

레포지토리

≡ temp	
# 챕터번호	3

01. 레포지토리

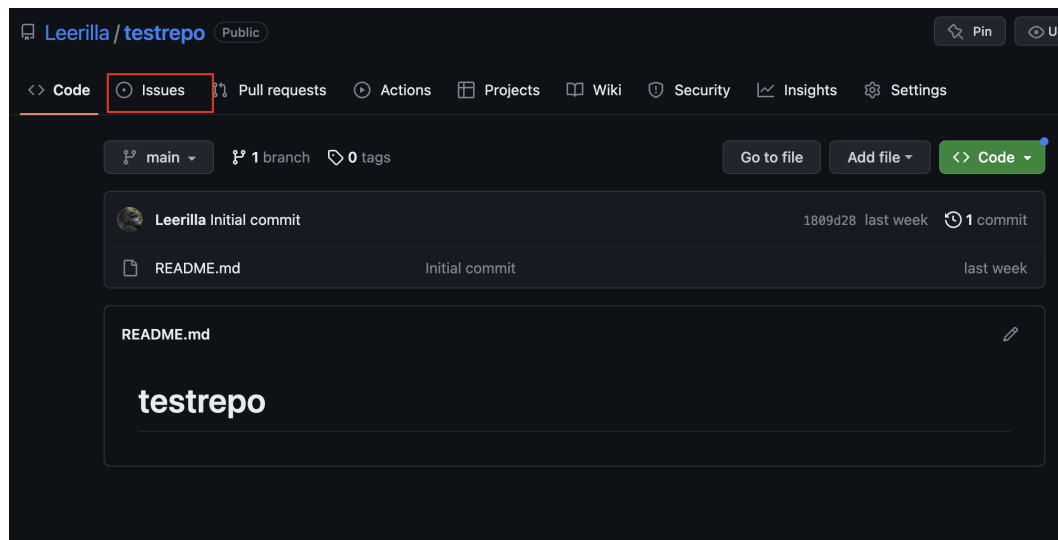
01-01. 레포지토리

01-01-01. 이슈

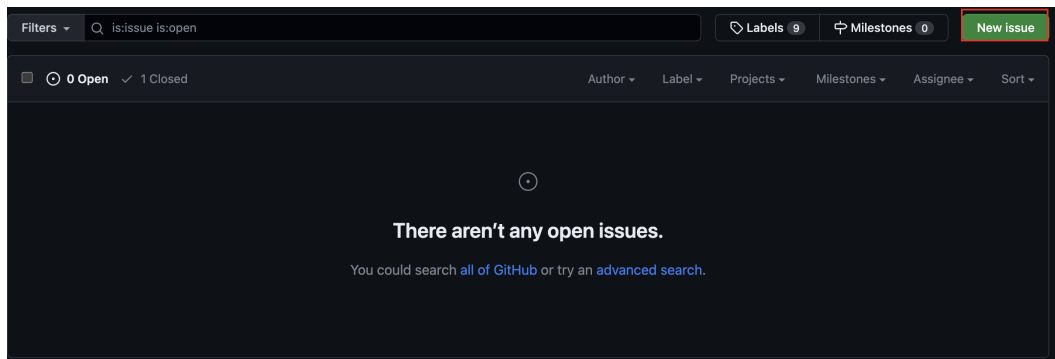
Git에서 이슈는 버그, 기능 요청 또는 프로젝트와 관련된 기타 작업을 추적하고 관리하는 방법으로 기본적으로 설명, 상태, 우선 순위, 관련 레이블 또는 이정표와 같은 문제에 대한 정보가 포함된 티켓 또는 작업이다. 이슈를 사용하면 프로젝트의 다른 기여자와 소통하거나 협업을 할 수 있게 되며 진행 상황을 추적하고 작업이 적시에 완료되도록 할 수 있다는 장점이 있다.

A. 이슈 생성하기

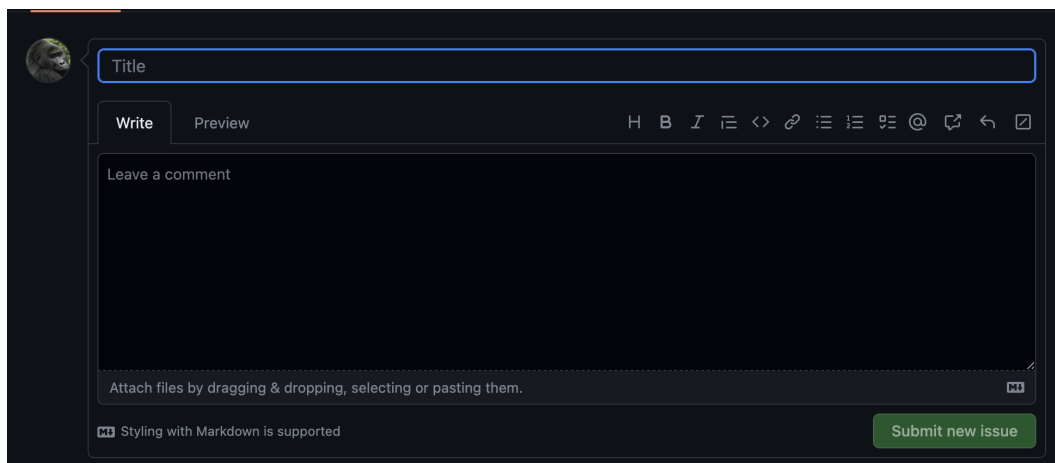
a. 이슈 탭으로 이동



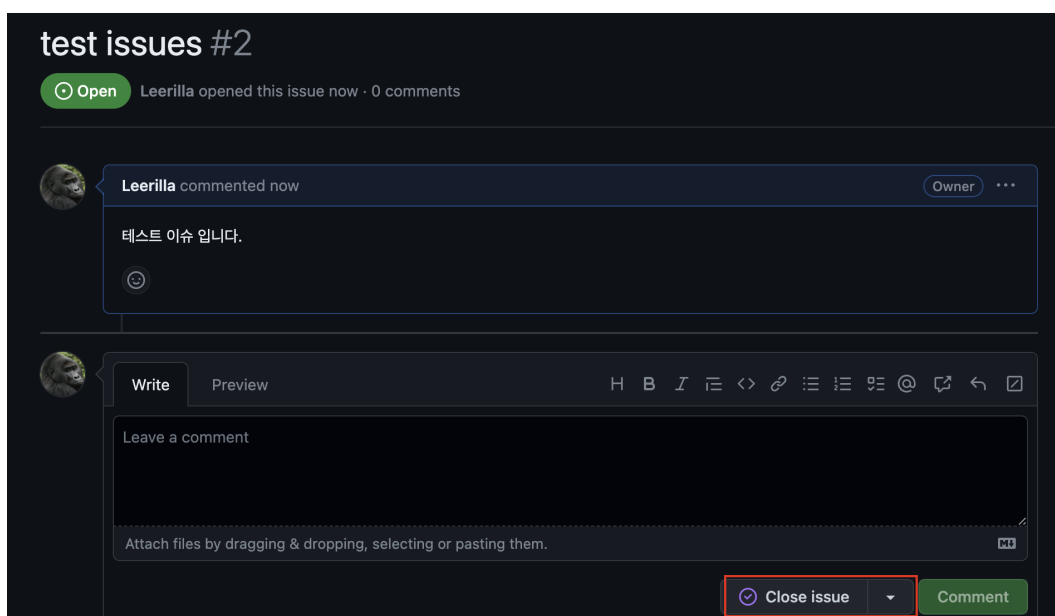
b. New Issue를 클릭한다.



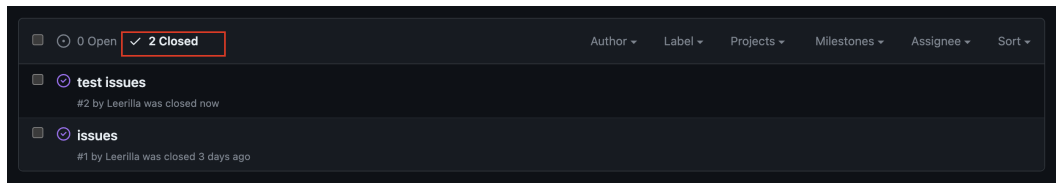
c. Title에 이슈 제목을 입력하고 Comment에 이슈 내용을 기입한다.



d. 이슈가 생성되면 다음과 같은 화면이 나오게 된다.
아래의 Leave Comment에 댓글을 작성하고 Comment를 클릭하면 다른 사람이 이슈를 닫을 수 있다.

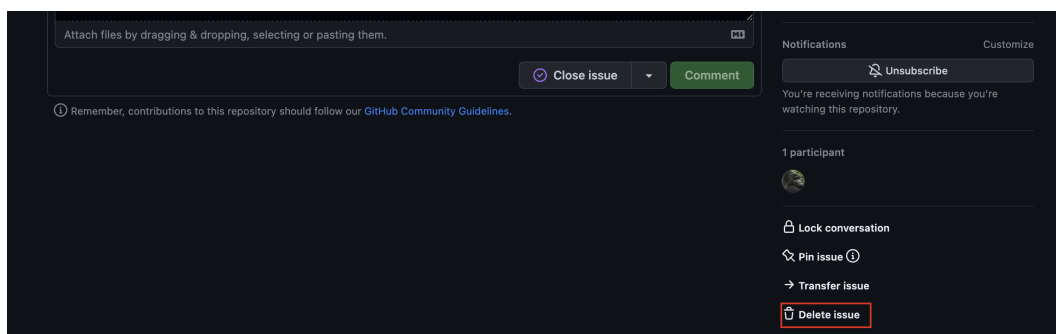


- e. 완료된 이슈는 이슈 탭에서 closed를 클릭하면 확인할 수 있다.



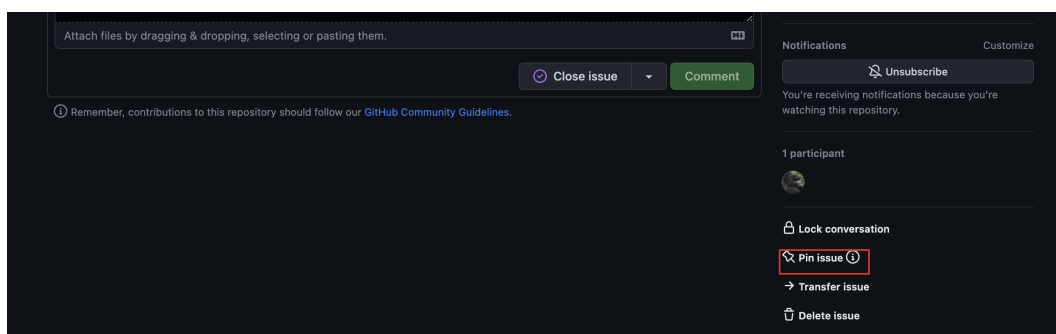
B. 이슈 삭제하기

- a. 삭제하고자 하는 이슈를 클릭하고 제일 하단으로 내린뒤 Delete issue를 클릭한다.

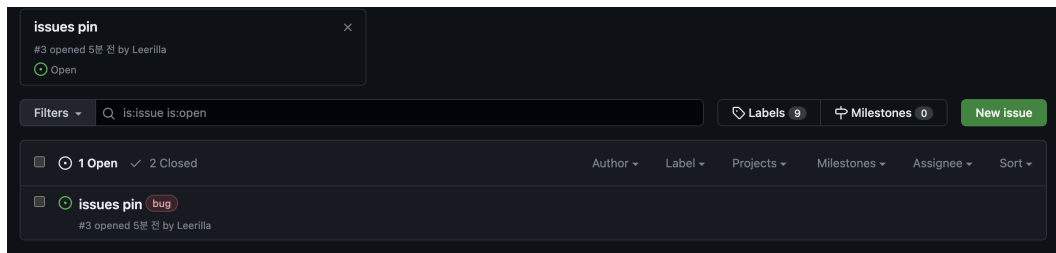


C. issue 상단에 고정하기

- a. 긴급한 이슈 혹은 우선순위가 높은 이슈는 모두가 보기 쉽게 하여 빠르게 작업을 하는 것이 좋은데
이러한 경우 issue pin 기능을 이용하여 해당 이슈를 상단에 고정하는 것이 가능하다.
스크롤을 아래로 내린뒤 pin issue를 클릭하면 해당 이슈는 상단에 고정된다.



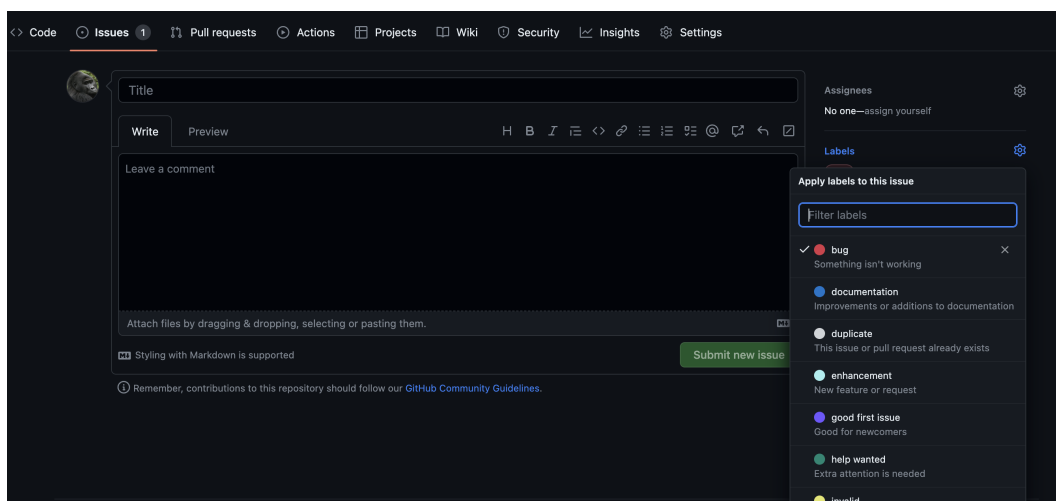
- b. 상단에 고정된 이슈는 다음과 같이 확인하는 것이 가능하다.



D. labels 적용하기

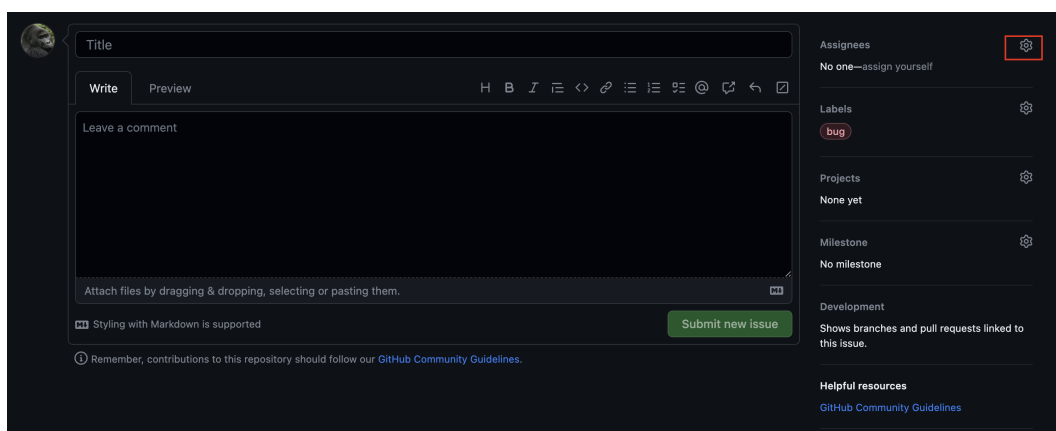
- 아래와 같이 이슈를 생성하는 부분에서 우측 메뉴에서 labels의 설정(⚙️) 아이콘을 클릭한다.

이후 해당 내용에서 issues에 맞는 labels를 클릭한다.



E. 이슈 담당자 설정하기

- issues 생성 메뉴에서 우측에 Assignees의 설정(⚙️) 아이콘을 클릭한다. 이후 해당 이슈를 해결할 담당자를 선택한다.

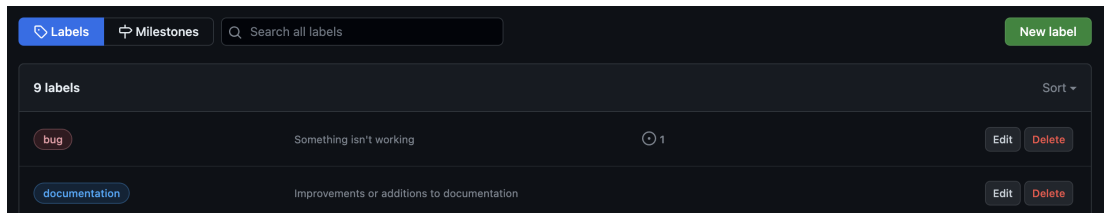


1-1-2. labels

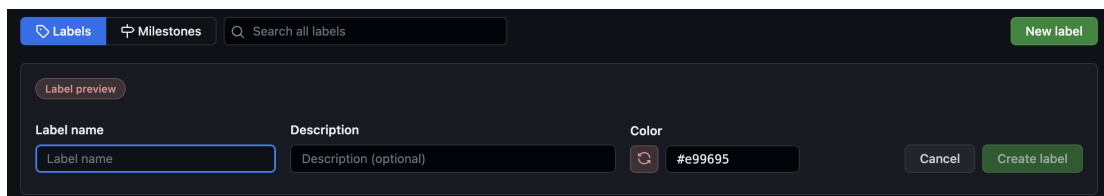
이슈를 생성하게 되면 issues가 기본 설정으로 되어 있다. 그러나 우리가 이러한 이슈를 여러개 생성해서 관리를 하게 된다고 하면 모든 문제를 이슈인 하나의 태그로 식별을 해야하기 때문에 실행중인 서비스에서 오류가 발생된 것인지 확인하기가 어렵다. 그렇기 때문에 이슈를 생성하면 labels를 이용하여 식별을 해주게 된다.

A. labels 생성하기

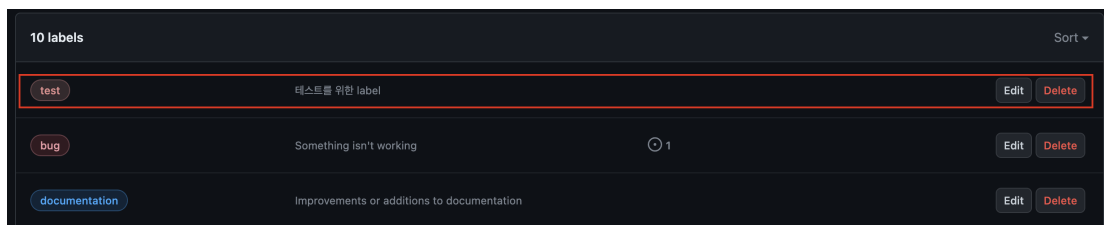
- a. New label을 클릭한다.



- b. 생성된 필드에 label의 이름과 생상 설명을 작성해준다.

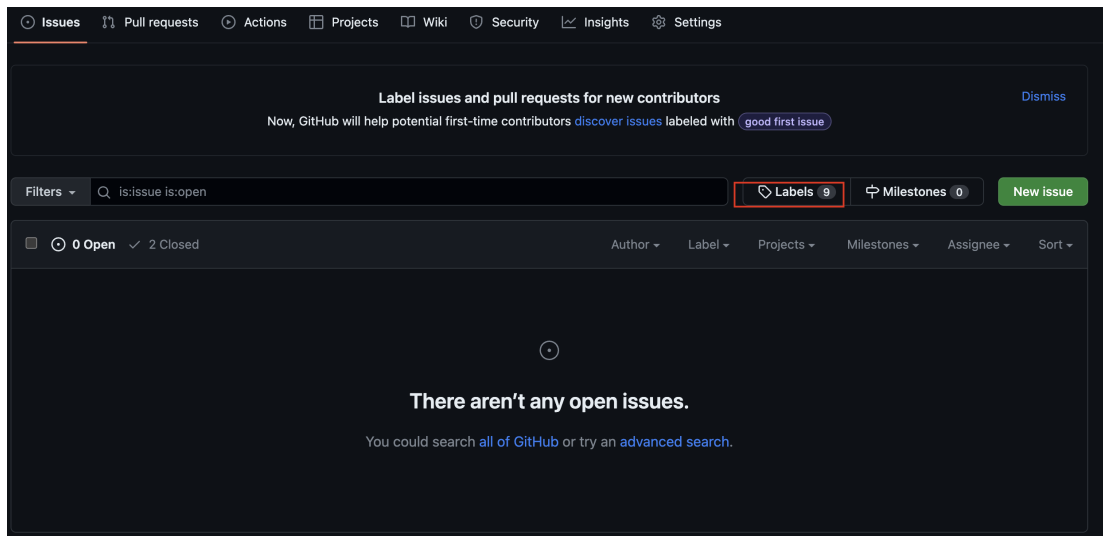


- c. labels를 추가하면 다음과 같이 추가 된 것을 볼 수 있다.

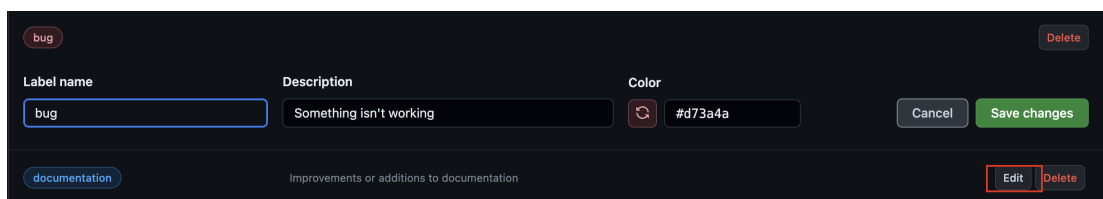


B. labels 편집하기

- a. issues 메뉴에서 labels를 클릭한다.

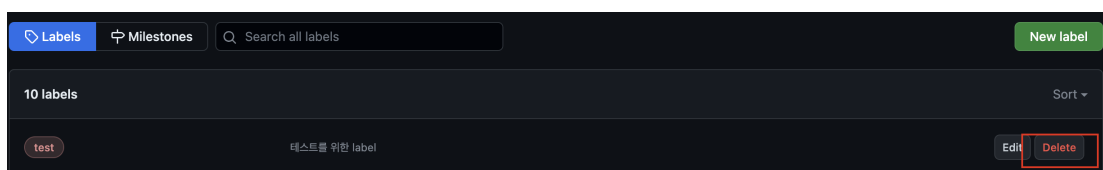


- b. 9가지 labels가 기본으로 설정되어 있으며 필요시 추가 생성 및 이름, 색상, 내용의 변경도 가능하다.
 변경하고자 하는 labels에 Edit 부분을 클릭하면 상단에 Bug와 같은 화면이 표시가 되며 위 내용을 수정하여 변경이 가능하다.



C. labels 삭제하기

- a. 삭제하고자 하는 labels의 Delete를 클릭한다.



1-1-3. Pull requests

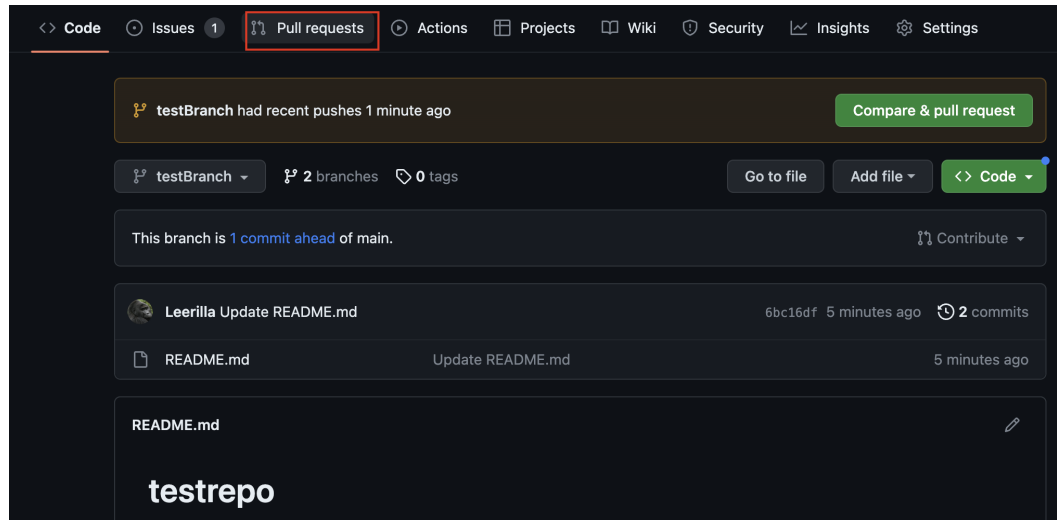
GitHub에서 호스팅되는 프로젝트의 코드베이스에 대한 변경 사항을 제안할 수 있는 기능이 다.

제안된 변경 사항이 기본 코드베이스에 병합되기 전에 다른 사람들이 제안된 변경 사항을 검토하고 의견을 제시할 수 있도록 하여 개발자가 코드 변경 사항에 대해 공동 작업을 수행할 수 있는 방법이다.

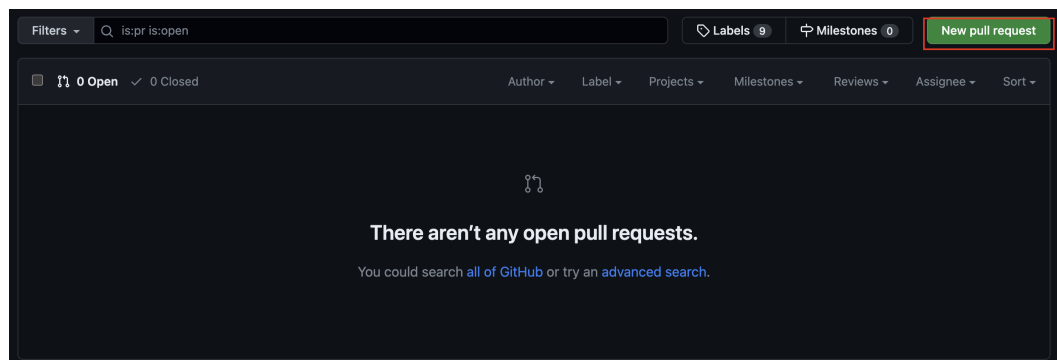
A. Project Pull Request

Git branch 전략을 통해 각각의 브랜치를 생성해서 코드를 관리한뒤 하나의 브랜치로 합치는 경우 Pull Request 요청을 해야하는데 다음과 같은 방식으로 할 수 있다.

a. Pull Requests를 클릭한다.



b. New Pull Request를 클릭한다.

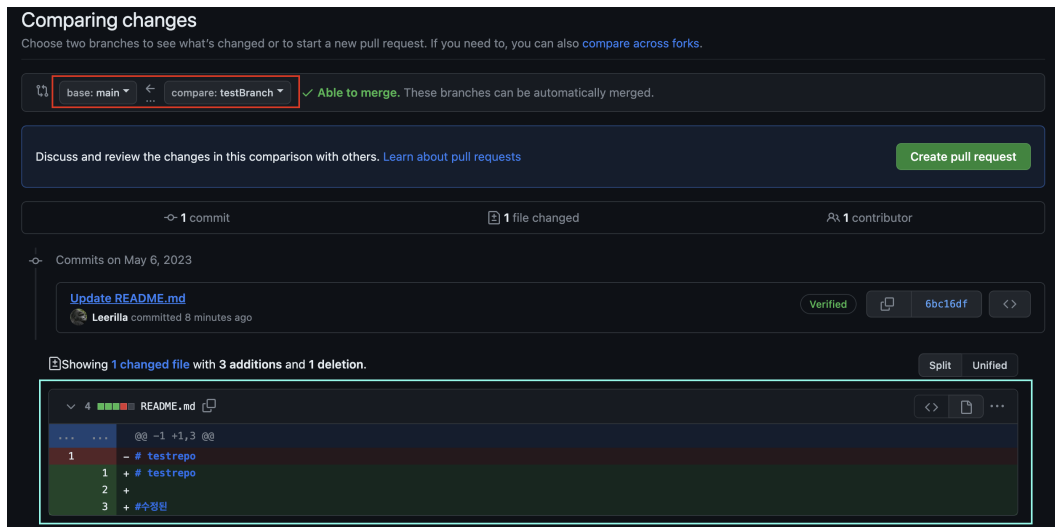


c. 빨간색 테두리의 경우 우측이 변경된 Branch 이고 좌측이 병합하고자 하는 브랜치이다.

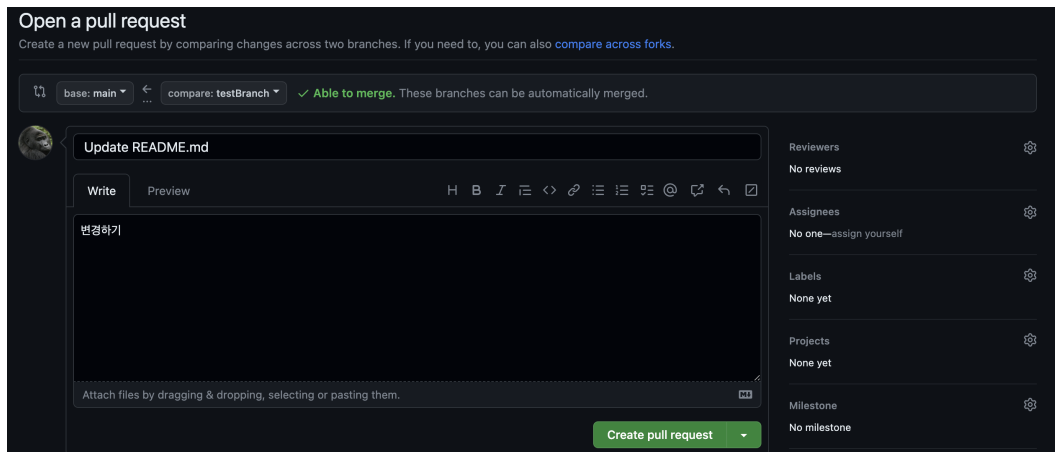
청녹색 테두리는 변경된 내용을 알려주는 공간이다.

다음 공간에서 충돌등의 문제를 해결할 수 있다.

Create Pull Request를 클릭한뒤 요청을 날린다.



d. Create pull Request 병합하고자 하는 이유를 작성하여 요청을 보낸다.

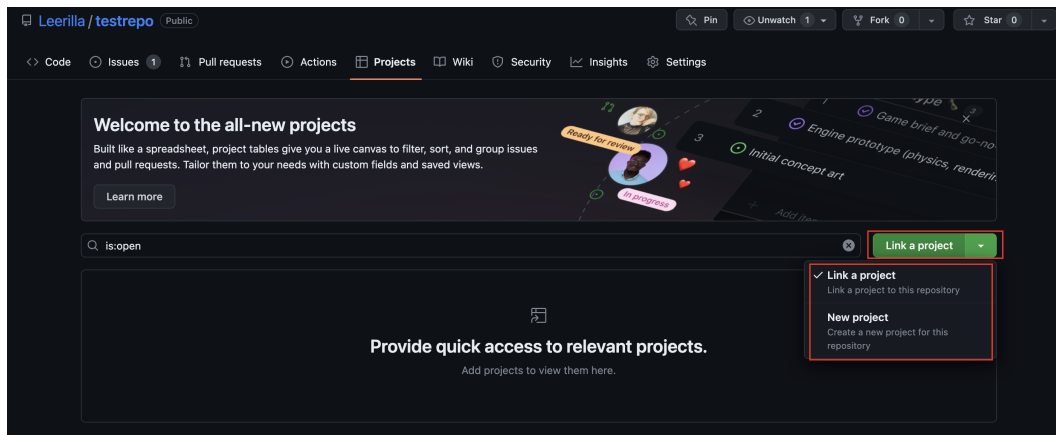


1-1-4. Project

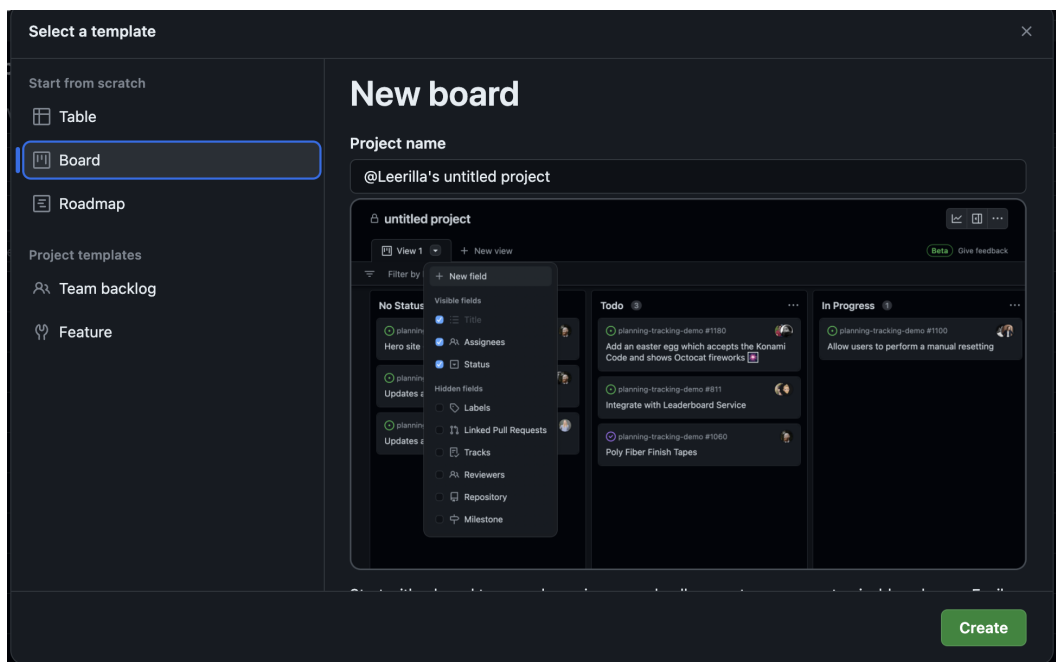
Project 메뉴는 Git을 이용하여 프로젝트를 관리하는 부분으로 개발 회사에서는 FP(Function Point)를 도출하여 기능을 산출하고 프로젝트의 단가를 산출하는 방식으로 프로젝트를 수주받게 되는데 이와 같이 신규 프로젝트 완성을 위해 개발해야 하는 항목과 우선 순위 등을 정하여 PM과 같은 프로젝트 매니저 혹은 관리자가 업무를 할당하게 된다.

A. Project 사용하기

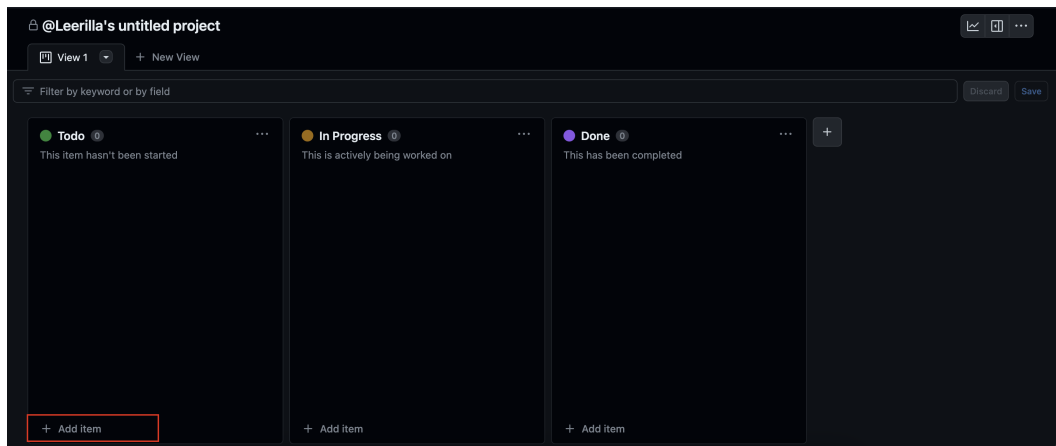
- a. 상단의 Project 탭을 클릭하면 아래와 같은 화면이 나오게 되며 화살표를 클릭하면 아래와 같이 기존 프로젝트와 연결할 것인지 아니면 새로운 프로젝트를 생성할 것인지 선택하는 부분이 나오게 된다.



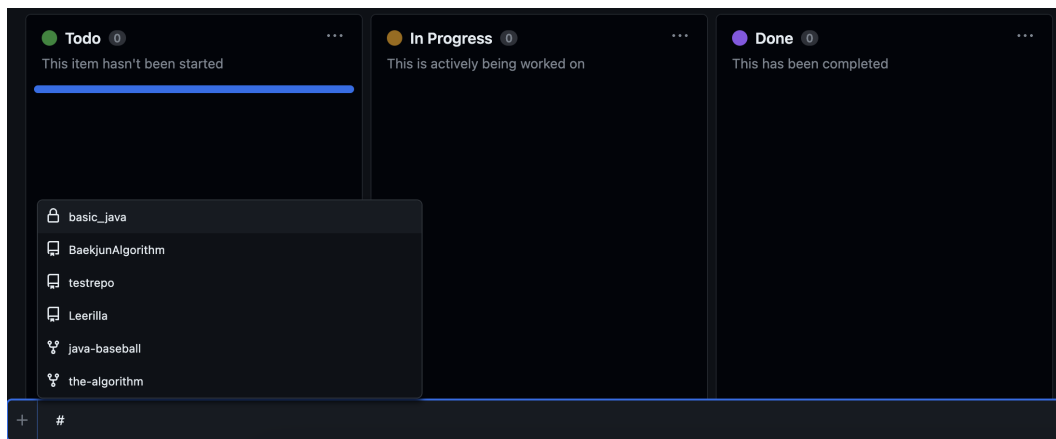
- b. New Project를 클릭하고 생성을 하면 아래와 같이 프로젝트 관리 UI를 선택하는 화면이 나온다.
- 사용자가 편리한 방식으로 선택하면 되지만 보통은 Board 형식으로 프로젝트를 관리한다.
- Board는 칸반보드 UI로 프로젝트의 작업을 시각화하고 제한하여 작업의 효율성을 높이는 애자일 관리 도구이다.



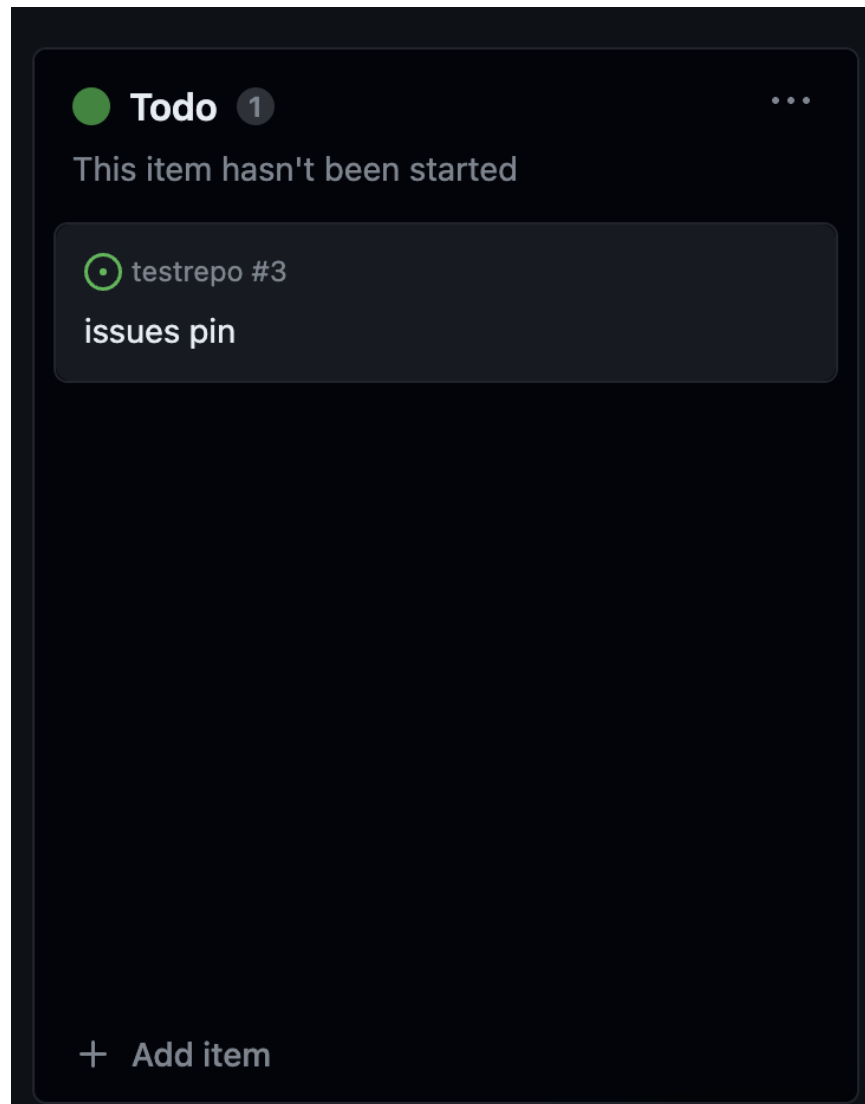
- c. 하단에 +Add item을 클릭한다.



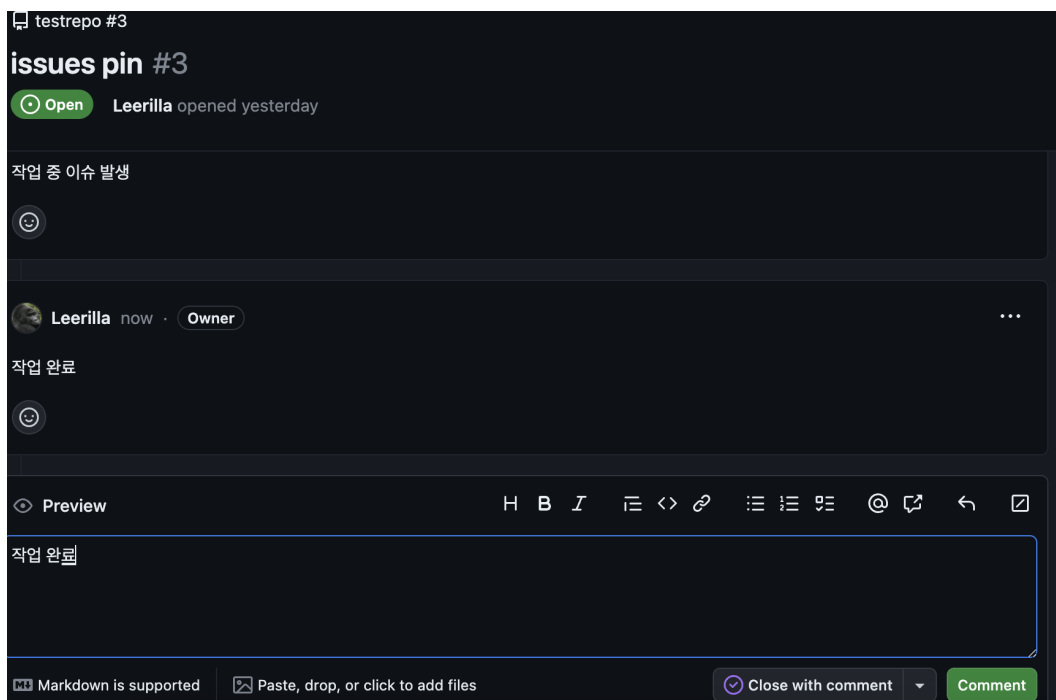
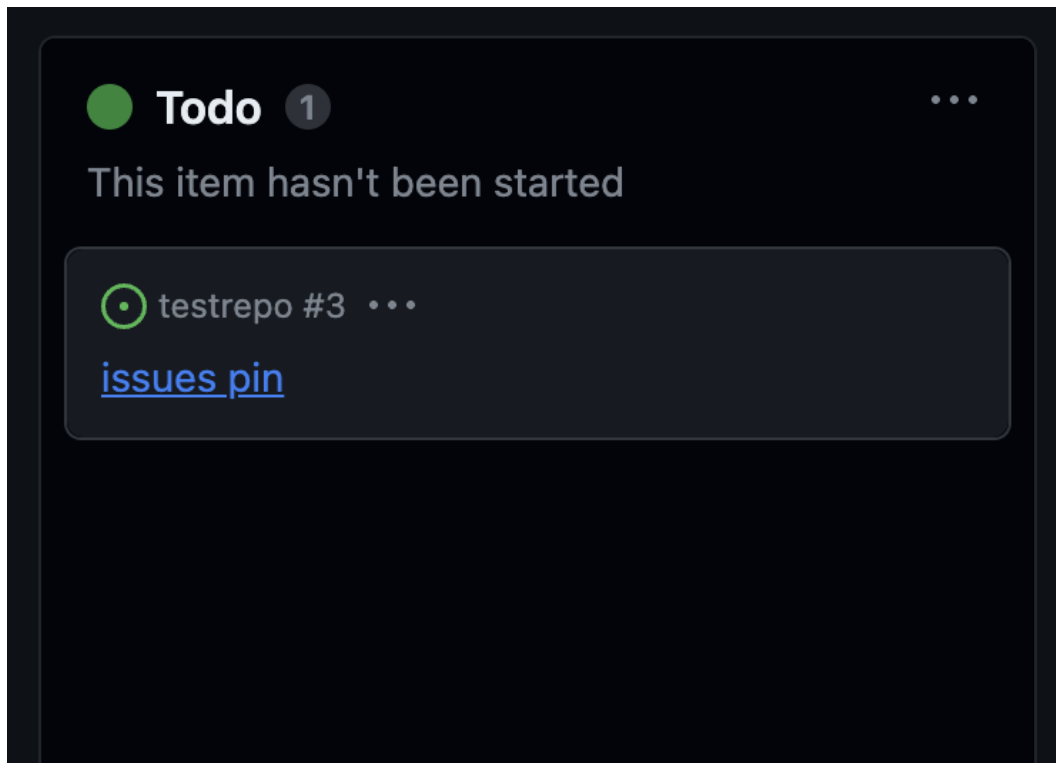
- d. 클릭을 하게 되면 다음과 같이 하단의 메뉴바가 변경되는데 여기서 #을 입력하여 등록하고자 하는 레파지토리를 선택한다. 이후 해당 레파지토리에 등록된 issues를 선택한다.



- e. 이후 등록을 하게 되면 다음과 같이 등록된 것을 볼 수 있다.
이후 작업이 완료가 된 것은 드래그를 활용해 이동을 하는 것이 가능하다.



- f. 해당 작업을 완료하거나 이슈가 있는 경우 내용을 등록하여 완료를 하거나 진행 전 확인 사항등의 내용을 기입하는 것이 가능하다.
1. 해당 작업의 제목을 클릭한다.
 2. 작업의 내용을 작성한다.
 3. 이슈를 닫는다.



1-1-5. Wiki

소프트웨어를 개발하다 보면 가장 중요한 것은 문서라는 것을 알 수 있을것이다. 이는 소프트웨어의 규모가 커지면 커질수록 점차 해당 문서의 중요도가 더 높아지게 되는데 이는 규모가 커짐에 따라서 복잡도가 증가하게 된다.

이때 문서화 된 것이 없다면 작업자 간의 커뮤니케이션 비용이 증가하게 되며 새로운 멤버가 추가 되거나 개발 완료이후 유지보수 및 업데이트 등의 추가 작업이 요구되는 상황이 발생하

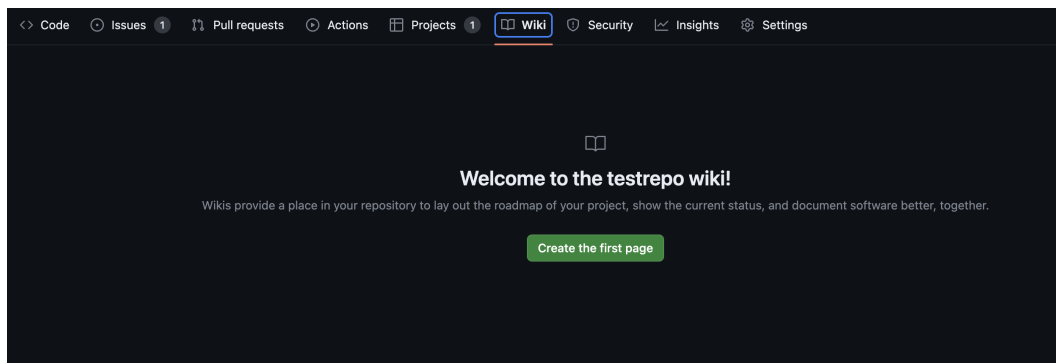
게 된다면 인수인계 비용이 배로 증가하는 문제가 발생되기 때문이며 오픈소스를 창시하고자 하는 경우 다른 개발자가 나의 프로젝트의 목적 및 구조와 개념을 이해할 수 있는 가이드를 남겨야 다른 사용자가 쉽게 참여할 수 있게 된다.

Wiki는 이러한 것과 같이 우리가 개발하는 프로젝트의 가이드 문서와 같은 역할을 하게 되며 귀찮더라도 작성하는 것을 습관화 하는 것이 중요하다.

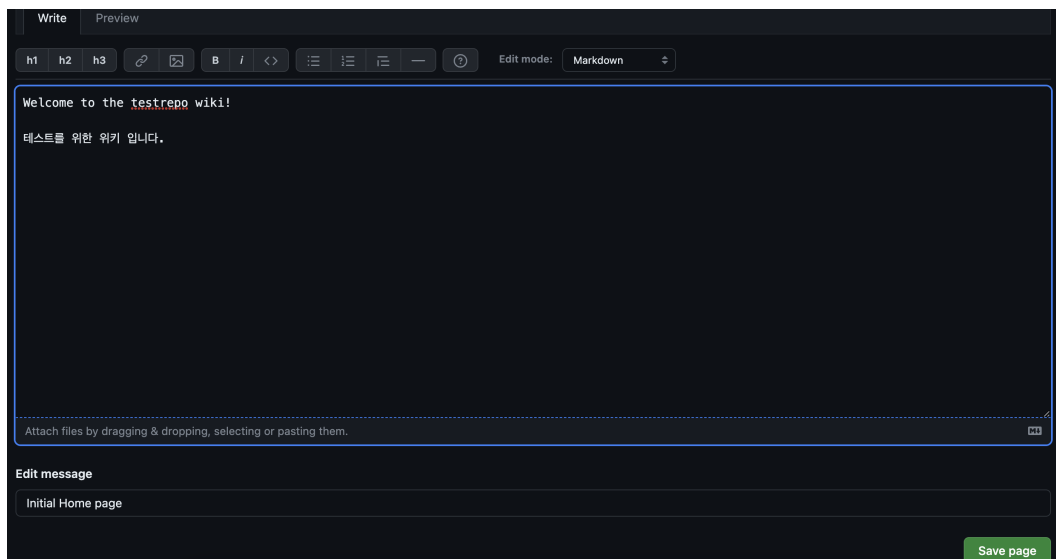
A. Wiki 작성하기

Wiki는 마크다운 언어로 작성이 된다.

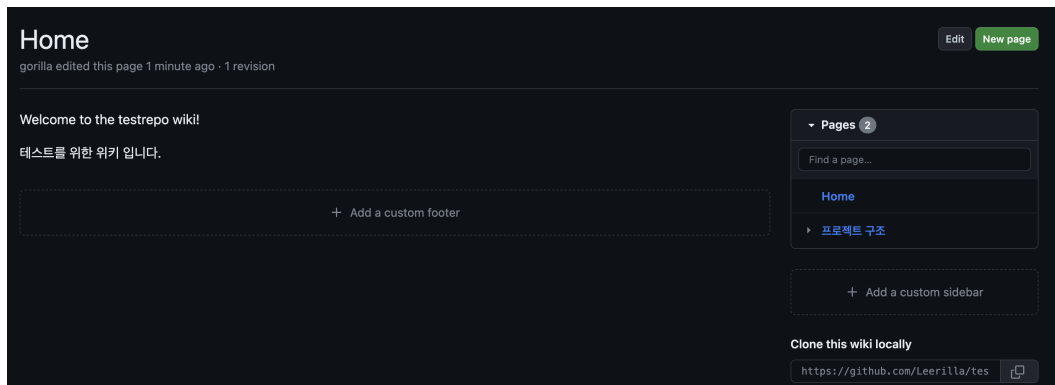
- a. 상단의 Wiki 탭을 클릭하고 이후 Create the first Page를 클릭한다.



- b. 위키의 내용을 작성하고 하단의 Save를 클릭한다.



- c. 저장한 이후 새로운 내용을 추가하고자 하는 경우 우측 상단에 New Page를 클릭하여 내용을 작성
작성이 완료되면 Pages에 목록이 추가되는 것을 볼 수 있다.

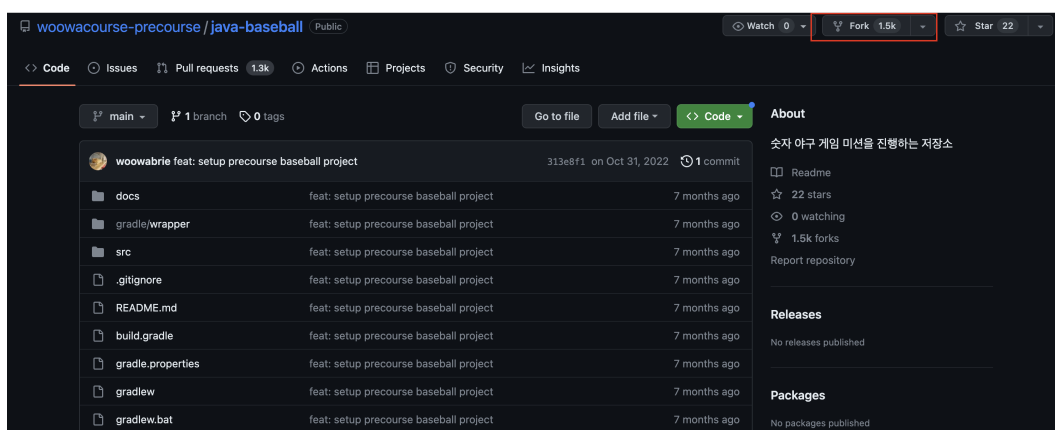


1-1-6. Fork

fork는 다른 사람의 GitHub Repository에서 내가 어떤 부분을 수정하거나 추가 기능을 넣고 싶을 때 해당 repository를 내 GitHub Repository로 그대로 복제하는 기능으로 다른 사람의 Repository와 연결되어 original의 변경 사항을 계속 모니터링 할 수 있는 기능이다.

A. 다른 레파지토리 Fork하기

- a. Fork하고자 하는 Repository를 선택한뒤 우측 상단에 Fork를 클릭한다.





- b. Fork하고자 하는 Repository의 이름을 설정한뒤 Create Fork를 클릭한다.

Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks.](#)

Owner * **Repository name ***


 Leerilla / java-baseball 

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

숫자 야구 게임 미션을 진행하는 저장소

☒ **Copy the `main` branch only**
 Contribute back to woowacourse-precourse/java-baseball by adding your own branch. [Learn more.](#)

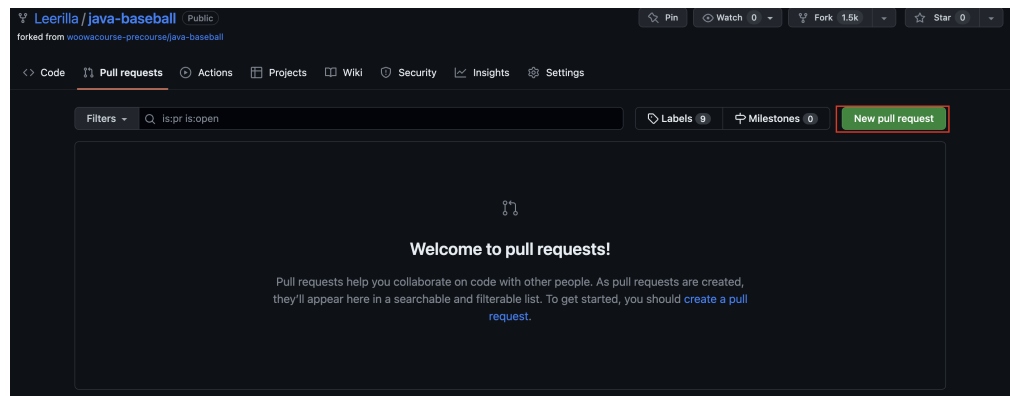
 You are creating a fork in your personal account.

Create fork

B. Fork PR(Pull request)

Fork 이후 Origin Repository에 Pr을 요청하는 것이 가능하며 해당 기능은 오픈 소스 작업과 같이 권한이 없으나 제안을 하고 싶은 경우 사용할 수 있다.

a. Pull Request 메뉴로 이동 후 New Pull Request 클릭



b. 1번은 Fork 사용자가 변경사항을 반영하고자 하는 브랜치를 선택하는 공간이다.

2번은 Fork 대상의 레파지토리를 선택하고 요청을 날리고자 하는 브랜치를 선택하는 공간이다.

위 설정을 마무리 하면 Create Pull Request 요청을 하면 해당 레파지토리의 주인의 승인 후 반영이 되도록 설정이 되어 있다.

