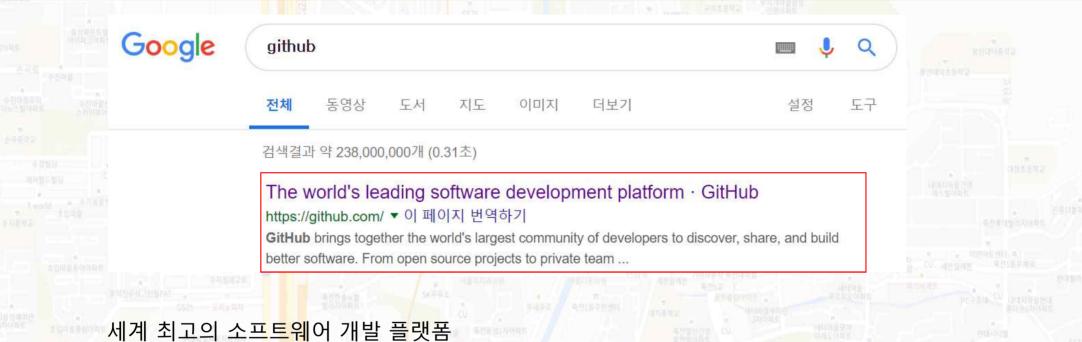


## 목차

- 1. Github란
- 2. Wiki
- 3. Issue, Milestone, Projects
- 4. Pull Request

## 1. Github 란?



GitHub는 세계에서 가장 큰 개발자 커뮤니티를 모아 더 나은 소프트웨어를 발견, 공유 및 구축합니다. 오픈 소스 프로젝트부터 개인 팀까지...







**Wiki** 

<u>III</u> Insights



#### 개발에 필요한 정보들을 문서로 정리

#### RestfulAPI

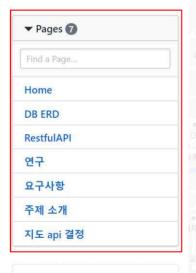
박은영 edited this page 22 hours ago · 5 revisions

#### Edit New Page

#### Ver1.0

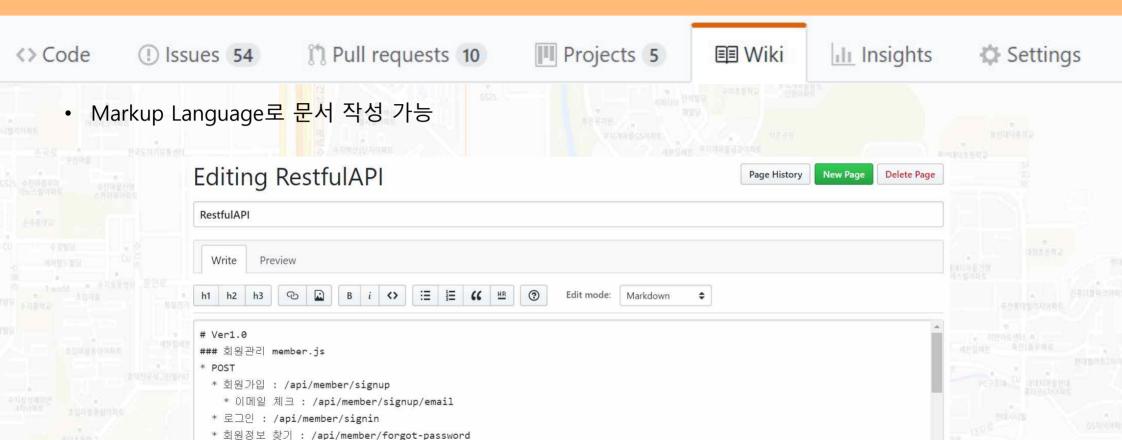
#### 회원관리 member.js

- POST
  - 회원가입:/api/member/signup
    - 이메일 체크 : /api/member/signup/email
  - 로그인 : /api/member/signin
  - 회원정보 찾기 : /api/member/forgot-password
  - 나의 친구 검색 : /api/member/search-friend
  - o 유저 검색 : /api/member/search-member
- DELETE
  - o 회원탈퇴 : /api/member
- PUT



+ Add a custom sidebar

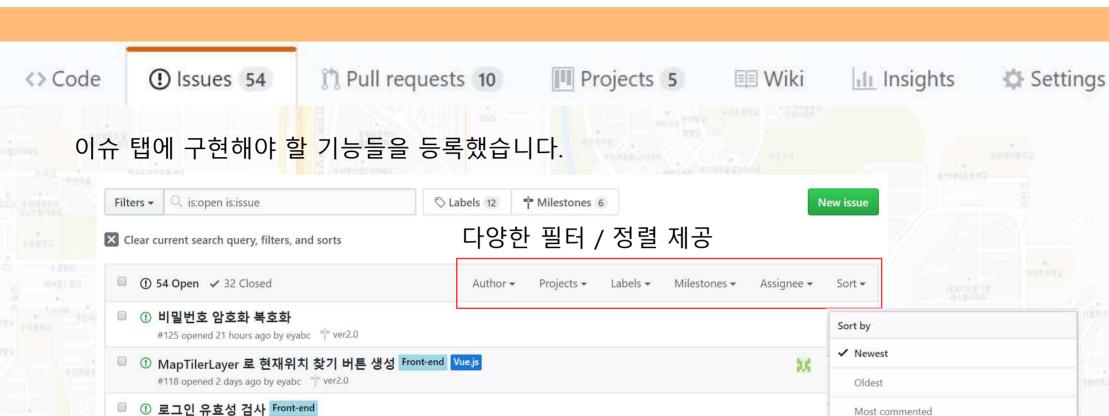
Clone this wiki locally



\* 나의 친구 검색 : /api/member/search-friend \* 유저 검색 : /api/member/search-member

\* DELETE

\* 회원탈퇴 : /api/member



#112 opened 5 days ago by eyabc 🔭 ver 1.0

#106 opened 6 days ago by eyabc \* ver2.0

#105 opened 6 days ago by eyabc 🐈 ver2.0

□ package-lock.json 충돌에 대한 해결 이슈 생각필요#103 opened 6 days ago by eyabc† 연구 Study

□ ① [메인페이지] 구글 지도 api 보안기능

□ ① 사용자 트래픽 추정

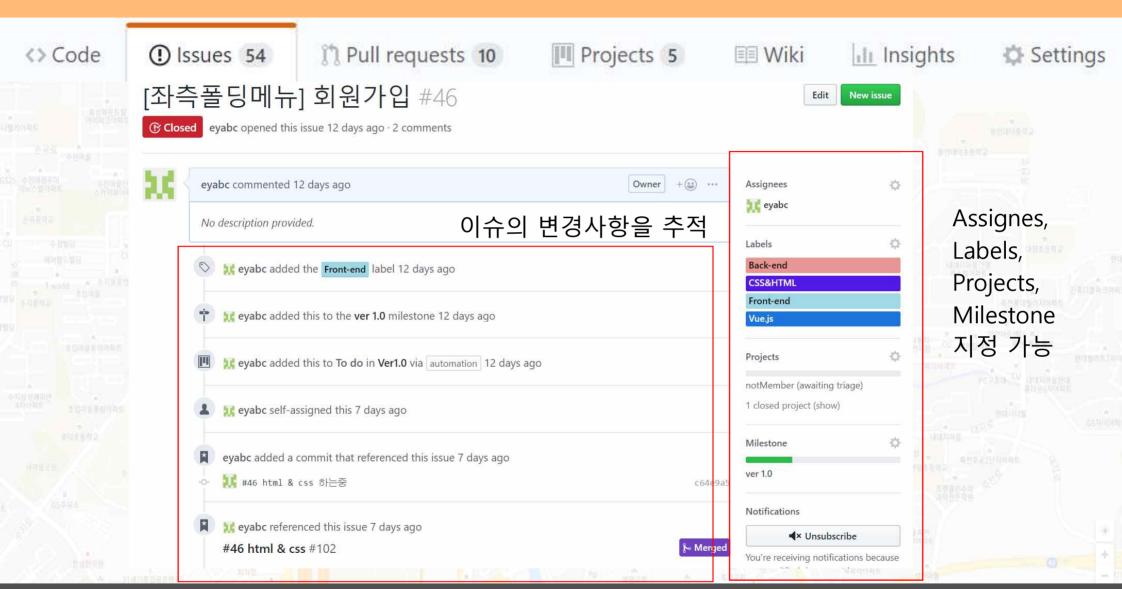
Least commented

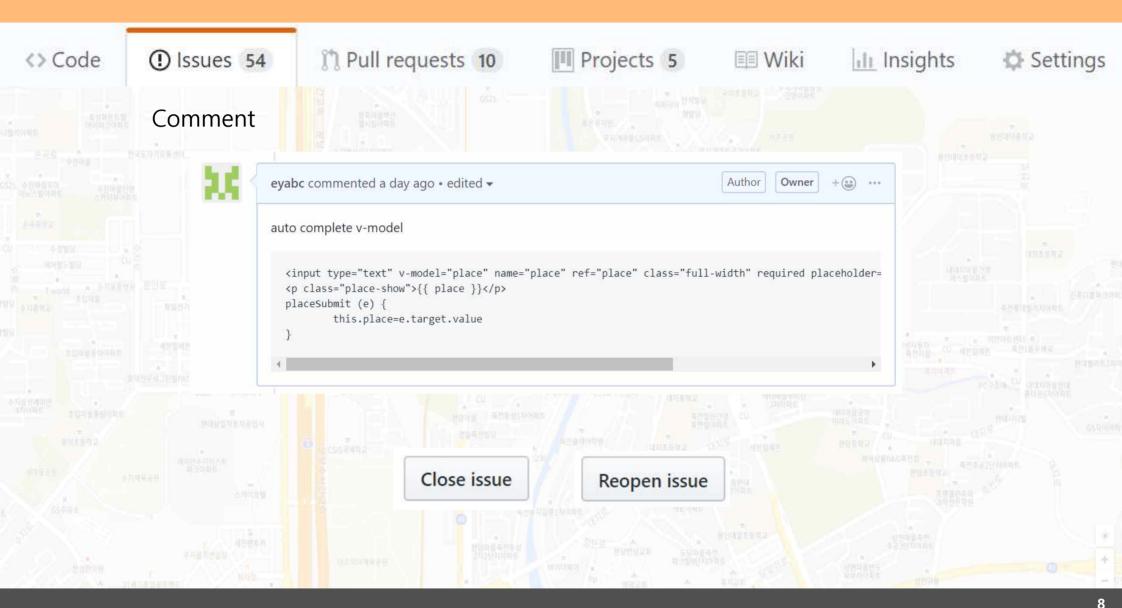
Recently updated

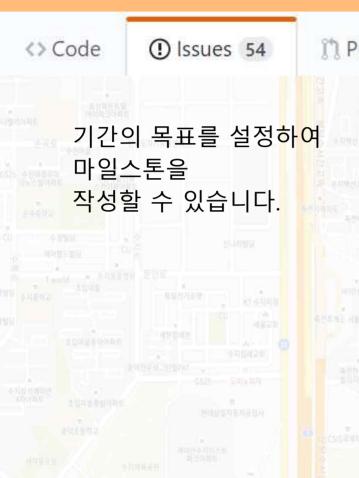
Most reactions

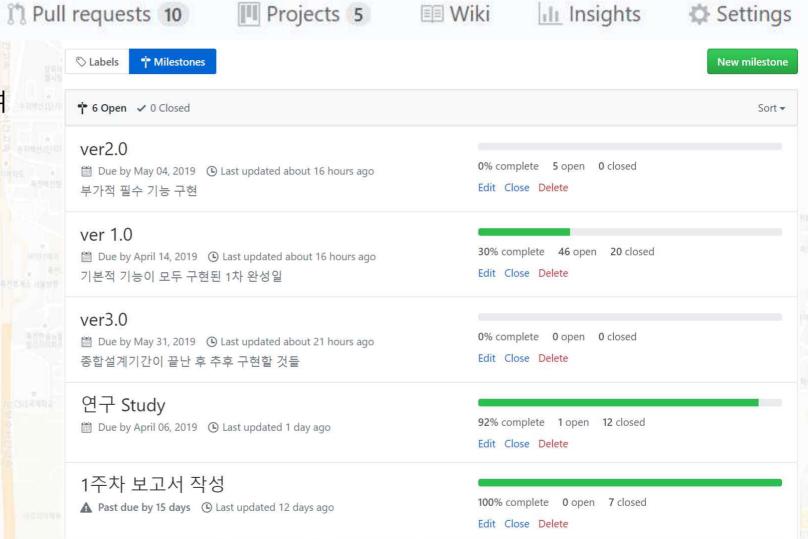
Least recently updated

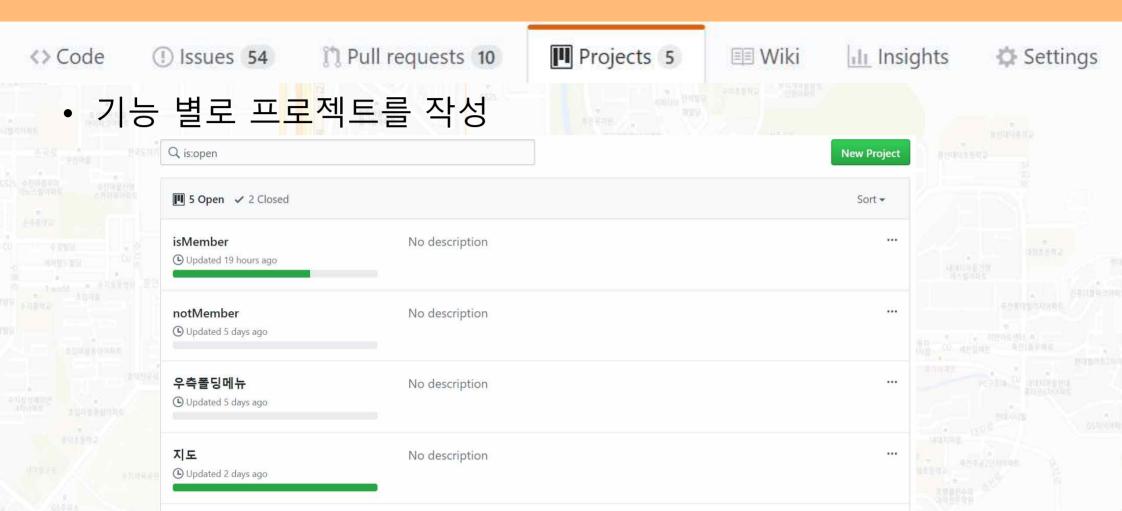
00











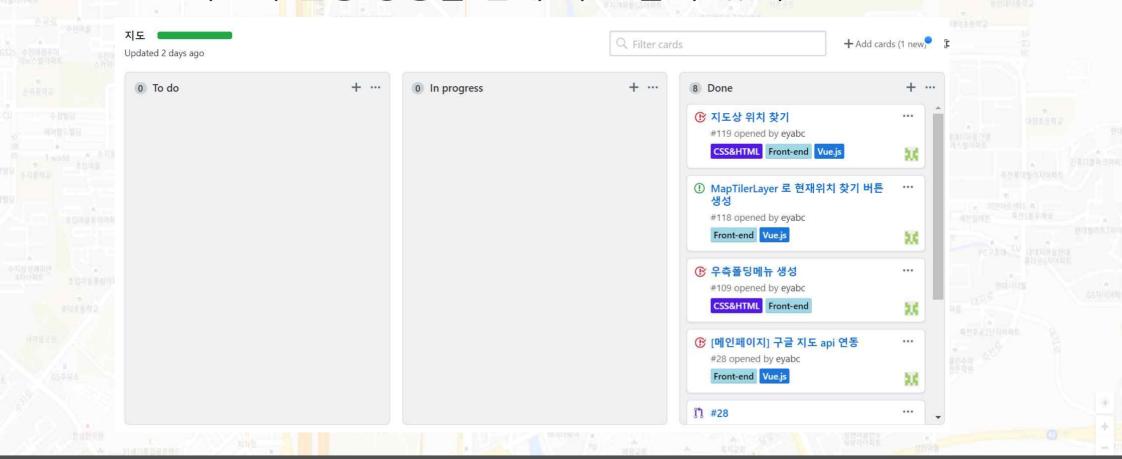
No description

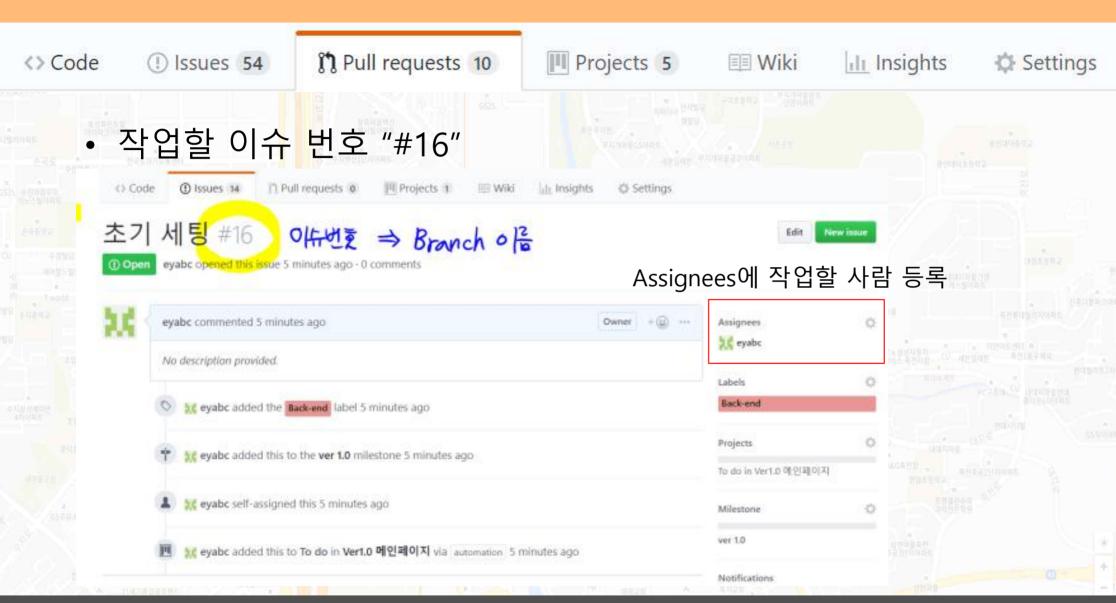
Ver.1.0 공통

(L) Updated 20 hours ago



## • 프로젝트의 진행 상황을 쉽게 확인 할 수 있다.





<> Code

! Issues 54

n Pull requests 10

Projects 5

Wiki Wiki

III Insights



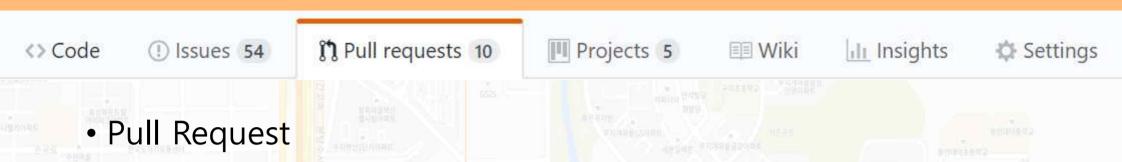
### • 프로젝트에서 브랜치를 만드는 목적

소프트웨어를 개발할 때에 개발자들은 동일한 소스코드를 함께 공유하고 다루게 됩니다. 동일한 소스코드 위에서 어떤 개발자는 버그를 수정하기도 하고 또 다른 개발자는 새로운 기능을 만들어 내기도 하죠. 이와 같이 여러 사람이 동일한 소스코드를 기반으로 서로 다른 작업을 할 때에는 각각 서로 다른 버전의 코드가 만들어 질 수 밖에 없습니다.

이럴 때, 여러 개발자들이 동시에 다양한 작업을 할 수 있게 만들어 주는 기능이 바로 '브랜치 (Branch)' 입니다. 각자 독립적인 작업 영역(저장소) 안에서 마음대로 소스코드를 변경할 수 있지요. 이렇게 분리된 작업 영역에서 변경된 내용은 나중에 원래의 버전과 비교해서 하나의 새로운 버전으로 만들어 낼 수 있습니다.

#### 📕 브랜치(branch)란?

<u>브랜치란 독립적으로 어떤 작업을 진행하기 위한 개념</u>입니다. 필요에 의해 만들어지는 각각의 브랜치는 다른 브랜치의 영향을 받지 않기 때문에, 여러 작업을 동시에 진행할 수 있습니다.



Branch 에서 작업을 한 후 원래의 흐름(여기서는 master)에 작업 내용을 반영하기 위해 필요함. <> Code

(!) Issues 54

n Pull requests 10

Projects 5

**Wiki** 

III Insights

Settings \$

• 이슈번호와 동일하게 branch를 생성 합니다.

```
User@DESKTOP-8GFD4IJ MINGW64 /e/Programming/capstone-2019 (master)

User@DESKTOP-8GFD4IJ MINGW64 /e/Programming/capstone-2019 (master)

S git branch
#16

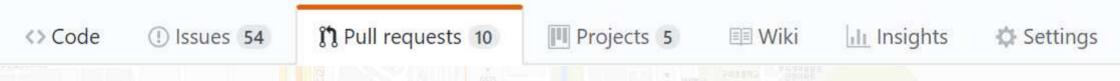
* master

User@DESKTOP-8GFD4IJ MINGW64 /e/Programming/capstone-2019 (master)

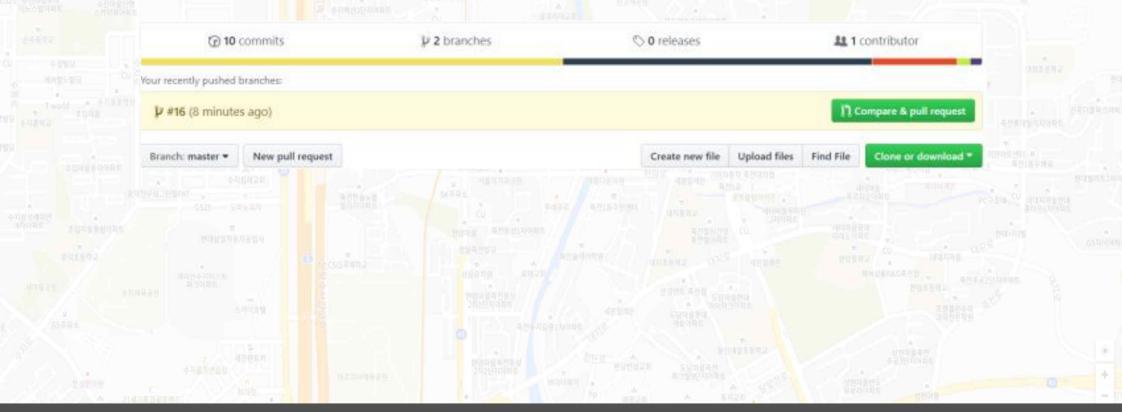
S git checkout "#16"

Switched to branch '#16'
```

- 작업을 완료 <mark>한</mark> 뒤 이슈번호로 commit을 작성하고 push를 해준다.
  - git commit -m "#16 front-end 초기세팅 완료"
  - git push origin "#16"



• 본인 계정의 github 저장소에 Compare & pull request가 활성화 되어 있습니다.









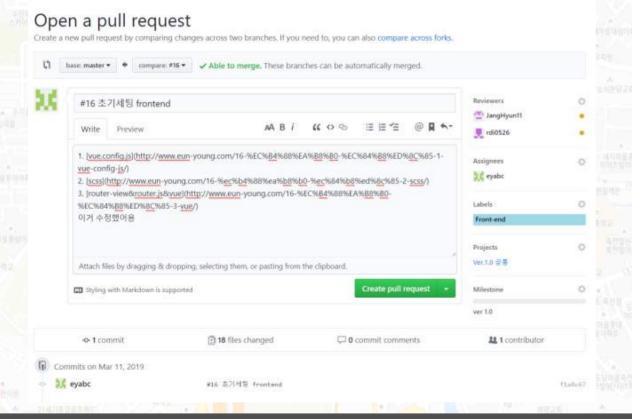




ll Insights



## pull Request를 작성합니다.

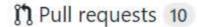


Branch에서 Master로 Merge하기 위해 필요한 과정입니다.

파일이 어떻게 바뀌었는지도 조회가능



(!) Issues 54



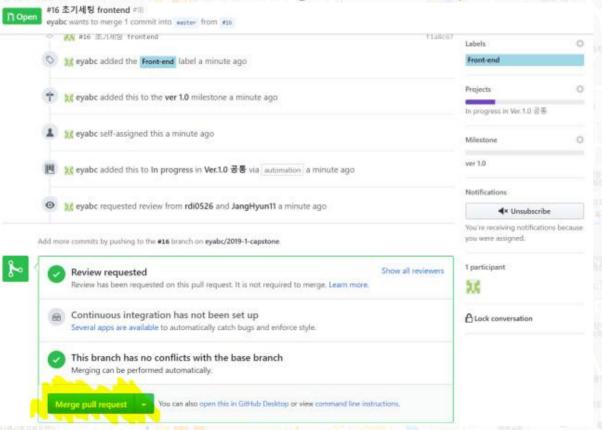
Projects 5

Wiki

III Insights



## Code Review & Merge Pull Request



Pull Request 을 받은 Asignees는 코드 변경 내역을 확인하고 Merge를 합니다.

conflict 존재 : Merge를 할 수 없어, 실수를 방지할 수 있습니다.

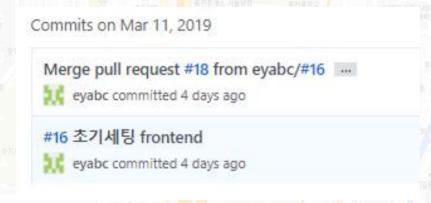
github 에서 conflict를 수정할 수도 있습니다.



- (!) Issues 54
- 🎵 Pull requests 10
- Projects 5
- **Wiki**
- III Insights

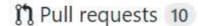


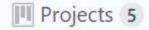
- Merge 의 세 가지 옵션
- 1. Merge pull Request
  - master의 commit에 branch에서 commit 한 것이 모두 올라갑니다.
  - 사진에서 실<mark>제로 commit</mark>을 1번 하였고. Merge pull Request를 했다는 사실도 commit에 올라갔습니다. .













II Insights

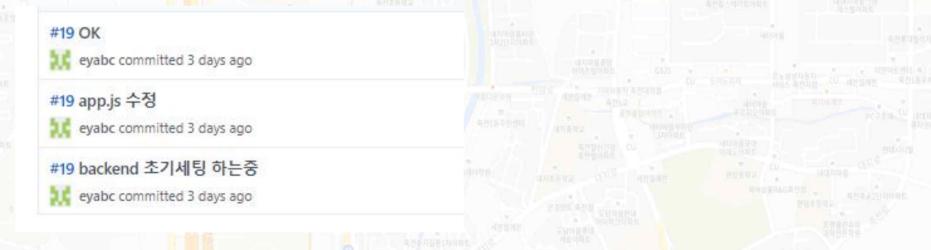


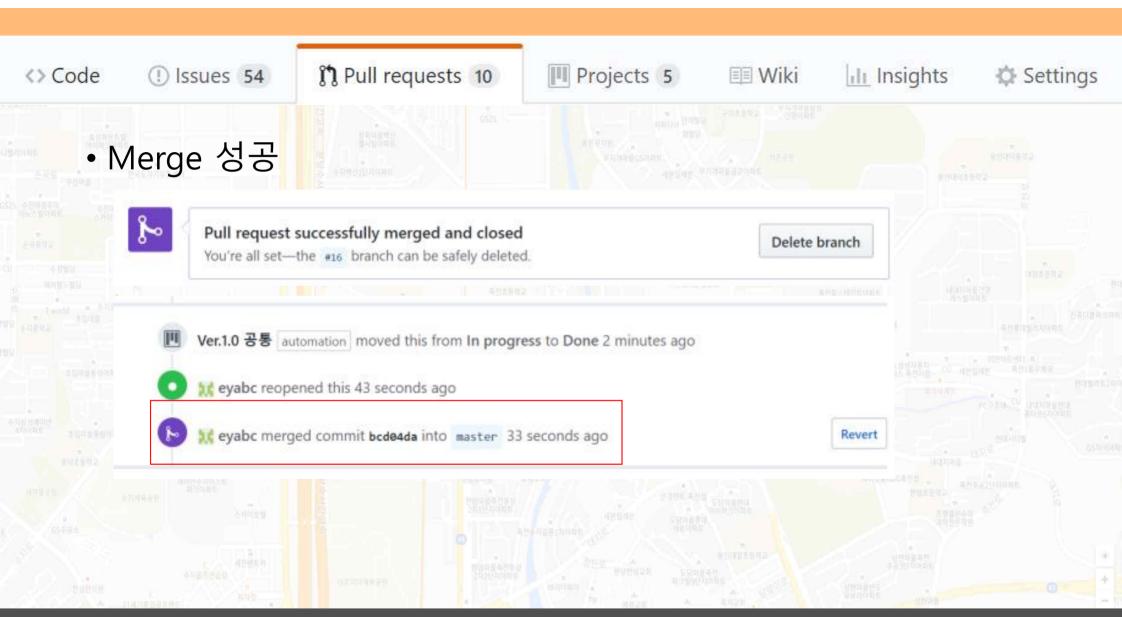
#### 2. Squash and merge

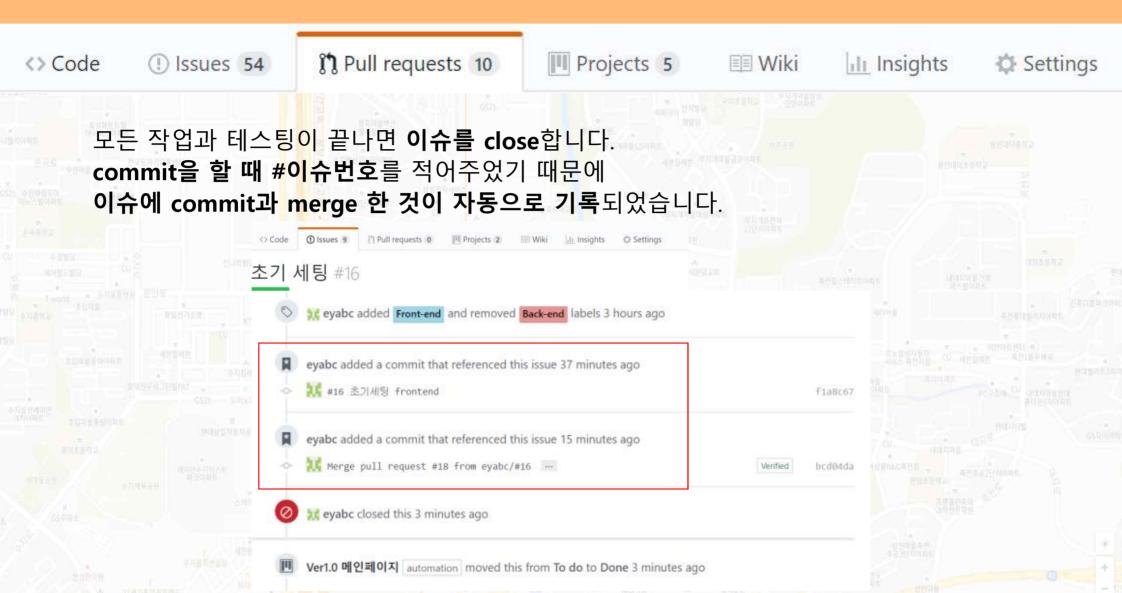
- master의 commit에 branch에서 최근에 commit한 내용 하나가 올라갑니다.
- merge를 하<mark>기</mark> 전, branch의 이전 commit을 부가적으로 기록할지도 정할 수 있습니다.
- ... 을 눌러보면 branch에서 커밋한 이전 내역이 기록되어 있습니다.



3. Rebase and merge master의 commit에 branch에서 커밋한 내용이 모두 올라가지만 merge했다는 commit 은 추가되지 않는다.







# 참고 사이트



# 누구나 쉽게 이해할 수 있는 Git 입문 버전 관리를 완벽하게 이용해보자

https://backlog.com/git-tutorial/kr/

Pull Request Merge 옵션

https://meetup.toast.com/posts/122

# 다음 발표 주제

