

스페셜 세션 - Git 협업

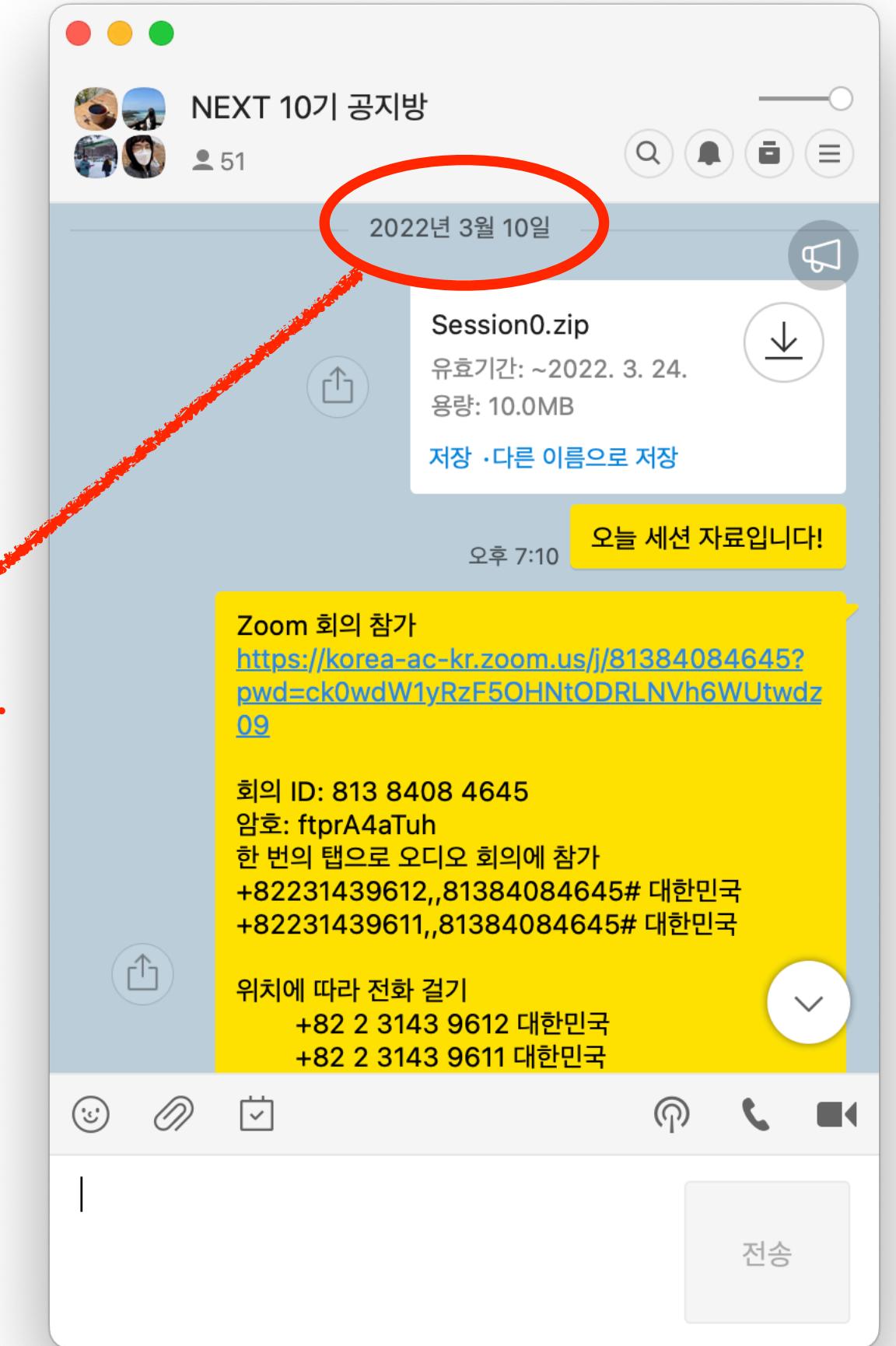
NEXT X LIKELION 전혜린

2022.05.14

아기다리 고기다리던 Git 협업 세션!!

여러분 드디어 또 Git 세션입니다

첫세션이 벌써 두 달 전…



첫 세션 때는 'Git for individuals (나를 위한 버전 관리)'를 배워보았다면,
이번에는 '**Git for teams** (팀을 위한 버전 관리)'를 배워보겠습니다.

Index

Detail Curriculum



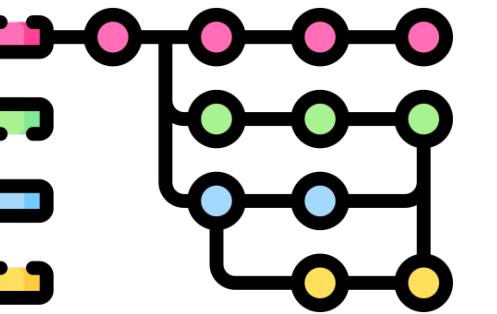
0. CLI & Git 명령어 복습



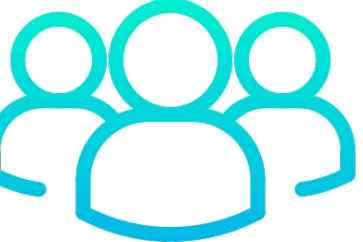
1. 협업할 때 Git 을 쓰는 이유



2. Git 협업을 위한 VSCode Extension 설치



3. Git 심화



4. 팀별 실습

0. CLI (Command Line Interface) & Git 명령어 복습

CLI 총 정리

[경로 탐색] **pwd**: print (current) working directory

[경로 이동] **cd**: change directory

[파일 확인] **ls**: list segments

ls -a: list segments of all

ls -l: list directory with long listing

ls -al: list directory of all with long listing

[파일 생성] **touch**: create a file

mkdir: make a directory

[파일 제거] **rm**: remove a file

rm - rf: remove forcibly including directory and the files in it

0. CLI (Command Line Interface) & Git 명령어 복습

Git 명령어 총 정리

<git 초기 설정>

[전역 사용자명/이메일 구성하기] git config --global user.name \${user.name}
git config --global user.email \${user.email}
git config --list

[새로운 저장소 초기화] git init

[저장소 복제] git clone \${깃헙 주소}

[Local & Remote 저장소 연결] git remote add origin \${본인 레포 주소}
git remote -v

<git 파일 스테이징 & 푸시>

[git 스테이징 영역에 새로운 파일 추가] git add \${file_name}
git status

[git 스테이징 영역에 존재하는 파일 커밋하기] git commit -m "\${commit message}"
git log

[Github 레포에 코드 올리기] git push origin \${branch name}

0. CLI (Command Line Interface) & Git 명령어 복습

Git 명령어 총 정리

<git 초기 설정>

[전역 사용자명/이메일 구성하기] git config --global user.name \${user.name}
git config --global user.email \${user.email}
git config --list

[새로운 저장소 초기화] git init

[저장소 복제] git clone \${깃헙 주소}

[Local & Remote 저장소 연결] git remote add origin \${분인 레포주소}

git remote -v

<git 파일 스테이징 & 푸시>

[git 스테이징 영역에 새로운 파일 추가] git add \${file_name}
git status

[git 스테이징 영역에 존재하는 파일 커밋하기] git commit -m "\${commit message}"
git log

[Github 레포에 코드 올리기] git push origin \${branch name}

하.지.만

이것은 꼭히 일부에 불과했다?

1. 협업할 때 Git 을 쓰는 이유

Why Git for Teams?

Q) 협업할 때 Why using Git?

각자 작업한 내용을 **버전 별로 분산 관리하기** 위해!

나중에 commit 기록을 확인하며 **기능 별로 합칠 수 있답니다!**

코드 중복 방지에도 용이해요!

1. 협업할 때 Git 을 쓰는 이유

Why Git for Teams?

그려려면, commit을 다른 사람이 쉽게 알아볼 수 있도록 세세하게 & 잘 기록해두어야겠죠?

커밋 메세지 구조:

<타입>[적용 범위(선택 사항)]: <설명>

타입 종류:

Fix - 코드 베이스에서 버그를 패치하는 커밋 타입

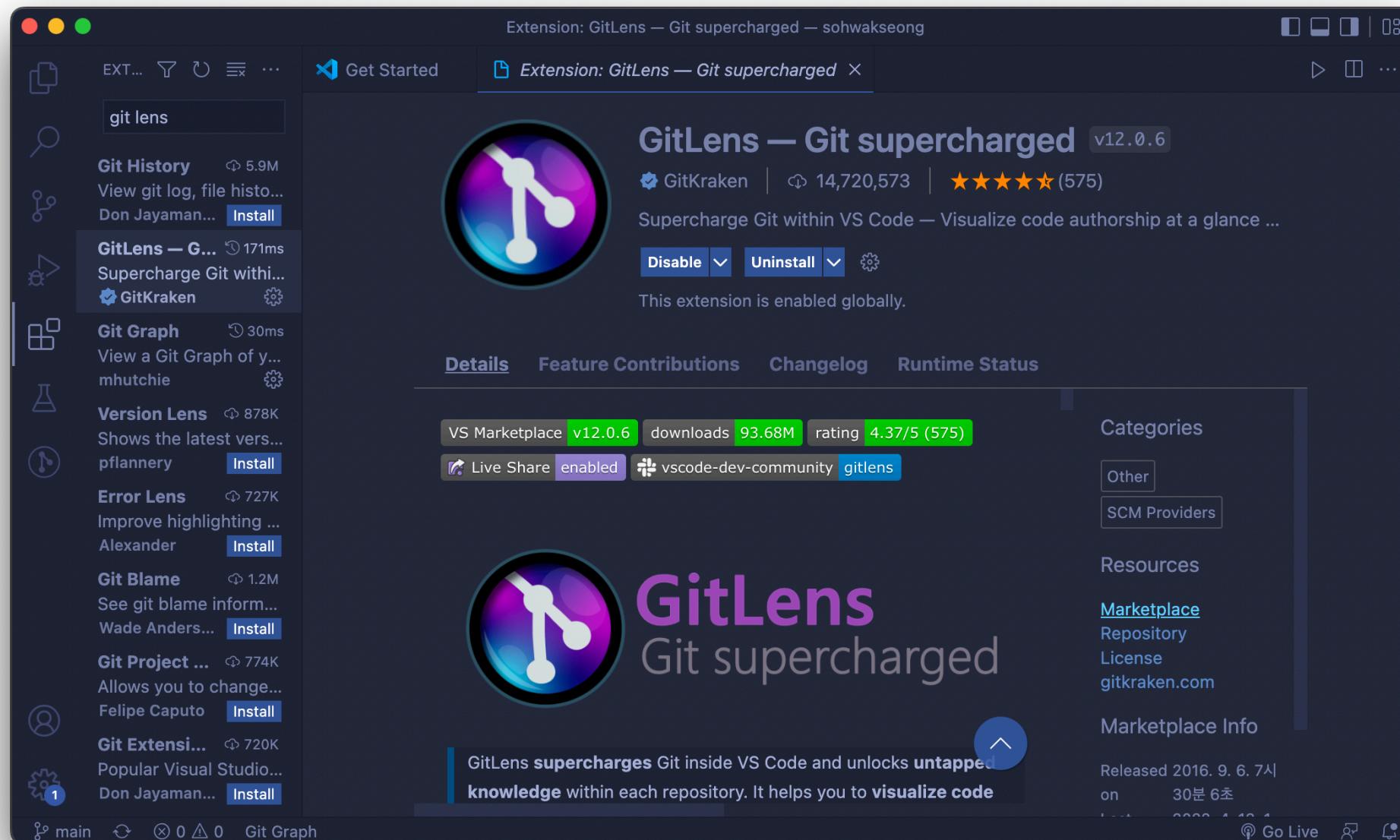
Feat - 코드베이스에서 새 기능이 추가되는 커밋 타입

- ↗ **Feat/signup style**
#16 by young924 was merged on 16 Jan
- ↗ **Feat/welcome login style**
#15 by young924 was merged on 16 Jan
- ↗ **Feat/nickname search**
#14 by young924 was merged on 15 Jan
- ↗ **Feat/daily achv 기능 추가**
#7 by Parkhanyoung was merged on 14 Jan
- ↗ **[feat]: star 앱 및 모델 생성**
#4 by Parkhanyoung was merged on 13 Jan
- ↗ **[#2][feat]: 초기세팅**
#3 by Parkhanyoung was merged on 12 Jan

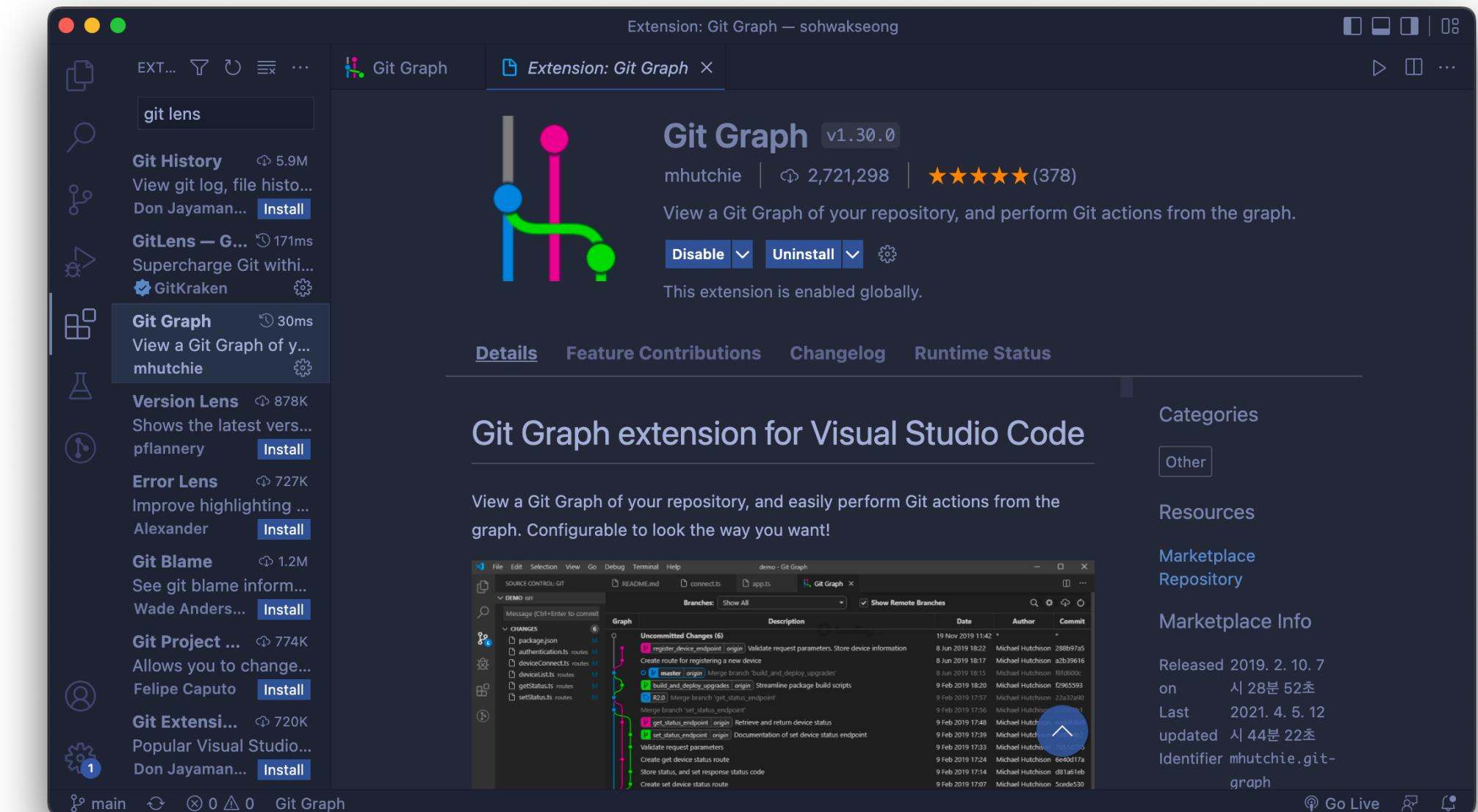
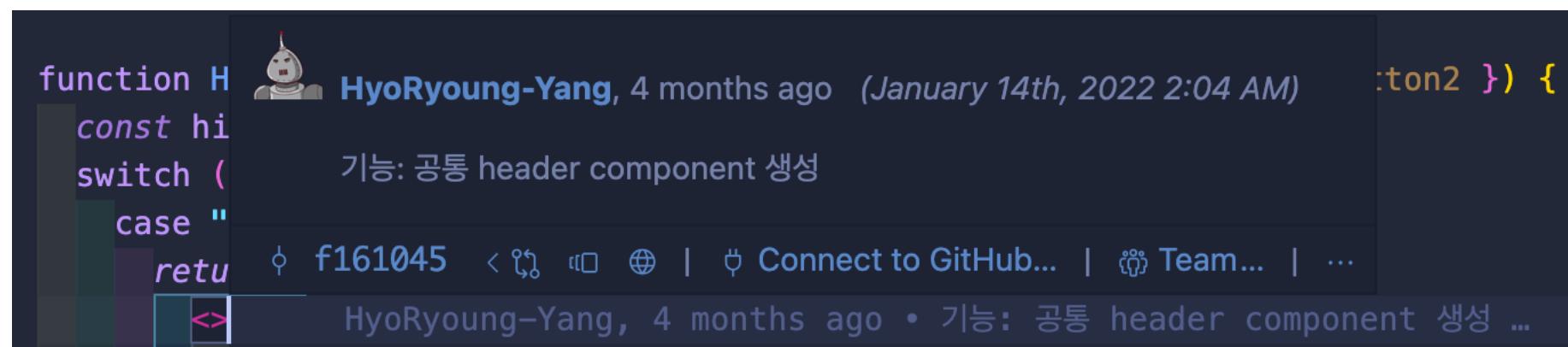
2. Git 협업을 위한 VSCode Extension 설치

Git Lens & Git Graph

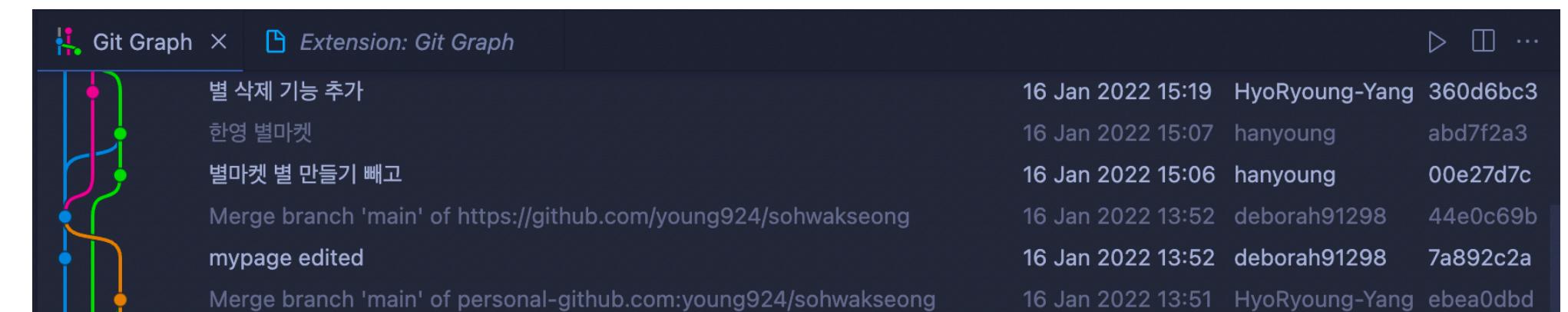
Git Lens & Git Graph 설치하기



Ex)



Ex)



3. Git 심화

3-1. Fork

fork란 무엇인가?

다른 사람의 Github repository에서 내가 어떤 부분을 수정 or 추가하고 싶을 때,
해당 repository를 내 Github repository로 그대로 **복제**하는 기능이다.
fork한 저장소는 원본(다른 사람의 github repository)과 연결되어 있다.



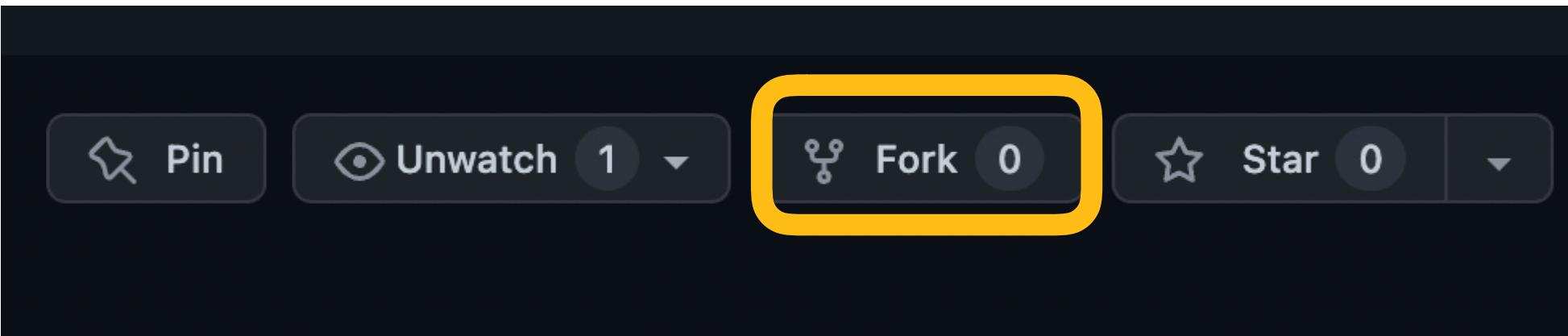
우선 이번 실습을 위한 Github repository를 **fork**하고,
그 주소를 **git clone** 으로 복제해와서 실습 진행!

3. Git 심화

3-1. Fork

아래 주소에 접속하여 fork 해주세요!

https://github.com/LionHyerin/NEXT_GitCollaboration



새로 fork 된 Github repository 의 주소는 아래가 되겠죠?

[https://github.com/\\${여러분의 깃헙 아이디}/NEXT_GitCollaboration](https://github.com/${여러분의 깃헙 아이디}/NEXT_GitCollaboration)

3. Git 심화

3-1. Fork

* 실습 Time *

* 주의: 이번 세션은 next_likelion2022 폴더가 아닌 .git 파일이 없는 새로운 폴더에서 진행해주세요!

mkdir SpecialSession_GitCollab => 상위 폴더에 .git 파일 없어야 됨!

cd SpecialSession_GitCollab

git clone \${포크한 주소}

ex) git clone [https://github.com/\\${여러분의 깃헙 아이디}/NEXT_GitCollaboration](https://github.com/${여러분의 깃헙 아이디}/NEXT_GitCollaboration)

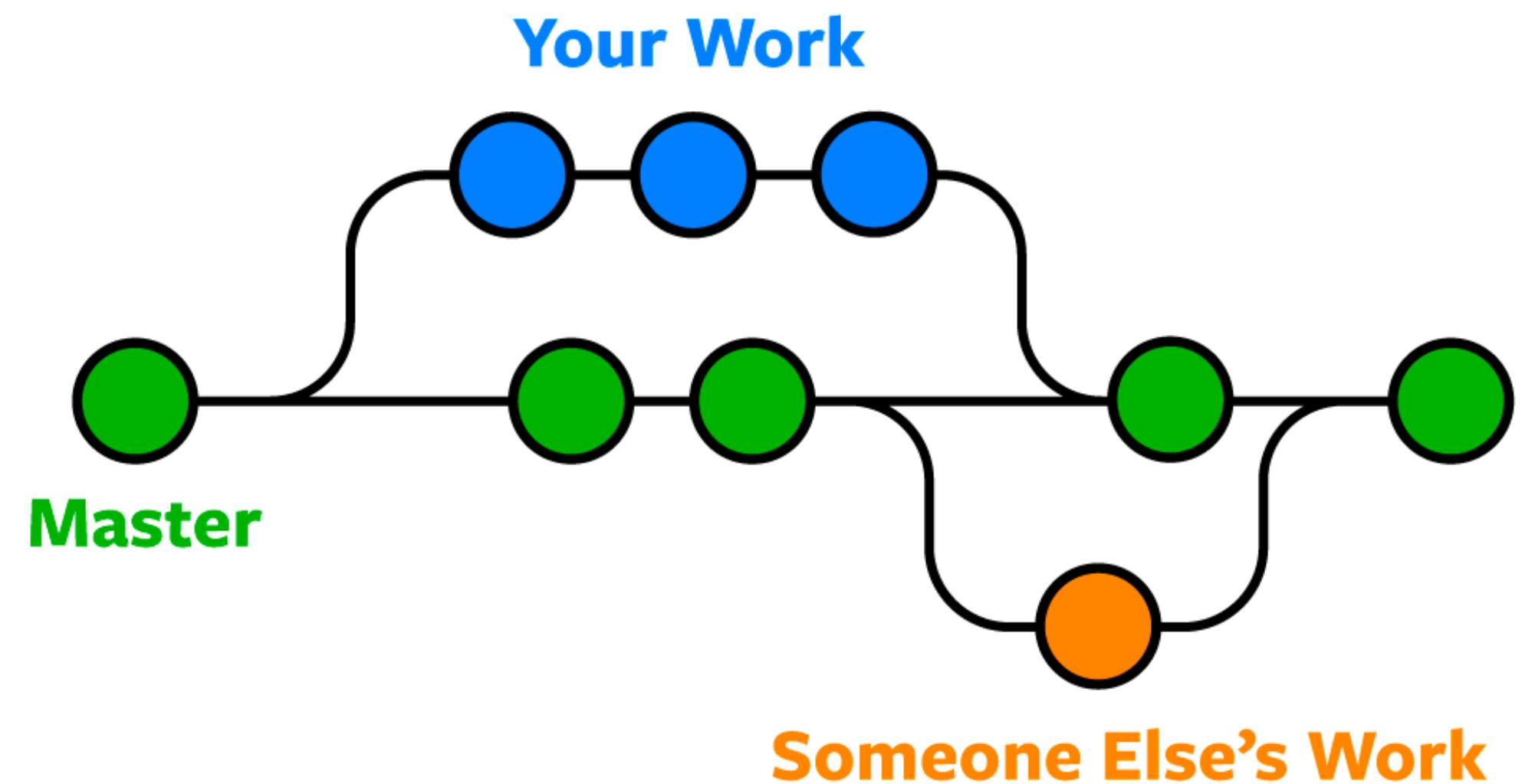
3. Git 심화

3-2. Branch

Branch란 무엇이죠?

브랜치(branch) = 독립적인 작업을 위해 줄기를 파는 것

필요에 의해 만들어지는 각각의 브랜치는 다른 브랜치의 영향을 받지 않기 때문에, 여러 작업을 동시에 진행할 수 있습니다.



3. Git 심화

3-2. Branch

Branch 생성: `git branch "${브랜치 이름}"`

Branch 이동: `git checkout "${브랜치 이름}"`

Branch 생성 및 이동: `git checkout -b "${브랜치 이름}"`

Branch 삭제: `git branch -d "${브랜치 이름}"`

현재 Branch 확인 명령어: `git branch ('Q'로 나가기)`

모든 Branch 확인 명령어: `git branch -r ('Q'로 나가기)`

처음 Git 설치 ➔ 'master/main' 브랜치로 되어 있다.

여기서 '체크아웃(checkout)' 명령어를 실행하면, 원하는 브랜치로 전환하여 독립적인 작업이 가능하다!

체크아웃 실행 시 ➔ 전환되기 전의 브랜치 안에 있는 **마지막 커밋 내용**이 작업 트리에 펼쳐진다.

브랜치가 전환된 후에 실행한 커밋은 전환한 브랜치에 추가된다! (전환되기 전의 커밋은 x)

<브랜치 이름 컨벤션>

새로운 기능을 추가할 브랜치 ➔ `feat/<추가할 기능 이름>`

3. Git 심화

3-2. Branch

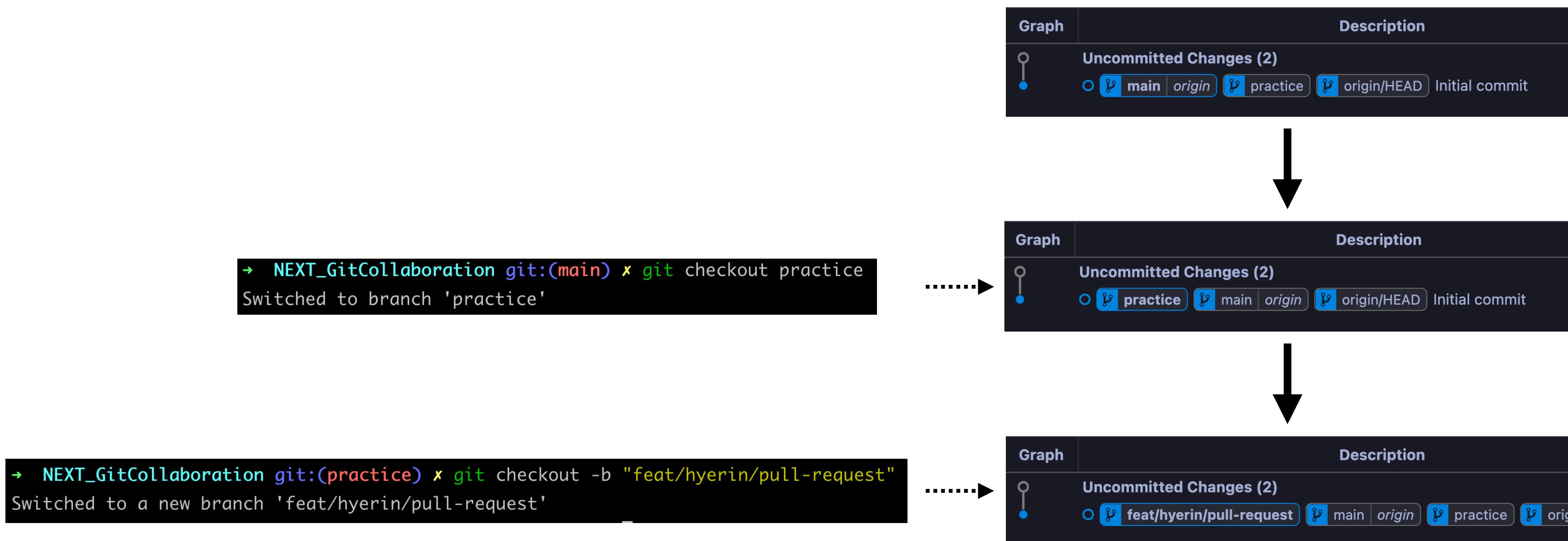
* 실습 Time *

```
cd NEXT_GitCollaboration  
(code . 으로 vscode 열어주기)  
git checkout "practice"  
git checkout -b "feat/${본인 닉네임}/pull-request"
```

3. Git 심화

3-2. Branch

Checkout 후, 변경된 branch “git graph에서 확인 가능!”



3. Git 심화

3-3. Reset

Reset

시간을 **아예** 과거의 특정 commit 으로 되돌린다.

Revert

현재에 있으면서 과거의 특정 commit 들만 삭제한다.

공통점: 과거로 되돌아간다.

차이점: 과거로 되돌리겠다는 내용도 commit 이력에 남는가?

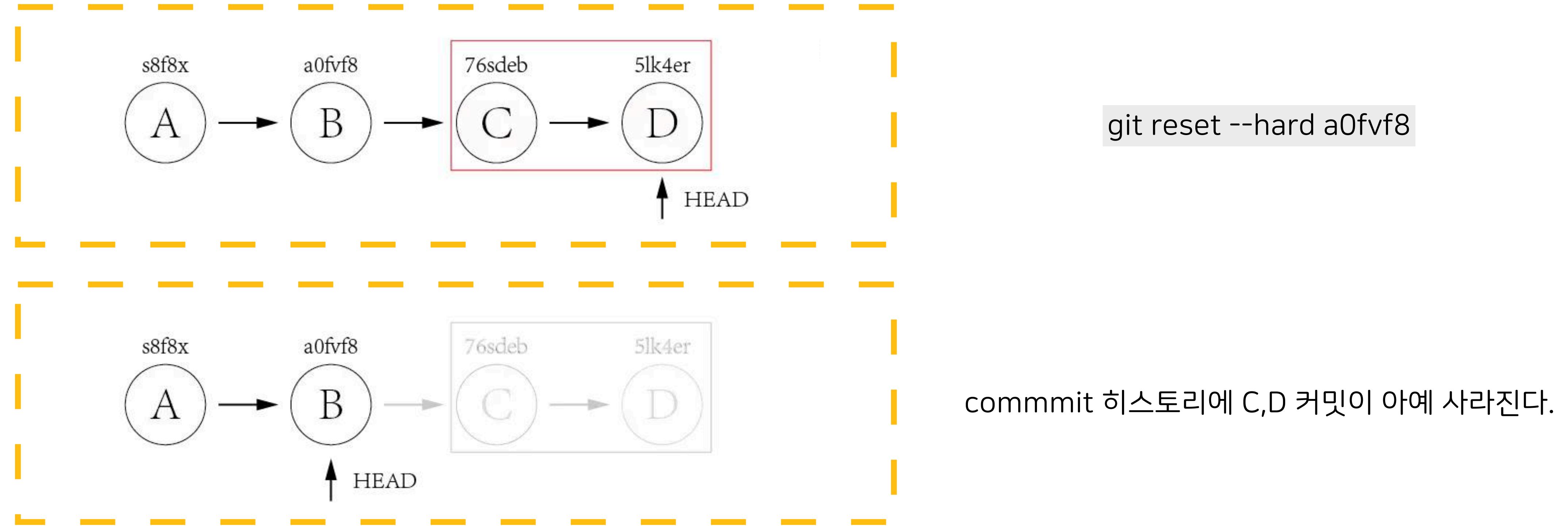
=> Reset 은 no

=> Revert 는 yes

3. Git 심화

3-3. Reset & Revert

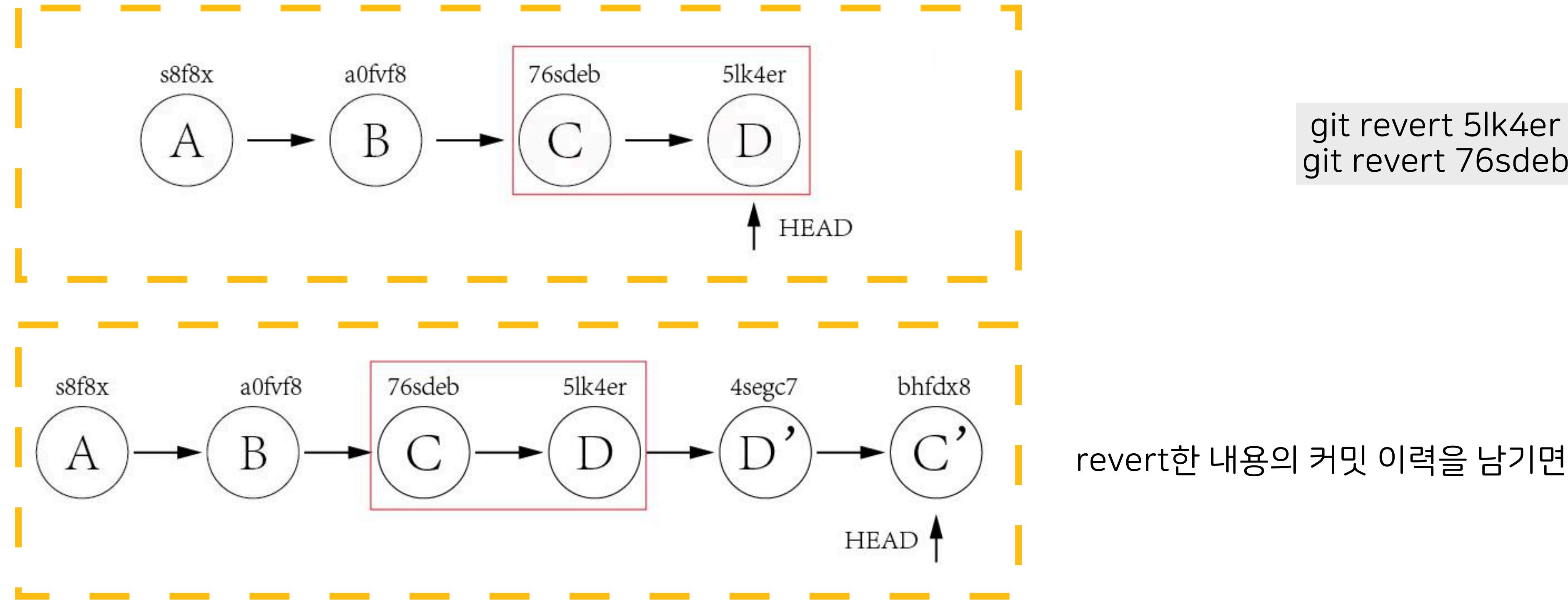
Reset



3. Git 심화

3-3. Reset & Revert

Revert



3. Git 심화

3-3. Reset & Revert

Reset or Revert?

협업프로젝트에서 다른 팀원과 같은 branch를 공유하는 상황일때,
혹은 이전 커밋 기록이 필요할 수 있을 때, 가급적 revert를 사용한다.
그 외의 독립적인 작업 or 개인프로젝트에서는 reset을 주로 사용한다.

3. Git 심화

3-3. Reset & Revert

Reset 방법

```
git reset --soft ${commit ID}  
git reset --mixed ${commit ID} = git reset ${commit ID}  
git reset --hard ${commit ID}  
git reset HEAD~10  
git reset HEAD^
```

- **soft** : commit된 파일들을 staging area로 돌려놓는다. (commit 하기 전 상태로)
- **mixed**(default) : commit된 파일들을 working directory로 돌려놓는다. (add 하기 전 상태로)
- **hard** : commit된 파일들 중 tracked 파일들을 working directory에서 삭제한다.
(Untracked 파일은 여전히 Untracked로 남는다.)
- **HEAD~취소할커밋수** : 현재로부터 원하는 만큼의 커밋이 취소된다.
- **HEAD^** : 가장 최근의 커밋이 취소된다. (기본옵션 mixed)

3. Git 심화

3-3. Reset & Revert

Revert 방법

```
git revert ${commit ID}
```

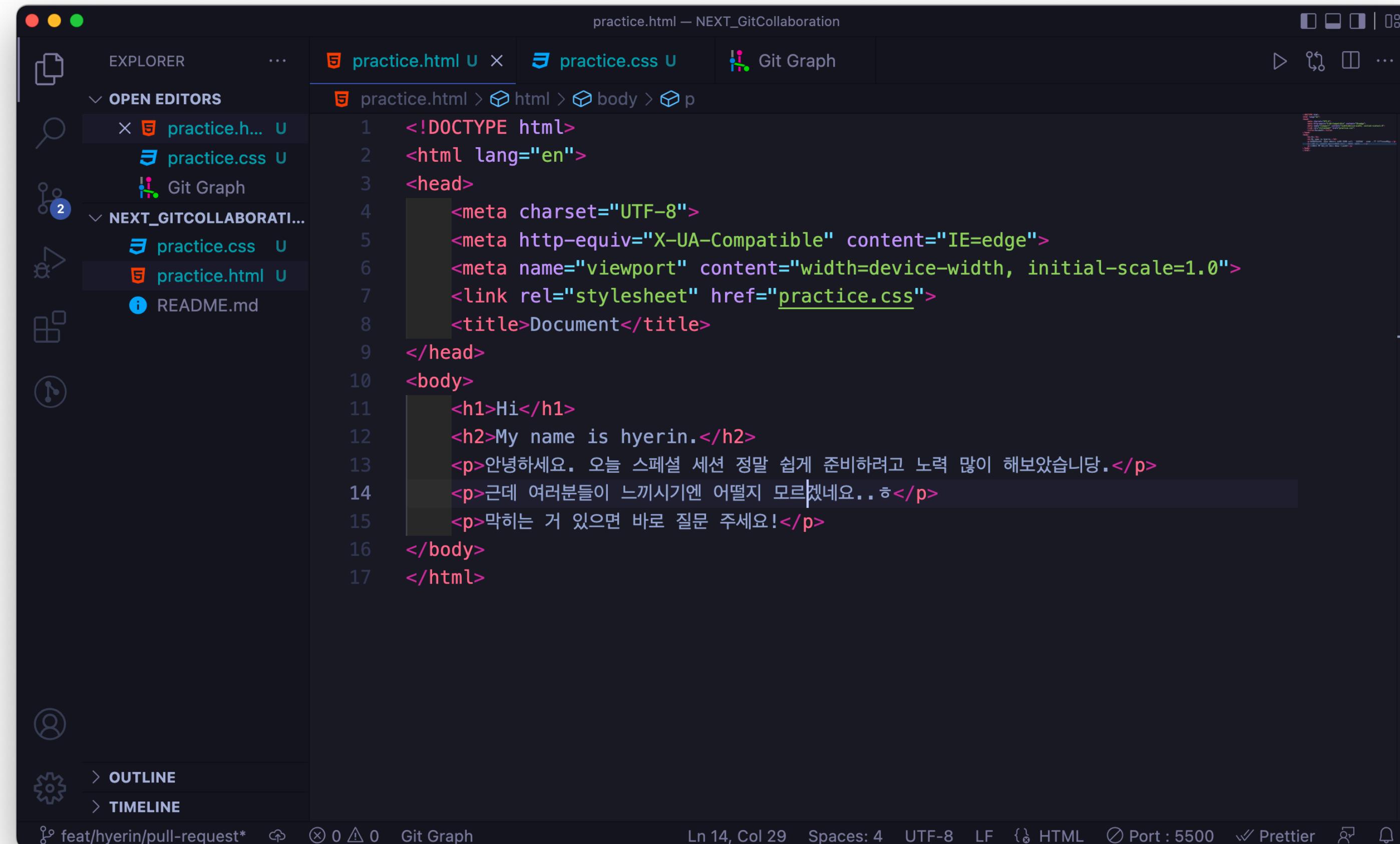
=> 돌아가고자 하는 commit ID를 적어준다.

3. Git 심화

3-4. 레포지토리 브랜치에서 pull-request 해보기

* 실습 Time *

practice.html 채워주기



A screenshot of a dark-themed code editor (VS Code) showing the file `practice.html`. The editor has a tab bar with `practice.html`, `practice.css`, and `Git Graph`. The `practice.html` tab is active, displaying the following HTML code:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link rel="stylesheet" href="practice.css">
<title>Document</title>
</head>
<body>
<h1>Hi</h1>
<h2>My name is hyerin.</h2>
<p>안녕하세요. 오늘 스페셜 세션 정말 쉽게 준비하려고 노력 많이 해보았습니다.</p>
<p>근데 여러분들이 느끼시기엔 어떨지 모르겠네요.. 흠</p>
<p>막히는 거 있으면 바로 질문 주세요!</p>
</body>
</html>
```

The code editor interface includes an Explorer sidebar on the left with files like `practice.h...`, `practice.css`, and `Git Graph`. The bottom status bar shows the file path `feat/hyerin/pull-request*`, line and column numbers `Ln 14, Col 29`, and other settings.

3. Git 심화

3-4. 레포지토리 브랜치에서 pull-request 해보기

* 실습 Time *

```
git add .  
git commit -m "feat: html 자기소개 추가"
```

(3분 동안 css 간단하게 꾸며보기)

```
git add .  
git commit -m "feat: css 꾸미기 추가"
```

=> 기능 별로 commit 나눠서 기록하는 습관 들이기!
(이거 습관 안 들여놓으면 나중에 힘들어요.. 제 얘기..)

3. Git 심화

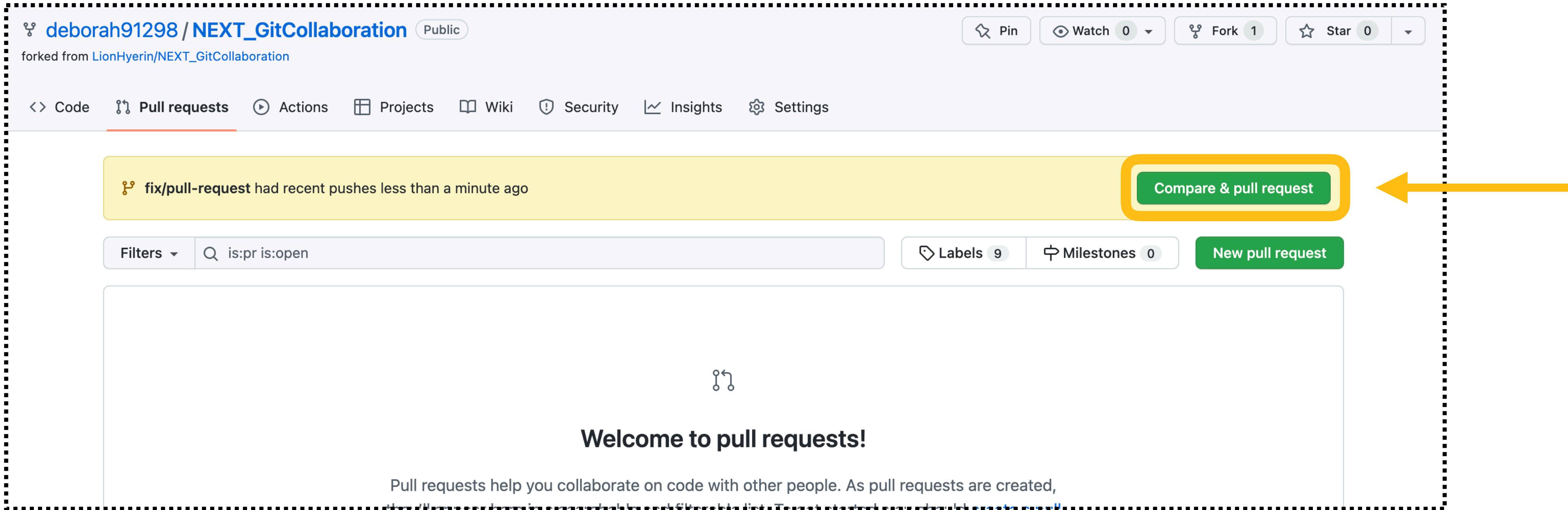
3-4. 레포지토리 브랜치에서 pull-request 해보기

* 실습 Time *

```
git checkout -b "fix/pull-request"  
git push origin "fix/pull-request"
```

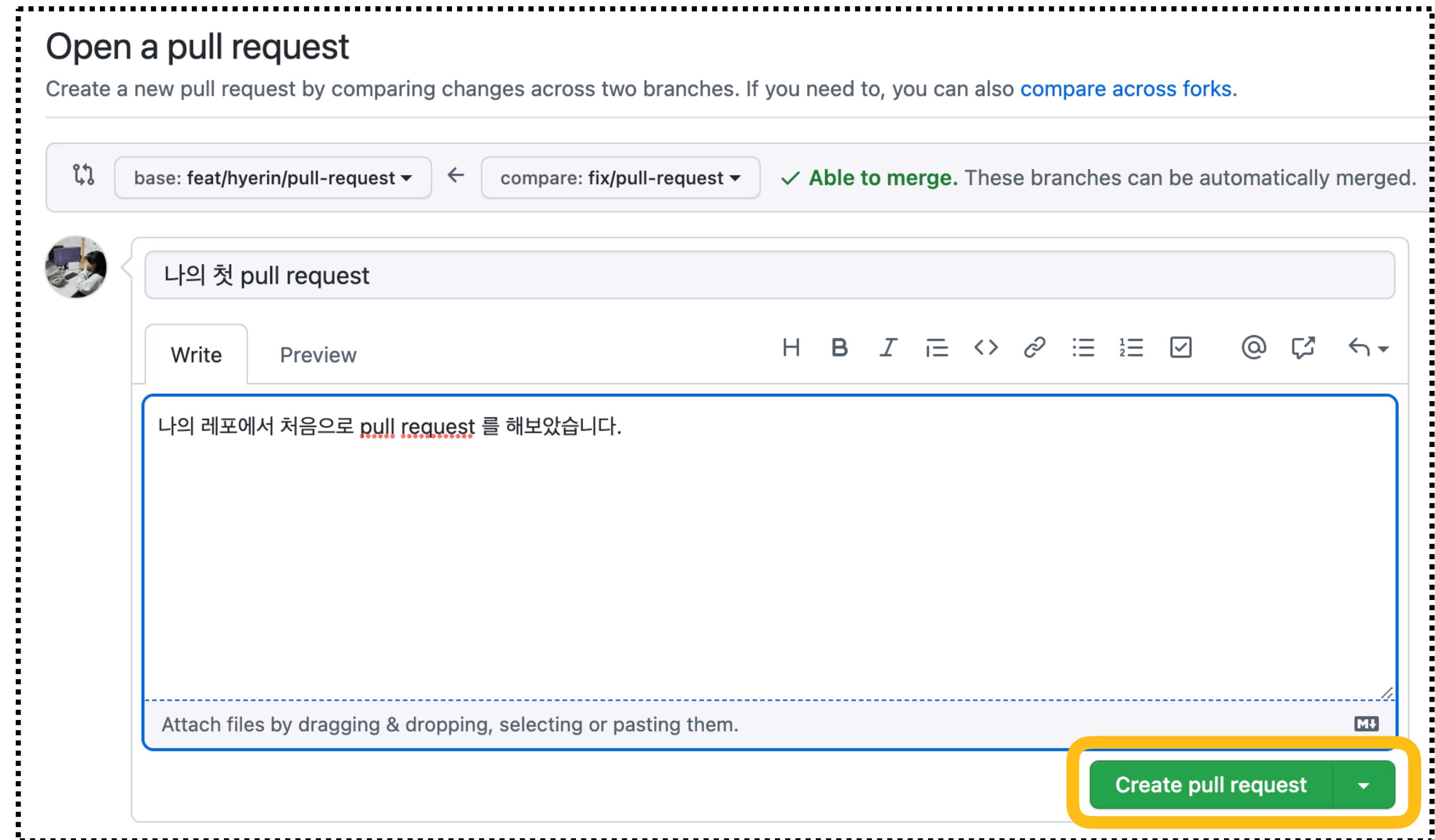
3. Git 심화

3-4. 레포지토리 브랜치에서 pull-request 해보기



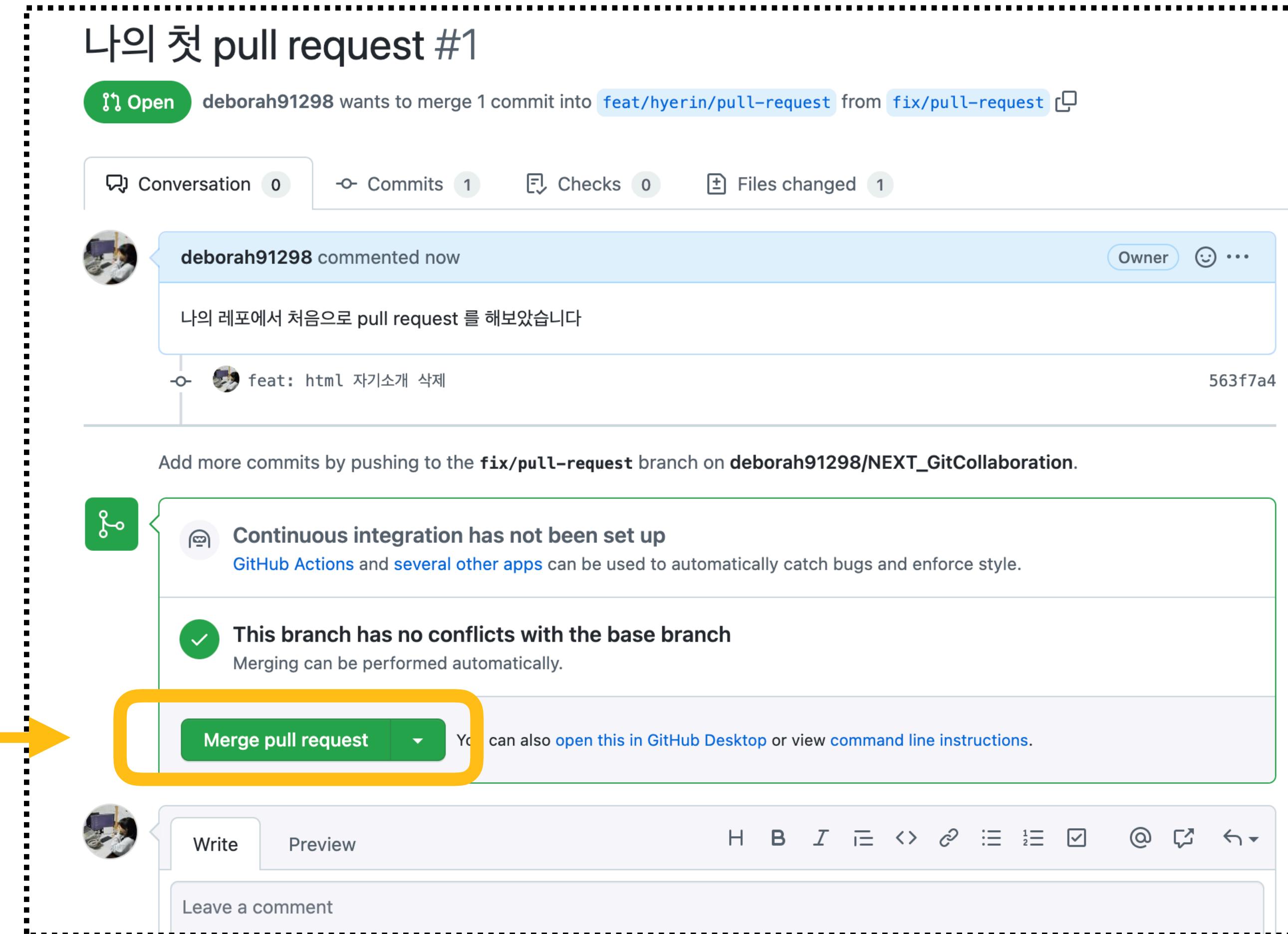
3. Git 심화

3-4. 레포지토리 브랜치에서 pull-request 해보기



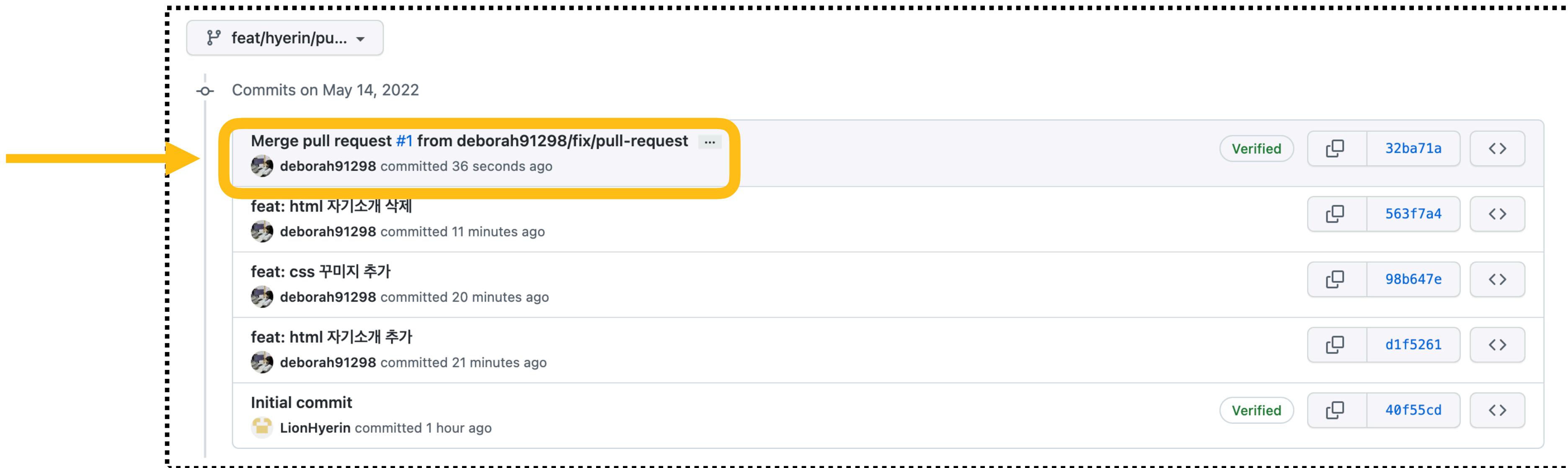
3. Git 심화

3-4. 레포지토리 브랜치에서 pull-request 해보기



3. Git 심화

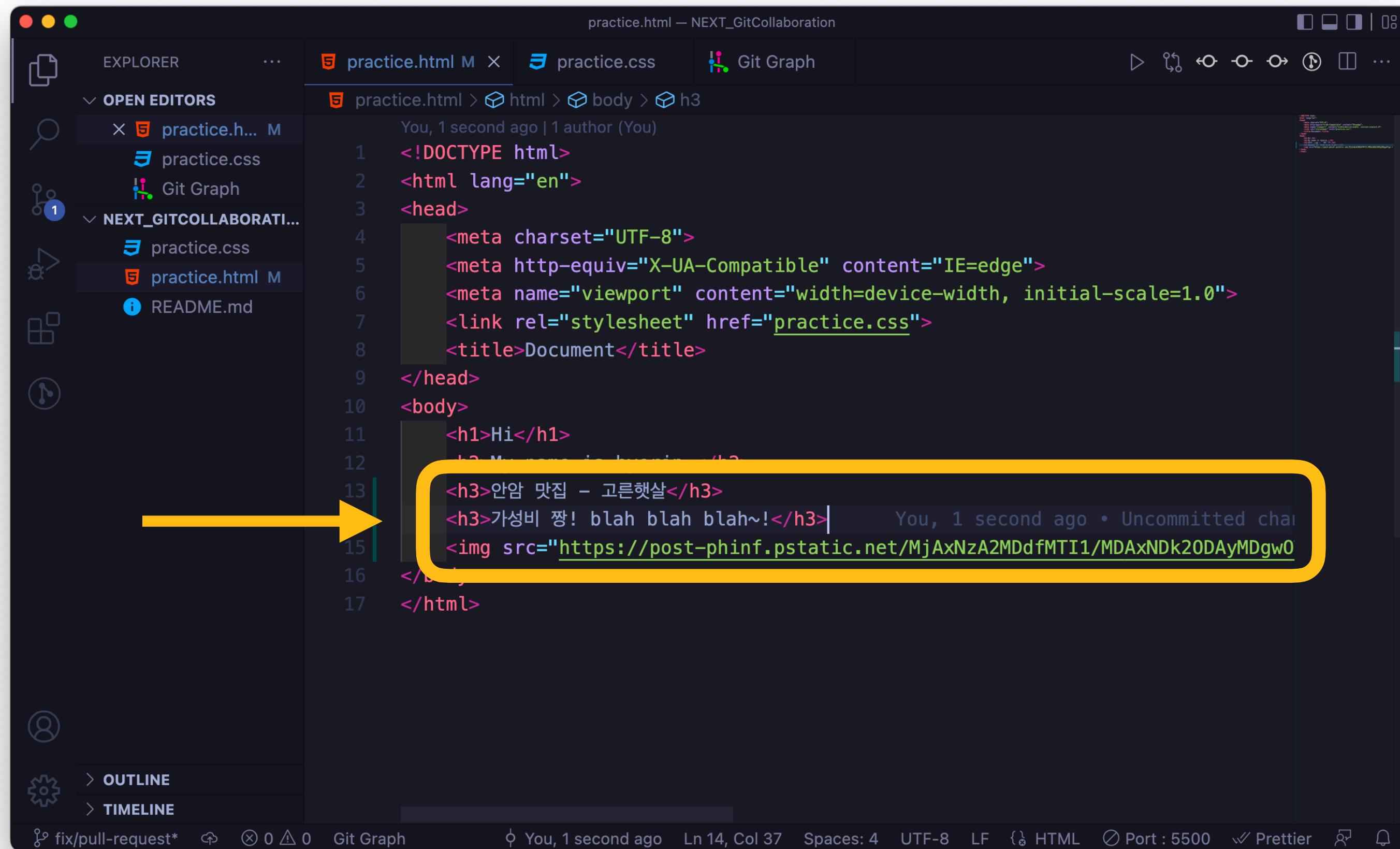
3-4. 레포지토리 브랜치에서 pull-request 해보기



3. Git 심화

3-5. 다른 레포지토리에서 pull-request 해보기

이런식으로 안암 맛집 추천 글 작성하고 “fix/pull-request” 브랜치에서 commit & push까지 완료해주세요.



A screenshot of the Visual Studio Code (VS Code) interface. The title bar shows "practice.html — NEXT_GitCollaboration". The Explorer sidebar on the left lists files: "practice.html M", "practice.css", and "Git Graph" under the "NEXT_GITCOLLABORATION" folder. The main editor area displays the "practice.html" file content:

```
You, 1 second ago | 1 author (You)
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <link rel="stylesheet" href="practice.css">
8   <title>Document</title>
9   </head>
10  <body>
11    <h1>Hi</h1>
12    <h2>안암 맛집 추천</h2>
13    <h3>안암 맛집 – 고른햇살</h3>
14    <h3>가성비 짱! blah blah blah~!</h3>
15    
16  </body>
17 </html>
```

A yellow arrow points from the left towards the bottom right of the code editor, where a yellow box highlights the last three lines of the code. The status bar at the bottom indicates "fix/pull-request*" and "Port : 5500".

3. Git 심화

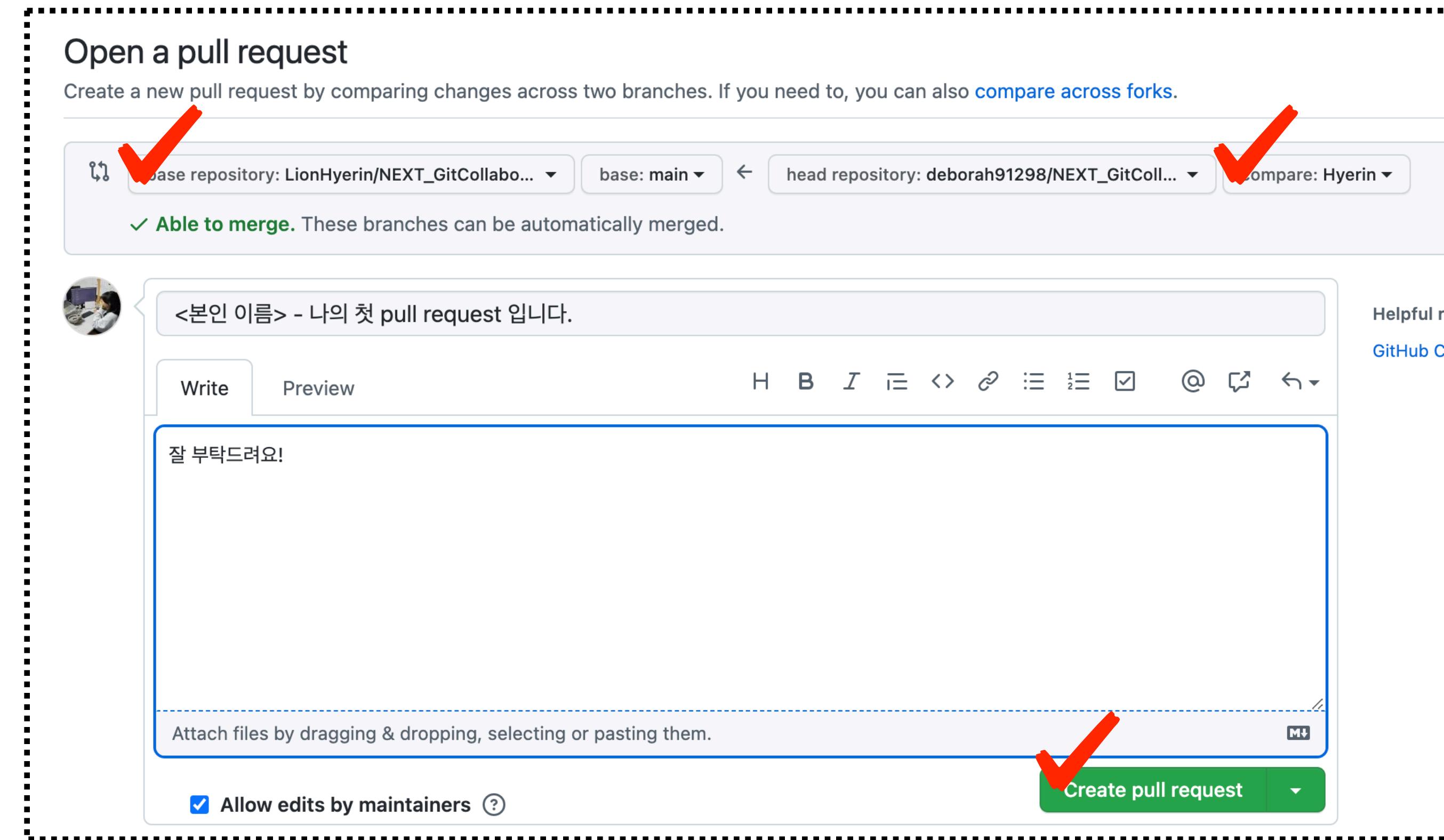
3-5. 다른 레포지토리에서 pull-request 해보기

```
git checkout -b "${본인 영어 이름}" ➔ (예. Hyerin)  
git pull origin 'fix/pull-request'  
git push origin "${본인 영어 이름}"
```

The screenshot shows a GitHub interface for a repository. At the top, there is a yellow notification bar with the message "Hyerin had recent pushes less than a minute ago". To the right of the bar is a green button labeled "Compare & pull request", which is highlighted with a yellow arrow pointing from the text above. Below the notification bar is a search bar with the query "is:pr is:open". To the right of the search bar are buttons for "Labels 9", "Milestones 0", and "New pull request". Further down are filters for "0 Open" and "1 Closed", and dropdown menus for "Author", "Label", "Projects", "Milestones", "Reviews", "Assignee", and "Sort". The main content area displays the message "There aren't any open pull requests." and a link to "all of GitHub" or "advanced search".

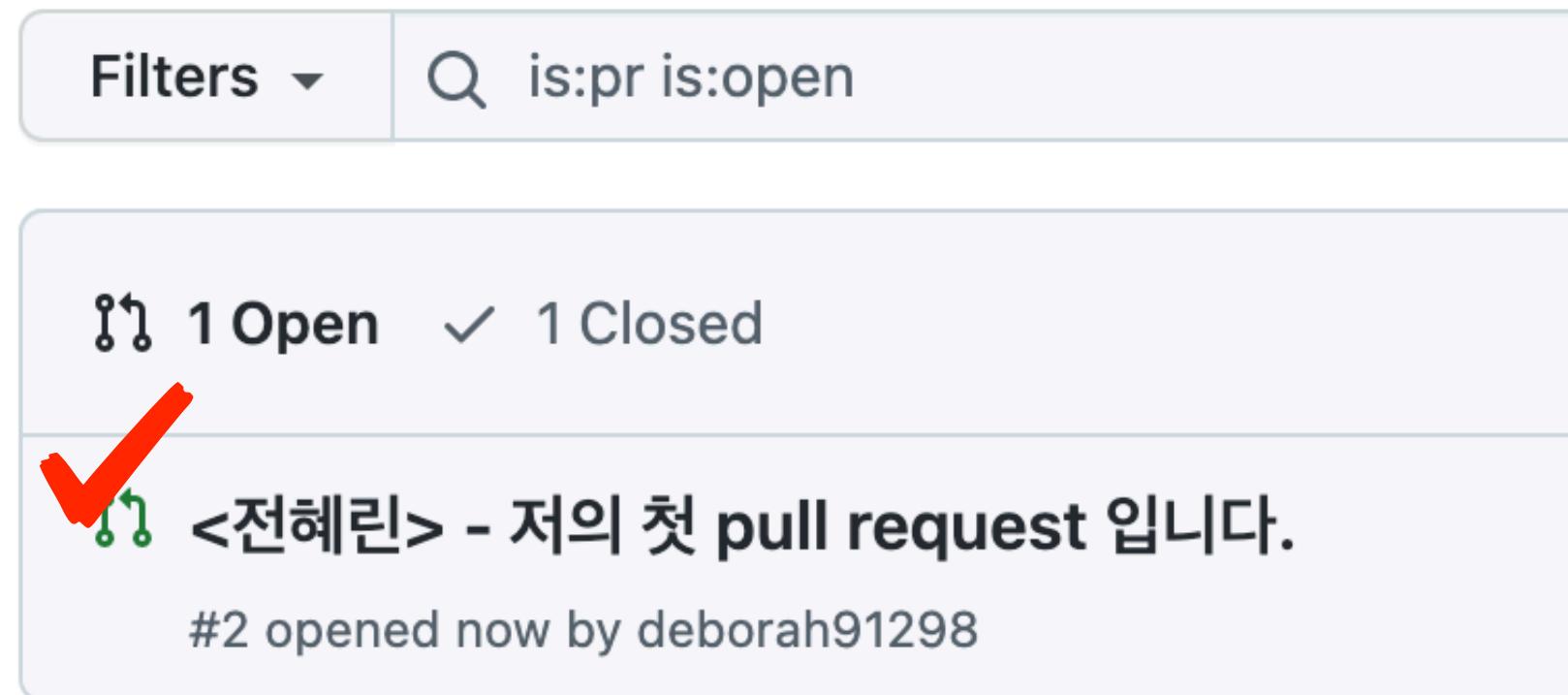
3. Git 심화

3-5. 다른 레포지토리에서 pull-request 해보기



3. Git 심화

3-5. 다른 레포지토리에서 pull-request 해보기

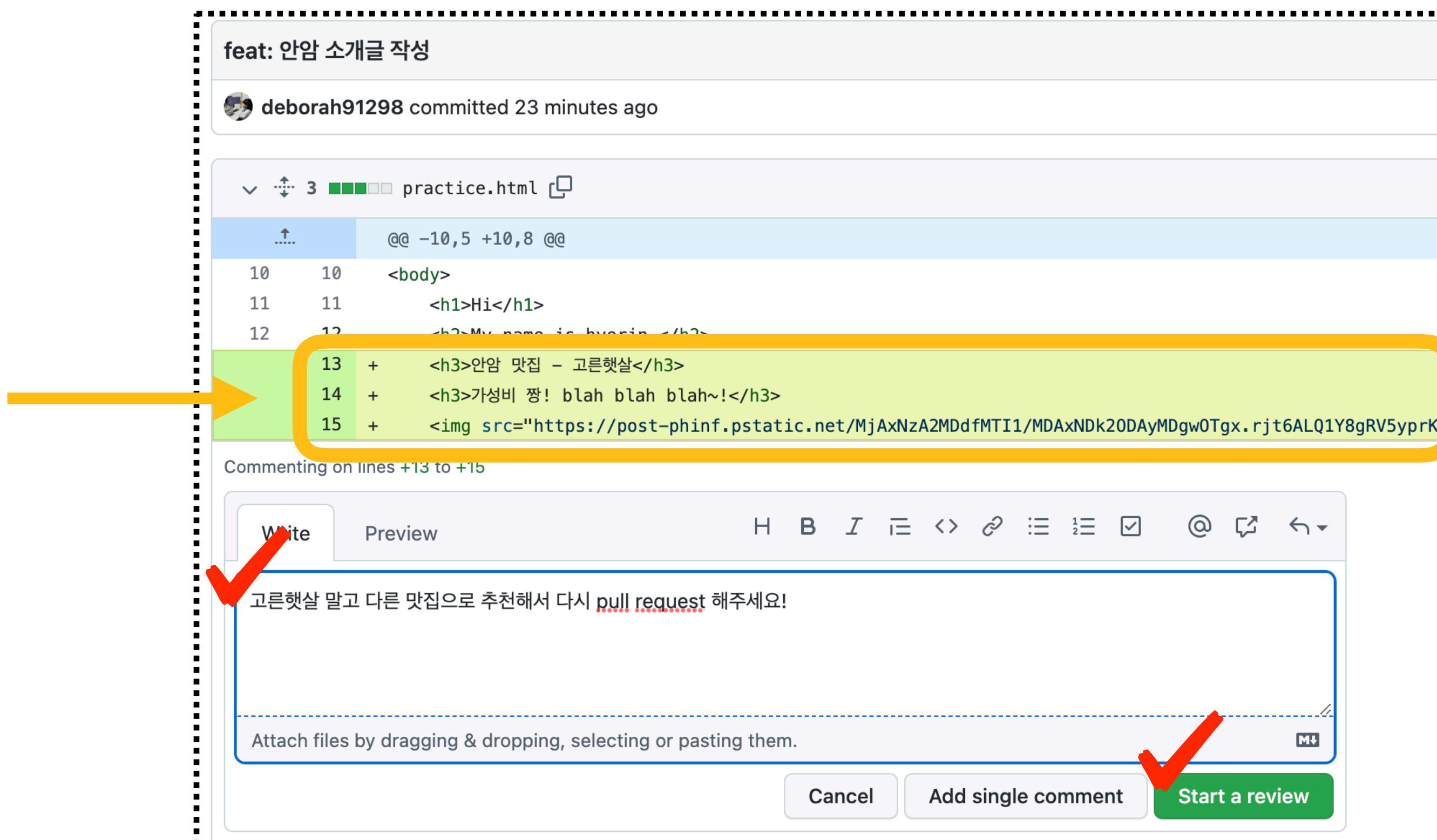


3. Git 심화

3-6. Code Review 하기!

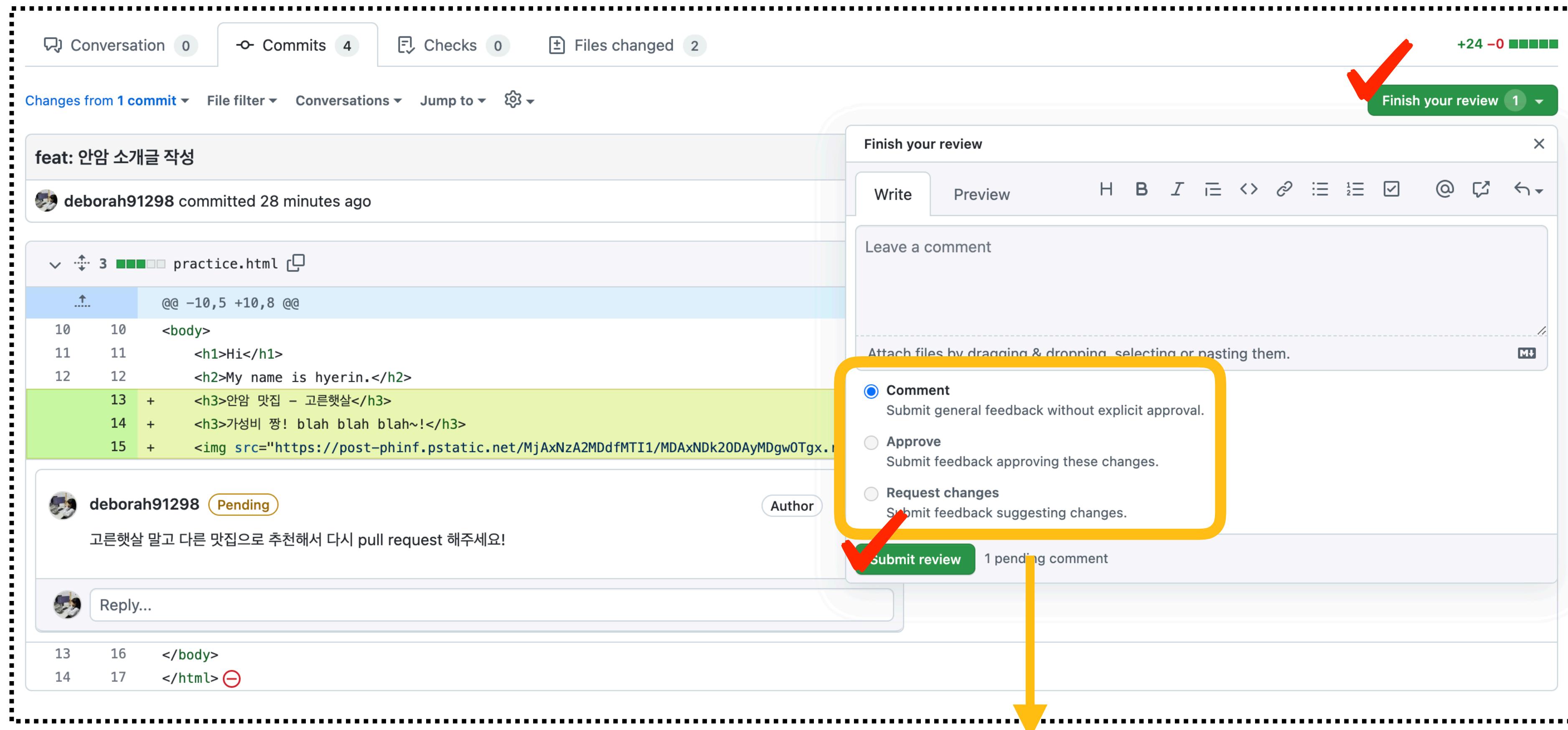
코드 리뷰!! 아주 중요하죠?
다른 사람 코드에 댓글 달아봅시다!

Drag&Drop으로
의견 달고 싶은
부분 코드 선택 가능!



3. Git 심화

3-6. Code Review 하기!



원하는 댓글 옵션 선택 가능!

3. Git 심화

3-6. Code Review 하기!

The screenshot shows a GitHub pull request interface. At the top, it says "deborah91298 reviewed now". Below this, there's a code diff for a file named "practice.html". The diff shows three additions:

```
practice.html
13 + <h3>안암 맛집 – 고른햇살</h3>
14 + <h3>가성비 짱! blah blah blah~!</h3>
15 + <img src="https://post-phinf.pstatic.net/MjAxNzA2MDdfMTI1/MDAxNDk2ODAyMDgwOTgx.rjt6ALQ1Y8g...>
```

Below the code diff, a comment from "deborah91298 now" reads: "고른햇살 말고 다른 맛집으로 추천해서 다시 pull request 해주세요!". There are "Reply..." and "Resolve conversation" buttons at the bottom of the comment.

4. 팀별 실습

Special Session 실습

1. 한 명이 해커톤을 위한 레포지토리를 만든다.
(Public으로, Readme 파일 추가해서)
2. 레포지토리 Collaborators에 팀원을 추가한다.
3. 각 팀원은 Readme 파일에 본인 자기소개를 작성하여 pull request 를 한다.
4. 모든 팀원 다 코드 리뷰 후 merge 한다.
5. 완성된 readme 파일 링크를 과제 링크에 제출한다. (~5/15 세션 전)