2차 보고서

협업 방법 소개

**3조**

**주제** 지도 기반 채팅 커뮤니티 시스템

**팀명** 하이파이브

**팀원** 박은영(32151839), 류다인(32151495), 김장현(32160974)

사용중인 협업 방법

1. 구글 Docs
2. 깃 허브
3. 구글 Docs

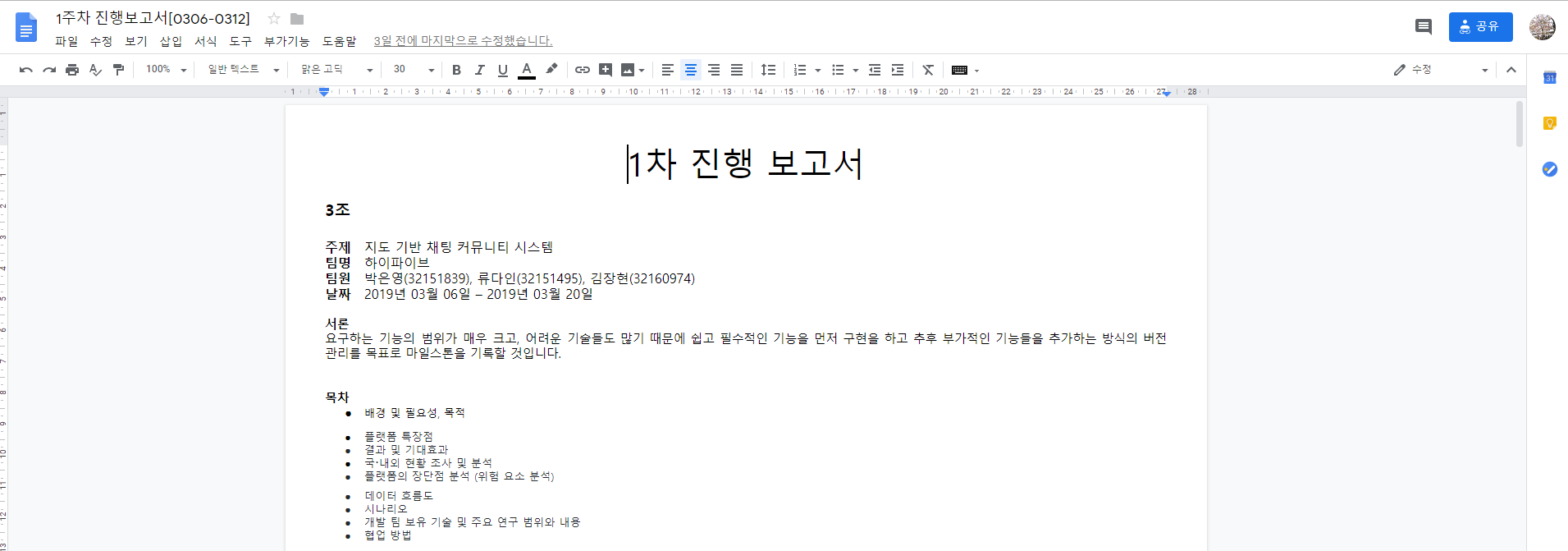
링크: <https://docs.google.com/document/>

* 소개

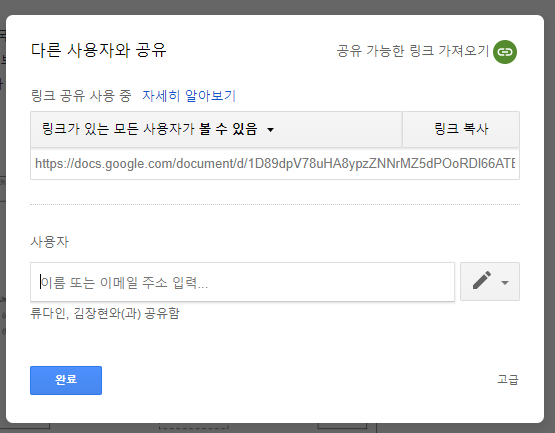
구글의 문서도구

[웹](https://ko.wikipedia.org/wiki/%EC%9B%B9) 기반 서비스로, [워드 프로세서](https://ko.wikipedia.org/wiki/%EC%9B%8C%EB%93%9C_%ED%94%84%EB%A1%9C%EC%84%B8%EC%84%9C), [스프레드시트](https://ko.wikipedia.org/wiki/%EC%8A%A4%ED%94%84%EB%A0%88%EB%93%9C%EC%8B%9C%ED%8A%B8), [프레젠테이션](https://ko.wikipedia.org/wiki/%ED%94%84%EB%A0%88%EC%A0%A0%ED%85%8C%EC%9D%B4%EC%85%98_%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4), 그림 등 기능 제공

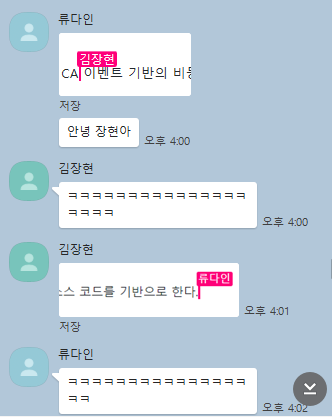
* 활용 방법 소개
  1. 1차 보고서 작성 할 때 활용을 시작하였습니다

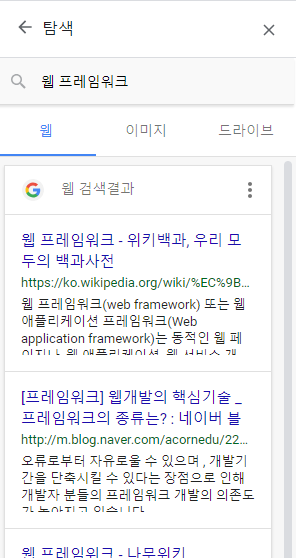
.

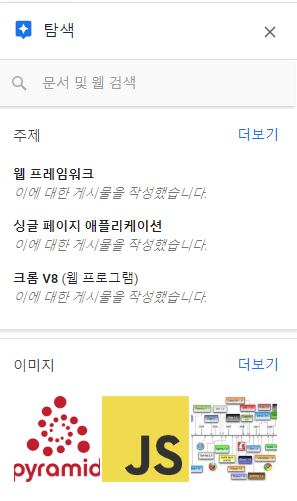
* 1. 팀원들과 문서를 공유하여 하나의 문서를 동시에 편집할 수 있습니다.



* 1. 문서 작업중인 사람을 실시간으로 확인할 수 있습니다.



* 1. 문서에 나오는 단어를 분석하여 구글이 정보를 추천해 주는 기능도 있습니다.



* 1. 문서의 변경사항이 추적이 가능합니다.

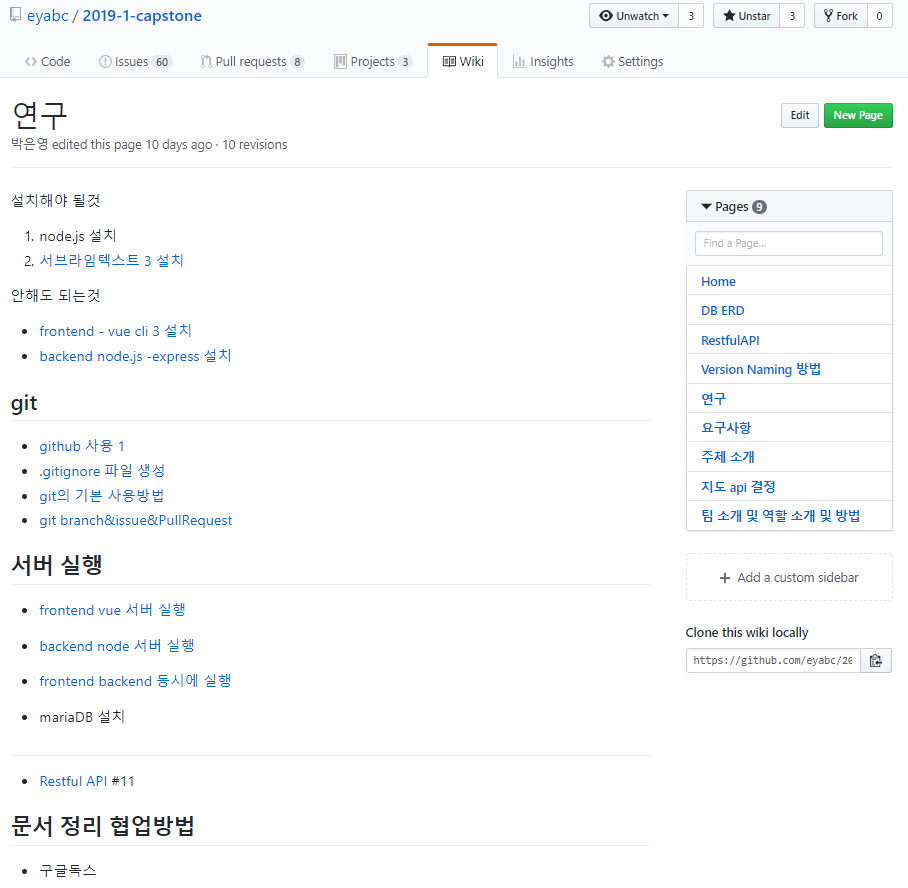


1. 깃 허브

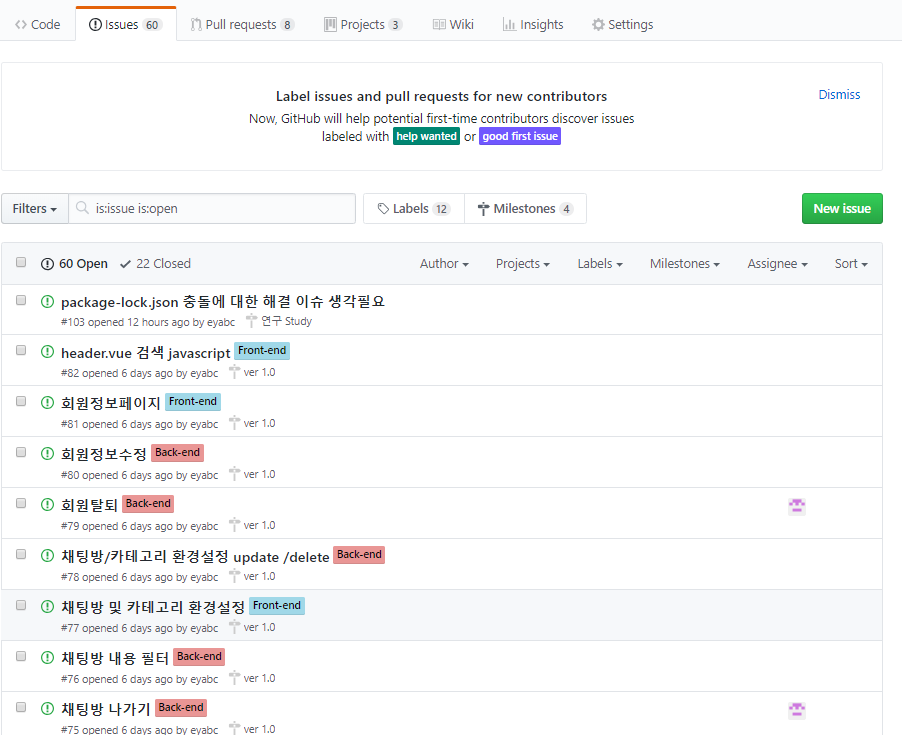
* 저장소 링크

<https://github.com/eyabc/2019-1-capstone>

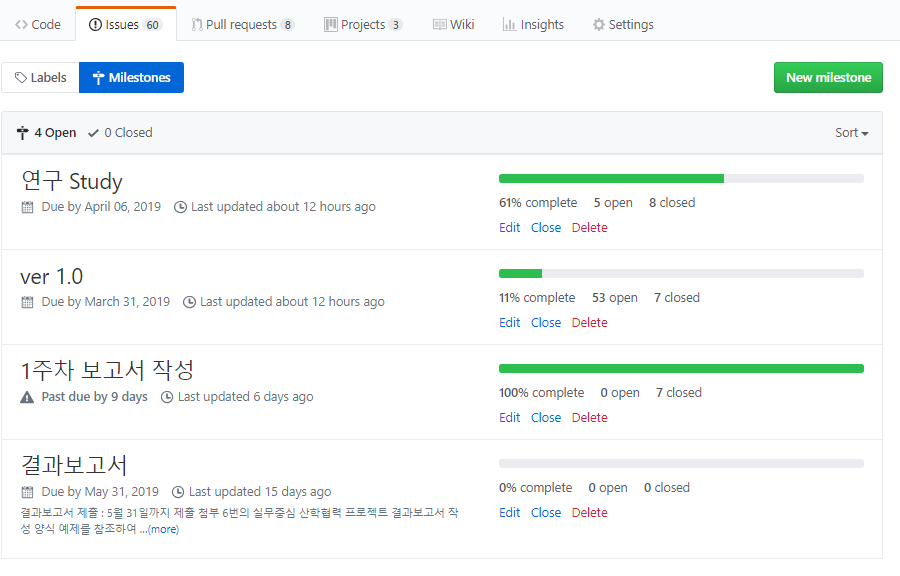
* WiKi
  1. 연구자료, RestfulAPI, 요구사항, 등 문서를 정리

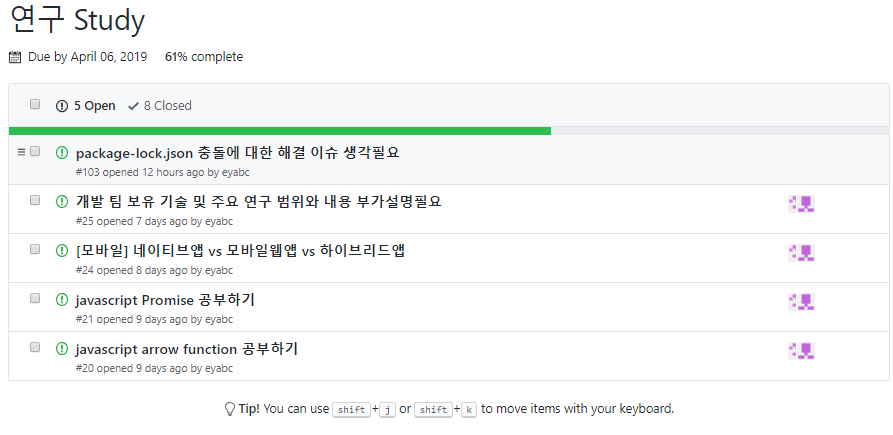
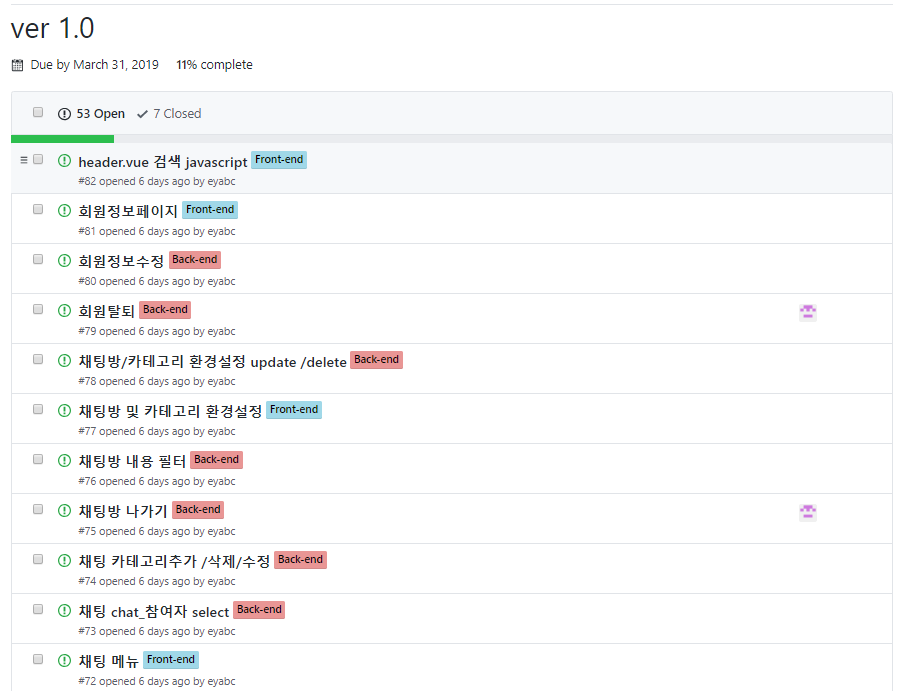


* 1. RestfulAPI, 프로토타입, 요구사항을 참조하여 이슈 탭에 1버전의 구현해야 할 기능들을 등록했습니다.

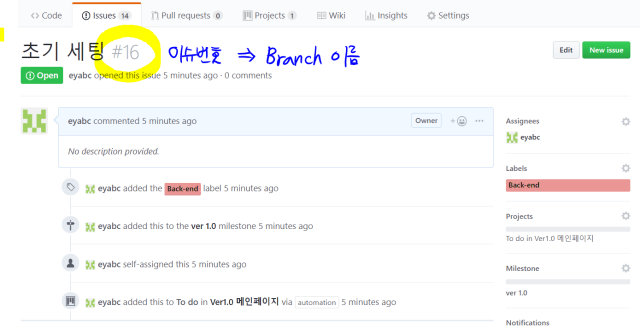


* + - 이슈 작성자, 프로젝트, 라벨, 마일스톤, 책임자 등으로 정렬 필터 가능합니다
  1. 기간의 목표를 설정하여 이슈탭의 마일스톤에 등록했습니다.

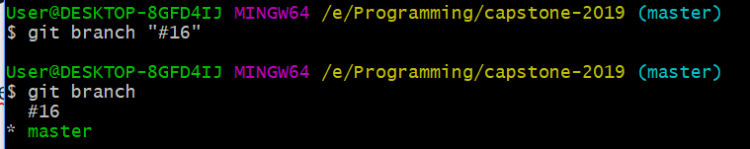


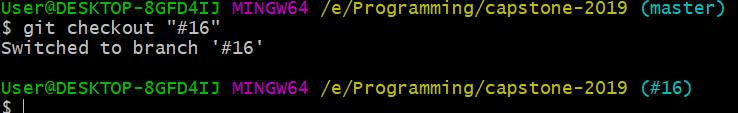
 

* 1. Github branch & pull Request & Issue 의 활용
     + 이슈 #16 번을 작업 하려고 합니다. Assignees에 자신을 등록합니다.



* + - 이슈번호와 동일하게 branch를 생성합니다.





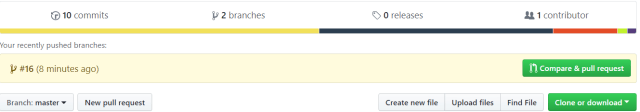
* + - 이슈에 해당하는 작업을 수행합니다

git add .

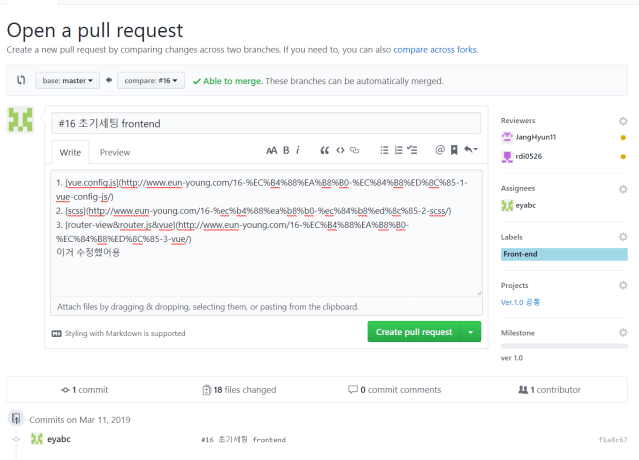
git commit -m “#16 front-end 초기세팅 완료”

git push origin “#16”

* + - push 완료 후 본인 계정의 github 저장소에 들어오면 Compare & pull reqeust 버튼이 활성화 되어 있습니다.



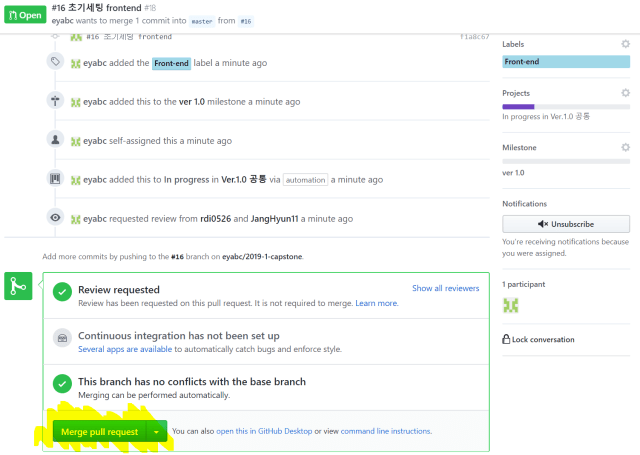
* + - pull Request를 작성합니다. 스크롤을 내려보면 파일이 어떻게 바뀌었는지도 조회가능합니다.



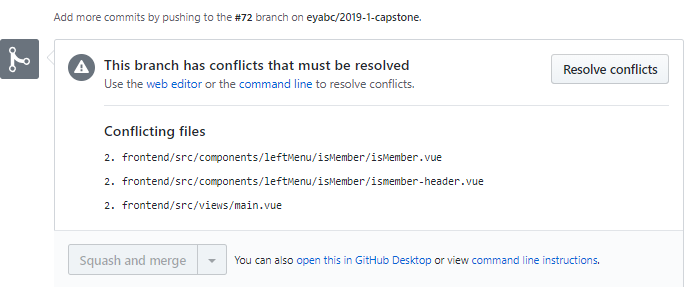
* + - 코드리뷰, Merge Pull Request

PR을 받은 Asignees는 코드 변경 내역을 확인하고 Merge를 합니다.

Conflict 가 있을 경우는 Merge를 할 수 없어, 실수를 방지할 수 있습니다.



* + - * Conflict를 웹에서 gui로 수정할 수 있습니다.





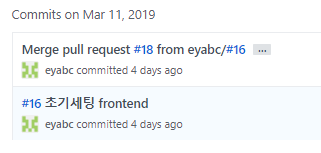
* + - Merge 할 때 세가지 옵션이 있습니다.

실험적 결과이므로 <https://meetup.toast.com/posts/122> 에서 정확한 개념을 믿어주세요.

* + - * Merge pull Request

merge를 했을 때 master의 commit에 branch에서 commit 한 것이 모두 올라갑니다.

사진에서 실제로 commit을 한번 하였고. Merge pull Request를 했다는 것도 커밋이 되었습니다.



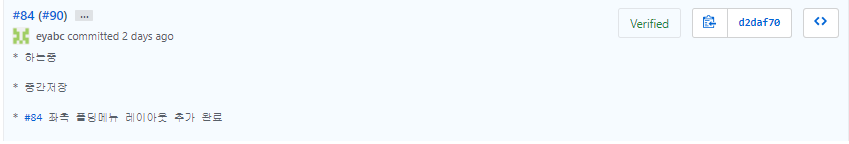
* + - * Squash and merge

master의 commit에 branch에서 최근에 커밋한 내용 하나가 올라갑니다.

… 을 눌러보면 브랜치에서 커밋한 모든 내역이 기록되어있습니다.

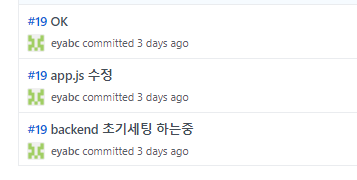
저희 조는 이 기능을 주로 사용합니다.



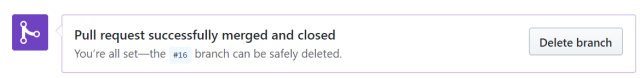
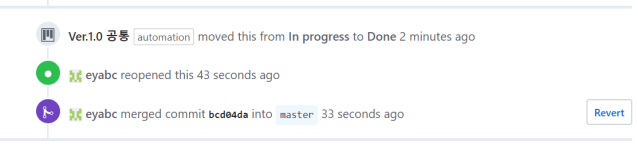


* + - * Rebase and merge

master의 commit에 branch에서 최근에 커밋한 내용으로 모두 올라가고 merge했다는 commit 은 추가되지 않는다.



* + - Merge 성공! 및 확인을 합니다.

* + - 모든 작업과 테스팅이 끝나면 이슈를 close합니다.

commit을 할 때 #이슈번호를 적어주어 이슈에 commit과 merge 한것이 자동으로 기록되었습니다.

