개발자 , 커미터

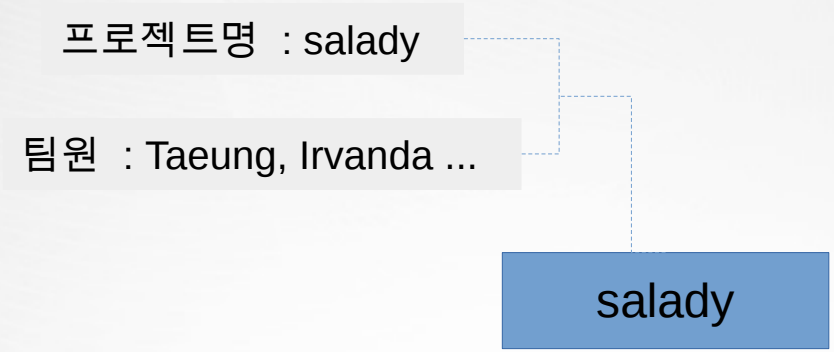
행복한 개발자

(참고)
https://wiki.kldp.org/wiki.php/KoreanOpenSourceCommitter
http://dirkriehle.com/publications/2014-2/the-open-source-software-developer-career-and-its-benefits/
http://www.payscale.com/research/US/Skill=Open_Source/Salary
http://nolongernew.blogspot.kr/2010/02/economic-motivation-of-open-source.html

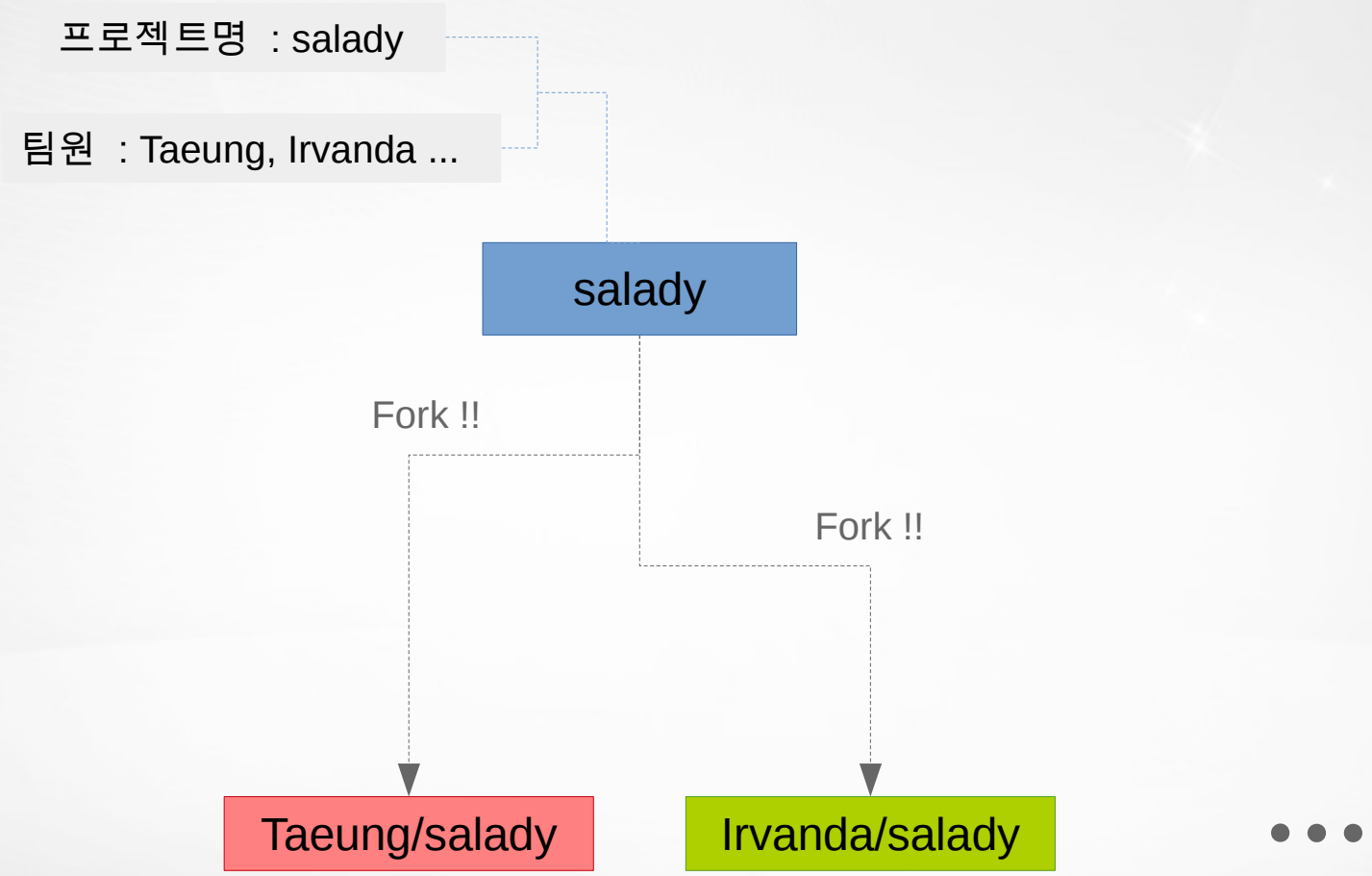
Contents

Chapter 7. 프로젝트와 Git 운용전략

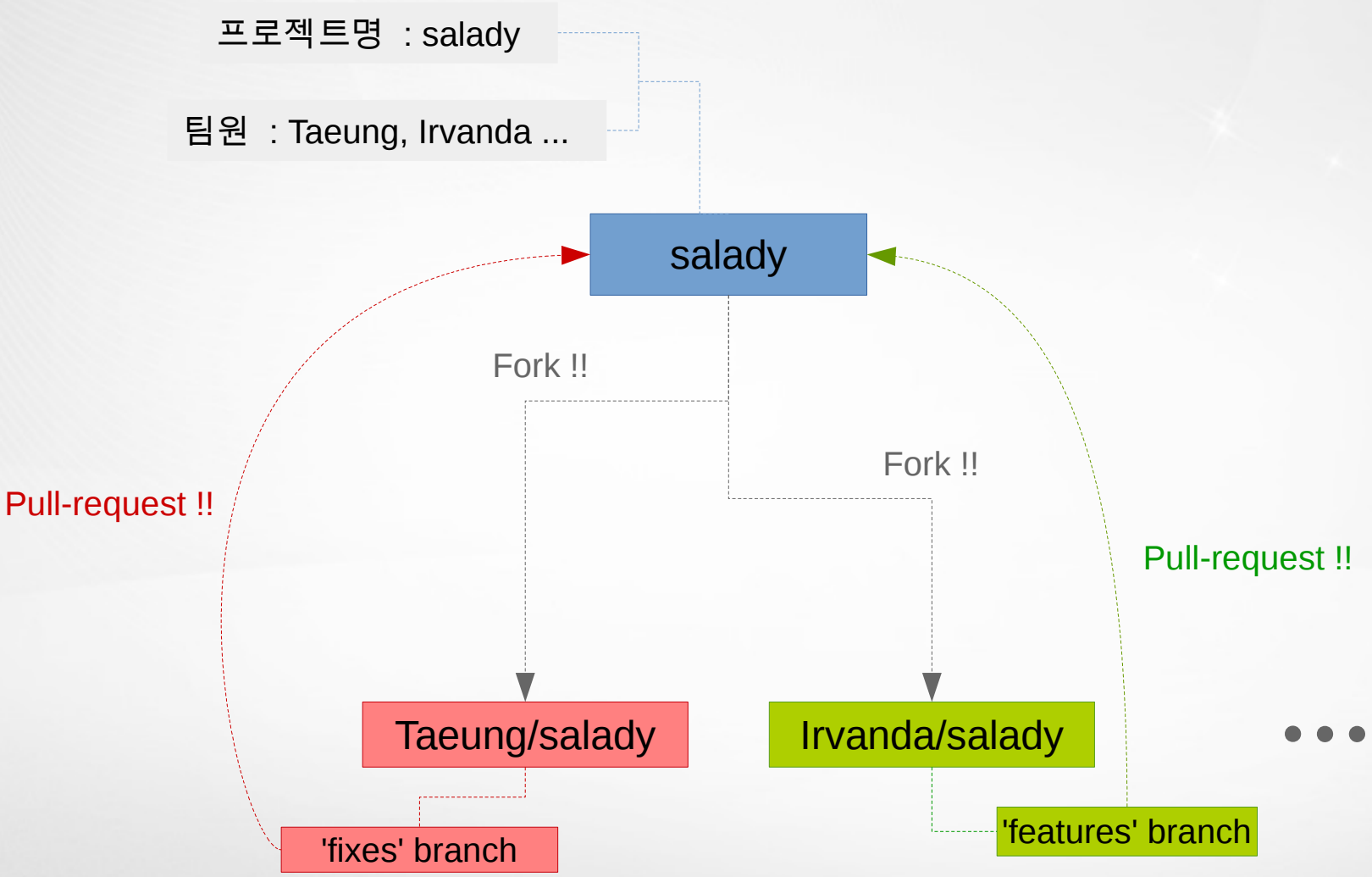
프로젝트와 Git 운용전략



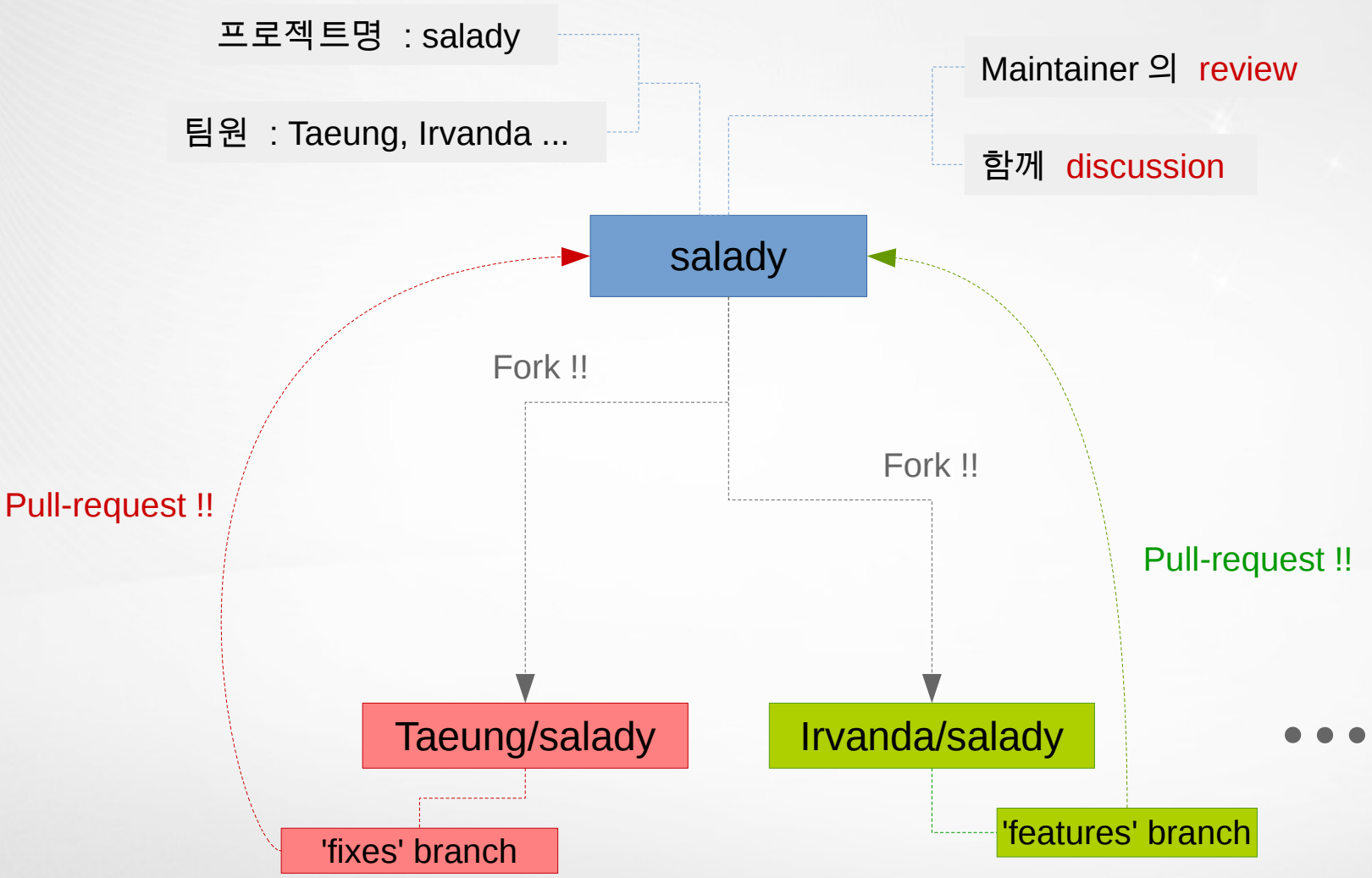
프로젝트와 Git 운용전략



프로젝트와 Git 운용전략



프로젝트와 Git 운용전략



Contents

Chapter 8. Git 관련 자주 묻는 질문과 대답

CHAPTER 08

Git 관련 자주 묻는 질문과 대답

1) Git 과 Github 의 차이는 ?

Git 은 각 컴퓨터 (local) 에 설치되어 소스코드관리가 가능한 프로그램이고
Github 는 remote 저장소가 있는 외부서버를 지칭한다 .

2) Commit 과 Push 의 차이는 ?

commit 은 local 작업폴더에 history 를 쌓는것이어서 외부망 (internet) 을 안쓰고
Push 는 remote 저장소 (Github 등) 에 history 를 쌓는것이어서 외부망 (intenet) 이 필요하다 .

3) Fetch 와 Pull 의 차이는 ?

Remote 저장소 (Github 등) 로 부터 최신 commit 정보들을 가져오는것은 매한가지이나
Fetch 는 가져와서 임시폴더 (.git) 에 저장하고
Pull 은 바로 현재 branch 에 merge 작업을 동반한다 .

Git 관련 자주 묻는 질문과 대답

4) Rebase 와 Merge 의 차이는 ?

둘다 두 branch 의 차이점 (commits) 을 합치는것은 매한가지나
Rebase 는 합치기 전에 되감기 (rewinding) 를 하고
Merge 는 안하고 합친다 .

5) Branch 명령엔 브랜치 변경 기능이 왜 없을까 ? Checkout 의 의미 ?

Checkout 이라는 의미는 이전 VCS 에서 부터 쓰이던 용어로 '도서관 대출' 의미를 갖고
이것을 통해서 브랜치를 Git 시스템으로 부터 가져온다 .

Git 관련 자주 묻는 질문과 대답

6) Reset 과 Stash 의 차이 ?

Stash 는 특정 add 한 목록 (index) 를 **임시 저장**을 해두는 기능이고
Reset 은 add 한 목록 (index) 를 **삭제**할때 이용할 수 있다 .

7) Revert 와 Reset 의 차이 ?

Revert 는 특정 commit 에 대한 **취소 commit** 을 만들어내는것이고
Reset 은 **특정 commit** 삭제할때 이용할 수 있다 .

8) Checkout -- 〈file name〉 과 Reset 의 차이 ?

Checkout 은 도서관에서 대출받듯 git 이라는 시스템을 통해서 파일 또는 브랜치 등을
가져오는 기능을 한다 . Reset 은 add 되어 만들어진 리스트 (index) 를 지울 수 있다 .

Git / GitHub Training

How to use git

- Git 간단한 정의 / 기능
- Git 기본 실습 (How)
- Git 이해하기 (Why, What)
- **Git 고급 (Advanced)**
- Opensource (with Git)

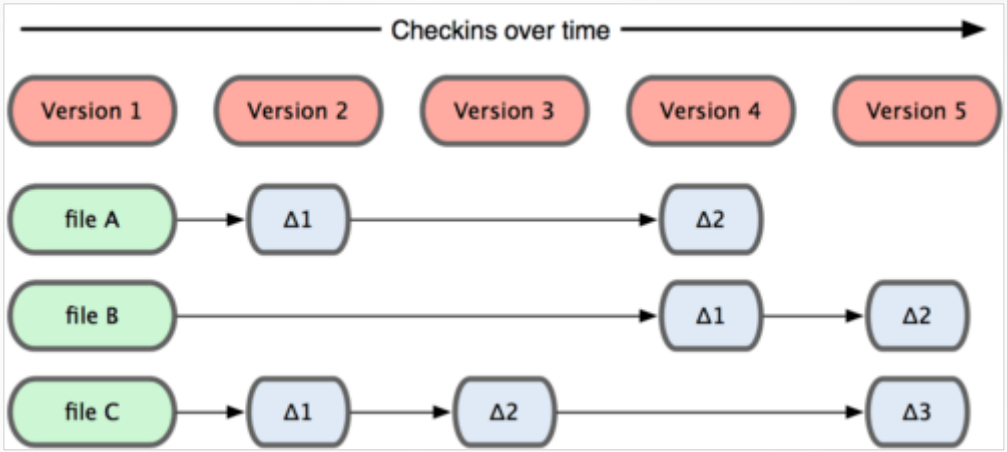
Contents

Chapter 9. Branch 운용방식과 Checkout 의 의미

CHAPTER 09

Git 에서 파일을 관리하는 원리

기존 VCS 시스템은 시간순으로 변화 (delta) 분을 저장 , 관리하는 방식

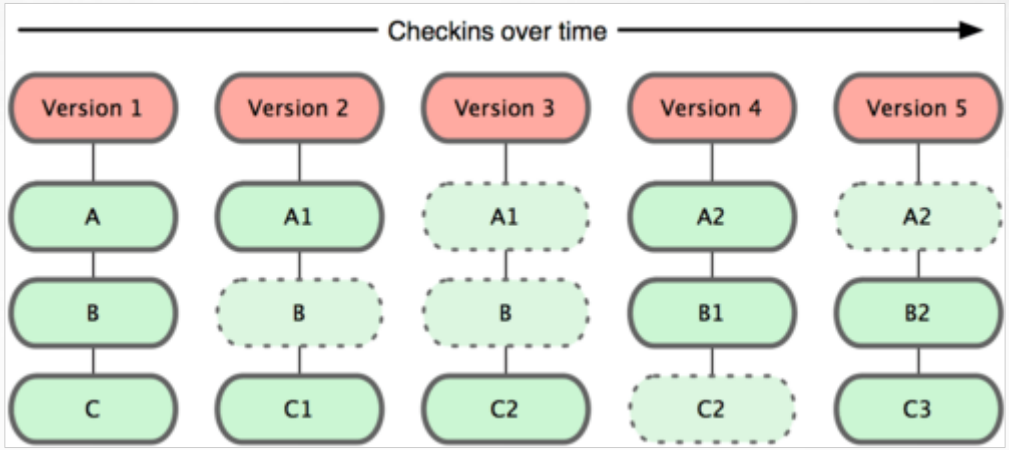


https://git-scm.com/book/ko/v2/%EC%8B%9C%EC%9E%91%ED%95%98%EA%B8%B0-Git-%EA%B8%B0%EC%B4%88

Subversion 또는 비슷한 기타 VCS 와 Git 의 가장 큰 차이점은 데이터를 저장 방법에 있다

Git 에서 파일을 관리하는 원리

Git 의 데이터는 파일 시스템의 스냅샷이라 할 수 있다



https://git-scm.com/book/ko/v2/%EC%8B%9C%EC%9E%91%ED%95%98%EA%B8%B0-Git-%EA%B8%B0%EC%B4%88

파일이 달라지지 않았으면 Git 은 성능을 위해서 파일을 저장하지 않고

단지 이전 상태의 파일에 대한 링크만 저장한다

CHAPTER 09

Branch 란 ?

Git 은 데이터를 Change Set 이나 변경사항 (Diff) 으로

기록하지 않고 일련의 스냅샷으로 기록

Git 의 브랜치는 커밋 사이를 가볍게 이동할 수 있는 어떤 포인터 같은 것

Checkout 의 의미

"Check out" 도서관에서 대출의미로 이해 해보자

* 참고 URL (http://english.stackexchange.com/questions/211634/what-checkout-is-supposed-to-mean-in-git-checkout-sentence)

1) 현재 브랜치의 최신 commit 역사까지의 기준으로 특정파일을 가져온다 (특정파일복구)

```
# git checkout -- <file name>
```

2) git 시스템을 이용하여 develop 이라는 브랜치를 가져온다 (브랜치 변경)

```
# git checkout develop
```

CHAPTER 09

Merge 로 2 개의 브랜치 합치기

1) 방금작업한 develop 브랜치가 현재 브랜치인지 확인하자 (status 로도 확인 가능)

```
# git branch
```

2) 추가 브랜치 만들어보자

```
# git checkout -b test
```

3) touch 로 빈파일하나 만들어서 commit 만들어보자 (; 로 명령어들을 연속적 실행 가능)

```
# touch test; git add test; git commit -sm “test”
```

4) 현재브랜치(develop)을 기준으로 추가브랜치(test)을 합치자

```
# git checkout develop; git status; git merge test
```

Branch 운용관리 Git Flow

http://stackoverflow.com/questions/35419185/git-workflow-example

Chapter 10. Rebase

CHAPTER 10

Rebase 기능

Rebase 사용하는 시나리오 !!

commit 을 역사의 한단위 '블럭'이라 하고 블럭들의 모임을 'tree'라 할때

내가 쌓은 블럭을 잠시 빼고

(뺀 나머지) 기준이 되는 tree 를 최신 업데이트 한 후에

그 위에 다시 내 블럭을 쌓아 올릴때 쓸 수 있다.

Stage 10. Rebase 하기

1) Chapter 4. 의 Stage 8. 에서 fork 후 clone 했던 프로젝트경로로 이동하여 upstream 을 추가하자

```
# git remote add upstream https://github.com/Taeung/git-training.git
```

2) upstream 의 dev 브랜치를 가져오자

```
# git fetch upstream dev
```

3) 현재 내 브랜치가 develop 인지 확인하자

```
# git status; git branch
```

4) rebase 하자

```
# git rebase upstream/dev
```

Stage 11. 중간에 낀 Commit 수정하기

고난이도 기능 중 하나 rebase --interactive

1) commit 최초기록 부터 2 번째 commit 을 수정해보자

```
# git rebase -i --root
```

2) vi 에디터가 열리면 수정하려는 commit(2 번째) 앞에 edit 이라 적자

3) 상태확인해서 rebase 진행 정상적인지 보고

```
# git status
```

4) commit 정보 수정("packing knapsack:" commit 메시지에 추가) 하고 --continue 로 마무리

```
# git commit --amend -sm "packing knapsack: Add knapsack problem PDF"
```

```
# git rebase --continue
```

안전한 Rebase 방법

1) 먼저 upstream 으로 부터 최신 commit 들을 .git 으로 받아온다

```
# git fetch upstream
```

2) 현재 작업한 commit 이 7 개라면 8 번째를 edit 지정하고 되감는다

```
# git rebase -i HEAD~8
```

3) 작업을 시작지점을 강제 업데이트한다

```
# git reset --hard upstream/master
```

4) 작업 시작지점이 업데이트 되었다면 continue 로 되감았던 내용을 풀어낸다

```
# git rebase --continue
```

Contents

Chapter 11. Stash, Clean, Config

CHAPTER 11

Stash 기능

Stash 란 add 하지 않은 변경사항 임시저장 해두는 기능

```
taeung ~/git/linux-perf/tools/perf
 :> git status
On branch perf/config/default-value-v6
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working
directory)

    modified:   util/config.c
    modified:   util/config.h

no changes added to commit (use "git add" and/or "git commit -a")
```

Stash 실습

1) add 작업이 이루어 지지 않은 변경사항에 대해서 임시저장을 한다

```
# git stash
```

2) 현재 임시저장된 사항에 대한 리스트를 확인한다

```
# git stash list
```

3) 특정 stash 기록을 적용한다

```
# git stash apply stash@{0}
```

4) 특정 stash 기록을 삭제한다

```
# git stash drop stash@{0}
```

CHAPTER 11

Clean 기능

추적되지 않거나 필요없는 파일 모두 청소하는 기능

```
taeung ~/git/linux-perf/tools/perf
 :> git status
On branch perf/config/default-value-v6
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    needless/
    unnecessary

nothing added to commit but untracked files present (use "git add" to track)
```

Clean 실습

1) 추적되지 않는 불필요한 파일을 삭제한다

```
# git clean -n
```

2) 강제옵션을 주고 삭제할 수도 있다

```
# git clean -f
```

3) 파일들 뿐만아니라 디렉토리 (폴더) 도 삭제를 원할때는 -d 옵션을 포함한다

```
# git clean -f -d
```

Config 기능과 설정파일 적용순서

Local Config : .git/config

Global Config : $HOME/.gitconfig

System Config : /etc/gitconfig

Config 실습

1) 모든 설정 내용을 출력한다

```
# git config -a
```

2) system 기준 config 파일 내용만 확인한다

```
# git config --system --list
```

3) global 기준 ($HOME/.gitconfig) 에 user 이름 설정한다

```
# git config --global user.name “Taeung Song”
```

Chapter 12. Blame, Cherry-pick, Submodule

Blame 이란

해당 소스라인 대해서 누가 마지막으로 수정을 했는지

commit ID 추적이 가능하다 .

```
07800601 Documentation/perf_counter/config.c (Ingo Molnar              2009-04-20 15:00:56 +0200   1) /*
5f9273d6 tools/perf/util/config.c            (Namhyung Kim             2011-12-13 22:52:03 +0900   2)  * config.c
5f9273d6 tools/perf/util/config.c            (Namhyung Kim             2011-12-13 22:52:03 +0900   3)  *
5f9273d6 tools/perf/util/config.c            (Namhyung Kim             2011-12-13 22:52:03 +0900   4)  * Helper functions for p..
5f9273d6 tools/perf/util/config.c            (Namhyung Kim             2011-12-13 22:52:03 +0900   5)  * Originally copied from..
07800601 Documentation/perf_counter/config.c (Ingo Molnar              2009-04-20 15:00:56 +0200   6)  *
07800601 Documentation/perf_counter/config.c (Ingo Molnar              2009-04-20 15:00:56 +0200   7)  * Copyright (C) Linus Torval..
07800601 Documentation/perf_counter/config.c (Ingo Molnar              2009-04-20 15:00:56 +0200   8)  * Copyright (C) Johannes S..
07800601 Documentation/perf_counter/config.c (Ingo Molnar              2009-04-20 15:00:56 +0200   9)  *
07800601 Documentation/perf_counter/config.c (Ingo Molnar              2009-04-20 15:00:56 +0200  10)  */
07800601 Documentation/perf_counter/config.c (Ingo Molnar              2009-04-20 15:00:56 +0200  11) #include "util.h"
07800601 Documentation/perf_counter/config.c (Ingo Molnar              2009-04-20 15:00:56 +0200  12) #include "cache.h"
4b6ab94e tools/perf/util/config.c            (Josh Poimboeuf           2015-12-15 09:39:39 -0600  13) #include <subcmd/exec-cmd.h>
0b93da17 tools/perf/util/config.c            (Namhyung Kim             2014-01-14 12:02:15 +0900  14) #include "util/hist.h"
aa61fd05 tools/perf/util/config.c            (Wang Nan                 2015-07-21 11:13:34 +0000  15) #include "util/llvm-utils.h"
20105ca1 tools/perf/util/config.c            (Taeung Song              2016-04-14 16:53:18 +0900  16) #include "config.h"
07800601 Documentation/perf_counter/config.c (Ingo Molnar              2009-04-20 15:00:56 +0200  17)
```

Blame 활용

1) 어떤 파일이든 누가 어느 라인을 수정했는지 파악해보자

```
# git blame report_card.c
```

2) 해당 commit ID 를 이용하여 그 당시 commit 정보확인

```
# git show <commit ID>
```

CHAPTER 12

Cherry-pick 기능

특정 commit 혹은 여러개의 commit 을 가져올때 사용하는 기능

주로 다른 브랜치에 존재하는 commit 을 가져올때 사용한다 .

Cherry-pick 실습

1) 하나의 커밋을 가져오기

```
# git cherry-pick <commit ID>
```

2) 범위 지정하여 여러개의 커밋 가져오기 “ git cherry-pick (더오래된커밋)..(기준커밋)”

```
# git cherry-pick A..B
```

3) 범위 지정하여 여러개의 커밋 가져오기 (브랜치 명으로 예시)

```
# git cherry-pick perf-infra-v9~2..perf-infra-v9
```

CHAPTER 12

Submodule 기능

프로젝트를 수행하다 보면 다른 프로젝트를 함께 사용해야 하는 경우가 종종 있다.

Git 저장소 안에 다른 Git 저장소를 디렉토리로 분리해 넣는 것이 서브모듈이다.

CHAPTER 12

Submodule 실습

1) 서브모듈 추가하기

```
# git submodule add <추가 repository URL>
```

2) 추가한 모듈에 대해서 확인해보자

```
# cat .gitmodules
```

Submodule 실습

1) 서브모듈을 포함하는 프로젝트 clone 하기

```
# git clone <해당 프로젝트 repository URL>
```

2) 포함된 서브모듈정보를 확인해본다

```
# cat .gitmodules
```

3) 원래 포함해야하는 서브모듈에 대한 초기화

```
# git submodule init
```

4) 비어있던 서브모듈 소스 받아오기 (업데이트)

```
# git submodule update
```

Git / GitHub Training

How to use git

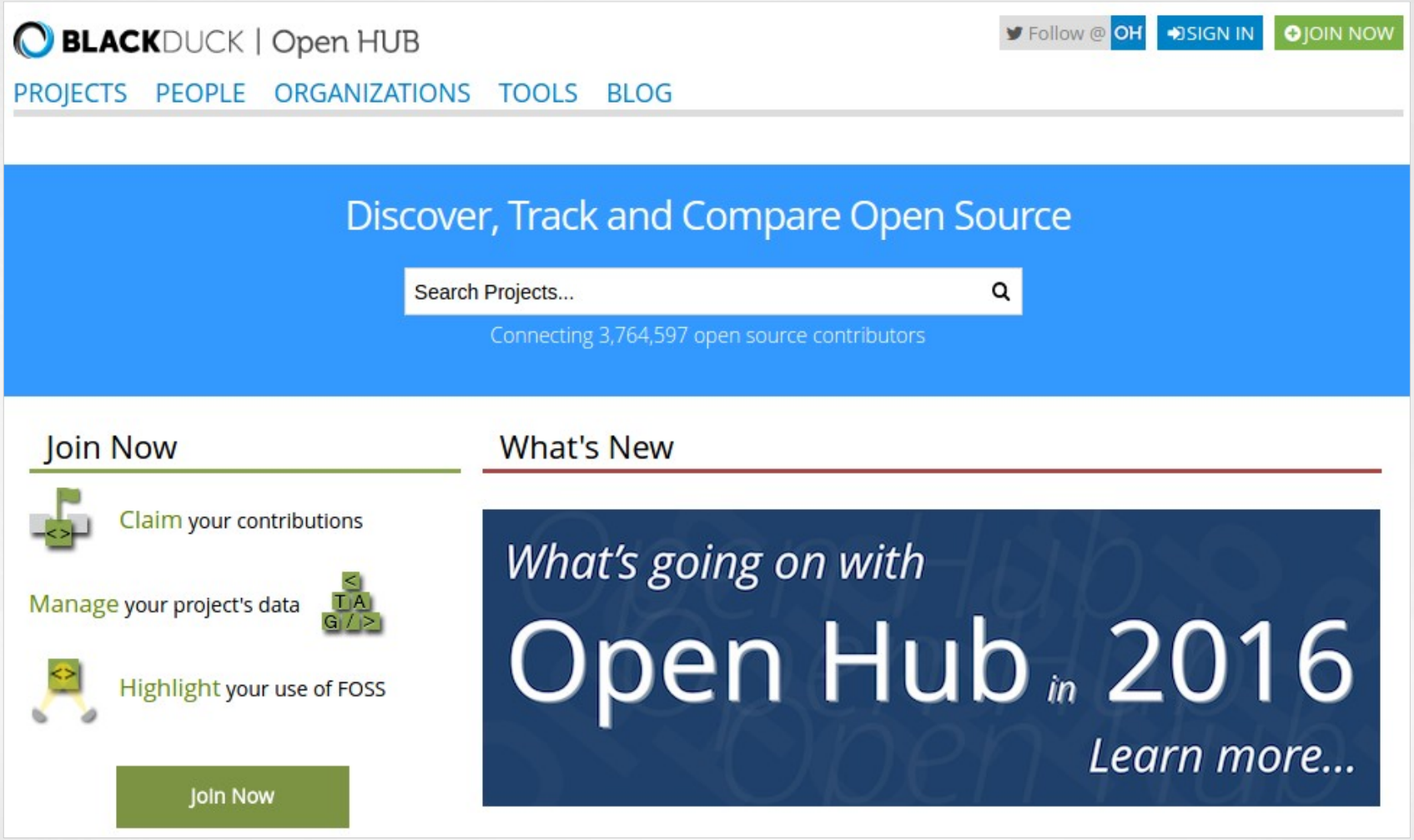
- Git 간단한 정의 / 기능
- Git 기본 실습 (How)
- Git 이해하기 (Why, What)
- Git 고급 (Advanced)
- Opensource (with Git)

Contents

Chapter 13. Git 을 활용해 여러가지 오픈소스 살펴보기

CHAPTER 13

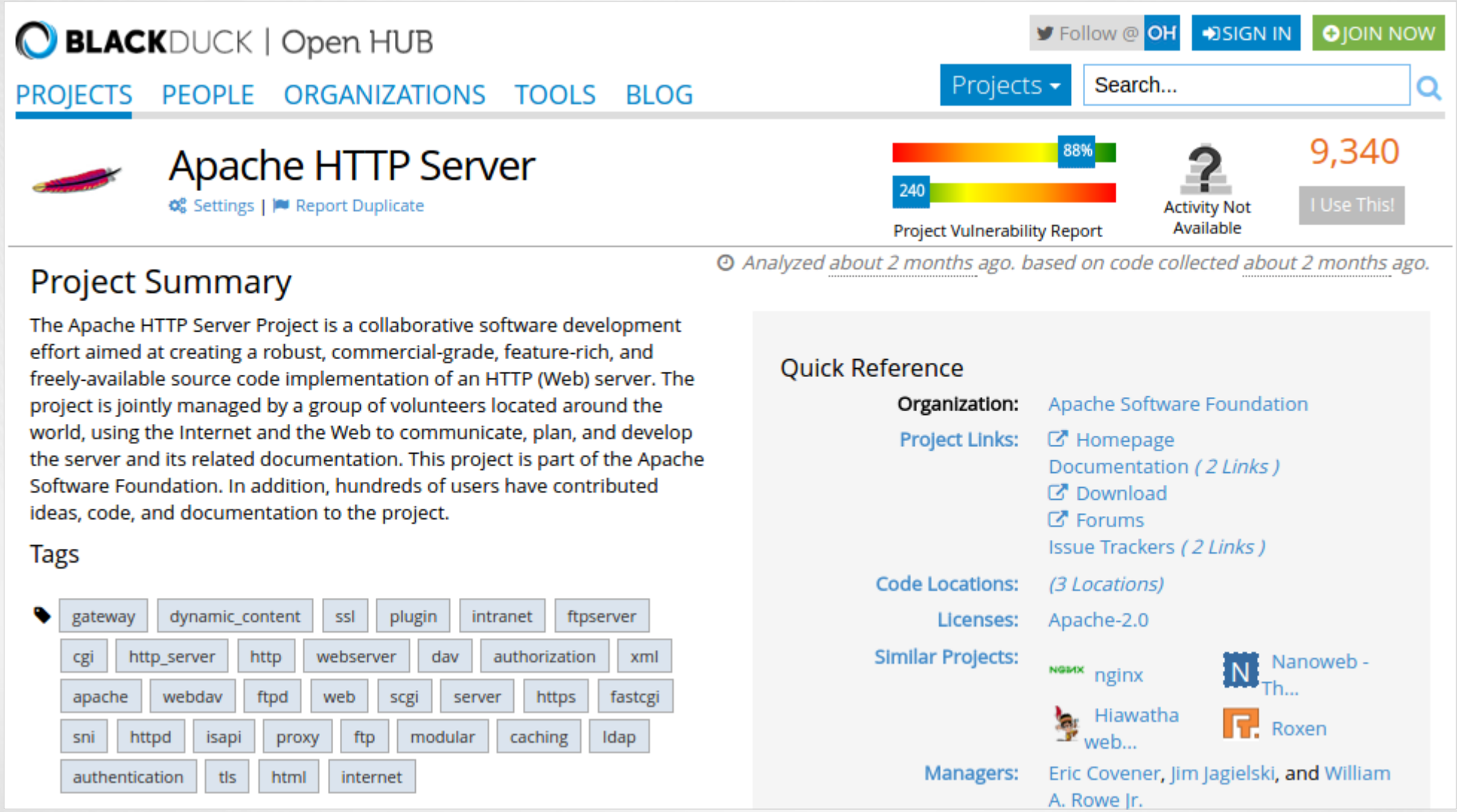
Openhub 로 다양한 오픈소스 찾아보기



https://www.openhub.net/

CHAPTER 13

Openhub 로 다양한 오픈소스 찾아보기



https://www.openhub.net/p/apache

Openhub 로 다양한 오픈소스 찾아보기

In a Nutshell, Apache HTTP Server...

... has had 76,803 commits made by 120 contributors representing 1,808,624 lines of code

... is mostly written in C with an average number of source code comments

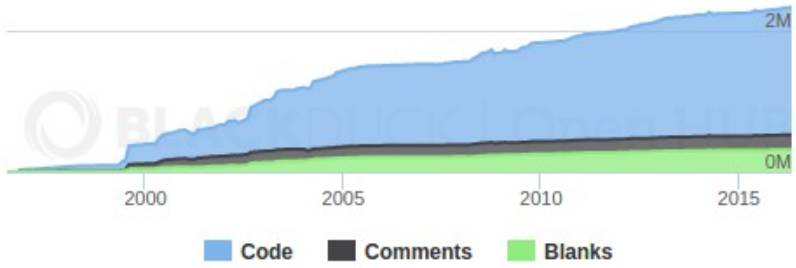
... has a well established, mature codebase maintained by a very large development team with increasing Y-O-Y commits

... took an estimated 508 years of effort (COCOMO model) starting with its first commit in July, 1996 ending with its most recent commit about 2 months ago

Languages

XML	53%	C	33%
Forth	10%	20 Other	4%

Lines of Code



Activity

30 Day Summary

Apr 24 2016 — May 24 2016

257 Commits

16 Contributors

including 1 new contributor

12 Month Summary

May 24 2015 — May 24 2016

2774 Commits

Up + 450 (19%) from previous 12 months

28 Contributors

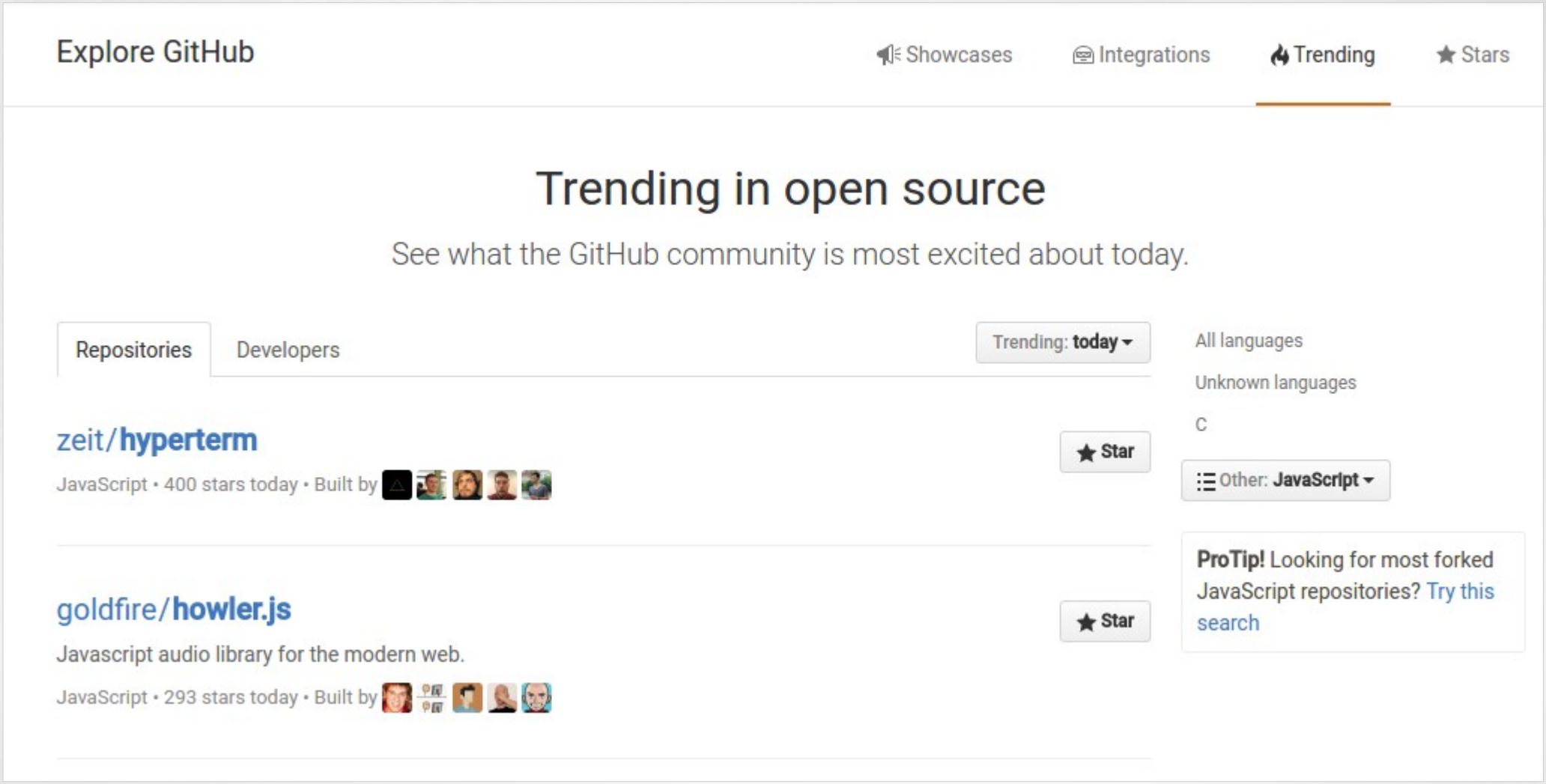
Down 0 (0%) from previous 12 months

Commits per Month

https://www.openhub.net/p/apache

CHAPTER 13

Github 에서 다양한 오픈소스 찾아보기



https://github.com/trending/javascript

Git 을 활용한 다양한 오픈소스 분석

1) 해당 프로젝트 clone 을 한다

```
# git clone <repository URL>
```

2) shortlog 명령을 이용해서 통계를 내보자

```
# git shortlog
```

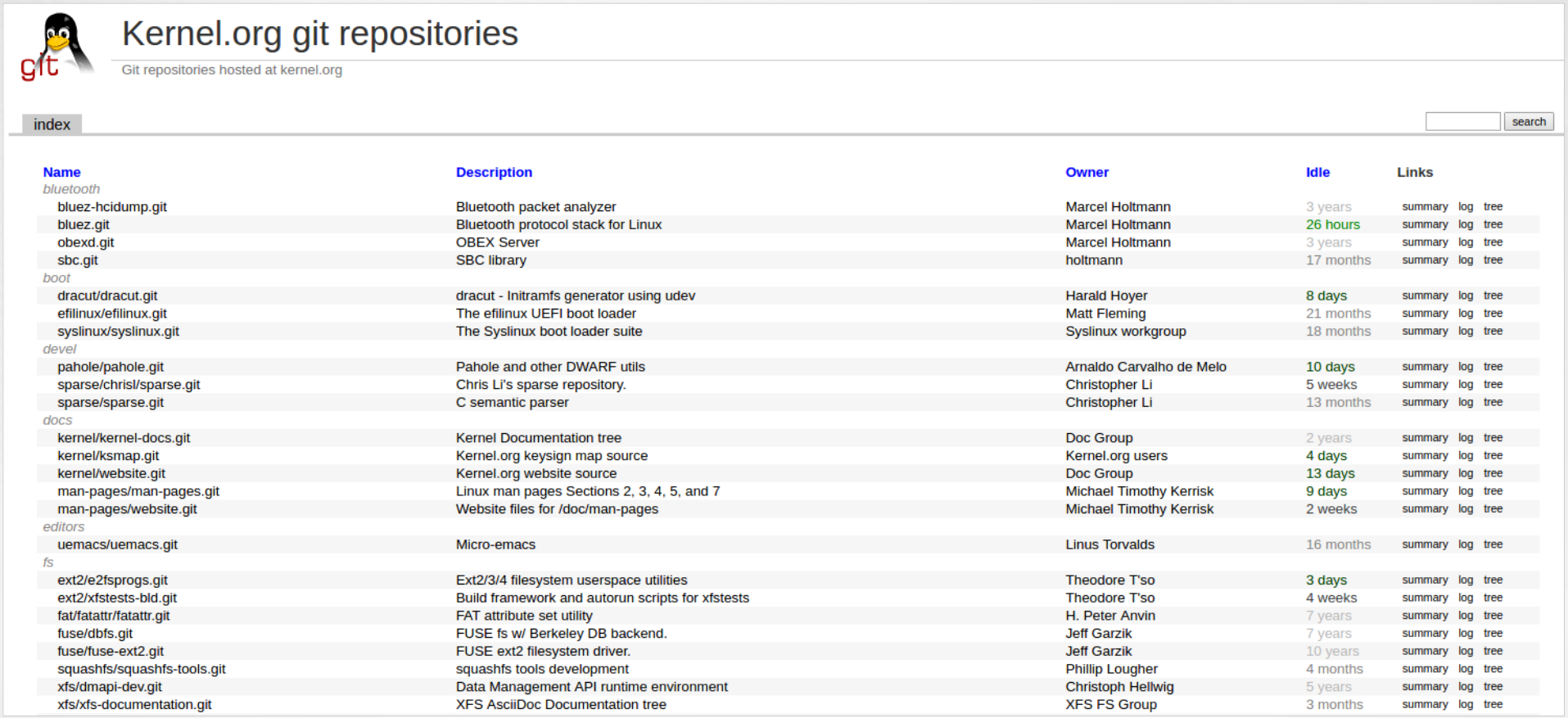
3) tig, gitk, log 명령등으로 commit 들을 살펴보자

```
# tig
```

4) 궁금하거나 영향력있는 개발자의 commit 만 살펴보자

```
# tig --author=”Taeung Song”
```

Linux Kernel 오픈소스 살펴보기



Kernel.org git repositories

Git repositories hosted at kernel.org

index

search

Name	Description	Owner	Idle	Links
bluetooth				
bluez-hcidump.git	Bluetooth packet analyzer	Marcel Holtmann	3 years	summary log tree
bluez.git	Bluetooth protocol stack for Linux	Marcel Holtmann	26 hours	summary log tree
obexd.git	OBEX Server	Marcel Holtmann	3 years	summary log tree
sbc.git	SBC library	holtmann	17 months	summary log tree
boot				
dracut/dracut.git	dracut - Initramfs generator using udev	Harald Hoyer	8 days	summary log tree
efilinux/efilinux.git	The efilinux UEFI boot loader	Matt Fleming	21 months	summary log tree
syslinux/syslinux.git	The Syslinux boot loader suite	Syslinux workgroup	18 months	summary log tree
devel				
pahole/pahole.git	Pahole and other DWARF utils	Arnaldo Carvalho de Melo	10 days	summary log tree
sparse/chrisl/sparse.git	Chris Li's sparse repository.	Christopher Li	5 weeks	summary log tree
sparse/sparse.git	C semantic parser	Christopher Li	13 months	summary log tree
docs				
kernel/kernel-docs.git	Kernel Documentation tree	Doc Group	2 years	summary log tree
kernel/ksmap.git	Kernel.org keysign map source	Kernel.org users	4 days	summary log tree
kernel/website.git	Kernel.org website source	Doc Group	13 days	summary log tree
man-pages/man-pages.git	Linux man pages Sections 2, 3, 4, 5, and 7	Michael Timothy Kerrisk	9 days	summary log tree
man-pages/website.git	Website files for /doc/man-pages	Michael Timothy Kerrisk	2 weeks	summary log tree
editors				
uemacs/uemacs.git	Micro-emacs	Linus Torvalds	16 months	summary log tree
fs				
ext2/e2fsprogs.git	Ext2/3/4 filesystem userspace utilities	Theodore T'so	3 days	summary log tree
ext2/xfstests-bld.git	Build framework and autorun scripts for xfstests	Theodore T'so	4 weeks	summary log tree
fat/fatattr/fatattr.git	FAT attribute set utility	H. Peter Anvin	7 years	summary log tree
fuse/dbfs.git	FUSE fs w/ Berkeley DB backend.	Jeff Garzik	7 years	summary log tree
fuse/fuse-ext2.git	FUSE ext2 filesystem driver.	Jeff Garzik	10 years	summary log tree
squashfs/squashfs-tools.git	squashfs tools development	Phillip Lougher	4 months	summary log tree
xfs/dmapi-dev.git	Data Management API runtime environment	Christoph Hellwig	5 years	summary log tree
xfs/xfs-documentation.git	XFS AsciiDoc Documentation tree	XFS FS Group	3 months	summary log tree

https://git.kernel.org/cgit

Git 으로 Linux Kernel 오픈소스 살펴보기

1) Linux Kernel 공식 repository URL 로 프로젝트 받기

```
# git clone git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git
```

2) shortlog 명령을 이용해서 통계를 내보자

```
# git shortlog
```

3) tig, gitk, log 명령등으로 commit 들을 살펴보자

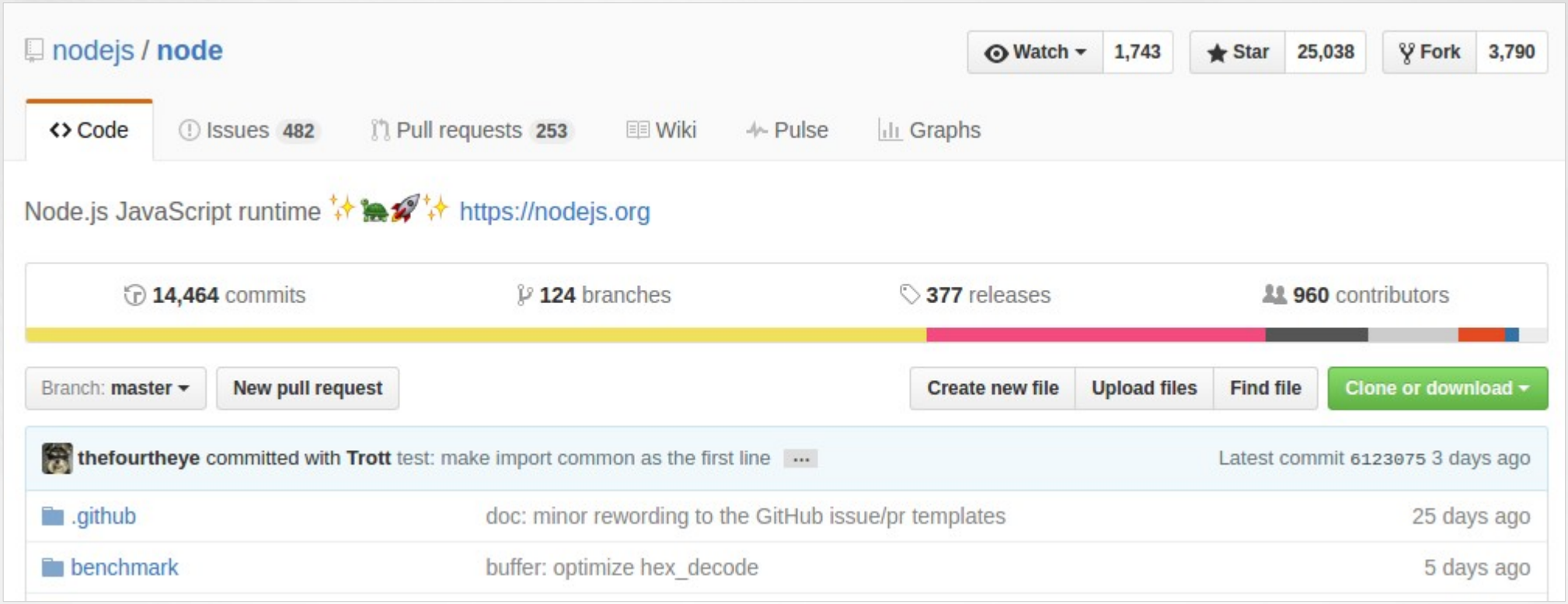
```
# tig
```

4) 궁금하거나 영향력있는 개발자의 commit 만 살펴보자

```
# tig --author=”Namhyung Kim”
```

CHAPTER 13

Node.js 오픈소스 살펴보기



https://github.com/nodejs/node

CHAPTER 13

Git 으로 Node.js 오픈소스 살펴보기

1) Linux Kernel 공식 repository URL 로 프로젝트 받기

```
# git clone git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git
```

2) shortlog 명령을 이용해서 통계를 내보자

```
# git shortlog
```

3) tig, gitk, log 명령등으로 commit 들을 살펴보자

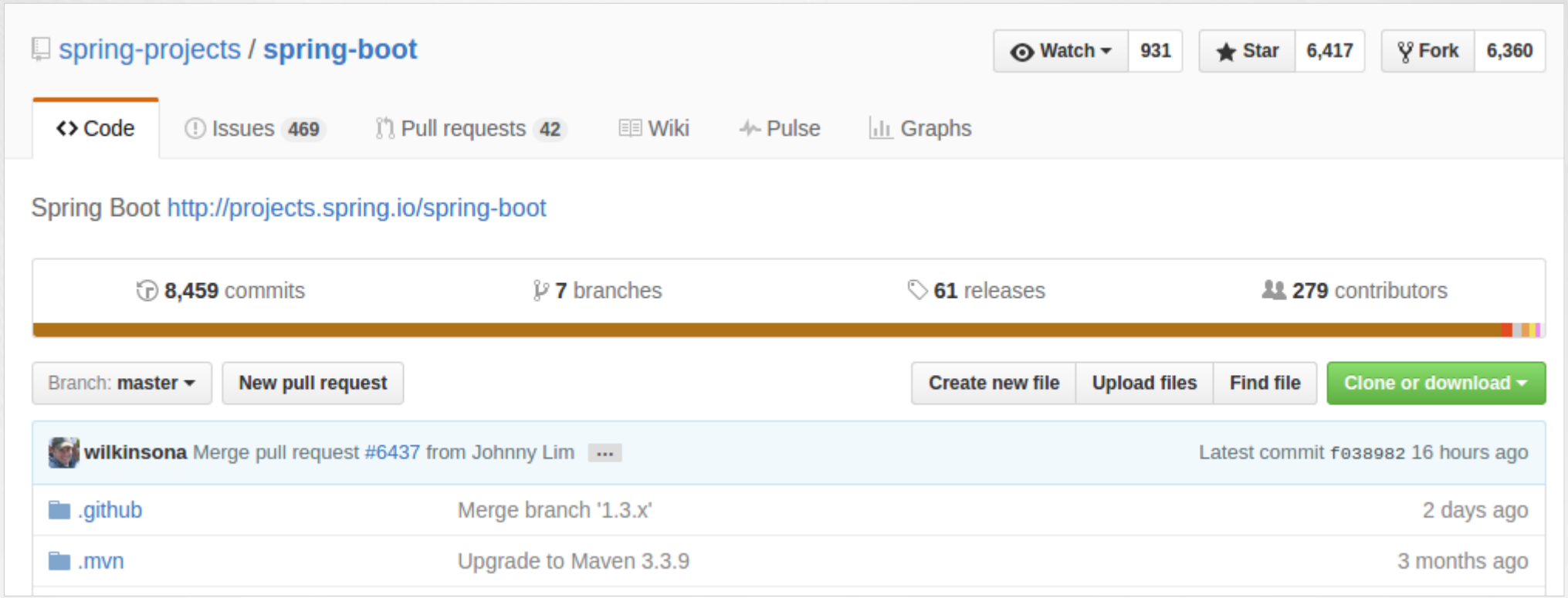
```
# tig
```

4) 궁금하거나 영향력있는 개발자의 commit 만 살펴보자

```
# tig --author=”Ben Noordhuis”
```

CHAPTER 13

Springboot 오픈소스 살펴보기



https://github.com/spring-projects/spring-boot

CHAPTER 13

Git 으로 Springboot 오픈소스 살펴보기

1) Linux Kernel 공식 repository URL 로 프로젝트 받기

```
# git clone https://github.com/spring-projects/spring-boot.git
```

2) shortlog 명령을 이용해서 통계를 내보자

```
# git shortlog
```

3) tig, gitk, log 명령등으로 commit 들을 살펴보자

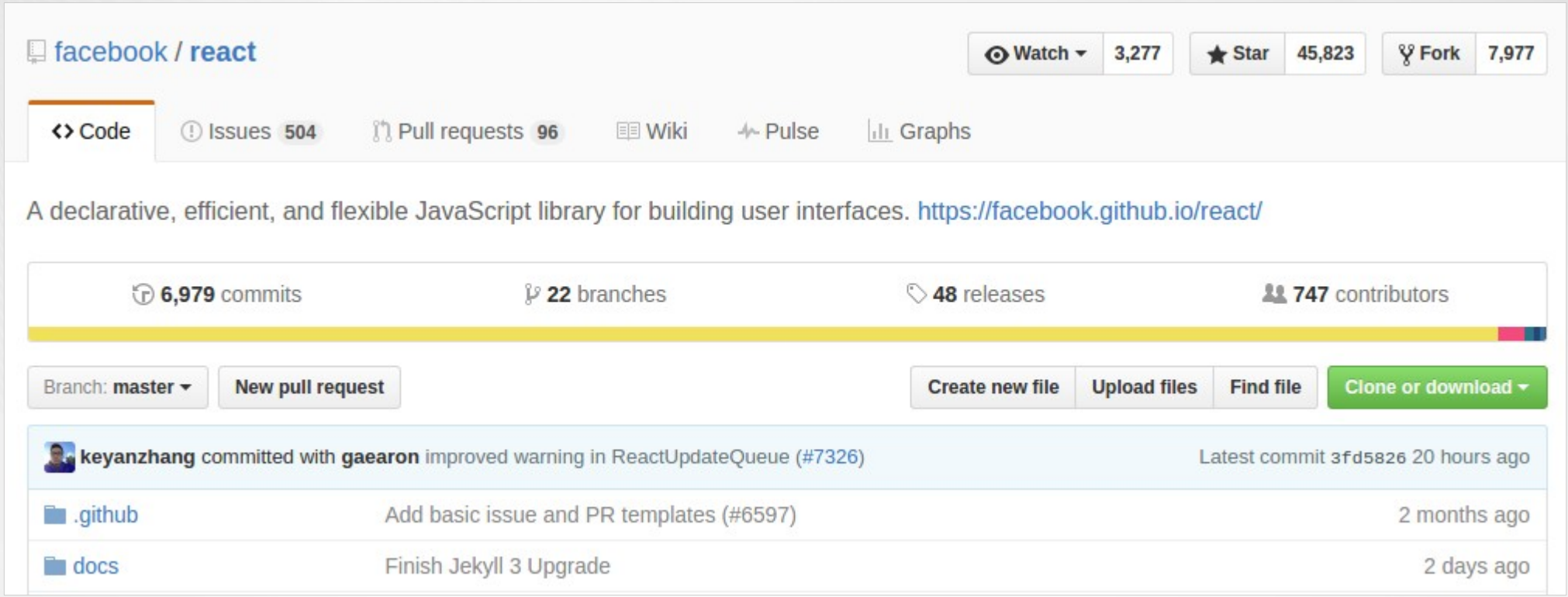
```
# tig
```

4) 궁금하거나 영향력있는 개발자의 commit 만 살펴보자

```
# tig --author=”Phil Webb”
```

CHAPTER 13

React 오픈소스 살펴보기



https://github.com/facebook/react

Git 으로 React 오픈소스 살펴보기

1) Linux Kernel 공식 repository URL 로 프로젝트 받기

```
# git clone https://github.com/facebook/react.git
```

2) shortlog 명령을 이용해서 통계를 내보자

```
# git shortlog
```

3) tig, gitk, log 명령등으로 commit 들을 살펴보자

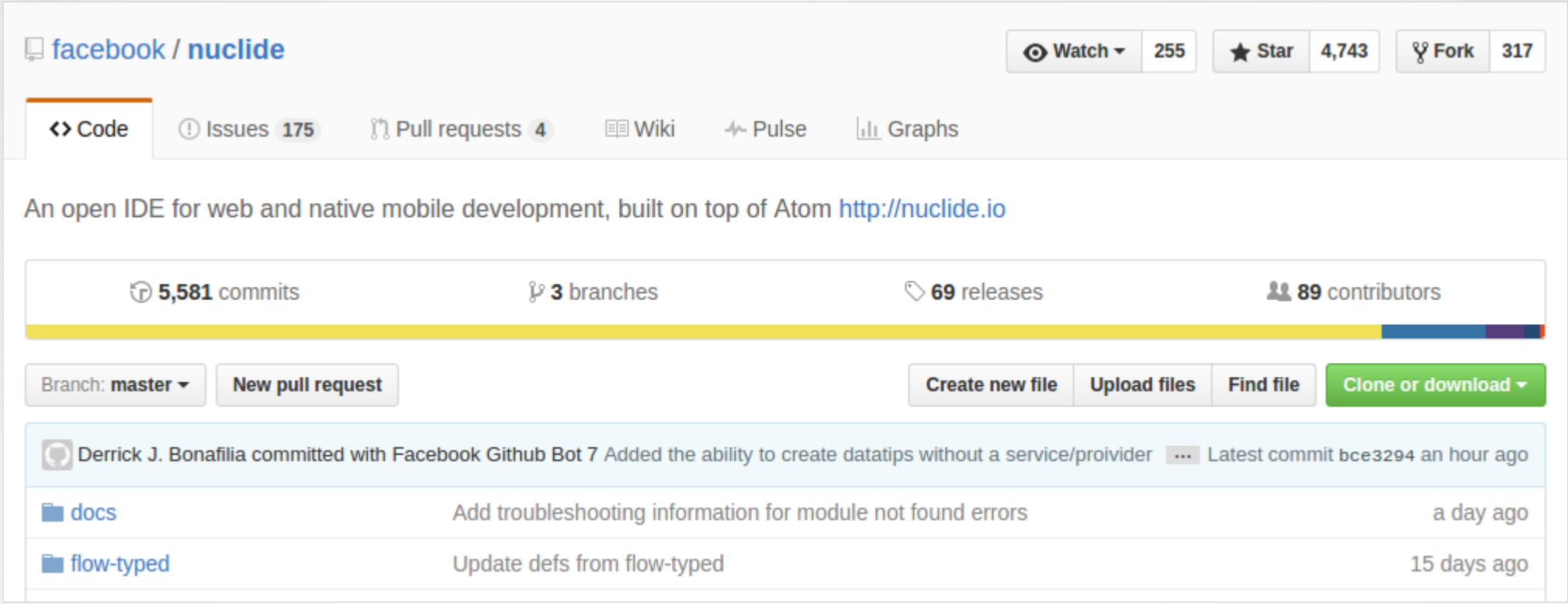
```
# tig
```

4) 궁금하거나 영향력있는 개발자의 commit 만 살펴보자

```
# tig --author=”Ben Alpert”
```

CHAPTER 13

Nuclide 오픈소스 살펴보기



https://github.com/facebook/nuclide

Git 으로 Nuclide 오픈소스 살펴보기

1) Linux Kernel 공식 repository URL 로 프로젝트 받기

```
# git clone https://github.com/facebook/nuclide.git
```

2) shortlog 명령을 이용해서 통계를 내보자

```
# git shortlog
```

3) tig, gitk, log 명령등으로 commit 들을 살펴보자

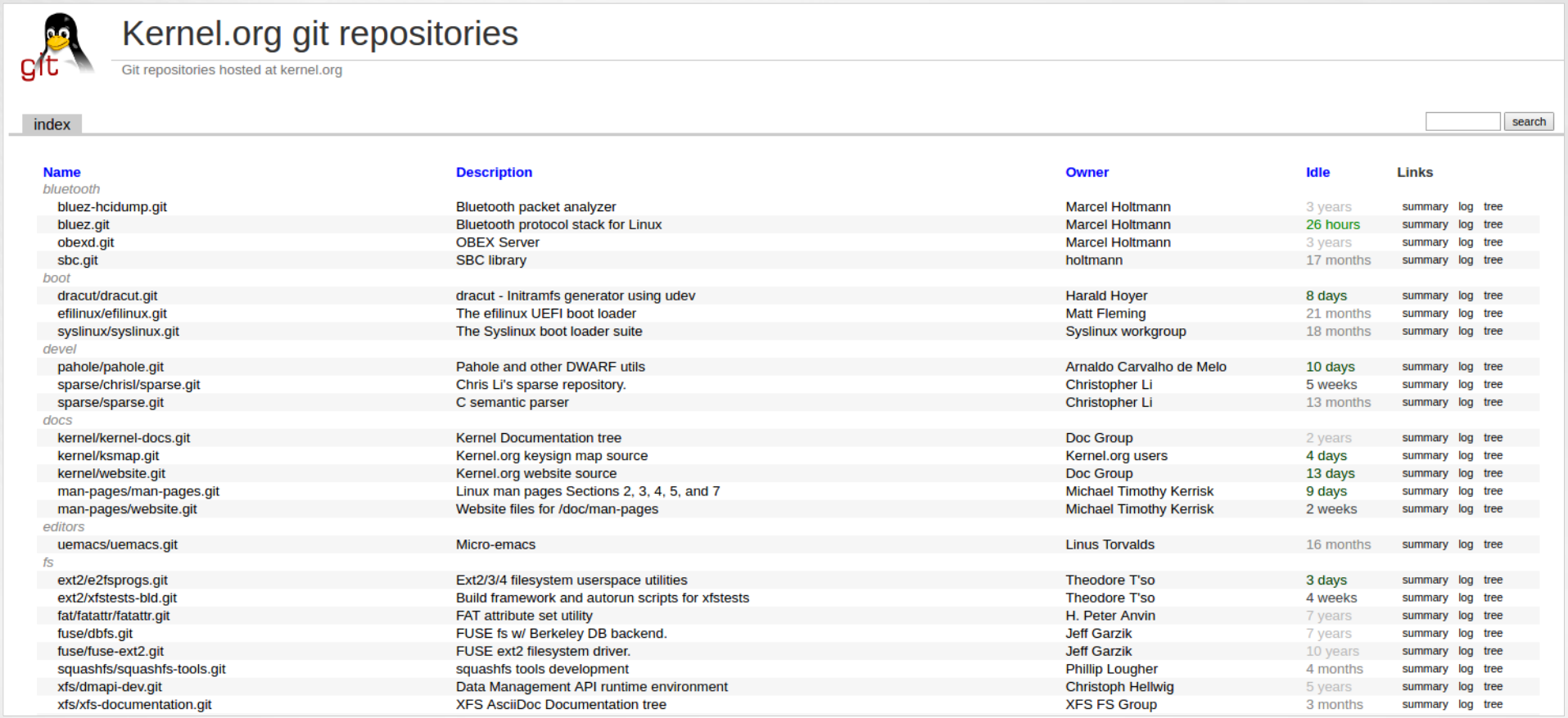
```
# tig
```

4) 궁금하거나 영향력있는 개발자의 commit 만 살펴보자

```
# tig --author=”Chen Shen”
```

Chapter 14. Linux Kernel Contribution

해당되는 Linux Kernel 공식 repository



Kernel.org git repositories

Git repositories hosted at kernel.org

index

search

Name	Description	Owner	Idle	Links
bluetooth				
bluez-hcidump.git	Bluetooth packet analyzer	Marcel Holtmann	3 years	summary log tree
bluez.git	Bluetooth protocol stack for Linux	Marcel Holtmann	26 hours	summary log tree
obexd.git	OBEX Server	Marcel Holtmann	3 years	summary log tree
sbc.git	SBC library	holtmann	17 months	summary log tree
boot				
dracut/dracut.git	dracut - Initramfs generator using udev	Harald Hoyer	8 days	summary log tree
efilinux/efilinux.git	The efilinux UEFI boot loader	Matt Fleming	21 months	summary log tree
syslinux/syslinux.git	The Syslinux boot loader suite	Syslinux workgroup	18 months	summary log tree
devel				
pahole/pahole.git	Pahole and other DWARF utils	Arnaldo Carvalho de Melo	10 days	summary log tree
sparse/chrisl/sparse.git	Chris Li's sparse repository.	Christopher Li	5 weeks	summary log tree
sparse/sparse.git	C semantic parser	Christopher Li	13 months	summary log tree
docs				
kernel/kernel-docs.git	Kernel Documentation tree	Doc Group	2 years	summary log tree
kernel/ksmap.git	Kernel.org keysign map source	Kernel.org users	4 days	summary log tree
kernel/website.git	Kernel.org website source	Doc Group	13 days	summary log tree
man-pages/man-pages.git	Linux man pages Sections 2, 3, 4, 5, and 7	Michael Timothy Kerrisk	9 days	summary log tree
man-pages/website.git	Website files for /doc/man-pages	Michael Timothy Kerrisk	2 weeks	summary log tree
editors				
uemacs/uemacs.git	Micro-emacs	Linus Torvalds	16 months	summary log tree
fs				
ext2/e2fsprogs.git	Ext2/3/4 filesystem userspace utilities	Theodore T'so	3 days	summary log tree
ext2/xfstests-bld.git	Build framework and autorun scripts for xfstests	Theodore T'so	4 weeks	summary log tree
fat/fatattr/fatattr.git	FAT attribute set utility	H. Peter Anvin	7 years	summary log tree
fuse/dbfs.git	FUSE fs w/ Berkeley DB backend.	Jeff Garzik	7 years	summary log tree
fuse/fuse-ext2.git	FUSE ext2 filesystem driver.	Jeff Garzik	10 years	summary log tree
squashfs/squashfs-tools.git	squashfs tools development	Phillip Lougher	4 months	summary log tree
xfs/dmapi-dev.git	Data Management API runtime environment	Christoph Hellwig	5 years	summary log tree
xfs/xfs-documentation.git	XFS AsciiDoc Documentation tree	XFS FS Group	3 months	summary log tree

https://git.kernel.org/cgit

해당되는 Linux Kernel 공식 repository

1) Mainline 에서 최상위 공식 Repository 인 리누스토발즈의 linux 프로젝트를 clone 한다

```
# git clone git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git
```

2) 생성된 linux 폴더에서 MAINTAINERS 파일을 열어서 작업할 프로젝트의 공식 repository 주소를 확인하자

```
      List of maintainers and how to submit kernel changes

Please try to follow the guidelines below.  This will make things
easier on the maintainers.  Not all of these guidelines matter for every
trivial patch so apply some common sense.
...(omitted)
PERFORMANCE EVENTS SUBSYSTEM
M:      Peter Zijlstra <peterz@infradead.org>
M:      Ingo Molnar <mingo@redhat.com>
M:      Arnaldo Carvalho de Melo <acme@kernel.org>
R:      Alexander Shishkin <alexander.shishkin@linux.intel.com>
L:      linux-kernel@vger.kernel.org
T:      git git://git.kernel.org/pub/scm/linux/kernel/git/tip/tip.git perf/core
S:      Supported
F:      kernel/events/*
...(omitted)
F:      tools/perf/
```

PATCH 작업 (contrubution) 거리 찾기

메인테이너가 쉽게받아줄거라 예상되는 PATCH/Pull-request 는 ?

1 위 - Refactoring (가장 부담 zero)

2 위 - Bugfix (명확한 버그라면)

3 위 - Documentation (첫 접근성은 좋으나 기술적인 description 이 문제)

4 위 - Minor features

그 이후 new feature 나 core 부분은 신뢰가 필요하다고봄 (다수의 commits)

PATCH 파일 만들고 Code Style 검사

1) 해당 프로젝트 repository 주소로 clone 하거나 upstream 으로 등록한다

```
# git clone git://git.kernel.org/pub/scm/linux/kernel/git/tip/tip.git
```

2) 소스파일 추가 및 수정에 대해서 서명을 포함해서 commit 만든다

```
# git add <file> && git commit -s
```

3) 완성된 commit 을 다시한번 확인해본다

```
# git show
```

4) 가장 최신 커밋 1 개를 patch 파일로 만들어낸다

```
# git format-patch -1
```

5) 완성된 .patch 파일의 Code Style 을 검사 한다

```
# $LINUX/scripts/checkpatch.el something.patch
```

PATCH 세트와 Cover letter

1) 여러개의 commit 을 PATCH 세트를 cover letter 와 함께 생성한다 (일곱번째 보내는것을 가정)

```
# git format-patch -7 --cover-letter --subject-prefix="PATCH v7"
```

2) 함께 생성된 cover letter 에 대해 작성한다

```
# emacs 0000-cover-letter.patch
```

Maintainer 찾기 및 PATCH mail 전송

1) MAINTAINERS 파일을 토대로 해당 프로젝트의 maintainer, reviewer 등을 찾는다

```
# emacs $LINUX/MAINTAINERS
```

2) PATCH 세트 혹은 파일을 메인테이너등에게 메일로 전송한다

```
# git send-email --confirm=never \
--to "Arnaldo Carvalho de Melo <acme@kernel.org>" \
--cc "linux-kernel@vger.kernel.org" \
--cc "Jiri Olsa <jolsa@kernel.org>" \
--cc "Namhyung Kim <namhyung@kernel.org>" \
--cc "Ingo Molnar <mingo@kernel.org>" \
--cc "Peter Zijlstra <peterz@infradead.org>" \
--cc "Alexander Shishkin <alexander.shishkin@linux.intel.com>" \
--cc "Masami Hiramatsu <mhiramat@kernel.org>" \
--cc "Wang Nan <wangnan0@huawei.com>" *.patch
```

PATCH mail 개발 방식 소개

이 변화분 (commits) 을 통한 Review 와 Discussion 가능 (pull-request, PATCH mail)

Arnaldo Carvalho de Melo November 23, 2015 at 11:17 PM
To: Taeung Song Cc: Namhyung Kim, linux-kernel@vger.kernel.org, Ingo Molnar, Jiri Olsa
Re: [PATCH v11 RESEND 02/24] perf tools: Add perf-config document

Em Sun, Nov 22, 2015 at 07:11:56PM +0900, Taeung Song escreveu:
Add perf-config document to describe the perf configuration
and a subcommand 'list'

Worked this time, applied, will continue processing the other patches soon.

- Arnaldo

See More

tip-bot for Taeung Song Today at 5:20 PM
To: linux-tip-commits@vger.kernel.org Cc: Ingo Molnar, and 7 more...
[tip:perf/core] perf config: Add initial man page

```
Commit-ID:  7d6852432acb3b09fc3ec45dd65421d34eebe3b5
Gitweb:     http://git.kernel.org/tip/7d6852432acb3b09fc3ec45dd65421d34eebe3b5
Author:     Taeung Song <treeze.taeung@gmail.com>
AuthorDate: Sun, 22 Nov 2015 19:11:56 +0900
Committer:  Arnaldo Carvalho de Melo <acme@redhat.com>
CommitDate: Mon, 23 Nov 2015 18:31:25 -0300

perf config: Add initial man page

Add perf-config document to describe the perf configuration and a
'list' subcommand.

Signed-off-by: Taeung Song <treeze.taeung@gmail.com>
Cc: Jiri Olsa <jolsa@redhat.com>
Cc: Namhyung Kim <namhyung@kernel.org>
Link: http://lkml.kernel.org/r/63AD9B57-7B8C-46F8-8F18-0FFEB9A6A1BC@gmail.com
Signed-off-by: Arnaldo Carvalho de Melo <acme@redhat.com>
---
tools/perf/Documentation/perf-config.txt | 103 +++++++++++++++++++++++++++++++++
1 file changed, 103 insertions(+)
```

tip-bot for Taeung Song Today at 5:20 PM
To: linux-tip-commits@vger.kernel.org
Cc: tglx@linutronix.de, Jiri Olsa, linux-kernel@vger.kernel.org, Ingo Molnar, hpa@zytor.com, Taeung Song, Namhyung Kim, acme@redhat.com hide
[tip:perf/core] perf tools: Add 'perf config' command

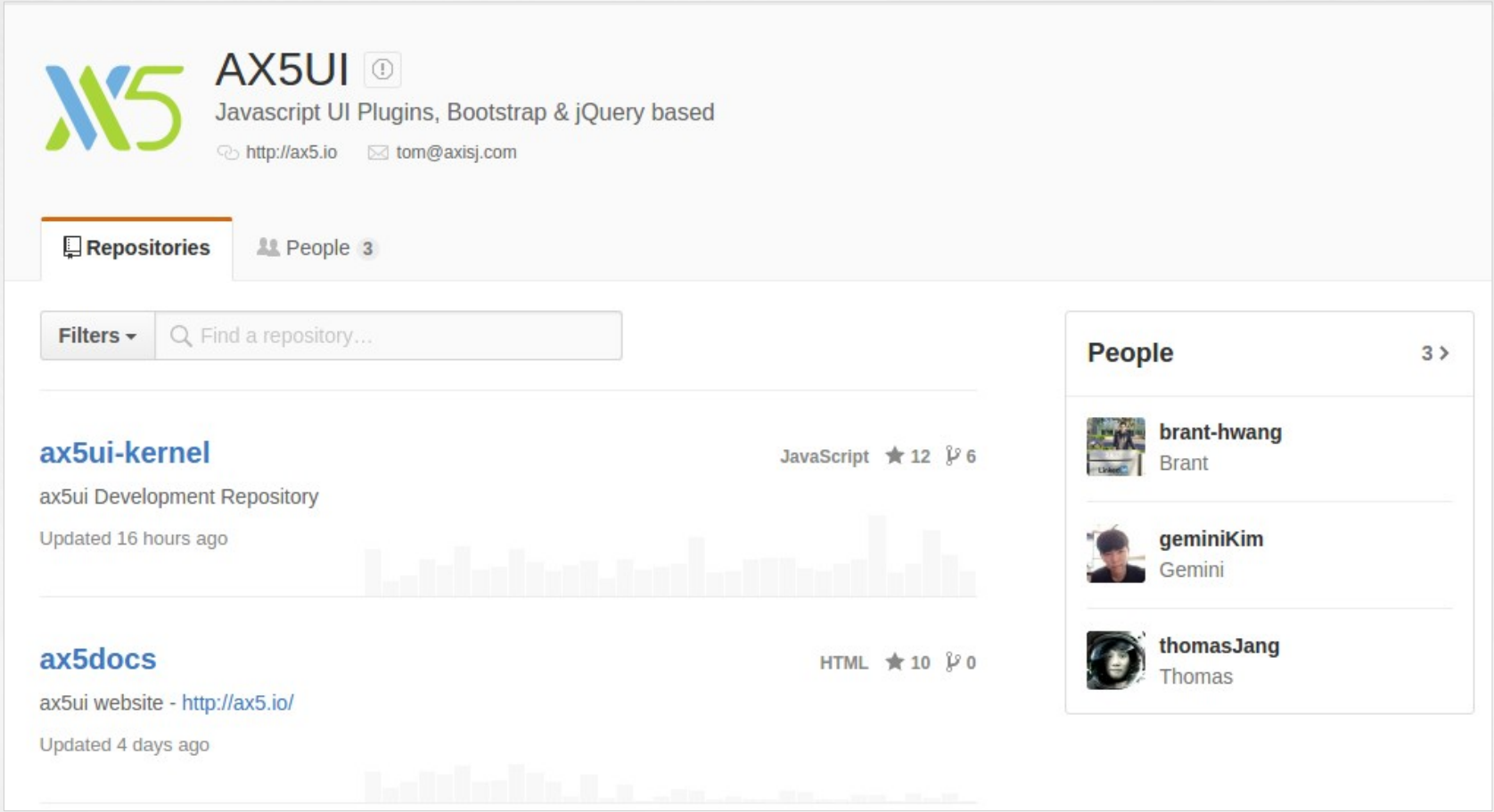
```
Commit-ID:  30862f2c5725c46afcfab5af710fdf5163bf0f81
Gitweb:     http://git.kernel.org/tip/30862f2c5725c46afcfab5af710fdf5163bf0f81
Author:     Taeung Song <treeze.taeung@gmail.com>
AuthorDate: Tue, 17 Nov 2015 22:53:21 +0900
Committer:  Arnaldo Carvalho de Melo <acme@redhat.com>
CommitDate: Mon, 23 Nov 2015 18:31:24 -0300

perf tools: Add 'perf config' command

The perf configuration file contains many variables to change various
aspects of each of its tools, including output, disk usage, etc.

But looking at the state of configuration is difficult and there's no
documentation about config variables except for the variables in
perfconfig.example exist.

So this patch adds a 'perf-config' command with a '--list' option.

  perf config [options]

  display current perf config variables.
  # perf config -l | --list

Signed-off-by: Taeung Song <treeze.taeung@gmail.com>
Acked-by: Namhyung Kim <namhyung@kernel.org>
Cc: Jiri Olsa <jolsa@redhat.com>
Link: http://lkml.kernel.org/r/1447768424-17327-1-git-send-email-treeze.taeung@gmail.com
Signed-off-by: Arnaldo Carvalho de Melo <acme@redhat.com>
---
tools/perf/Build           |  1 +
tools/perf/builtin-config.c | 66 +++++++++++++++++++++++++++++++++++++++++++++
tools/perf/builtin.h       |  1 +
tools/perf/command-list.txt |  1 +
tools/perf/perf.c          |  1 +
5 files changed, 70 insertions(+)
```


SK 주식회사 C&C

Chapter 15. GitHub 기반 오픈소스 Contribution

CHAPTER 15

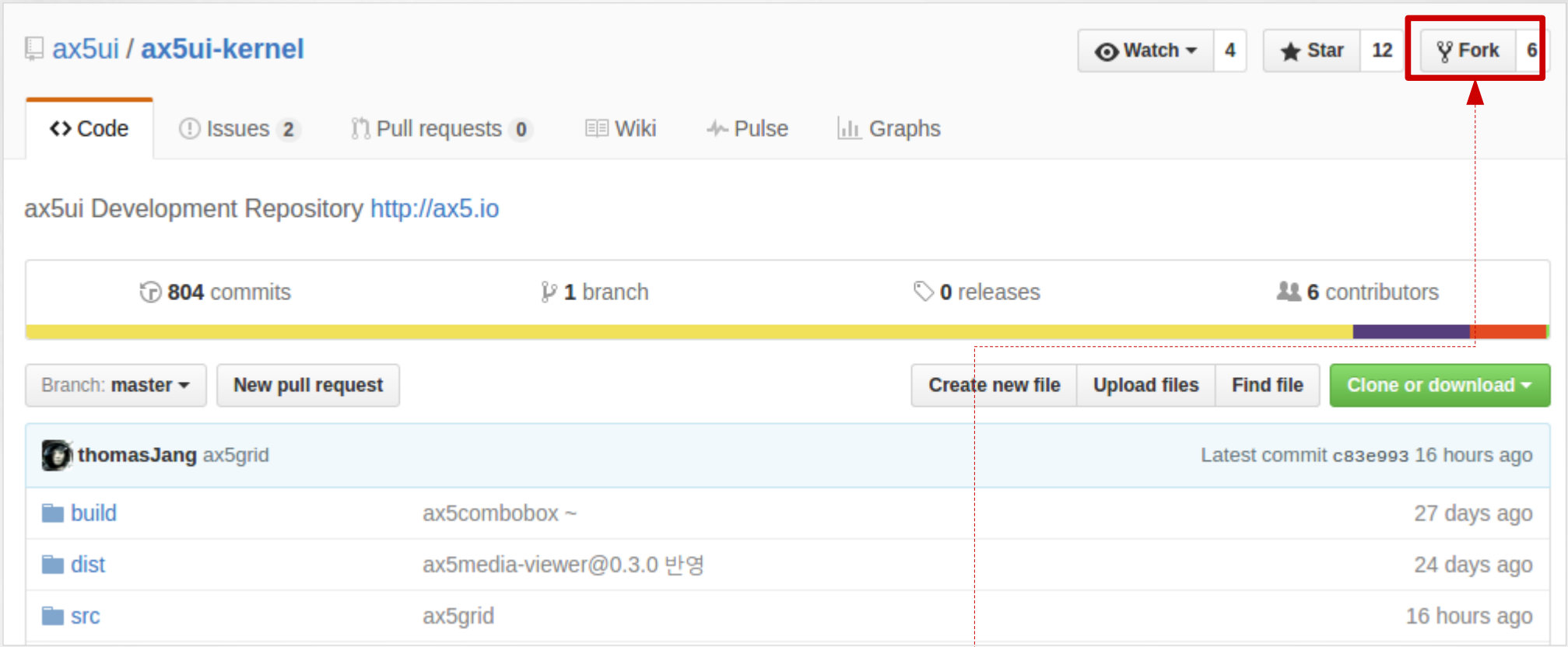
오픈소스 ax5ui



https://github.com/ax5ui

CHAPTER 15

ax5ui 프로젝트 Fork 하기



Fork 버튼을 눌러서 프로젝트가 생성되었는지 확인하자 (본인 Github)

CHAPTER 15

Pull-request 로 작업내용 제출

1) Fork 한 프로젝트를 내려받는다

```
# git clone https://github.com/<본인 ID>/ax5ui-kernel.git
```

2) develop 브랜치를 만든다

```
# cd ax5ui-kernel && git checkout -b develop
```

3) 소스파일 추가 및 수정에 대해서 서명을 포함해서 commit 만든다

```
# git add <file> && git commit -s
```

4) 완성된 commit 을 다시한번 확인해본다

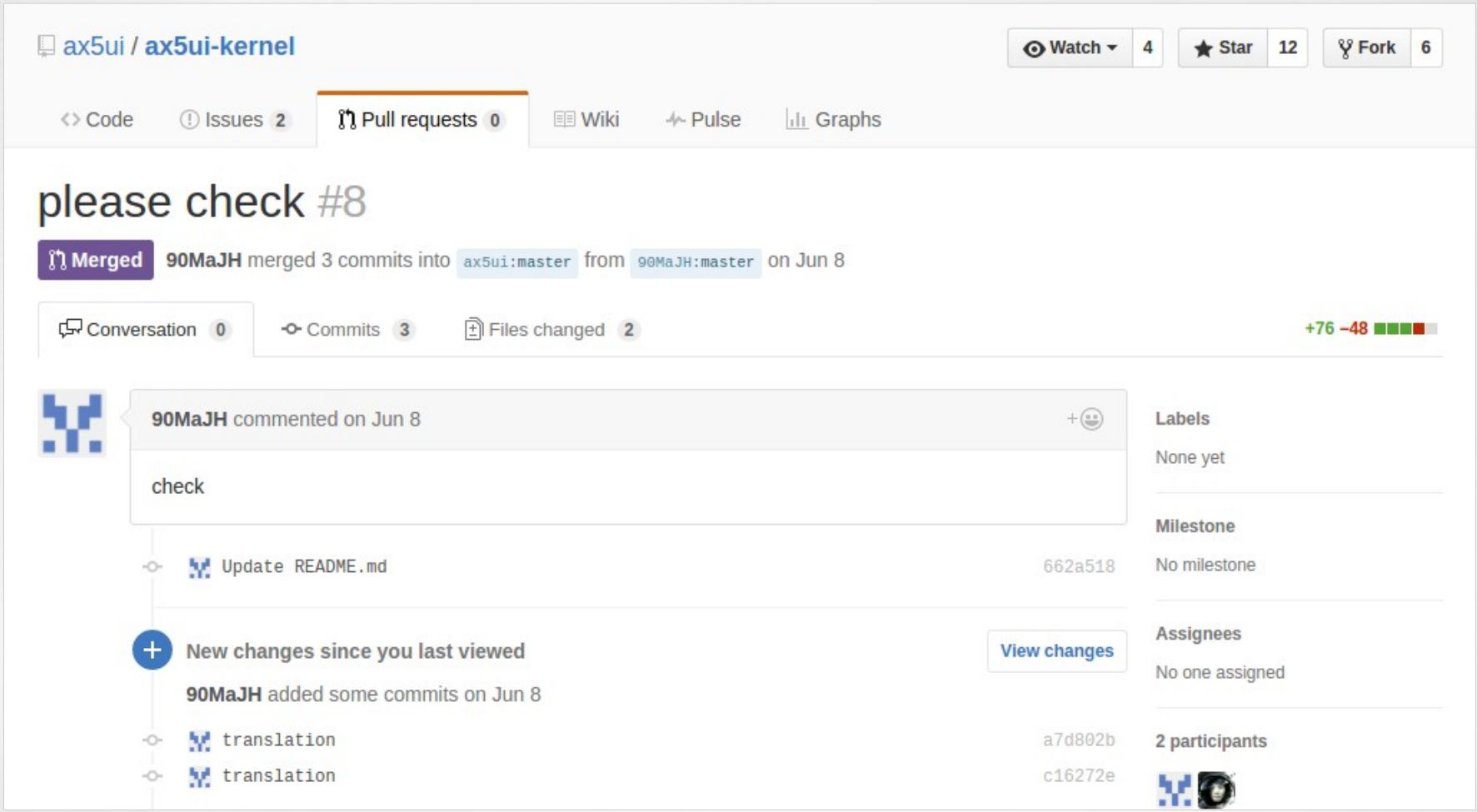
```
# git show
```

5) 본인 fork 한 프로젝트에 push 를 하고 pull-request 진행

```
# git push origin develop
```

CHAPTER 15

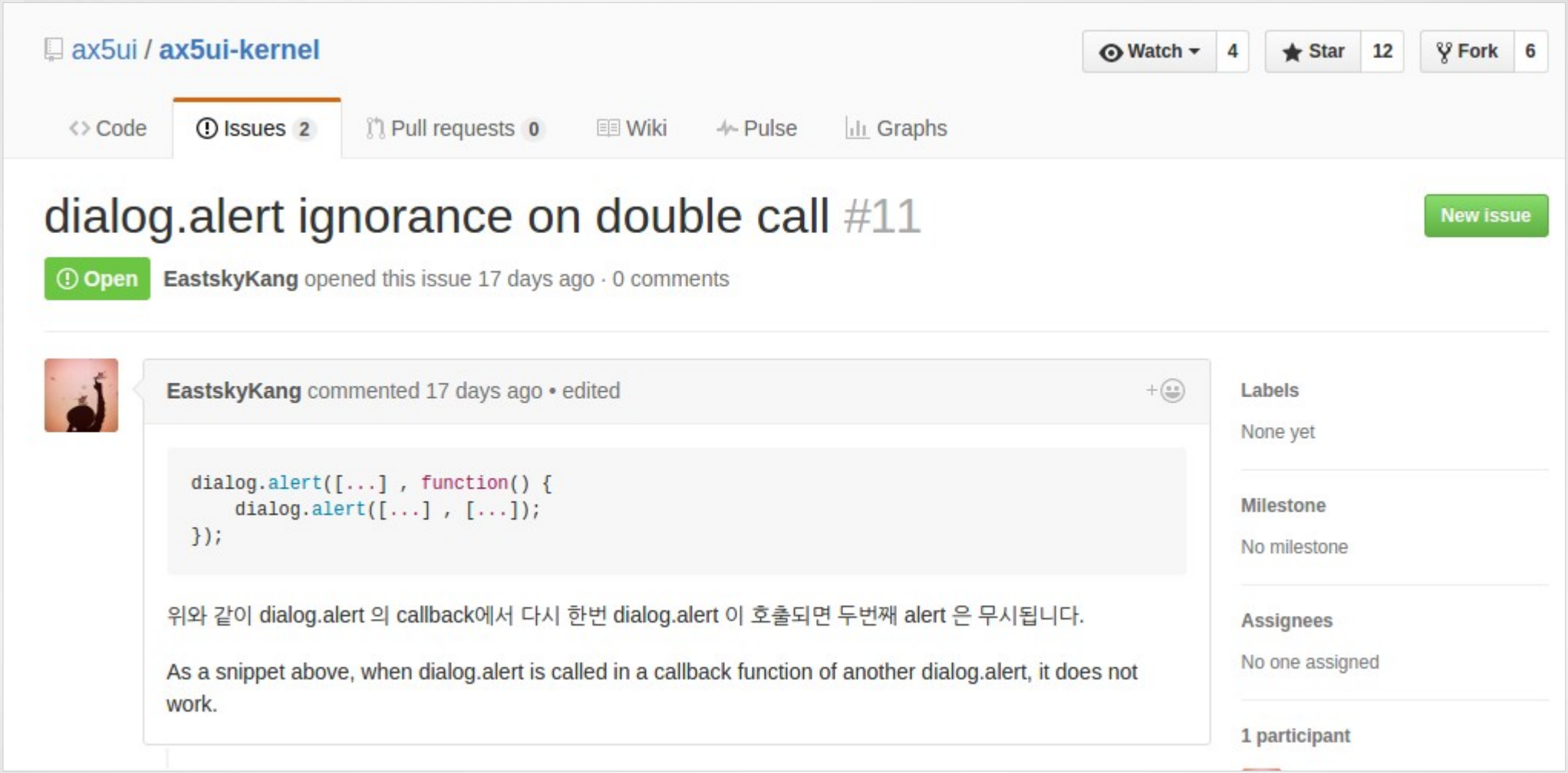
Pull-request 로 작업내용 예시



https://github.com/ax5ui/ax5ui-kernel/pull/8

CHAPTER 15

Issue tracking 기능 활용



https://github.com/ax5ui/ax5ui-kernel/issues/11

Q & A

Git / Github Training

SK 주식회사 C&C