

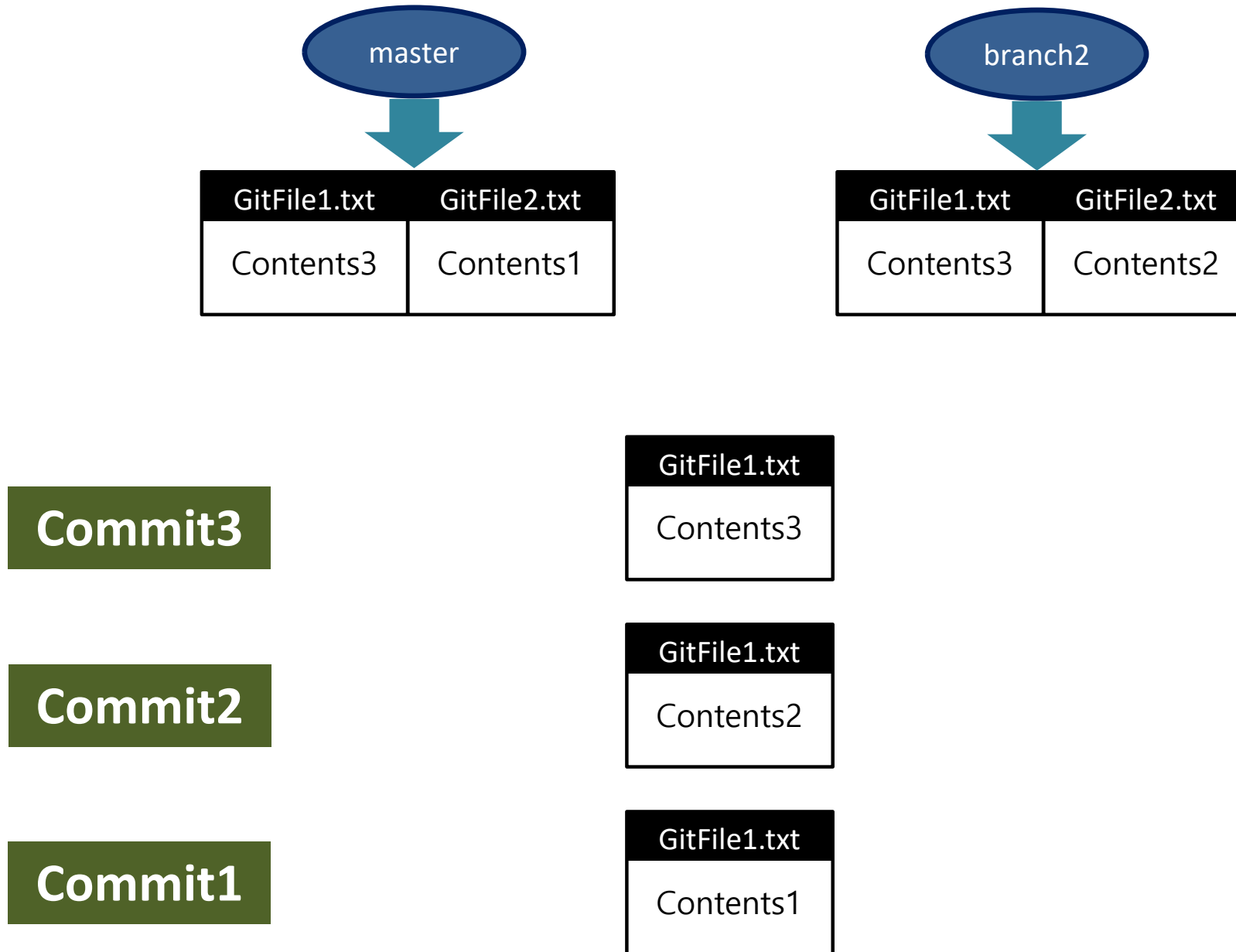
GUI 환경에서의 충돌 해결, pull request, pull

세종대학교 이은상

충돌

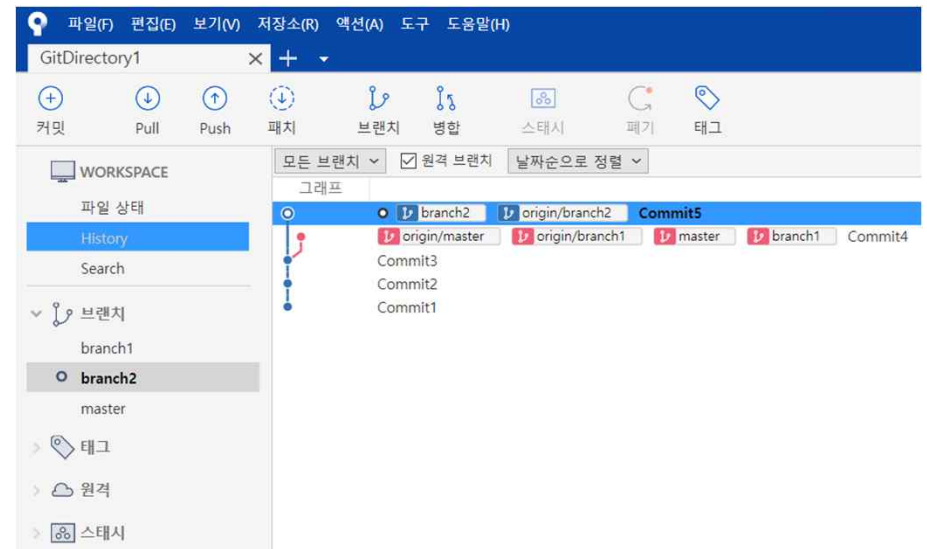
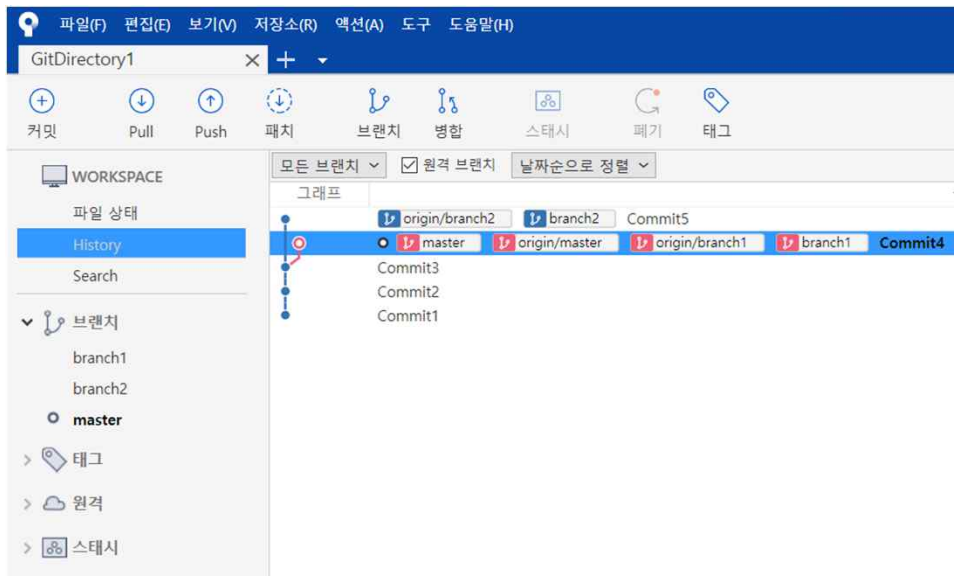
- 서로 다른 브랜치에서 동일한 부분을 다르게 수정한 후 병합을 시도할 때 충돌이 발생한다.
- 예시)
 - master 브랜치 최근 커밋은 GitFile2.txt 내용을 "Contents1"이라고 작성. branch2 브랜치 최근 커밋에선 GitFile2.txt 내용을 "Contents2"라고 작성.
 - 두 브랜치를 병합할 경우 충돌이 발생

충돌



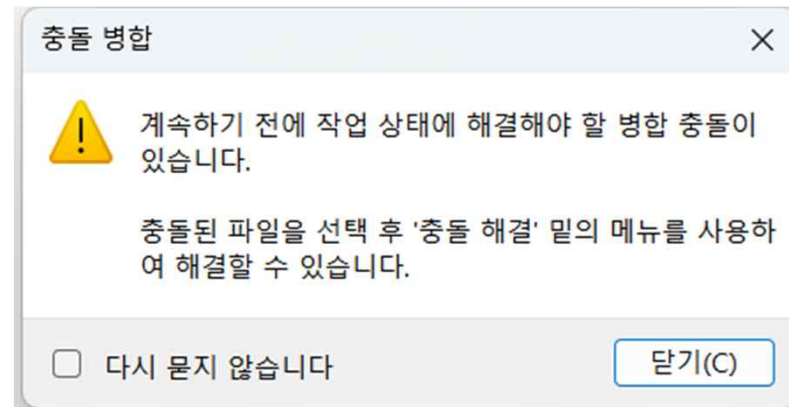
충돌 실습

- 충돌이 걱정될 때 나만 쓰는 브랜치(branch2)를 기준으로 먼저 병합하고 문제 없으면 같이 쓰는 master 브랜치에 반영하는 것이 안전함
- branch2 브랜치로 체크아웃함



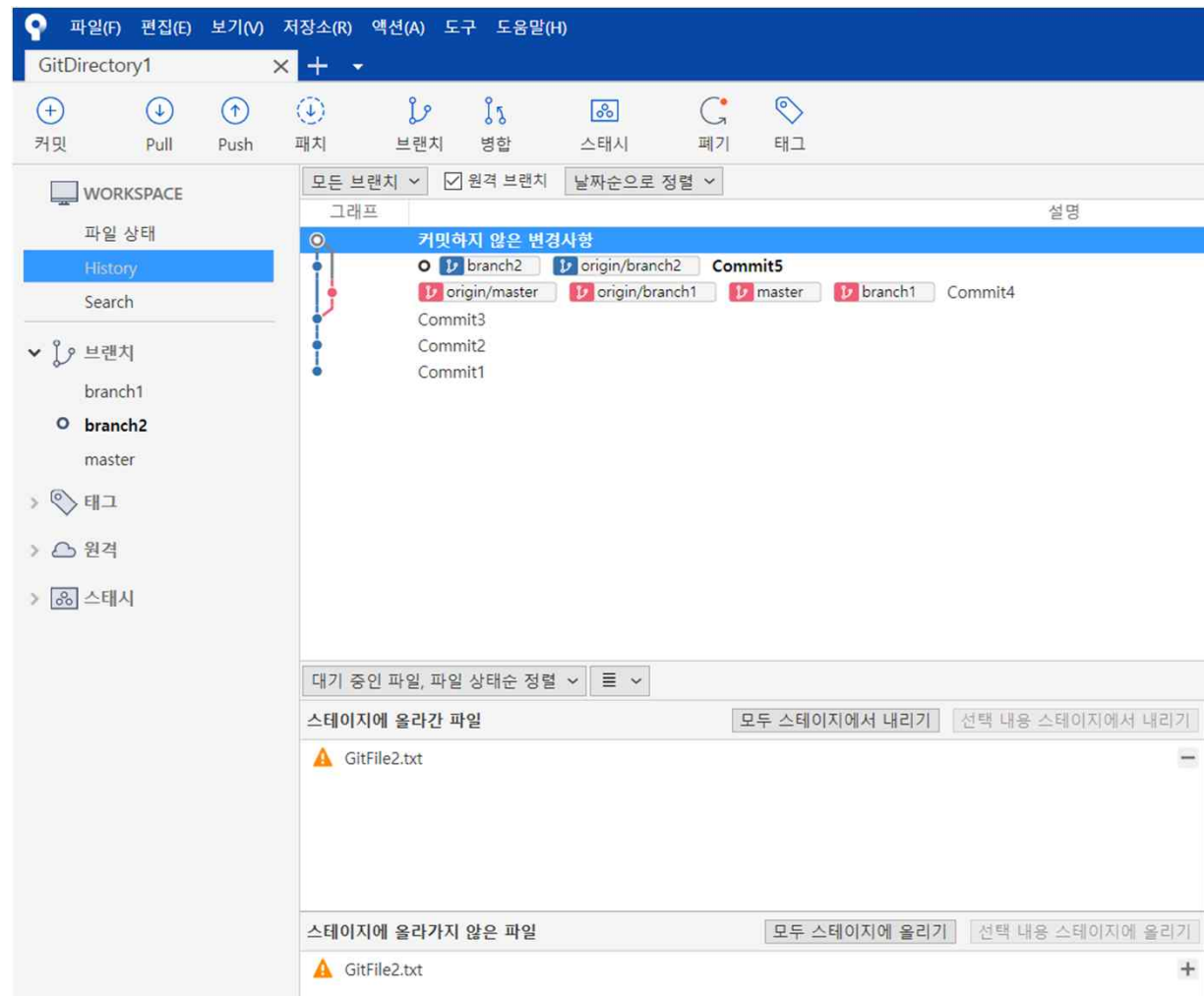
충돌 실습

- 병합하기를 원하는 커밋(master 브랜치의 최신 커밋)에서 마우스 오른쪽 버튼을 눌러 [병합] 메뉴를 선택함
- 그러면 병합하기 전에 해결해야 하는 병합 충돌이 있다고 창이 뜬다



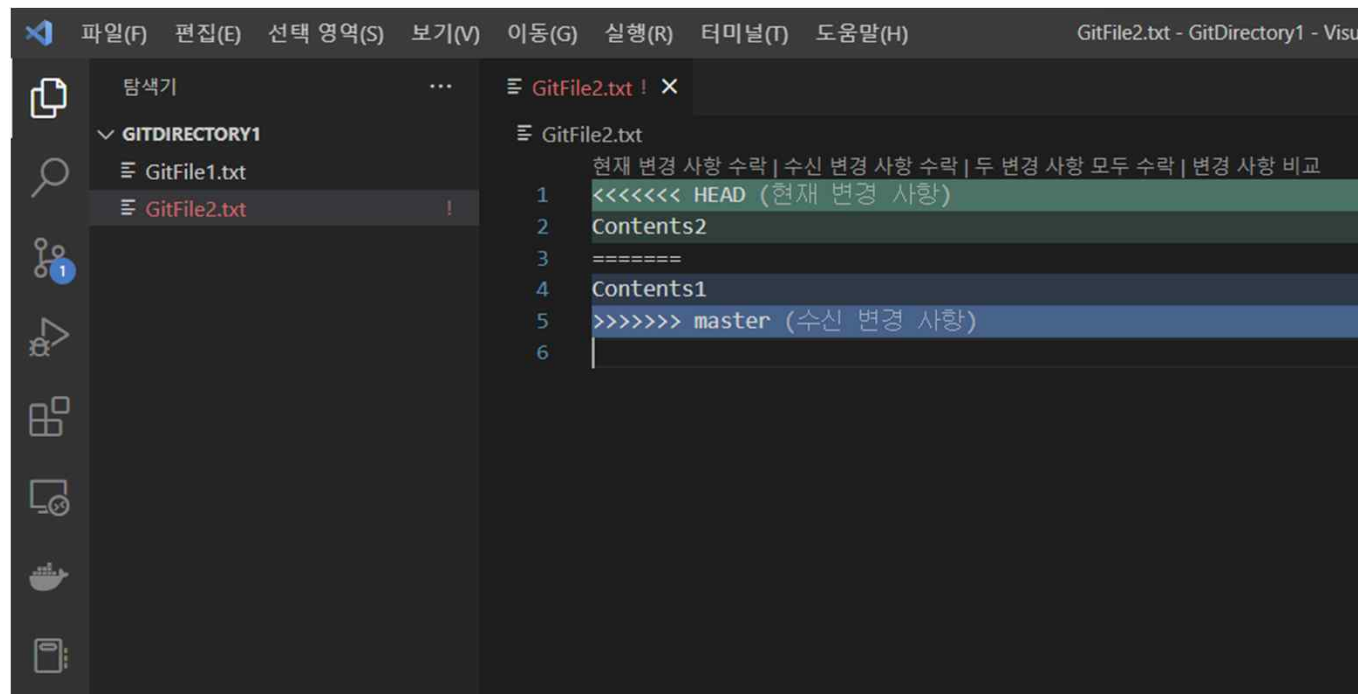
충돌 실습

- GitFile2.txt에 충돌이 일어나서 스테이지에 올라가지 않음
- 이 충돌이 난 파일을 고치면 병합을 진행할 수 있음



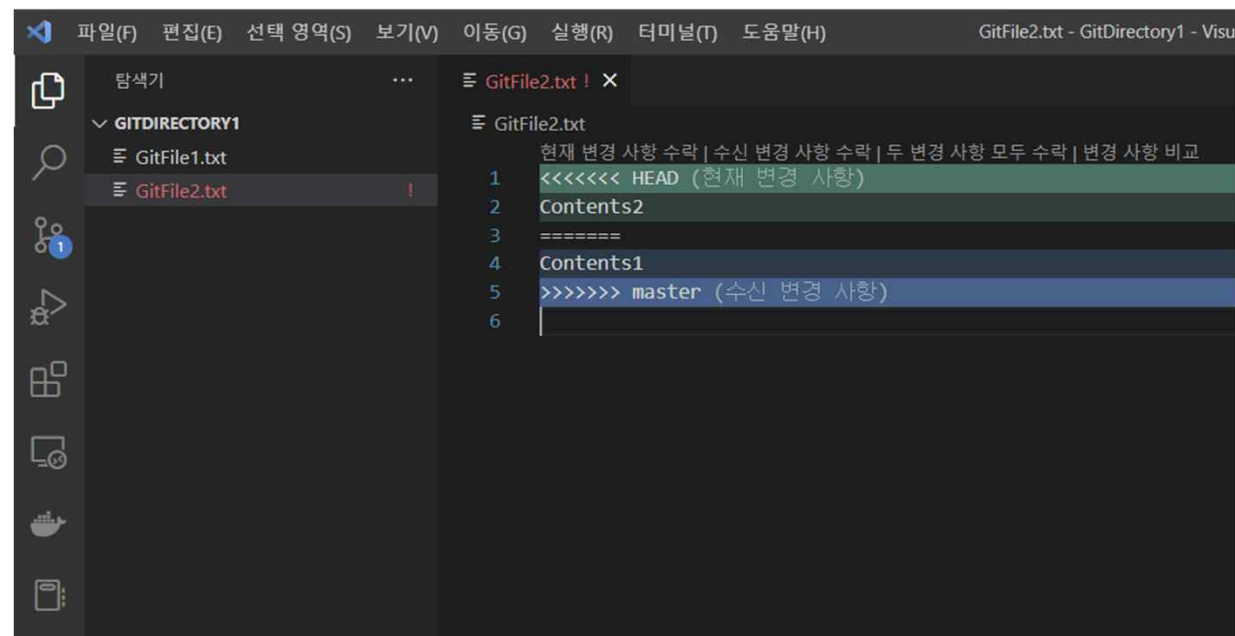
충돌 실습

- VSCode에서 충돌이 난 파일(GitFile2.txt)를 열
- Git이 충돌이 난 부분을 자동으로 표시함



충돌 실습

- 현재 변경 사항
 - 현재 변경 사항은 기준으로 한 브랜치(branch2)에서의 변경 사항
 - 예) Contents2
- 수신 변경 사항
 - 기준 브랜치에 합치려는 브랜치(master)에서의 변경 사항
 - 예) Contents1



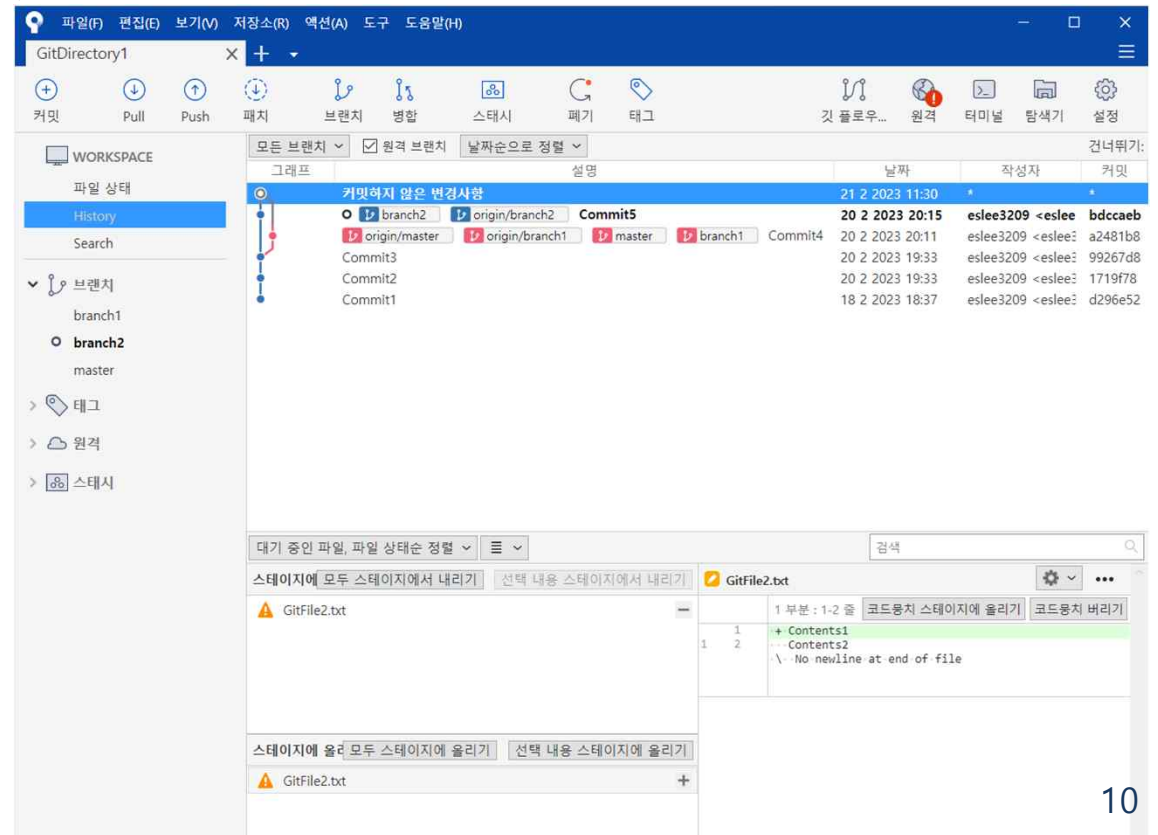
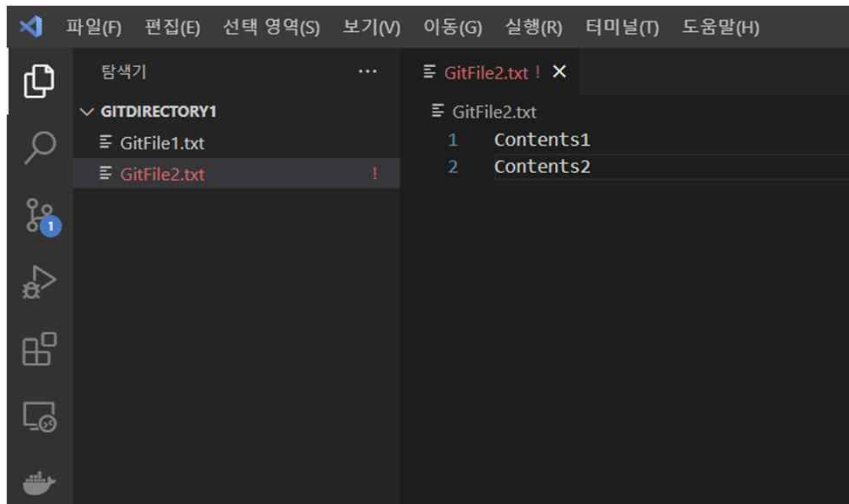
충돌 실습

- ===== 를 기준으로 위는 기준 브랜치(branch2)이며 아래는 합치려는 브랜치(master)임
- 충돌 수정하는 방법은 크게 네 가지
- 1. 현재 변경 사항 수락
- 2. 수신 변경 사항 수락
- 3. 두 변경 사항 모두 수락
- 4. 나름대로 '<<<<<<HEAD', '>>>>>master', '======'와 같은 쓸데 없는 라인 수동으로 지우고 원하는대로 수정을 해준 후 저장

```
파일(F) 편집(E) 선택 영역(S) 보기(V) 이동(G) 실행(R) 터미널(T) 도움말(H) GitFile2.txt - GitDirectory1 - Visual Studio Code
GitFile2.txt ! X
GitFile2.txt
현재 변경 사항 수락 | 수신 변경 사항 수락 | 두 변경 사항 모두 수락 | 변경 사항 비교
1 <<<<<< HEAD (현재 변경 사항)
2 Content2
3 =====
4 Content1
5 >>>>> master (수신 변경 사항)
6
```

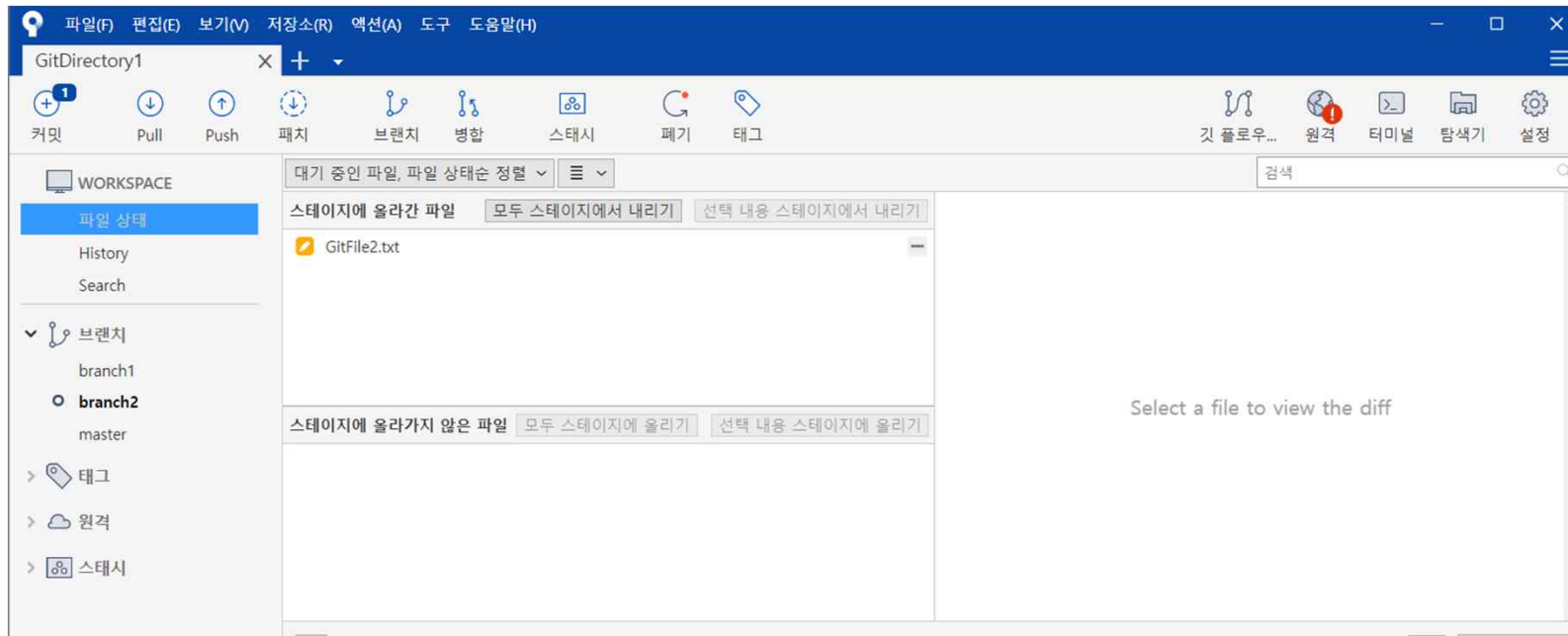
충돌 실습

- 4번의 방법으로 수정함. 다 지우고 Contents1, Contents2를 순서대로 입력하고 저장
- 소스트리에 돌아가서 GitFile2.txt(스테이지에 올라가지 않은 파일)을 선택해서 오른쪽 미리보기 창을 보면 충돌이 해결된 것을 알 수 있음



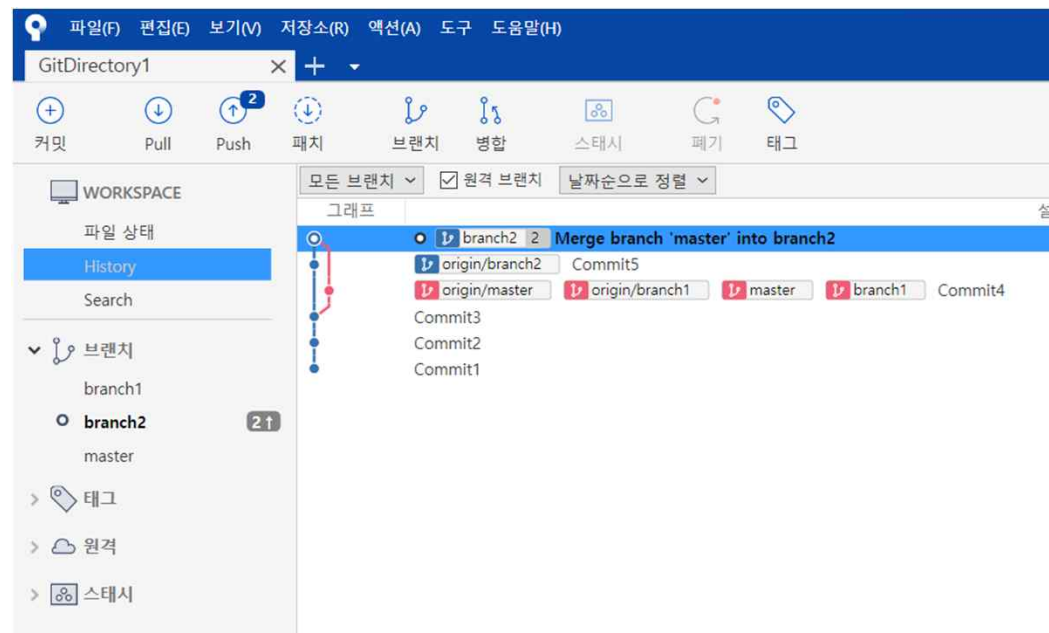
충돌 실습

- 이제 GitFile2.txt의 [+] 버튼을 클릭해서 스테이지 위로 add함.
- 이제 병합을 위해 [커밋] 버튼을 누름. 아래 창 영문 메시지가 자동으로 들어감. 그대로 두고 커밋 버튼을 누름.



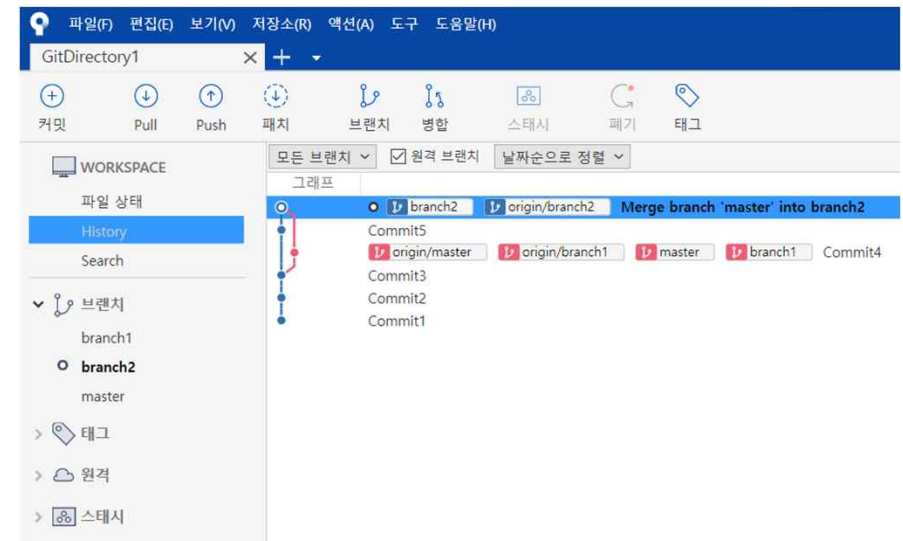
충돌 실습

- 그러면 성공적으로 병합이 됨



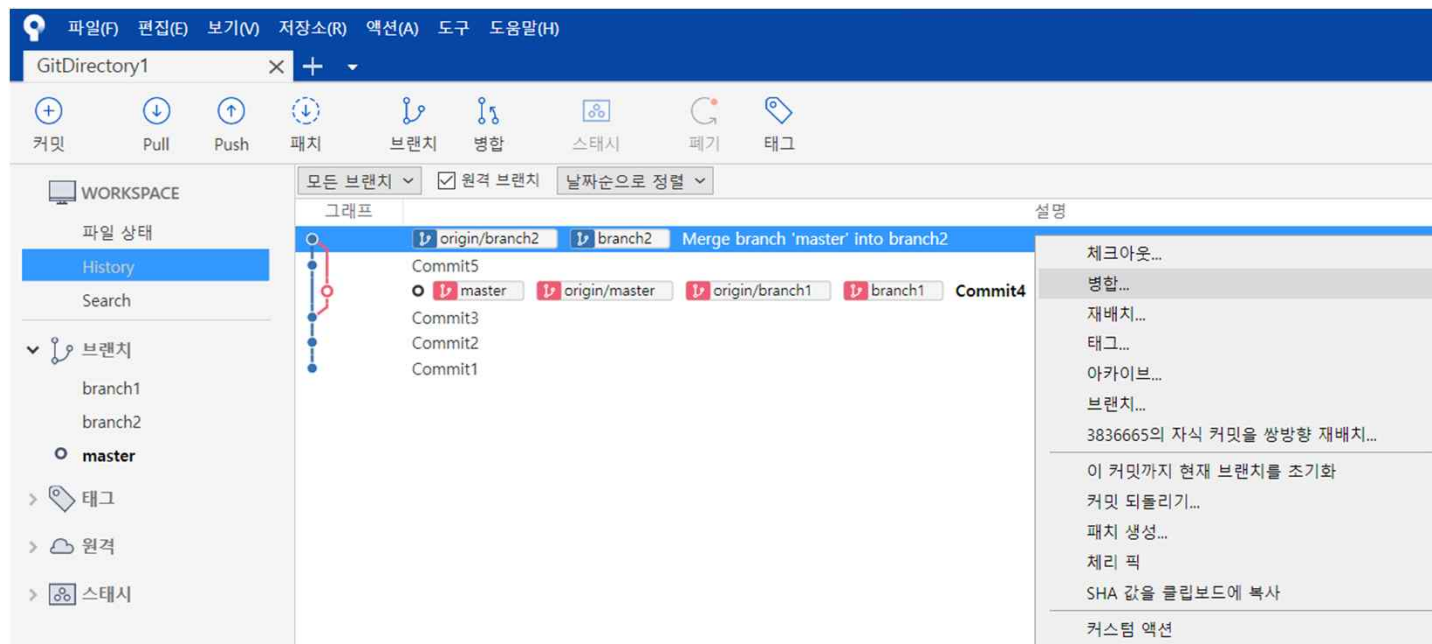
충돌 실습

- Push를 눌러 원격저장소에도 반영함



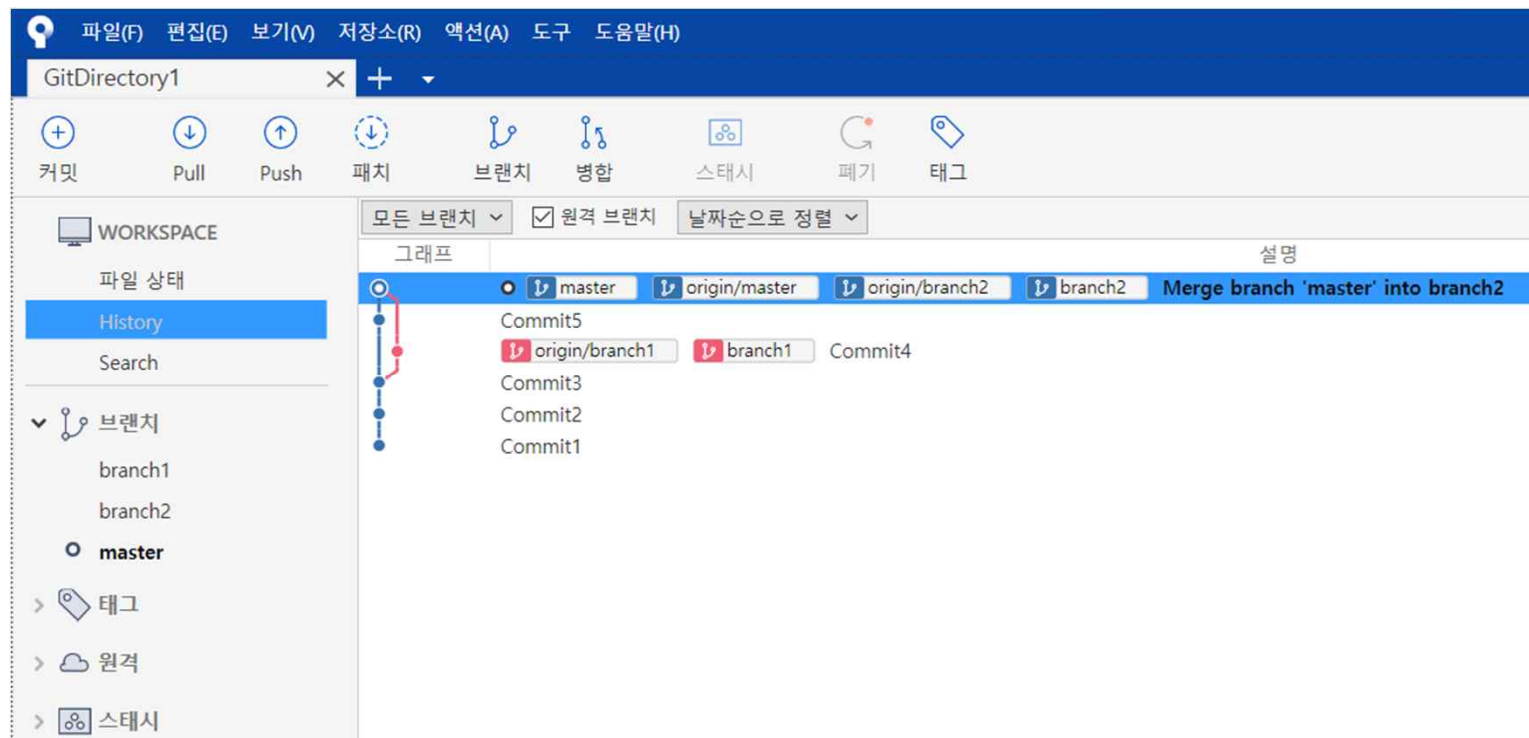
충돌 실습

- 마지막으로 branch2에 병합한 것을 master 브랜치에도 반영함
- master 브랜치를 기준으로 병합하기 위해 master 브랜치로 체크아웃함
- branch2에 마우스 오른쪽 클릭한 후 [병합] 선택
- 이후 원격저장소에 Push함



충돌 실습

- 병합한 커밋이 성공적으로 master 브랜치에도 반영이 됨

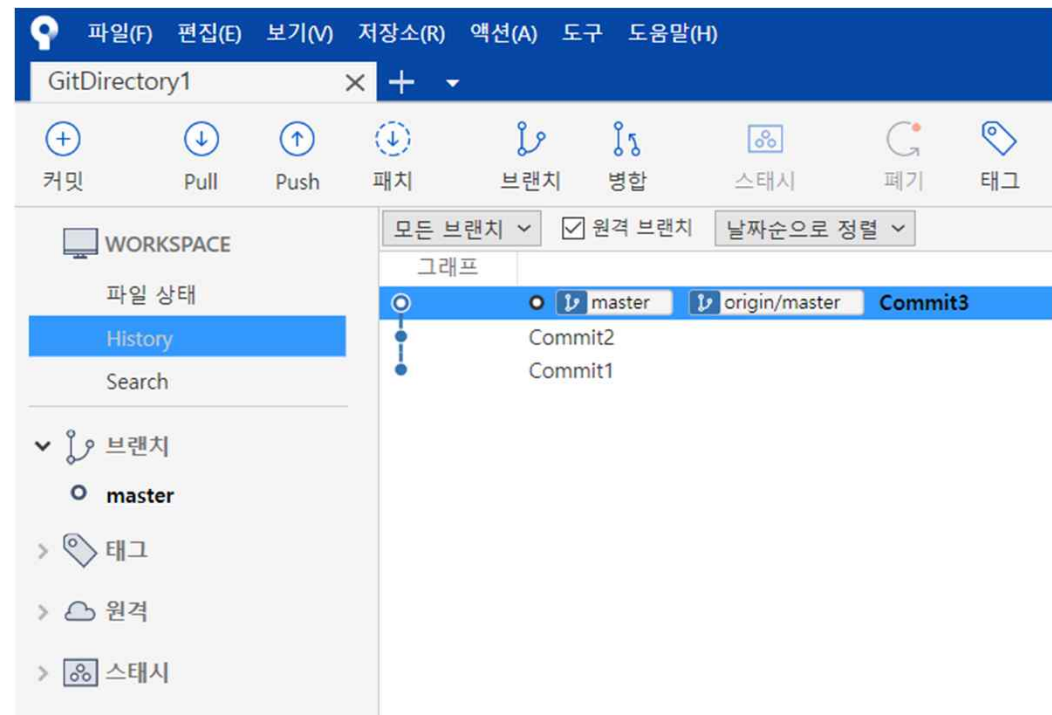
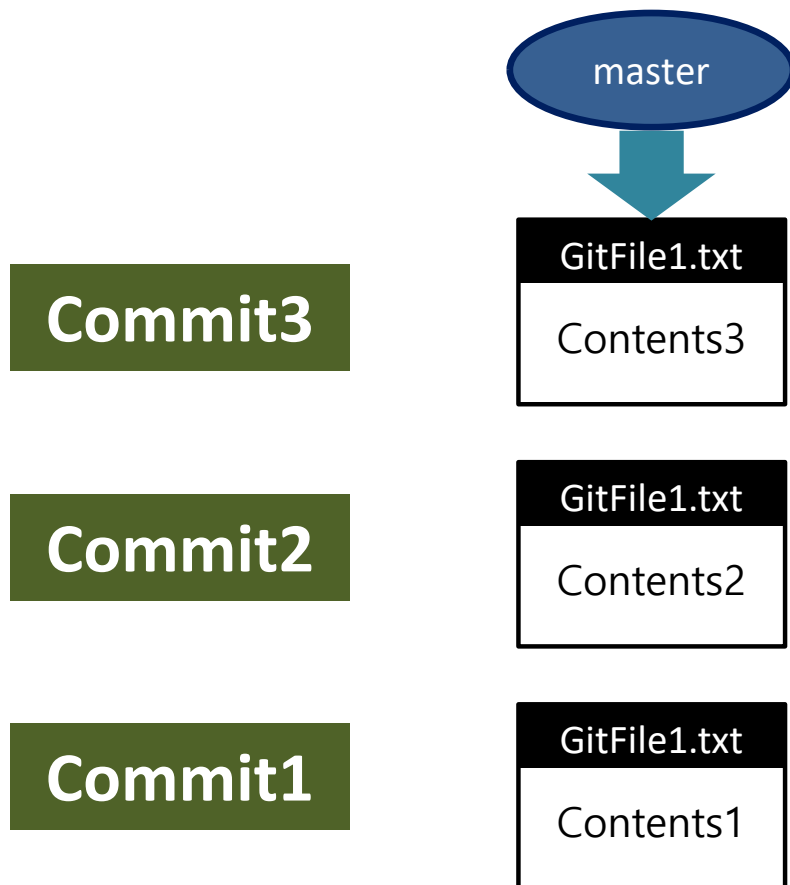


Pull Request

- 두 협력자가 master 브랜치에는 완벽한 코드만 두기로 약속하고, 각자 자신의 브랜치(예, branch1, branch2)에서 작업한 후에 완성이 되면 master 브랜치에 합친다고 함
- 내가 이 브랜치에서 뭘 바꿨는지 협력자가 확인할 수 있는 과정을 거치는 것이 좋음. 이 때, 사용하는 것이 pull request임
- Pull request는 협력자에게 브랜치 병합을 요청하는 메시지를 보내는 것임. "A브랜치로 B브랜치를 병합해도 될까요? 수정사항은 다음과 같습니다" 라고 메시지를 예의있게 보내는 것임

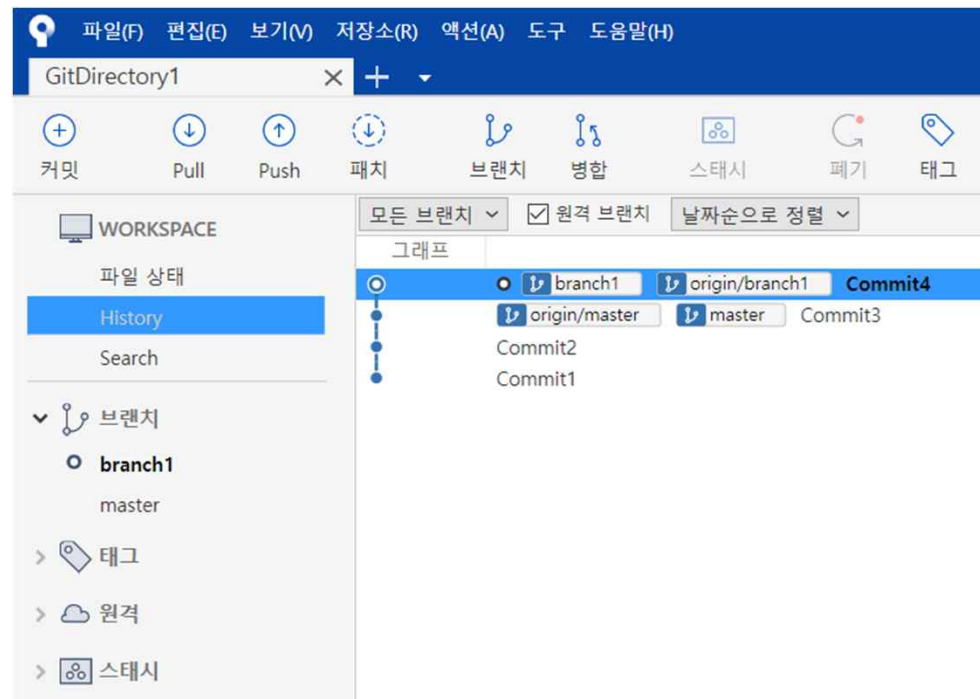
Pull Request 실습

- 아래 상황에서 실습을 한다고 함. master 브랜치가 가장 최근 커밋을 가리키고 있는 상황



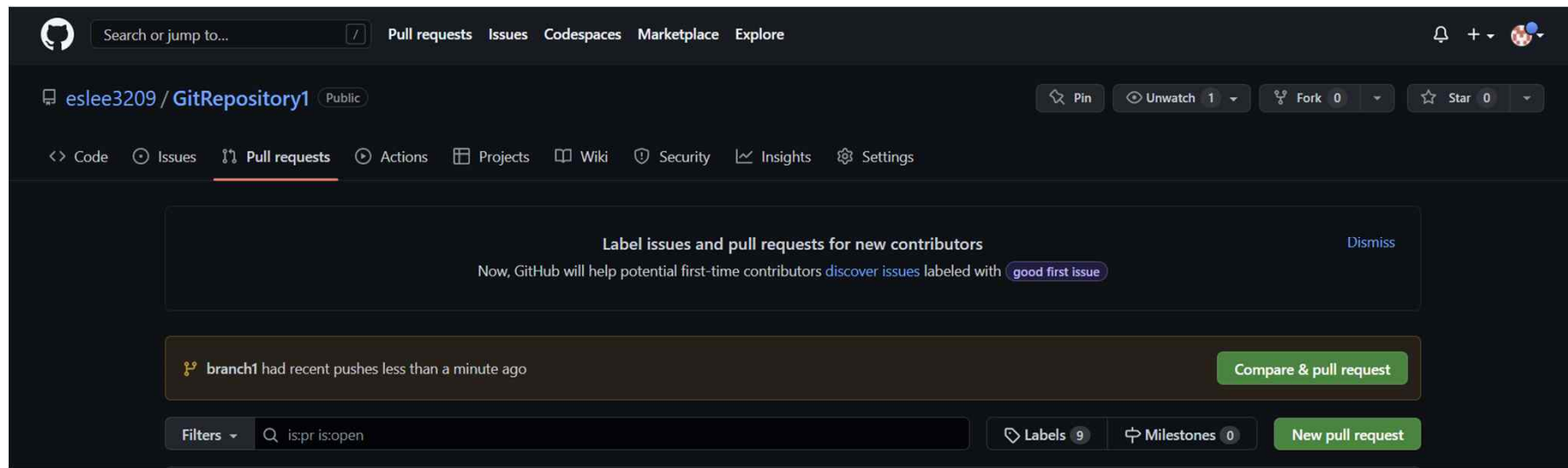
Pull Request 실습

- branch1이라는 브랜치를 생성함
- VSCode 통해 새 파일을 생성하고 커밋/푸시함
 - 예) 파일명: GitFile2.txt
 - 내용: Contents1
 - 커밋 메시지: Commit4



Pull Request 실습

- 이제 GitHub의 해당 원격저장소로 들어간다.
 - 예) GitRepository1 원격저장소
- 방금 푸시한 브랜치가 있으면 github에서는 친절하게 방금 푸시한 브랜치(branch1)를 알려줌.
- [Compare & pull request] 버튼은 pull request를 보낼 수 있는 버튼임
- "최근에 이 브랜치에 코드를 업데이트했으니 협력자에게 pull request를 보내려면 이 버튼을 클릭하라"는 뜻임



Pull Request 실습


- [Compare & pull request] 버튼을 클릭함
- 정중하게 병합을 요청할 수 있는 메시지를 적을 페이지가 나옴
- 여기서 베이스 브랜치와 비교 브랜치를 설정해야함
- 베이스 브랜치는 병합 결과물이 올라갈, 즉, 기준이 되는 브랜치임
- 비교 브랜치는 현재 기준 브랜치의 비교대상이 되는 브랜치임
- 예) base: master, compare: branch1


Pull Request 실습

- pull requests 제목과 메시지를 입력함
 - 예) pull requests 제목: PullRequests1
 - pull requests 내용: PullRequestsContents1
- 추가한 커밋 등 이상 없는지 확인한 후 검토가 끝나면 [Create pull request] 버튼을 클릭함


Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

 base: master

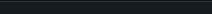
 compare: branch1

✓ Able to merge. These branches can be automatically merged.

 PullRequests1

Write

Preview



PullRequestsContents1

Attach files by dragging & dropping, selecting or pasting them.

Create pull request

Reviewers

No reviews

Assignees

No one—assign yourself

Labels

None yet

Projects

None yet

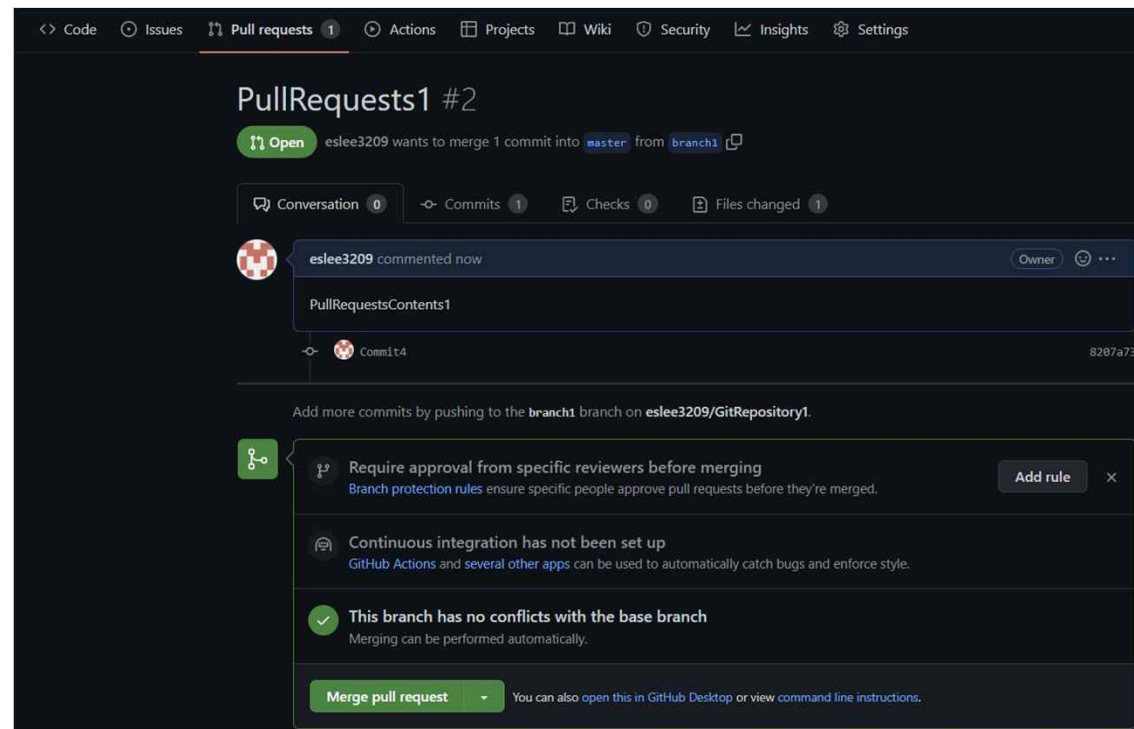
Milestone

No milestone

Development

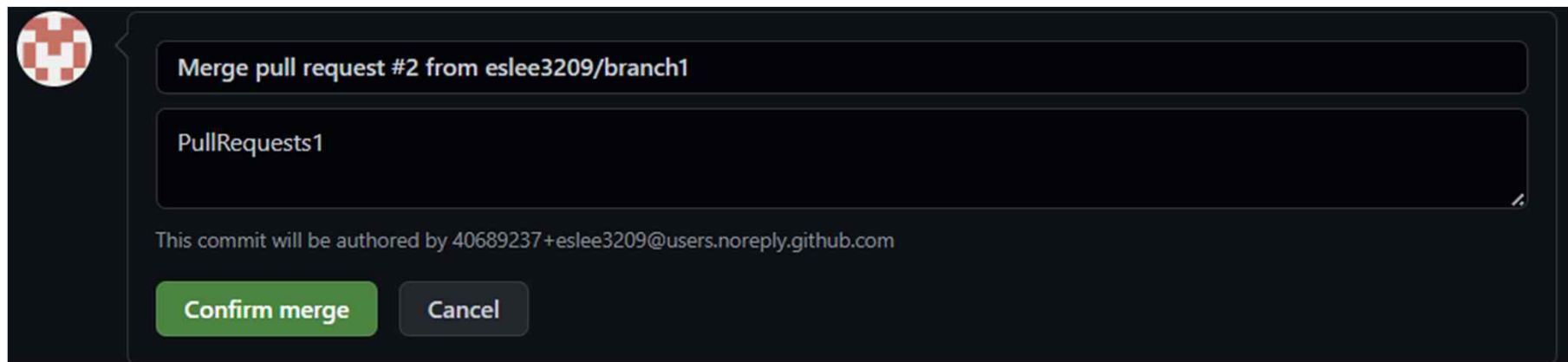
Pull Request 실습

- 그러면 방금 만든 pull request가 보임
- 이제 협력자가 이 pull request를 확인하고 새로 추가된 코드를 검토할 수 있음
- 코드의 라인마다 댓글을 달 수 있어 해당 코드가 왜 고쳐졌는지, 어떻게 개선할지 등을 pull requests에서 토론할 수 있음
- 이 pull request를 수락하거나 수정 요청 혹은 병합할 수 있음



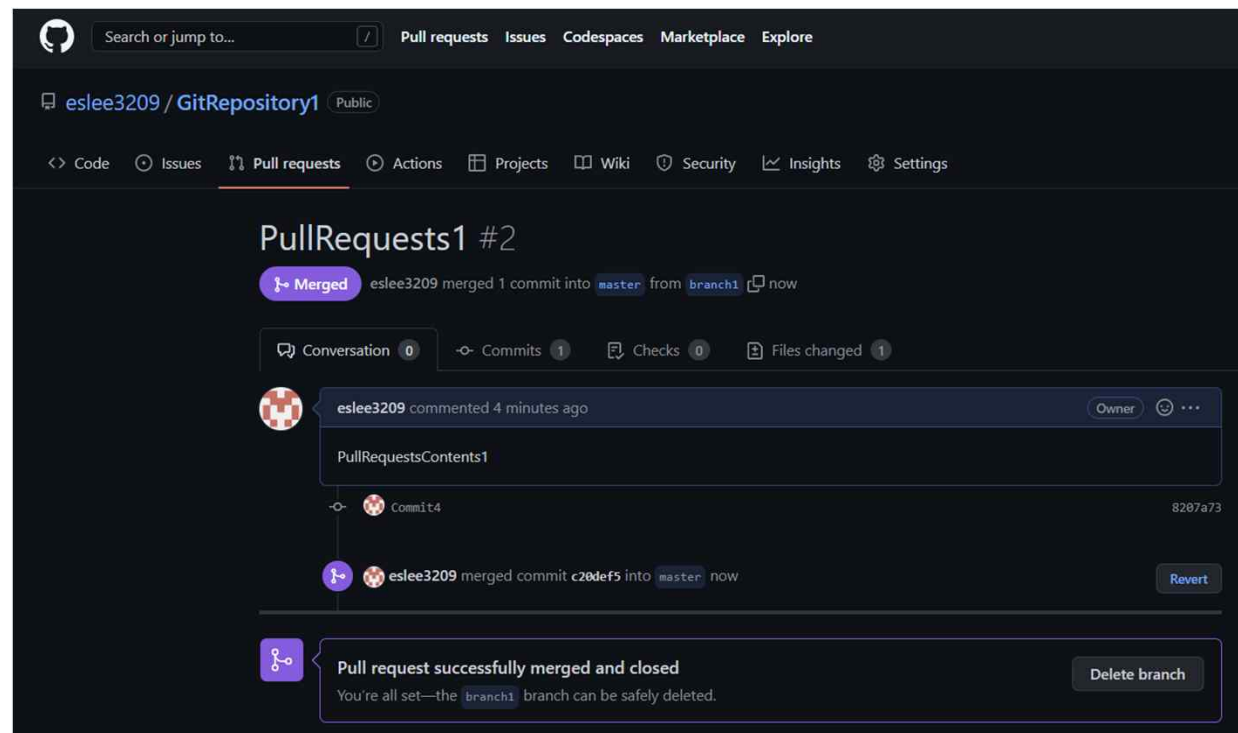
Pull Request 실습

- [Merge pull request]를 클릭해보면 병합 커밋을 만들 수 있는 입력 창이 보임
- [Confirm merge] 버튼을 클릭함



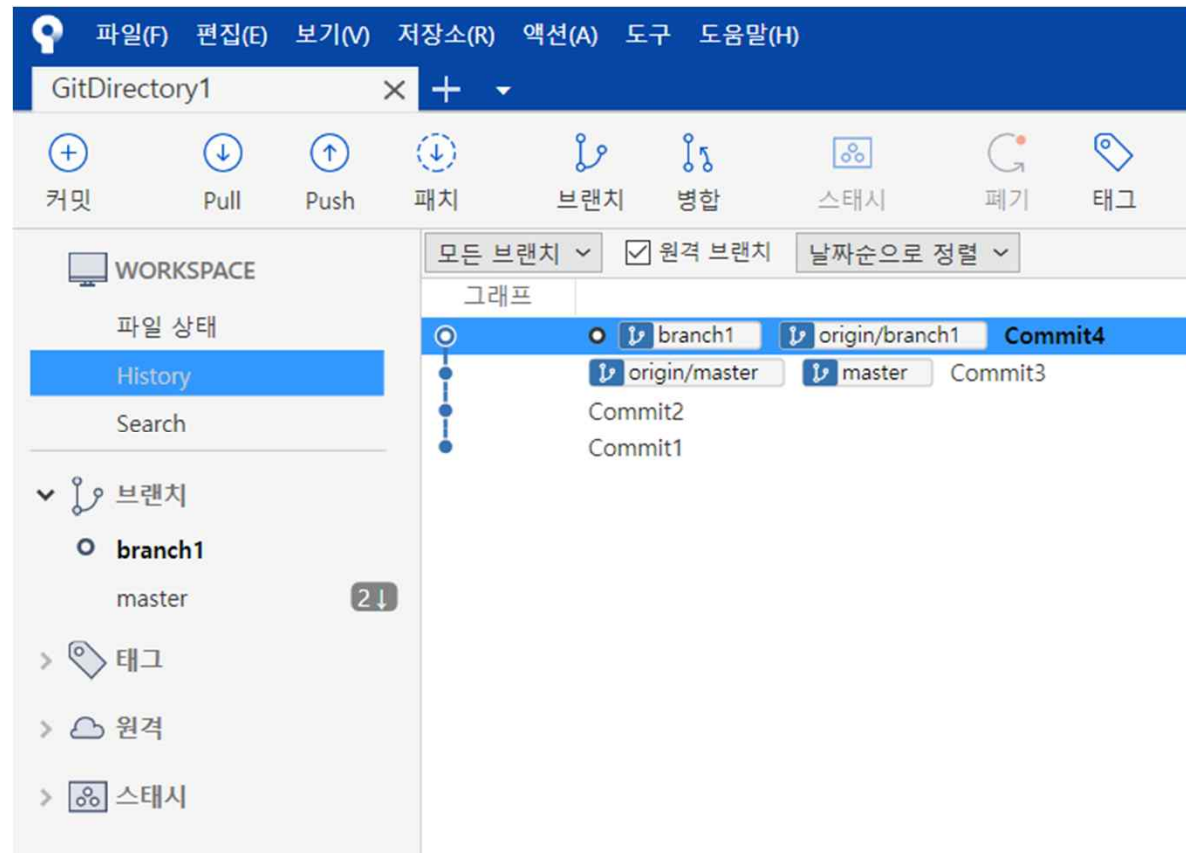
Pull Request 실습

- 성공적으로 병합이 이루어짐. 즉, 예의바른 병합이 완료됨. "Merged"라고 표시되면 잘 병합된 것임
- pull request에 있던 숫자 1은 없어짐



Pull Request 실습

- 소스트리를 보면 아직 원격저장소의 origin/master 브랜치는 옛날 커밋을 가리키고 있음
- git에서 새로운 이력을 업데이트해주어야함

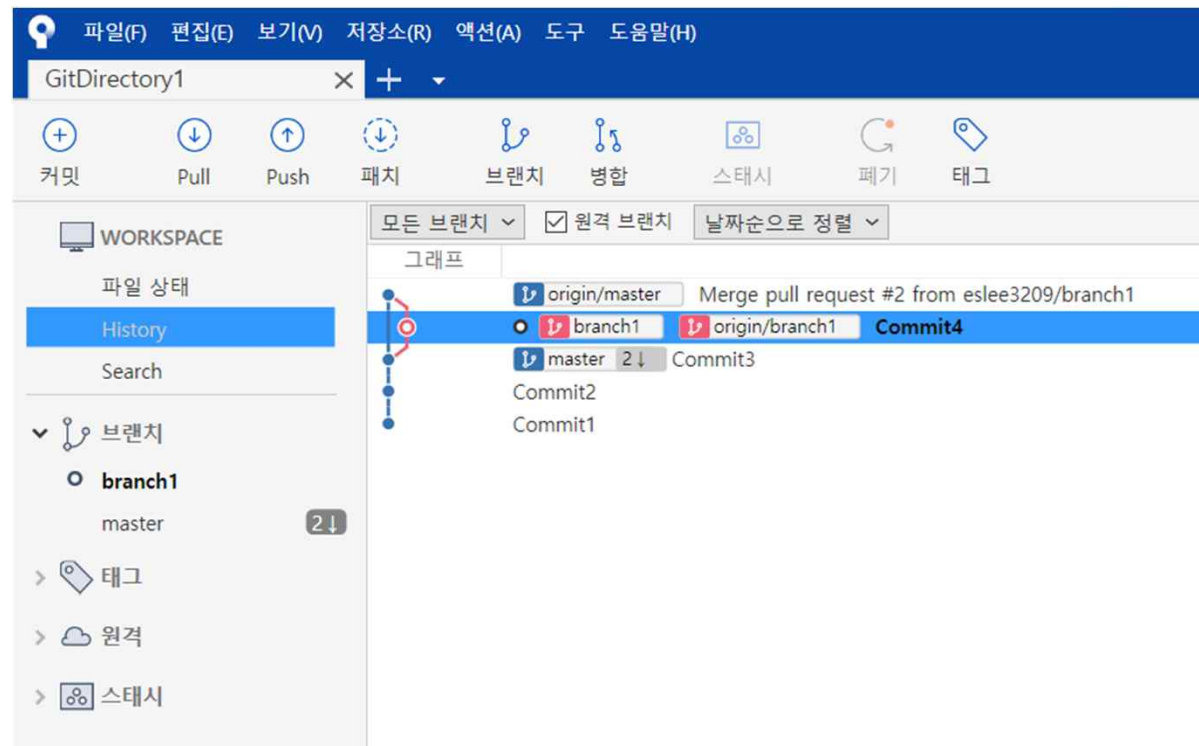


패치

- 패치는 간단히 말하면 새로고침 기능임. Git에서 새로운 이력을 업데이트하는 명령임. 특히, 원격저장소에서 변경된 사항을 업데이트하는 것임
- Pull은 실제 로컬저장소에 코드를 내려받는 데에 비해 패치는 그래프만 업데이트함

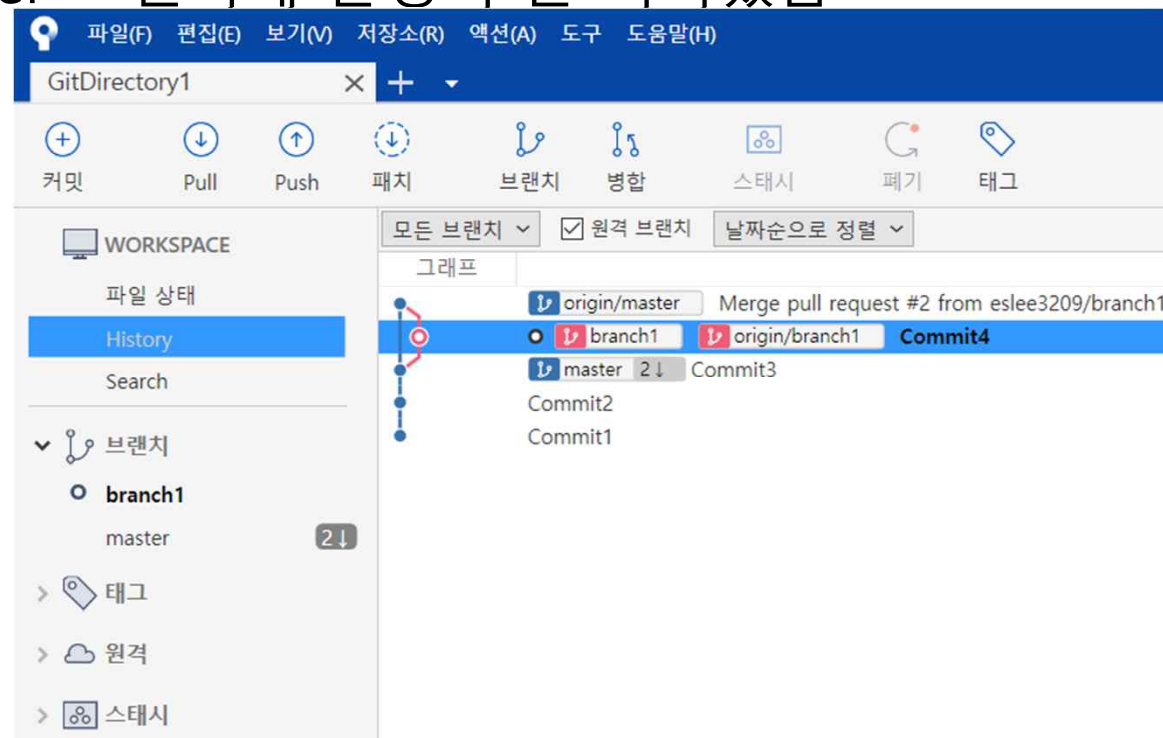
Pull Request 실습

- 패치를 하기 위해서는 소스트리 상단 [패치] 아이콘 누르고 팝업 창에서 [확인] 버튼을 클릭함. 그러면 업데이트가 됨



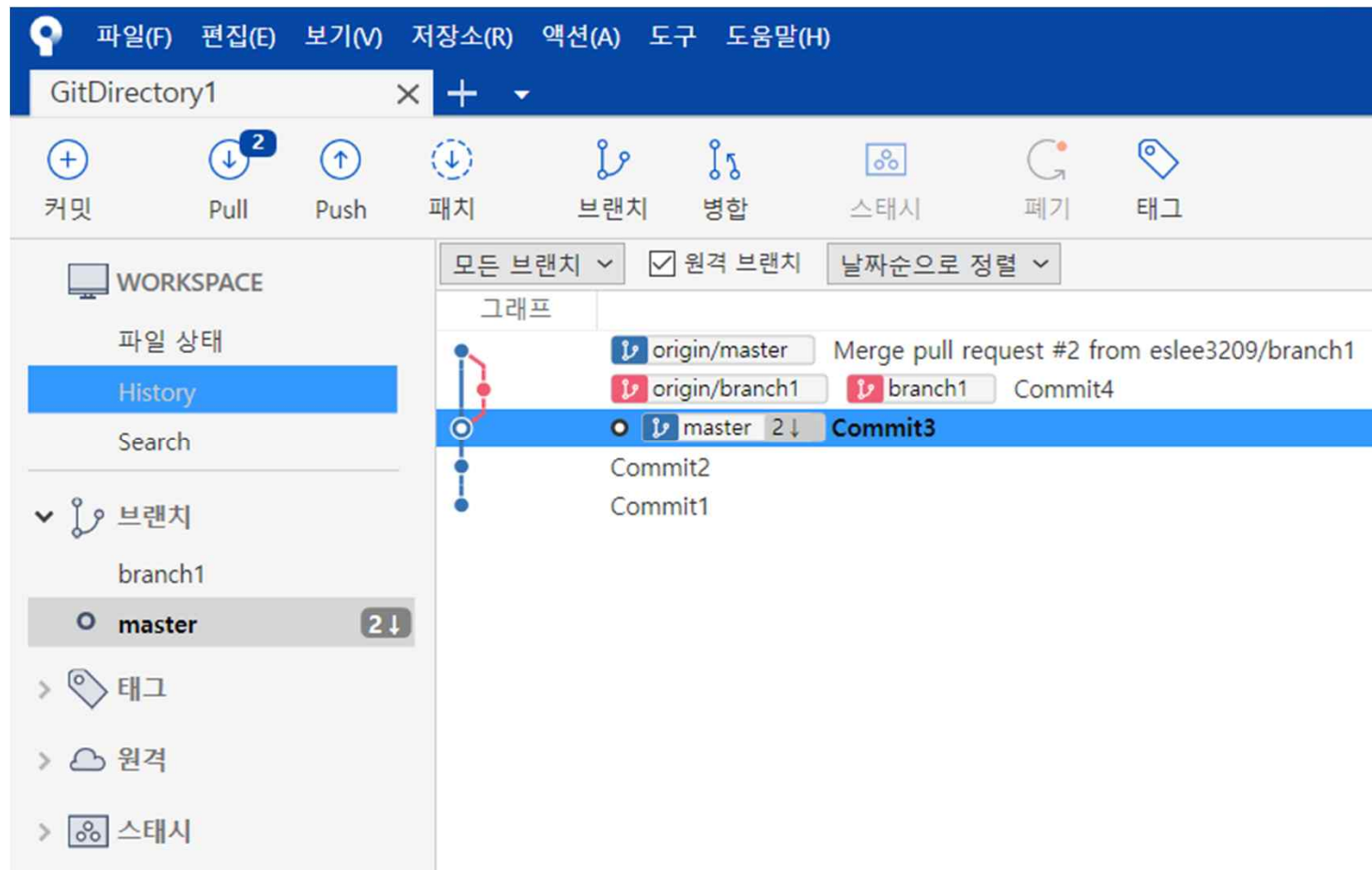
Pull

- 원격저장소는 최근 커밋이 업데이트 되었지만 로컬 폴더에서는 반영이 안 되어있는 경우가 있음
- 이 경우 pull을 하여 원격저장소의 새로운 커밋을 로컬저장소에 갱신해야함
- 사실 pull은 패치와 병합의 합성임
- 현재 실습 상황에서도 원격저장소의 master 최신 커밋이 로컬저장소의 master 브랜치에 반영이 안 되어있음



Pull 방법

- 먼저 pull할 브랜치로 체크아웃함. 즉, 원격저장소의 브랜치에 비해 로컬저장소 브랜치는 최신이 아닌 브랜치로
- 본 실습에서는 master 브랜치로 체크아웃함



Pull 방법

- 소스트리 상단의 [Pull] 아이콘을 클릭하고 [확인]을 누름.
그러면 pull이 완료됨
- master와 origin/master 모두 "Merge pull request ..." 커밋을 가리킴

