



KH_2502_I CLASS



vcs

VCS(Version Control System)

- 파일 변화를 시간에 따라 기록했다가 이후 특정 시점의 버전을 다시 꺼내올 수 있는 시스템
- 동일한 정보에 대한 여러 버전을 관리
- 버전을 통해 시간적으로 변경 사항과 변경 사항을 작성한 작업자 추적 가능

파일 이름을 통한 관리



파일 및 폴더를 복사하여 관리



CVS



Centralized Version Control



DVC



Distributed Version Control



CVS(Centralized Version Control)

- 서버에서 버전의 히스토리를 관리
- 각 개발자들이 원하는 내용을 서버에 업데이트하여 즉각적으로 동기화
- 서버에 문제가 발생하는 경우 다른 개발자들 또한 업무가 불가능



DVC(Distributed Version Control)

- 서버뿐만 아니라 각 개발자들이 동일한 버전 히스토리를 가지고 있는 구조
- 문제 발생 시 서버에 문제가 발생하더라도 복원이 가능하며, 업무 가능



Git / GitHub

Git



- 로컬저장소의버전관리시스템
- 소스코드나파일버전을관리하는시스템
- 개발자의작업컴퓨터에있는버전관리시스템

GitHub



- 클라우드저장소를저장하는형태의버전관리시스템
- 공개저장소(public)와비공개저장소(private)로구분

여럿이 협업하는 경우 각각 개인의 Git을 가지고 있고 하나의 GitHub를 통해서 관리



Git / GitHub



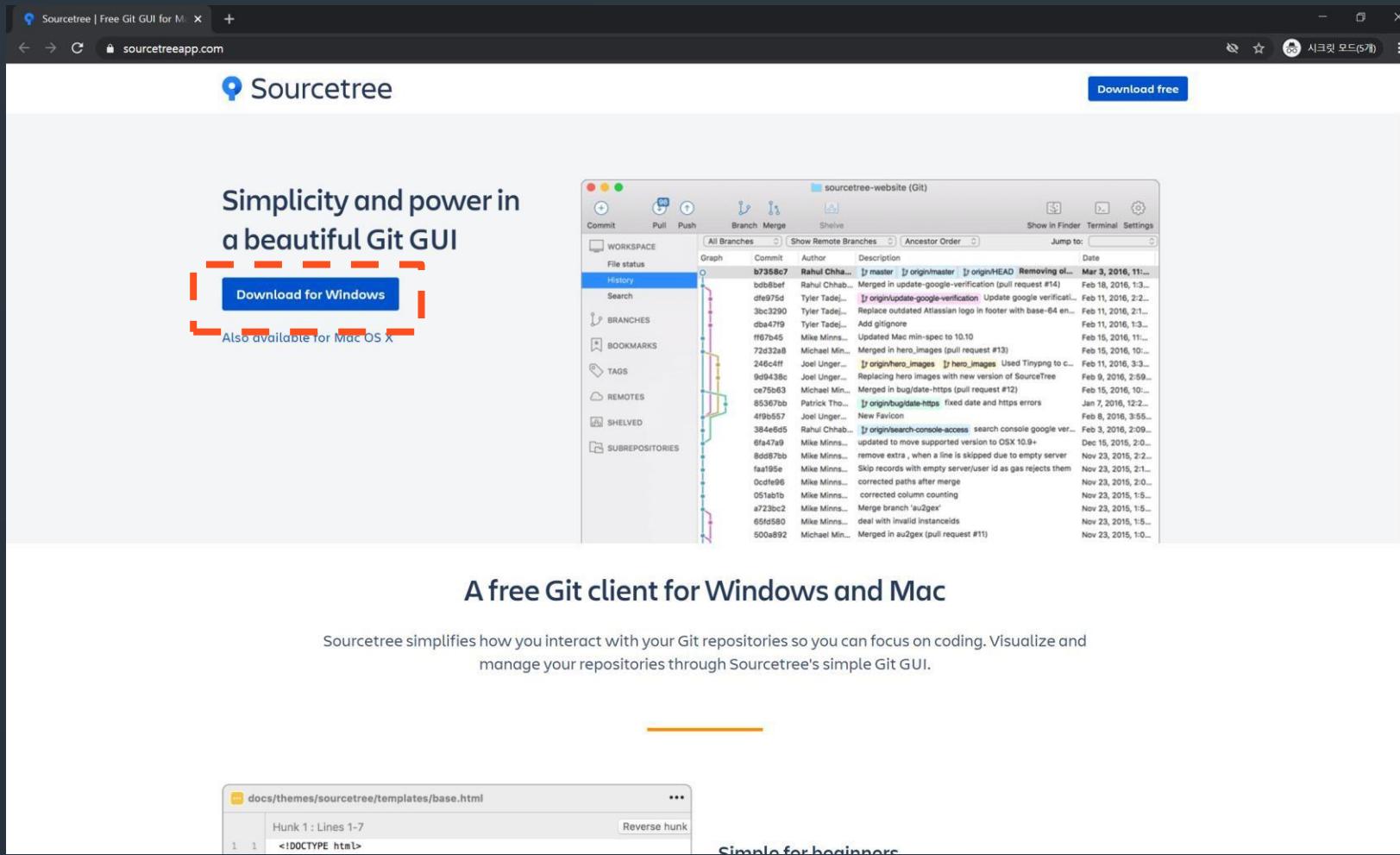


Sourcetree

설치

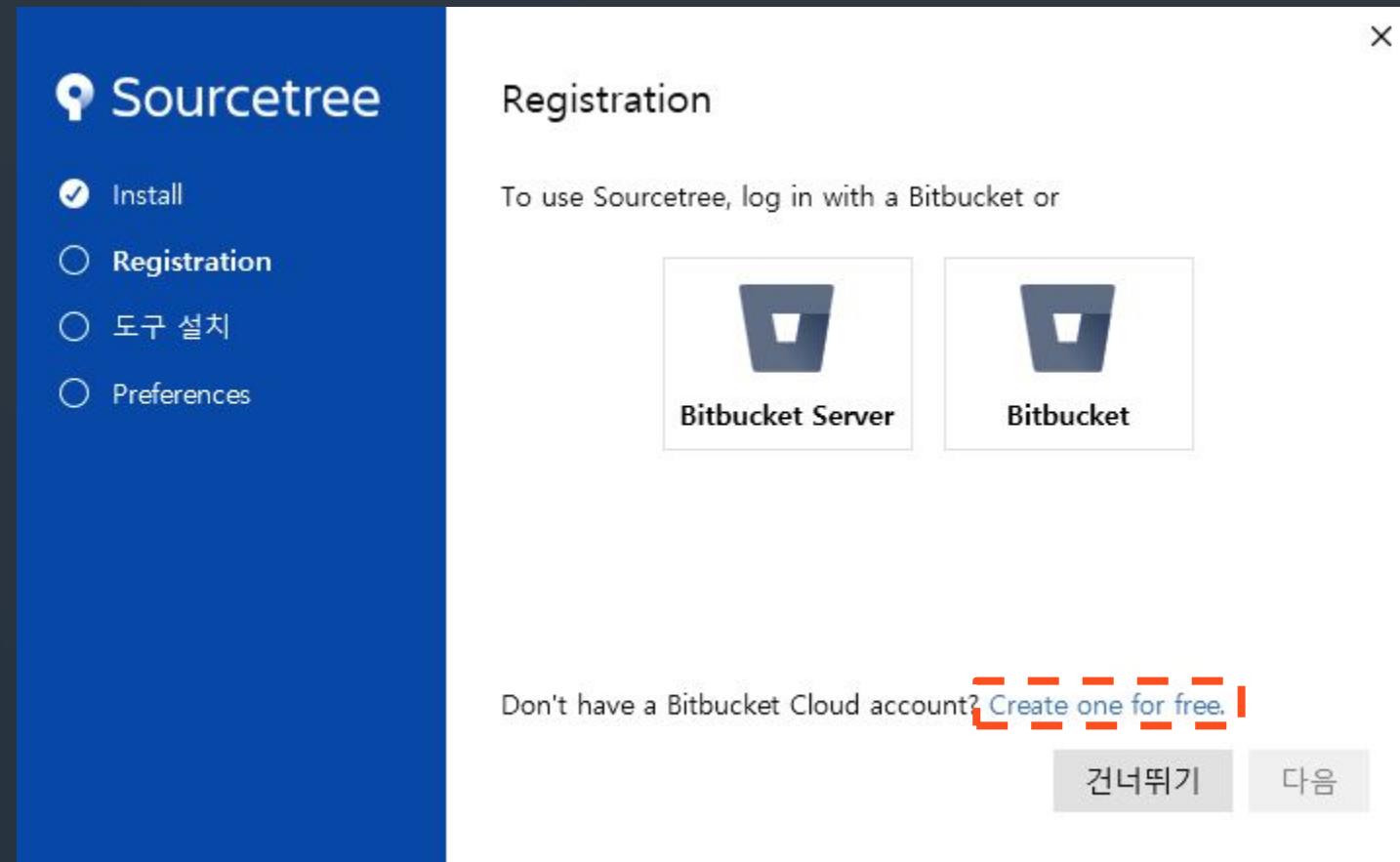
Sourcetree 다운로드

- Sourcetree는 Git을 통한 버전관리를 할 수 있는 GUI Tool
- <https://www.sourcetreeapp.com/> 접속 후 다운로드



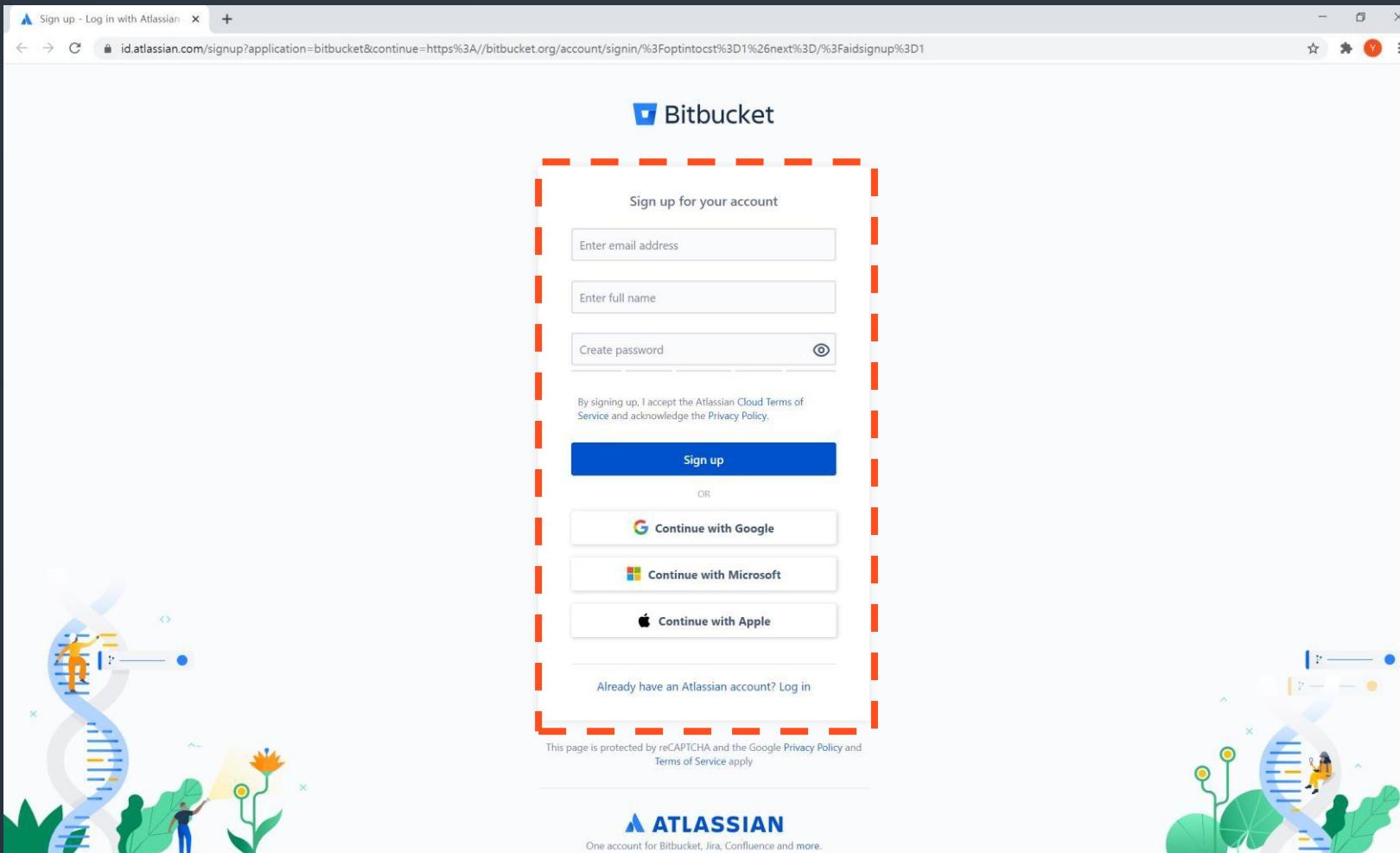
Sourcetree 설치

- 계정생성(최초에는 계정을 생성하고, 계정이 있는 경우 바로 진행)



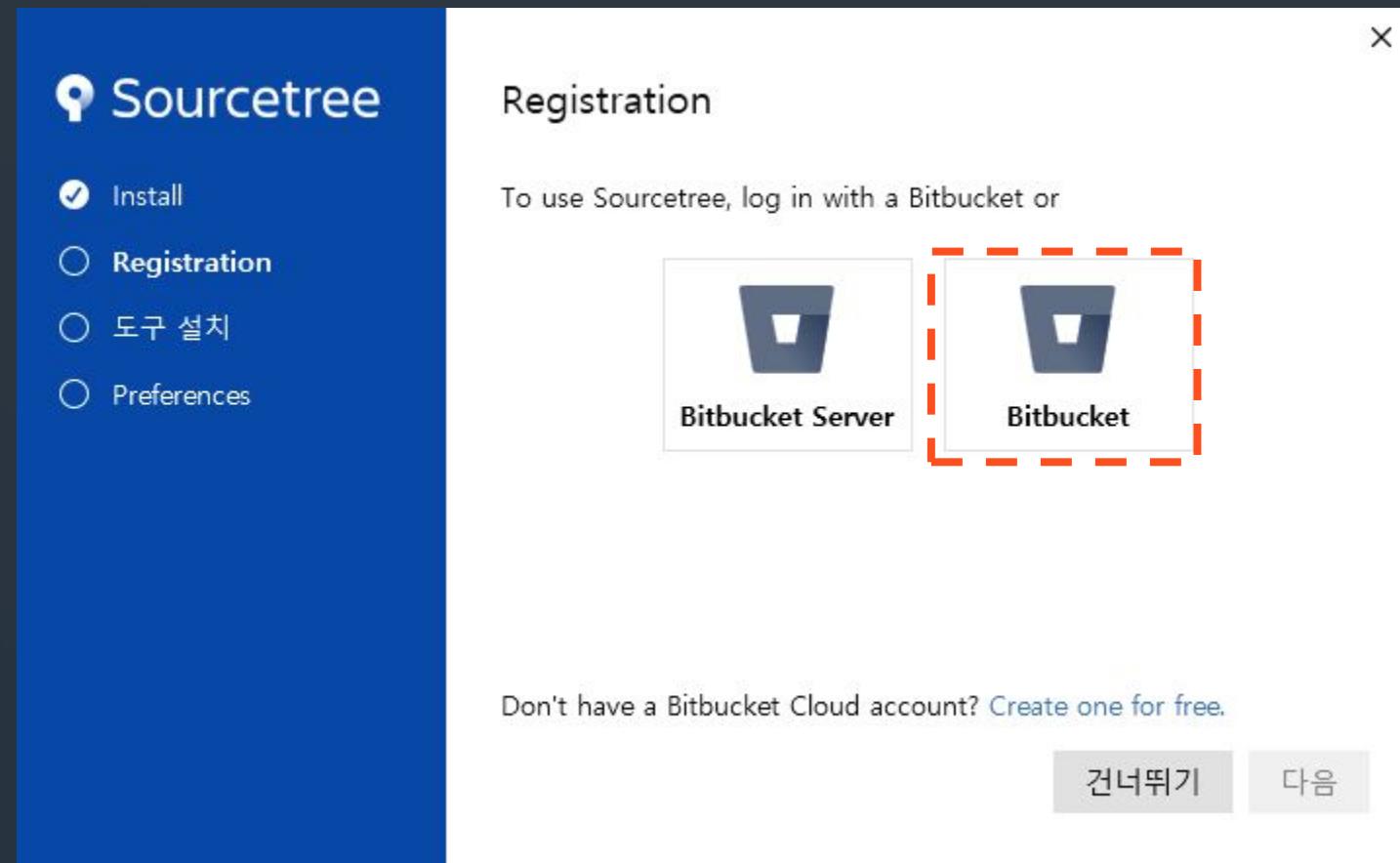
Sourcetree 설치

- 가입정보 입력 후 가입(EMAIL 확인 과정 필요)



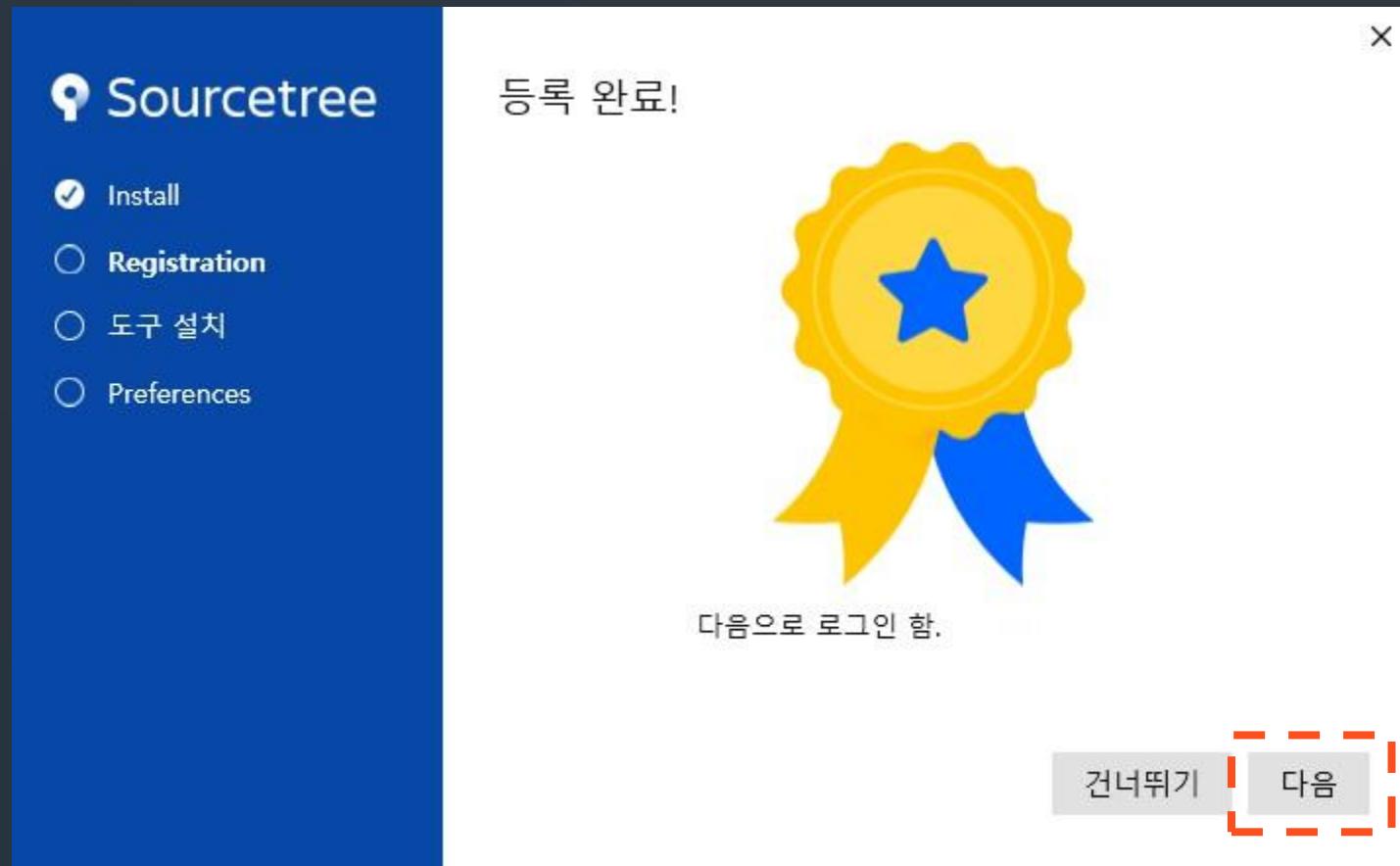
Sourcetree 설치

- Bitbucket을 눌러 가입계정 인증



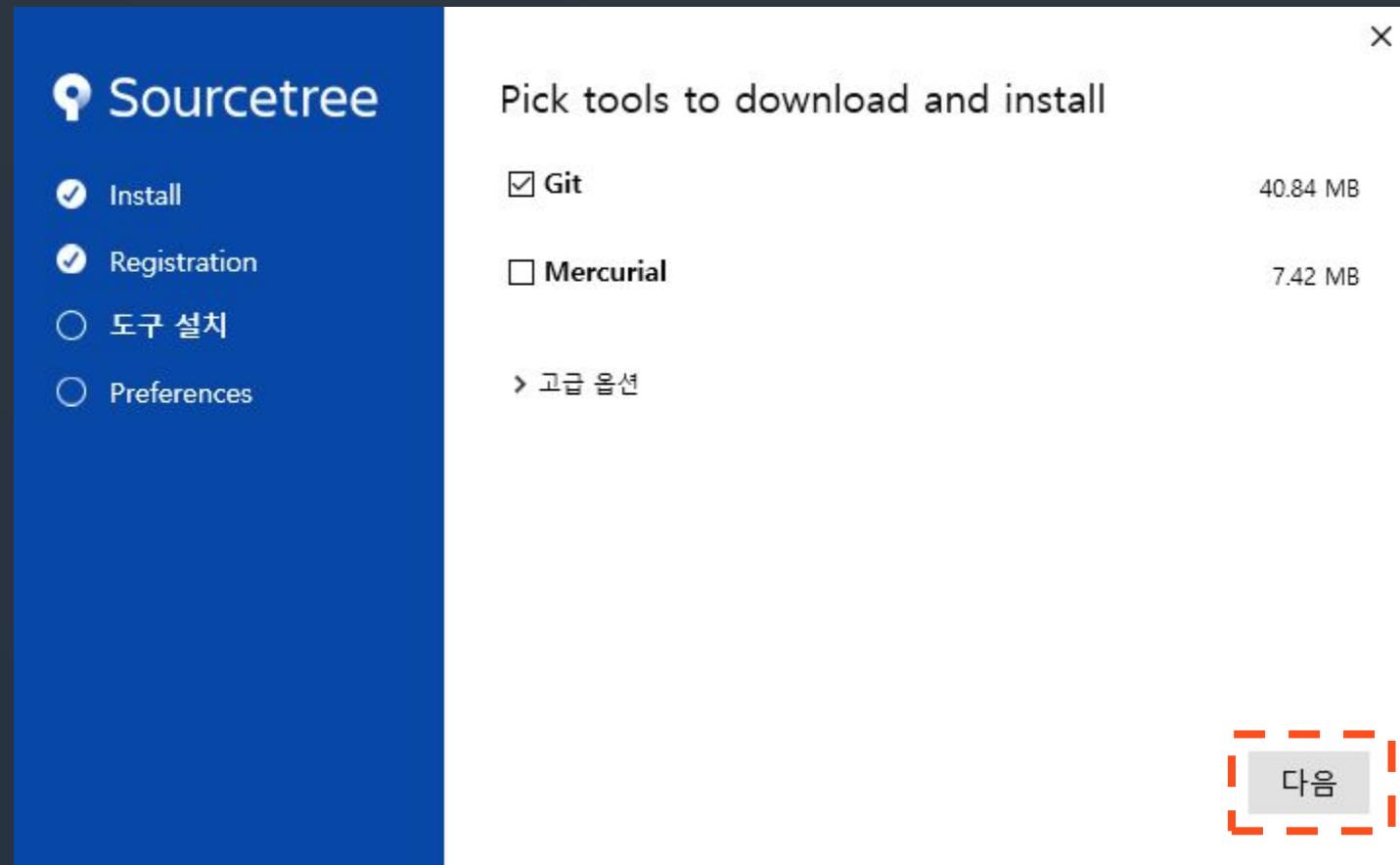
Sourcetree 설치

- 인증완료화면



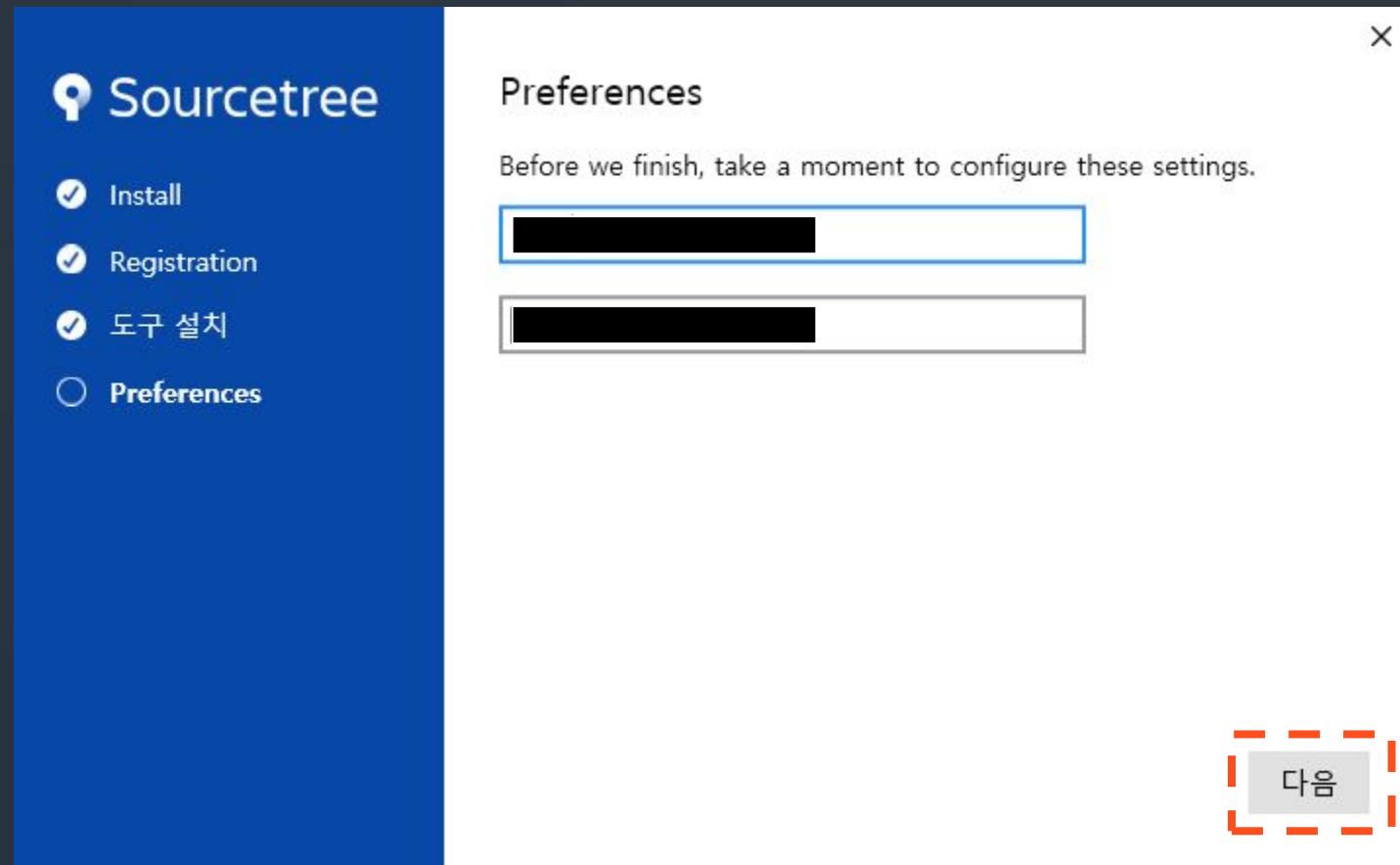
Sourcetree 설치

- 소스트리에서 사용할 버전관리도구 설치(Git 설치)



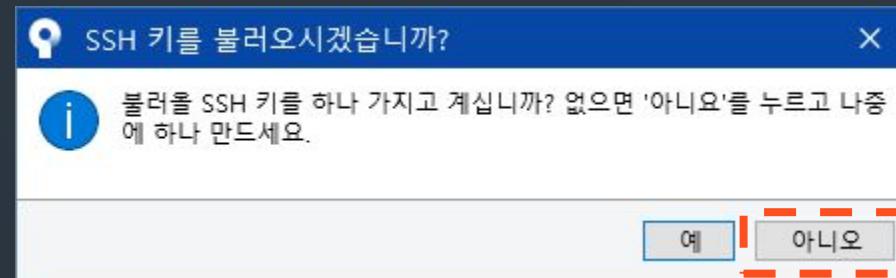
Sourcetree 설치

- 사용할 이름과 EMAIL주소등록



Sourcetree 설치

- SSH키가 현재 없으므로 '아니오' 선택

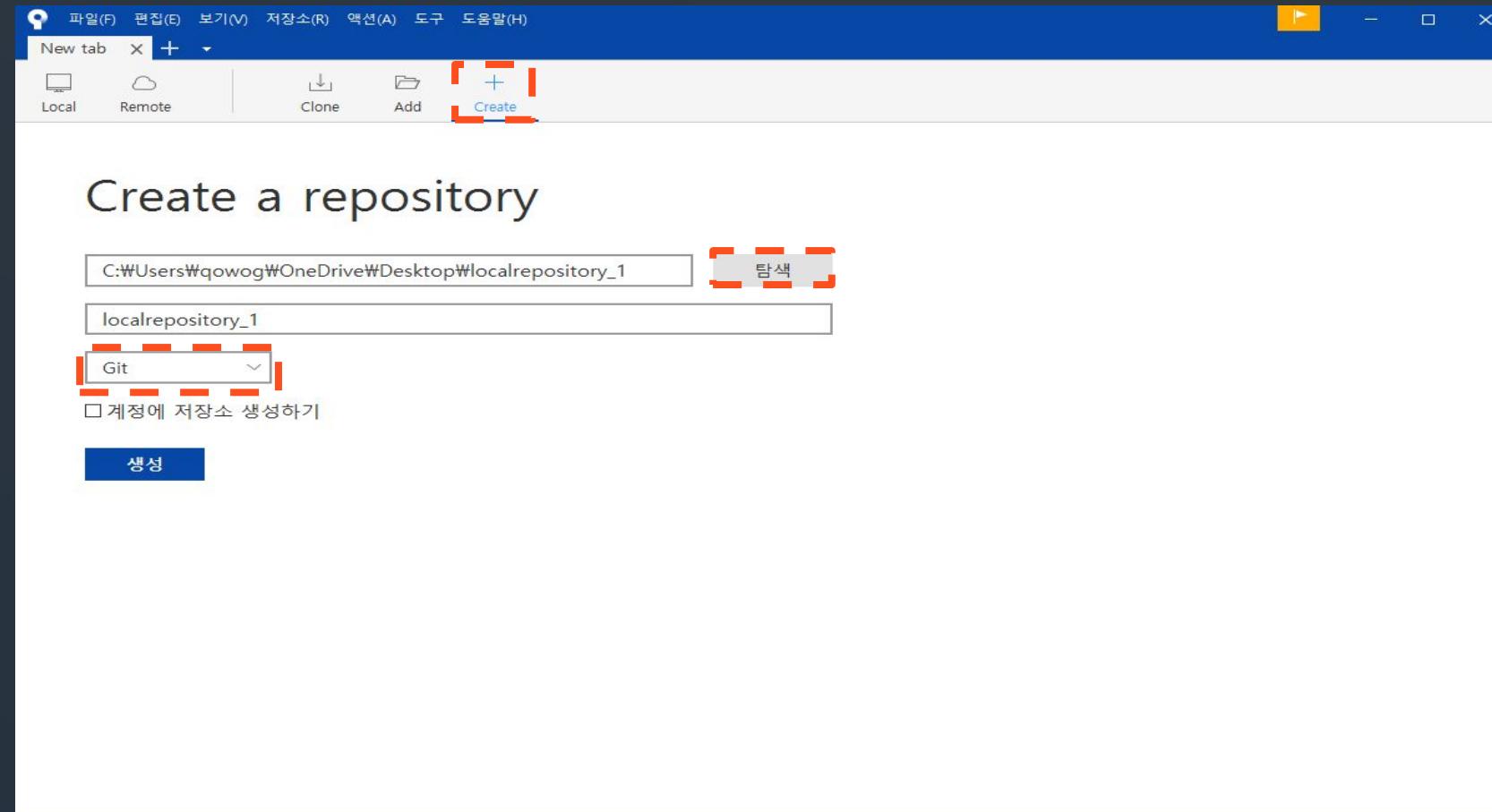




로컬 저장소 관리

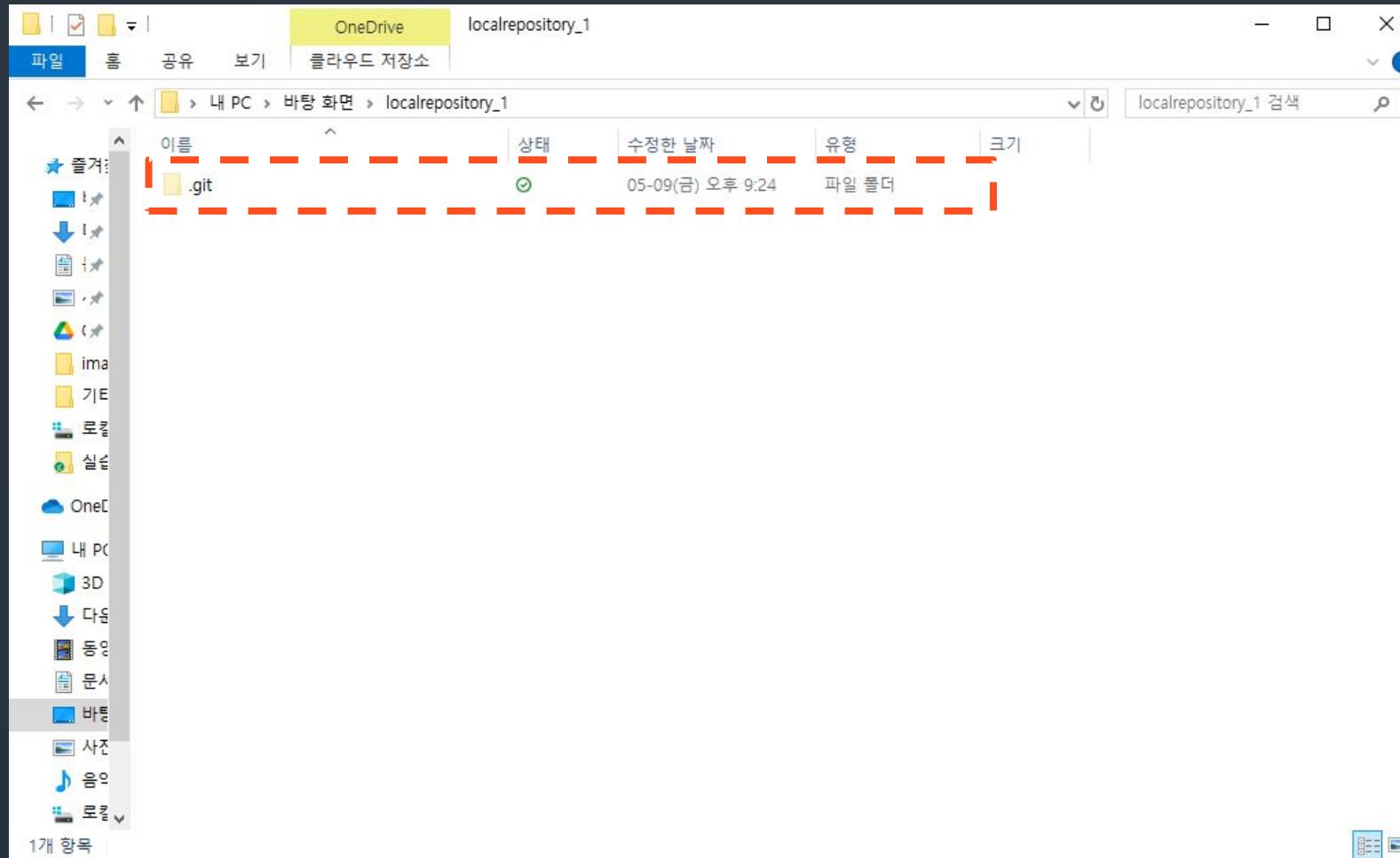
버전관리할 파일들이 있는 폴더 선택

- 선택된 폴더는 이제 Sourcetree에 의해서 파일 관리 시작



Sourcetree에서 선택된 폴더

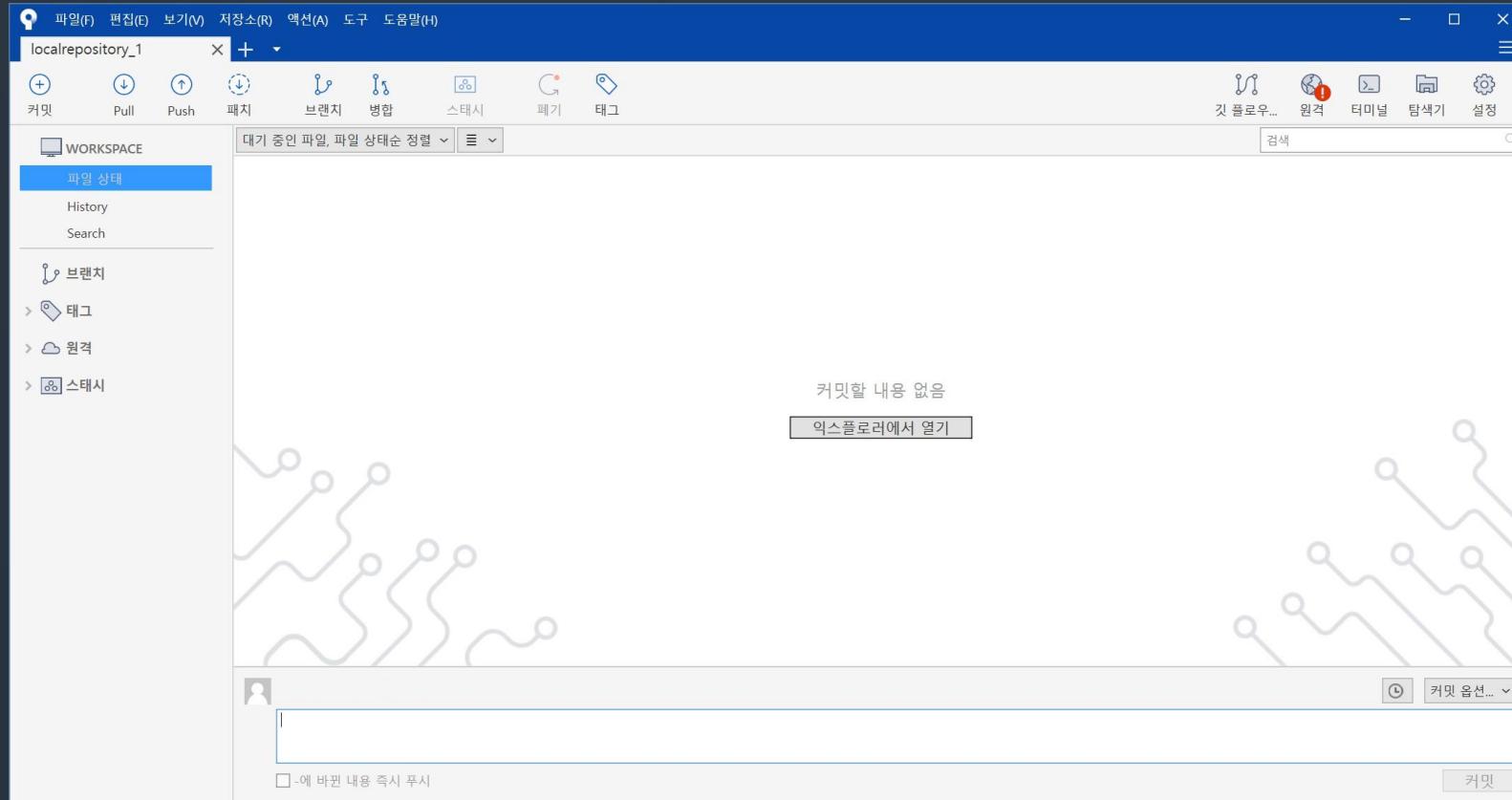
- 선택된 폴더는 .git이라는 숨겨진 폴더가 생성(보기 - 숨긴 항목 표시 체크해야 보임)





Sourcetree 화면

- Sourcetree는 해당 폴더 내부에 있는 파일들의 변화를 감지(파일생성/변경/삭제 등)





폴더에 새로운 파일 생성

- first.html 파일을 새로 생성하여 작성

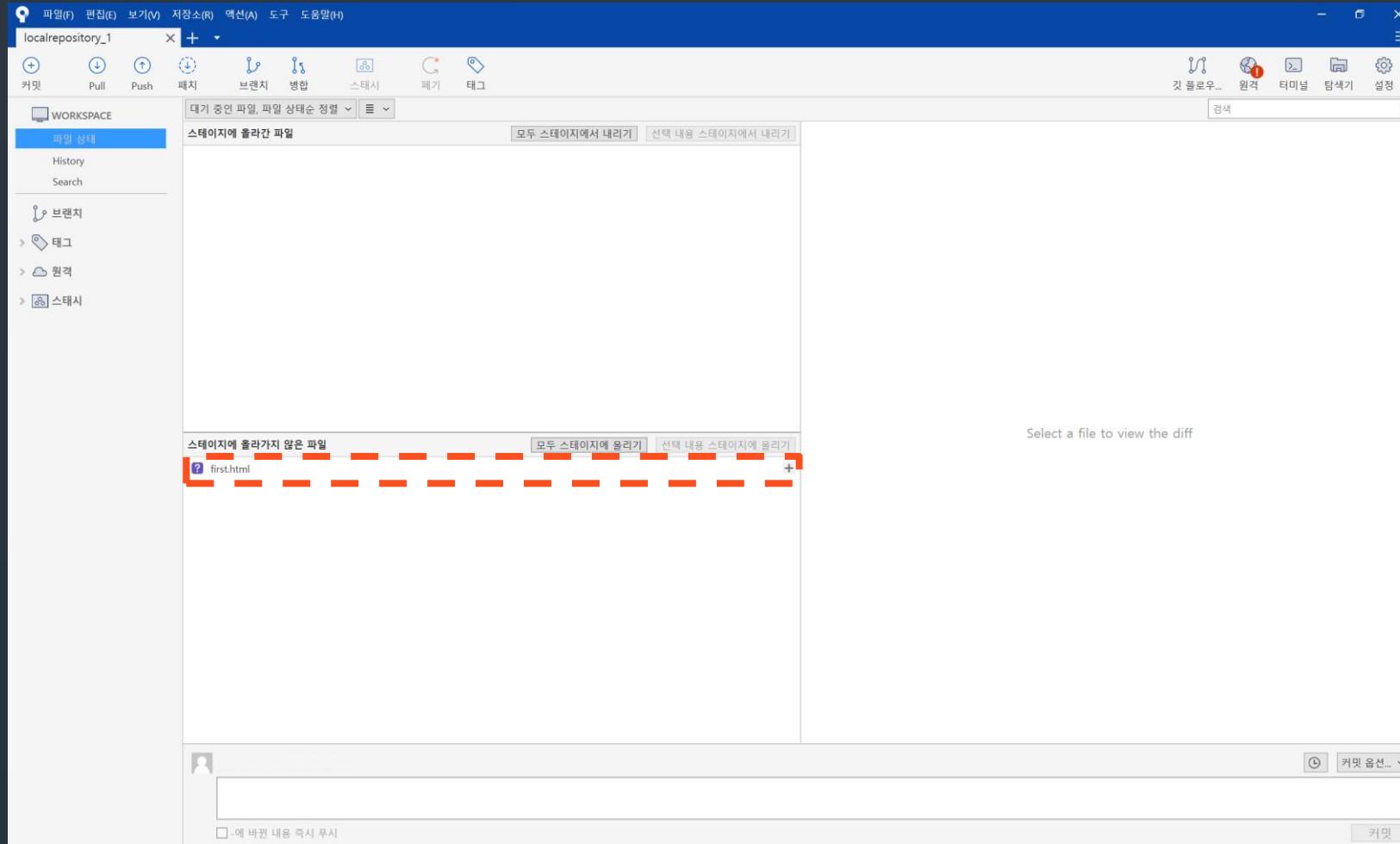


```
first.html > html
1  <!DOCTYPE html>
2  <html lang="kr">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Document</title>
7  </head>
8  <body>
9      <h1>첫번째 HTML파일</h1>
10 </body>
11 </html>
```



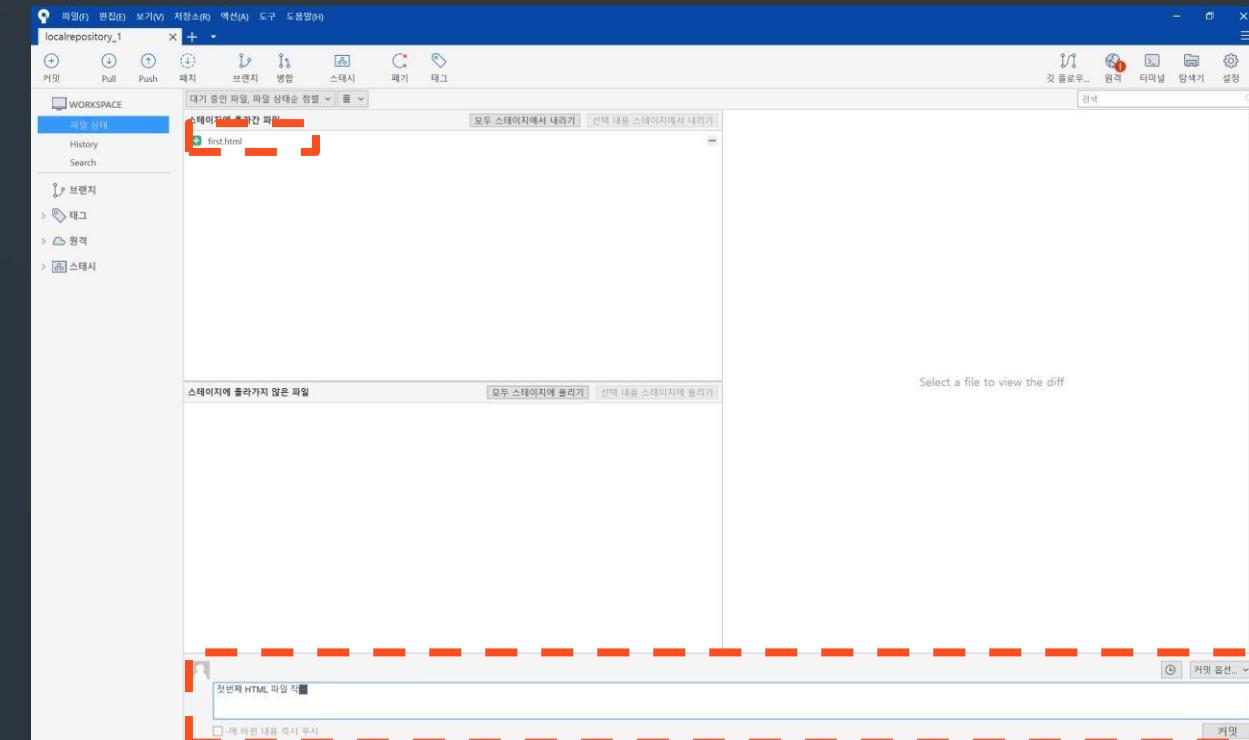
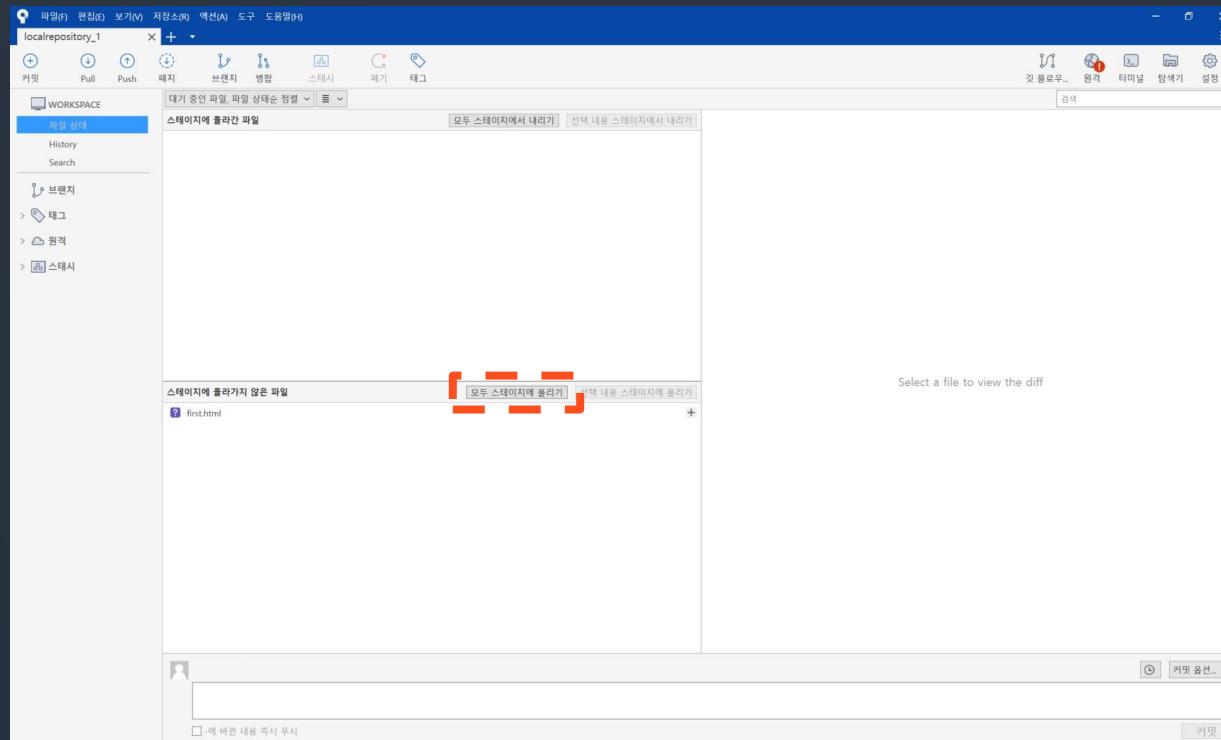
폴더의 변화를 감지하여 화면에 출력

- 새로 생성된 first.html 파일을 화면에 보여줌



변화된 폴더의 상태를 적용

- 모두 스테이지에 올리기 클릭
- first.html 파일이 상단으로 이동
- 변경내용에 대한 설명 작성 후 커밋 클릭





첫번째버전 생성완료

- 새로 생성된 first.html 파일을 적용하여 첫번째 버전 생성완료

The screenshot shows a Git commit history for a local repository named 'localrepository_1'. The commit message is '첫번째 HTML 파일 작성' (Create first HTML file) and the commit hash is '2da76bf'. The commit was made on '날짜' (Date) and is part of the 'master' branch.

Commit Details:

- Author: 원격 (Remote)
- Date: 날짜 (Date)
- Commit Hash: 2da76bf
- Message: 첫번째 HTML 파일 작성 (Create first HTML file)

File Content:

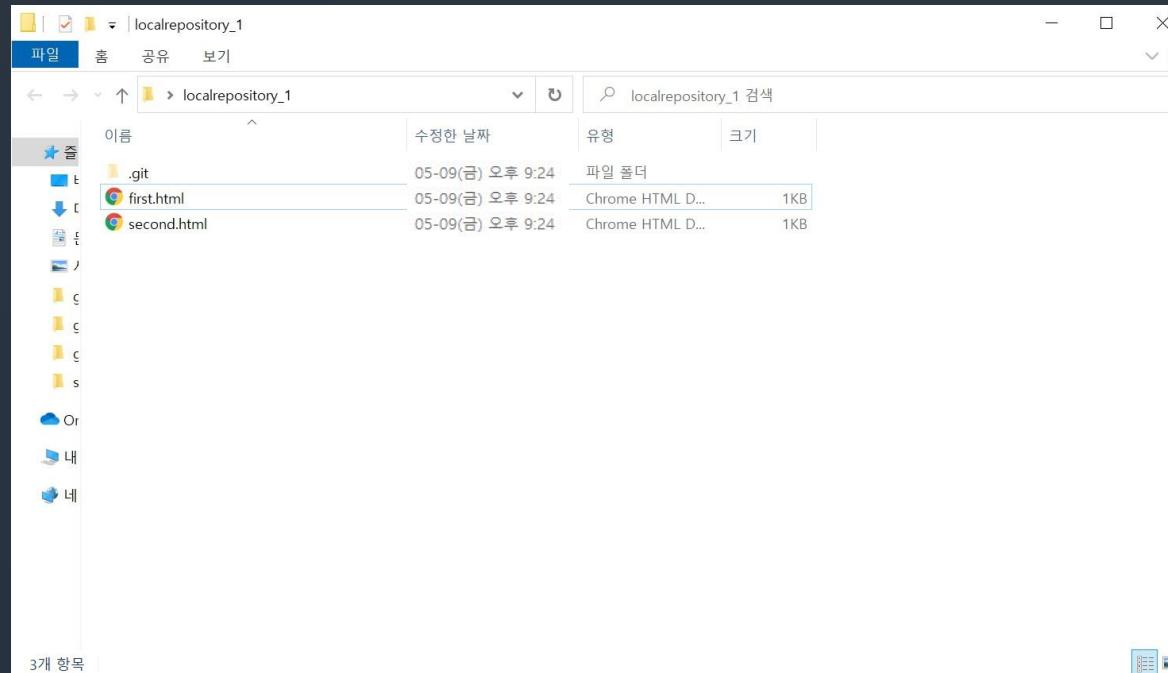
```
+<!DOCTYPE html>
+<html lang="kr">
+<head>
+<meta charset="UTF-8">
+<meta name="viewport" content="width=device-width, initial-scale=1.0">
+<title>Document</title>
+</head>
+<body>
+<h1>첫번째 HTML파일</h1>
+</body>
+</html>
```

Line 11: \ No newline at end of file



폴더내부상태변경

- first.html 파일에 내용추가
- second.html 파일 새로 생성



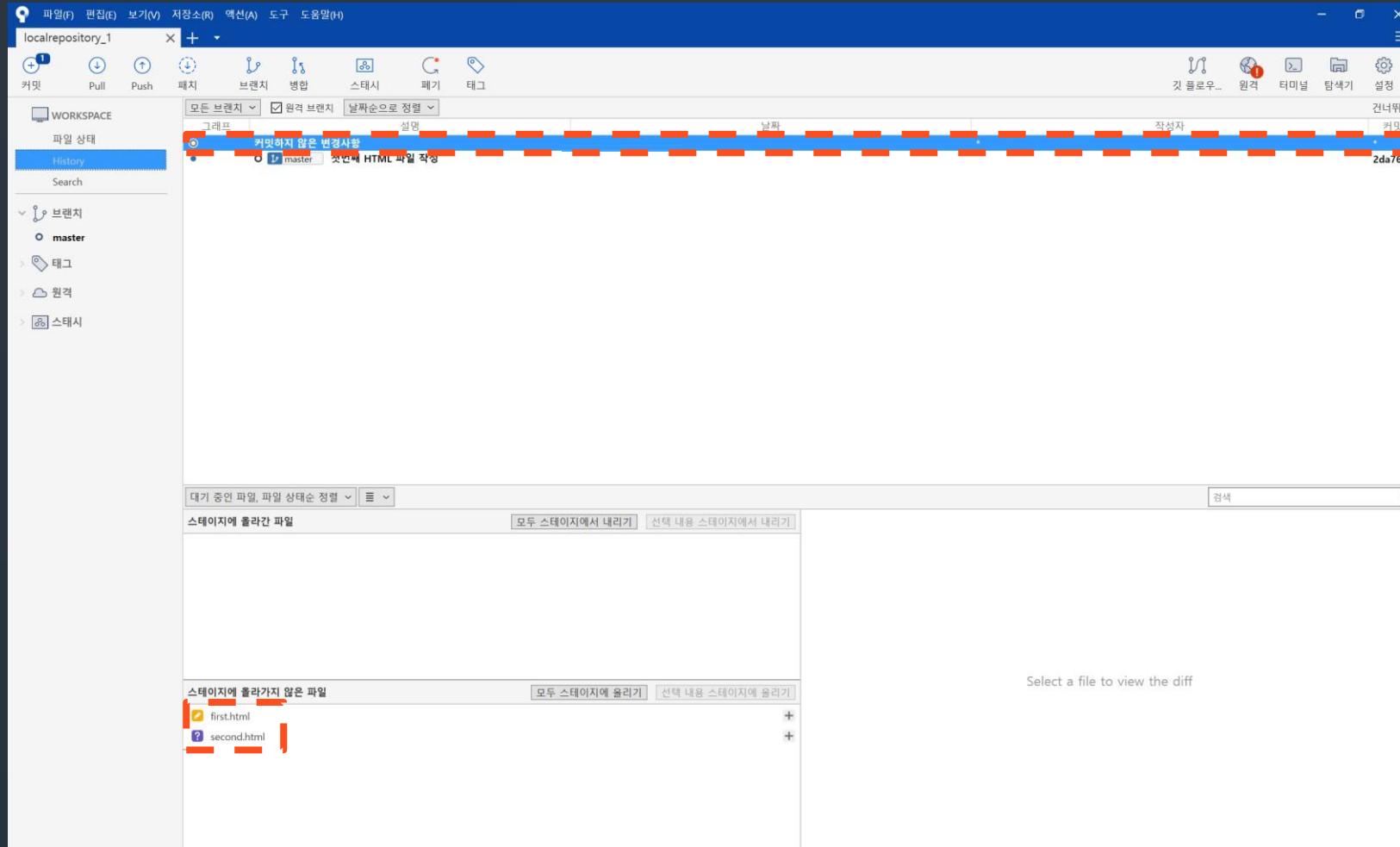
```
↳ first.html > html > body > h2
1  <!DOCTYPE html>
2  <html lang="kr">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Document</title>
7  </head>
8  <body>
9      <h1>첫번째 HTML파일</h1>
10     <h2>첫번째 HTML파일 내용 추가!!</h2>
11  </body>
12 </html>
```



```
↳ second.html > html > body > h1
1  <!DOCTYPE html>
2  <html lang="kr">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Document</title>
7  </head>
8  <body>
9      <h1>두번째 HTML파일</h1>
10 </body>
11 </html>
```

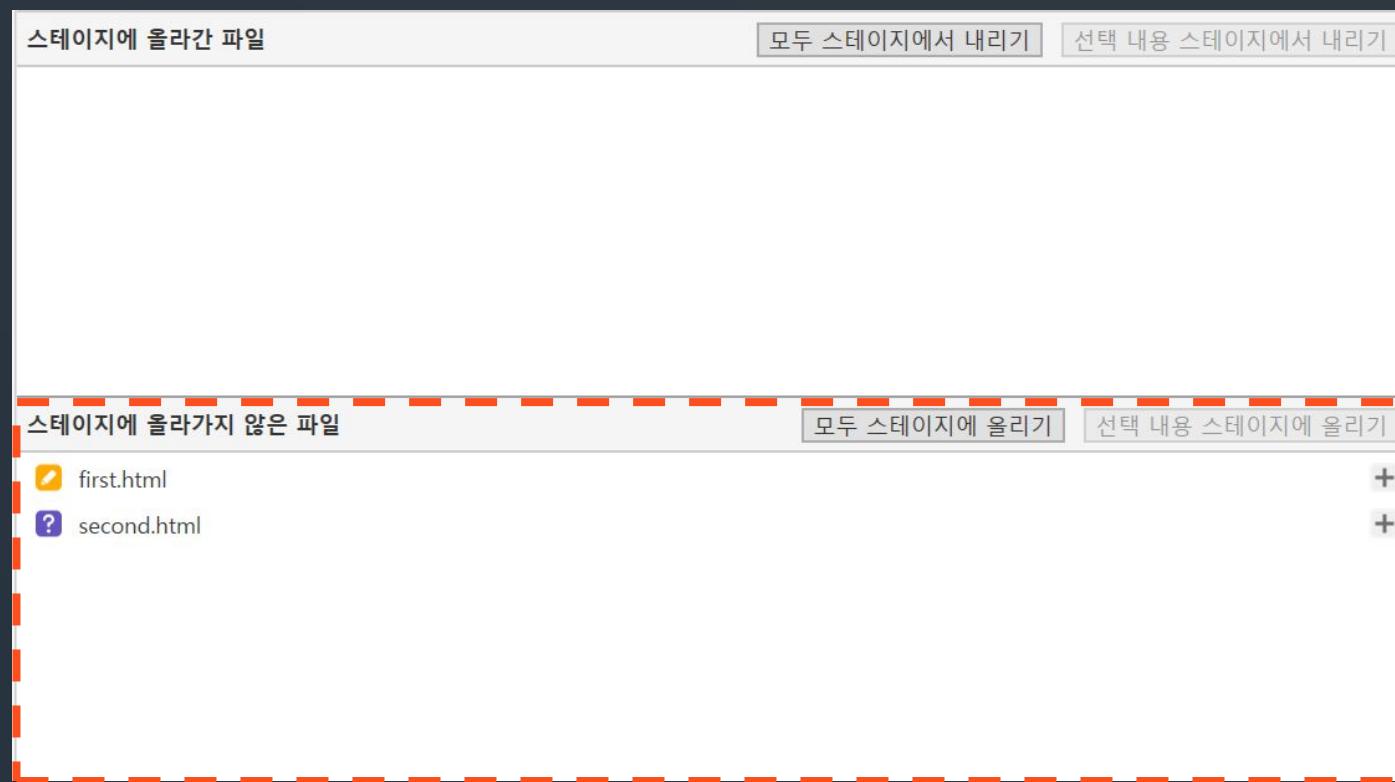
폴더의 변화를 감지하여 화면에 출력

- 새로 생성된 파일과 기존파일의 변화를 동시에 감지하여 화면에 출력



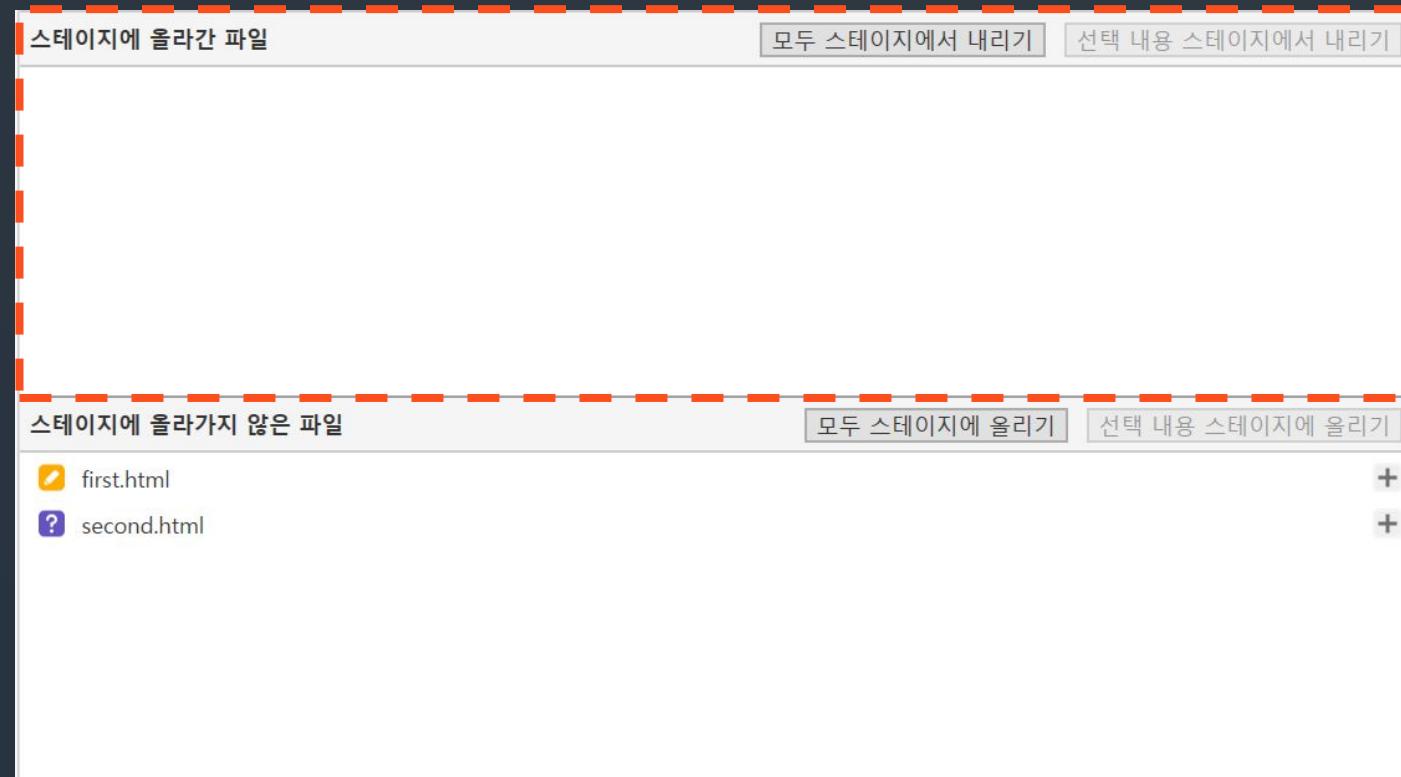
working copy

- 관리하고 있는 폴더에서 변경사항이 발생하는 파일들이 확인되는 공간
- 새로 생성된 파일, 변경된 파일, 삭제된 파일을 아이콘으로 다르게 표현
- 현재 상태에서 수정이 완료되어 다음버전에 포함시킬 파일을 선택하여 상단으로 이동



staging area(또는 index라고도 함)

- working copy에 있는 파일 중 수정이 완료되어 다음버전에 적용하기 위한 파일들이 올라오는 공간
- 커밋을 하는 경우 staging area에 있는 파일들의 변경사항만 다음 버전에 적용



add

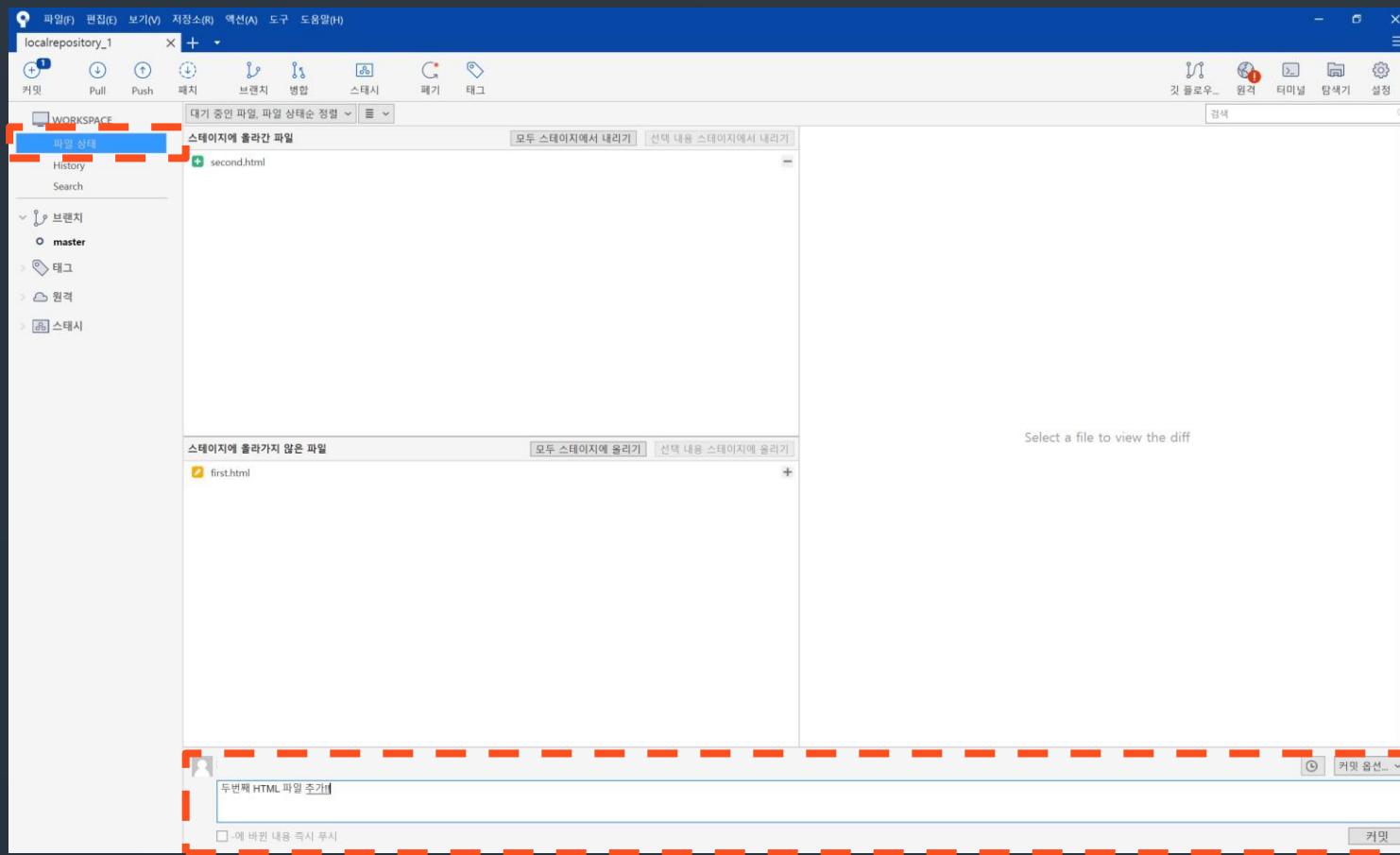
- **working copy → staging area로 파일을 옮기는 행위**
- **second.html만 상단으로 옮겨진 상태로 커밋을 하면 다음 버전에 second.html의 변경사항은 적용**
- **first.html의 변경사항은 다음버전에 적용 X**





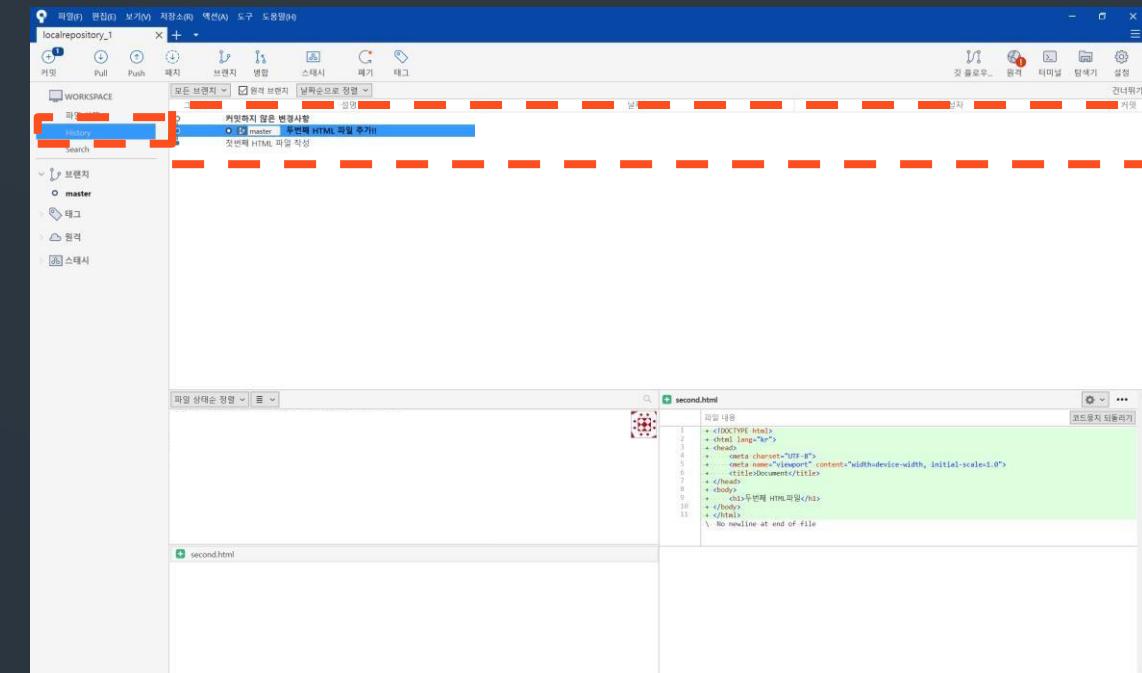
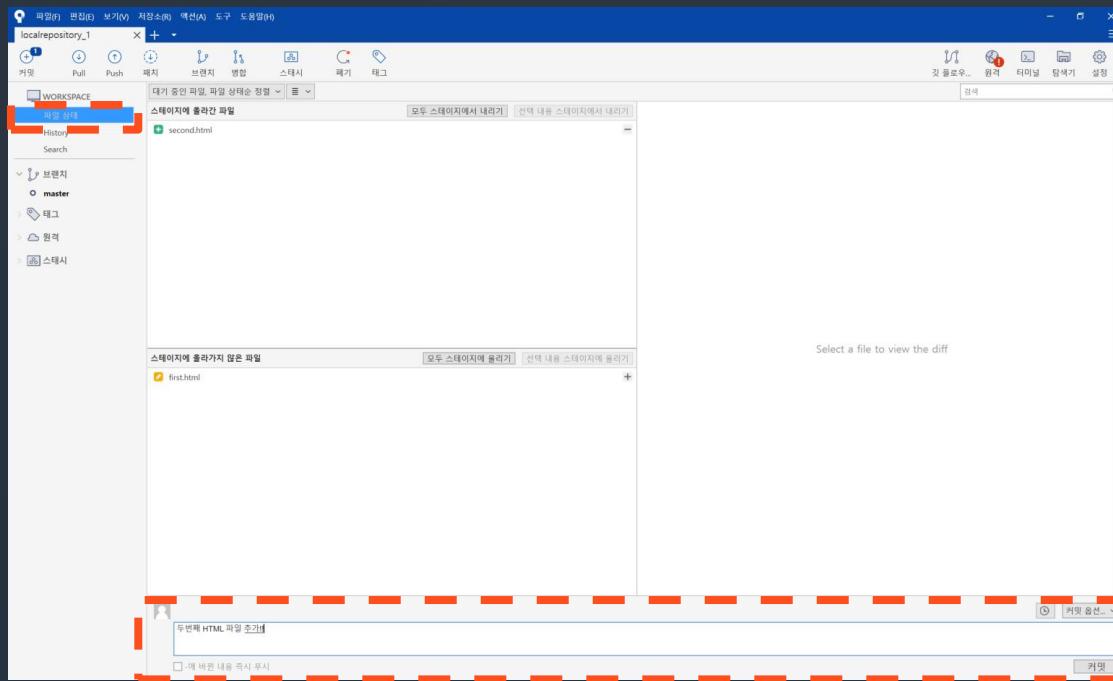
두번째 버전 생성

- second.html 파일만 추가하여 커밋



두번째 버전 생성

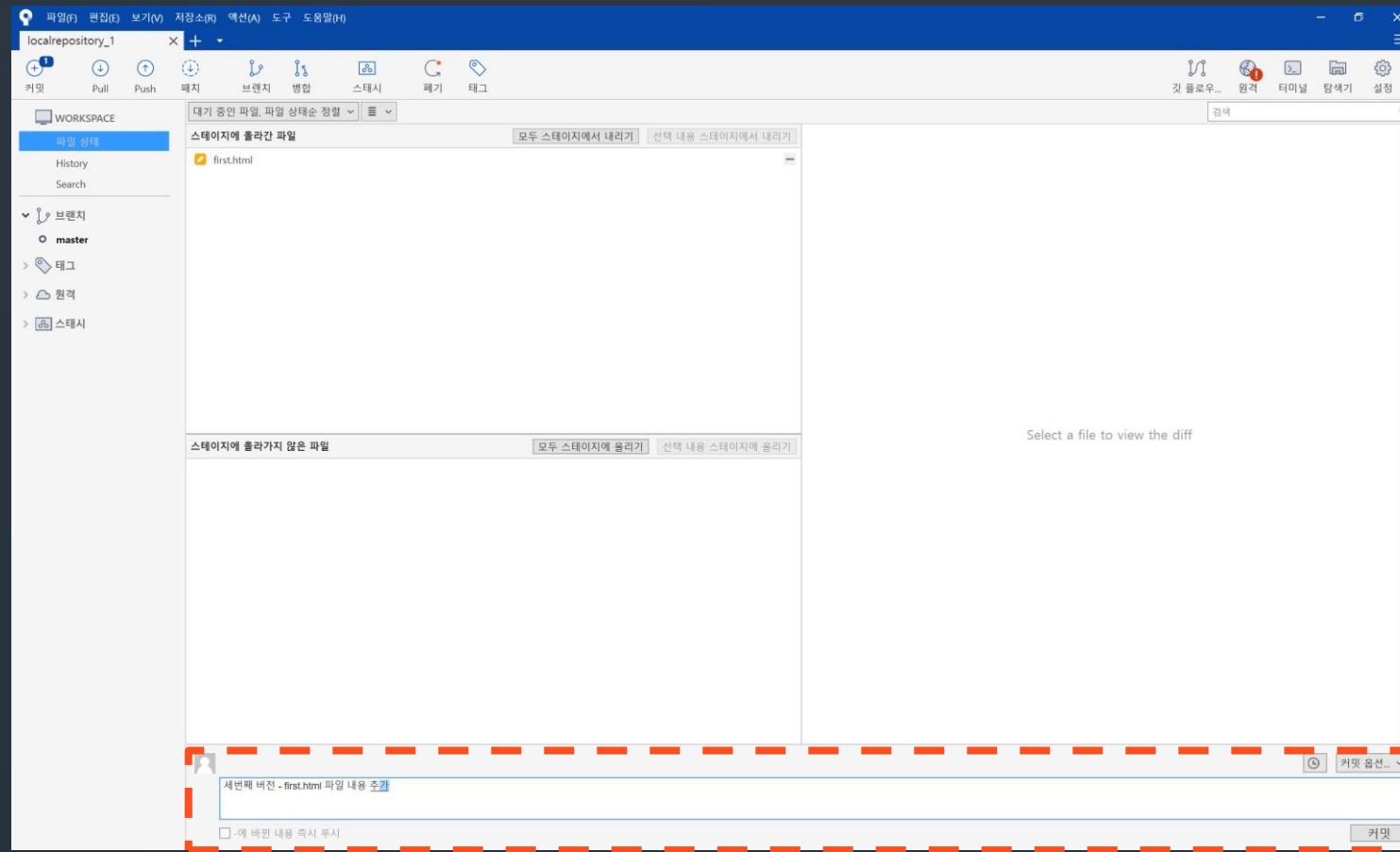
- **second.html 파일만 추가하여 커밋**
- **두번째 버전에는 second.html 파일은 추가되어있고, first.html 파일의 추가사항은 적용X**
- **first.html 파일에 추가된 내용은 커밋하지 않은 변경사항으로 남아있음**
- **원하는 파일들에 대해서만 버전적용이 가능**





세번째 버전 생성

- first.html 파일에 추가된 내용을 커밋하여 세번째 버전 생성





네번째 버전 생성

- second.html 파일에 ul태그를 추가하여 4번째 버전 생성

The screenshot shows the GitHub desktop application interface. The main window displays the code for 'second.html' with the following content:

```
<!DOCTYPE html>
<html lang="kr">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Document</title>
</head>
<body>
<h1>두번째 HTML파일</h1>
<ul>
<li>목록1</li>
<li>목록2</li>
<li>목록3</li>
</ul>
</body>
</html>
```

The code editor has syntax highlighting and line numbers. A status bar at the bottom indicates a commit message: "네번째 버전 - second.html 파일에 목록태그 추가".

The screenshot shows the DOM structure of the 'second.html' page as seen in a browser's developer tools. The structure is as follows:

- <html>
 - <head>
 - <meta charset="UTF-8">
 - <meta name="viewport" content="width=device-width, initial-scale=1.0">
 - <title>Document</title>
 - <body>
 - <h1>두번째 HTML파일</h1>
 -
 - 목록1
 - 목록2
 - 목록3
- </html>



변경사항 되돌리기-커밋전

- first.html 파일에 p태그 추가 후 sourcetree에서 폐기
- 파일이 변경전 상태로 되돌아감

```
first.html > html > body > p
1  <!DOCTYPE html>
2  <html lang="kr">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Document</title>
7  </head>
8  <body>
9      <h1>첫번째 HTML파일</h1>
10     <h2>첫번째 HTML파일 내용 추가!!</h2>
11     <p>메시지 추가!!!!!!</p>
12 </body>
13 </html>
```



The screenshot shows the SourceTree application interface. The top navigation bar includes '파일(F)', '편집(E)', '보기(V)', '저장소(R)', '옵션(A)', '도구(G)', and '도움말(H)'. Below the menu is a toolbar with icons for '커밋(C)', 'Pull', 'Push', '폐기(B)', '브랜치', '병합(M)', '스테이지(S)', and '태그(T)'. The main workspace is titled 'localrepository_1' and shows a 'WORKSPACE' section with '파일 상태' (File Status) selected. A red box highlights the '폐기' (Discard) button in the toolbar. The left pane displays a file tree for 'first.html'. The right pane shows the contents of 'first.html' with several changes highlighted in green and red. A red box highlights the line '11 11 <p>메시지 추가!!!!!!</p>' which has been modified. The bottom status bar indicates '커밋' (Commit).

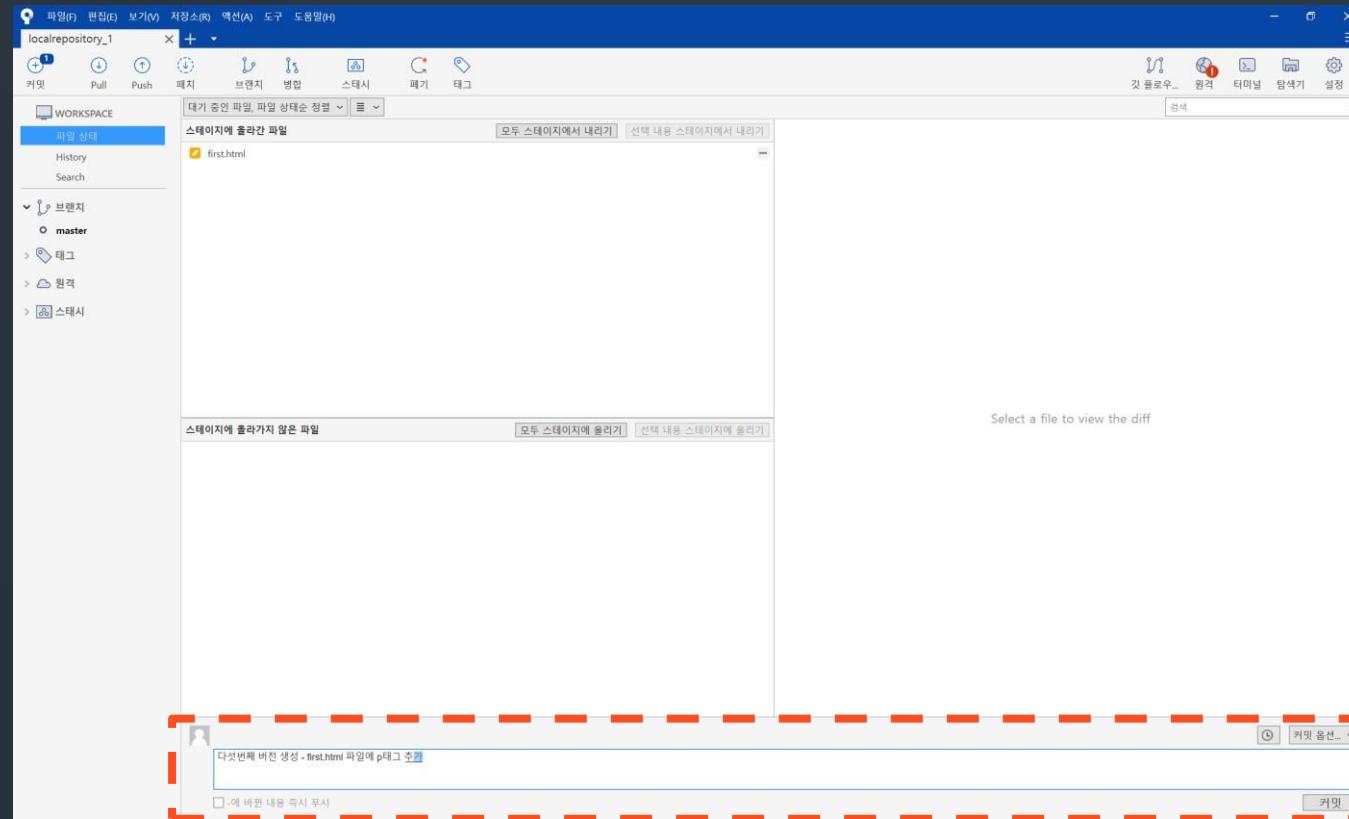
```
first.html
1 부분 : 8-13 줄
8 8      <body>
9 9          <h1>첫번째 HTML파일</h1>
10 10         <h2>첫번째 HTML파일 내용 추가!!</h2>
11 11         <p>메시지 추가!!!!!!</p>
12 12     </body>
13 13 </html>
\ No newline at end of file
```

```
first.html > html > body
1  <!DOCTYPE html>
2  <html lang="kr">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Document</title>
7  </head>
8  <body>
9      <h1>첫번째 HTML파일</h1>
10     <h2>첫번째 HTML파일 내용 추가!!</h2>
11 </body>
12 </html>
```



다섯번째 버전 생성

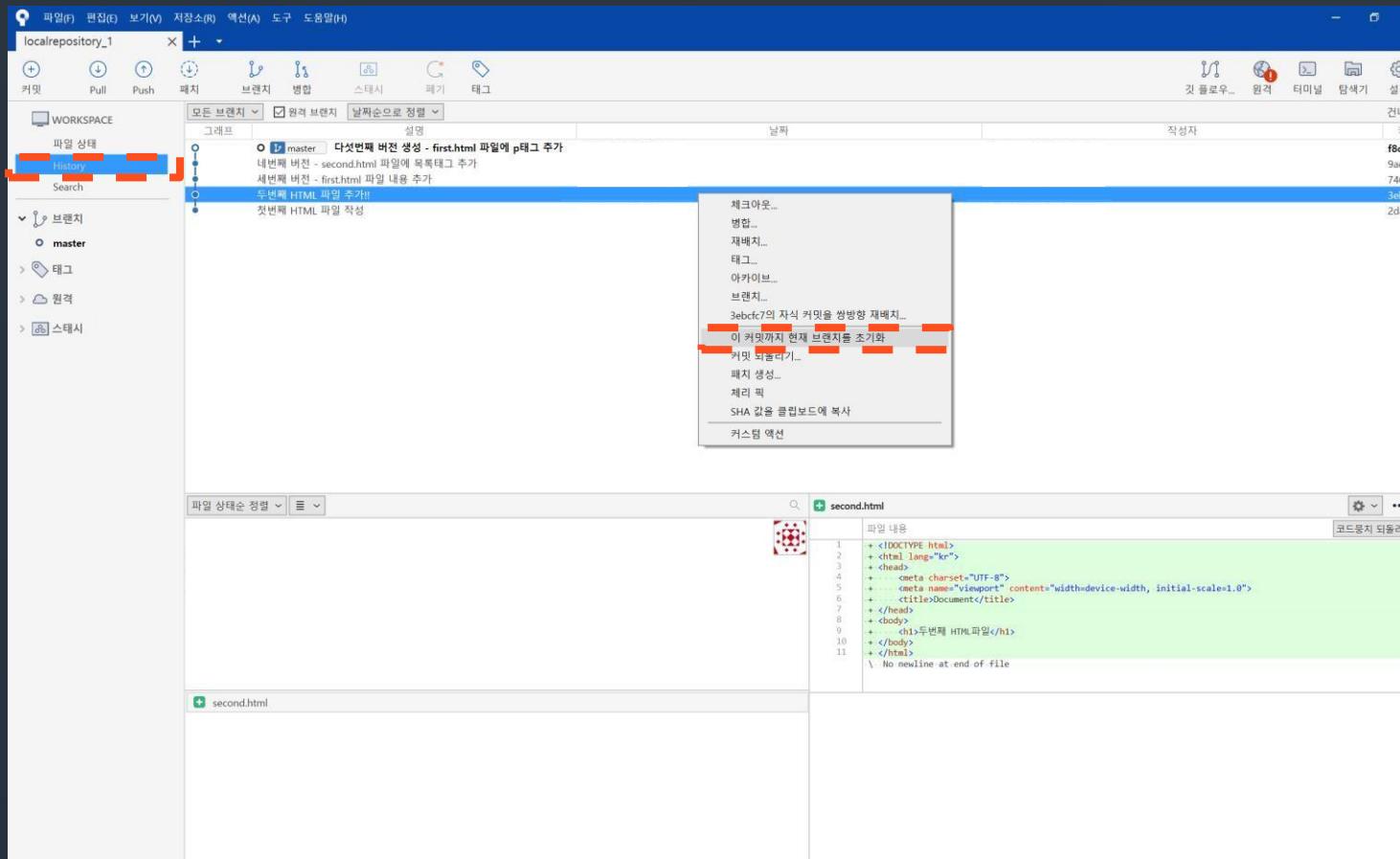
- first.html 파일에 p태그 추가 후 다섯번째 버전 생성



```
first.html > html > body > p
1  <!DOCTYPE html>
2  <html lang="kr">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Document</title>
7  </head>
8  <body>
9      <h1>첫번째 HTML파일</h1>
10     <h2>첫번째 HTML파일 내용 추가!!</h2>
11     <p>메세지 추가!!!!</p>
12 </body>
13 </html>
```

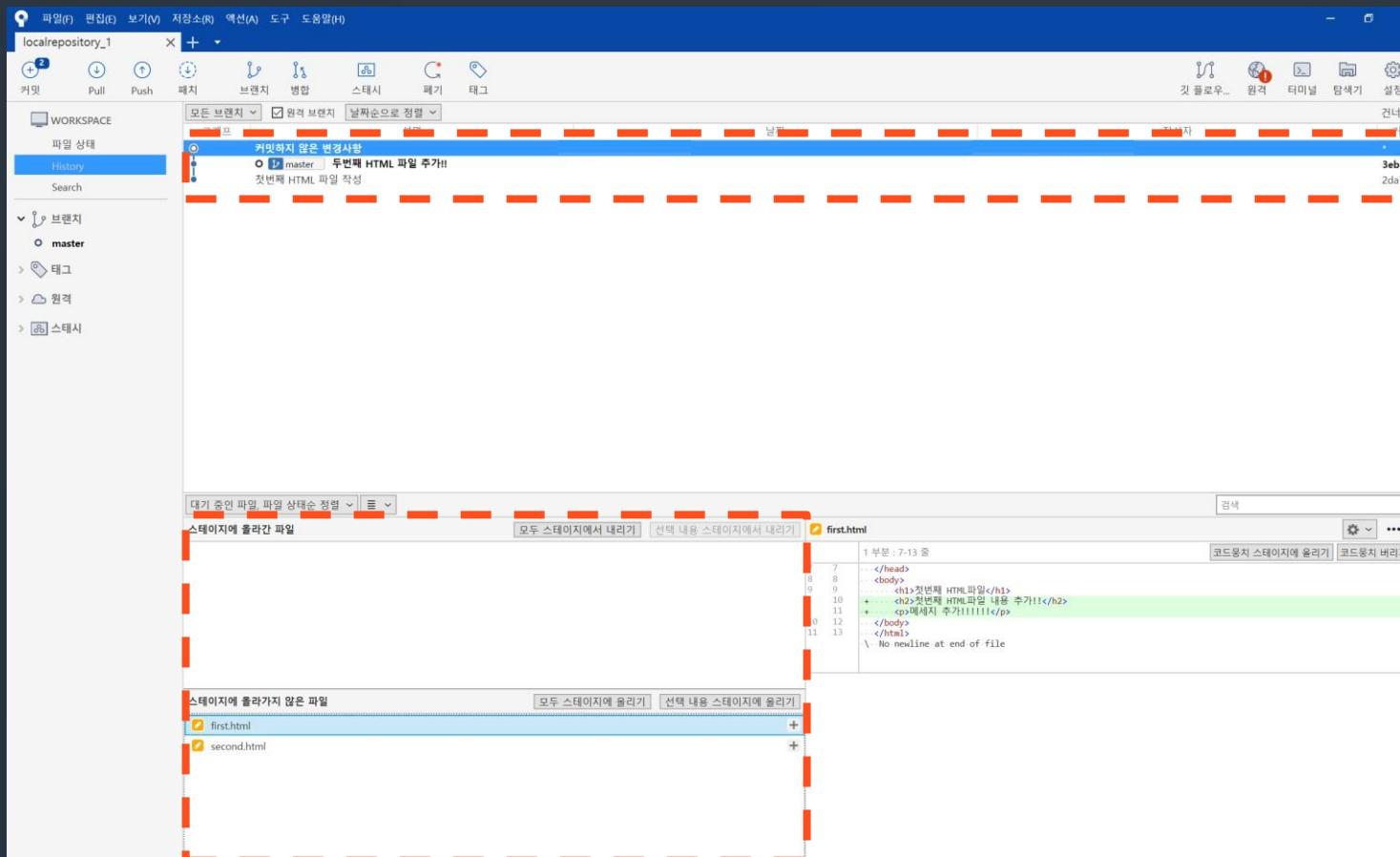
변경사항 되돌리기 - 커밋 후 (reset)

- 두번째 버전에서 마우스 오른쪽 클릭 후 '이 커밋까지 현재 브랜치를 초기화' 선택
- 초기화 모드는 3가지 존재 soft, mixed, hard(default : mixed)
- mixed 선택 후 적용



변경사항 되돌리기 - 커밋 후 (reset)

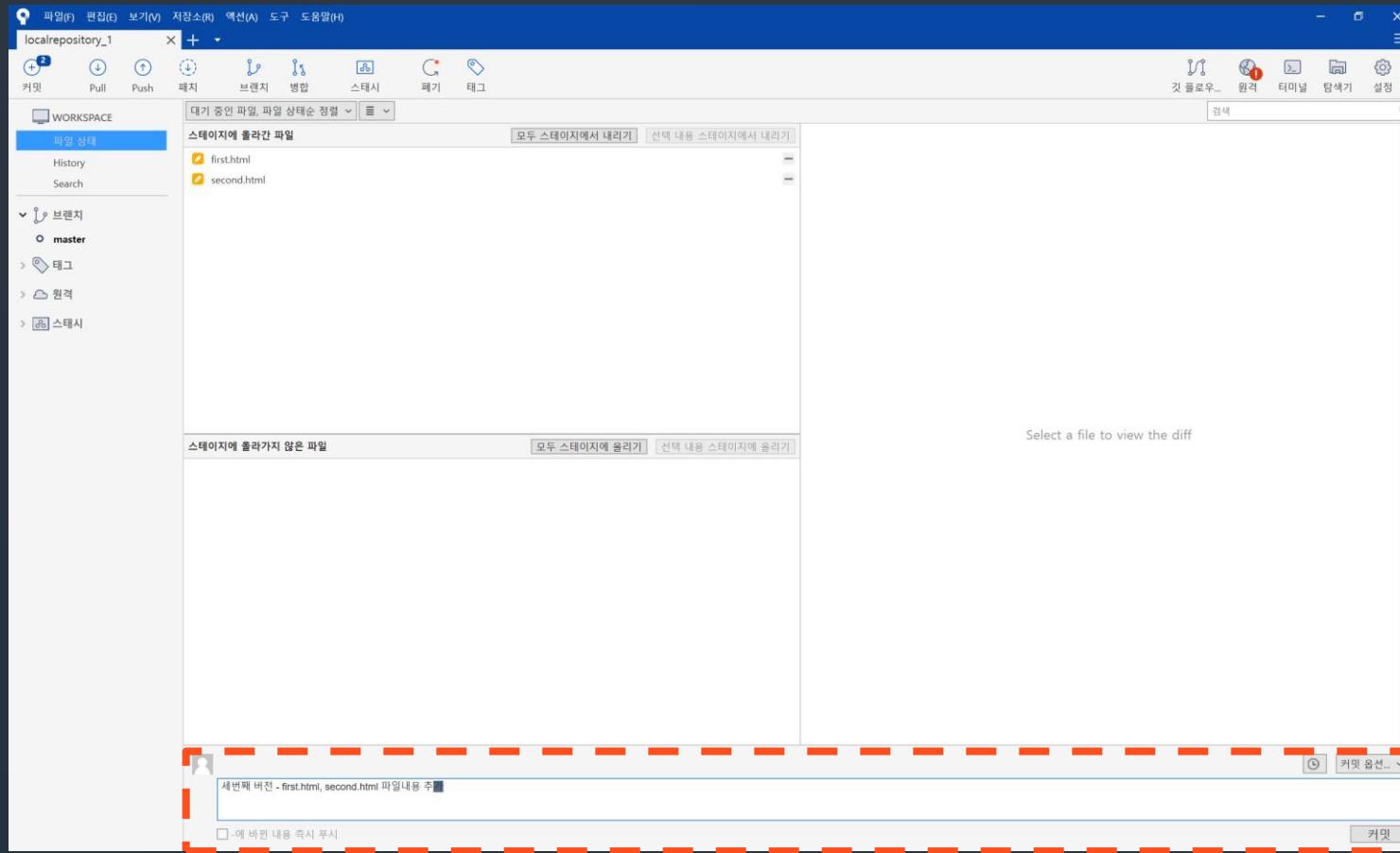
- 두 번째 버전인 상태로 로컬저장소 초기화
- soft인 경우 파일은 변경된 상태 유지하며 변경파일은 staging area에 위치
- mixed인 경우 파일은 변경된 상태를 유지하며 변경파일을 working copy에 위치
- hard인 경우 파일도 두 번째 버전 상태로 reset





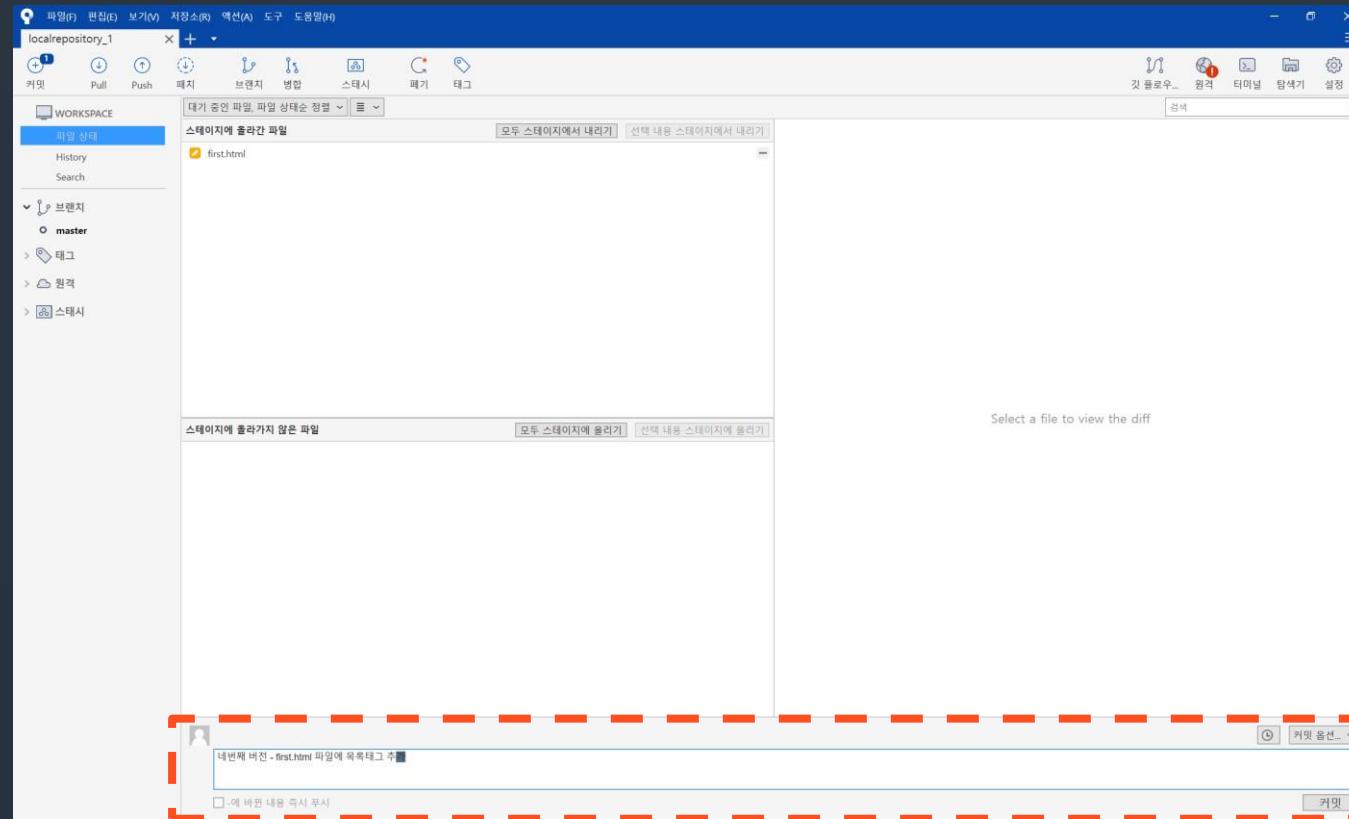
새로운 세번째 버전 생성

- 두개 변경사항 모두 적용하여 새로운 세번째 버전 생성



네번째 버전 생성

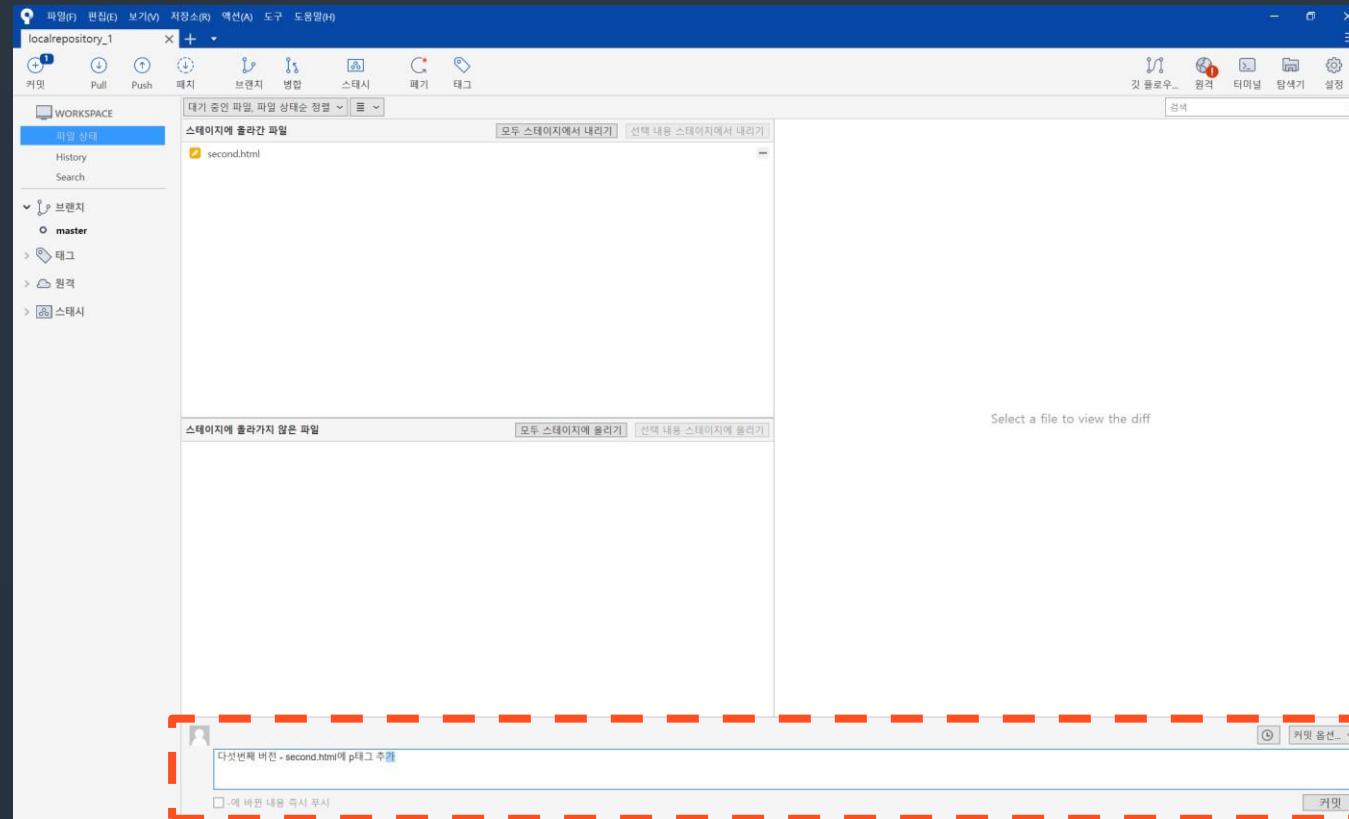
- first.html 파일에 목록태그 추가 후 네번째 버전 생성



```
first.html > html > body > ul > li
1  <!DOCTYPE html>
2  <html lang="kr">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Document</title>
7  </head>
8  <body>
9      <h1>첫번째 HTML파일</h1>
10     <h2>첫번째 HTML파일 내용 추가!!</h2>
11     <ul>
12         <li>메뉴1<li>
13         <li>메뉴2<li>
14     </ul>
15 
16 </body>
17 </html>
```

다섯번째 버전 생성

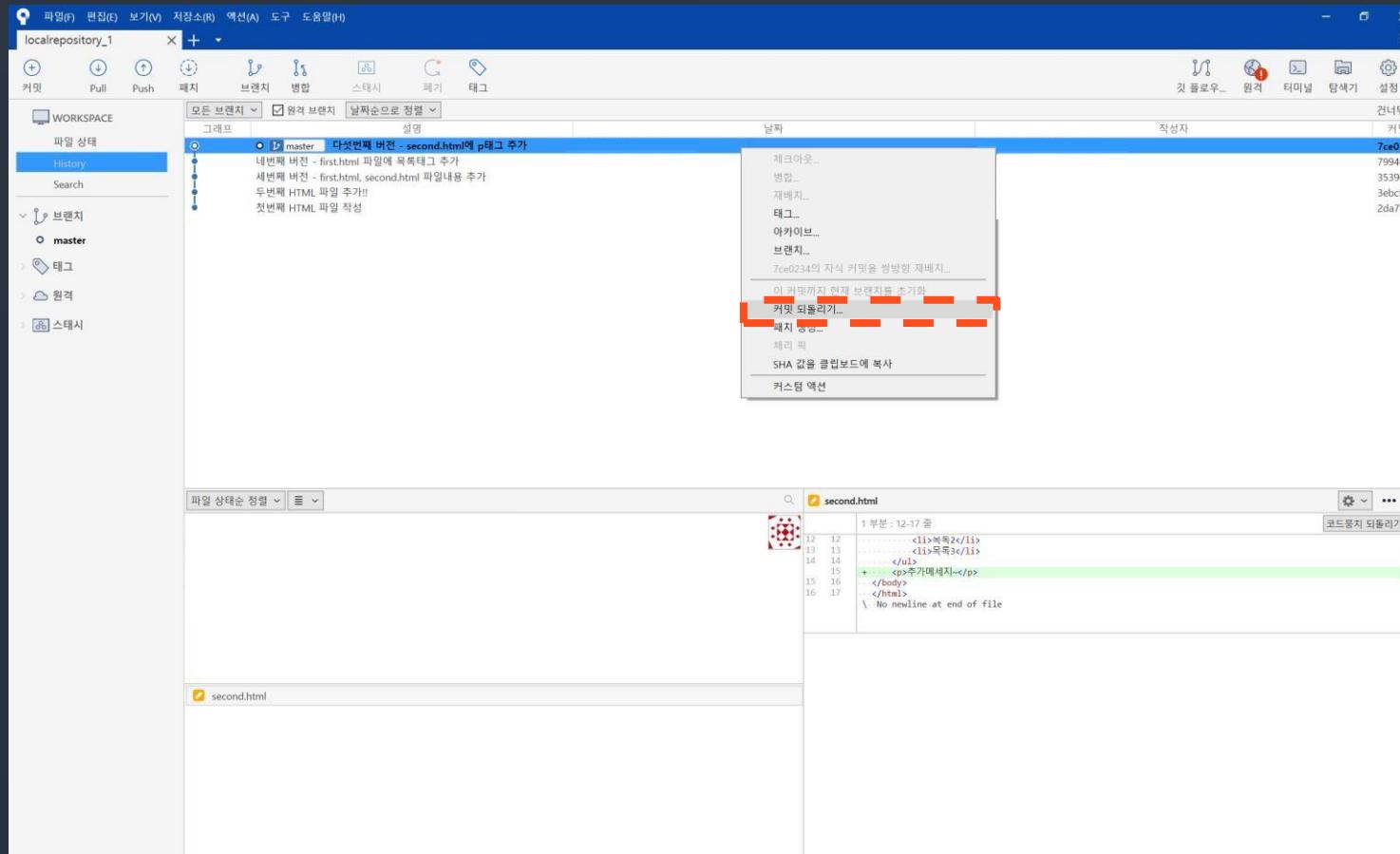
- second.html 파일에 p태그 추가 후 다섯번째 버전 생성



```
second.html > html
1  <!DOCTYPE html>
2  <html lang="kr">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Document</title>
7  </head>
8  <body>
9      <h1>두번째 HTML파일</h1>
10     <ul>
11         <li>목록1</li>
12         <li>목록2</li>
13         <li>목록3</li>
14     </ul>
15     <p>추가메세지~</p>
16 </body>
17 </html>
```

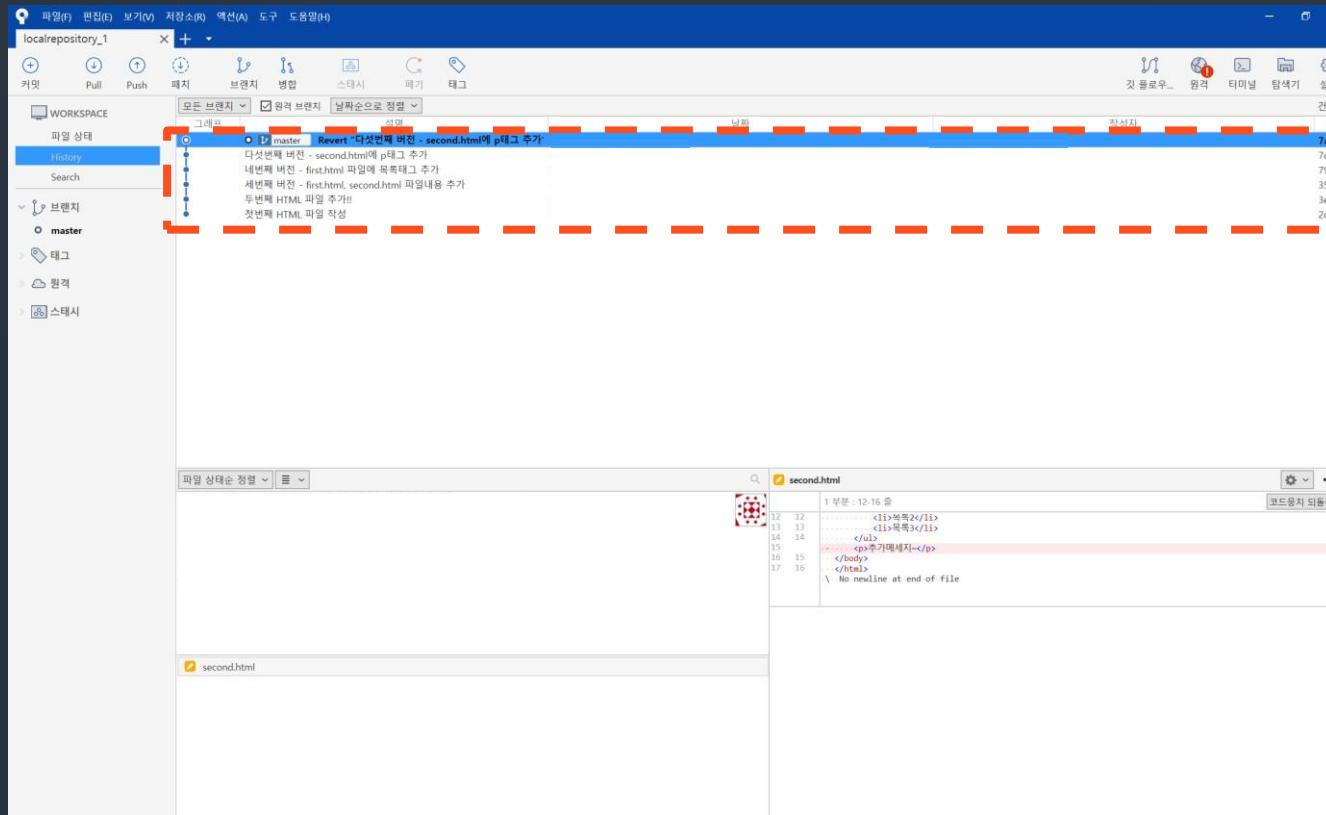
변경사항 되돌리기-커밋후(revert)

- 다섯번째 버전에서 마우스 오른쪽 클릭 후 '커밋 되돌리기...' 선택(가장 최신 버전에서 오른쪽 클릭)



변경사항 되돌리기-커밋후(revert)

- 5번째 커밋에 적용되었던 내용이 초기화 됨(second.html 파일의 p태그가 없어짐)
- History에 커밋을 되돌린 기록이 남음
- 만약 여러 단계를 되돌리고 싶은 경우 순차적으로 하나씩 revert를 수행(선택한 커밋만 되돌리는 가능)



The screenshot shows a code editor displaying the second.html file. The content has been partially removed or modified, specifically the list items and the closing body tag, illustrating the effect of reverting a previous commit.

```
<!DOCTYPE html><html lang="kr"><head><meta charset="UTF-8"><meta name="viewport" content="width=device-width, initial-scale=1.0"><title>Document</title></head><body><h1>두번째 HTML파일</h1><ul><li>목록1</li><li>목록2</li><li>목록3</li></ul></body></html>
```

폐기 / reset / revert

구분	설명
폐기	<ul style="list-style-type: none">- 아직 커밋 되지 않은 변경사항을 되돌림- 버전에 아무 영향을 주지 않고 작업사항을 취소
reset	<ul style="list-style-type: none">- 커밋된 변경사항을 되돌림- 해당하는 시점으로 되돌아가는 개념으로 history에 남지 않음- soft, mixed, hard 옵션이 존재
revert	<ul style="list-style-type: none">- 커밋된 변경사항을 되돌림- 선택한 버전의 변경사항을 되돌리며, 되돌린 상태를 새로운 버전으로 History 관리- 원격저장소 연동 시 reset을 사용하지 않고, revert만 사용

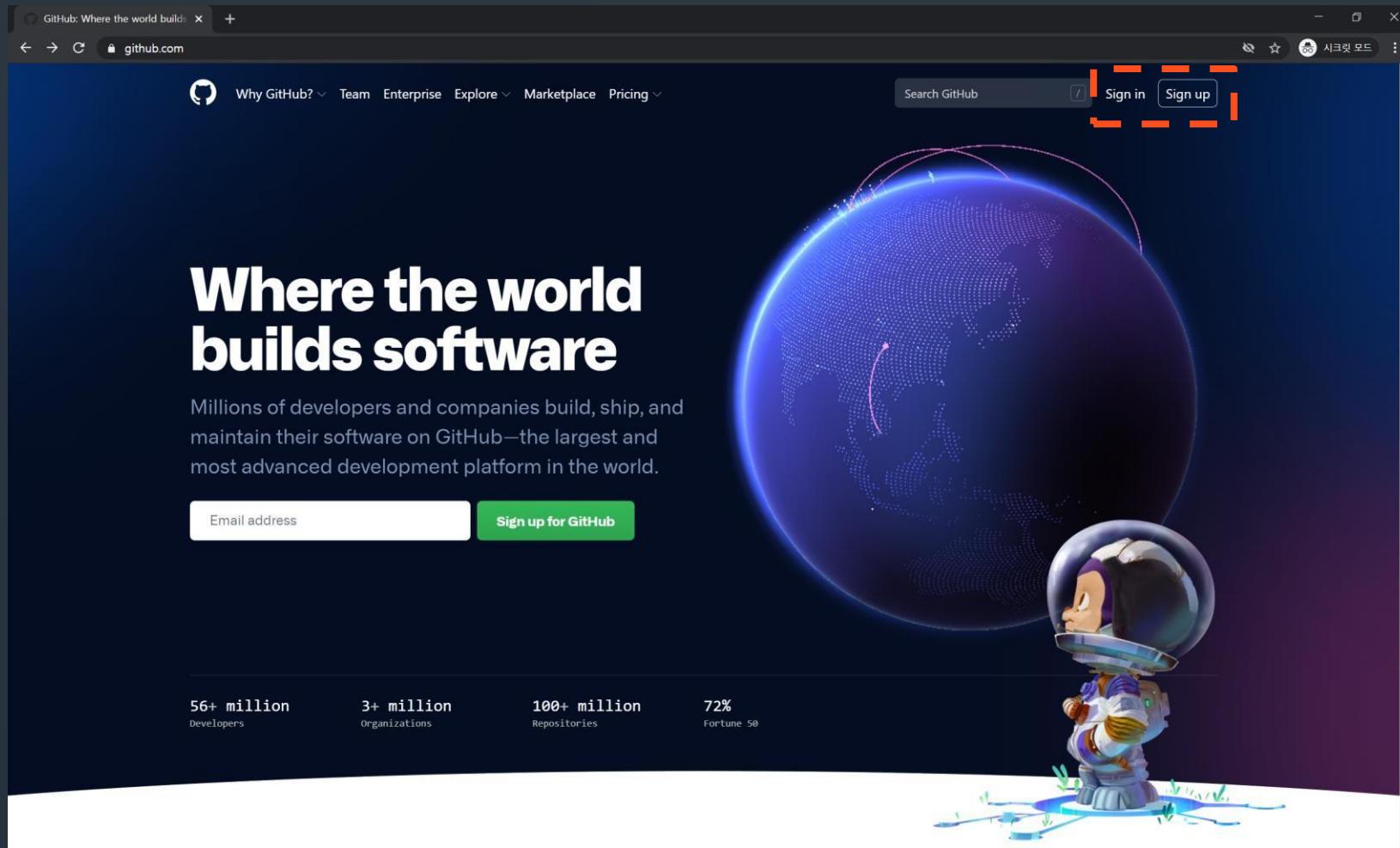




원격 저장소 관리

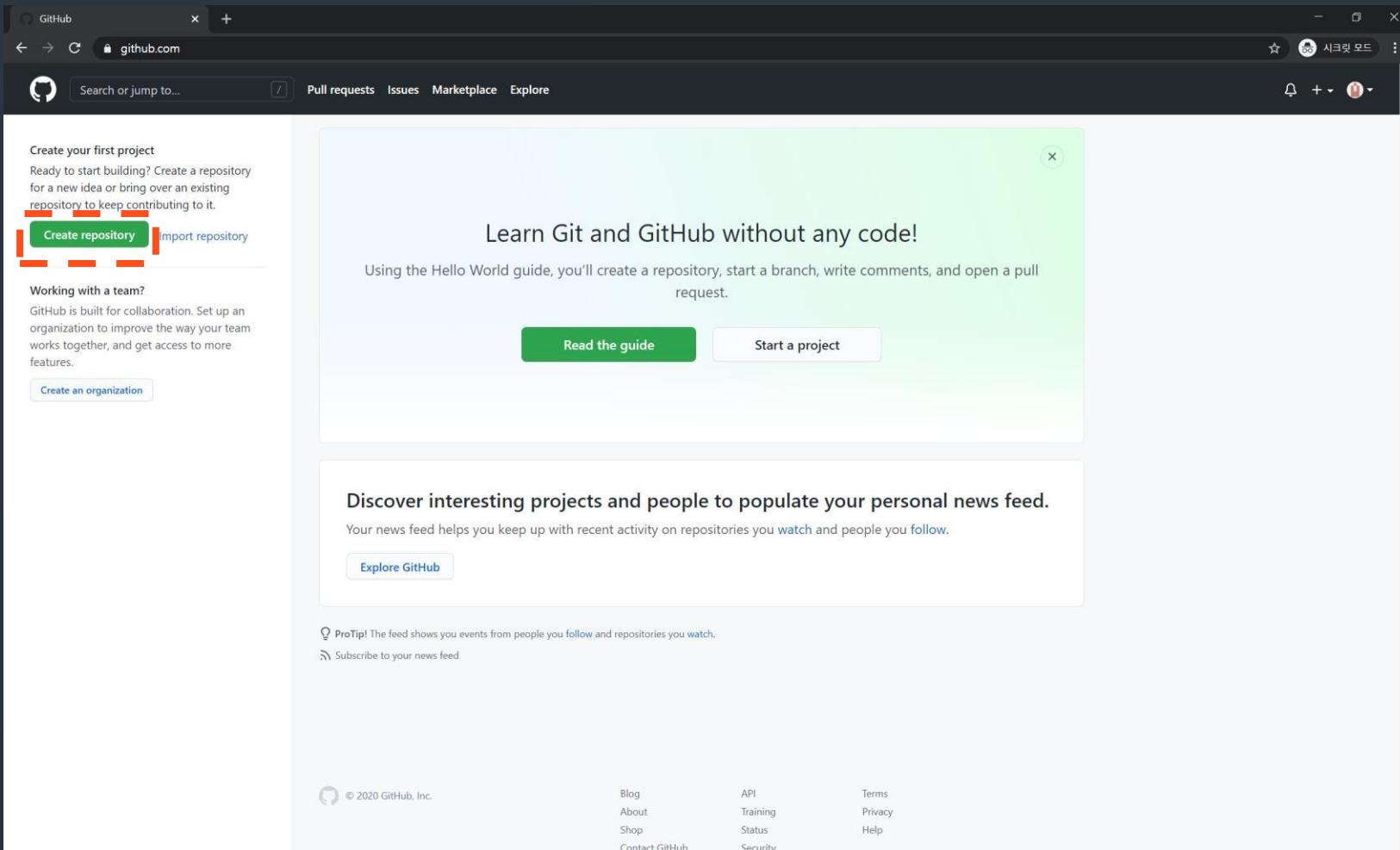
Github 회원가입

- 원격저장소 사용을 위한 Github 회원 가입 및 로그인
- <https://github.com/>



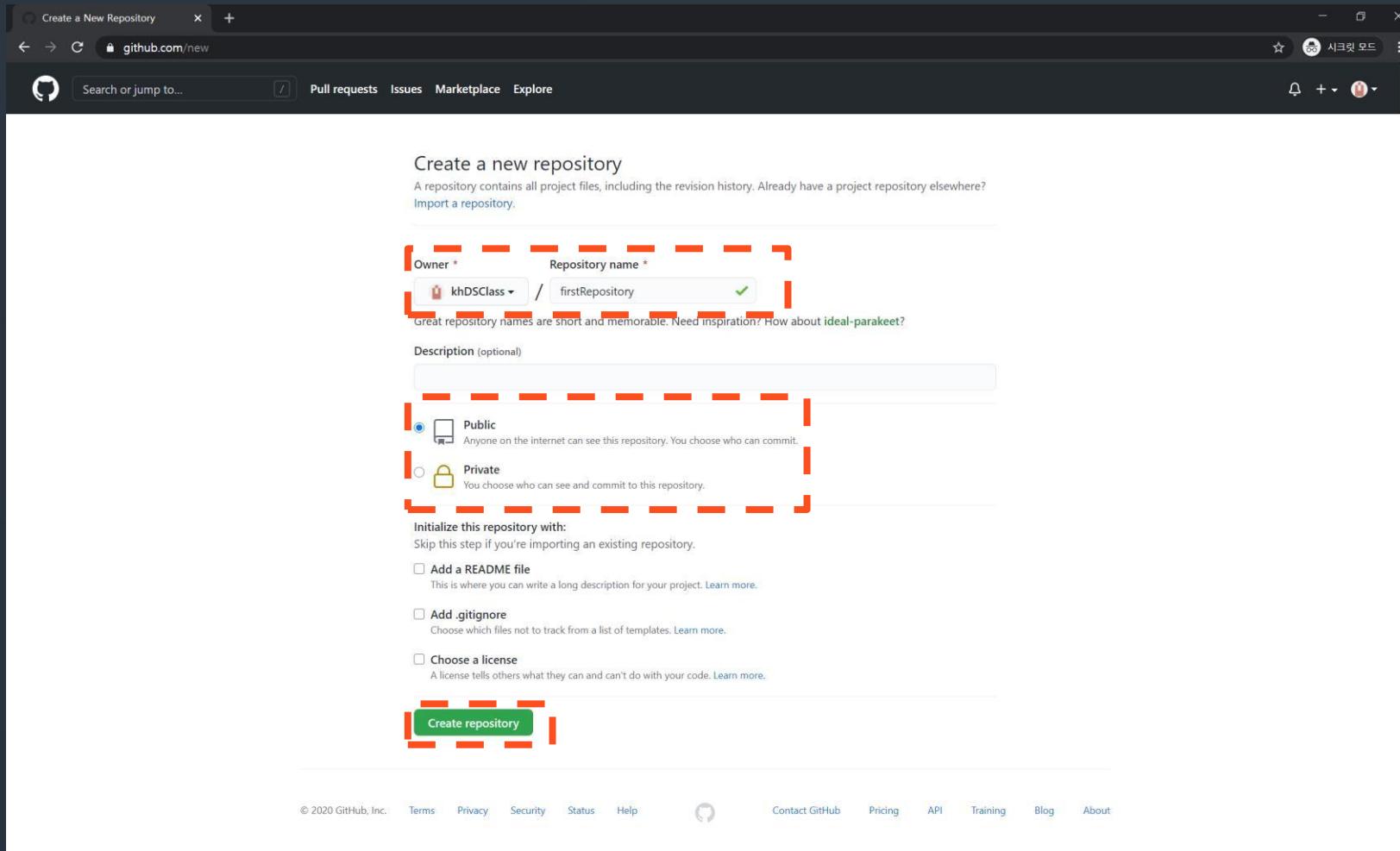
원격저장소 생성

- 로그인후 원격저장소 생성



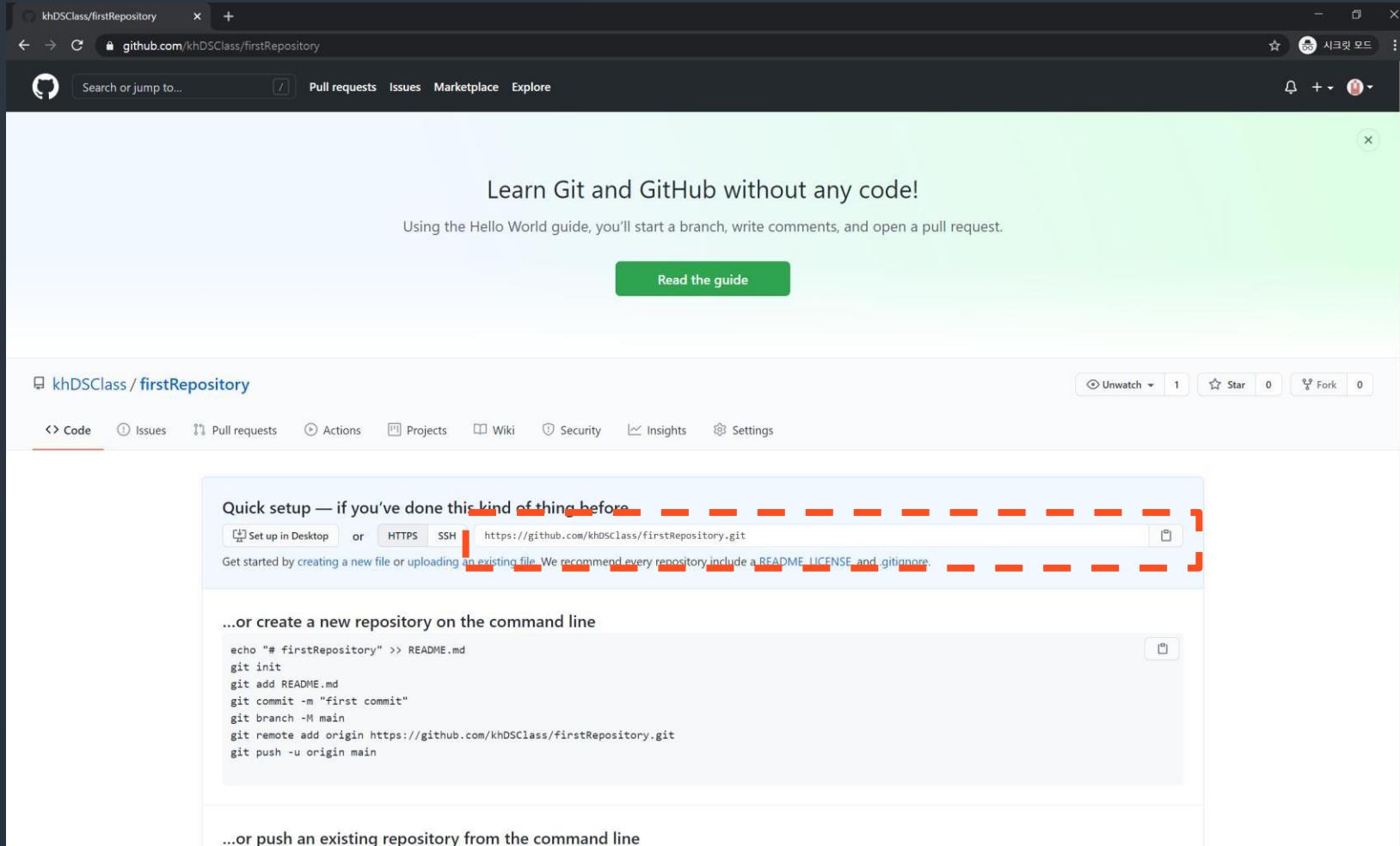
원격저장소 생성

- 원격저장소 이름 지정
- public은 공개저장소로 다른 사람들과 협업 가능 / private는 비공개 저장소



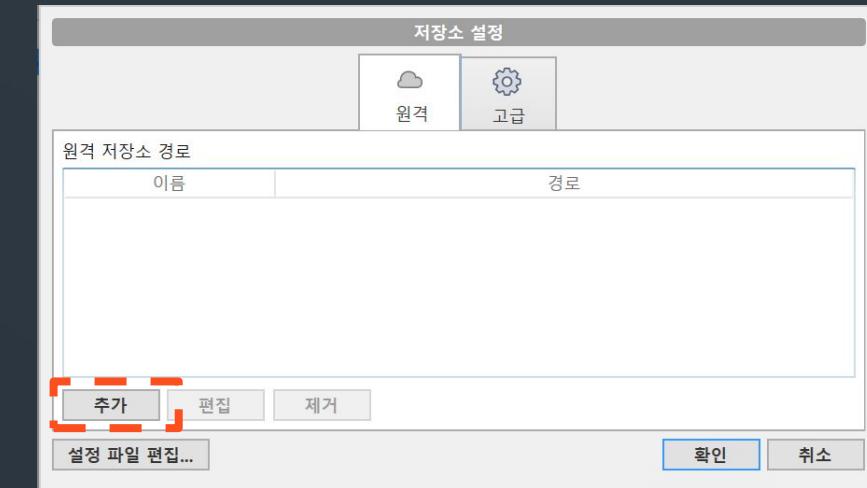
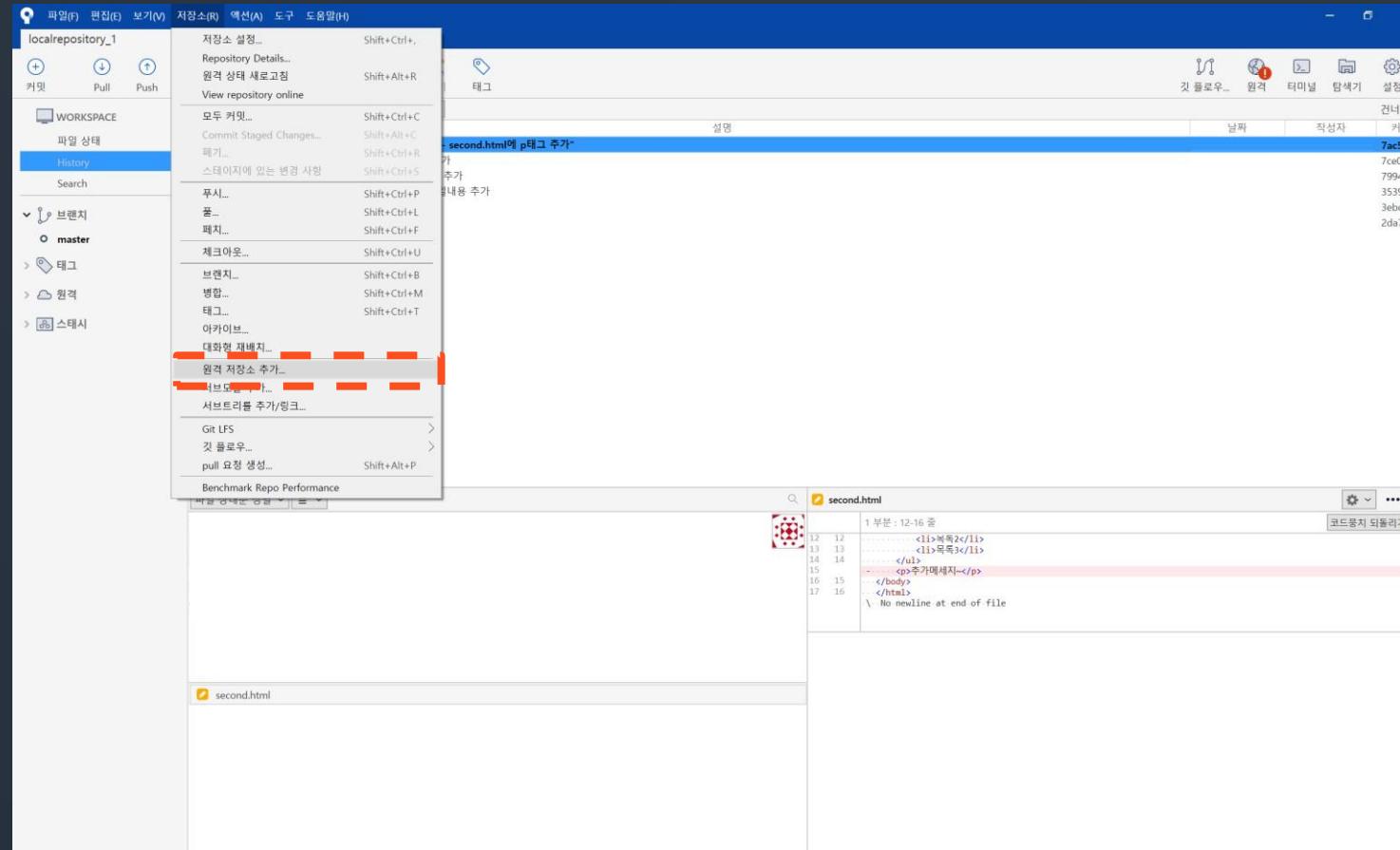
원격저장소 생성완료

- 원격저장소 생성 완료된 화면
- 원격저장소 주소를 통해 로컬저장소와 원격저장소를 동기화



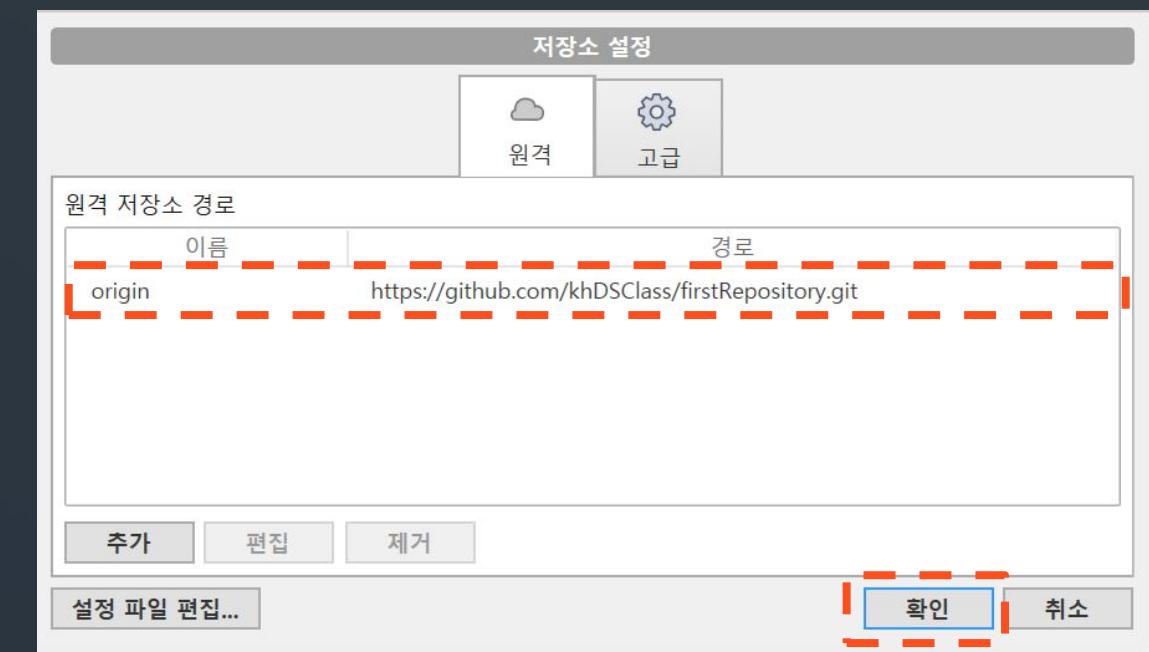
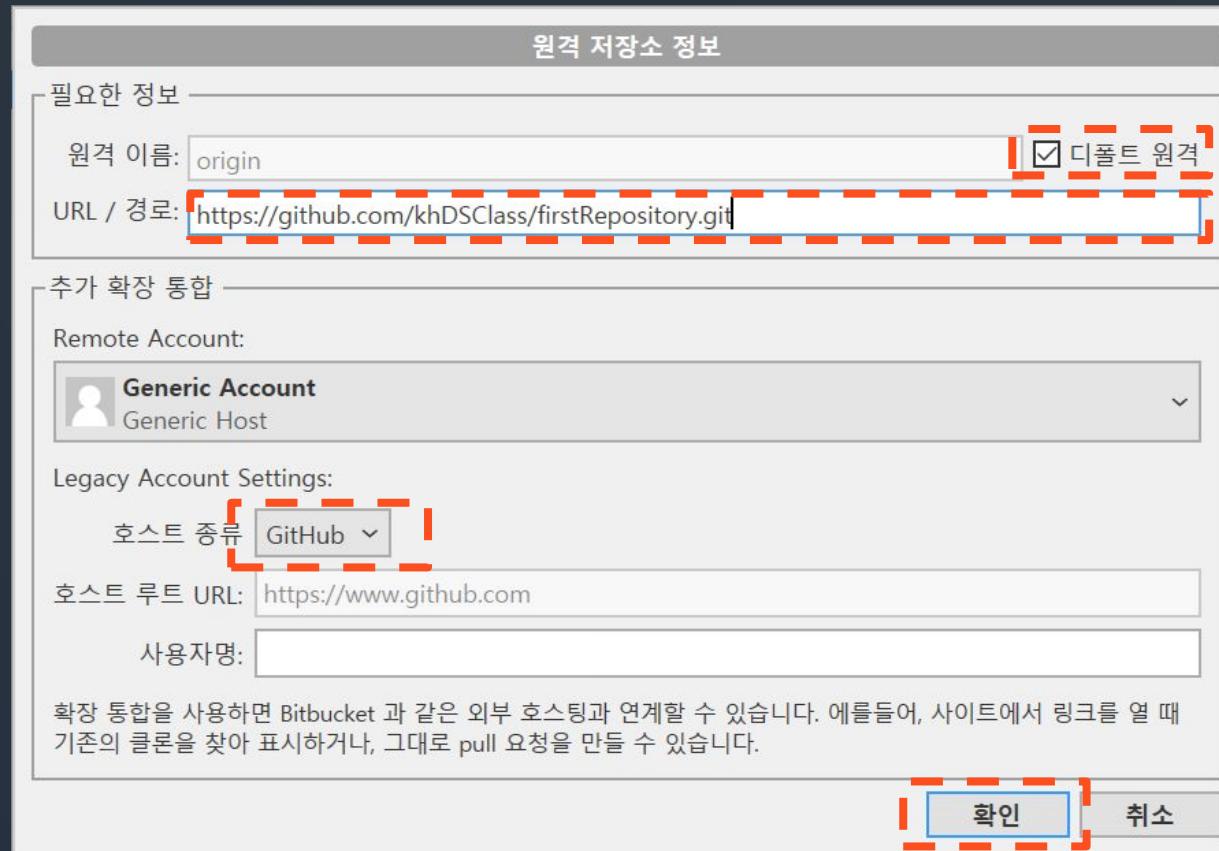
Sourcetree에 원격저장소 추가

- 저장소 - 원격저장소 추가
- 저장소 설정 - 추가



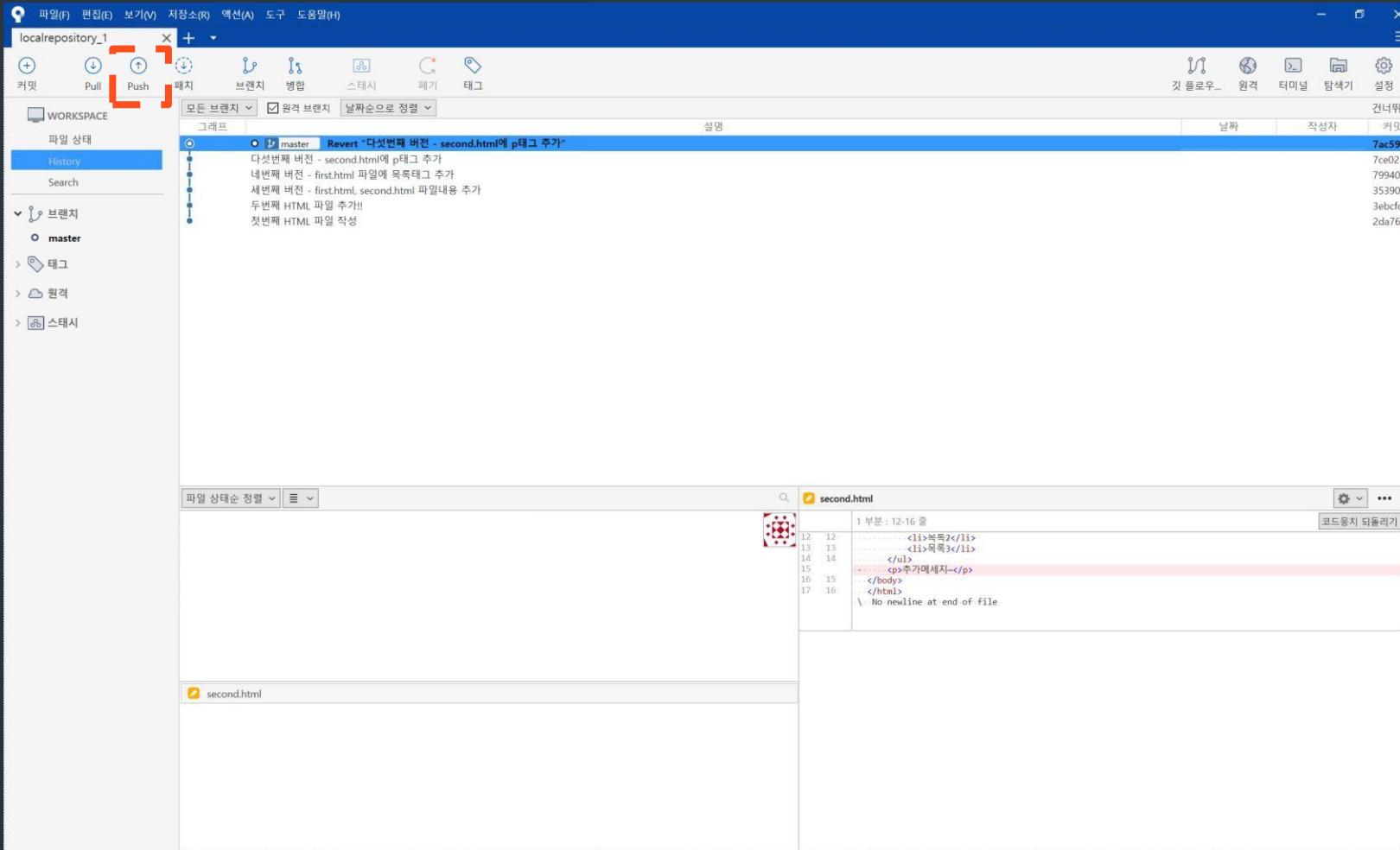
Sourcetree에 원격저장소 추가

- URL/경로에 github 원격저장소 추가 후 확인



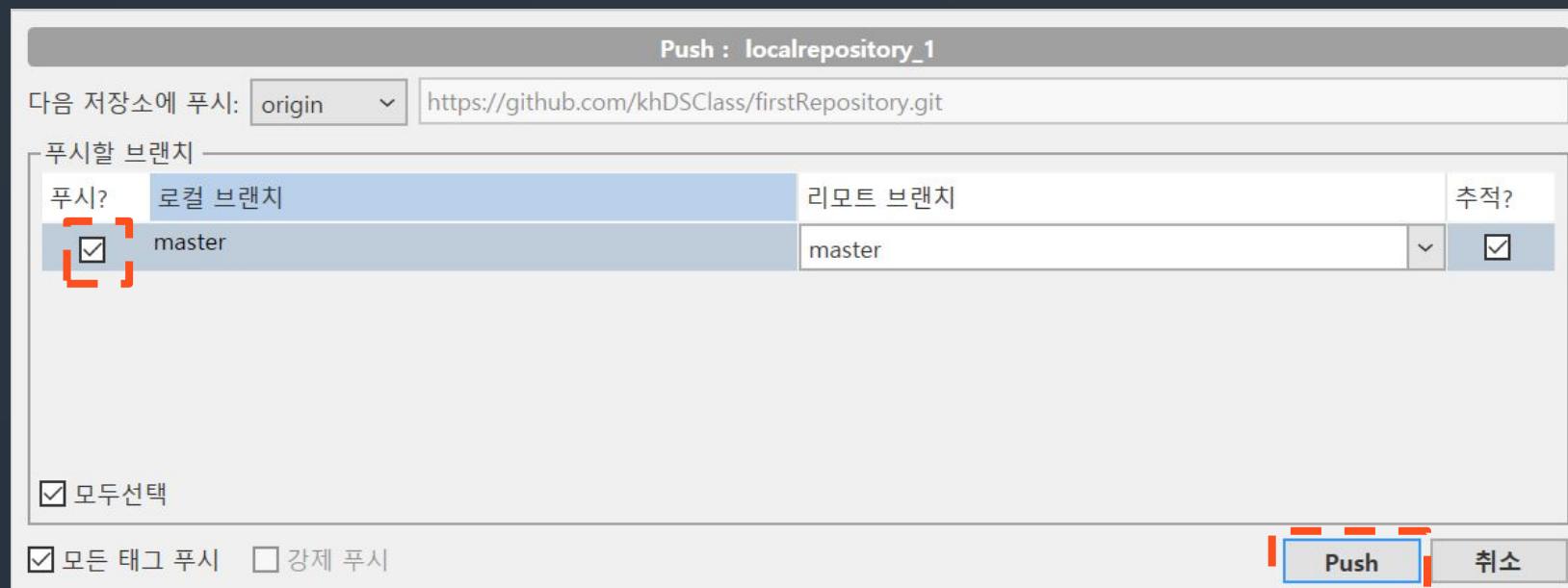
로컬저장소와 원격저장소 동기화

- push를 통해 원격저장소의 상태를 로컬저장소의 상태와 같은 상태로 동기화



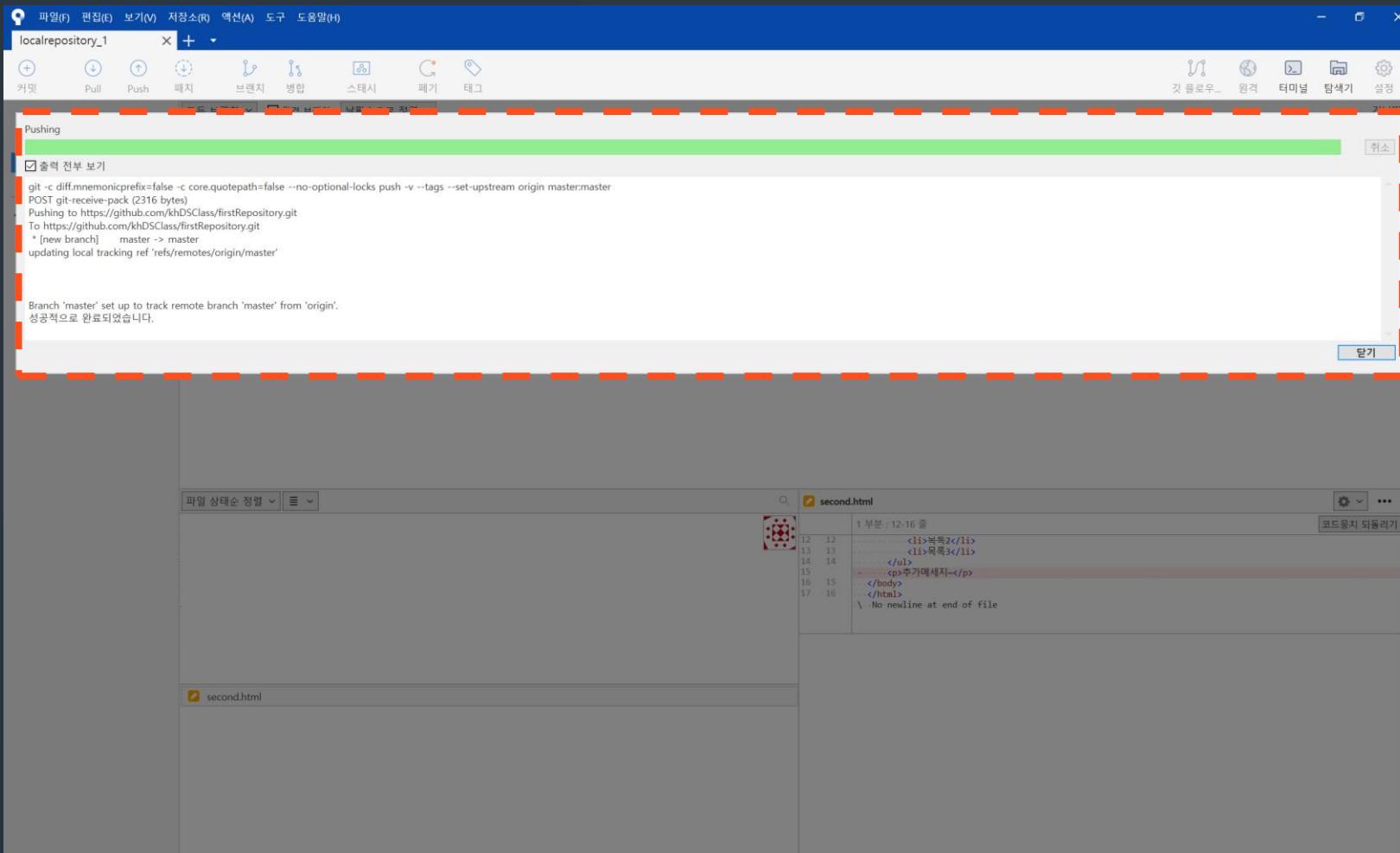
로컬저장소와 원격저장소 동기화

- master 브랜치를 push(브랜치는 이후에 진행)
- github 계정과 비밀번호 입력(한번 입력하면 저장하여 사용 가능)



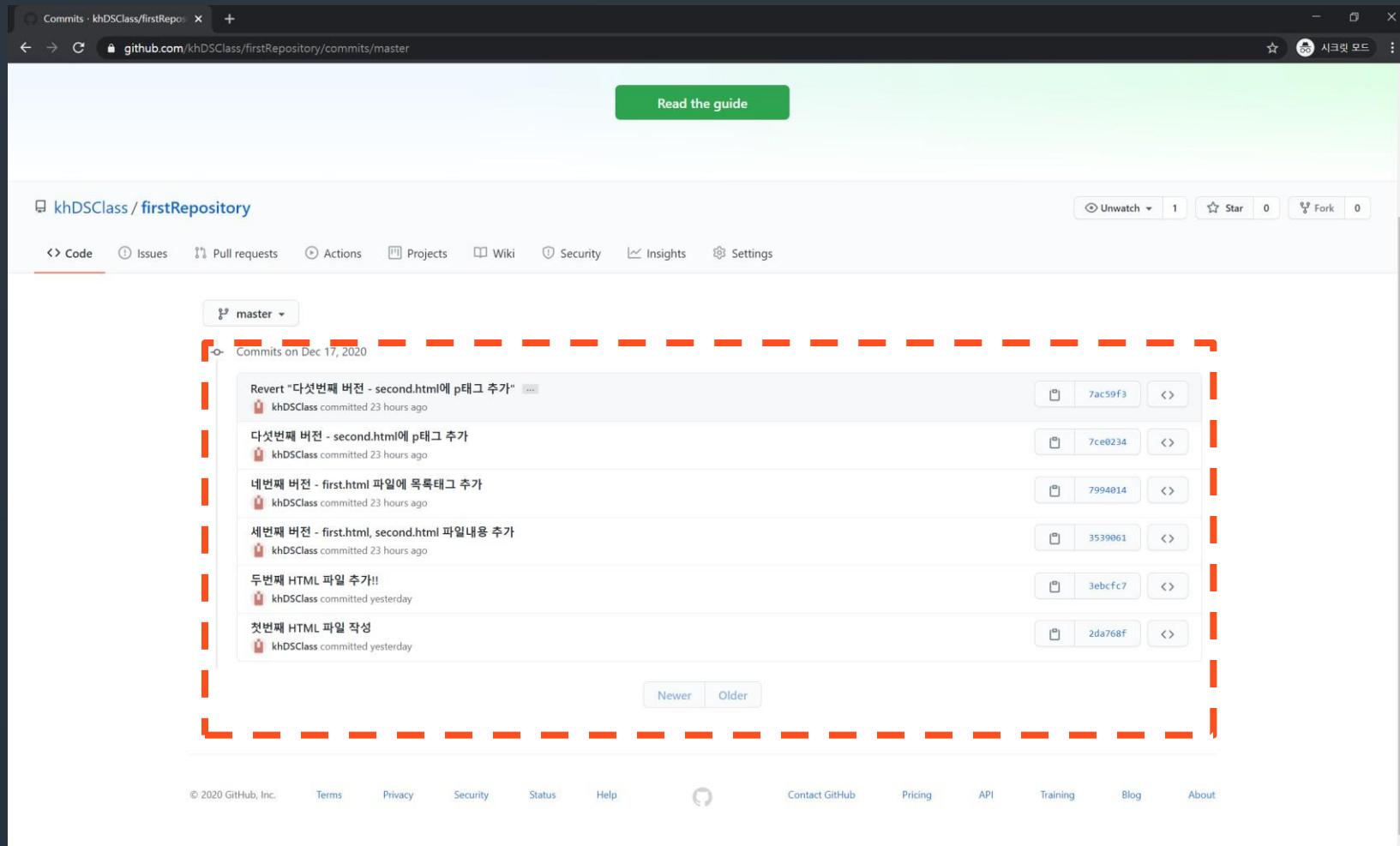
로컬저장소와 원격저장소 동기화 성공화면

- push성공 메세지



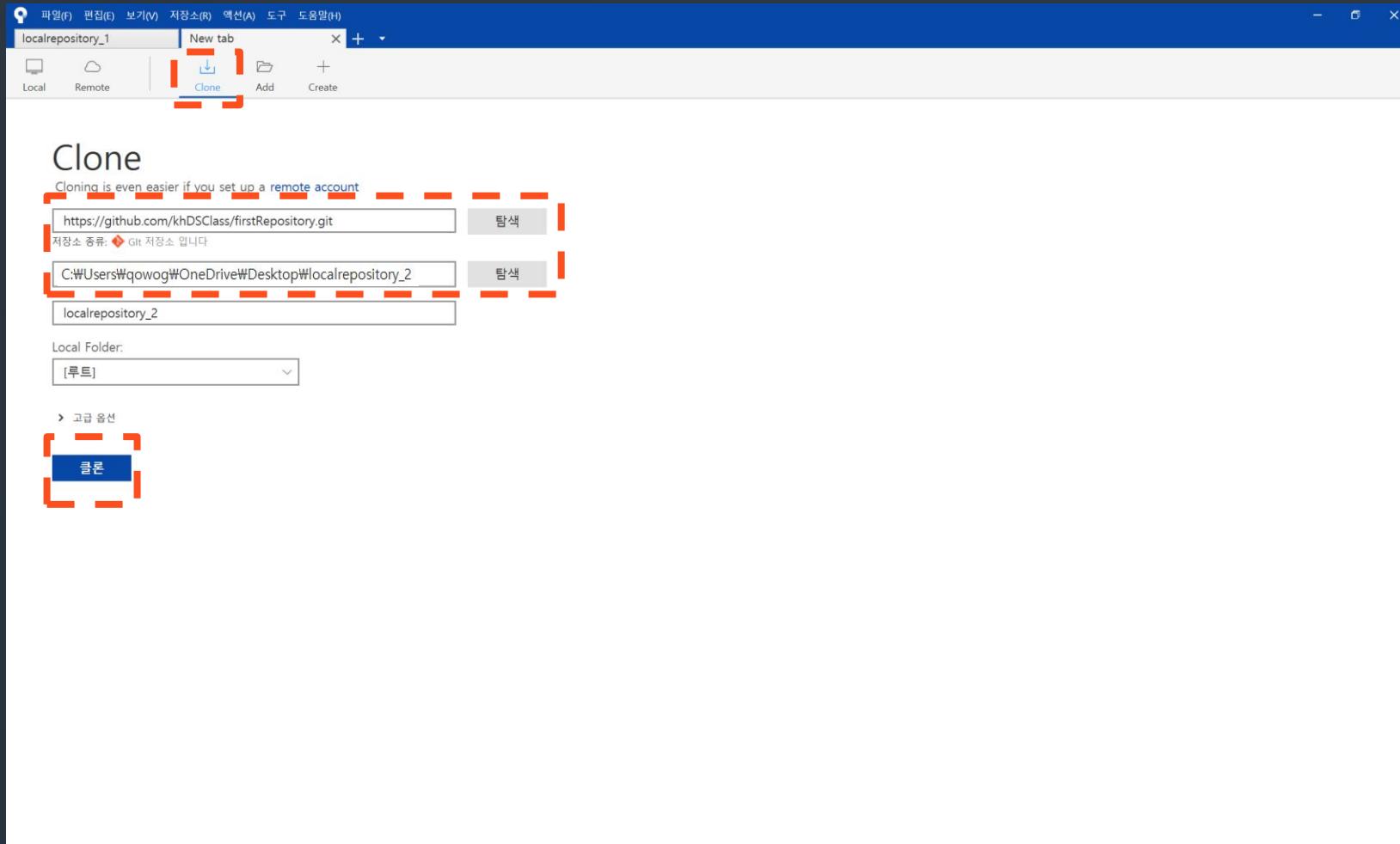
원격저장소 상태 확인

- 로컬저장소의 history와 동일한 history 생성완료



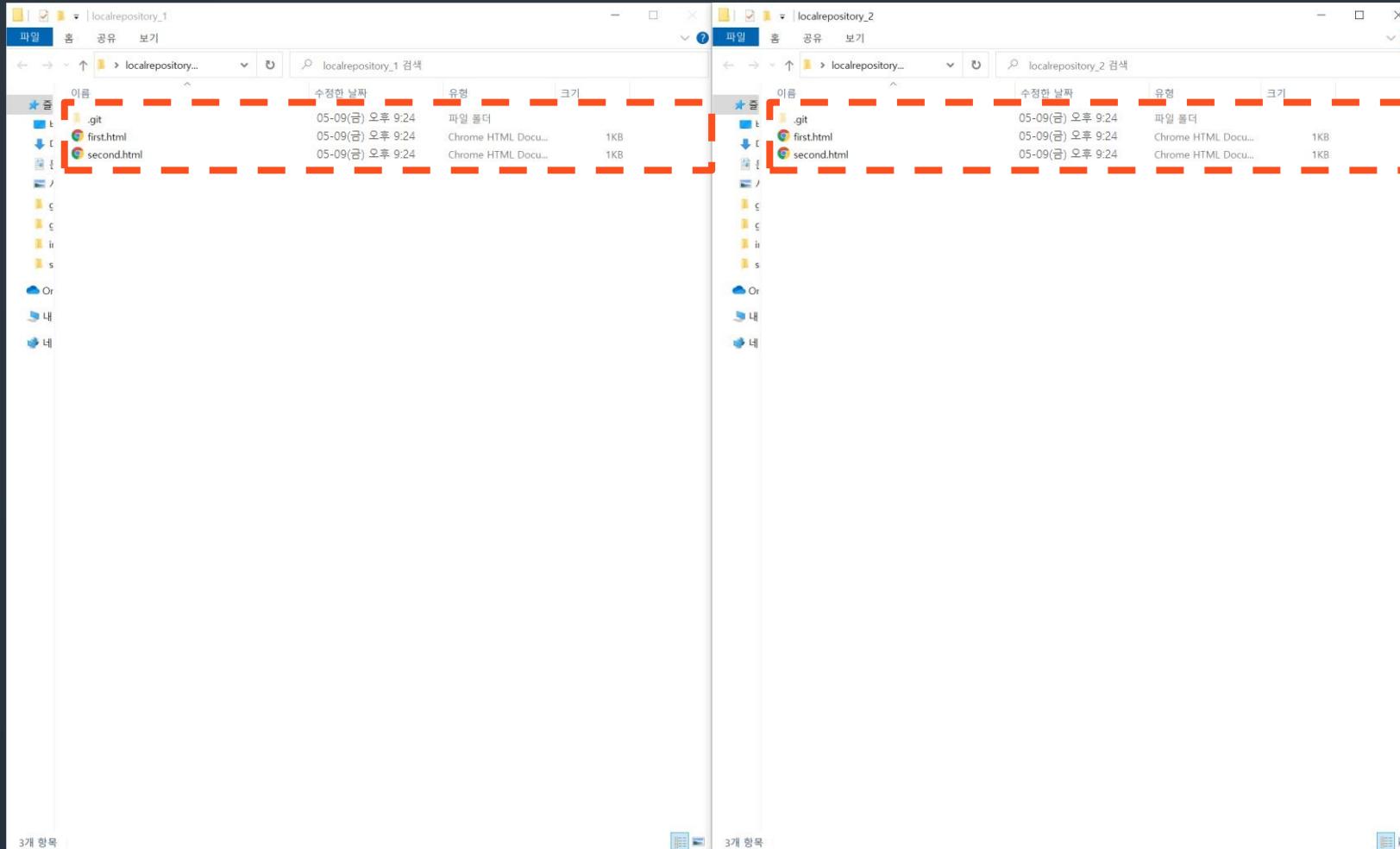
원격저장소의 데이터 로컬저장소로 가져오기

- Clone을 통해서 원격저장소의 데이터를 로컬저장소로 복사(최초 1번)
- 원격저장소 주소 입력 / 로컬저장소 경로 입력



원격저장소의 데이터 로컬저장소로 가져오기

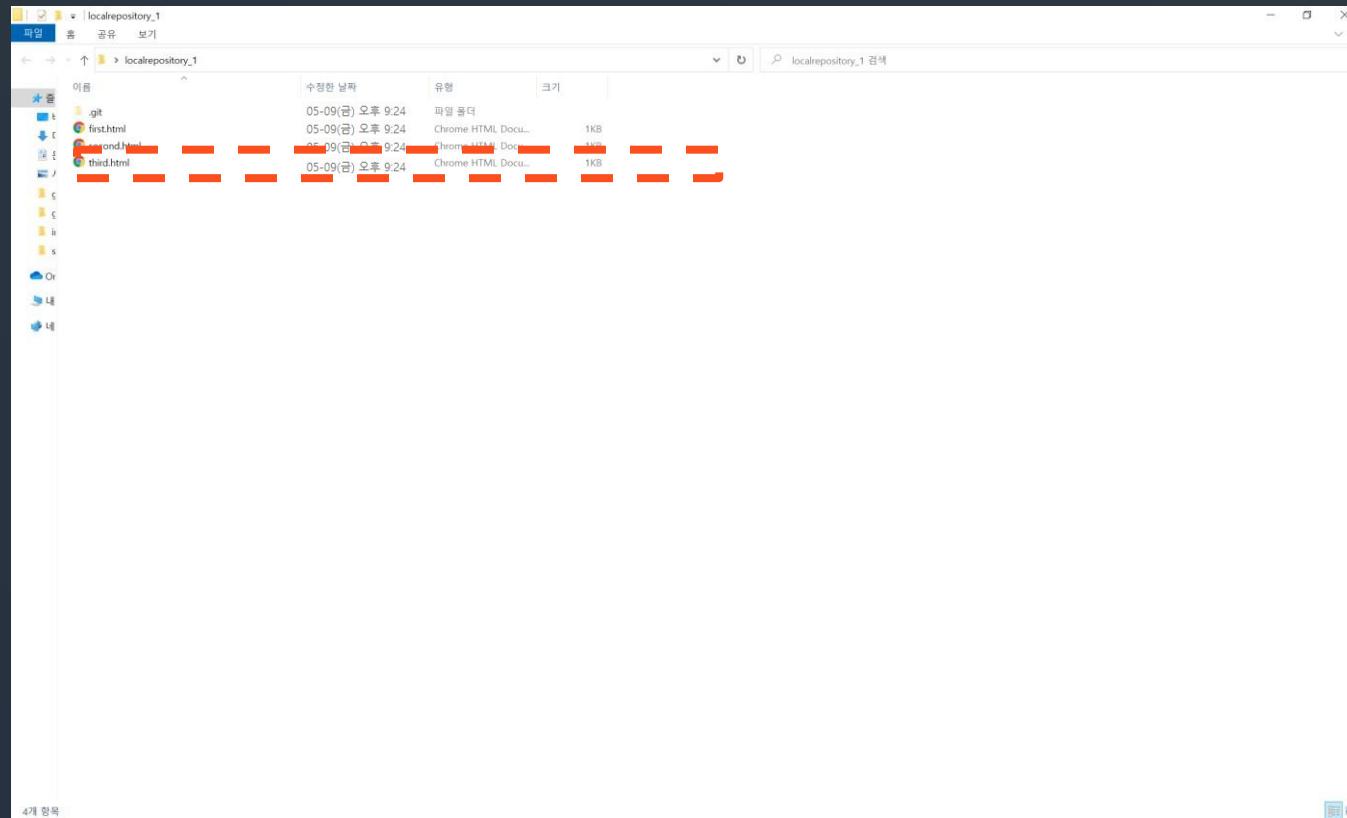
- **localrepository_1과 localrepository_2 폴더 내부의 파일이 동일한 상태**
- **다른 PC에서도 동일하게 동작**





로컬저장소에 추가파일 생성

- localrepository_1에 third.html 파일을 생성
- 다른 PC에서도 동일하게 동작

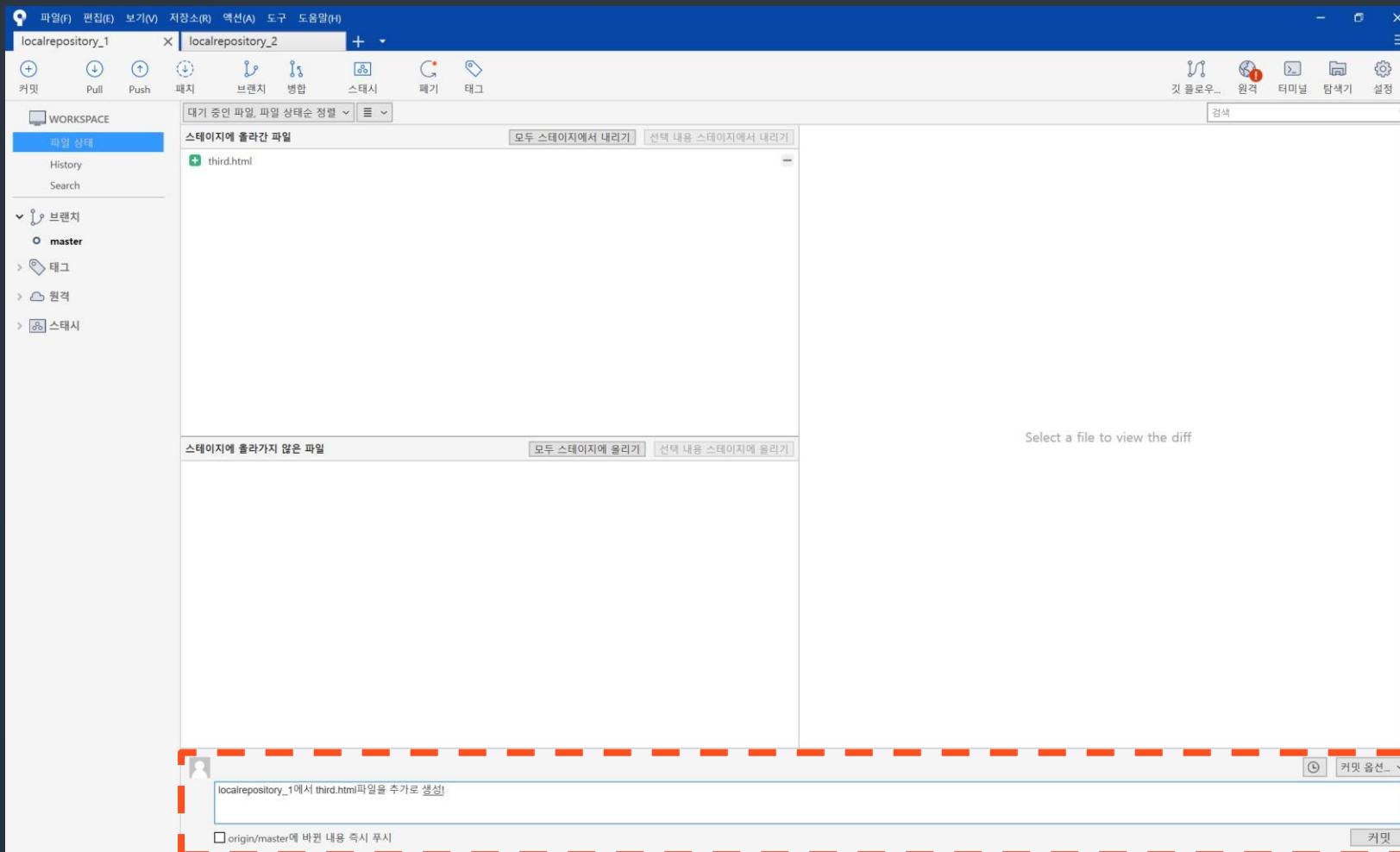


```
third.html > html
1   <!DOCTYPE html>
2   <html lang="kr">
3     <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Document</title>
7     </head>
8     <body>
9       <h1>세 번째 HTML 파일 생성</h1>
10    </body>
11  </html>
```



로컬저장소에 추가파일 생성

- localrepository_1에 third.html 파일을 생성
- localrepository_1에 커밋을 통한 새로운 버전 생성





로컬저장소에 새로운 버전 생성

- **localrepository_1**에 새로운 버전이 생성완료
- 로컬저장소에는 새로운 버전이 생성되었으나, 원격저장소에는 아직 적용하지 않은 상태

The screenshot shows the SourceTree application interface. At the top, there are tabs for '파일(F)', '편집(E)', '보기(V)', '저장소(R)', '액션(A)', '도구(G)', and '도움말(H)'. The current tab is '저장소(R)'. Below the tabs, there are buttons for '커밋(C)', 'Pull(P)', 'Push(P)', '페이지(P)', '브랜치(B)', '병합(M)', '스테시(S)', '태그(T)', and '설정(S)'. A red box highlights the '커밋(C)' button.

The main area is titled 'localrepository_1' and shows a timeline of commits. A blue box highlights the most recent commit:

```
localrepository_1에서 third.html 파일을 추가로 생성!
origin/master Revert "다섯번째 버전 - second.html에 p태그 추가"
다섯번째 버전 - second.html에 p태그 추가
네번째 버전 - first.html 파일에 목록태그 추가
세번째 버전 - first.html, second.html 파일내용 추가
두번째 HTML 파일 추가!!
첫번째 HTML 파일 작성
```

The commit hash for this latest commit is 9352c12. To the right of the commit list, there are icons for '깃 플로우(G)', '원격(R)', '터미널(T)', '탐색기(S)', and '설정(S)'. A red box highlights the '깃 플로우(G)' icon.

On the left side, there's a sidebar with sections for 'WORKSPACE', '파일 상태(FS)', 'History(H)', 'Search(S)', '브랜치(B)', 'master', '태그(T)', '원격(R)', and '스테시(S)'. A red box highlights the 'History(H)' section.

At the bottom, there's a code editor window titled 'third.html' with the following content:

```
<!DOCTYPE html>
<html lang="kr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>세번째 HTML 파일 생성</h1>
</body>
</html>
```

A red box highlights the bottom right corner of the code editor window.

로컬저장소에 새로운 버전 생성

- 원격저장소 및 localrepository_2에는 새로 만든 버전이 적용되어있지 않음
- localrepository_1 → 원격저장소 / 원격저장소 → localrepository_2 순서로 버전 적용

The screenshot shows the GitHub commit history for the repository 'khDSClass / firstRepository'. The 'master' branch has several commits:

- Revert "다섯번째 버전 - second.html에 p태그 추가" (7ac59f3)
- 다섯번째 버전 - second.html에 p태그 추가 (7ce0234)
- 네번째 버전 - first.html 파일에 목록태그 추가 (7994014)
- 세번째 버전 - first.html, second.html 파일내용 추가 (3539061)
- 두번째 HTML 파일 추가!! (3ebcfc7)
- 첫번째 HTML 파일 작성 (2da768f)

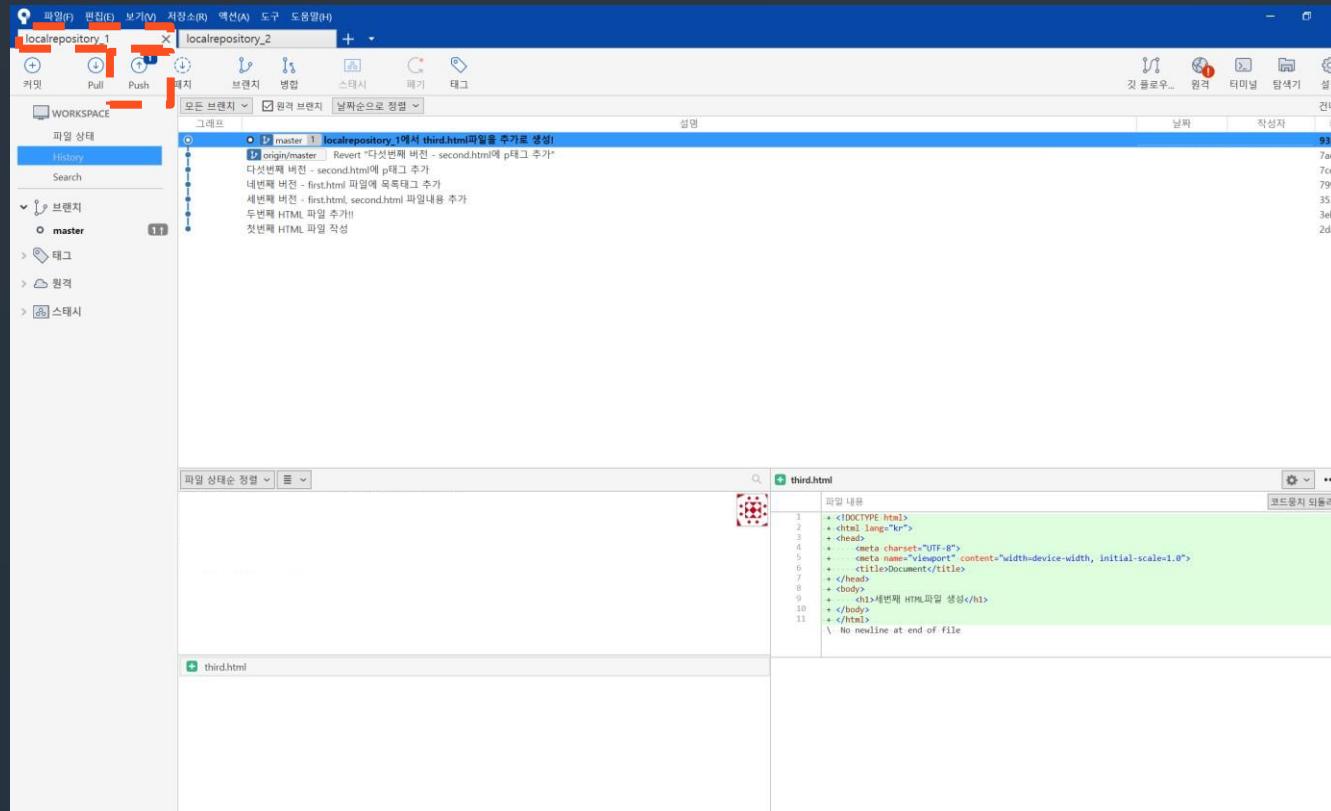
The screenshot shows the SourceTree interface with two repositories: 'localrepository_1' and 'localrepository_2'. The 'localrepository_2' tab is active, showing the commit history:

- origin/master (7ac59f3) - Revert "다섯번째 버전 - second.html에 p태그 추가"
- origin/HEAD (7ce0234) - 네번째 버전 - first.html 파일에 목록태그 추가
- origin/HEAD (3539061) - 세번째 버전 - first.html, second.html 파일내용 추가
- origin/HEAD (3ebcfc7) - 두번째 HTML 파일 추가!!
- origin/HEAD (2da768f) - 첫번째 HTML 파일 작성

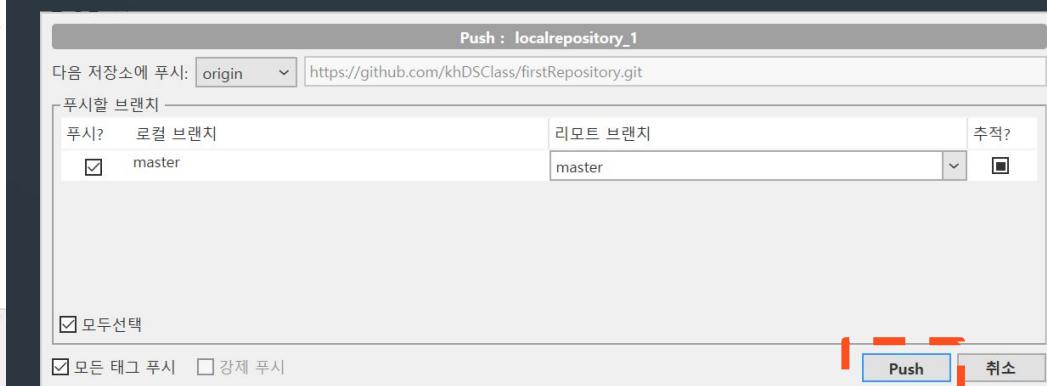
A detailed view of the 'second.html' file is shown in the bottom right, comparing the current state (12-16) with the previous state (11-15). The diff highlights changes in line 13 and 14.

localrepository_1 → 원격저장소로 새로운 버전 적용

- push를 통해서 원격저장소에 새로운 버전 적용

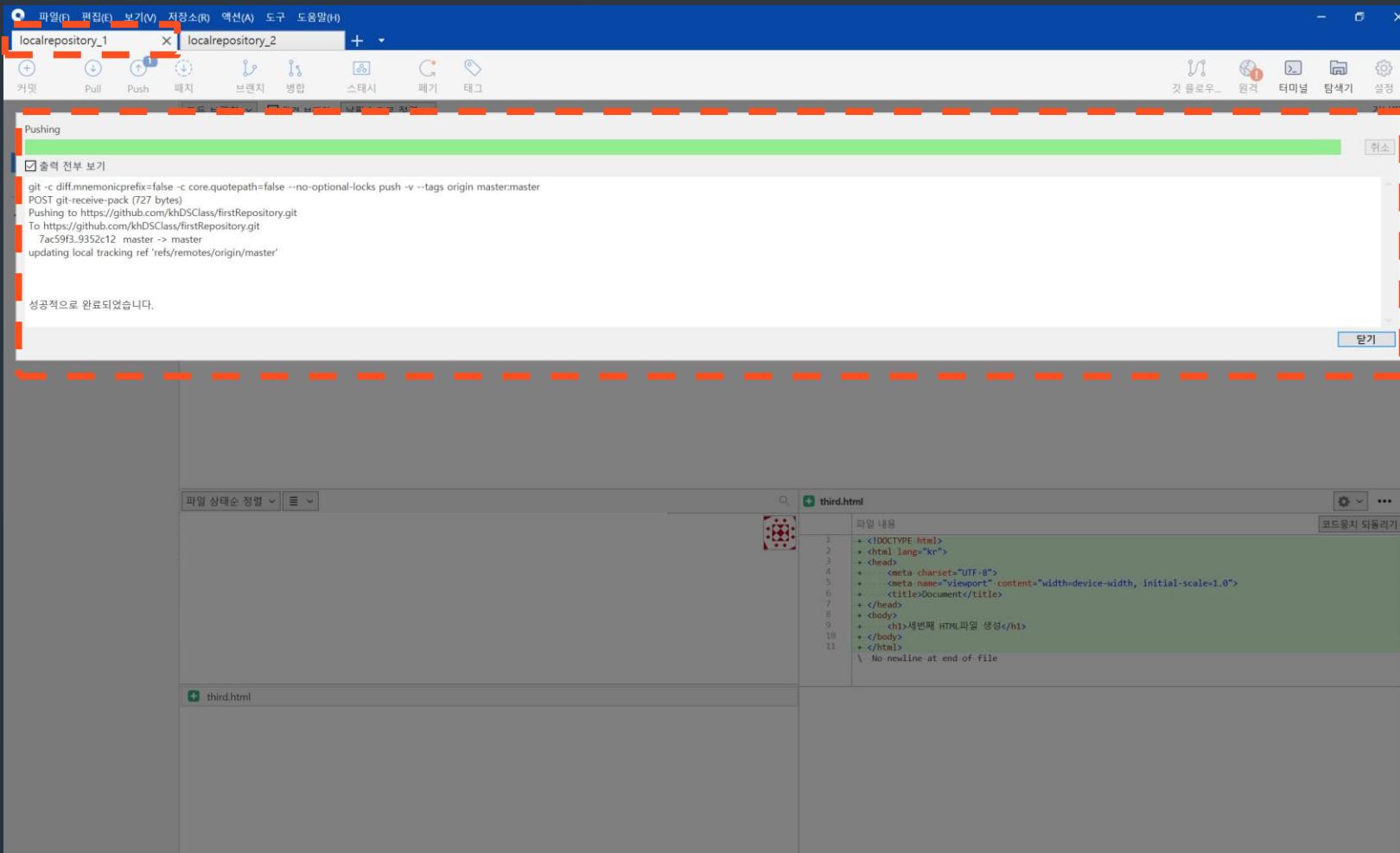


```
<!DOCTYPE html>
<html lang="kr">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Document</title>
</head>
<body>
<h1>세번째 HTML 파일 생성</h1>
</body>
</html>
```



로컬저장소와 원격저장소 동기화 성공화면

- push성공 메세지



원격저장소 및 localrepository_2 상태 확인

- 원격저장소에 새로운 버전 적용 완료
- 아직 localrepository_2에는 적용되지 않은

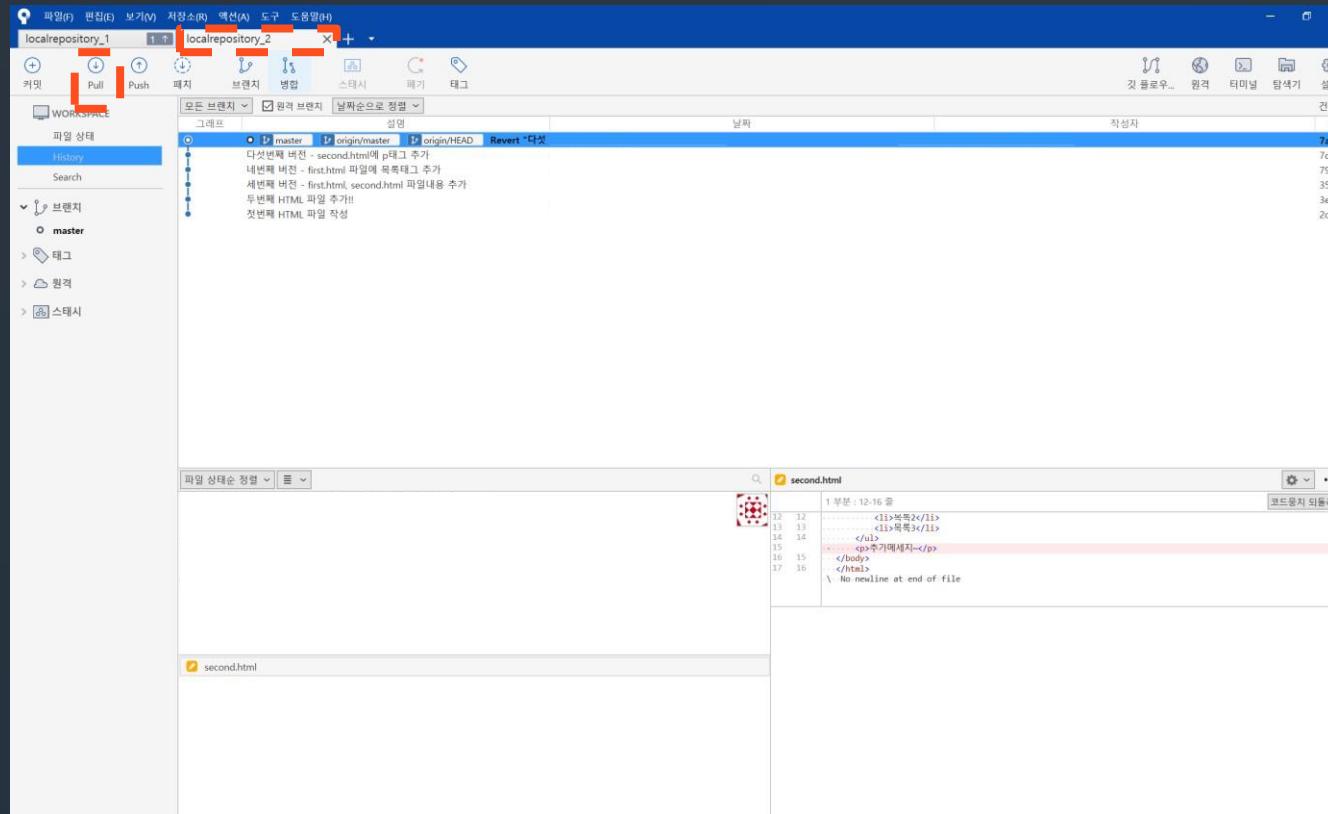
The screenshot shows the GitHub commit history for the repository 'khDSClass / firstRepository'. The 'Code' tab is selected. A red box highlights the commit 'localrepository_1에서 third.html 파일을 추가로 생성!' made by 'khDSClass' 14 minutes ago. Below it, several other commits are listed, all made by 'khDSClass' 2 days ago, including 'Revert "다섯번째 버전 - second.html에 p태그 추가"', '다섯번째 버전 - second.html에 p태그 추가', '네번째 버전 - first.html 파일에 목록태그 추가', '세번째 버전 - first.html, second.html 파일내용 추가', '두번째 HTML 파일 추가!', and '첫번째 HTML 파일 작성'. At the bottom, there are 'Newer' and 'Older' buttons.

The screenshot shows a Git client interface with two repositories: 'localrepository_1' and 'localrepository_2'. A red box highlights the 'localrepository_2' tab. The 'localrepository_2' window shows the 'History' tab with a list of commits identical to those on GitHub. It also shows the 'Workspace' tab with a 'second.html' file open in a code editor. The code editor shows the following content:

```
12 12 <li>목록2</li>
13 13 <li>목록3</li>
14 14 <p>추가메시지</p>
15 15 </ul>
16 16 </li>
17 17 </ul>
\ No newline at end of file
```

원격저장소 → localrepository_2 새로운 버전 적용

- pull을 이용하여 원격저장소의 최신버전을 로컬저장소에 적용



원격저장소와 로컬저장소 버전 동기화

- pull 성공 메세지

The screenshot shows a Windows terminal window titled "Pulling" with the following command history:

```
git -c diff.mnemonicprefix=false -c core.quotepath=false --no-optional-locks fetch origin
From https://github.com/khDSClass/firstRepository
 7ac59f3..9352c12  master -> origin/master

git -c diff.mnemonicprefix=false -c core.quotepath=false --no-optional-locks pull origin master
From https://github.com/khDSClass/firstRepository
 * branch            master    -> FETCH_HEAD

Updating 7ac59f3..9352c12
Fast-forward

third.html | 11 ++++++++-
1 file changed, 11 insertions(+)
create mode 100644 third.html

성공적으로 원료되었습니다.
```

Below the terminal, a code editor window titled "second.html" displays the following code:

```
<ul>
  <li>복록2</li>
  <li>복록3</li>
</ul>
<p>pull 성공 메세지</p>
```



원격저장소와 로컬저장소 버전 동기화

- 로컬저장소에 버전 적용

The screenshot shows the SourceTree application interface. At the top, there are tabs for 'localrepository_1' and 'localrepository_2'. The 'localrepository_2' tab is active, indicated by a red box. Below the tabs are buttons for '커밋' (Commit), 'Pull', 'Push', '페이지' (Page), '브랜치' (Branch), '병합' (Merge), '스테시' (Stash), '폐기' (Delete), and '태그' (Tag). To the right of these buttons are icons for '깃 플로우...', '원격', '터미널', '탐색기', and '설정'.

The main area displays a timeline of commits for the 'master' branch. A red box highlights the commit history. The commits are:

- Revert "다섯번째 버전 - second.html에 p태그 추가"
- 다섯번째 버전 - second.html 파일에 p태그 추가
- 네번째 버전 - first.html 파일에 목록태그 추가
- 세번째 버전 - first.html, second.html 파일내용 추가
- 두번째 HTML 파일 추가!!
- 첫번째 HTML 파일 작성

On the far right, a list of commit IDs is shown, each preceded by a '커밋' (Commit) label:

- 커밋 9352c12
- 커밋 7ac59f3
- 커밋 7ce0234
- 커밋 7994014
- 커밋 3539061
- 커밋 3ebcfc7
- 커밋 2da768f

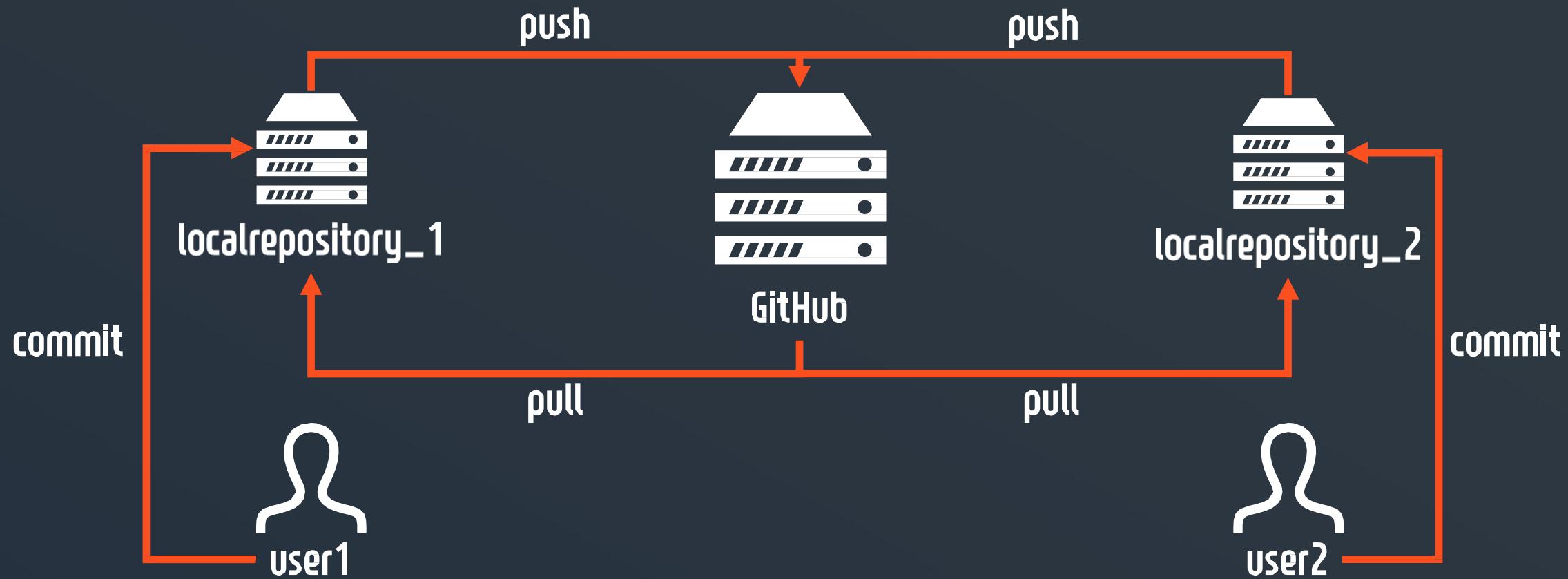
Below the commit history, there is a 'History' section with a 'Search' input field. The sidebar on the left includes sections for 'WORKSPACE', '파일 상태', 'History' (which is selected and highlighted with a red box), 'Search', '브랜치' (with 'master' selected), '태그', '원격', and '스테시'.

At the bottom, there is a code editor window titled 'second.html' with the following content:

```
1 부분 : 12-16 줄
12 12 <li>목록2</li>
13 13 <li>목록3</li>
14 14 </ul>
15 15 <p>추가메세지</p>
16 15 </body>
17 16 </html>
\ No newline at end of file
```

원격저장소를 통한 버전동기화

- 로컬 저장소에 변경사항을 새로운 버전으로 생성(commit)
- 로컬 저장소의 새로운 버전을 원격저장소에 동기화(push)
- 원격저장소의 변경된 버전을 로컬저장소로 동기화(pull)
- 순서를 유지하면 모든 원격저장소에서 만든 버전을 공유 가능





충돌 & 브랜치

충돌(Conflict)

- **localrepository_1에 있는 third.html 파일 내용 수정 → commit → push**
- **수정한 내용이 원격저장소까지 적용 완료**

The screenshot shows the GitHub 'Commits' page for the repository 'khDSClass/firstRepository'. The 'Code' tab is selected. A horizontal timeline bar at the top indicates commits from December 19, 2020, to December 17, 2020. Two specific commits on December 19, 2020, are highlighted with red boxes:

- localrepository_1 - h2태그 추가 (committed 1 minute ago)
- localrepository_1에서 third.html파일을 추가로 생성! (committed 4 hours ago)

Below this, a list of commits from December 17, 2020, shows the resolution of the conflict:

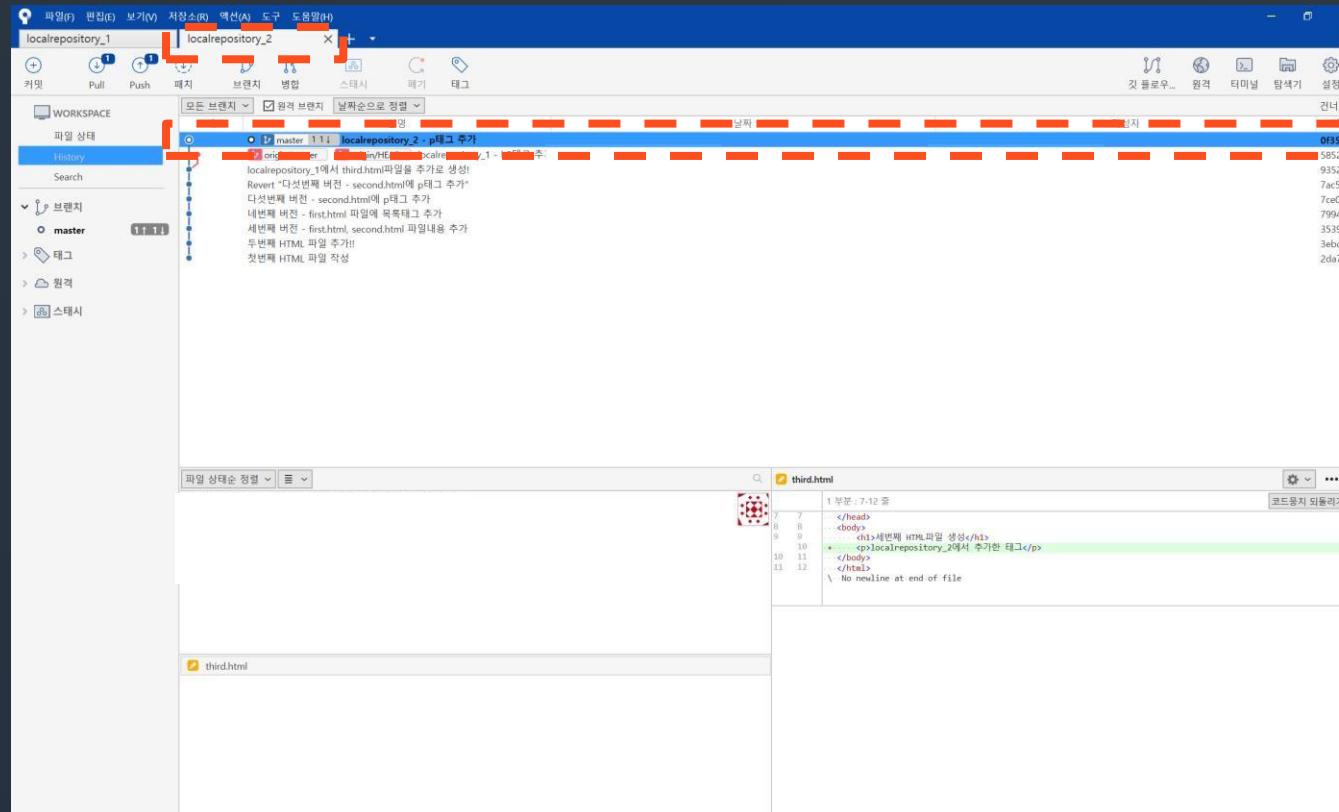
- Revert "다섯번째 버전 - second.html에 p태그 추가" (committed 2 days ago)
- 다섯번째 버전 - second.html에 p태그 추가 (committed 2 days ago)
- 네번째 버전 - first.html 파일에 목록태그 추가 (committed 2 days ago)
- 세번째 버전 - first.html, second.html 파일내용 추가 (committed 2 days ago)
- 두번째 HTML 파일 추가!! (committed 2 days ago)
- 첫번째 HTML 파일 작성 (committed 2 days ago)

At the bottom of the page are 'Newer' and 'Older' navigation buttons.

```
third.html > html
1  <!DOCTYPE html>
2  <html lang="kr">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Document</title>
7  </head>
8  <body>
9      <h1>세 번째 HTML파일 생성</h1>
10     <h2>localrepository_1에서 추가한 내용</h2>
11  </body>
12 </html>
```

충돌(Conflict)

- localrepository_2는 아직 pull을 하지 않은 상태이므로 해당내역이 third.html파일이 이전상태
- localrepository_2의 third.html파일을 수정→commit



```

C: > Users > Lee > Desktop > localrepository_2 > < third.html > html
1   <!DOCTYPE html>
2   <html lang="kr">
3   <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Document</title>
7   </head>
8   <body>
9     <h1>세번째 HTML파일 생성</h1>
10    <p>localrepository_2에서 추가한 태그</p>
11  </body>
12 </html>

```

충돌(Conflict)

- localrepository_1은 원격저장소와 버전이 동일한 상태
- localrepository_2는 원격저장소와 버전이 다른 상태
- localrepository_2는 현재상태에서 push가 불가능하여 pull을 통해 원격저장소의 변경사항까지 포함한 새로운 버전을 만들어서 다시 push

localrepository_1, 원격저장소 third.html

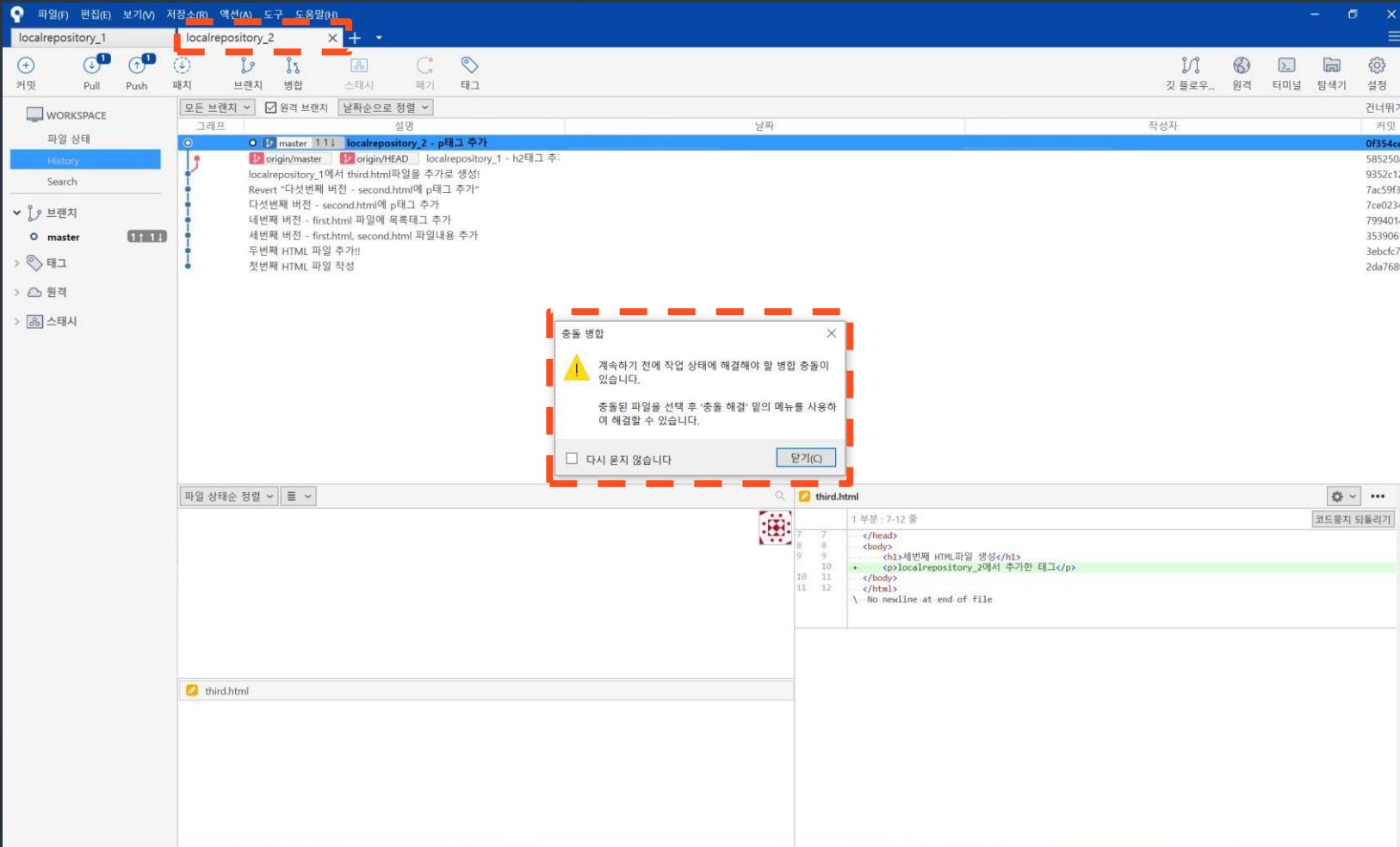
```
④ third.html > ⏺ html
1  <!DOCTYPE html>
2  <html lang="kr">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6  <title>Document</title>
7  </head>
8  <body>
9      <h1>세번째 HTML파일 생성</h1>
10     <h2>localrepository_1에서 추가한 내용</h2>
11  </body>
12 </html>
```

localrepository_2 third.html

```
C: > Users > Lee > Desktop > localrepository_2 > ④ third.html > ⏺ html
1  <!DOCTYPE html>
2  <html lang="kr">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6  <title>Document</title>
7  </head>
8  <body>
9      <h1>세번째 HTML파일 생성</h1>
10     <p>localrepository_2에서 추가한 태그</p>
11  </body>
12 </html>
```

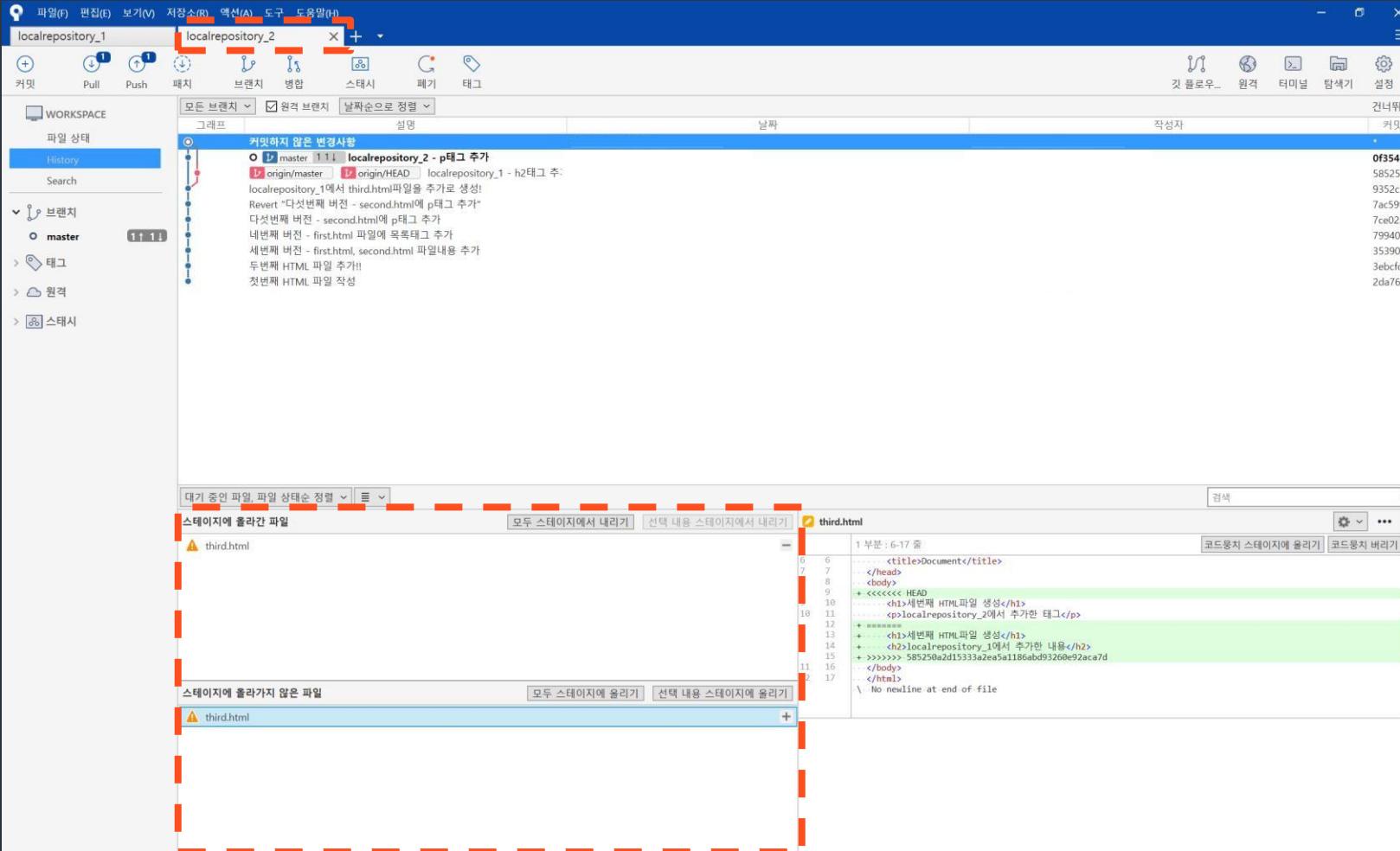
충돌(Conflict)

- localrepository_2에서 pull을 하면 충돌



충돌 (Conflict)

- **localrepository_2의 최신 버전과 원격저장소의 최신버전을 합치는 과정에서 충돌이 발생**
- **충돌이 발생한 파일이 working copy에 보임(third.html)**



충돌(Conflict)

- **localrepository_2의 third.html과 원격저장소의 third.html 파일이 동시에 적용되는 새로운 버전**
- 이런 경우 양쪽에서 모두 다 추가 / 둘 중 하나만 추가 등 여러 가지 가능성이 존재할 수 있으므로 git이 임의로 판단하지 않고 충돌이 발생시킴
- 충돌이 발생한 경우 사용자가 직접 해결하여 적용 → 이번에는 둘 다 적용하는 방식으로 진행

```
C: > Users > Lee > Desktop > localrepository_2 > third.html > html
1  <!DOCTYPE html>
2  <html lang="kr">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Document</title>
7  </head>
8  <body>
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
9  <<<<< HEAD (Current Change)
10 <h1>세 번째 HTML파일 생성</h1>
11 <p>localrepository_2에서 추가한 태그</p>
12 =====
13 <h1>세 번째 HTML파일 생성</h1>
14 <h2>localrepository_1에서 추가한 내용</h2>
15 >>>> 585250a2d15333azea5a1186abd93260e9zaca7d (Incoming Change)
16 </body>
17 </html>
```



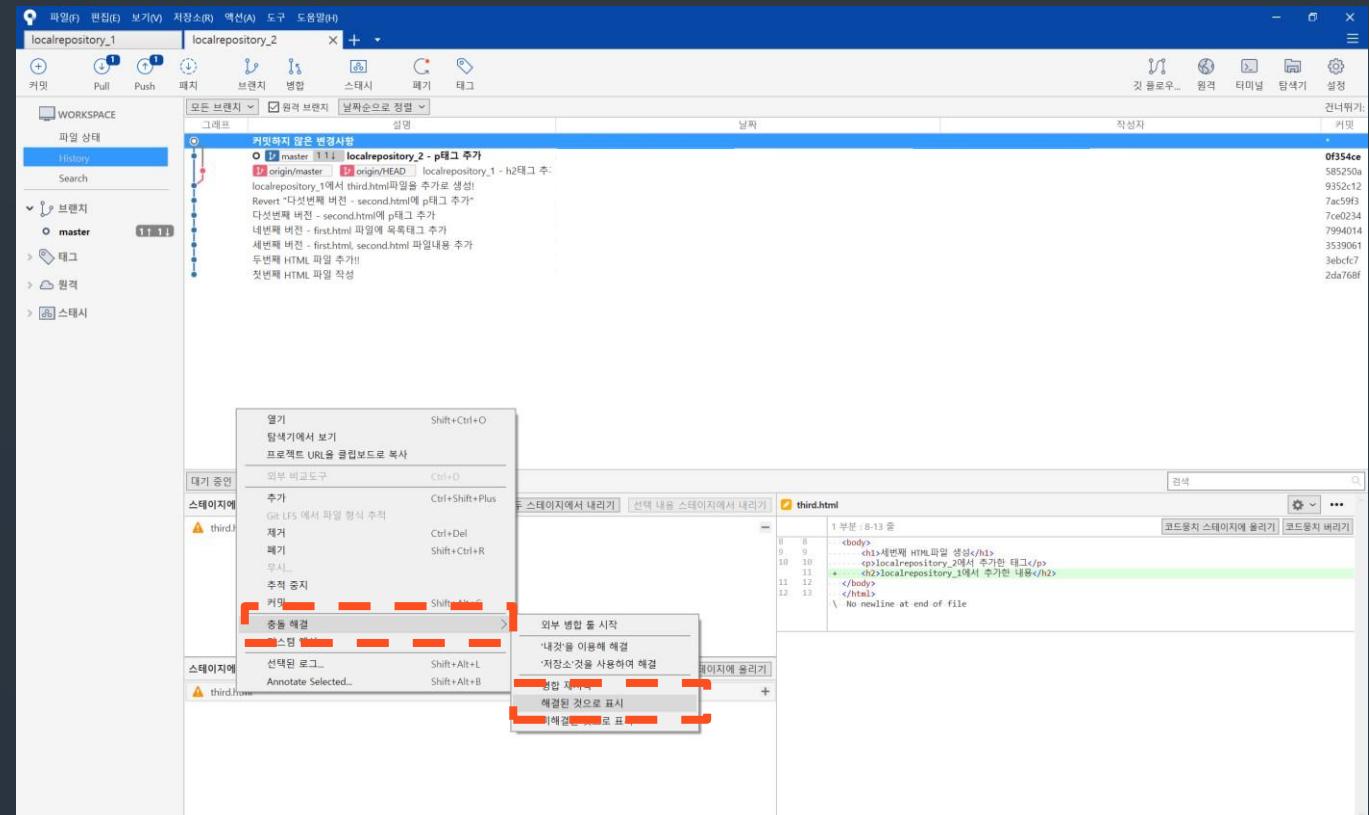
충돌(Conflict)

- 충돌된 파일을 정상상태로 수정
- working copy에서 충돌파일에 마우스 오른쪽 클릭 후 - '충돌 해결' - '해결된 것으로 표시' 선택

```

1  <!DOCTYPE html>
2  <html lang="kr">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Document</title>
7  </head>
8  <body>
9      <h1>세번째 HTML파일 생성</h1>
10     <p>localrepository_2에서 추가한 태그</p>
11     <h2>localrepository_1에서 추가한 내용</h2>
12 </body>
13 </html>

```



충돌 (Conflict)

- localrepository_1에서 원격저장소의 새로운 버전을 pull
- localrepository_2와 동일한 history로 적용

The screenshot shows the SourceTree application interface. The main window displays two repositories: localrepository_1 and localrepository_2. The localrepository_1 tab is active. In the center, a timeline view shows a merge conflict between the master branch of localrepository_1 and origin/master of localrepository_2. The conflict occurred at commit 8b9d3ac, which added a third.html file. The commit message is "third.html - 충돌 해결 버전!". The conflict is indicated by red markers on the timeline and a yellow warning icon in the status bar. Below the timeline, a diff viewer shows the contents of third.html. The code is as follows:

```
<body>
<h1>세번째 HTML파일 생성</h1>
<p>localrepository_2에서 추가한 테그</p>
<h2>localrepository_1에서 추가한 내용</h2>
</body>
</html>
```

A green highlight covers the line containing the text "localrepository_1에서 추가한 내용". A red error message at the bottom right of the diff viewer states "\ No newline at end of file".



충돌(Conflict)

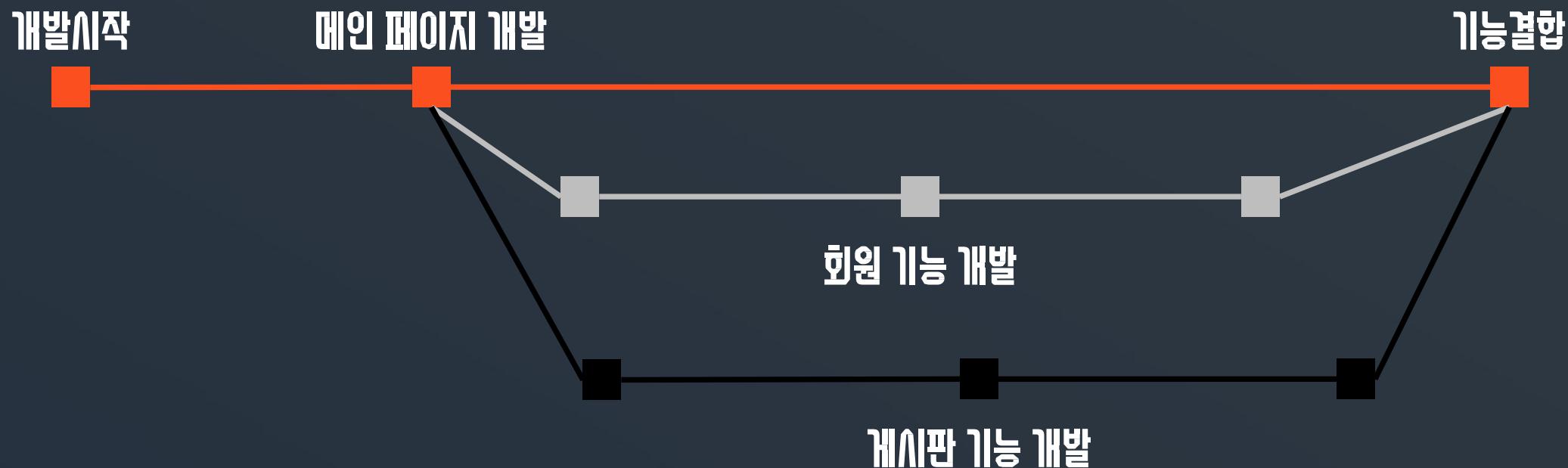
- 원격저장소의 변경사항과 로컬저장소의 변경사항을 하나의 버전으로 합치는 과정에서 여러 변경사항이 동시에 발생했을 때 git이 스스로 판단할 수 없는 경우 충돌이 발생
- 혼자서 사용하는 경우 충돌이 발생하는 경우가 많지 않고 해결 또한 비교적 손쉬운 편이나, 협업 시 충돌이 발생한 경우 다른 사람이 작성한 내용에 대한 수정이 필요하므로 확인하고 해결해야 함
- 충돌이 발생했을 때 해결하고 새로운 버전을 만들어서 원격저장소에 적용 해야 함
→ 충돌이 발생한 채로 commit하고 push를 강제로 하는 경우 전체적으로 문제가 발생할 수 있음
- 적당한 단위로 commit과 pull을 지속적으로 하여 작은 충돌을 해결하는 방식으로 진행



Branch

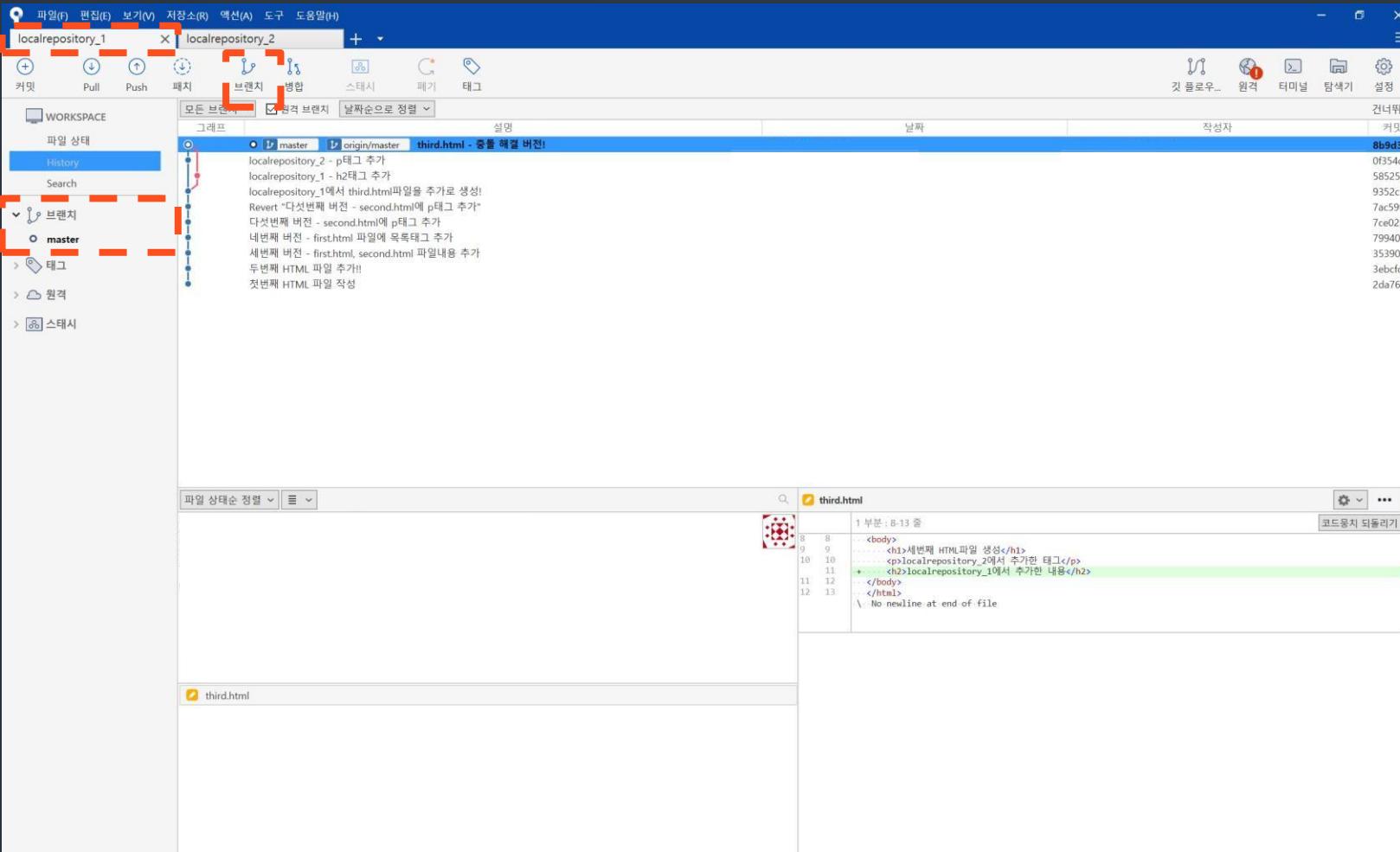
- 나뭇가지라는 뜻으로 코드를 복사하여 다른 작업흐름으로 분리해 내는 것
- 여러 작업흐름을 만들어서 동시에 작업하고 합치는 것을 통해 개발속도 향상 가능
- 개발 중 문제가 발생하더라도 원래 작업흐름에는 영향을 주지 않고 해당작업흐름에 대해서만 문제가 발생하여 협업 시 문제발생을 줄여 줌

홈페이지 개발 흐름



Branch

- 아무 설정 없이 사용하는 경우 기본적으로 master 브랜치가 생성되어 사용
- master 브랜치는 버전을 관리하는 용으로 사용하고 새 브랜치를 만들어서 작업



Branch

- **work-1**이라는 새로운 브랜치를 만들어서 사용
- **master/work-1**이라는 두개의 작업 흐름으로 관리
- 브랜치에 진한 글씨가 현재 브랜치



파일(F) 폴립(E) 보기(V) 저장소(R) 액션(A) 도구 도움말(H)

localrepository_1 localrepository_2

커밋 Pull Push 페지 브랜치 병합 스테시 페지 태그

모든 브랜치 원격 브랜치 날짜순으로 정렬

작성자 날짜

third.html - 충돌 해결 버전

작성자 날짜

8bb53ad 2013-04-13 10:35:40 +0900

585250a 2013-04-13 10:35:21 +0900

9352c12 2013-04-13 10:35:12 +0900

7ac59f3 2013-04-13 10:35:03 +0900

7ce0234 2013-04-13 10:34:54 +0900

7990414 2013-04-13 10:34:45 +0900

3539061 2013-04-13 10:34:36 +0900

3ebfcf7 2013-04-13 10:34:27 +0900

2da768f 2013-04-13 10:34:18 +0900

localrepository_1 - h2태그 추가

localrepository_1에서 third.html파일을 추가로 생성!

Revert "다섯번째 버전 - second.html에 p태그 추가"

다섯번째 버전 - second.html에 p태그 추가

네번쨰 버전 - first.html 파일에 목록태그 추가

세번쨰 버전 - first.html, second.html 파일내용 추가

두번쨰 HTML 파일 추가!!

첫번째 HTML 파일 작성

third.html

```

1 부분 : B-13 줄
 8   <body>
 9     <h1>세번째 HTML파일 생성</h1>
10    <p>localrepository_2에서 추가한 내용</p>
11    <h2>localrepository_1에서 추가한 내용</h2>
12  </body>
13  </html>
\ No newline at end of file

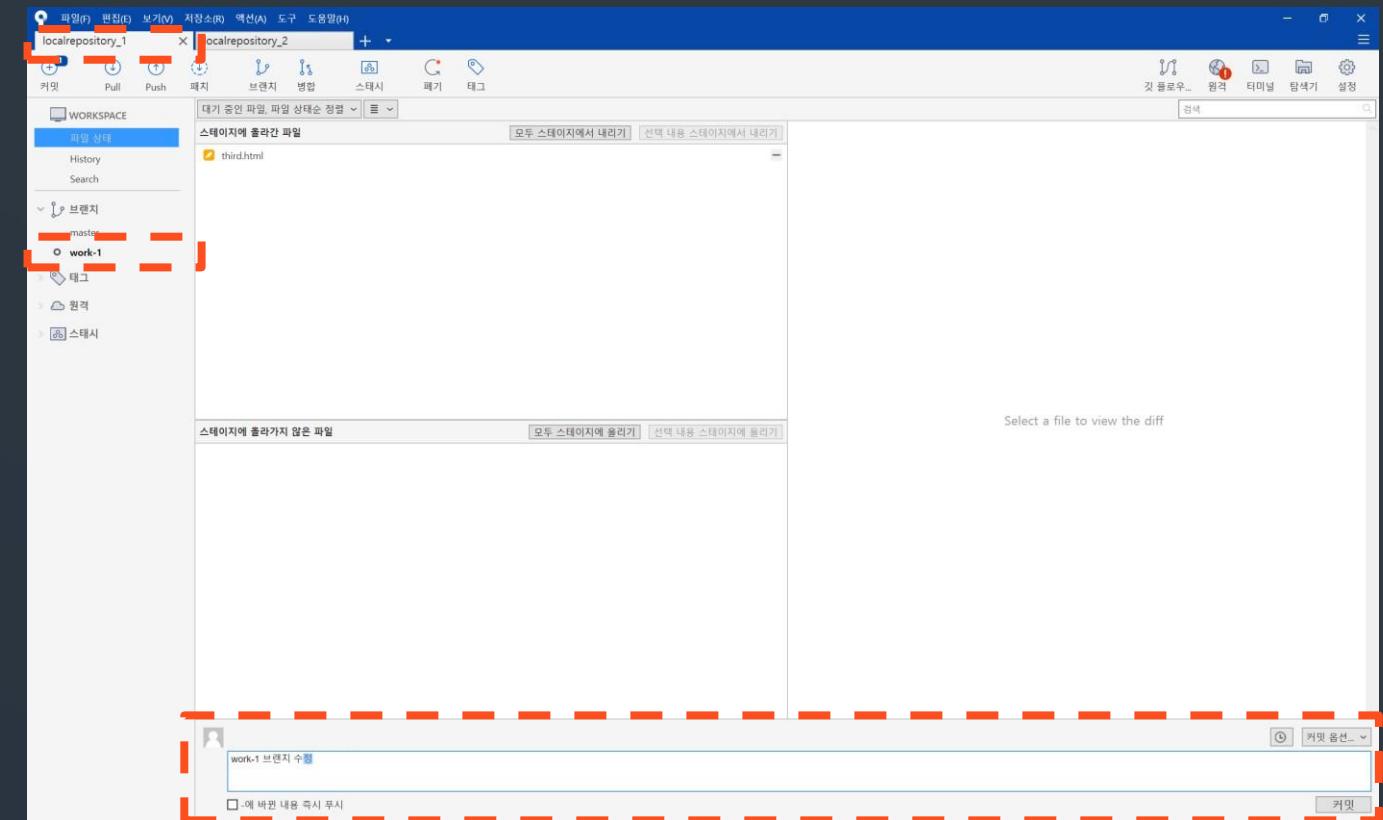
```



Branch

- **localrepository_1**에서 **work-1** 브랜치로 설정한 후 **third.html** 파일 수정
- **master** 브랜치는 버전을 관리하는 용으로 사용하고 새 브랜치를 만들어서 작업

```
<> third.html > html
1   <!DOCTYPE html>
2   <html lang="kr">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Document</title>
7   </head>
8   <body>
9       <h1>세번째 HTML파일 생성</h1>
10      <p>localrepository_2에서 추가한 태그</p>
11      <h2>localrepository_1에서 추가한 내용</h2>
12      <p>work-1브랜치에서 추가한 태그</p>
13   </body>
14 </html>
```



Branch

- history 확인 시 work-1태그는 새로 버전이 만들어진 상태 / master는 이전상태 유지
- master브랜치를 더블클릭하여 브랜치를 옮기고 third.html 파일을 확인

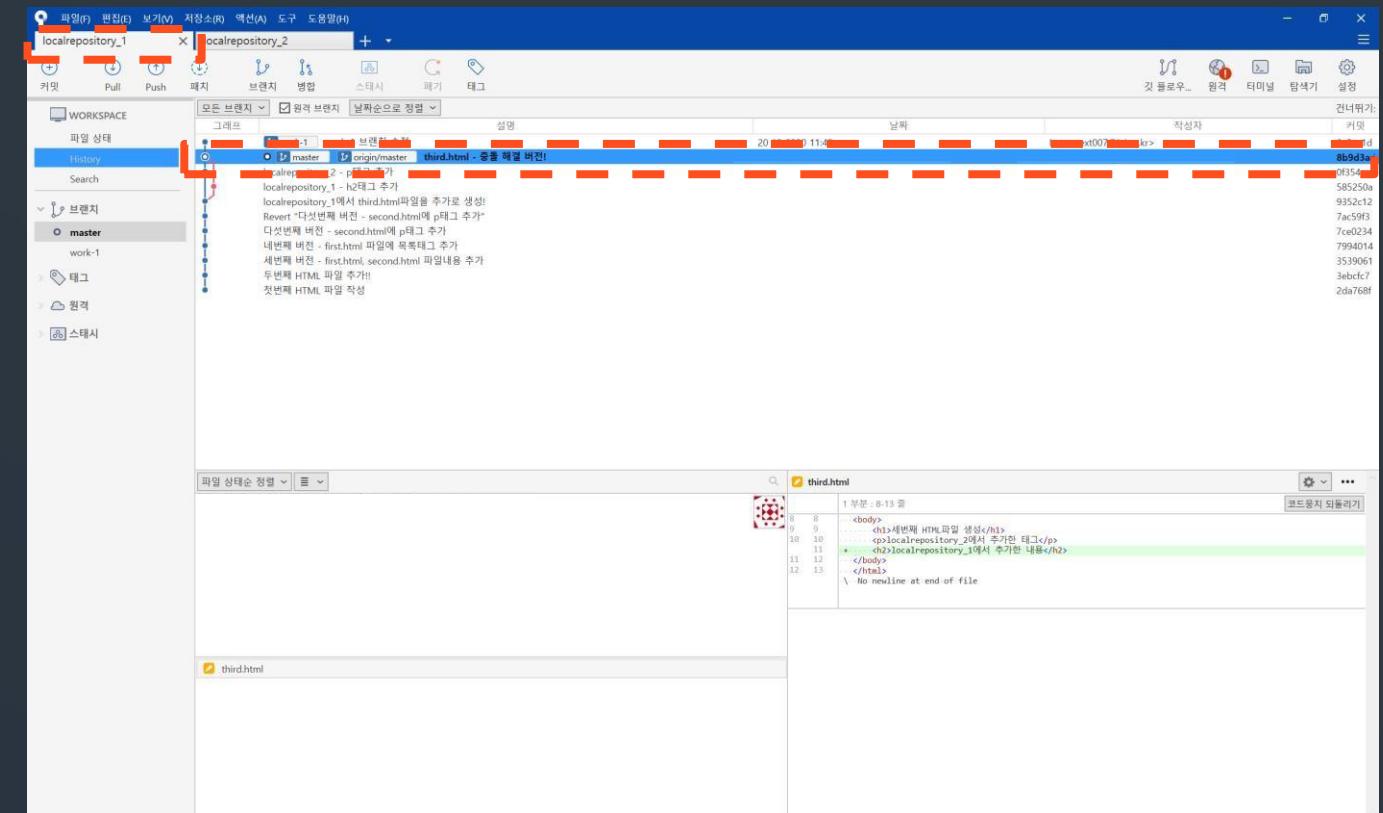
The screenshot shows the SourceTree application interface. At the top, there are two repository tabs: 'localrepository_1' and 'localrepository_2'. The main area displays a timeline of commits. A commit from 'work-1' is highlighted, showing a merge conflict with the 'master' branch. The commit message reads: 'third.html - 충돌 해결 버전!' (third.html - resolve conflict version!). Below the timeline, the 'History' tab is selected in the left sidebar, which also includes 'Search', 'Branches', 'Tags', 'Watches', and 'Status' sections. The 'Branches' section shows 'master' and 'work-1' branches. In the bottom right corner, a code editor window titled 'third.html' is open, showing the following code:

```
1 부분 : 8-14 줄
8 8 <body>
9 9 <h1>세번째 HTML파일 생성</h1>
10 10 <p>localrepository_2에서 추가한 테그</p>
11 11 <h2>localrepository_1에서 추가한 내용</h2>
12 12 <p>work-1브랜치에서 추가한 테그</p>
13 13 </body>
14 </html>
\ No newline at end of file
```

Branch

- master 브랜치는 변경사항이 없으므로 동일한 폴더의 third.html 파일에 추가 내용 없음
- 브랜치를 옮기면 해당 브랜치의 버전에 맞는 상태로 폴더 상태가 변화

```
< third.html > html
1   <!DOCTYPE html>
2   <html lang="kr">
3     <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Document</title>
7     </head>
8     <body>
9       <h1>세번째 HTML파일 생성</h1>
10      <p>localrepository_2에서 추가한 태그</p>
11      <h2>localrepository_1에서 추가한 내용</h2>
12    </body>
13  </html>
```

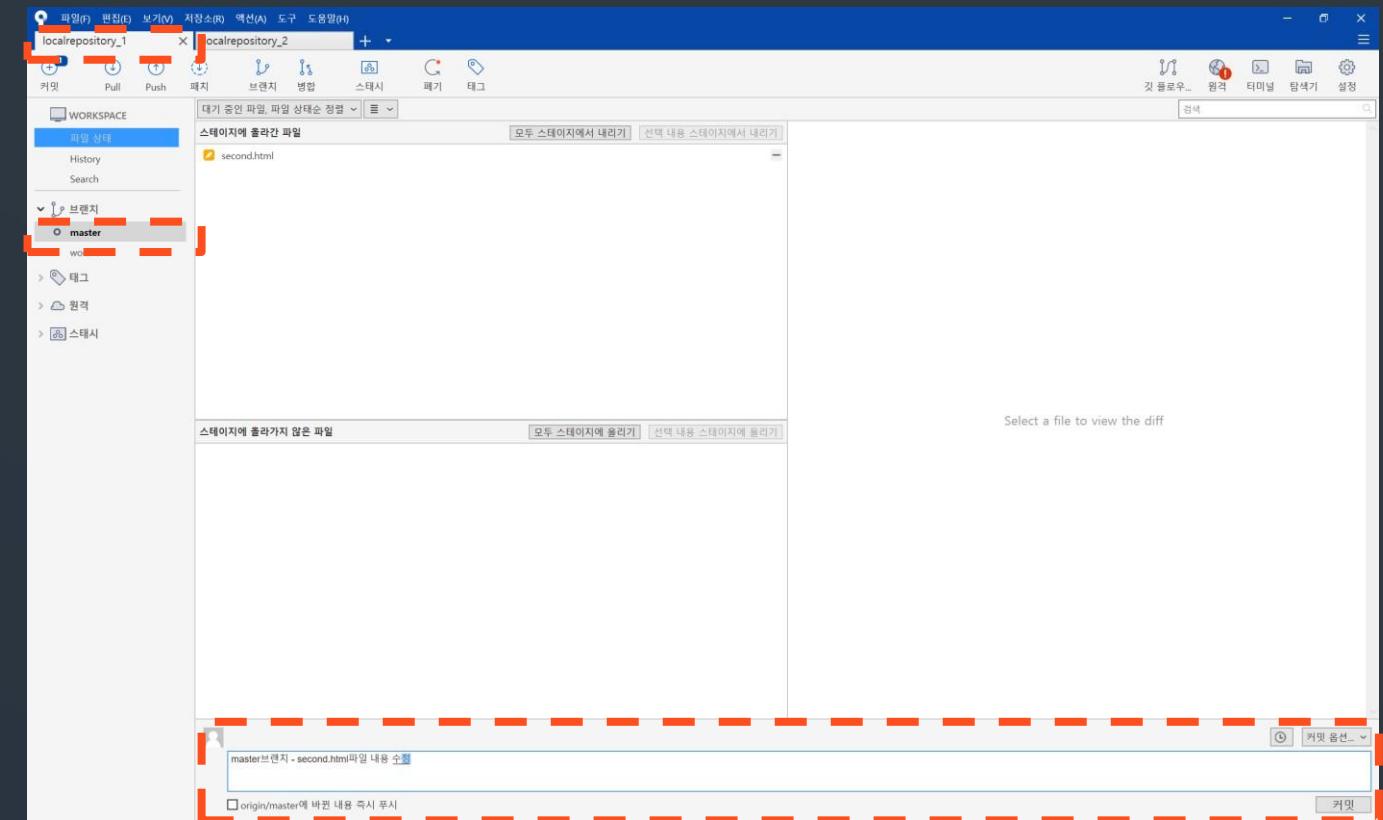




Branch

- master브랜치에서 second.html파일 내용 수정 후 커밋

```
<> second.html > html
1   <!DOCTYPE html>
2   <html lang="kr">
3     <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Document</title>
7     </head>
8     <body>
9       <h1>두번째 HTML파일</h1>
10      <ul>
11        <li>목록1</li>
12        <li>목록2</li>
13        <li>목록3</li>
14      </ul>
15      <h3>마스터브랜치 내용 추가</h3>
16    </body>
17  </html>
```





Branch

- master 브랜치: second.html 파일 변경 / third.html 파일 변화 없음
- work-1 브랜치: second.html 파일 변화 없음 / third.html 파일 변경

The screenshot shows the SourceTree interface with two repositories: localrepository_1 and localrepository_2. The master branch of localrepository_1 has a merge conflict with the work-1 branch of localrepository_2. The conflict is visible in the 'second.html' file, where the 'third.html' content has been added to the 'second.html' file.

LocalRepository_1 (Master Branch History):

- f987926 (Merge commit)
- 668ac1d
- 585250a
- 9352c12
- 7ac59f3
- 7ce0234
- 7994014
- 3539061
- 3ebcfc7
- 2da768f

LocalRepository_2 (Work-1 Branch History):

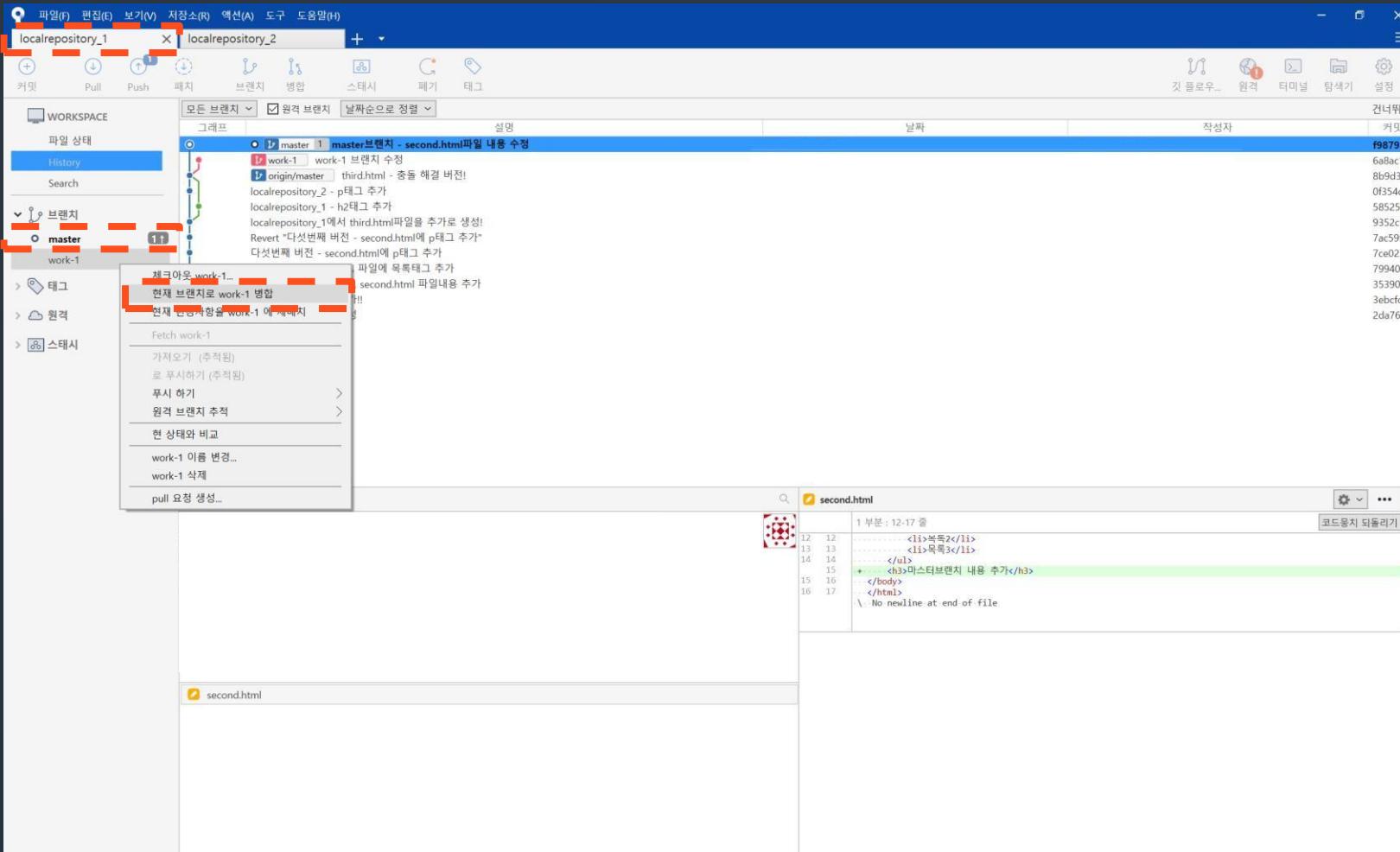
- 8fb9e300
- 8fb9e300
- 0f354ce
- 585250a
- 9352c12
- 7ac59f3
- 7ce0234
- 7994014
- 3539061
- 3ebcfc7
- 2da768f

File Content (second.html):

```
1 부분 : 12-17 줄
12 12 <li>목록2</li>
13 13 <li>목록3</li>
14 14 </ul>
15 15 <h3>마스터브랜치 내용 추가</h3>
16 16 </body>
17 17 </html>
\ No newline at end of file
```

병합(Merge)

- 두 브랜치의 변경사항을 하나로 합쳐 새로운 버전을 만들어내는 것
- master 브랜치인 상태에서 work-1 브랜치에 마우스 오른쪽 클릭 후 현재 브랜치로 병합(master에 적용)



병합(Merge)

- 두 브랜치에서 각각 작업한 걸 병합한 새로운 버전이 master 브랜치에 생성
- master 브랜치는 second.html, third.html 두 파일 모두 변경상태를 합친 상태가 됨

The screenshot shows the SourceTree interface with two repositories: localrepository_1 and localrepository_2. The localrepository_2 tab is active, showing a timeline of commits. A merge commit titled 'Merge branch 'work-1'' is highlighted, showing the history of changes from both branches. Below the timeline, the 'third.html' file is open in the editor, showing the merged content. The code highlights the new content added during the merge.

third.html content:

```
1 부분 : 8-14 줄
8 8 ...<body>
9 9 .....<h1>세번째 HTML파일 생성</h1>
10 10 <p>localrepository_2에서 추가한 테그</p>
11 11 .....<h2>localrepository_1에서 추가한 내용</h2>
12 12 <p>work-1브랜치에서 추가한 테그</p>
13 13 .....</body>
14 14 </html>
\ No newline at end of file
```

Branch

- master브랜치의 second.html, third.html 파일
- 두 브랜치에서 변경된 내용 모두 적용

```
↳ second.html > ⚡ html
1   <!DOCTYPE html>
2   <html lang="kr">
3     <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Document</title>
7     </head>
8     <body>
9       <h1>두번째 HTML파일</h1>
10      <ul>
11        <li>목록1</li>
12        <li>목록2</li>
13        <li>목록3</li>
14      </ul>
15      <h3>마스터브랜치 내용 추가</h3>
16    </body>
17  </html>
```

```
↳ third.html > ⚡ html
1   <!DOCTYPE html>
2   <html lang="kr">
3     <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Document</title>
7     </head>
8     <body>
9       <h1>세번째 HTML파일 생성</h1>
10      <p>localrepository_2에서 추가한 태그</p>
11      <h2>localrepository_1에서 추가한 내용</h2>
12      <p>work-1브랜치에서 추가한 태그</p>
13    </body>
14  </html>
```



Branch

- **work-1브랜치의 second.html, third.html 파일**
- **work-1에서는 병합하지 않았으므로 master브랜치에서 변경 내용이 적용되지 않음**
- **work-1에서도 모두 적용하고 싶은 경우 병합 진행**

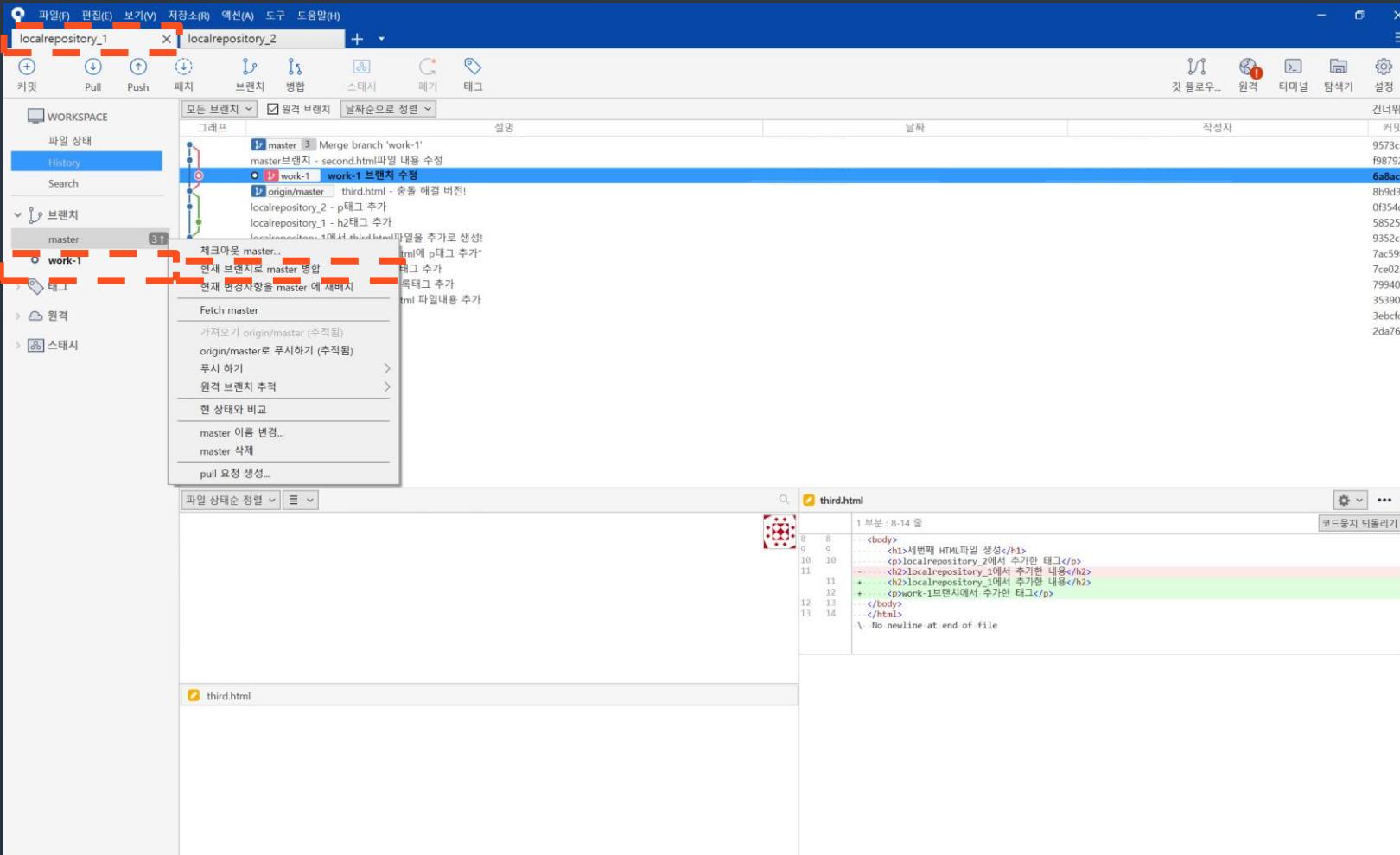
```
④ second.html > ⚡ html
1   <!DOCTYPE html>
2   <html lang="kr">
3   <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Document</title>
7   </head>
8   <body>
9     <h1>두번째 HTML파일</h1>
10    <ul>
11      <li>목록1</li>
12      <li>목록2</li>
13      <li>목록3</li>
14    </ul>
15  </body>
16 </html>
```

```
④ third.html > ⚡ html
1   <!DOCTYPE html>
2   <html lang="kr">
3   <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Document</title>
7   </head>
8   <body>
9     <h1>세번째 HTML파일 생성</h1>
10    <p>localrepository_2에서 추가한 태그</p>
11    <h2>localrepository_1에서 추가한 내용</h2>
12    <p>work-1브랜치에서 추가한 태그</p>
13  </body>
14 </html>
```



병합(Merge)

- 두 브랜치의 변경사항을 하나로 합쳐 새로운 버전을 만들어내는 것
- work-1 브랜치인 상태에서 master 브랜치에 마우스 오른쪽 클릭 후 현재 브랜치로 병합(work-1에 적용)



Branch

- 두 브랜치가 같은 버전으로 적용됨

The screenshot shows the SourceTree application interface. At the top, there are two repository tabs: 'localrepository_1' and 'localrepository_2'. Below the tabs is a toolbar with icons for Cut, Pull, Push, Diff, Branch, Merge, Stash, Log, and Tags.

The main area displays a timeline graph showing the history of both branches. The 'History' tab is selected in the left sidebar. The 'Branches' tab is also visible, showing the 'master' branch and the 'work-1' branch.

A detailed commit history is shown on the right side of the timeline:

- origin/master - third.html - 충돌 해결 버전!
- master - htm 내용
- work-1 브랜치 수정
- origin/master - third.html - 충돌 해결 버전!
- localrepository_2 - p태그 추가
- localrepository_1 - h2태그 추가
- localrepository_1에서 third.html파일을 추가로 생성!
- Revert "다섯번째 버전 - second.html에 p태그 추가"
- 다섯번째 버전 - second.html에 p태그 추가
- 네번째 버전 - first.html 파일에 목록태그 추가
- 세번째 버전 - first.html, second.html 파일내용 추가
- 두번째 HTML 파일 추가!!
- 첫번째 HTML 파일 작성

Below the timeline, a code editor window titled 'second.html' is open. It shows the following code:

```
1 부분 : 12-17 줄
<ul>
  <li>목록2</li>
  <li>목록3</li>
</ul>
<p>추가메세지</p>
```

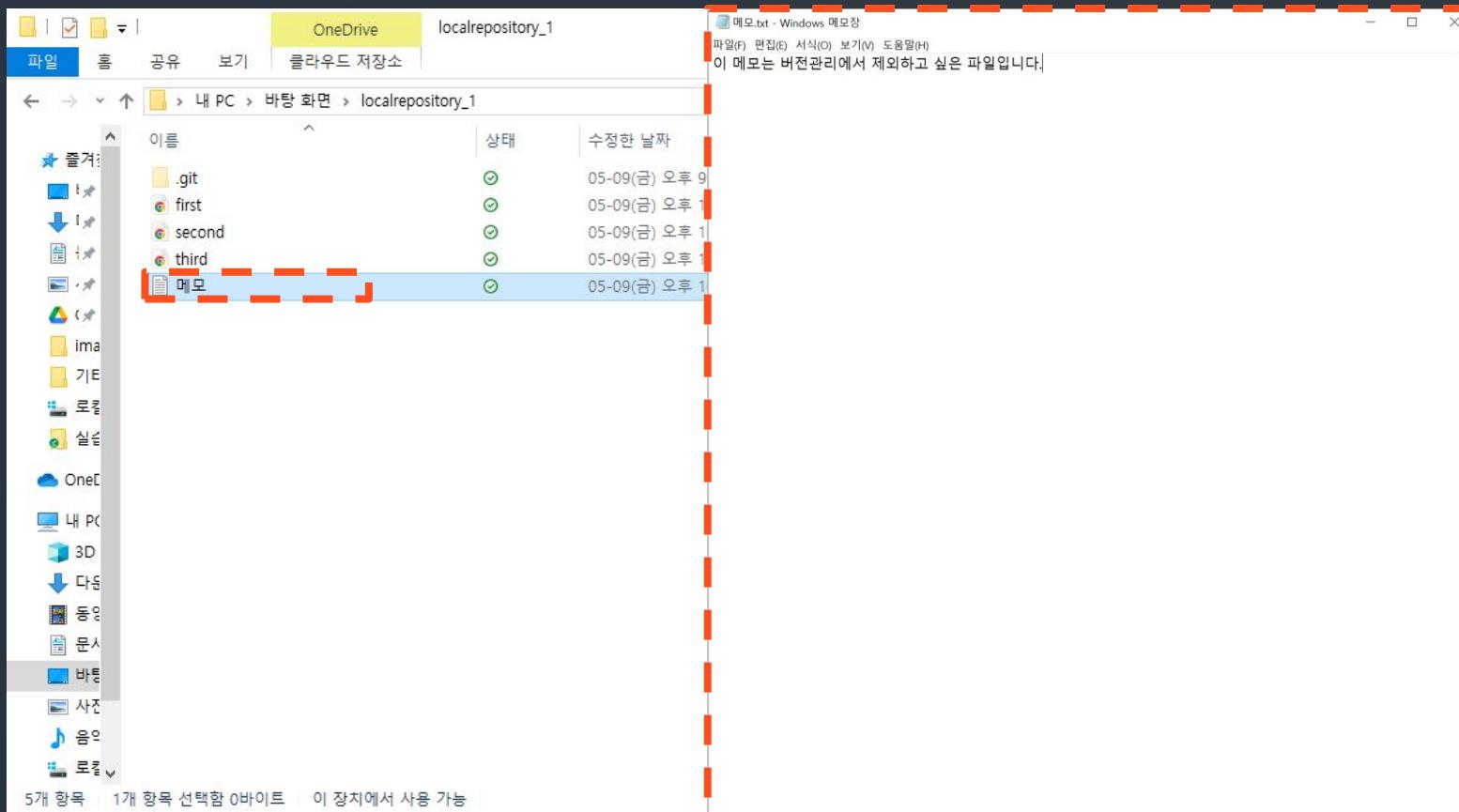
The line '추가메세지' is highlighted in green, indicating it is a conflict or a new addition from the 'work-1' branch.



git ignore

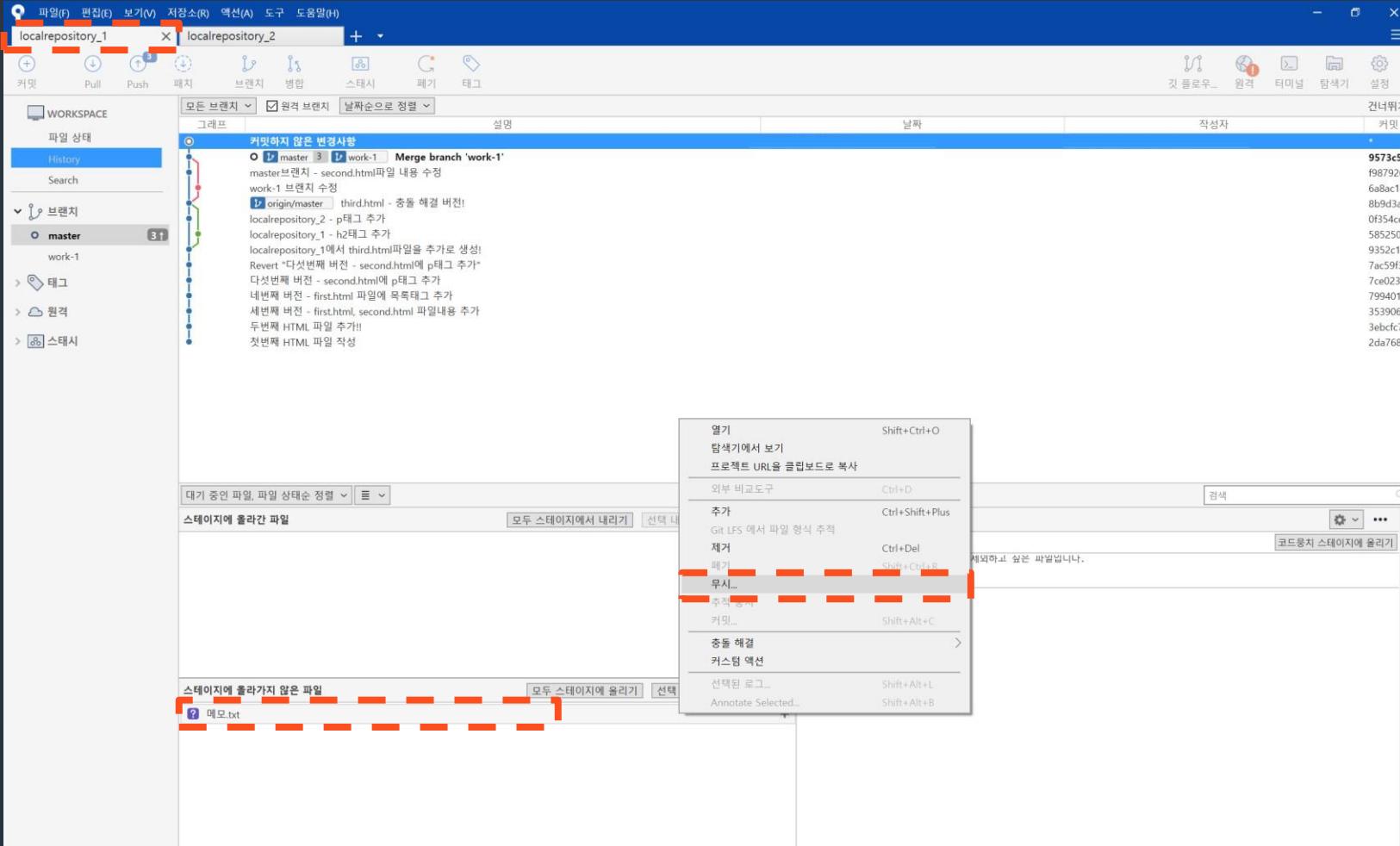
제외하기(git ignore)

- 현재 로컬저장소의 모든 파일에 대해서 변화를 감지하여 적용하고 원격저장소에도 적용
- 버전에 영향을 주지 않는 파일의 경우 버전관리에서 제외 가능
- localrepository_1에서 메모.txt 파일을 만들어서 안에 내용 작성(master 브랜치)



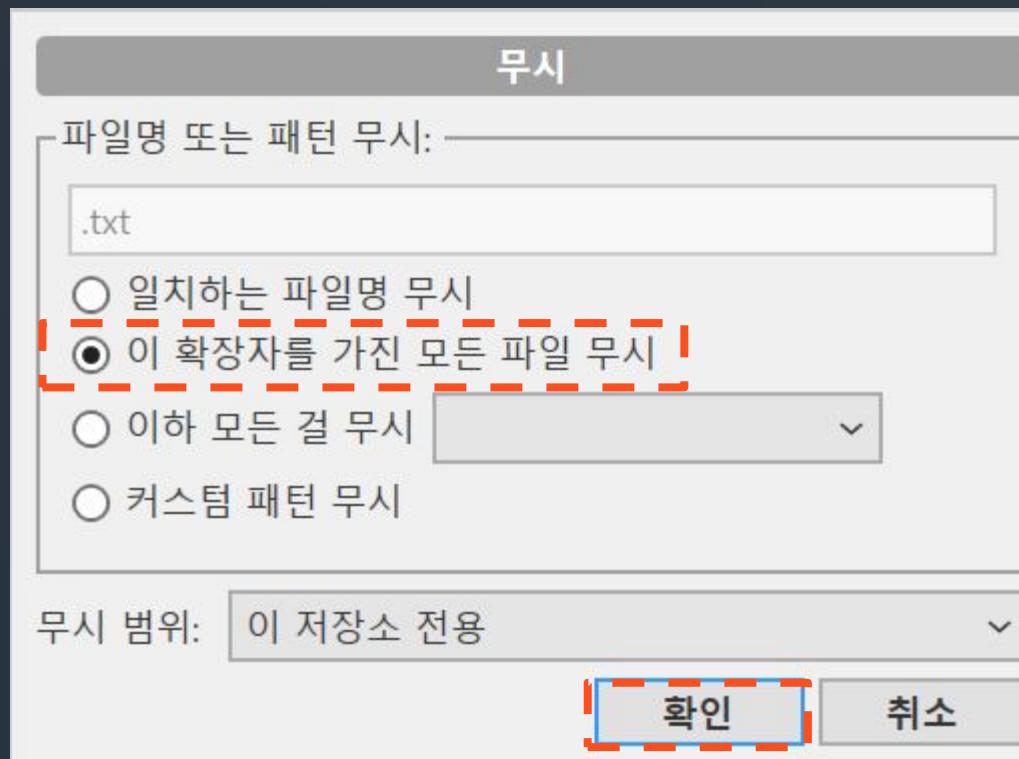
제외하기(git ignore)

- 로컬저장소에 새로 생성된 파일인 메모.txt가 감지→메모.txt는 버전관리에서 제외
- working copy에 메모.txt파일에 마우스 오른쪽을 클릭한 후 무시 선택



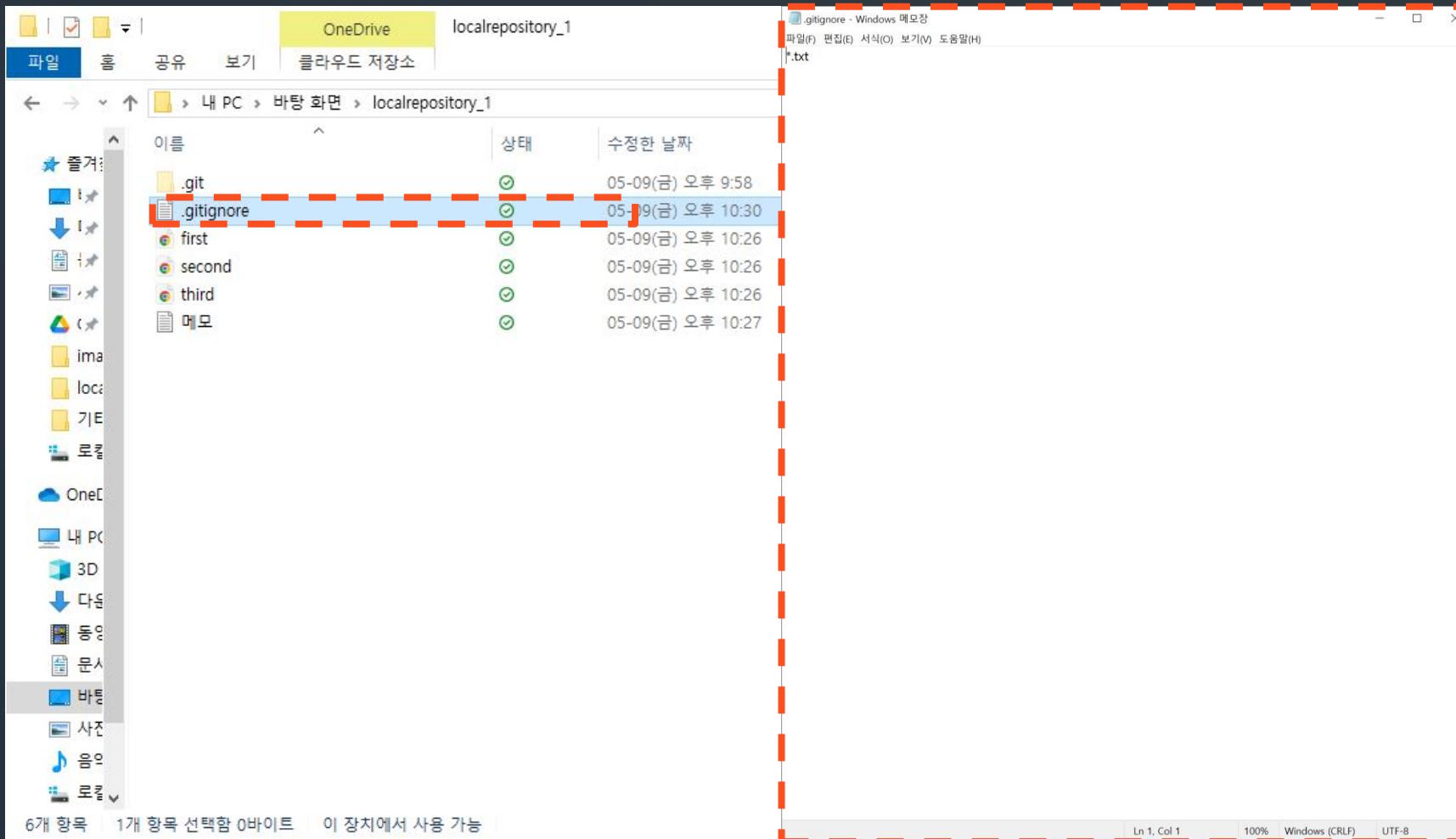
제외하기(git ignore)

- 버전관리에서 제외하는 옵션 선택
- 일치하는 파일명 무시: 현재 선택한 파일만 버전관리에서 제외
- 이 확장자를 가진 모든 파일 무시: 현재 선택한 파일과 같은 확장자를 가진 모든 파일들을 무시(txt 파일 무시)
- 이하 모든 걸 무시: 폴더를 선택하여 선택한 폴더 내부에 있는 파일들은 모두 무시
- 커스텀 패턴 무시: 패턴을 만들어서 패턴에 일치하는 파일들을 무시
- 확장자 옵션 선택 후 확인



제외하기(gitignore)

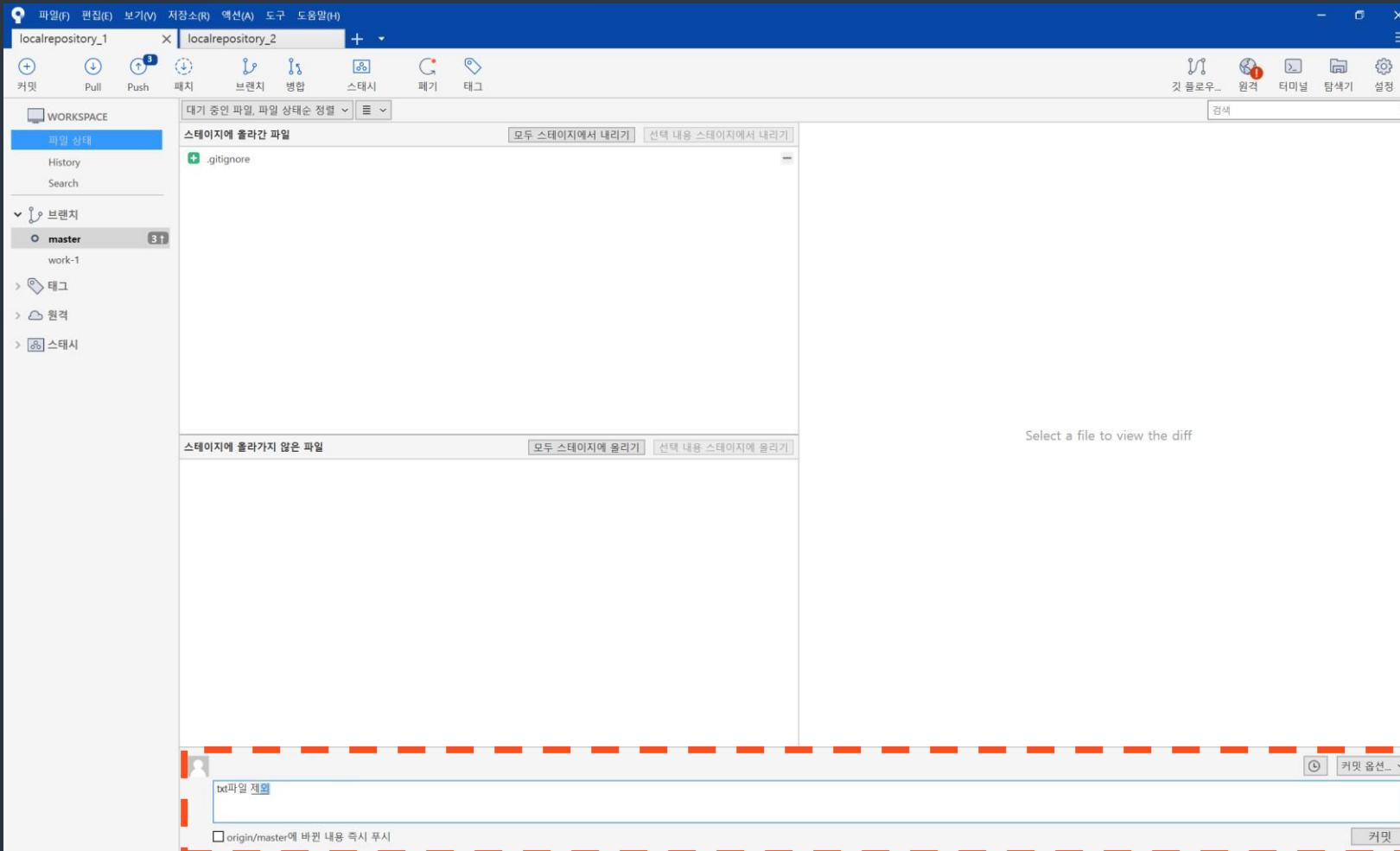
- 로컬저장소에 .gitignore 라는 파일이 자동으로 생성
- .gitignore 파일에 버전관리에서 제외할 파일에 대한 패턴을 입력하면 해당 파일들은 버전관리에서 제외





제외하기(git ignore)

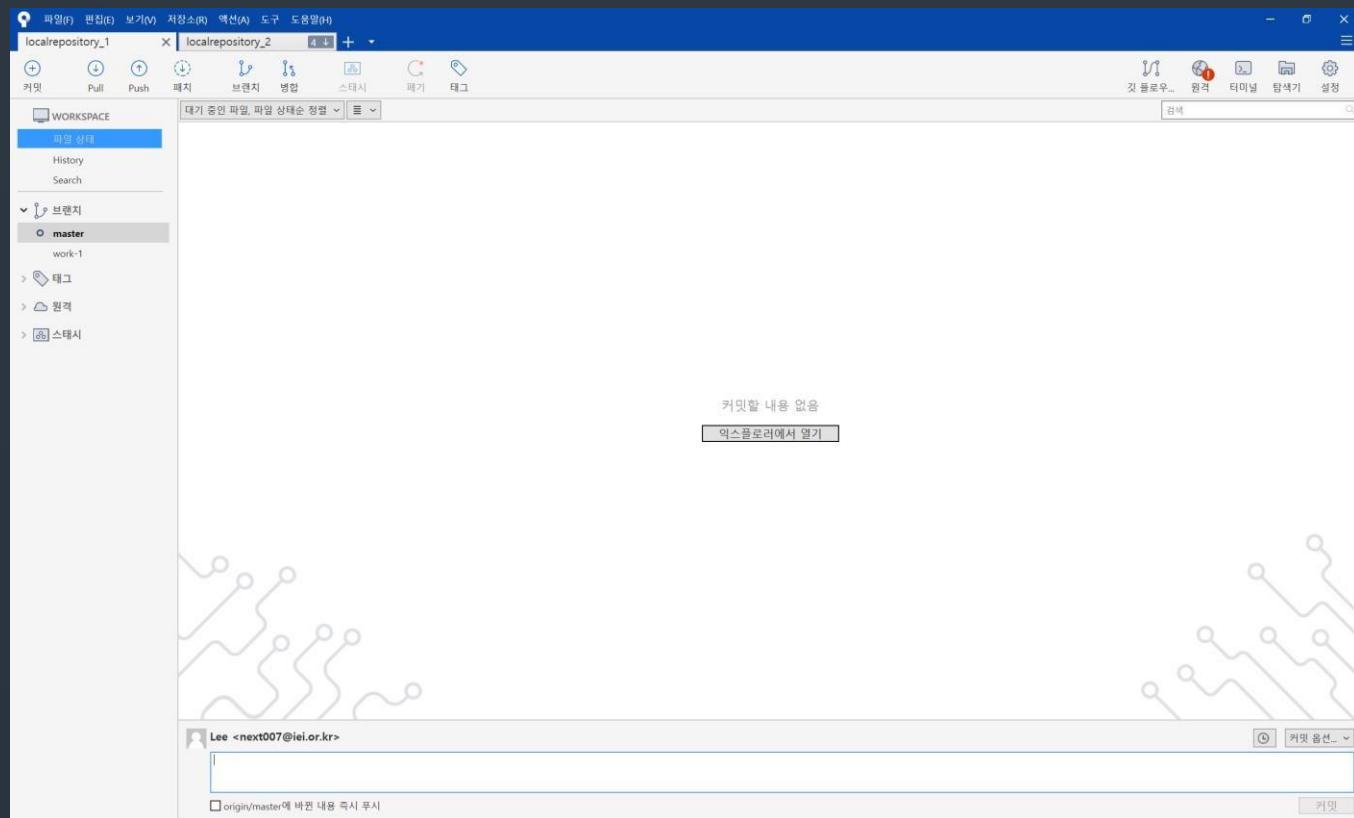
- .gitignore 파일도 저장소에 새로 생성된 것이므로 커밋 → push
- push로 원격저장소에 .gitignore 파일을 배포하면 원격저장소를 이용하는 로컬저장소 모두에 적용 가능



제외하기(git ignore)

- 새로운 txt파일을 만들어도 더 이상 추적X

📁	.git	05-09(금) 오후 9:58	파일 폴더
📄	.gitignore	05-09(금) 오후 10:30	텍스트 문서
🌐	first	05-09(금) 오후 10:26	Chrome HTML D...
🌐	second	05-09(금) 오후 10:26	Chrome HTML D...
🌐	third	05-09(금) 오후 10:26	Chrome HTML D...
📝	메모	05-09(금) 오후 10:27	텍스트 문서
📝	테스트	05-09(금) 오후 10:32	텍스트 문서



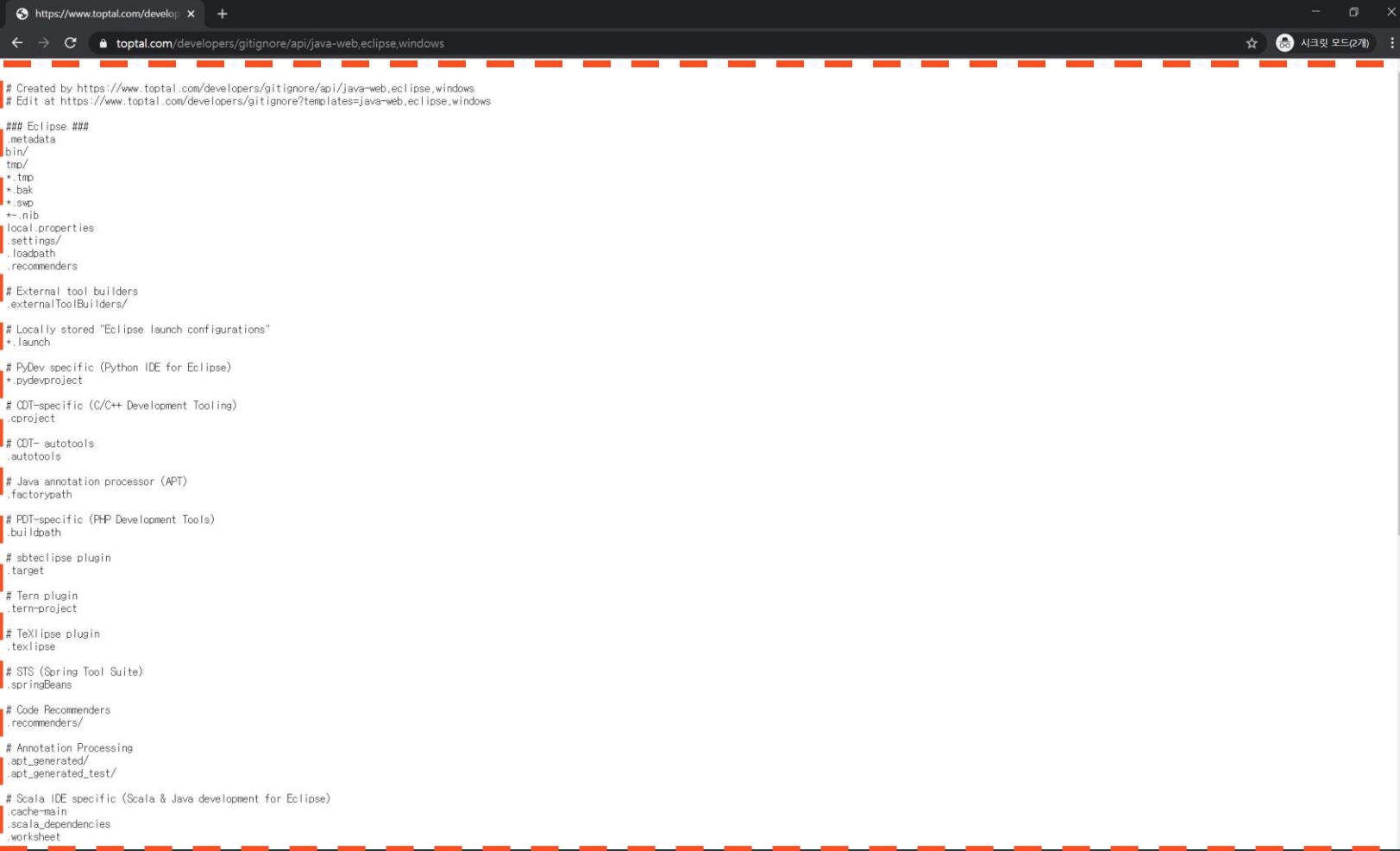
제외하기(gitignore)

- 제외하기 위한 내용들에 대한 것은 모두 .gitignore 파일에 계속 작성
- .gitignore 파일을 작성할 때는 glob이란 언어를 통해서 작성
- 일반적으로 개발환경이나 사용하는 에디터별로 ignore 무시해야 할 파일이 다름
- <https://www.toptal.com/developers/gitignore> 활용
- 개발환경, 운영체제, 사용하는 IDE 등을 입력하고 생성하면 ignore 파일 자동으로 작성



제외하기(gitignore)

- 생성된 내용을 .gitignore 파일에 복사해서 적용(커밋 → push)



The screenshot shows a browser window with the URL <https://www.toptal.com/developers/gitignore/api/java-web,eclipse,windows>. The page displays a long list of files and directories to be ignored, starting with comments about Eclipse and PyDev, followed by CDT, PDT, and various Eclipse plugin configurations like sbteclipse, tern-plugin, and TeXclipse.

```
# Created by https://www.toptal.com/developers/gitignore/api/java-web,eclipse,windows
# Edit at https://www.toptal.com/developers/gitignore?templates=java-web,eclipse,windows

### Eclipse ####
.metadata
bin/
tmp/
*.tmp
*.bak
*.swp
*-*.nib
local.properties
.settings/
.loadpath
.recommenders
.externalToolBuilders/
# Locally stored "Eclipse launch configurations"
.launch
# PyDev specific (Python IDE for Eclipse)
.pydevproject
# CDT-specific (C/C++ Development Tooling)
.cproject
# PDT-specific (PHP Development Tools)
.buildpath
# sbteclipse plugin
.target
# Tern plugin
.tern-project
# TeXclipse plugin
.texclipse
# STS (Spring Tool Suite)
.springBeans
# Code Recommenders
.recommenders/
# Annotation Processing
.apt_generated/
.apt_generated_test/
# Scala IDE specific (Scala & Java development for Eclipse)
.cache-main
.scala_dependencies
.worksheet
```

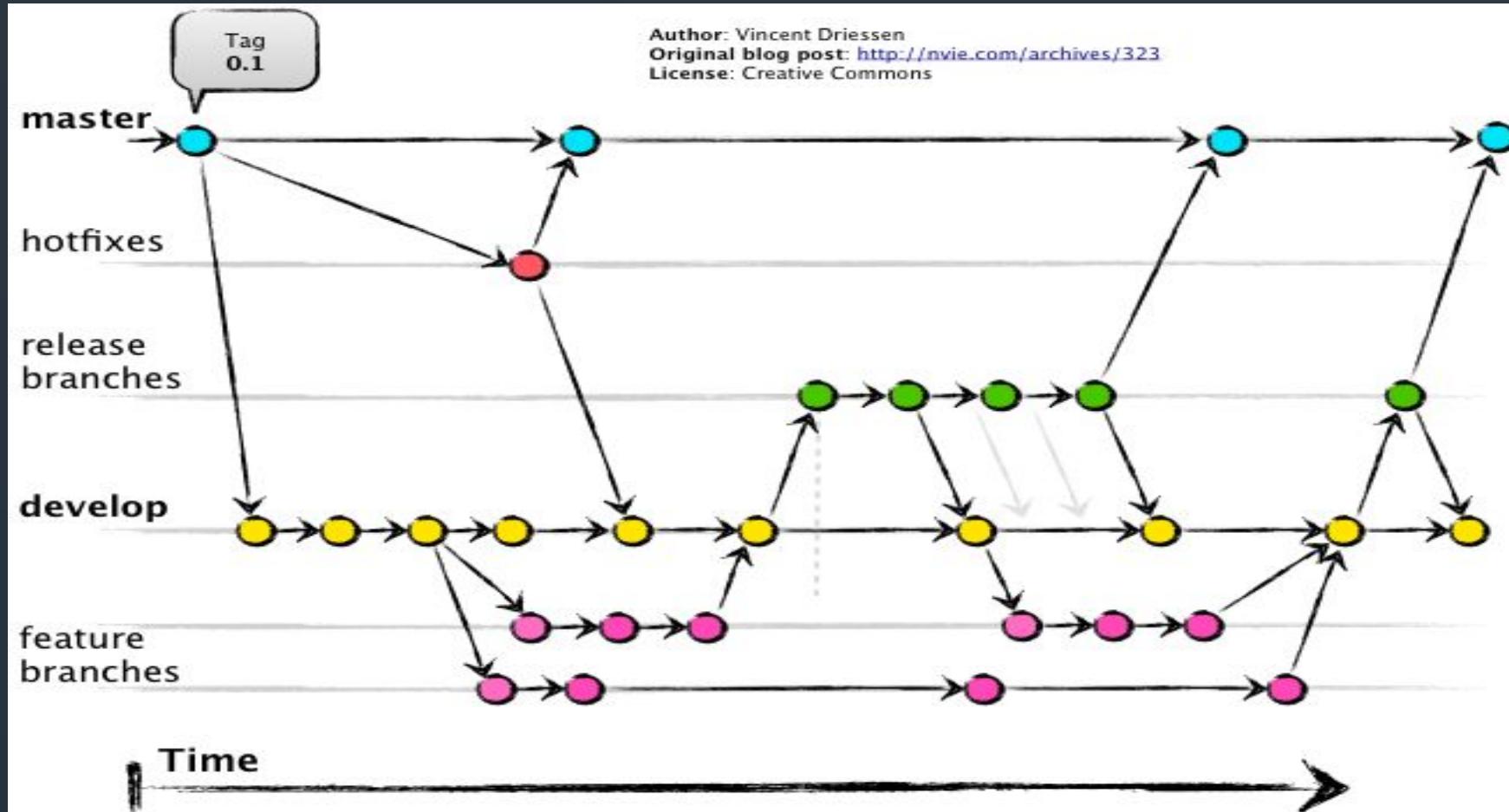


Web Project

협업

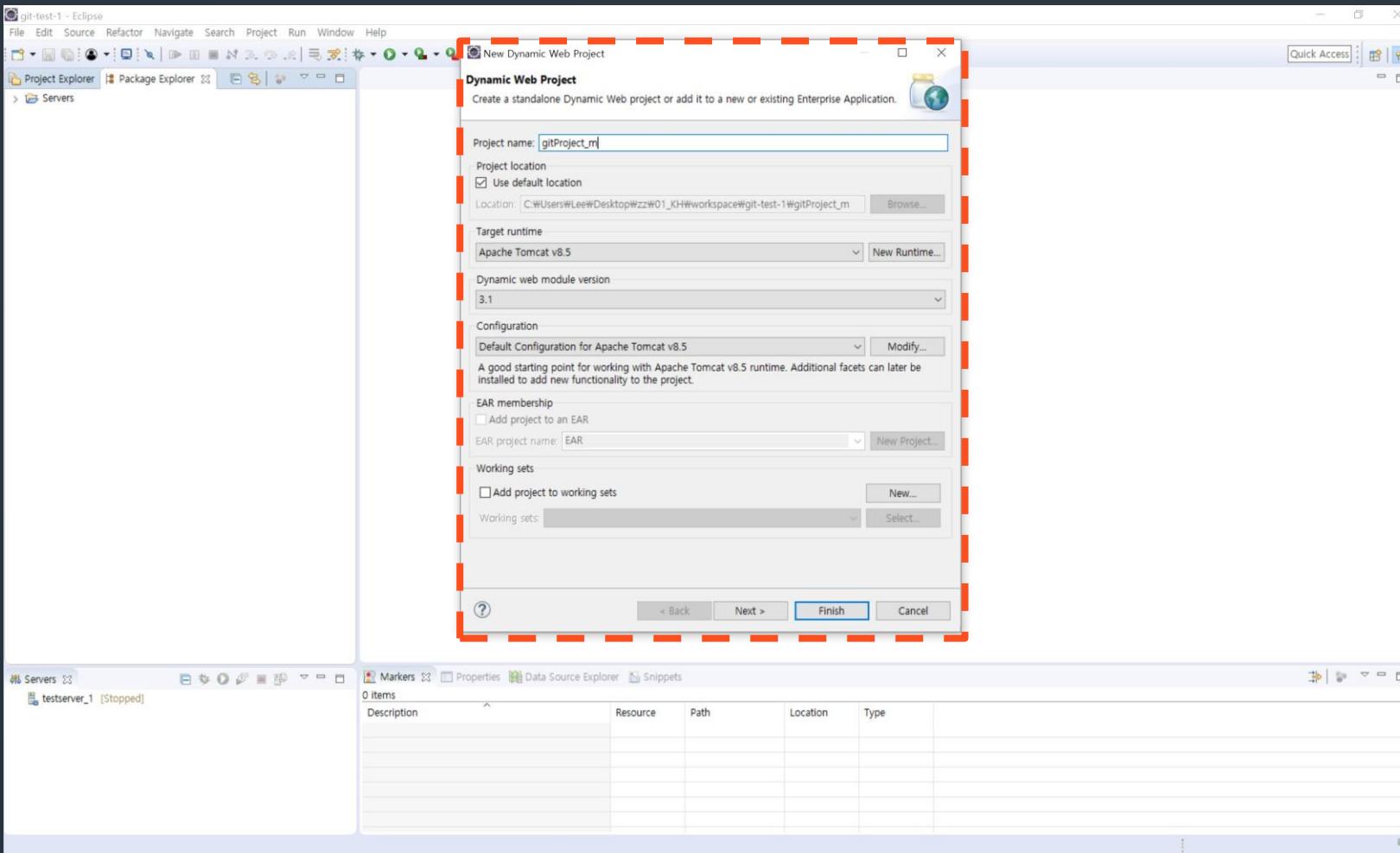
Git workflow

- 협업 시 사용되는 git 작업 흐름도
- PR(Pull Request) 방식으로 진행



최초 프로젝트 생성 (프로젝트 관리자)

- 최초 프로젝트를 1명이 생성하고 다른 팀원들은 pull을 받아서 진행
- 워크스페이스에 새 프로젝트 생성(이번 테스트는 Dynamic Web Project로 진행 / 타 프로젝트 무관)



최초 프로젝트 생성 (프로젝트 관리자)

- 테스트를 위해 프로젝트에 index.jsp 파일 생성 후 작성

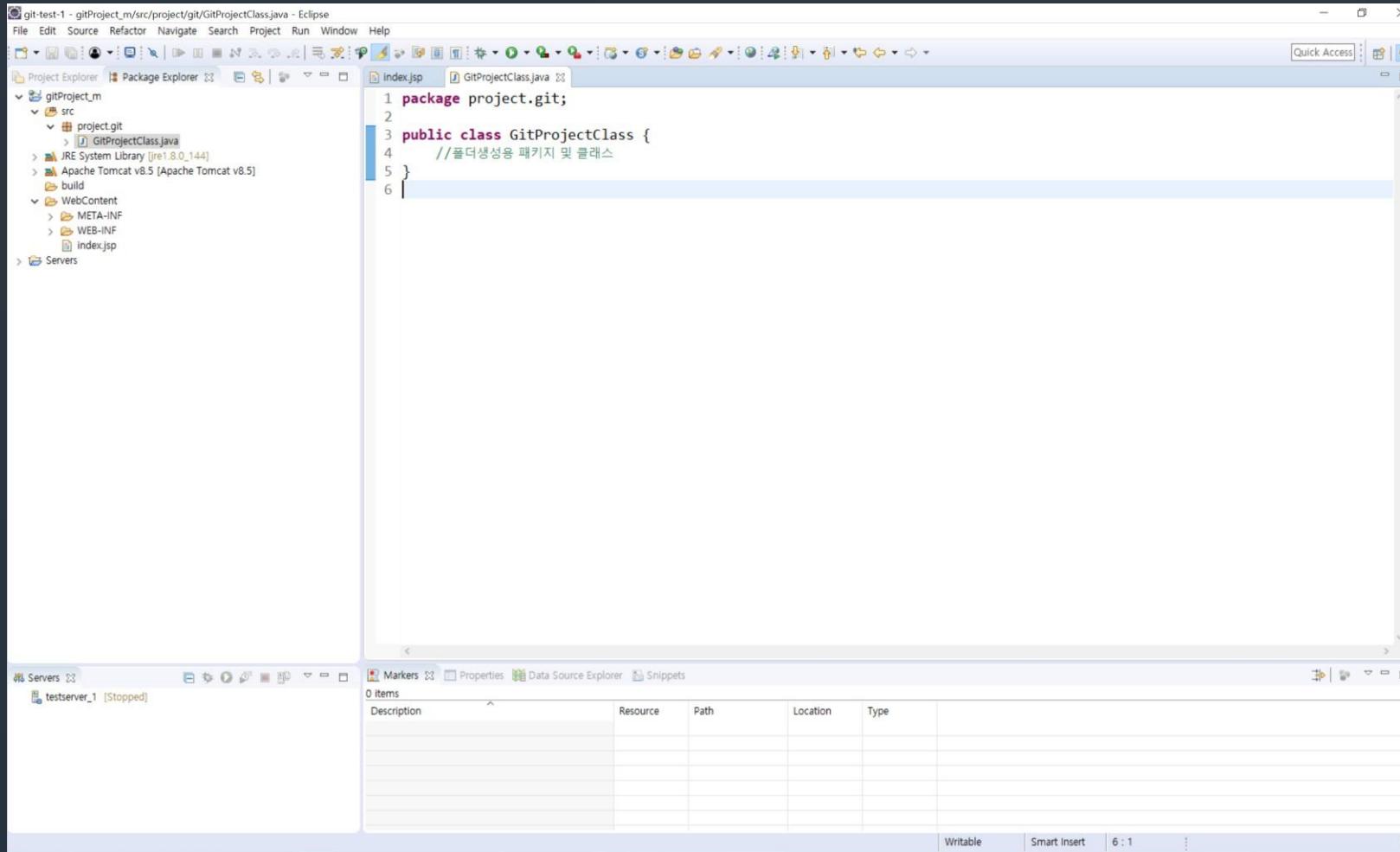
The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure: `gitProject_m` (expanded) containing `src`, `JRE System Library [jre1.8.0_144]`, `Apache Tomcat v8.5 [Apache Tomcat v8.5]`, `build`, and `WebContent` (expanded) containing `META-INF` and `WEB-INF` (expanded) containing `lib`, `Index.jsp`, and `web.xml`.
- Editor:** Displays the `index.jsp` file content:

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2     pageEncoding="UTF-8"%>
3 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
4<html>
5<head>
6 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
7 <title>Insert title here</title>
8 </head>
9<body>
10    <h1>프로젝트 메인페이지</h1>
11 </body>
12 </html>
```
- Servers:** Shows a stopped server named `testserver_1`.
- Bottom Bar:** Includes tabs for `Markers`, `Properties`, `Data Source Explorer`, and `Snippets`. The `Markers` tab shows 0 items.

최초 프로젝트 생성 (프로젝트 관리자)

- 내부에 파일이 없는 경우 폴더도 생기지 않기 때문에 src 폴더 생성을 위한 테스트 패키지 및 클래스 생성



버전관리 예외 설정(프로젝트 관리자)

- 프로젝트 관련 ignore 설정
- Windows, Eclipse, Java-Web(개발환경이 다른 경우 개발환경에 맞도록 ignore 작성)



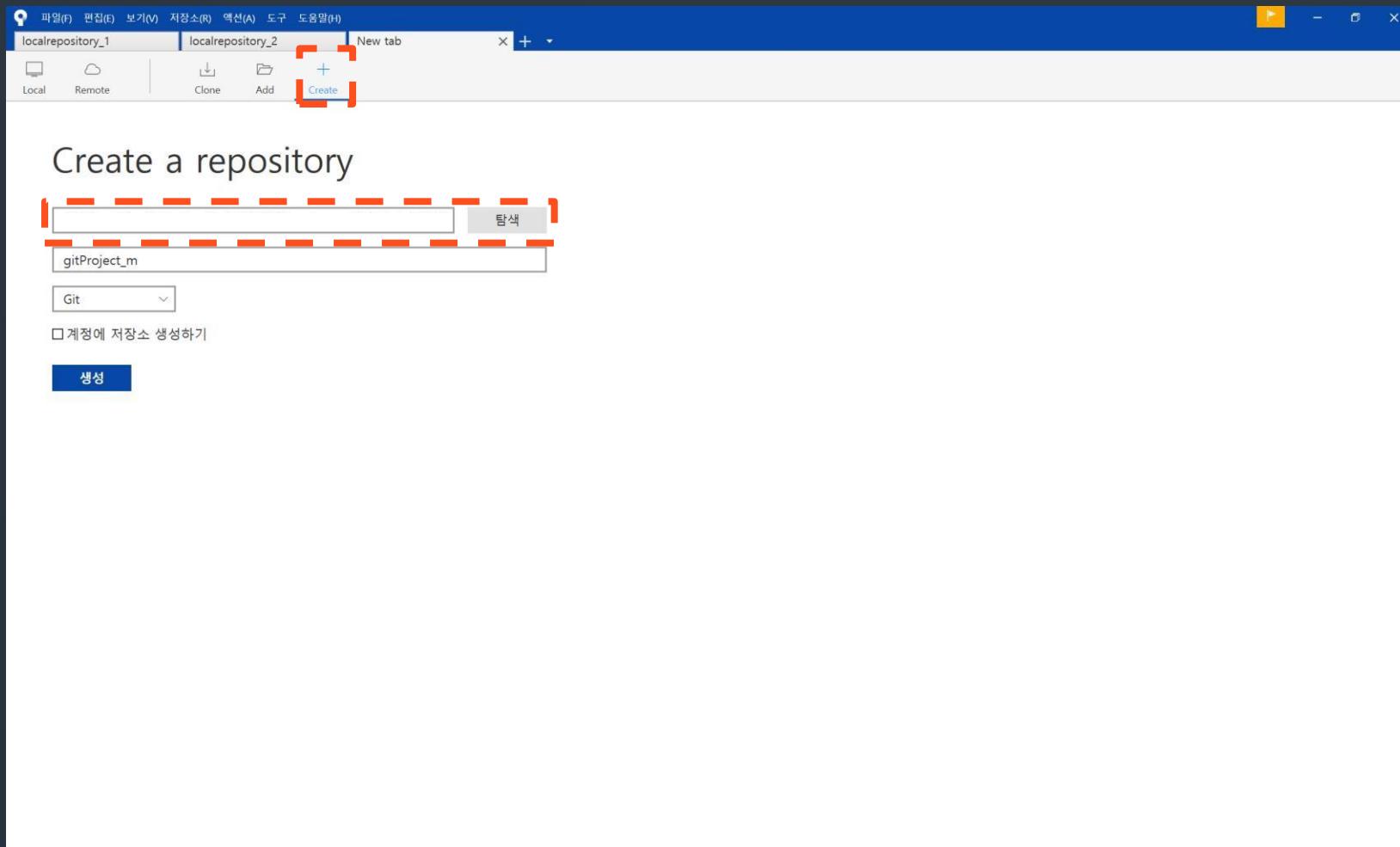
버전관리 예외 설정(프로젝트 관리자)

- **프로젝트 폴더 내 .gitignore 파일을 생성하여 ignore 목록 작성**
- **해당 ignore 내역을 미리 작성해야 협업 시 충돌이 적게 발생함(자동생성 내용 중 .settings/ 는 삭제하고 저장)**



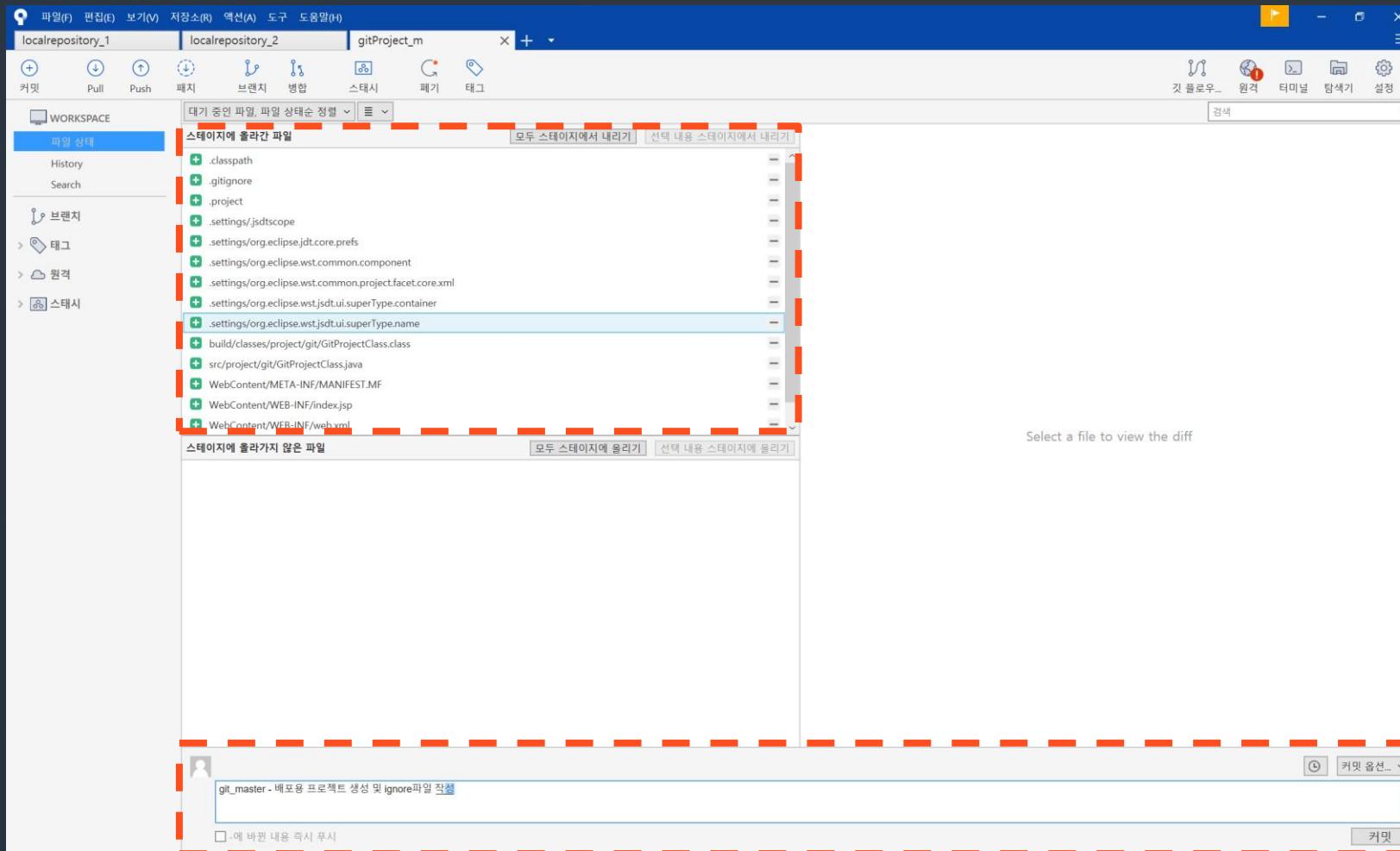
프로젝트를 git으로 관리 설정(프로젝트 관리자)

- **프로젝트 폴더를 탐색하여 git 저장소 설정**



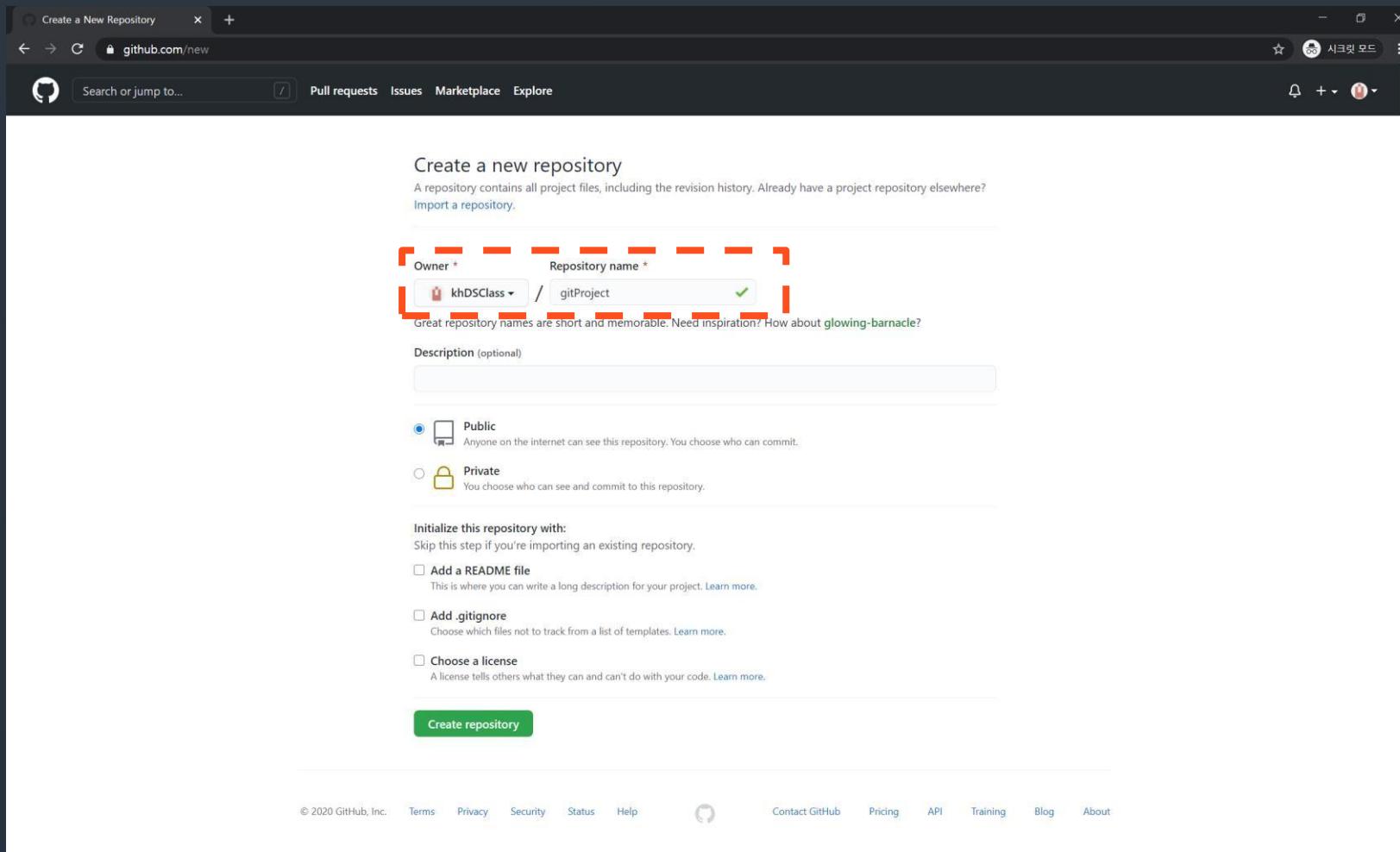
프로젝트를 git으로 관리 설정(프로젝트 관리자)

- 새로 만든 프로젝트 및 .gitignore 파일 커밋



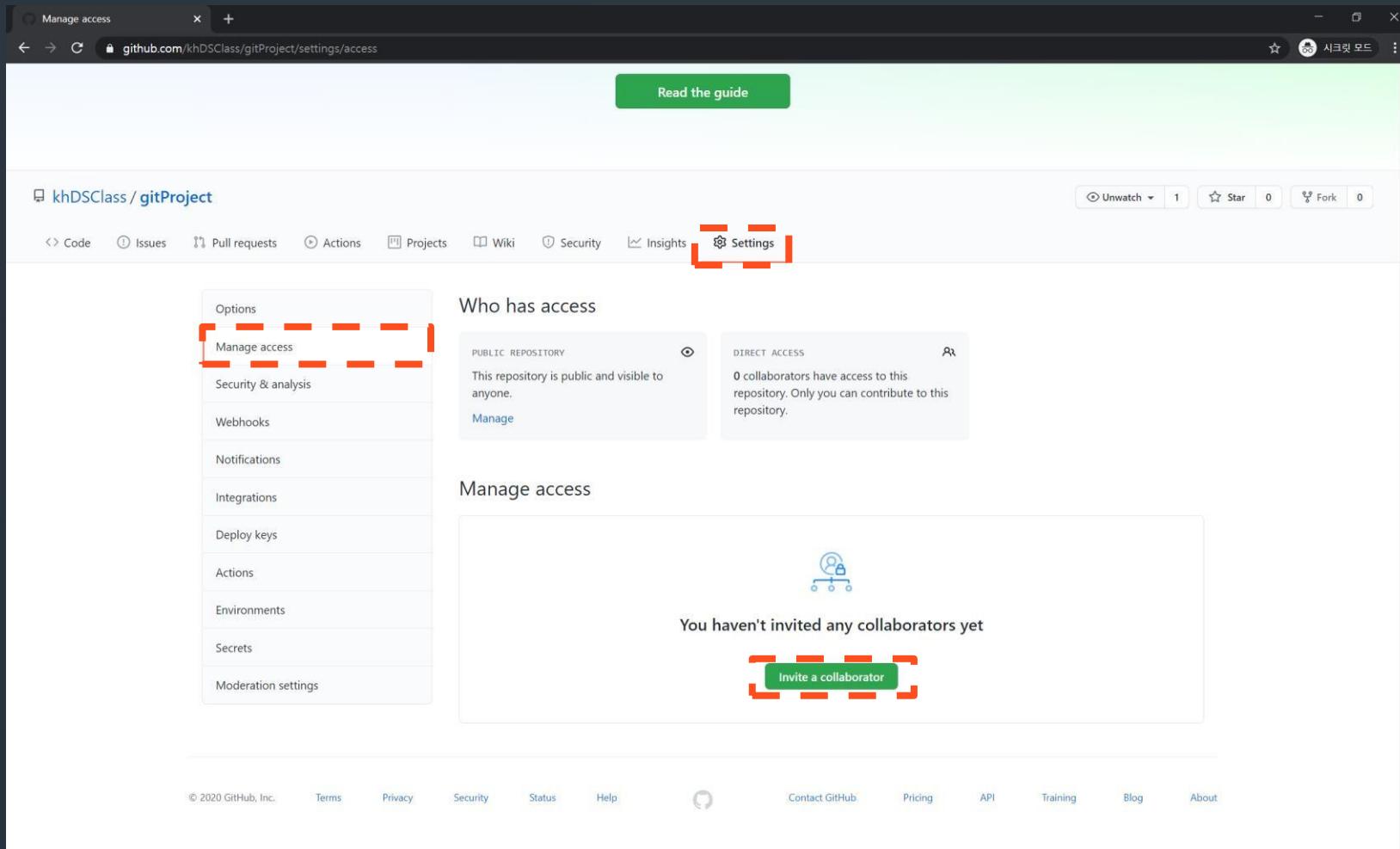
프로젝트를 git으로 관리 설정(프로젝트 관리자)

- github에 새로운 프로젝트를 관리할 새 저장소를 생성



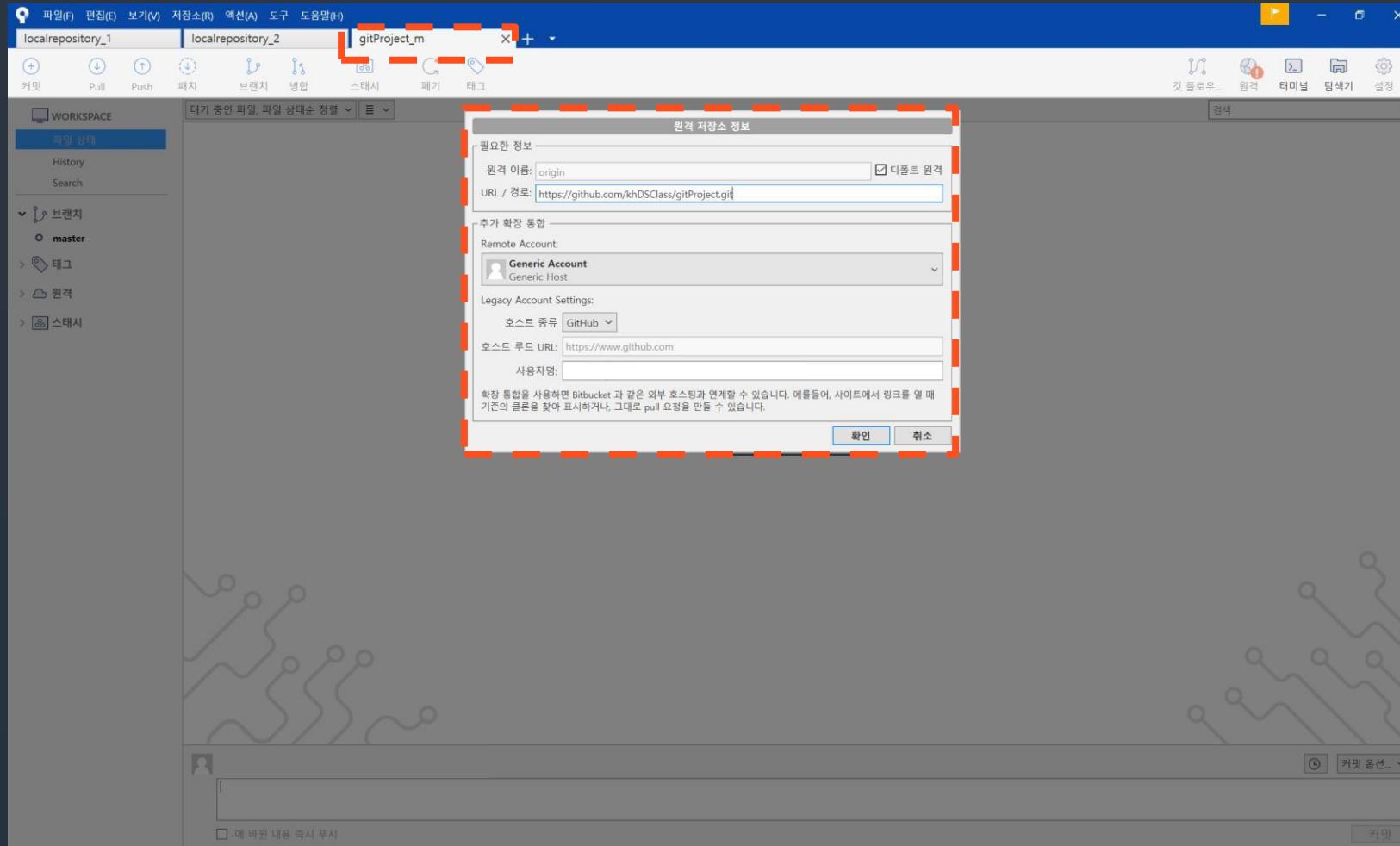
프로젝트를 git으로 관리 설정(프로젝트 관리자)

- 여러 사람이 동일한 원격저장소를 사용해야하기 때문에 원격저장소 계정에서 협업을 위한 초대
- collaborator로 초대 후 승인해야 해당 원격저장소에 push 가능



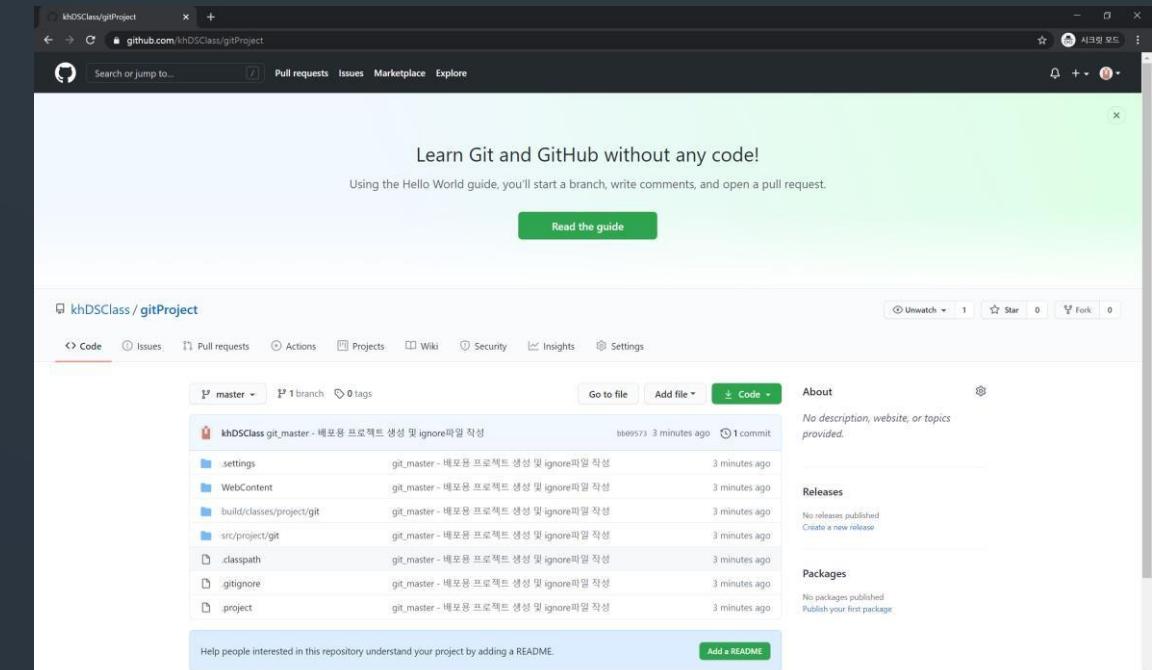
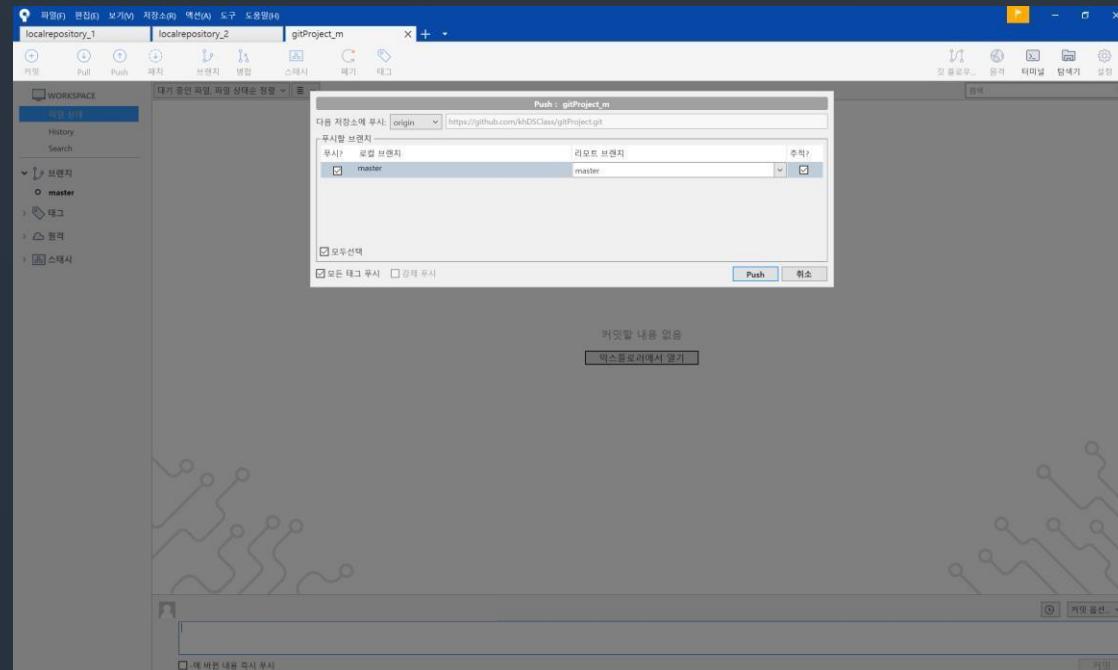
원격저장소 설정 (프로젝트 관리자)

- 새로 생성한 원격저장소를 소스트리에 추가



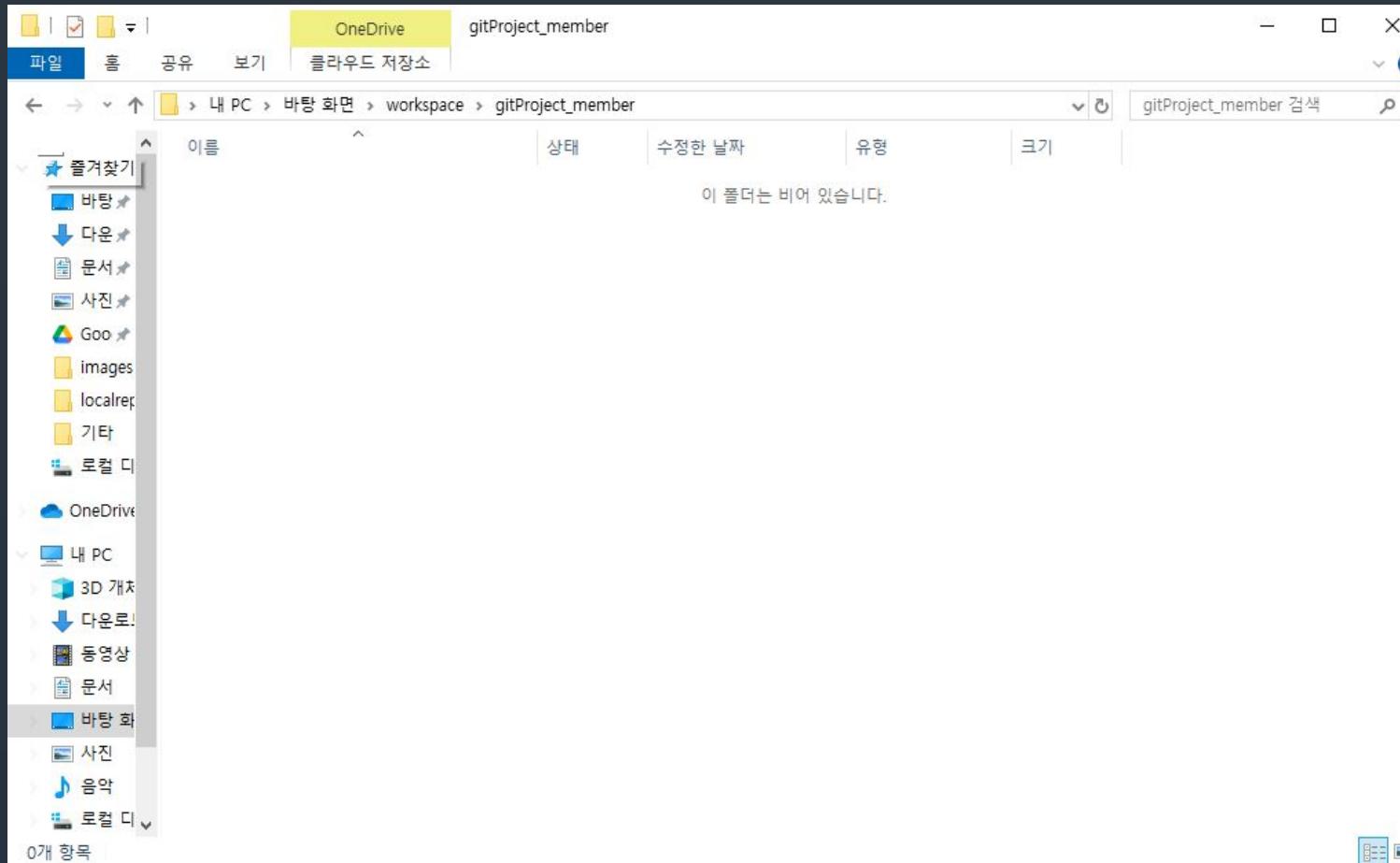
원격저장소 설정 (프로젝트 관리자)

- 최초 커밋사항을 그대로 push
- 원격저장소에서 push된 내용 확인



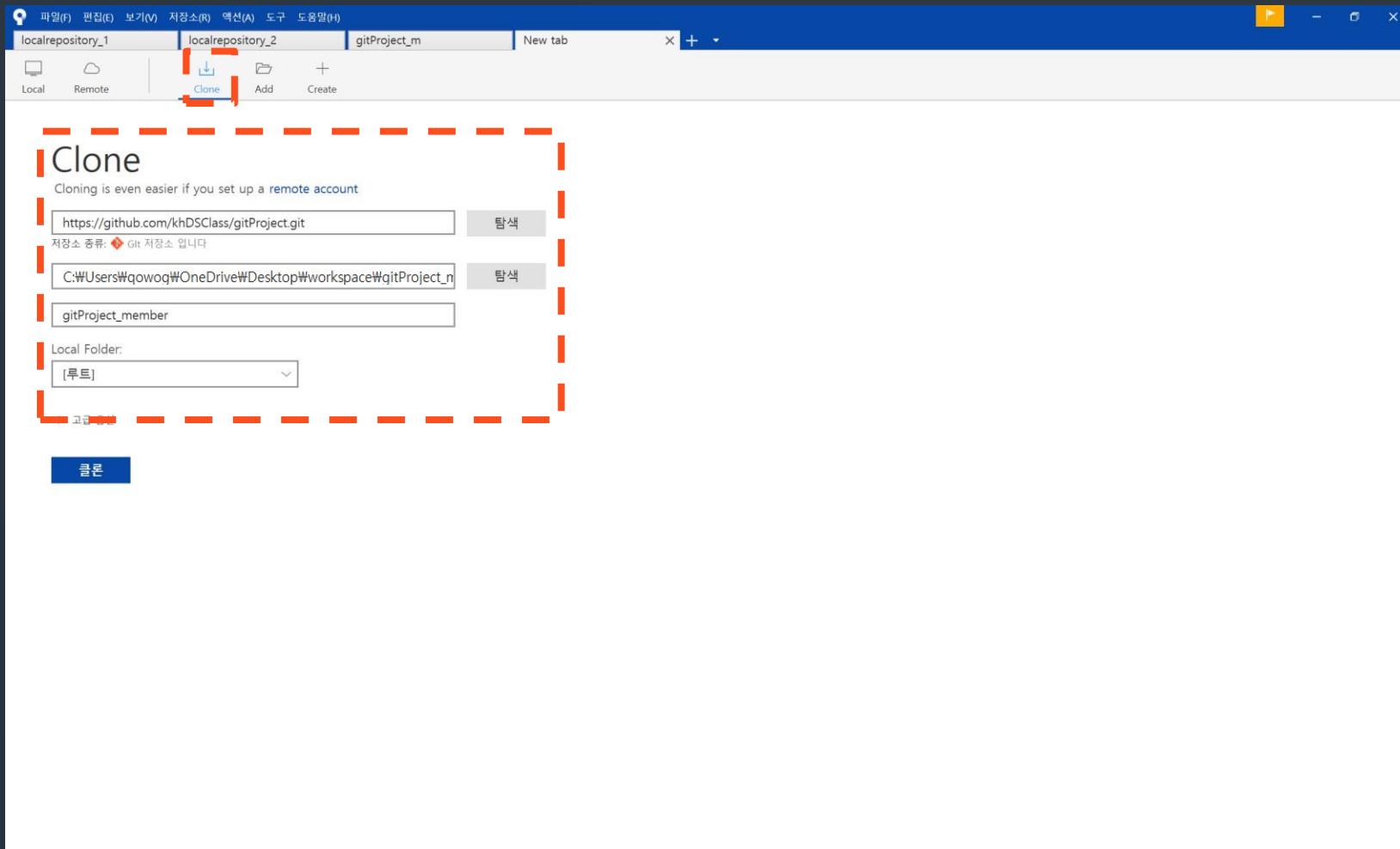
프로젝트 설정 (프로젝트 팀원)

- workspace에 프로젝트로 사용할 새 폴더 생성(프로젝트를 clone할 폴더)
- github collaborator 설정 수락



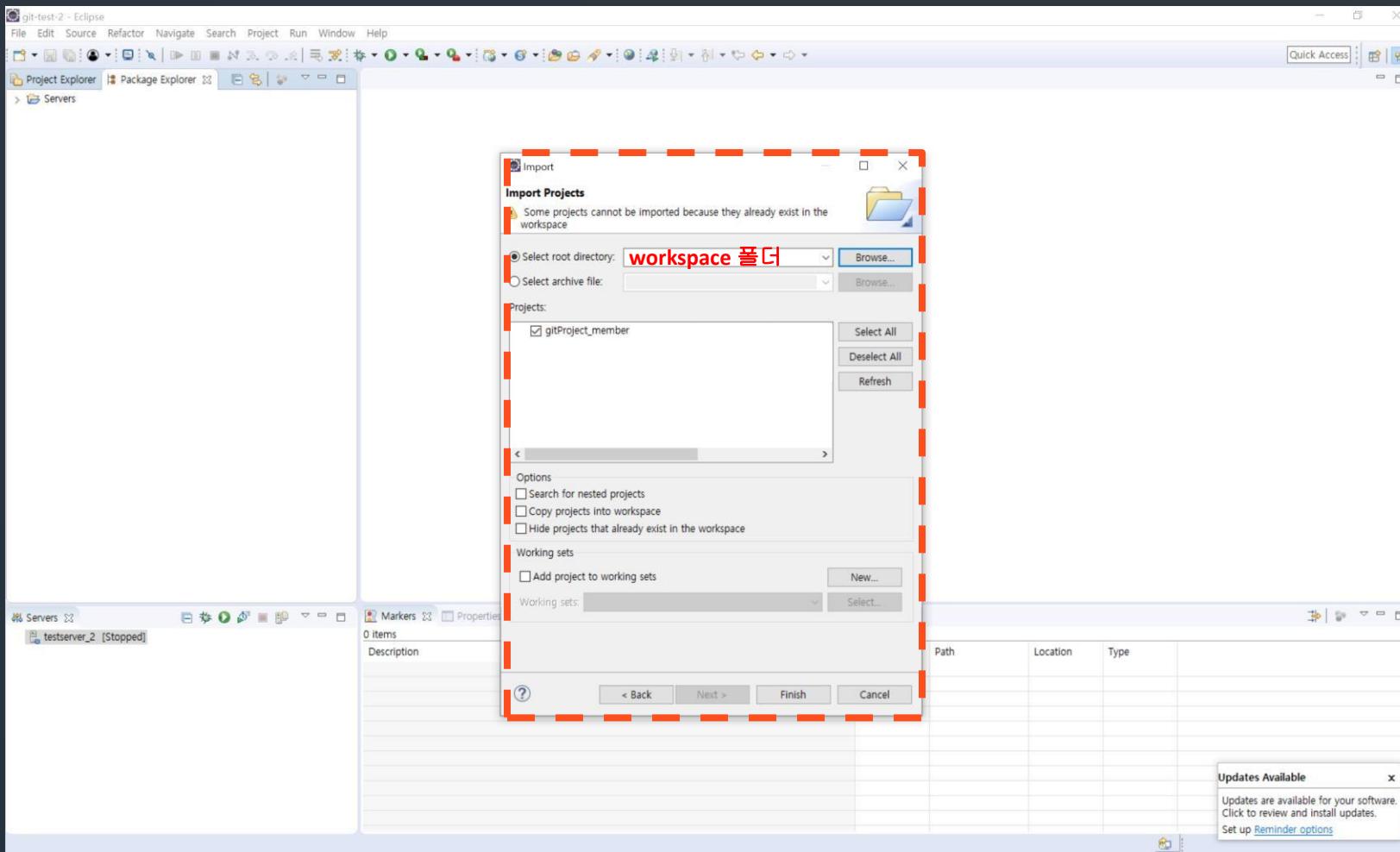
프로젝트 설정 (프로젝트 팀원)

- 새로 생성된 폴더에 github를 clone



프로젝트 설정 (프로젝트 팀원)

- clone한 폴더를 eclipse에서 import





프로젝트 설정 (프로젝트 팀원)

- 마스터에서 작성한 index.html 파일 확인



협업 작업원칙

- 모든 프로젝트 팀원은 개인의 branch에서 작업
- 작업이 끝나면 master-branch와 병합하여 충돌을 해결한 후 push(충돌이 없다면 바로 push)
- 충돌을 해결하여 push를 하게 되면 원격 저장소에는 에러가 없는 버전만 공유
- 원격 저장소에 충돌과 오류가 있는 상태가 되면 다른 사람들도 모두 영향을 받음

작업 순서

1. 시작 시 무조건 master-branch로 pull 진행

내가 push한 이후 다른 팀원이 push를 했을 가능성이 있으므로 변경사항이 없어도 무조건 pull

2. 개인 작업 branch로 master-branch 병합

3. 코드 작성

4. 코드 작성 후 master-branch에서 다시 pull

내가 코드를 작성하는 사이 다른 push내역이 존재할 수 있기 때문에 pull 먼저 진행

5. master를 현재 작업완료 된 branch로 병합하여 충돌내역 확인 후 해결

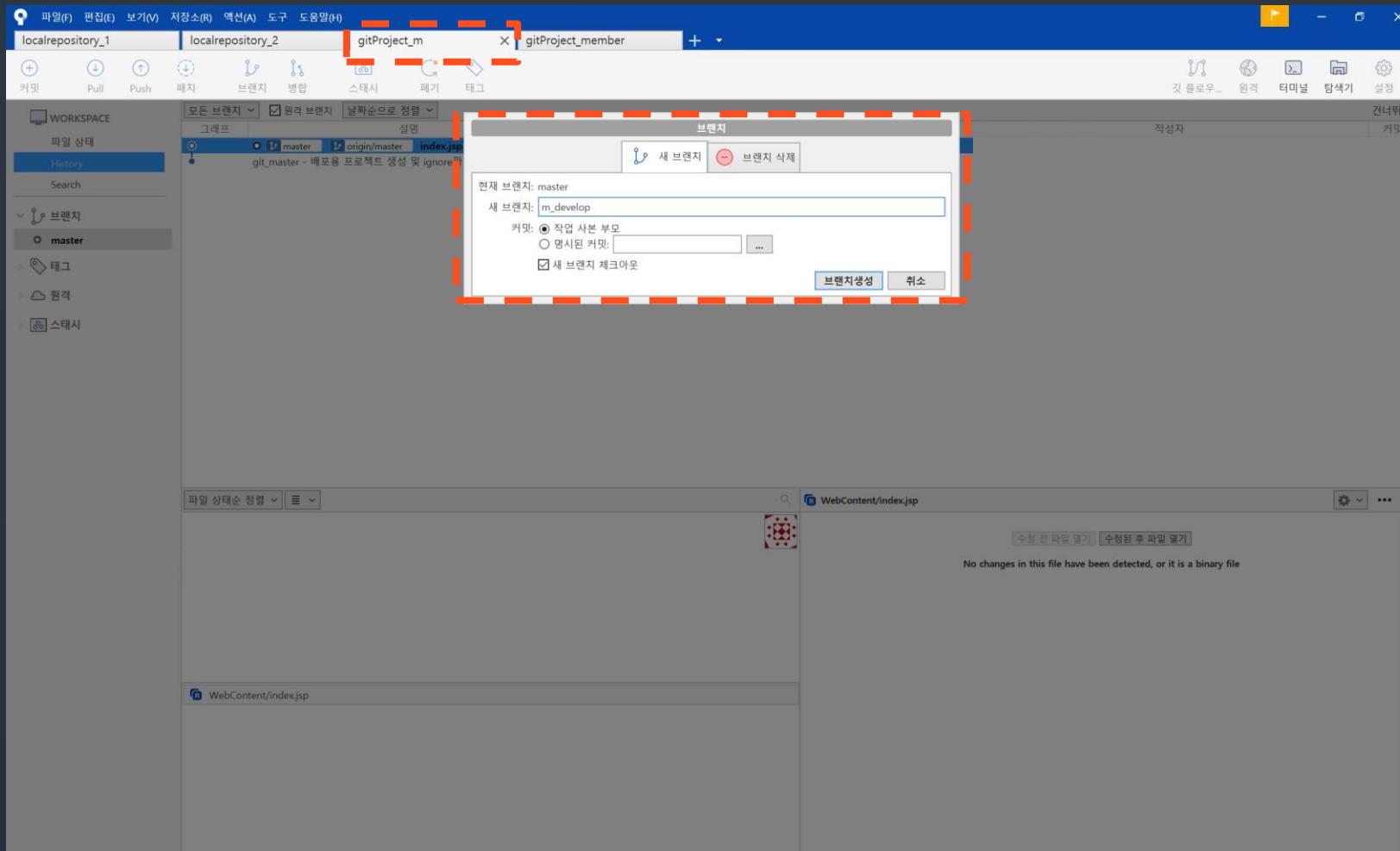
6. master로 현재 작업 branch를 병합하여 push

7. 하루에 한번 이상은 push를 진행하는 것을 권고



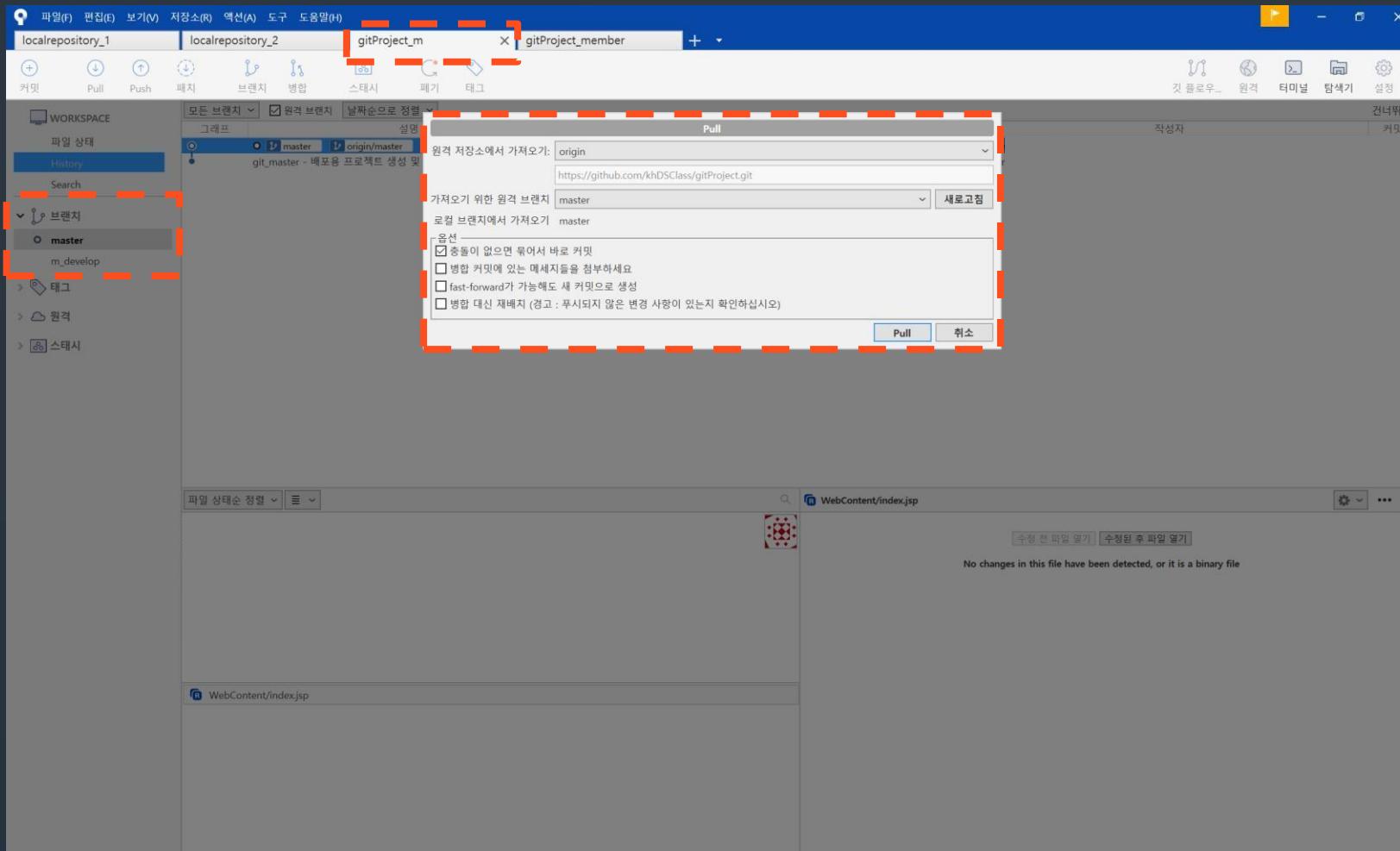
프로젝트 진행(프로젝트 관리자)

- 개인 작업용 branch 생성



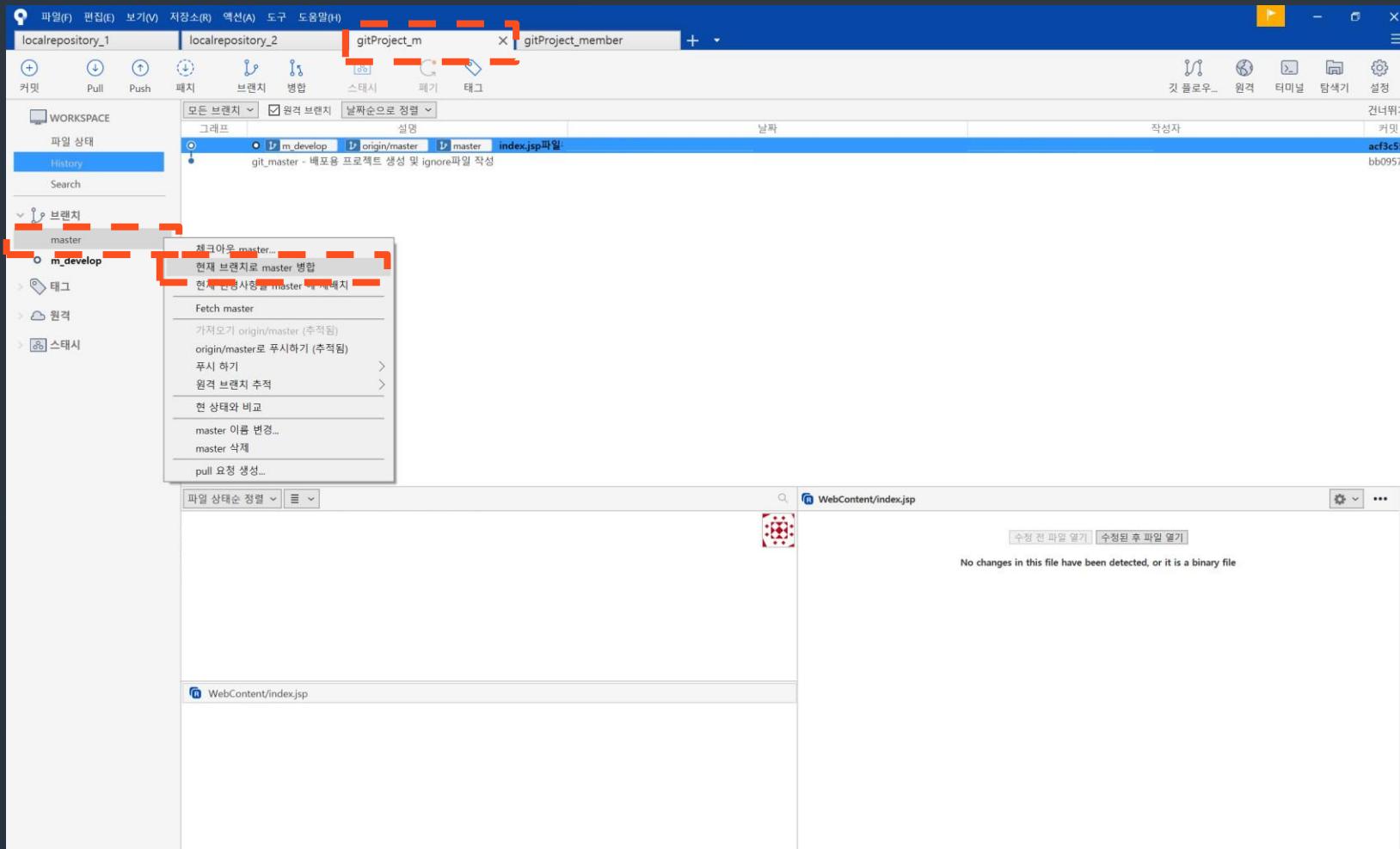
프로젝트 진행(프로젝트 관리자)

- master-branch에서 현재 원격저장소의 최신버전을 pull



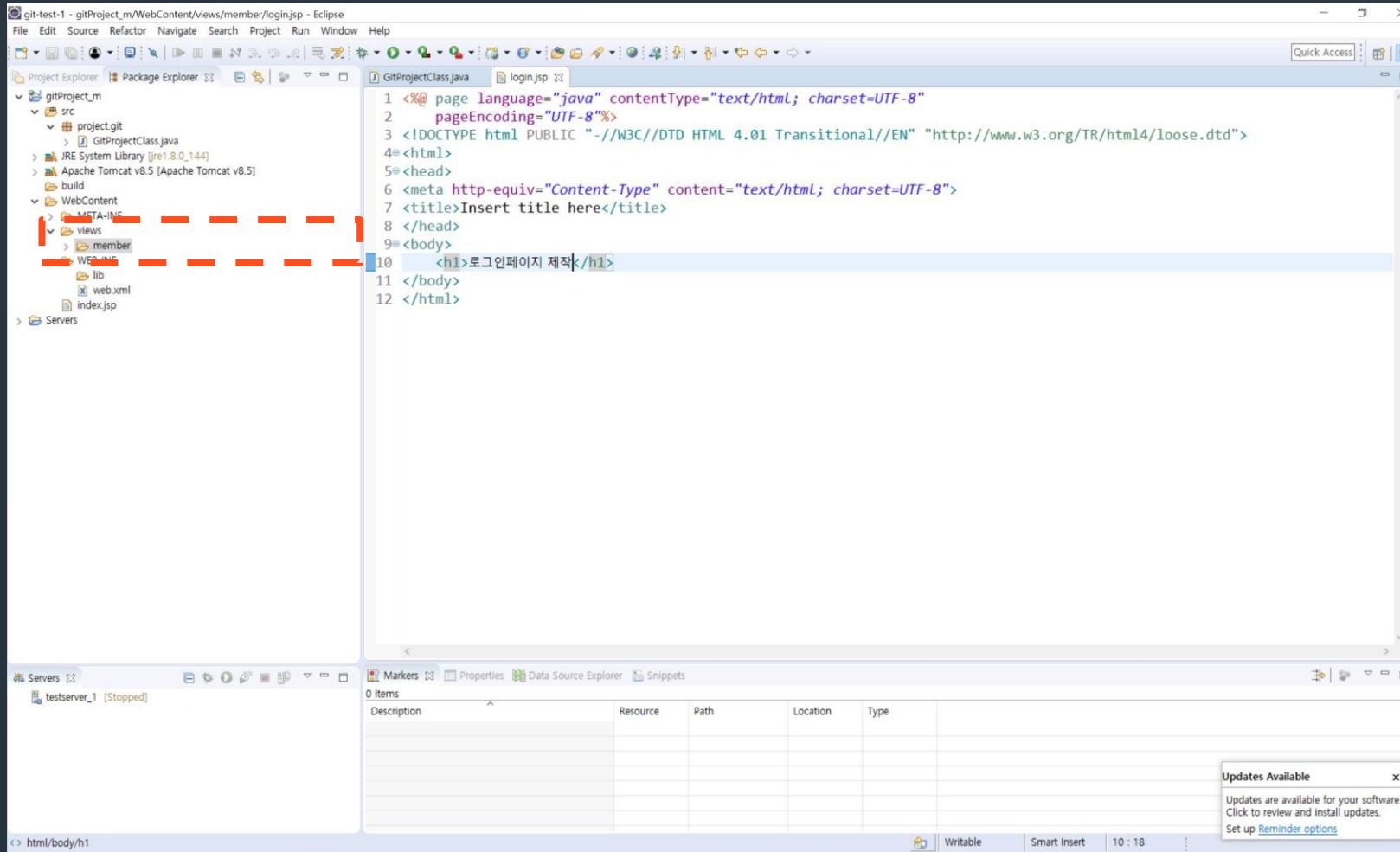
프로젝트 진행(프로젝트 관리자)

- 작업용 branch에서 현재 원격저장소의 최신버전을 병합하여 버전 적용
- 현재 테스트 상황은 pull된 버전도 동일하므로 변동사항없음



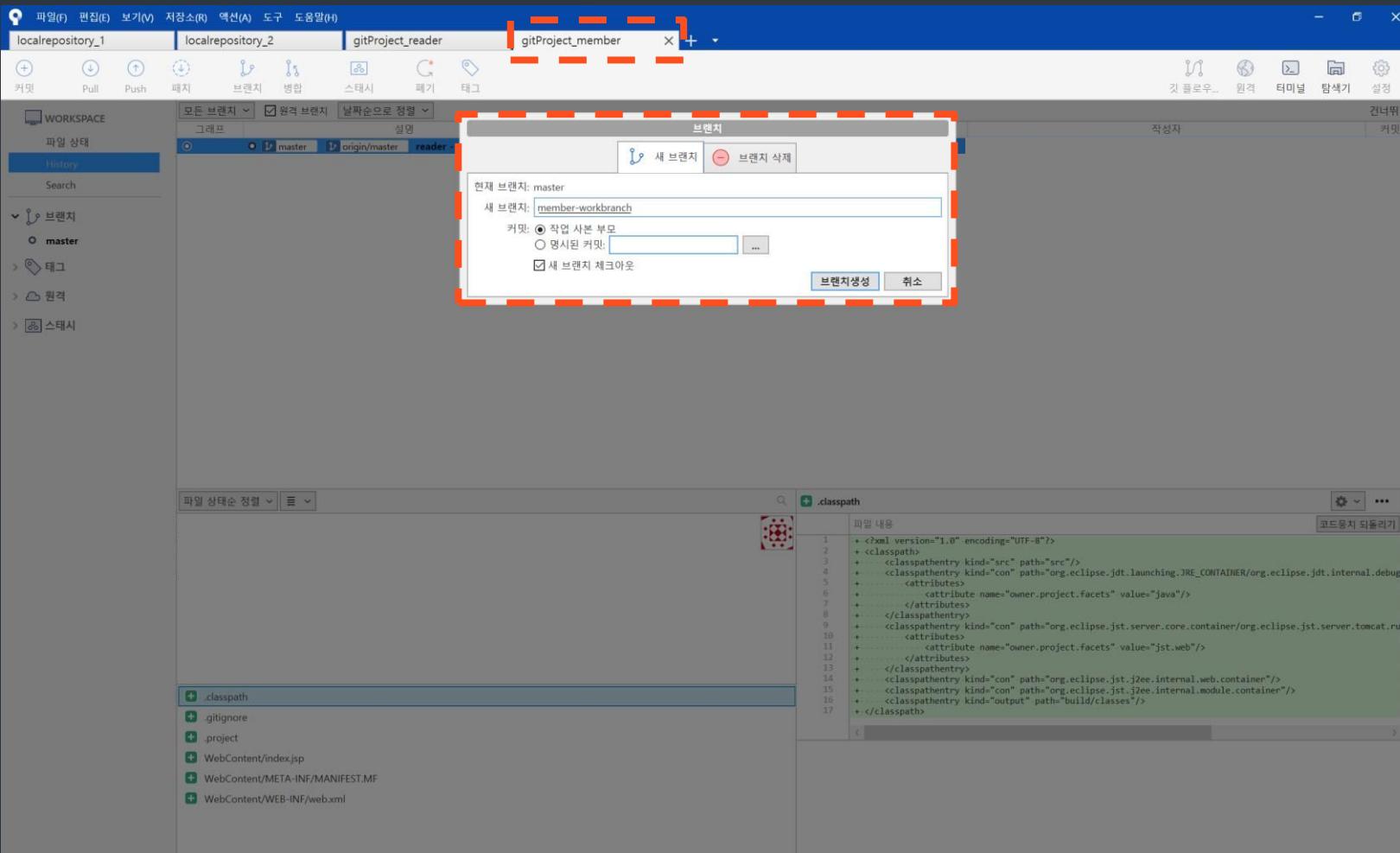
프로젝트 진행(프로젝트 관리자)

- 작업용 branch에서 코드 작성



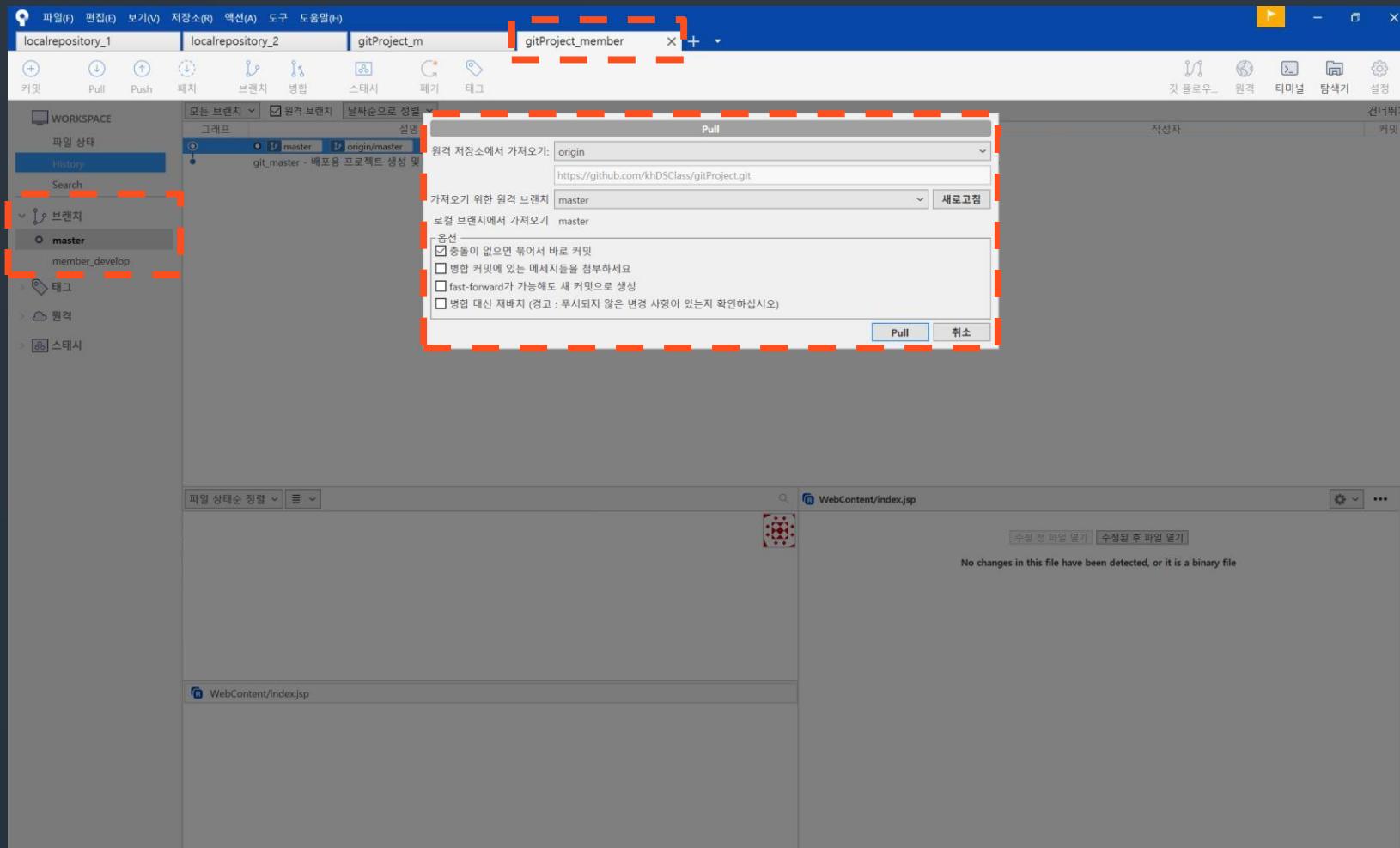
프로젝트 설정 (프로젝트 팀원)

- **프로젝트 관리자가 작업중인 동시에 프로젝트 팀원도 작업 시작**
- **작업용 branch 생성**



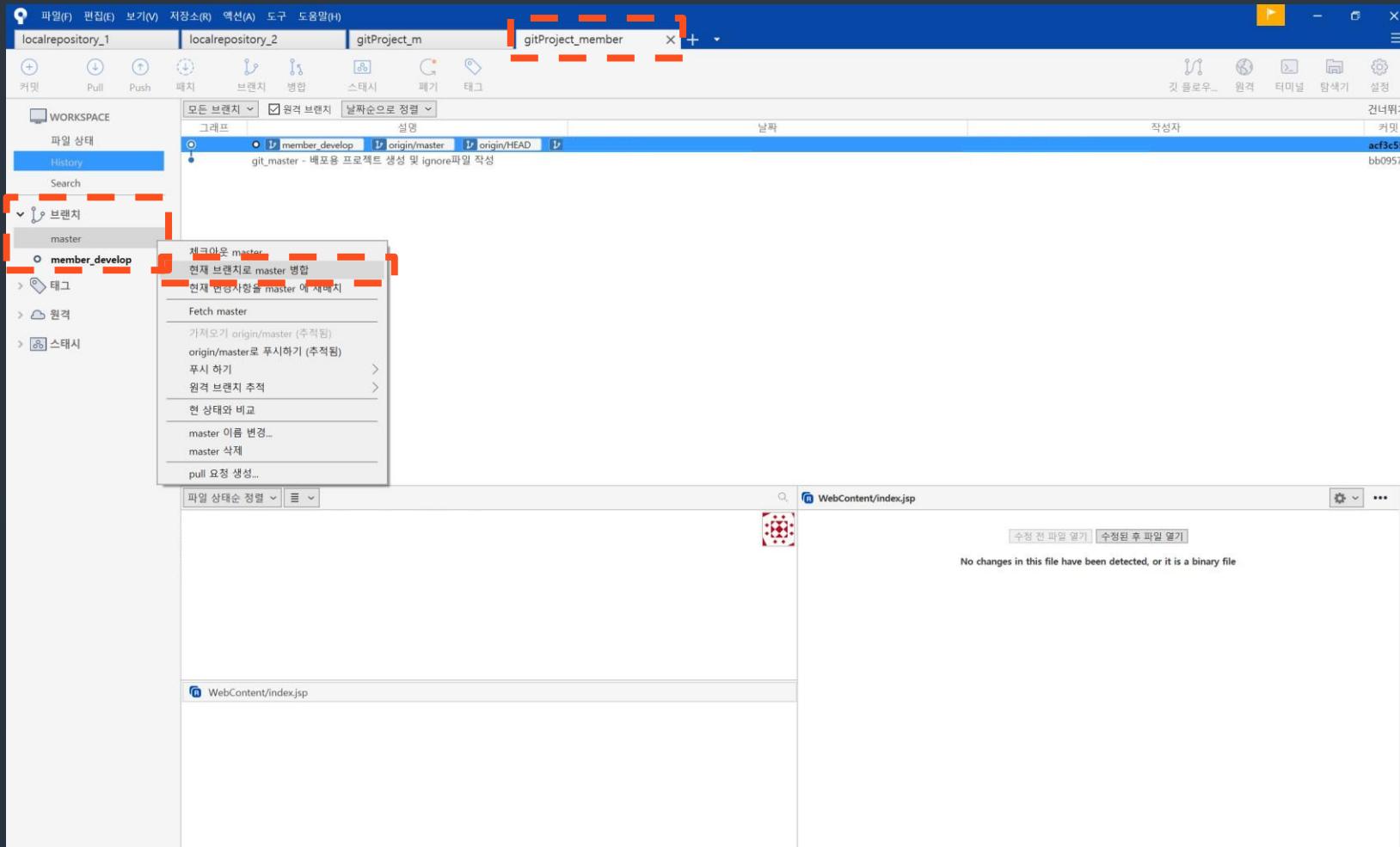
프로젝트 진행(프로젝트 팀원)

- master-branch에서 현재 원격저장소의 최신버전을 pull



프로젝트 진행(프로젝트 팀원)

- 작업용 branch에서 현재 원격저장소의 최신버전을 병합하여 버전 적용



프로젝트 진행(프로젝트 팀원)

- 작업용 branch에서 코드 작성

The screenshot shows the Eclipse IDE interface with the following details:

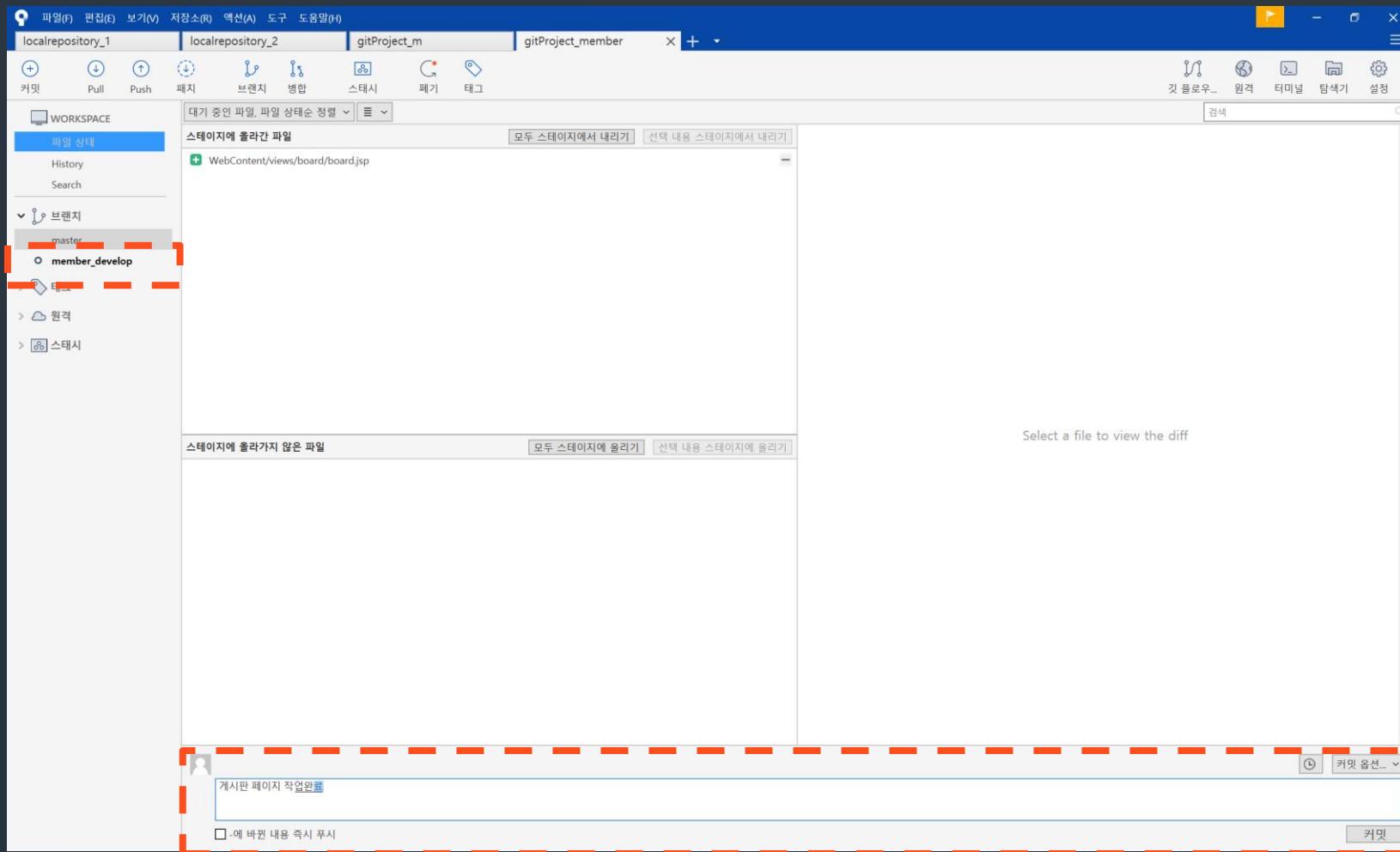
- Project Explorer:** Displays the project structure for "git-test-2 - gitProject_member". It includes a "src" folder containing "project.git" and "GitProjectClass.java", and a "WebContent" folder containing "WEB-INF", "views", and "board". The "board" folder contains "board.jsp".
- Editor:** Shows the content of "board.jsp". The code is as follows:

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2 pageEncoding="UTF-8"%>
3 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
4<html>
5<head>
6 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
7 <title>Insert title here</title>
8 </head>
9<body>
10 <h1>게시판 페이지 작성</h1>
11 </body>
12 </html>
```

- Servers:** Shows a single server named "testserver_2 [Stopped]".
- Markers:** Shows a table with 0 items.

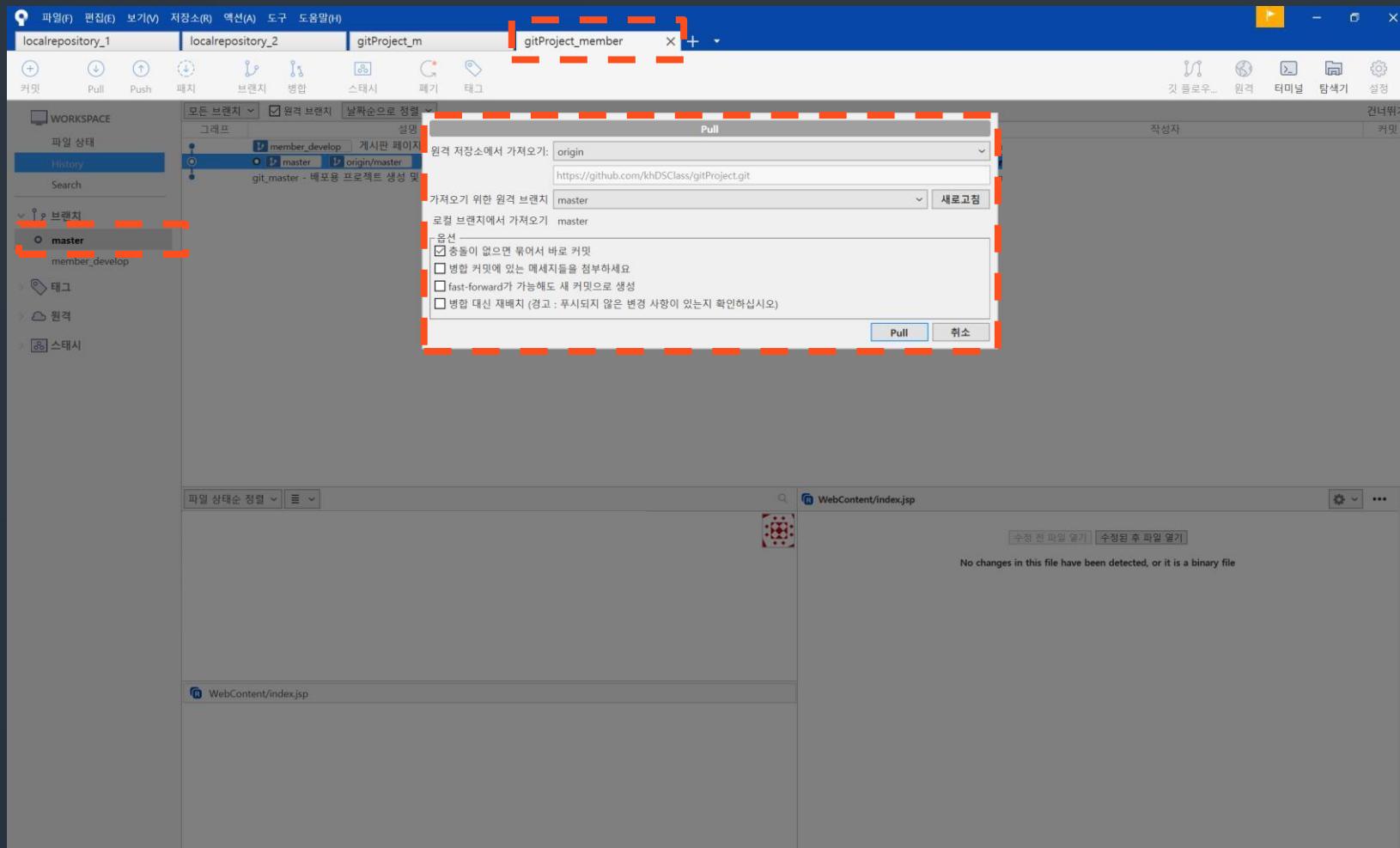
프로젝트 진행(프로젝트 팀원)

- 작업용 branch에서 코드 작성이 완료된 후 커밋



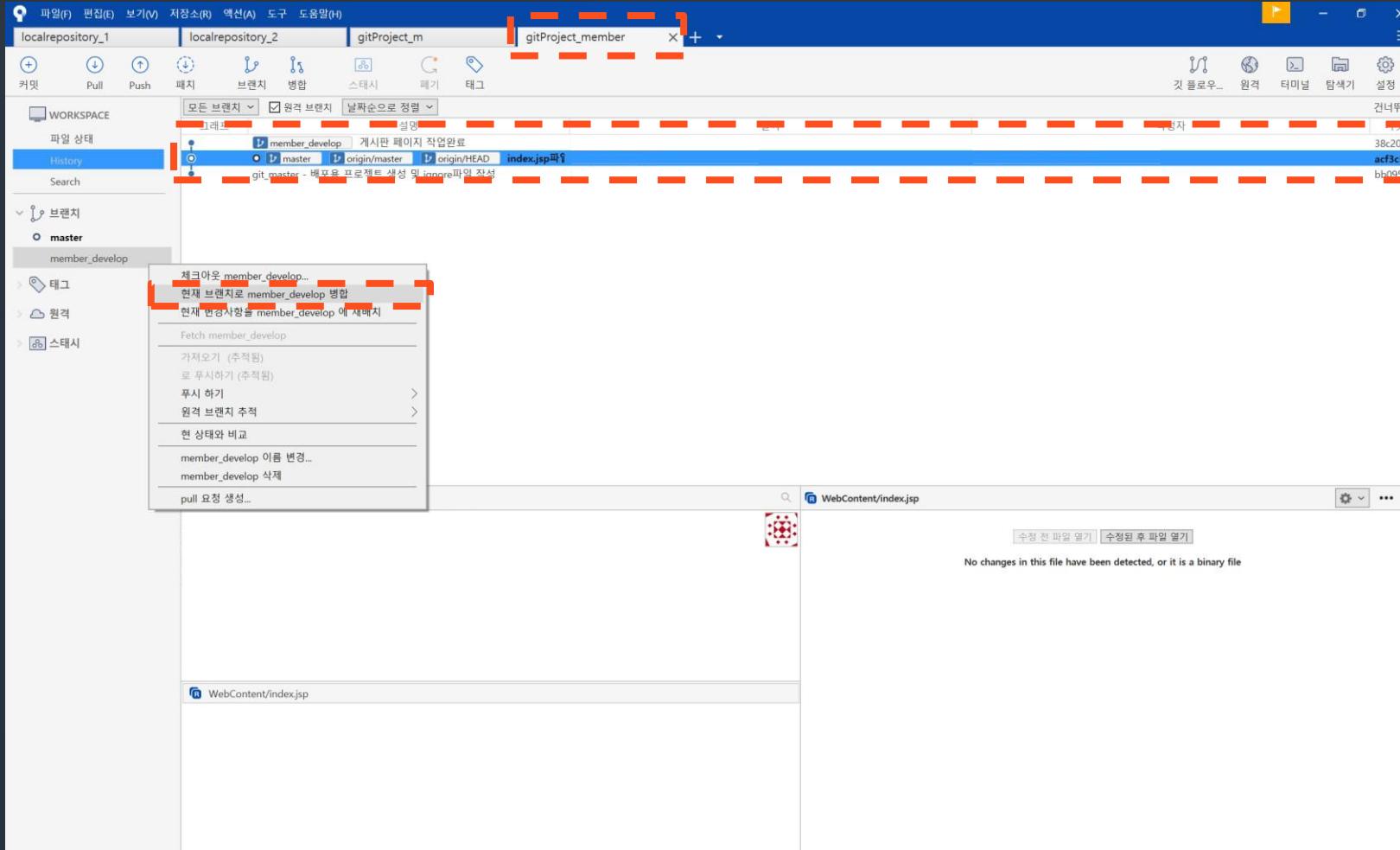
프로젝트 진행(프로젝트 팀원)

- 코드 작성 중 다른 팀원이 작업은 완료하고 push 했을 수 있으므로 master-branch에서 최신버전 pull



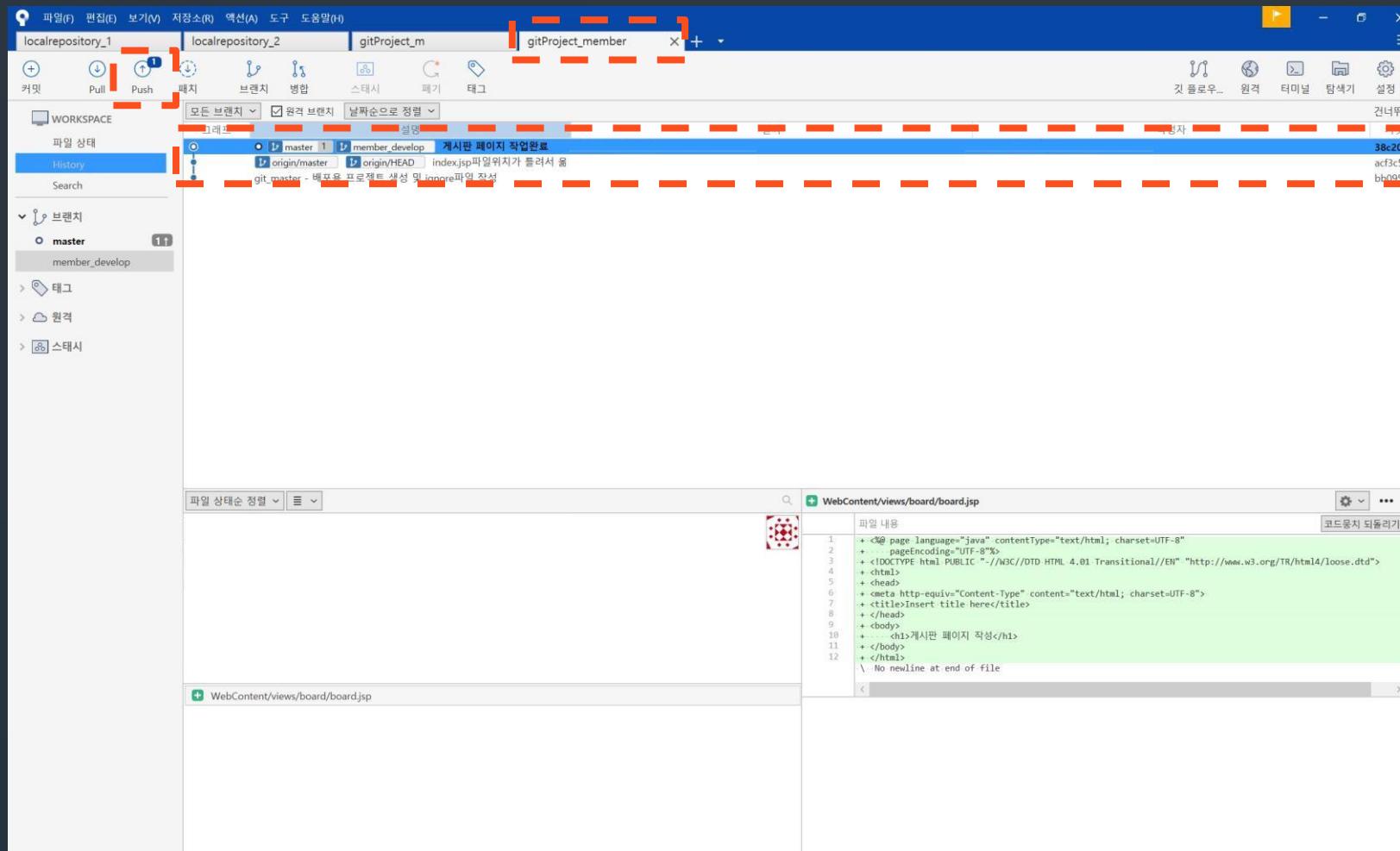
프로젝트 진행(프로젝트 팀원)

- 현재 master-branch는 이전상태와 달라진 게 없음
- master-branch에서 작업branch의 작업내용 병합



프로젝트 진행(프로젝트 팀원)

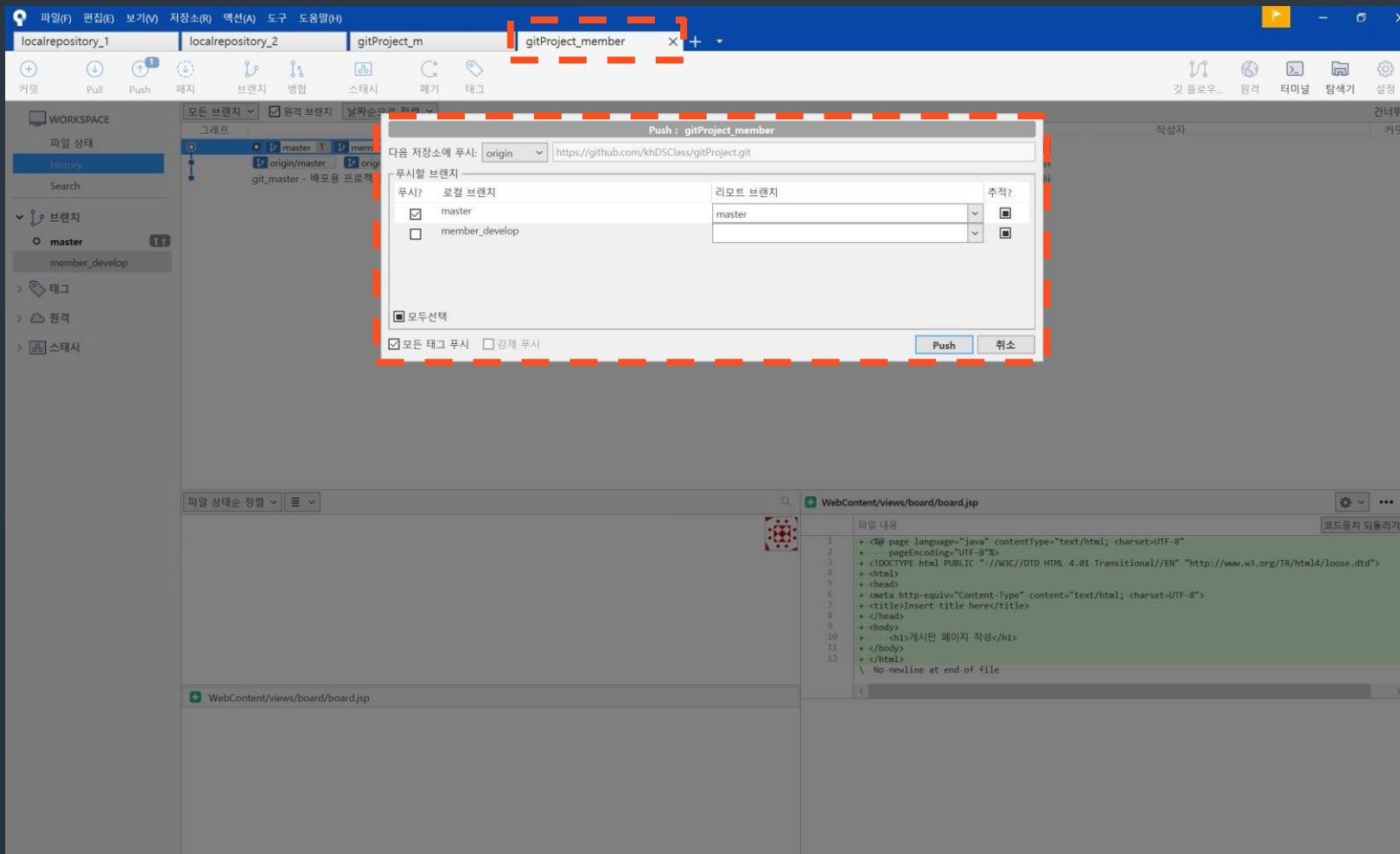
- 병합된 버전을 push





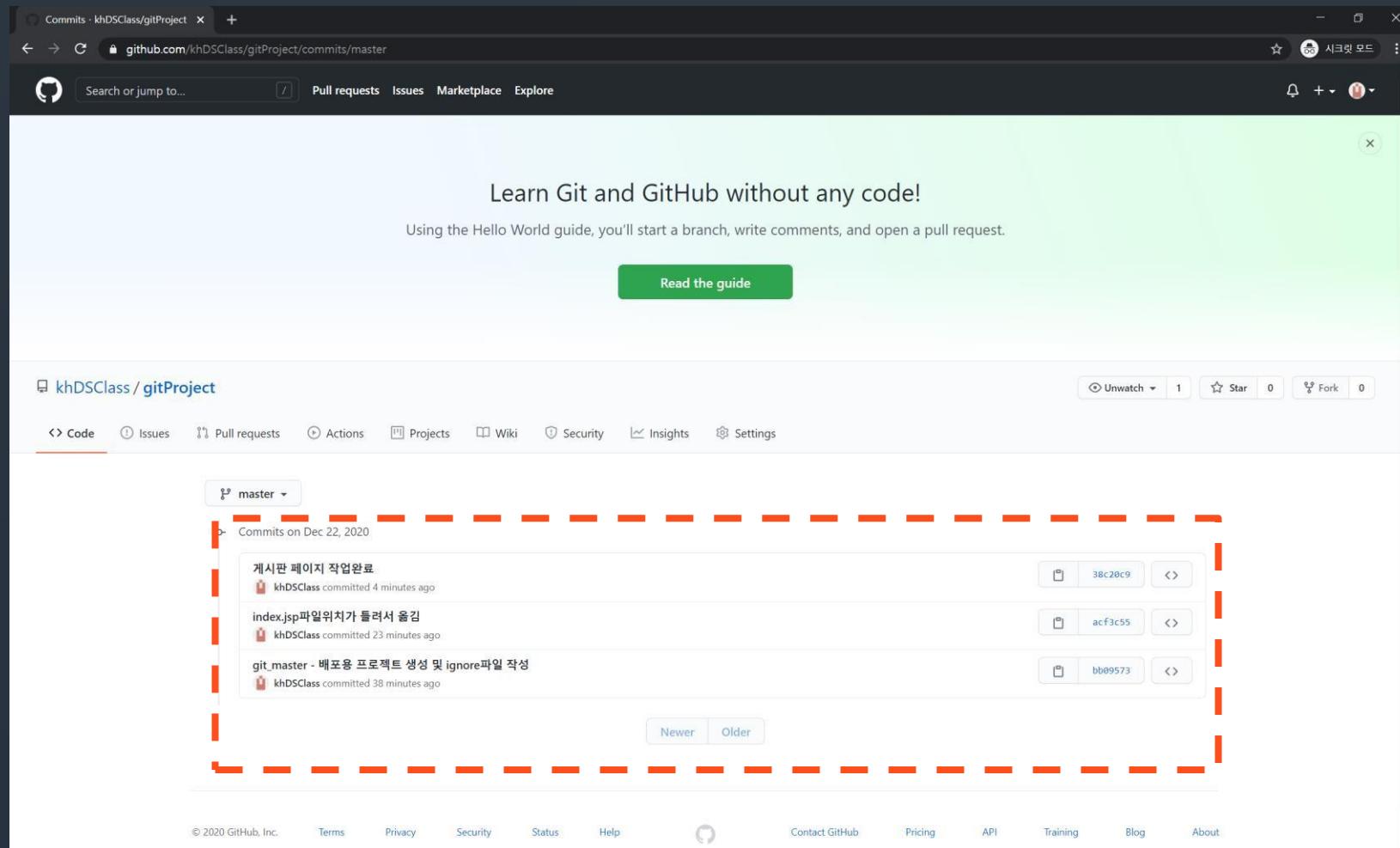
프로젝트 진행(프로젝트 팀원)

- master-branch를 push



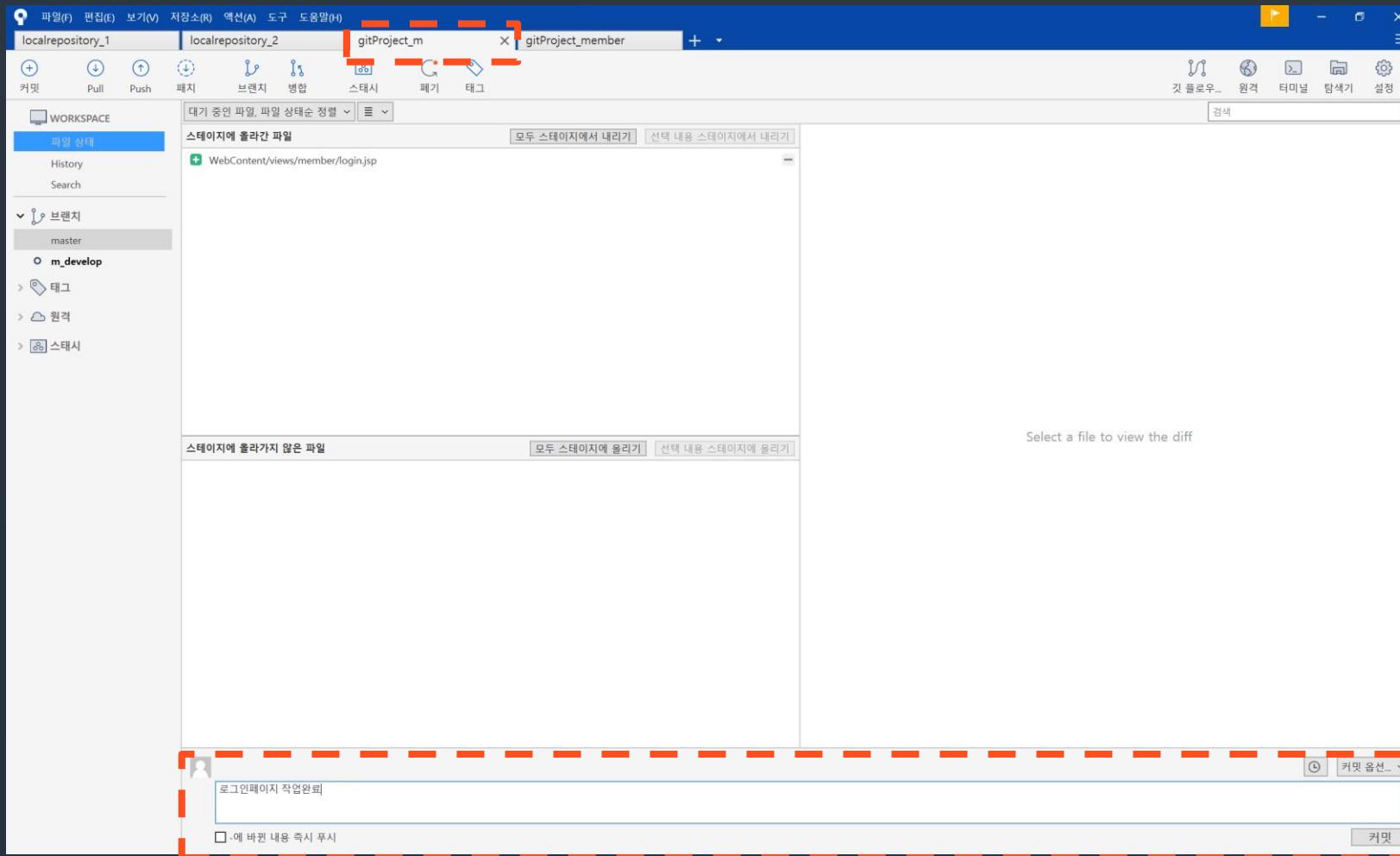
프로젝트 진행(프로젝트 팀원)

- 원격저장소의 push내역



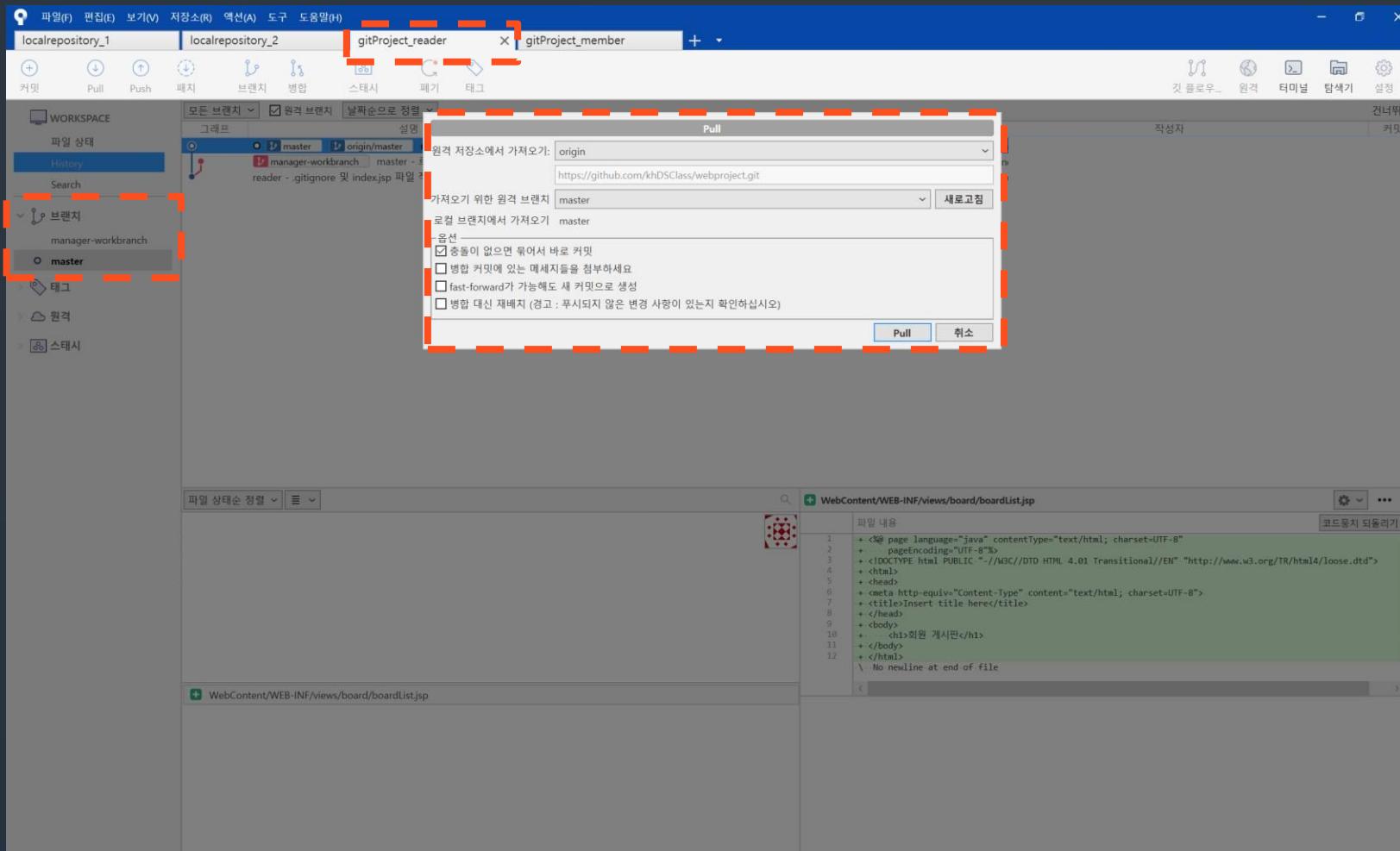
프로젝트 진행(프로젝트 관리자)

- 관리자 측 또한 작업이 완료된 경우 commit 진행



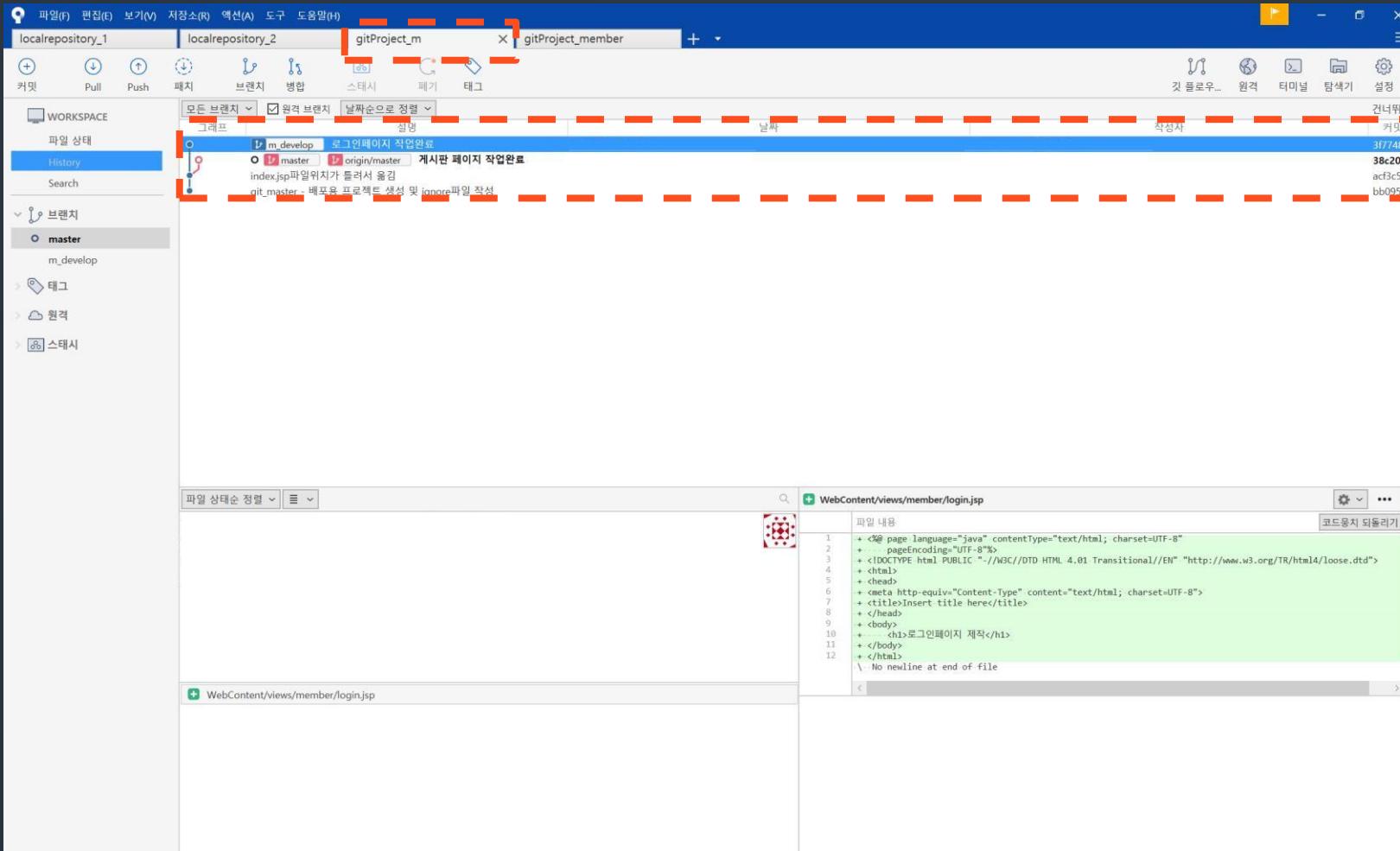
프로젝트 진행(프로젝트 관리자)

- 코드 작성 중 다른 팀원이 작업은 완료하고 push 했을 수 있으므로 master-branch에서 최신버전 pull



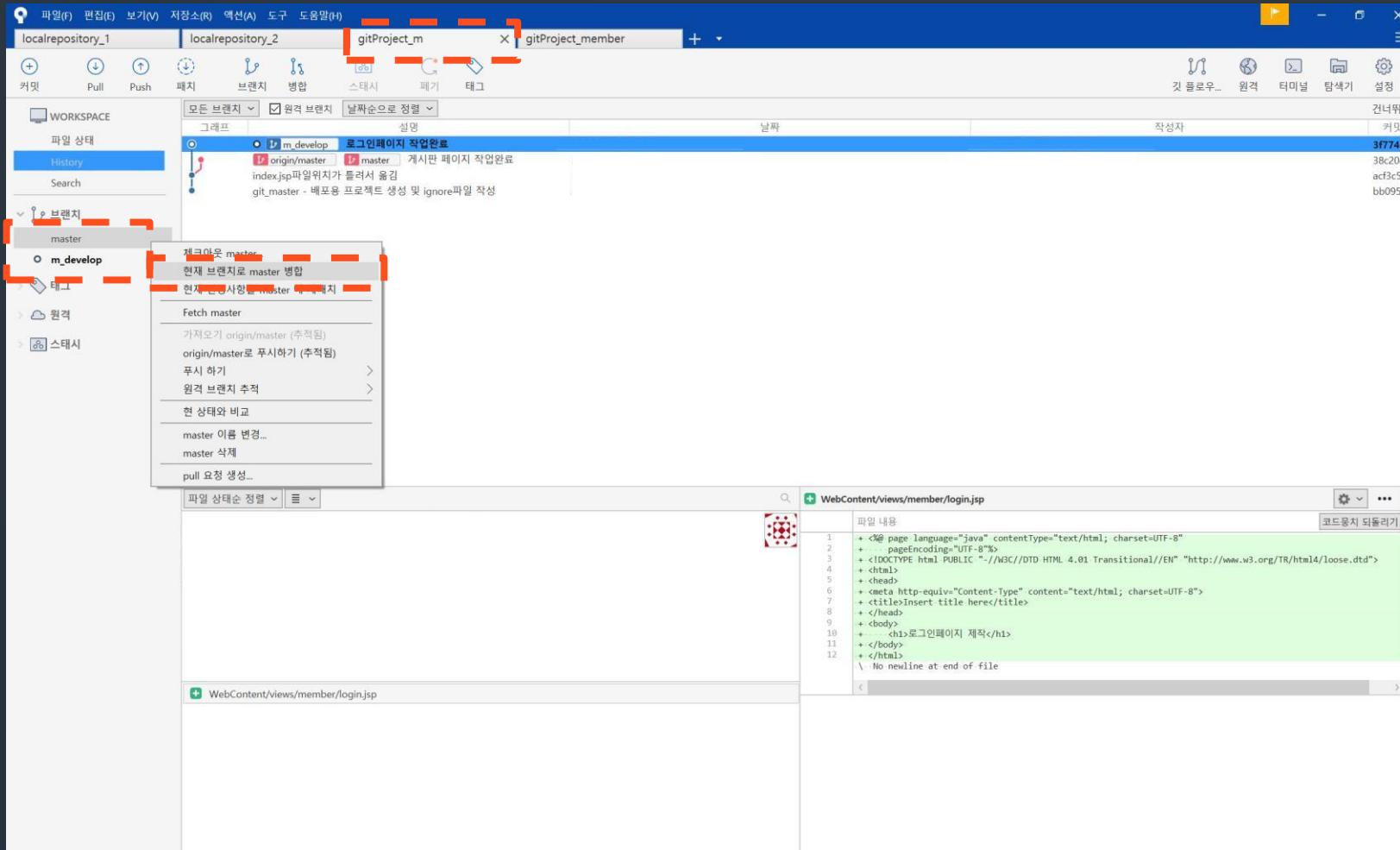
프로젝트 진행(프로젝트 관리자)

- master-branch에 다른 팀원이 작업한 내역이 적용 되었음
- 적용된 버전과 내가 작성한 버전이 충돌 할 가능성 있음



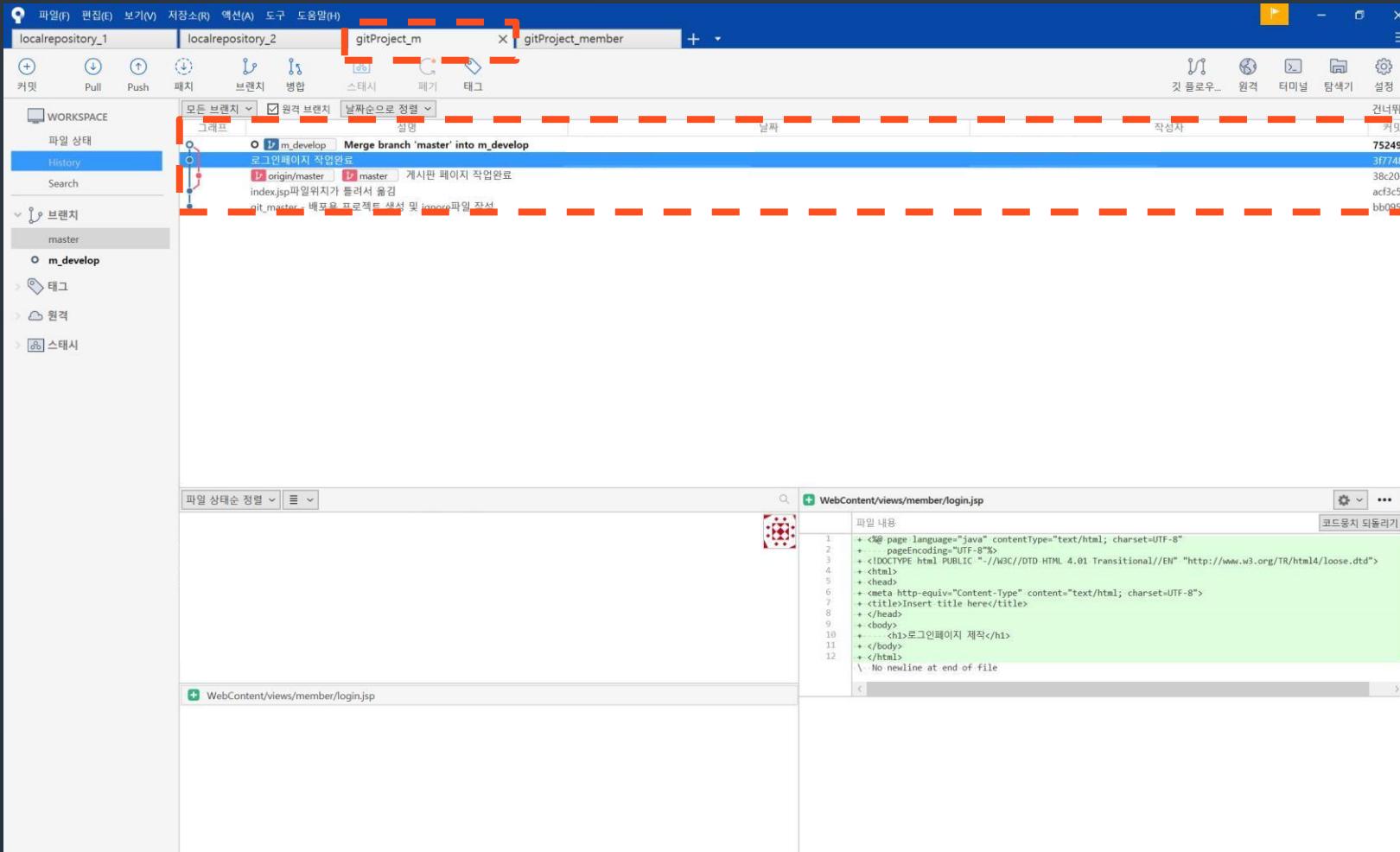
프로젝트 진행(프로젝트 관리자)

- 작업용 branch에서 master-branch를 병합하여 충돌여부 확인



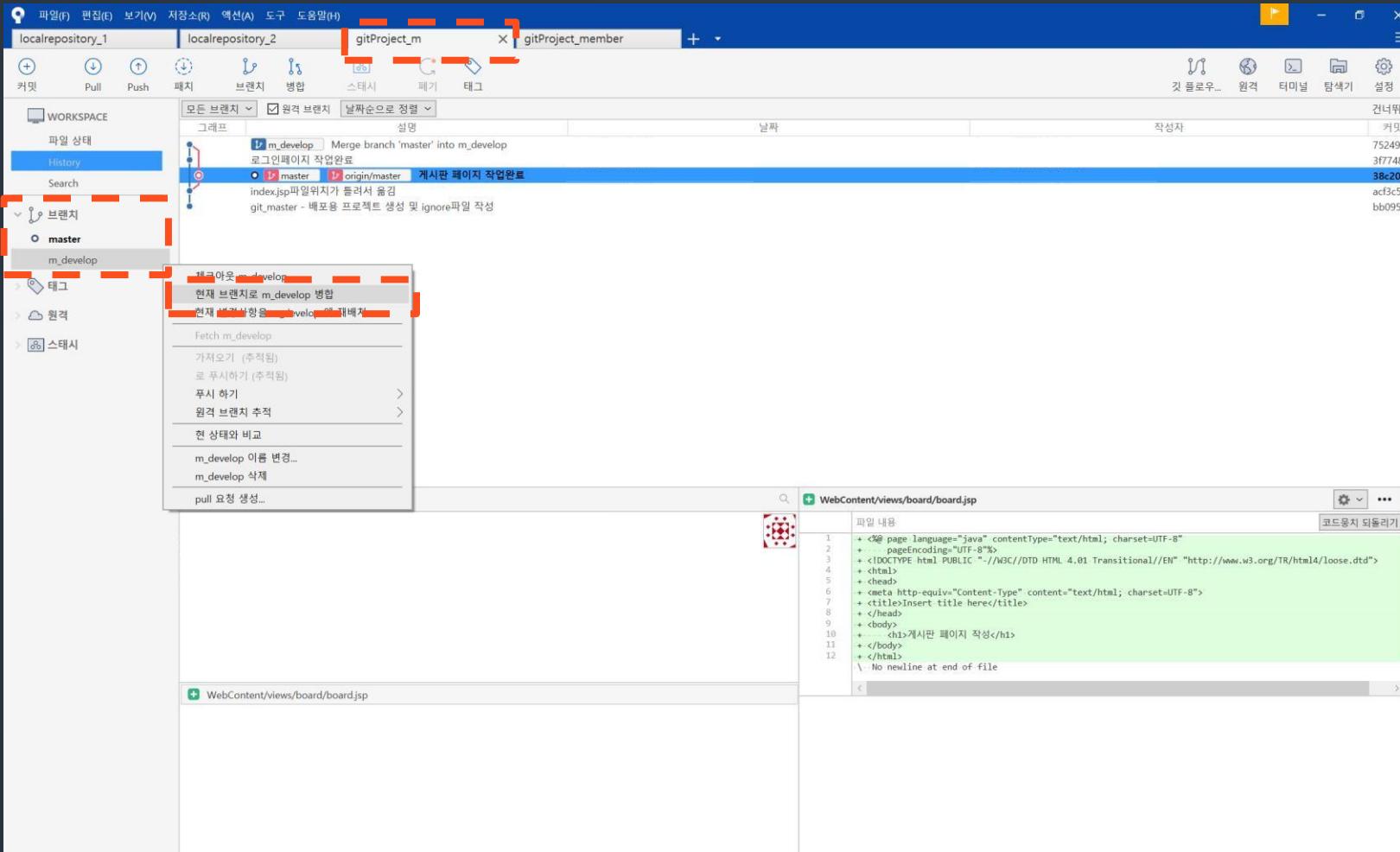
프로젝트 진행(프로젝트 관리자)

- 충돌이 없는 경우 합쳐진 새로운 버전 생성
- 충돌이 발생한 경우 push를 진행한 팀원과 협의하여 충돌해결(해결은 현재 충돌이 발생한 본인이 해결)



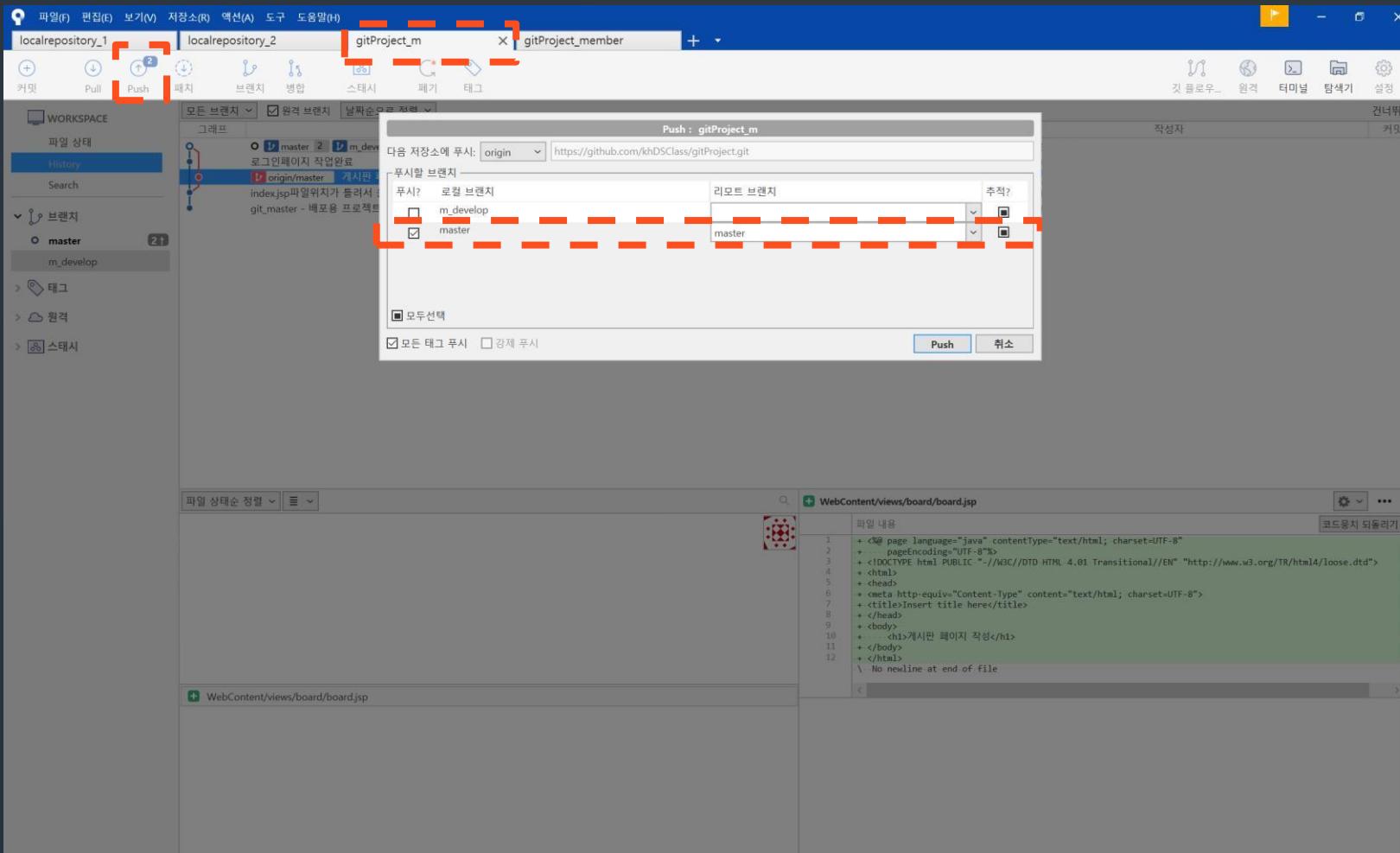
프로젝트 진행(프로젝트 관리자)

- 충돌이 해결되면 push를 위해 master-branch에서 작업용 branch를 병합
- 만약 충돌해결이 오래 걸렸다면 pull → 작업용 branch 병합 후 충돌확인 과정 반복



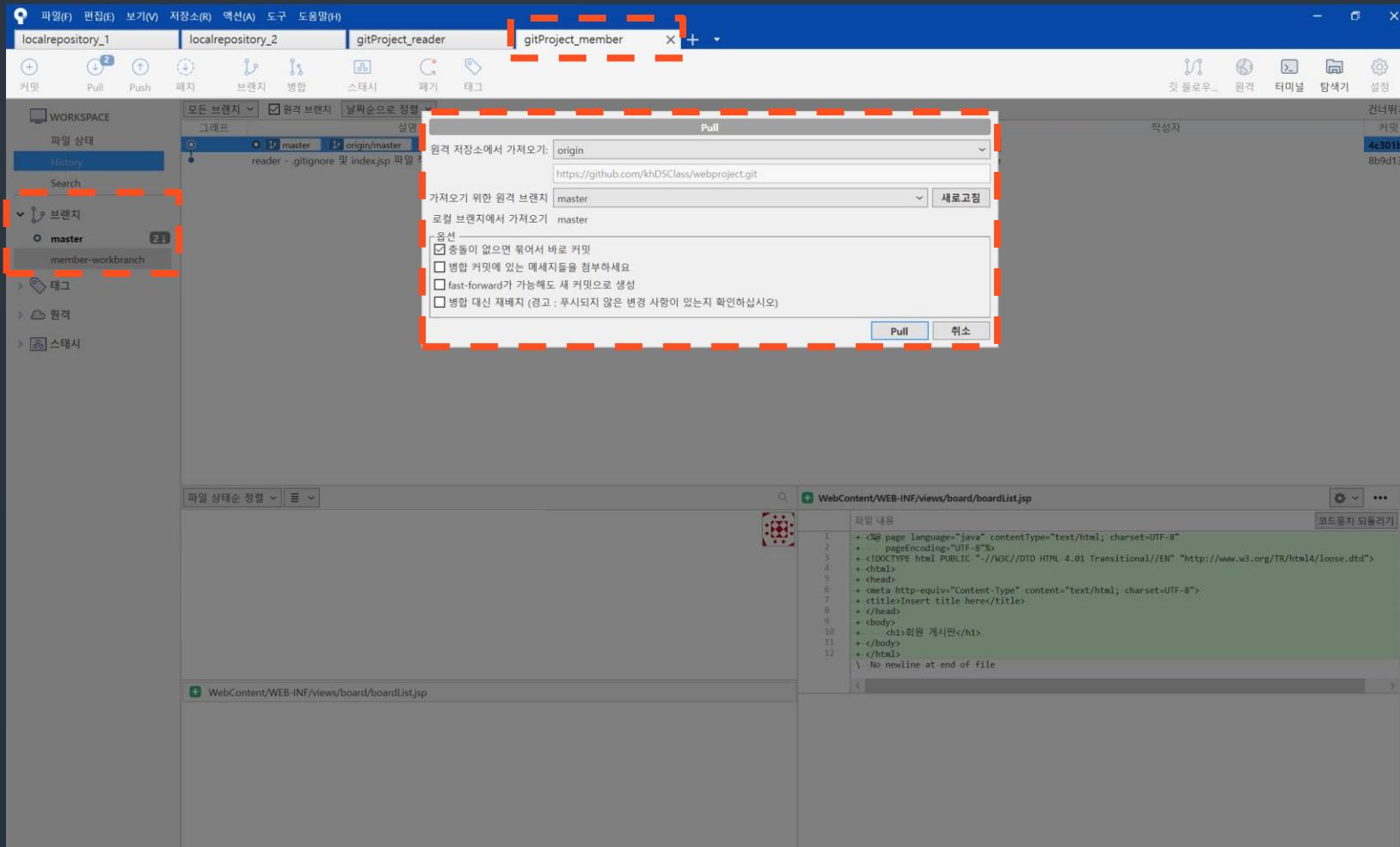
프로젝트 진행(프로젝트 관리자)

- master-branch에서 병합이 완료되면 push



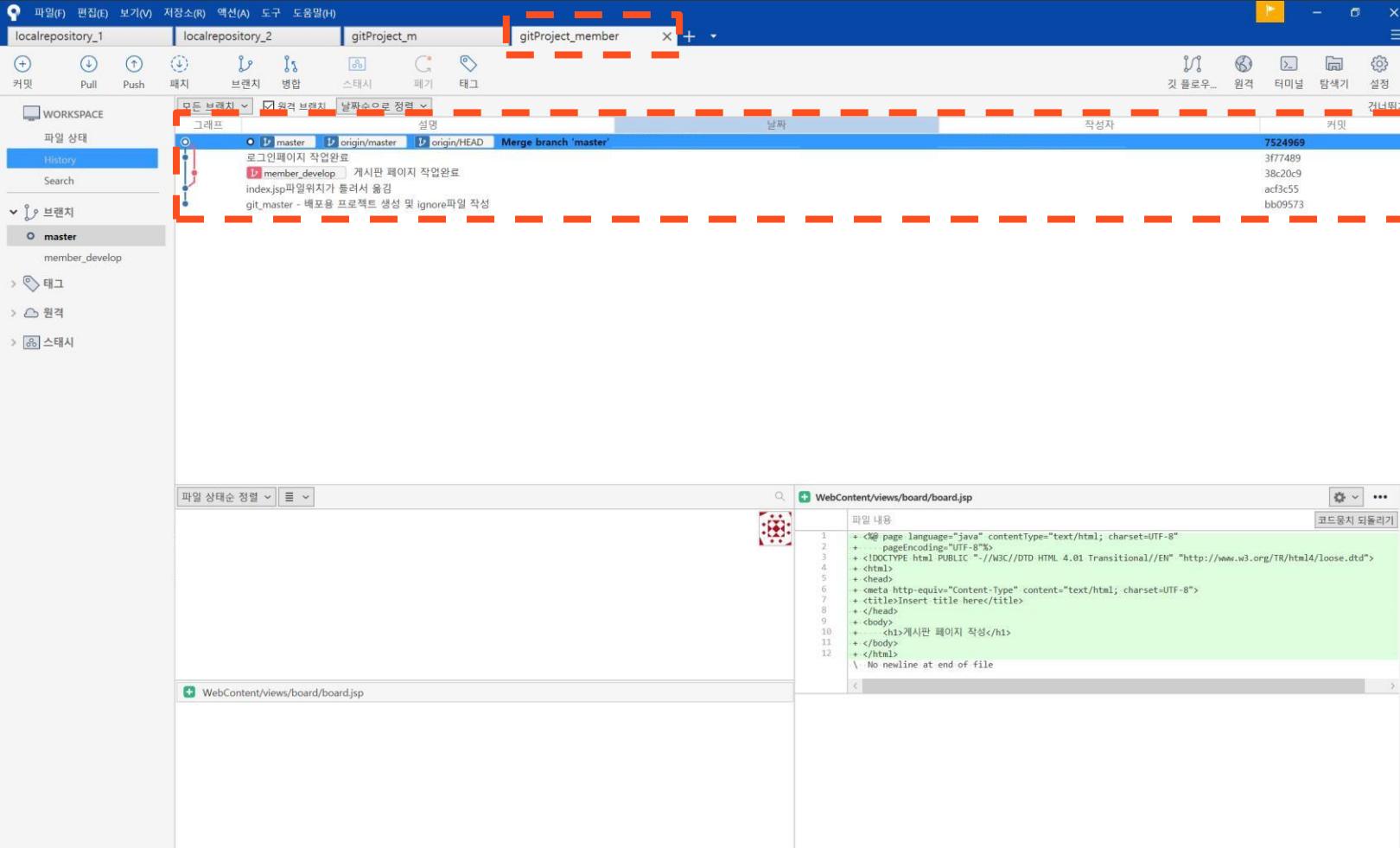
프로젝트 진행(프로젝트 팀원)

- master-branch에서 현재 원격저장소의 최신버전을 pull



프로젝트 진행(프로젝트 팀원)

- 다른 팀원이 작업한 내역이 그대로 적용



```
<%@ page language="java" contentType="text/html; charset=UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<h1>게시판 페이지 작성</h1>
</body>
</html>
```

프로젝트 진행(프로젝트 관리자)

- 다른 팀원이 작업한 내역이 그대로 적용

The screenshot shows the Eclipse IDE interface with a Java web project named "git-test-1". The Project Explorer view on the left displays the project structure:

- gitProject_m
- src
 - project.git
 - GitProjectClass.java
 - JRE System Library [jre1.8.0_144]
 - Apache Tomcat v8.5 [Apache Tomcat v8.5]
 - build
- WebContent
 - META-INF
 - views
 - board
 - board.jsp
 - member
 - login.jsp
 - WEB-INF
 - lib
 - web.xml
 - index.jsp
- Servers

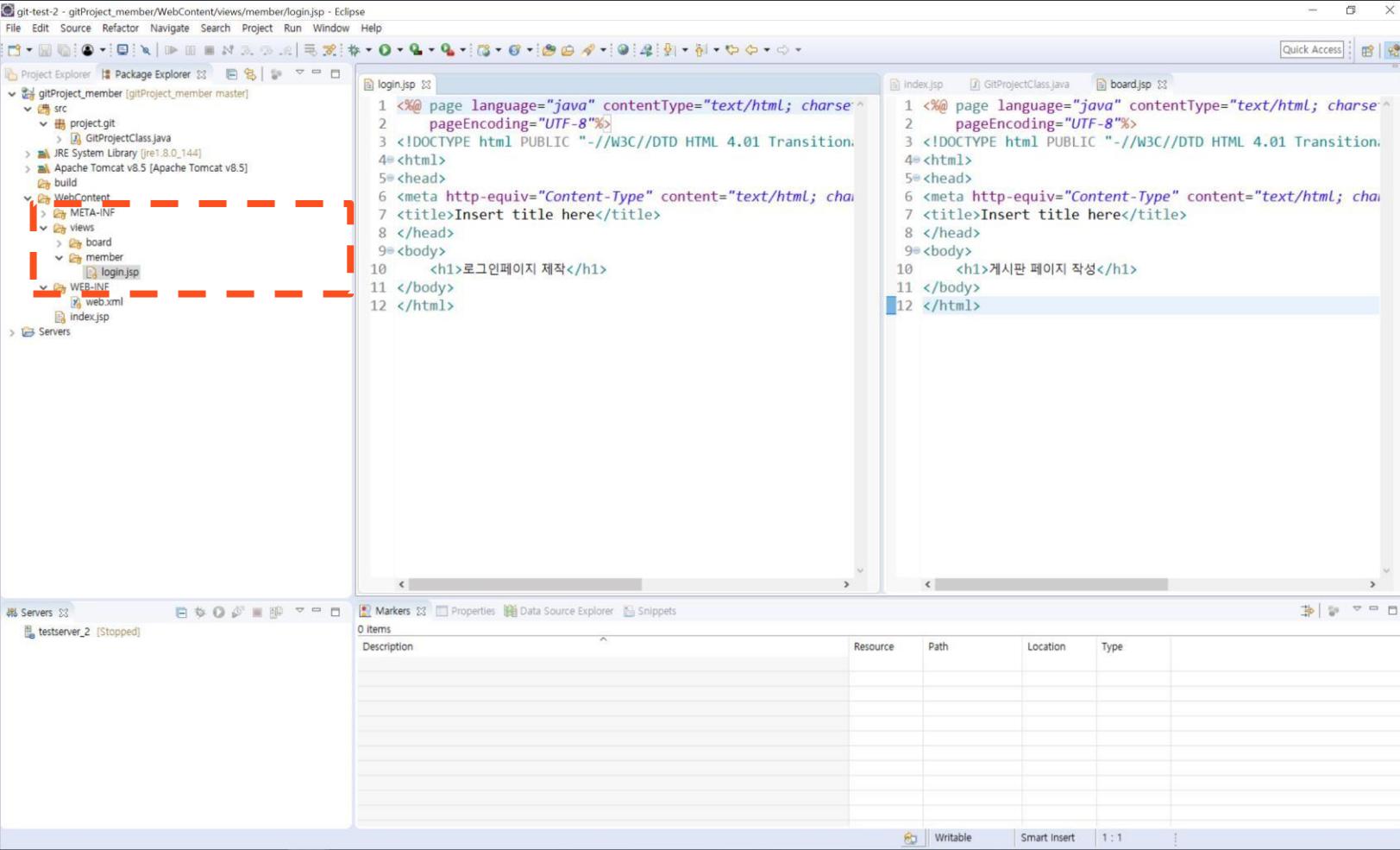
The central area contains two code editors:

- board.jsp:** Contains JSP code with a placeholder for a title.
- login.jsp:** Contains JSP code with a placeholder for a login page title.

The bottom of the screen shows the Eclipse interface with tabs like Servers, Properties, Data Source Explorer, and Snippets, and a status bar indicating the file is Writable and has a Smart Insert mode.

프로젝트 진행(프로젝트 팀원)

- 다른 팀원이 작업한 내역이 그대로 적용



The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure for "git-test-2 - gitProject_member". It includes a "src" folder containing "project.git" and "GitProjectClass.java", and a "WebContent" folder containing "views" (with "board" and "member" subfolders), "WEB-INF", and "index.jsp". A red dashed box highlights the "WebContent" folder.
- Editor Area:** Contains three tabs: "login.jsp", "index.jsp", and "board.jsp". The "login.jsp" tab is active, displaying the following code:

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8" %>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
3 <html>
4 <head>
5 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
6 <title>Insert title here</title>
7 </head>
8 <body>
9 <h1>로그인페이지 제작</h1>
10 </body>
11 </html>
```
- Servers View:** Shows a single server entry: "testserver_2 [Stopped]".
- Bottom Status Bar:** Displays "Writable", "Smart Insert", and "1 : 1".

Thank You