



팀 프로젝트를 위한 Git & GitHub 사용법

☰ 태그	배포용 팀프로젝트
☰ 교육기관	코딩온 포스코
☰ 과정명	K-Digital training 웹 개발자 양성 프로젝트 3기



교육장에서 제공되는 교육자료는 외부 반출 금지입니다. 블로그 업로드, 요약하여 게시, 타인에게 공유 등의 행위를 하지 말아주세요.

Git, GitHub으로 슬기로운 팀플을 해봅시다!!💪

용어 정리

- 원격 저장소와 로컬 저장소 동기화
 - 로컬 저장소에서 작업할 브랜치 생성
 - 해당 브랜치에서 기능 개발 진행
 - 기능 개발 완료 후, Github에 올리기
 - PR (Pull Request) 생성하기
 - Merge 하기
 - Merge 확인
 - Local/Remote Branch 삭제
- 주의 사항

용어 정리

- local = 내 PC = 로컬 저장소
- remote = origin = github repository = 원격 저장소

1. 원격 저장소와 로컬 저장소 동기화

새로운 branch 를 생성하기 전 local의 **main** 브랜치는 항상 최신이어야 합니다!

- main 브랜치로 이동 후
- 원격 저장소에 있는 내용을 local에 가져오기
 - 다른 사람이 main에 코드를 merge 했다면 **remote의 main에 변경 사항이 있으니 나의 local에도 반영해야겠죠?** (= **pull**)

```
> git checkout main # 로컬에서 main branch로 이동
> git pull origin main # origin의 main 브랜치를 나의 main 브랜치로 pull!
```

2. 로컬 저장소에서 작업할 브랜치 생성

▼ 새로운 기능을 개발할 때 새로운 브랜치를 생성해 작업

```
> git branch [브랜치_이름] # branch 생성
> git checkout [브랜치_이름] # 해당 branch 로 이동
> code . # 해당 branch에서 VSCode 열기

ex)
> git branch addNewsPage
> git checkout addNewsPage
```

▼ 🐹 한 번에 브랜치를 생성하고 해당 브랜치로 이동하기

```
> git checkout -b [브랜치_이름] # branch 만들면서 이동
> code . # 해당 branch에서 VSCode 열기
```

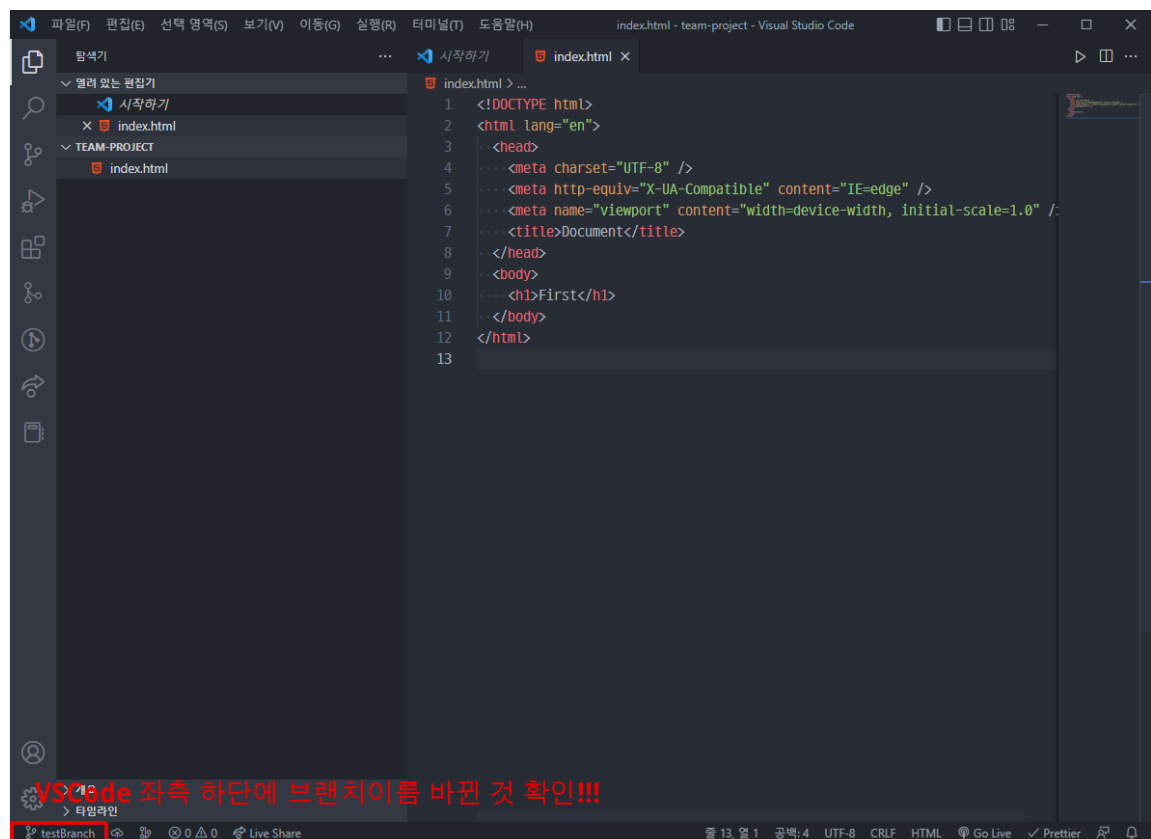
▼ Ex

```
MINGW64:/d/team-project

spreatic@DESKTOP-BHR026U MINGW64 /d/team-project (main)
$ git checkout -b testBranch
Switched to a new branch 'testBranch'

spreatic@DESKTOP-BHR026U MINGW64 /d/team-project (testBranch)
$ code .

spreatic@DESKTOP-BHR026U MINGW64 /d/team-project (testBranch)
$ |
```



▼ (MacOS) Terminal에서 `code .` 명령어 적용하기
[블로그 참고](#)

3. 해당 브랜치에서 기능 개발 진행

열심히 개발을 진행합니다💪

4. 기능 개발 완료 후, Github에 올리기

▼ `git add & commit`

```
> git add .  
> git commit -m "type: subject"  
  
ex) git commit -m "feat: 로그인 화면 구현"
```

▼ Commit Convention Type ([Ref](#))

- `feat`: 새로운 기능 추가
- `fix`: 버그 수정
- `docs`: 문서 수정
- `styles`: 코드 포매팅, 세미콜론 누락, 코드 변경이 없는 경우
- `refactor`: 코드 리팩토링
- ...

▼ `git push origin [브랜치_이름]`

```
# 원격 저장소에 해당 브랜치 이미 존재할 경우  
# 주의) 원격에 브랜치 없는데 git push 명령어만 입력하면, upstream 없다는 오류 발생!  
> git push  
  
# 원격 저장소에 해당 브랜치 없을 경우  
> git push origin [브랜치_이름]
```

```
MINGW64:/d/team-project

spreatic@DESKTOP-BHR026U MINGW64 /d/team-project (testBranch)
$ git add .

spreatic@DESKTOP-BHR026U MINGW64 /d/team-project (testBranch)
$ git commit -m "feat: do something"
[testBranch c24c6ae] feat: do something
1 file changed, 1 insertion(+)

spreatic@DESKTOP-BHR026U MINGW64 /d/team-project (testBranch)
$ git push origin testBranch
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 335 bytes | 335.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'testBranch' on GitHub by visiting:
remote:   https://github.com/sean-spreatic/team-project/pull/new/testBranch
remote:
To github.com:sean-spreatic/team-project.git
 * [new branch]      testBranch -> testBranch

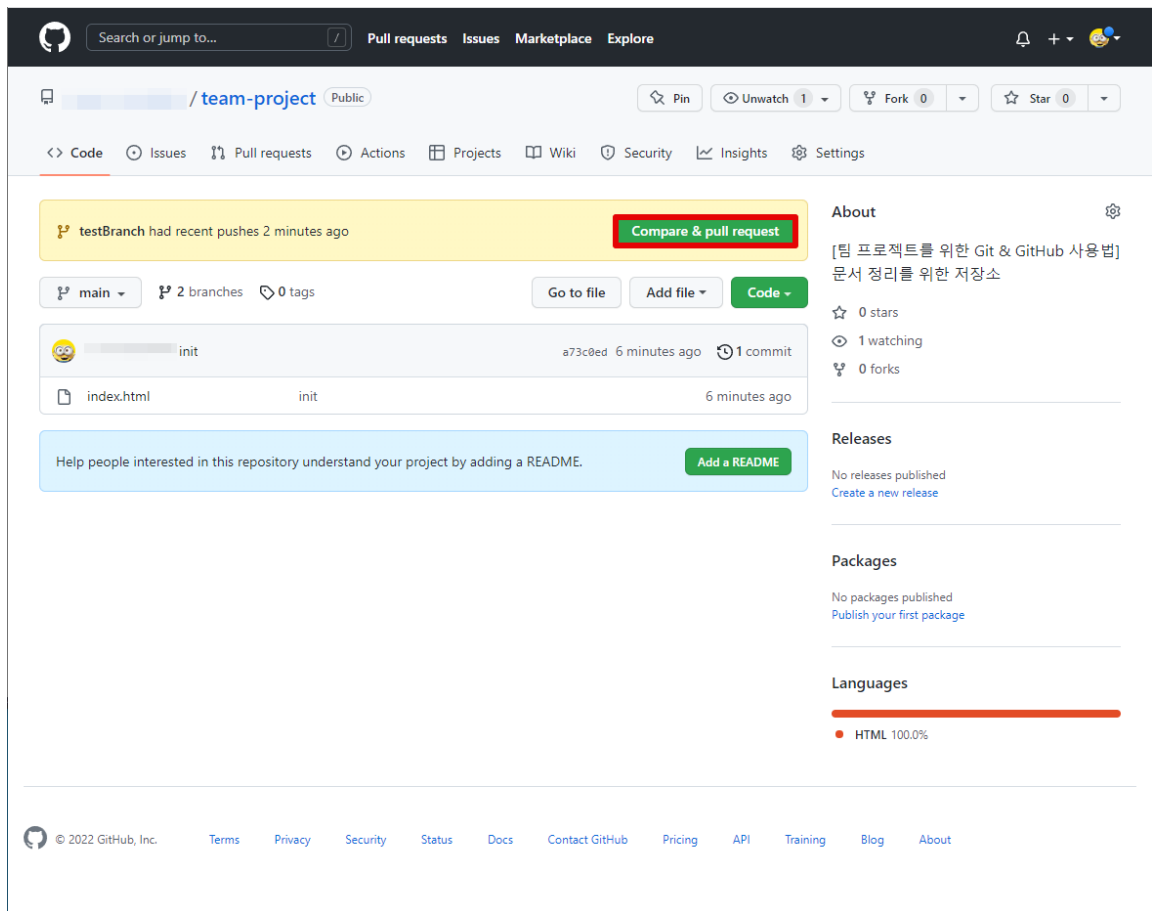
spreatic@DESKTOP-BHR026U MINGW64 /d/team-project (testBranch)
$
```

현재 local branch는 testBranch 이죠?
따라서 push 할 origin도 testBranch여야 합니다!!

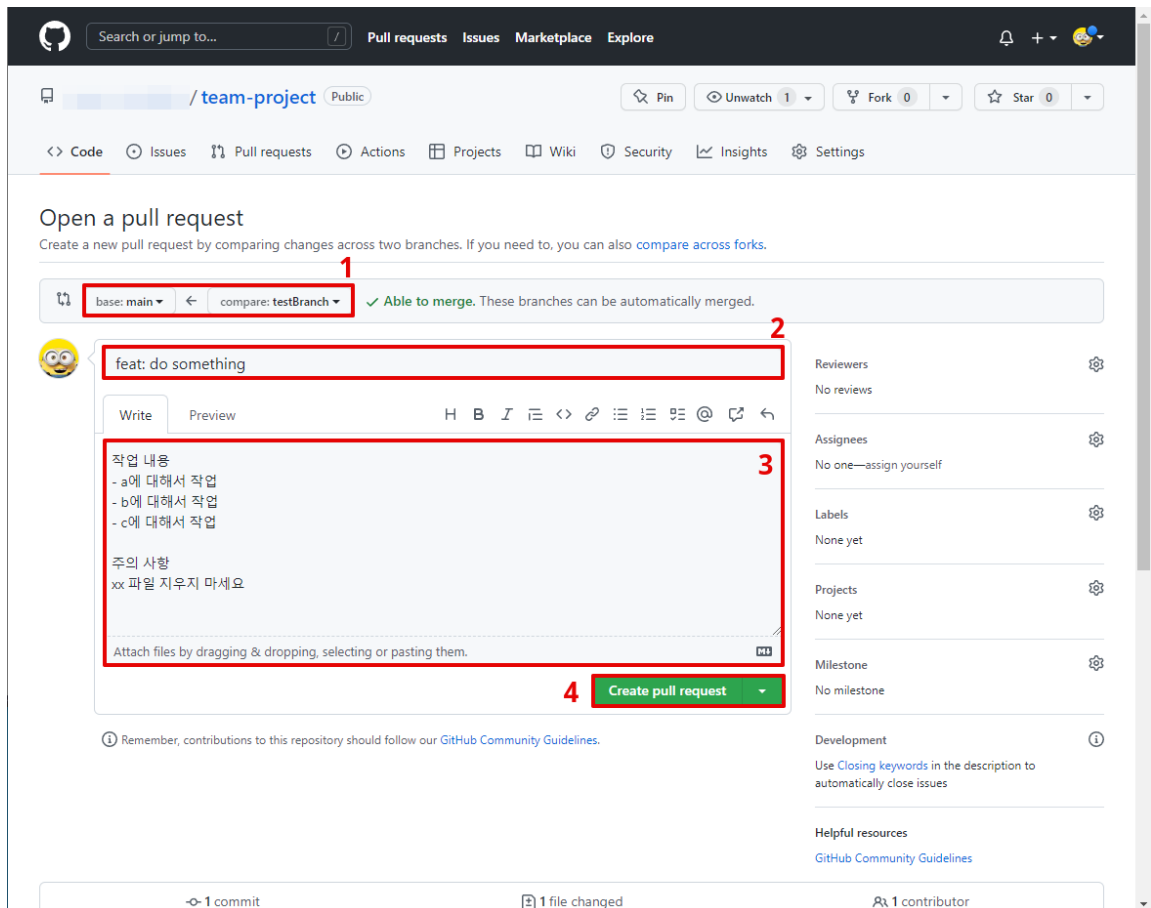
5. PR (Pull Request) 생성하기

여러분이 push 하신 branch 에 대한 pull request 문서를 생성하겠습니다!

- ▼ 1. GitHub Project Repository로 이동 → [Compare & pull request] 클릭

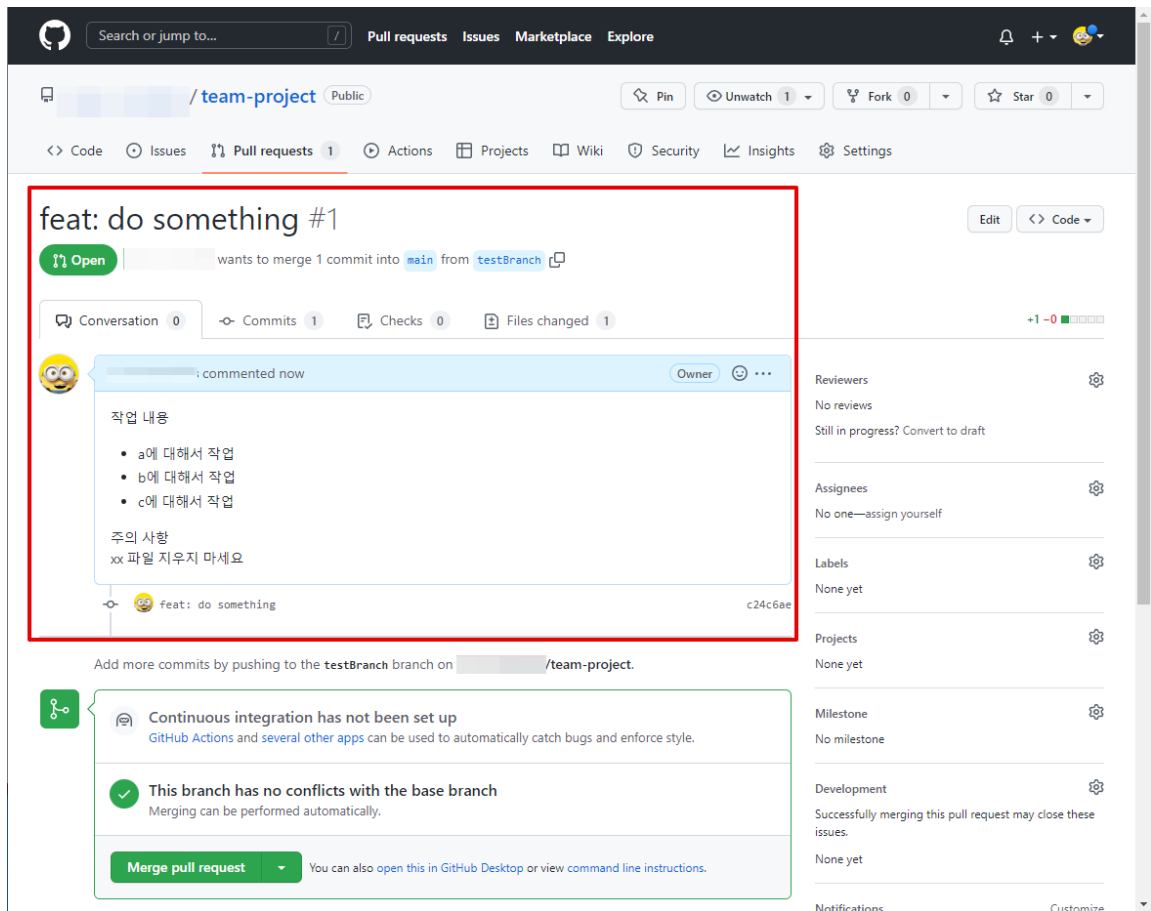


▼ 2. PR 문서 작성하기



- (1) base: **main** ← compare: **브랜치 이름**
 - base와 compare 올바른지 확인하기!
 - 해당 PR이 origin의 **브랜치 이름** branch와 origin의 **main** branch를 머지하는 작업임을 표시합니다.
- (2) pr title: 제목을 보고 어떤 작업을 확인했는지 파악할 수 있도록 간결 명확히 작성
- (3) pr body: 개요, 내용, 스크린샷, 링크, 개선 사항, 주의 사항 등 pr을 상세히 작성
- (4) [Create pull request] 버튼 클릭

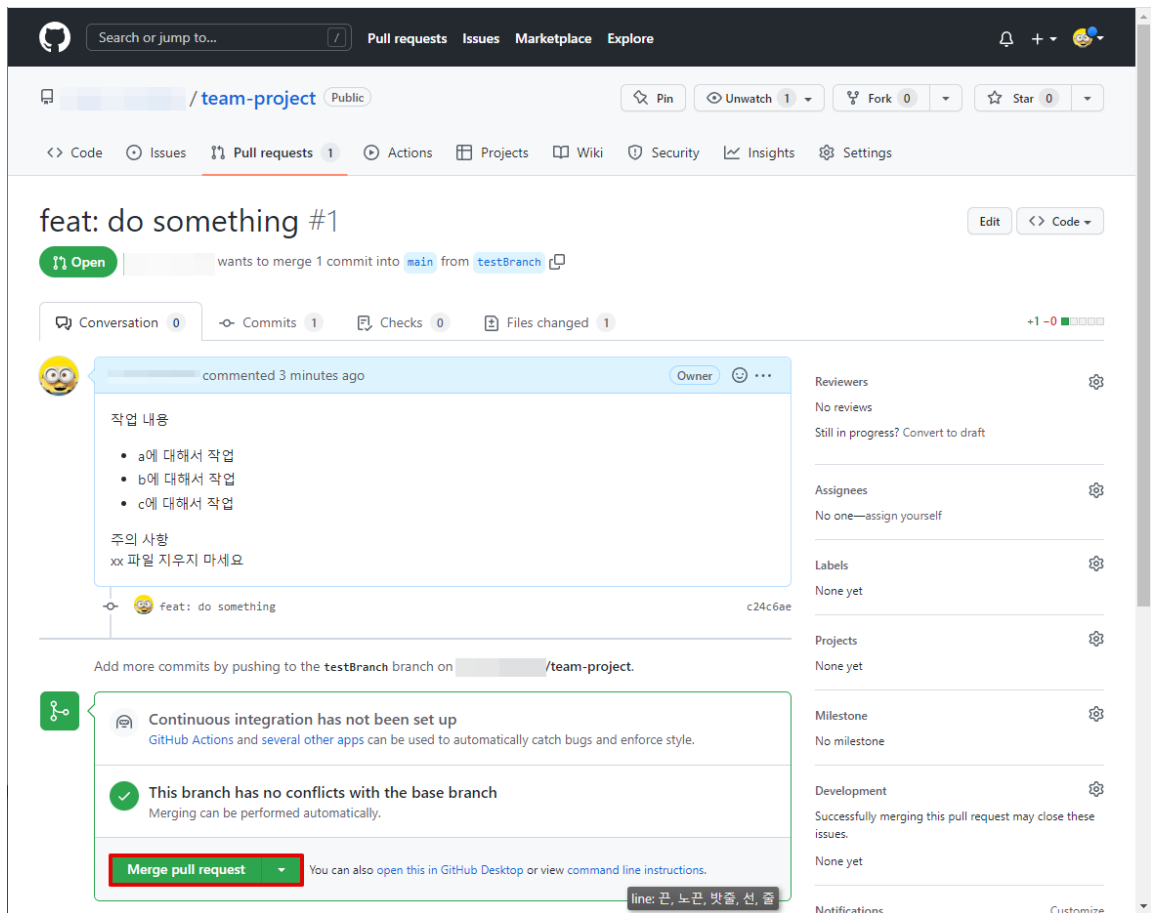
▼ 3. PR 문서 확인하기



문서가 올바르게 작성되었는지 확인!

6. Merge 하기

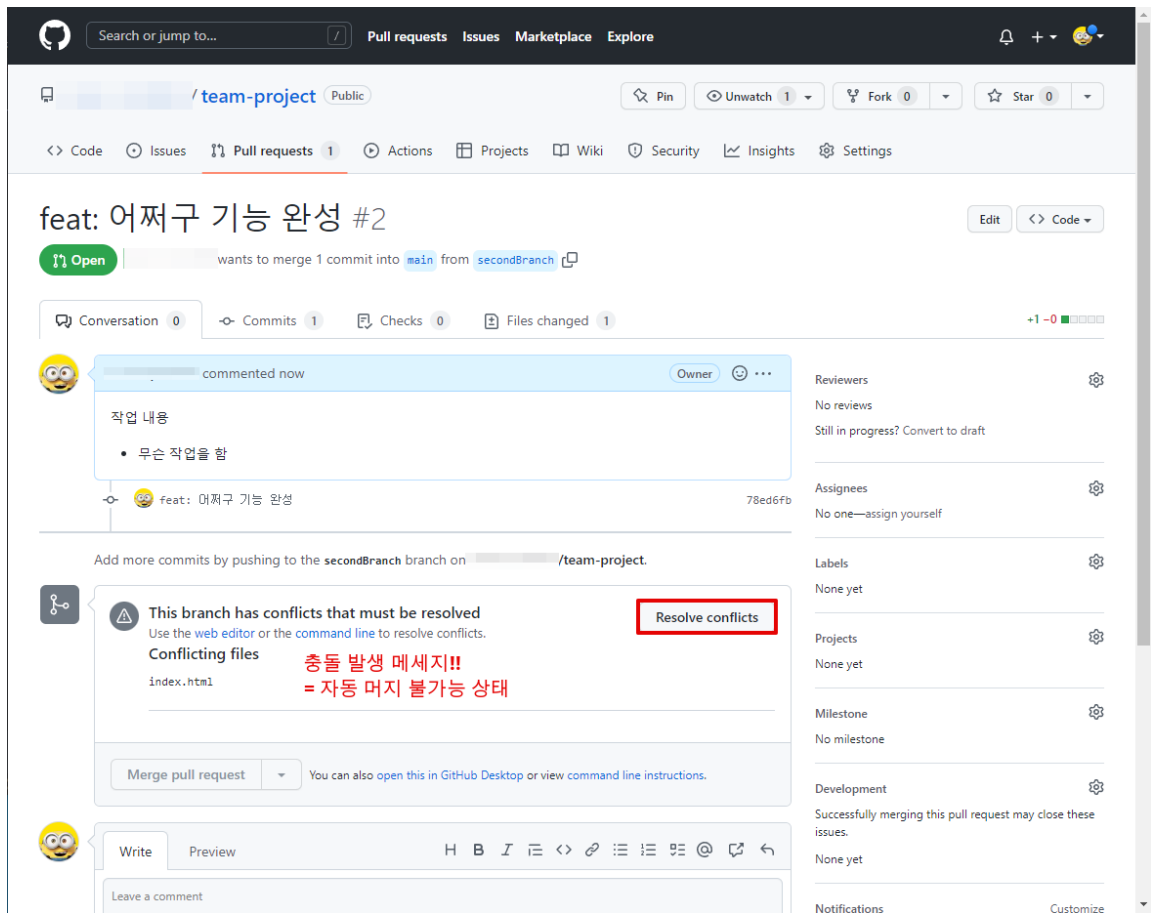
- ▼ 1. [Merge pull request] 버튼을 클릭해 Merge 하기



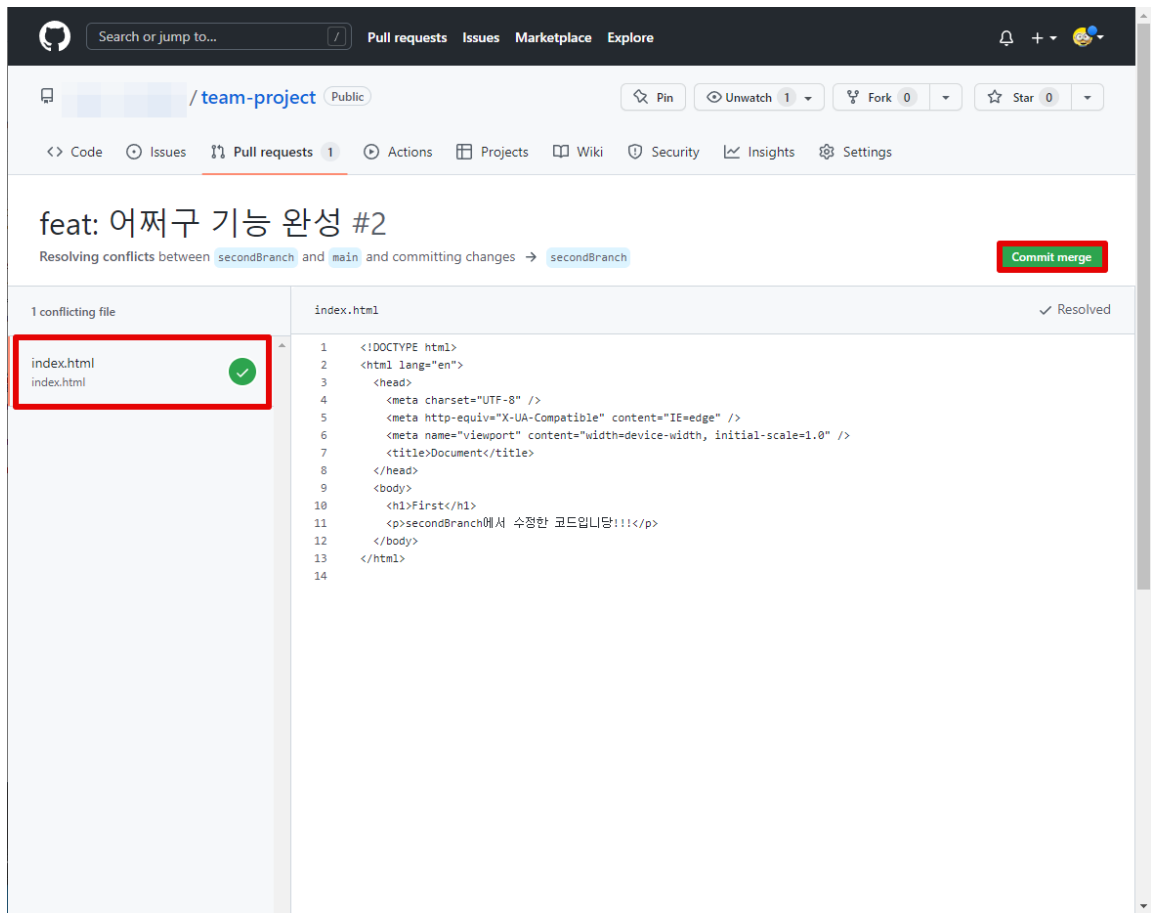
▼ 2. 만약, Code Conflict 발생시 코드 충돌을 반드시 해결

(아래 예시는 `secondBranch` 를 생성해 `main` branch와 코드 충돌을 의도적으로 발생시켜 충돌을 일으켰음)

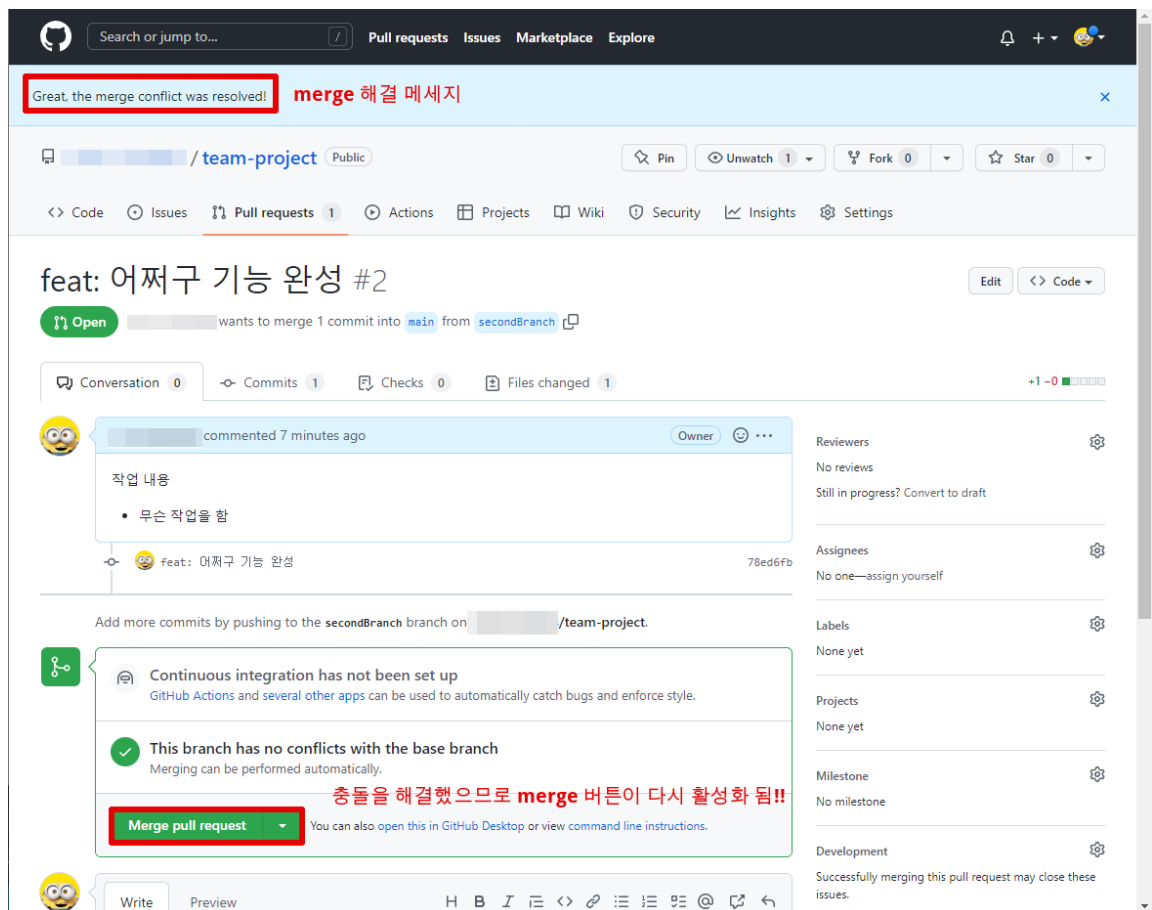
충돌 해결은 언제나 꼼꼼히! 해당 코드 관련자들이 모두 모여 코드 충돌을 해결한다.



PR에서 충돌 발생시 보이는 메시지

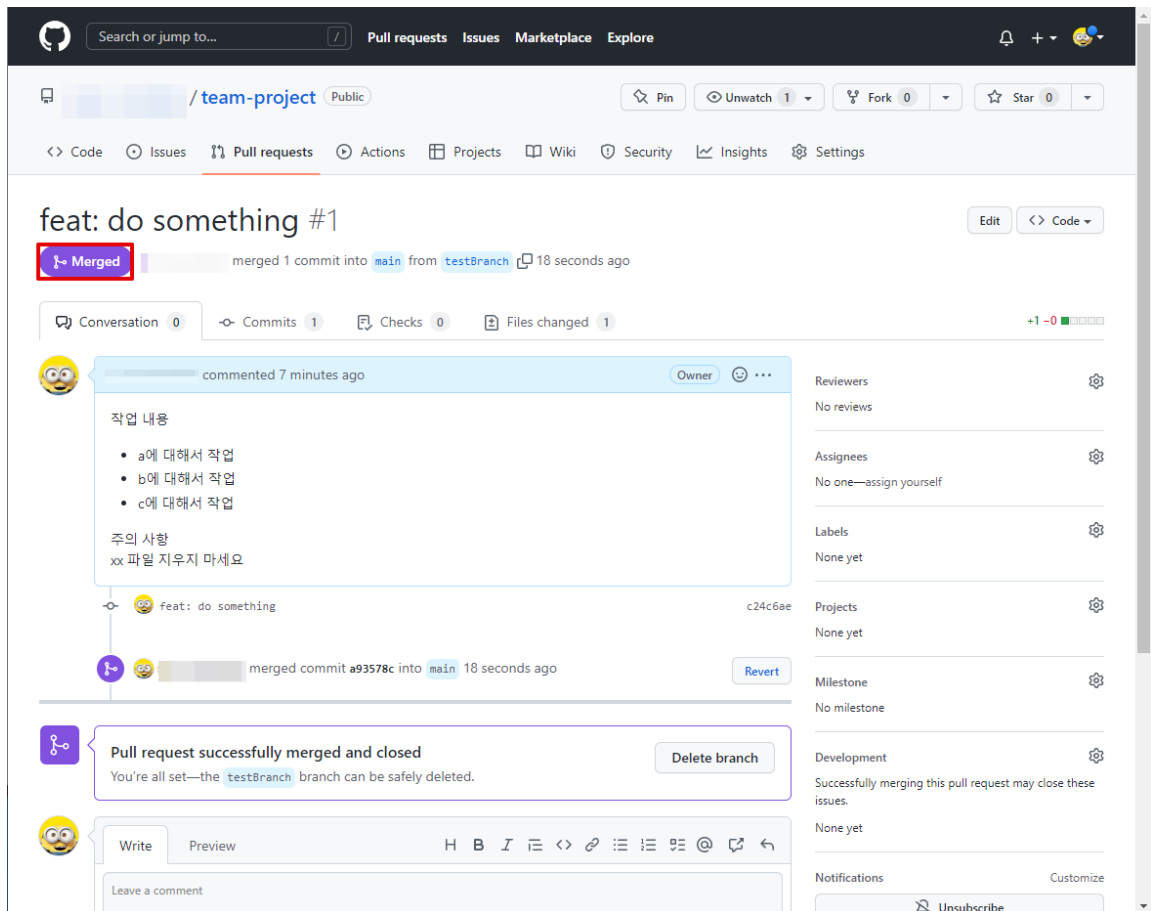


충돌 해결시 check 마크 표시됨 → [Commit merge] 버튼 클릭



충돌을 해결했으므로 [Merge pull request] 버튼 활성화

▼ 3. Code Conflict 해결 후, [Merge pull request] 버튼을 클릭해 머지 완료! Conflict 없으면 자동 머지가 됩니다!



머지 완료 화면

7. Merge 확인

지금까지 내 branch에서 작업하고, remote branch에 push 한 후, remote main으로 merge하는 것까지 마쳤습니다.

나의 local과 remote 저장소의 코드가 동기화가 되었습니다. 내 local에서 main branch로 이동한 후, 머지가 잘 되었는지 동작을 확인해봅시다.

```
> git checkout main # main branch 이동
> code . # VSCode 열어서 머지 내역 확인
```

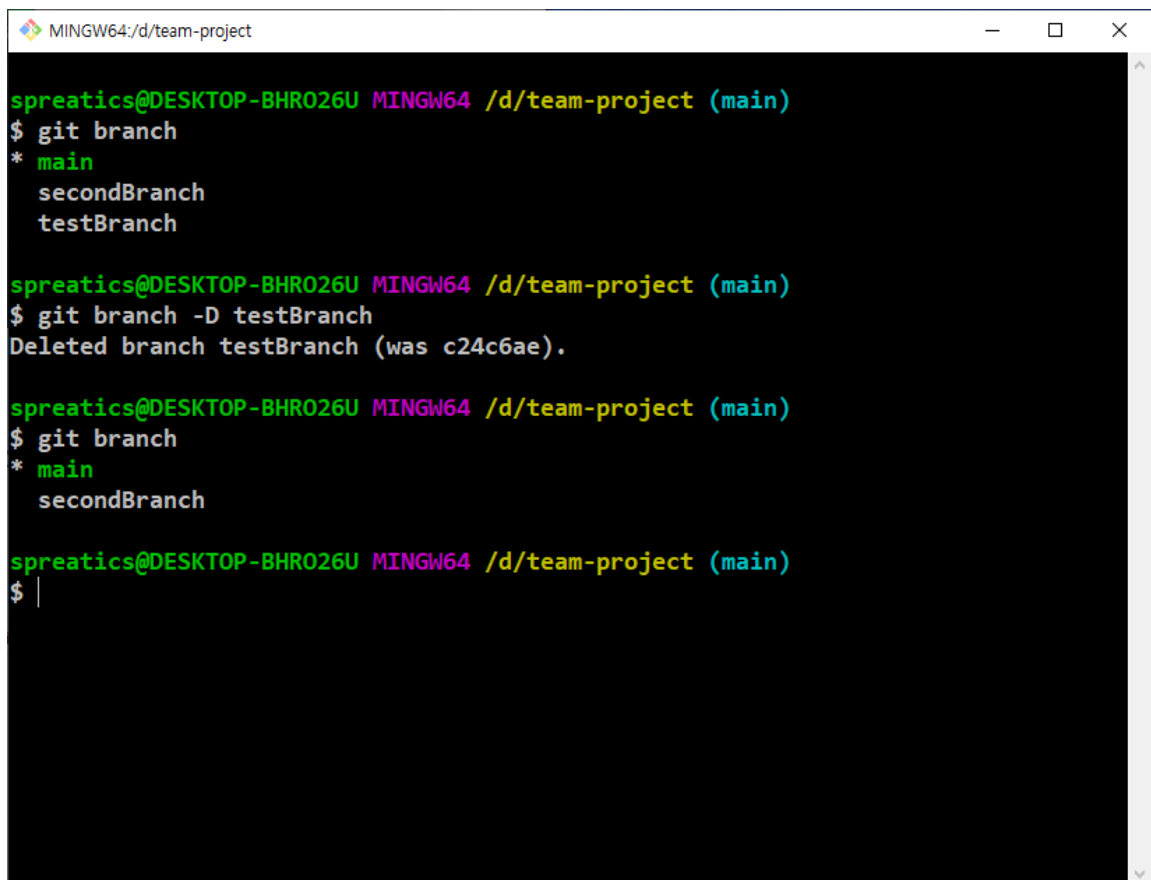
8. Local/Remote Branch 삭제

▼ Local 에서 브랜치를 삭제

더 이상 사용하지 않는 브랜치는 삭제함

(참고! 예를 들어 현재 test 브랜치에 있는 경우, test 브랜치 삭제 불가능. 다른 branch 로 이동한 후 삭제하여야 함)

```
> git checkout main # (현재 삭제하고자 하는 branch에 있다면) main branch 이동
> git branch # 현재 로컬에 존재하는 브랜치 목록 확인
> git branch -D [브랜치_이름]
> git branch # 삭제가 잘 되었는지 남있는 브랜치 목록 확인
```

A terminal window titled 'MINGW64:/d/team-project' showing a series of git commands. The user is in the 'main' branch. They run 'git branch' and see a list of branches: 'main' (current), 'secondBranch', and 'testBranch'. Then they run 'git branch -D testBranch' and receive the message 'Deleted branch testBranch (was c24c6ae)'. Finally, they run 'git branch' again and see only 'main' and 'secondBranch' listed.

```
MINGW64:/d/team-project

spreatic@DESKTOP-BHR026U MINGW64 /d/team-project (main)
$ git branch
* main
  secondBranch
  testBranch

spreatic@DESKTOP-BHR026U MINGW64 /d/team-project (main)
$ git branch -D testBranch
Deleted branch testBranch (was c24c6ae).

spreatic@DESKTOP-BHR026U MINGW64 /d/team-project (main)
$ git branch
* main
  secondBranch

spreatic@DESKTOP-BHR026U MINGW64 /d/team-project (main)
$ |
```

▼ Remote 브랜치를 삭제

로컬에서 브랜치를 삭제했다면, remote도 더 이상 사용하지 않는 브랜치를 삭제함

```
> git push origin --delete [삭제할_remote_브랜치_이름]
```

```
MINGW64:/d/team-project
spreatic@DESKTOP-BHR026U MINGW64 /d/team-project (main)
$ git branch
* main
  secondBranch
  testBranch

spreatic@DESKTOP-BHR026U MINGW64 /d/team-project (main)
$ git branch -D testBranch
Deleted branch testBranch (was c24c6ae).

spreatic@DESKTOP-BHR026U MINGW64 /d/team-project (main)
$ git branch
* main
  secondBranch

spreatic@DESKTOP-BHR026U MINGW64 /d/team-project (main)
$ git push origin --delete testBranch
To github.com:sean-spreatic/team-project.git
- [deleted]          testBranch

spreatic@DESKTOP-BHR026U MINGW64 /d/team-project (main)
$ git branch --remote
origin/main
origin/secondBranch
remote에 존재하는 브랜치 확인!!!
testBranch가 사라졌죠?

spreatic@DESKTOP-BHR026U MINGW64 /d/team-project (main)
$
```

주의 사항

- Project Repository에는 오류 코드를 올리지 않는다.
- Github repository에서 파일 삭제하지 않는다.
 - 파일을 삭제하고 싶은 경우, local에서 파일 삭제하시고, push 하시면 파일이 삭제된 변경 사항이 remote에 반영되겠죠? 😊
 - 단, 다른 팀원이 삭제하고자 하는 파일을 수정 중인지 확인해주세요.