

Github로

협업하기

Git과 Github



Git

로컬 파일의 변경사항을 기록하고 해당 파일에 대한 여러 사용자 간의 작업을 조율하기 위한 분산형 버전 관리 시스템(소스 코드를 관리하는 도구)

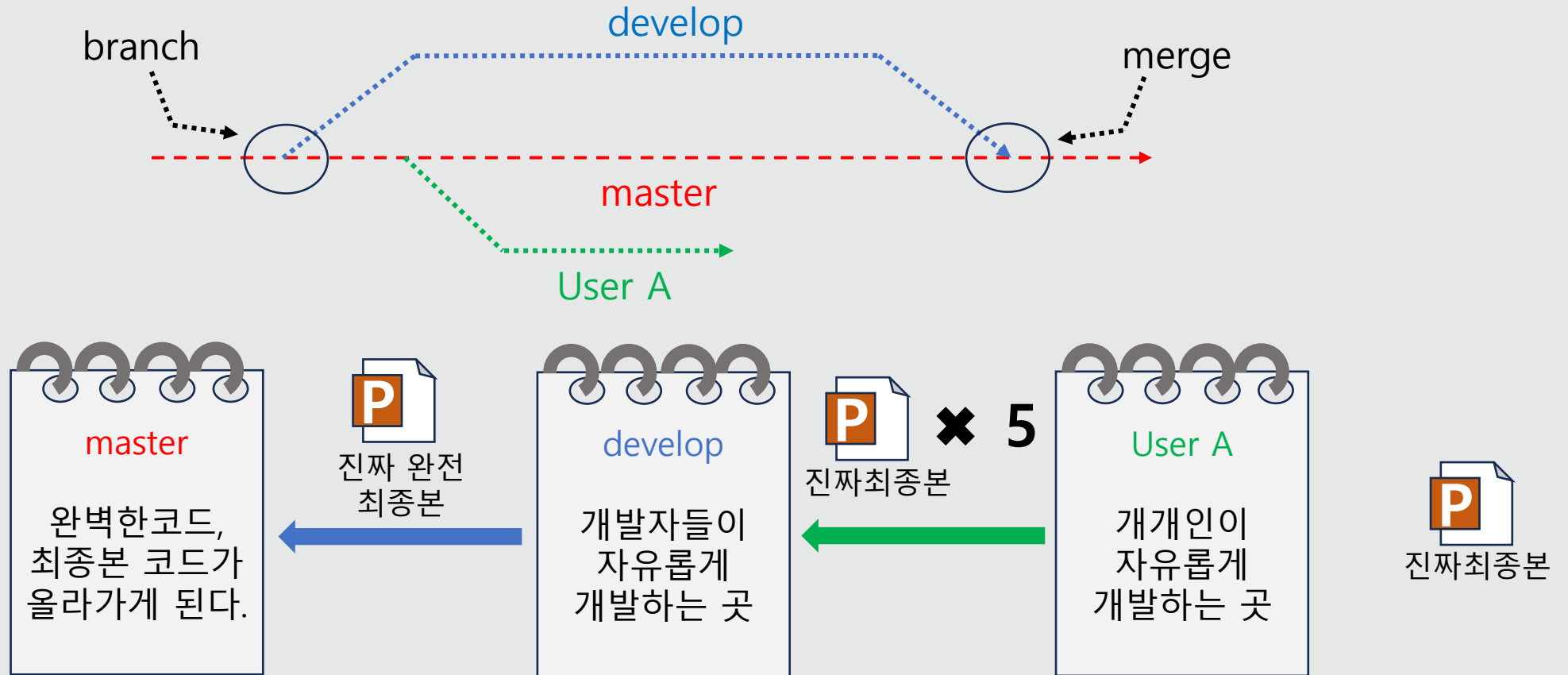
Github

Git 저장소를 관리하는 클라우드 기반 호스팅 서비스이다.

내 코드들을 백업해놓는 기능을 넘어서서 전세계의 오픈 소스 프로젝트들이 공유되고 개발자들을 도모하는 곳이다.

버전 관리, 소스 코드 공유, 분산 버전 제어 등이 가능한 원격 저장소

용어 설명



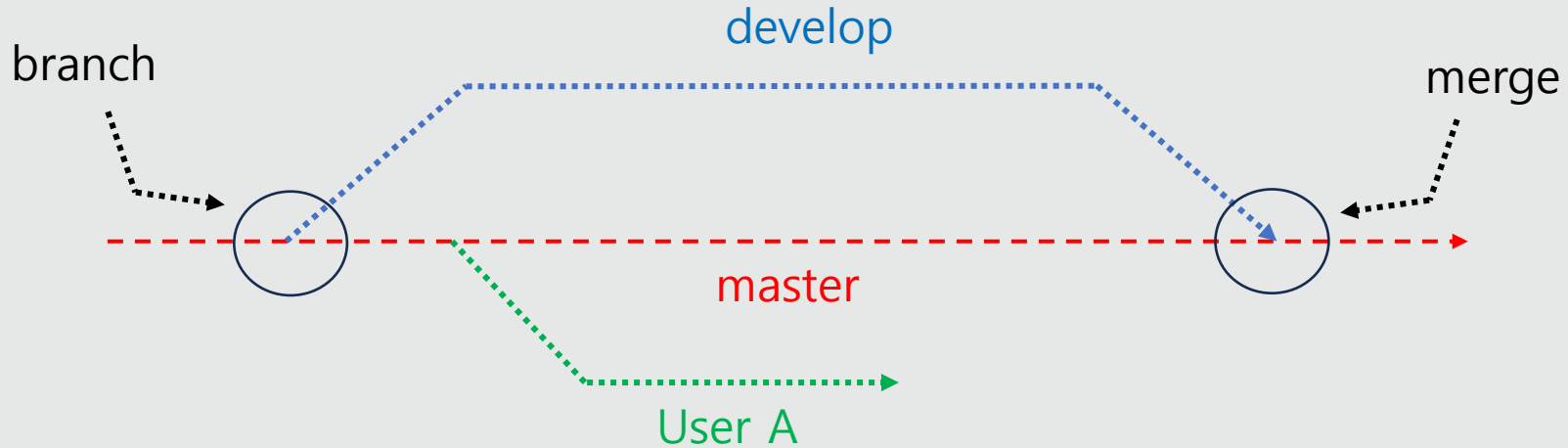
Branch

코드베이스를 분리하여 작업할 수 있는 개별적인 작업 공간을 의미합니다.
일반적으로 메인 코드베이스로부터 파생된 복사본이며, 해당 branch에서는 새로운 기능 개발, 버그 수정 등의 작업을 독립적으로 진행된다.

Merge

두 개 이상의 다른 branch에서 개발된 내용을 하나의 branch로 통합하는 과정을 말합니다.

용어 설명



깃 충돌

코드를 병합할 때 발생할 수 있는 문제로, 여러 개발자가 동시에 같은 파일의 다른 부분을 수정하거나, 한 개발자가 다른 branch에서 변경한 내용을 현재 작업 중인 branch에 병합할 때 일어나는 충돌이다.

Pull Request(PR)

branch로 내 코드를 보내는 작업

Git clone

github 저장소에 있는 파일을 내 pc에 복제하는 것

명령어 설명



깃허브에 코드 올리기

README.md

Update README.md

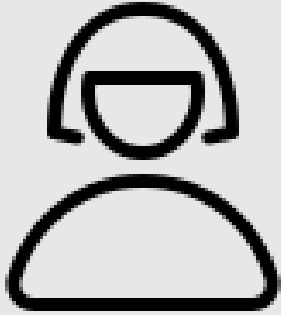
git add . - 추가할 파일 더하기 ("."은 모든 파일)
git commit -m "메시지 내용" - 히스토리 만들기 (m은 메시지 준말)
git push - Github에 코드 올리기

git checkout -b branch 이름 - Github에 브랜치 만들기

git checkout branch 이름 - 브랜치 이동

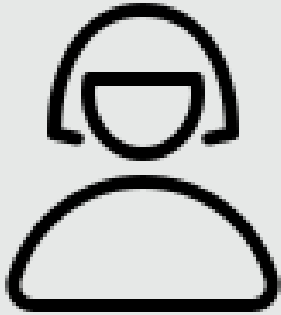
git merge branch 이름 - 내 브랜치와 branch 이름 브랜치를 합친다.

git pull origin branch 이름 - 브랜치에 코드 가져오기



프로젝트 팀장

팀원 1



팀원 2

- Git clone, 소스코드 올리기, PR

깃 충돌 해결

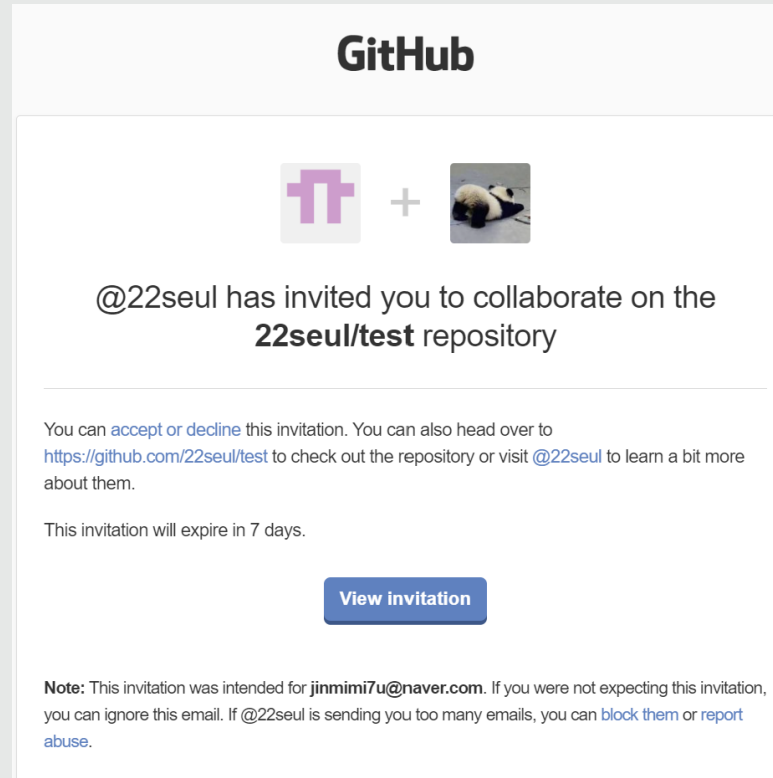
- 코드 merge, 코드 다시 올리기, master로 코드 배포

1. git clone 하기
2. branch 만들기
3. 개발해서 소스코드 github에 올리기
4. Pull Request(PR)하기
5. 깃 충돌 해결하기
6. master branch에 코드 배포하기

깃허브 협업

실습하기

1. git clone 하기



clone하기 전 invite 받기
(깃허브에 등록된 메일로 받을 수 있다.)

1. git clone 하기

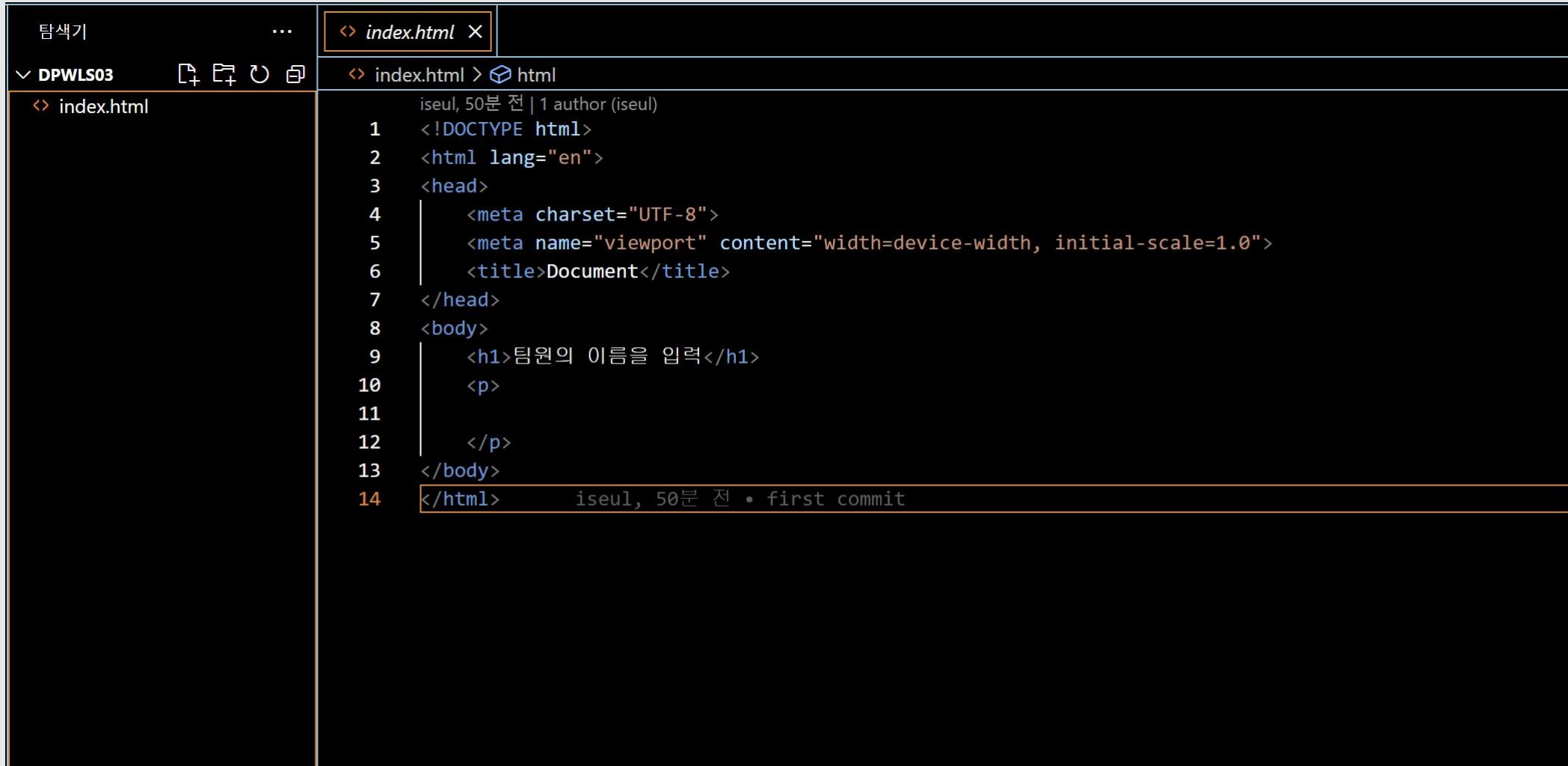
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 기능 및 개선 사항에 대한 최신 PowerShell을 설치 하세요! https://aka.ms/PSWindows

PS C:\Users\최예진\OneDrive\바탕 화면\project> git clone https://github.com/22seul/test.git dPwls03
Cloning into 'dPwls03'...
remote: Enumerating objects: 14, done.
remote: Counting objects: 100% (14/14), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 14 (delta 3), reused 8 (delta 2), pack-reused 0
Receiving objects: 100% (14/14), done.
Resolving deltas: 100% (3/3), done.
```

git clone + 깃허브 주소창 + 복제 파일 넣을 폴더 이름

1. git clone 하기

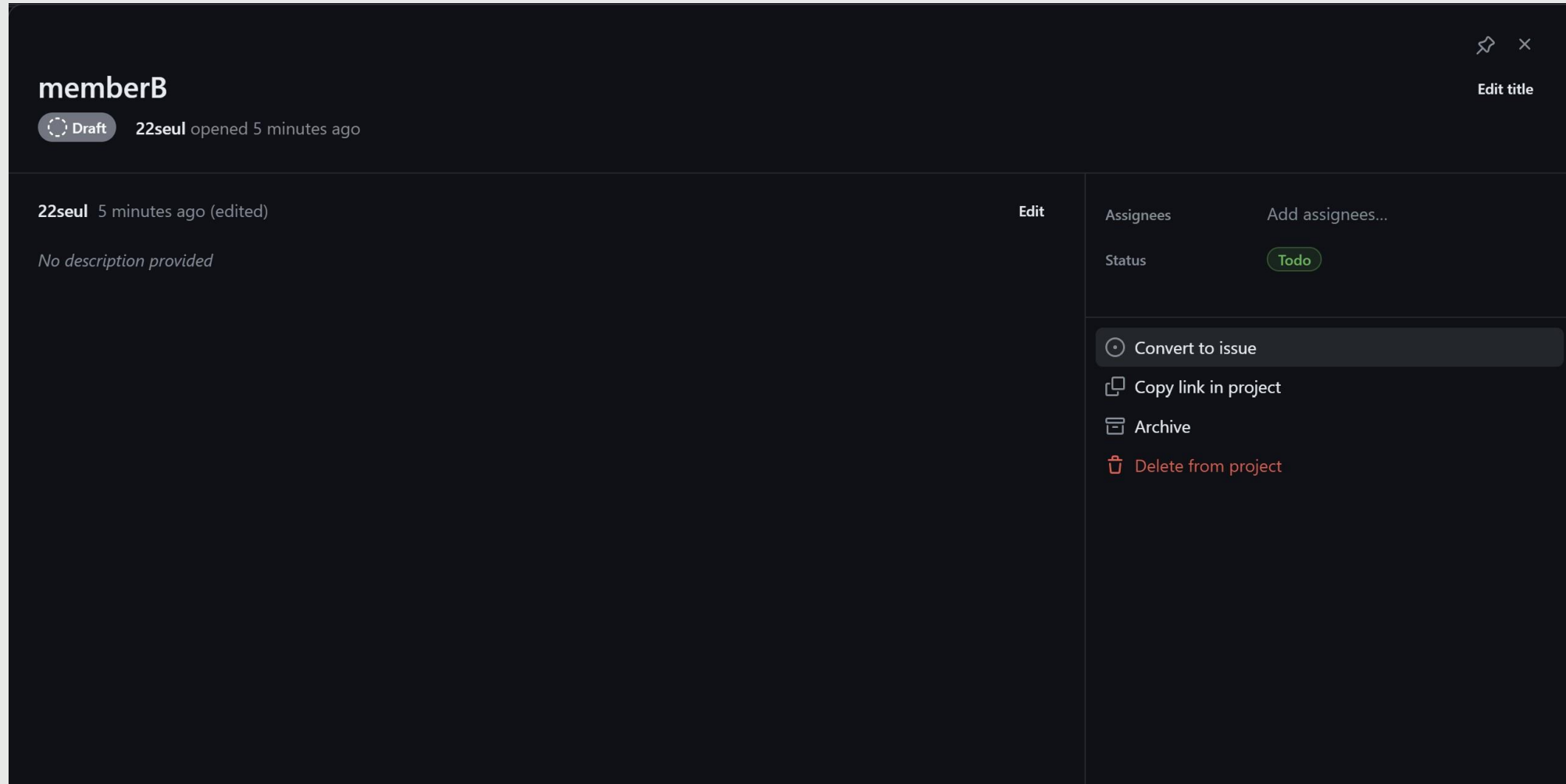


The screenshot shows the VS Code editor interface. On the left, the Explorer sidebar shows a project named 'DPWLS03' with a file 'index.html' selected. The main editor area displays the content of 'index.html'. The file is a standard HTML document with a head section containing meta tags for charset and viewport, and a body section containing a heading and a paragraph. The text is as follows:

```
iseul, 50분 전 | 1 author (iseul)
1 <!DOCTYPE html>
2 <html lang="en">
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>팀원의 이름을 입력</h1>
10  <p>
11
12  </p>
13 </body>
14 </html>    iseul, 50분 전 • first commit
```

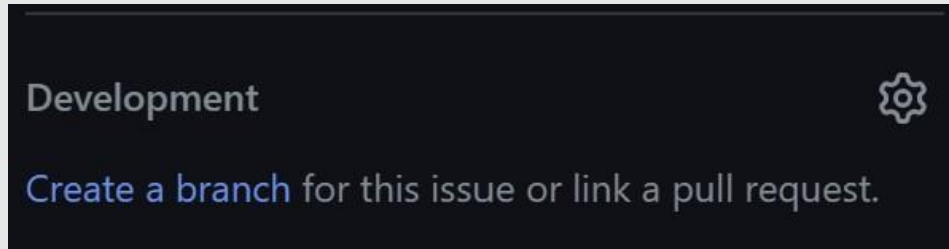
vscode에서 복제한 폴더 열기
마스터에 있던 index.html 파일 내용 그대로 있다.

2. Branch 만들기

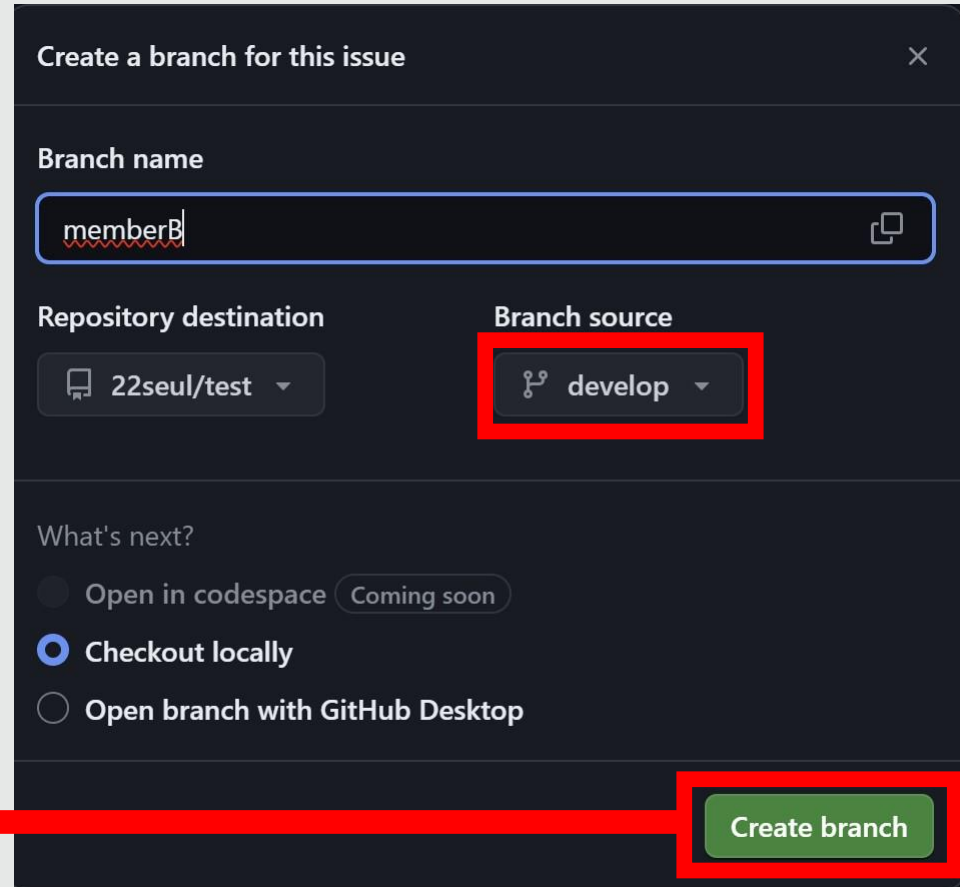
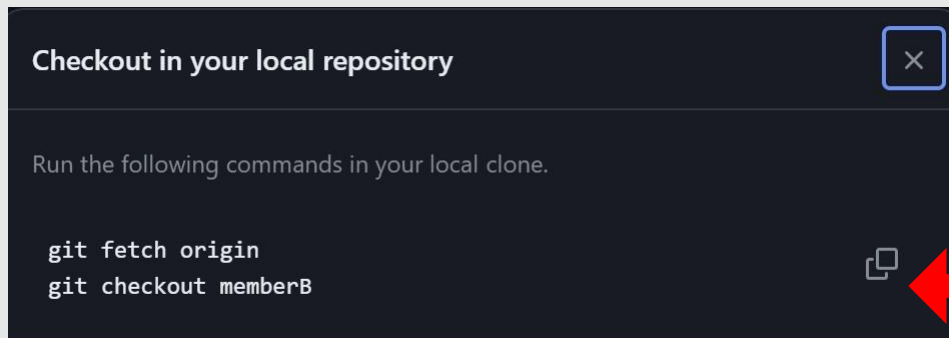


Convert to issue를 누르면 issue가 만들어진다.

2. Branch 만들기



Issue를 눌러서 오른쪽에 보면 create a branch



Branch name에 branch 이름을 적고 branch source는 develop으로 해서 create branch 눌러준다. 그러면 그림처럼 코드가 뜨게 되는데 저 코드를 복사한다.

2. Branch 만들기

```
문제  출력  디버그 콘솔  터미널  포트  GITLENS

최예진@BOOK-PMTSHF3Q4P MINGW64 ~/OneDrive/바탕 화면/project/dPwls03 (master)
$ git fetch origin
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 3 (delta 1), reused 3 (delta 1), pack-reused 0
Unpacking objects: 100% (3/3), 265 bytes | 44.00 KiB/s, done.
From https://github.com/22seul/test
* [new branch]      memberA    -> origin/memberA
* [new branch]      memberB    -> origin/memberB

최예진@BOOK-PMTSHF3Q4P MINGW64 ~/OneDrive/바탕 화면/project/dPwls03 (master)
$ git checkout memberB
Switched to a new branch 'memberB'
branch 'memberB' set up to track 'origin/memberB'.
```

vscode 터미널에서 git bash를 선택해주고 그곳에 복사한 코드를 붙여 넣어준다.
그러면 branch가 master에서 내가 만든 branch로 바뀌게 된다.

3. 개발해서 소스코드 github에 올리기

1. 코드에 내가 개발한 것을 개발하고 저장을 해준다.

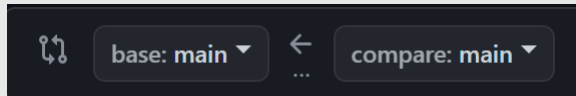
2. 터미널에

- git add .
- git commit -m "메시지 내용 쓰기"
- git push

3. github에 가면 내가 만든 branch가 있고 그곳에 내가 쓴 코드로 바뀌게 된다

4. Pull Request(PR)하기

1. Pull Request에서



base는 코드 받는 것 compare는 코드 보내는 것

5. 깃 충돌 해결하기



깃 충돌이 일어나는 이유는 develop branch 버전이 내가 알던 버전이 아니다.
이미 다른 코드가 들어가서 버전이 다르기 때문에 일어난다.

충돌 안 나게 하는 방법

1. 서로 다른 파일에서 작업하기
(같은 파일 같은 코드 줄에서 코드를 편집하기 때문에 깃 충돌이 일어나는 것)
2. Commit 자주 해주기
3. Merge 자주 해주기
4. Pull 자주 해주기

5. 깃 충돌 해결하기

Step 1: Clone the repository or update your local repository with the latest changes.

```
git pull origin
```



Step 2: Switch to the head branch of the pull request.

```
git checkout
```



Step 3: Merge the base branch into the head branch.

```
git merge
```



Step 4: Fix the conflicts and commit the result.

See [Resolving a merge conflict using the command line](#) for step-by-step instructions on resolving merge conflicts.

Step 5: Push the changes.

```
git push -u origin
```



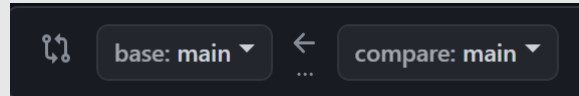
Command line으로 깃 충돌 해결하기
Step 따라서 해결하면 된다.

6. master branch에 코드 배포하기

모든 할 일이 끝나고 개발이 다 끝났다.

Develop branch도 완벽하고 더 이상 개발을 하지 않아도 되는 상태이고 모든 것이 완벽한 상태일 때 develop branch에 있는 파일을 master branch로 보낸다.

1. PR 만들기



Base를 master로 compare를 develop으로 설정

2. X 표시 두 개

- ❌ 1. 리뷰를 반드시 해라
- ❌ 2. 팀장이 branch 보호한 것 때문에 발생

3. X 표시 두 개를 해결하면 merge request가 가능하다.

4. Master 코드를 보면 최종본 코드가 있다.

Reference

<https://www.hohyeonmoon.com/blog/swift-git-github/>

<https://velog.io/@t1s113/Git-Github>

https://hackmd.io/@oW_dDxdsRoSpl0M64Tfg2g/ByfwpNJ-K - 명령어 모음

유튜브

<https://www.youtube.com/watch?v=leIVripbt2M> - 깃과 깃허브

<https://www.youtube.com/watch?v=tkkbYCajCjM> - 깃허브로 협업하기

<https://www.youtube.com/watch?v=PGQIJE4tHAs> - 깃 충돌

감사

합니다