

## 2025 깃헙 스터디

서강대학교 Release 학회





**이름 : 조현상**

**경력: AI Solution 개발자 1년차**

**소개:**

- FastAPI 기반의 서버를 구축하여 LLM을 활용한 AI 솔루션 개발
- 개발부터 배포까지 A to Z 전 과정 수행
- 실제 프로젝트의 최종 배포 및 운영 경험 보유

**취미:**

- 알고리즘 문제 풀기
- CS 지식 쌓기
- 파이썬 공부





## 스터디 소개

1. 본 스터디에서는 실무에 가까운 Git/GitHub 활용 역량을 키우는 것을 목표로 합니다.
2. Git/GitHub의 이론을 학습한 뒤, 간단한 실습을 통해 직접 적용해보며 이해를 높입니다.
3. 매주 알고리즘 1문제를 해결하고, 실무 협업에 사용하는 GitHub 컨벤션에 맞춰 GitHub에 체계적으로 업로드합니다.
4. 모든 참여자가 협업 중심의 Git/GitHub 활용에 익숙해지고, 실무에서도 자신 있게 사용할 수 있도록 하는 것이 본 스터디의 궁극적인 목표입니다.

### Git 스터디

단기간에 Git 고수되기

Q. 이 스터디의 학습 목표는 무엇인가요?

Git을 배우고 직접 실습해본다.  
Convention Rule에 맞춰서 팀 프로젝트를 경험해본다.

Q. 학습 방법과 계획은 어떻게 되나요?

Git의 기초적인 사용법을 익힌다.  
백준 문제풀이를 통해 Git 협업을 익힌다.

Q. 이 스터디에서는 무엇을 배우나요?

기초적인 Git 사용법을 배운다.  
실무에서 사용하는 branch 관리법을 배운다.

Q. 이 스터디에 참여하기에 권장되는 수준은 무엇인가요?

Python으로 백준 기초 문제를 풀 수 있다.  
Git 계정을 소유하고 있다.







## 스터디 일정

1주차 - 스터디 소개 & Git/GitHub 이론 설명

2주차 - 컨벤션 룰 설명 & 실습 설명

3주차 - GitHub 유지 보수 및 관리 하는 방법

4주차 - GitHub Tag 기능 소개 & 브랜치 전략

5주차 - 깃헙 소스 코드 관리 (gitignore, gitattributes, env, README)

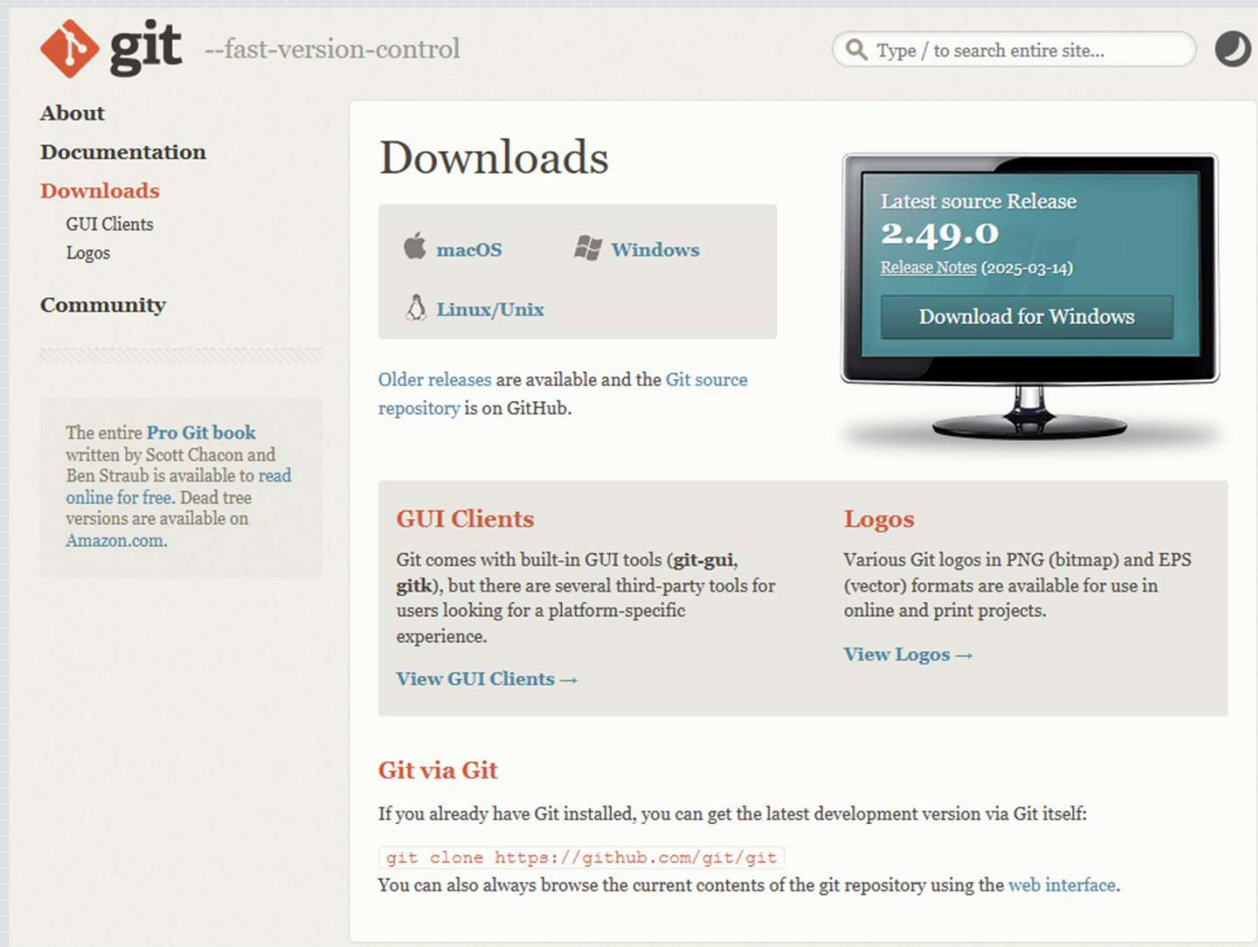
6주차 - 간단한 웹 백엔드 구현 및 웹 팀 프로젝트를 이용한 깃헙 핸들링 경험

7주차 - Git Action 을 사용한 CI 경험

(스터디 일정은 유동적으로 조정될 수 있습니다. 더 나은 방향을 위한 아이디어나 제안은 언제든지 환영합니다!)

Git이 아직 설치 안 되어 있다면, 아래 링크에서 미리 설치 부탁드립니다!

 <https://git-scm.com/downloads>



The screenshot shows the Git website's Downloads page. The header includes the Git logo, the tagline "--fast-version-control", and a search bar. The left sidebar contains links for "About", "Documentation", "Downloads" (highlighted), "GUI Clients", "Logos", and "Community". The main content area is titled "Downloads" and features a large monitor graphic displaying the "Latest source Release 2.49.0" with a "Download for Windows" button. Below the monitor, it states "Older releases are available and the Git source repository is on GitHub." The page is divided into two columns: "GUI Clients" and "Logos". The "GUI Clients" section mentions built-in tools like `git-gui` and `gitk`, and provides a link to "View GUI Clients →". The "Logos" section mentions various Git logos in PNG and EPS formats and provides a link to "View Logos →". At the bottom, the "Git via Git" section explains how to get the latest development version via Git itself, showing the command `git clone https://github.com/git/git` and a link to the "web interface".

**git** --fast-version-control

Search: Type / to search entire site...

**About**  
**Documentation**  
**Downloads**  
GUI Clients  
Logos  
**Community**

The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.

## Downloads

Latest source Release  
**2.49.0**  
[Release Notes \(2025-03-14\)](#)  
[Download for Windows](#)

Older releases are available and the Git source repository is on GitHub.

### GUI Clients

Git comes with built-in GUI tools (**git-gui**, **gitk**), but there are several third-party tools for users looking for a platform-specific experience.  
[View GUI Clients →](#)

### Logos

Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.  
[View Logos →](#)

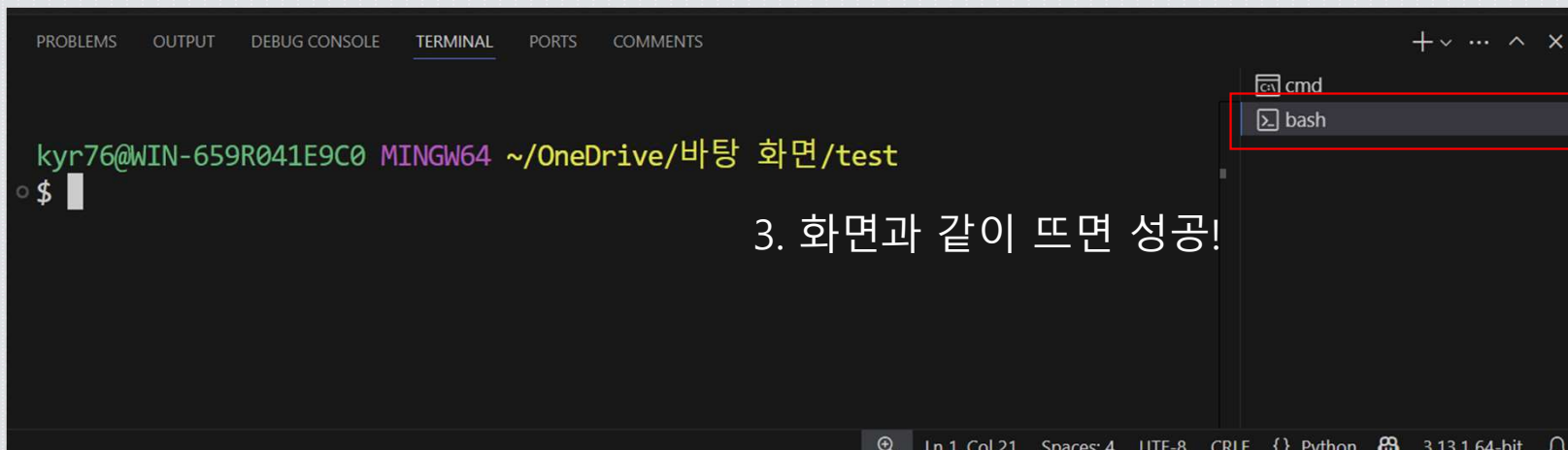
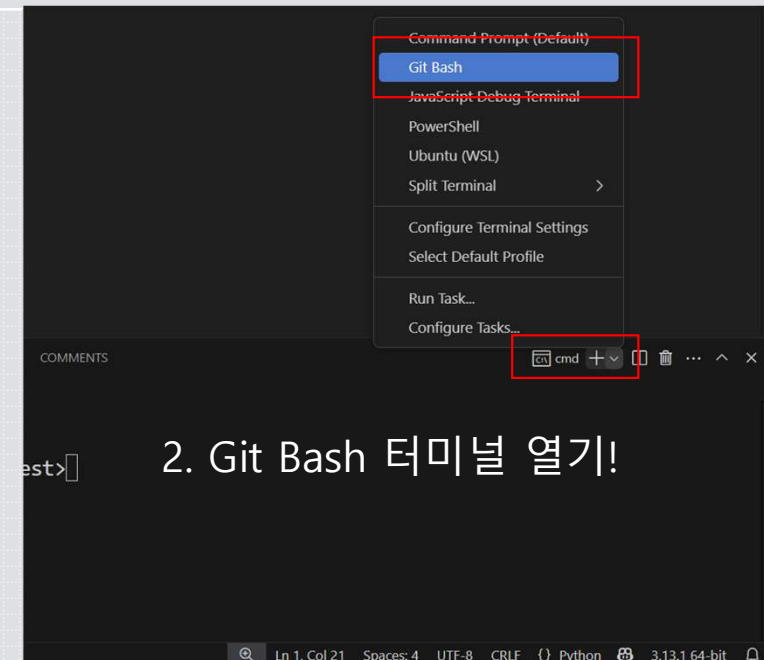
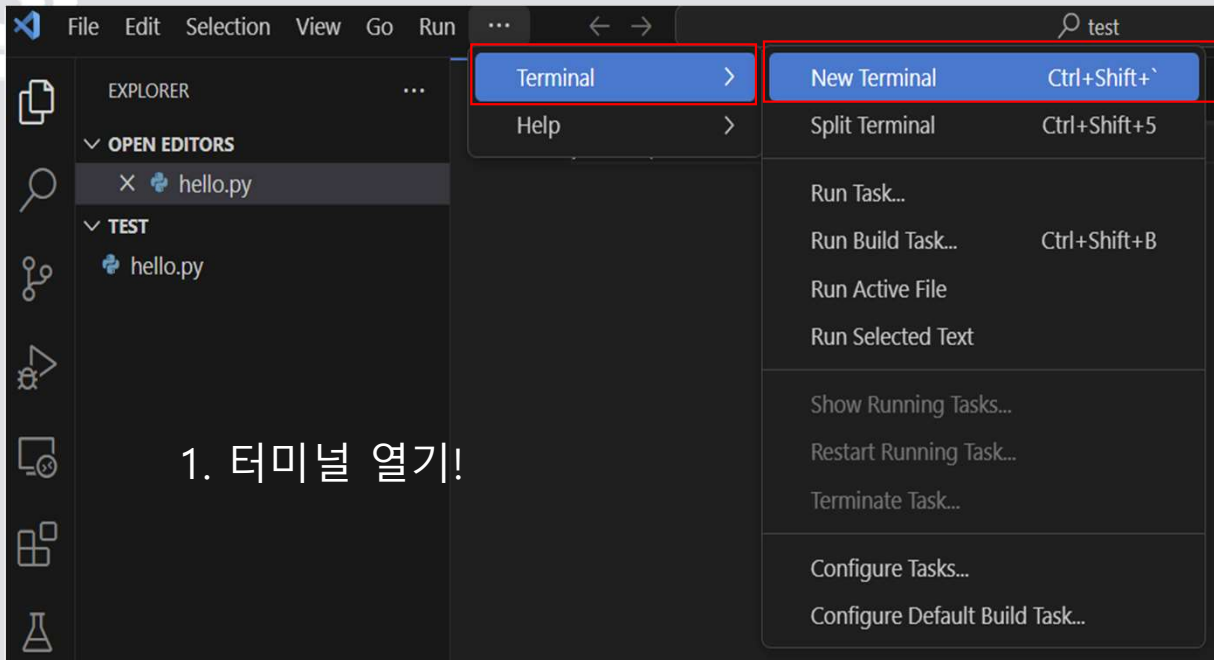
### Git via Git

If you already have Git installed, you can get the latest development version via Git itself:

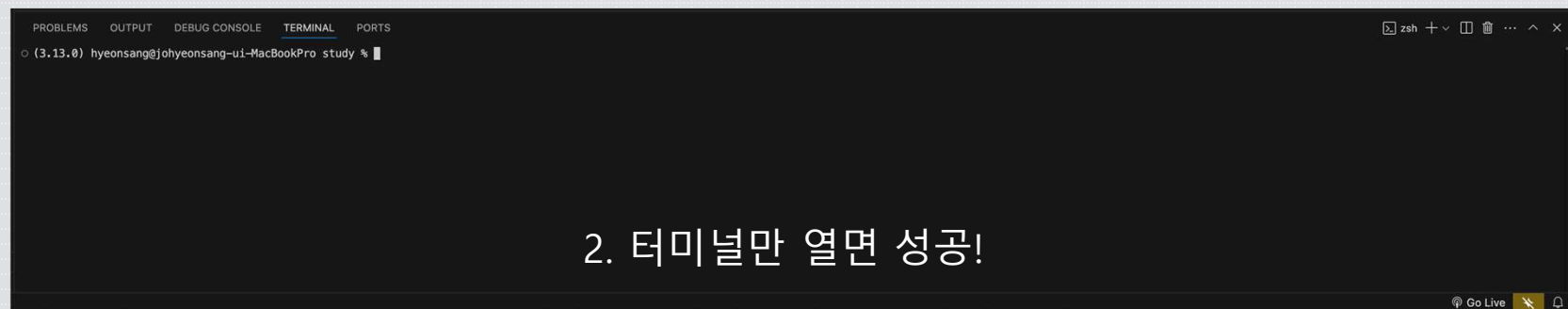
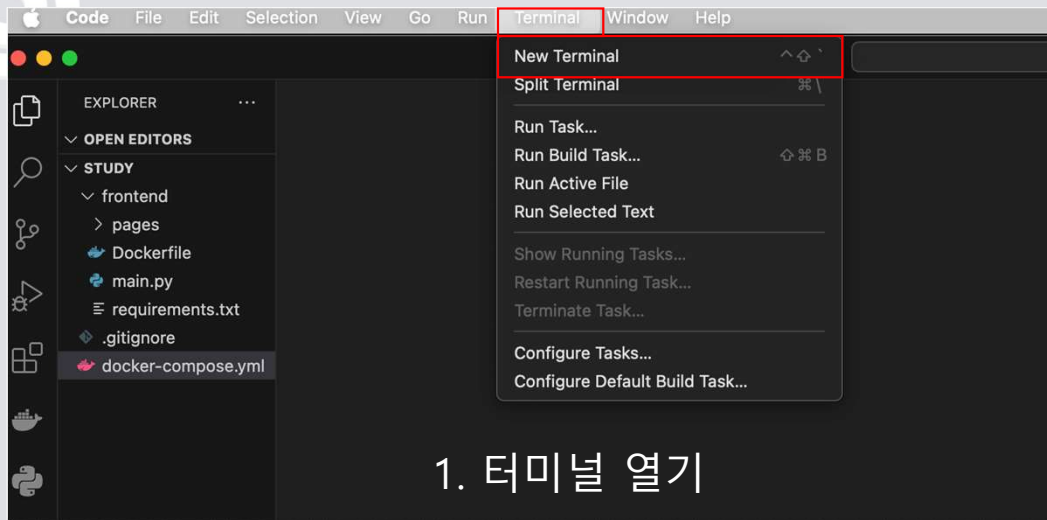
```
git clone https://github.com/git/git
```

You can also always browse the current contents of the git repository using the [web interface](#).

## 스터디 시작 전(윈도우)



## 스터디 시작 전(맥)





## 스터디 시작 전(공통)

깃헙 이메일, 아이디 등록 (등록하지 않으면 Commit을 할 수 없음)

```
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/test
• $ git config --global user.name "hyeonsang010716"

kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/test
• $ git config --global user.email "kyr76789@gmail.com"
```

화면과 같이 뜨면 성공!

```
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/test
• $ git config --global --list
user.email=kyr76789@gmail.com
user.name=hyeonsang010716
```

만약 해당 리포지토리만 적용하고 싶으면 `--global` 삭제  
(`git config user.name` / `git config user.email`)  
단 `git init` 이 먼저 되어있어야 함.

```
git config --global user.name ""
git config --global user.email ""
git config --global --list
```





## GitHub 이란?

간단하게 설명하면 **코드 저장소 플랫폼!**

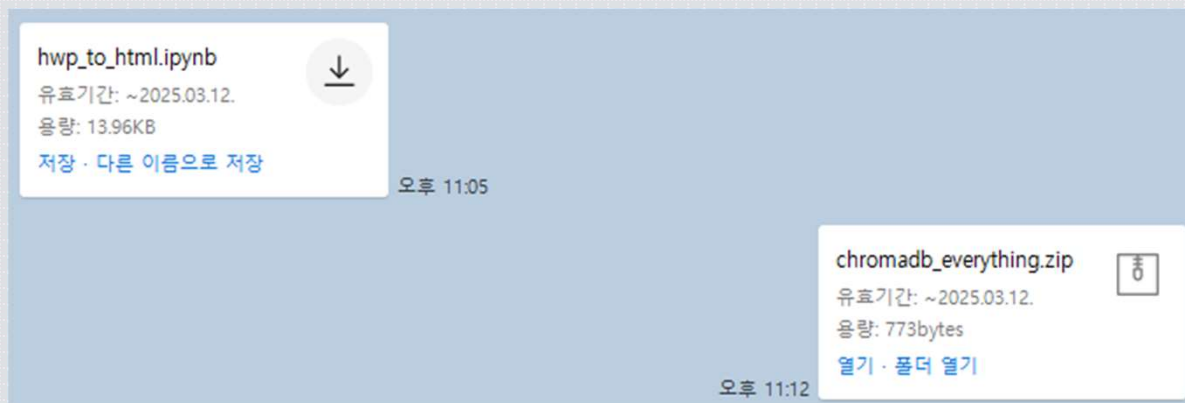
### <GitHub의 장점>

- **코드 백업 및 관리:** 변경 이력을 체계적으로 관리
- **협업:** 여러 명이 동시에 같은 프로젝트 작업 가능
- **이슈 관리 & 코드 리뷰:** 효율적인 커뮤니케이션과 품질 관리
- **오픈소스 기여:** 다양한 공개 프로젝트에 자유롭게 참여 가능



## GitHub 이란?

만약 GitHub을 사용하지 않는다면....

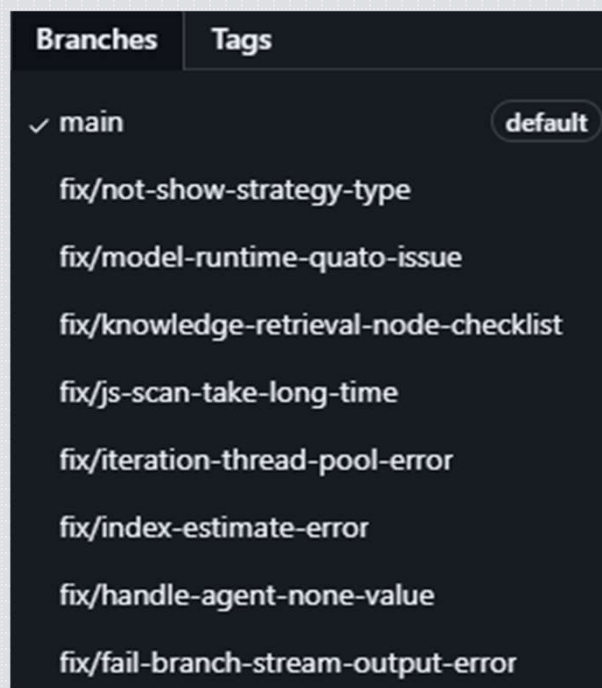


- 협업하는 사람이 3명 이상만 되어도 혼돈의 카오스 시작..
- 에러가 발생했을 때, 어디서부터 고였는지 알 수 없음..
- 내 소스 코드가 바뀌었는데 누가 바꿨는지 모름.. (마피아 게임 시작..)



## GitHub 이/란?

만약 GitHub을 사용한다면!!!



(브랜치 일부분)

협업하는 사람이 3명 이상만 되어도 혼돈의 카오스 시작..



N명에서 협업 가능

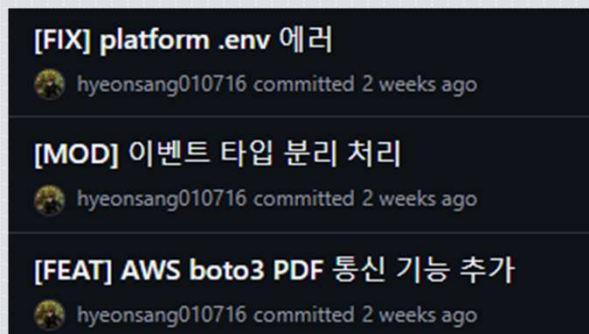
난 이 기능하고 있을 거니까  
넌 이 기능하고 있어~





## GitHub 이란?

만약 GitHub을 사용한다면!!!



(히스토리 일부분)

히스토리를 추적할 수 있음!

에러가 발생했을 때, 어디서부터 고였는지 알 수 없음..

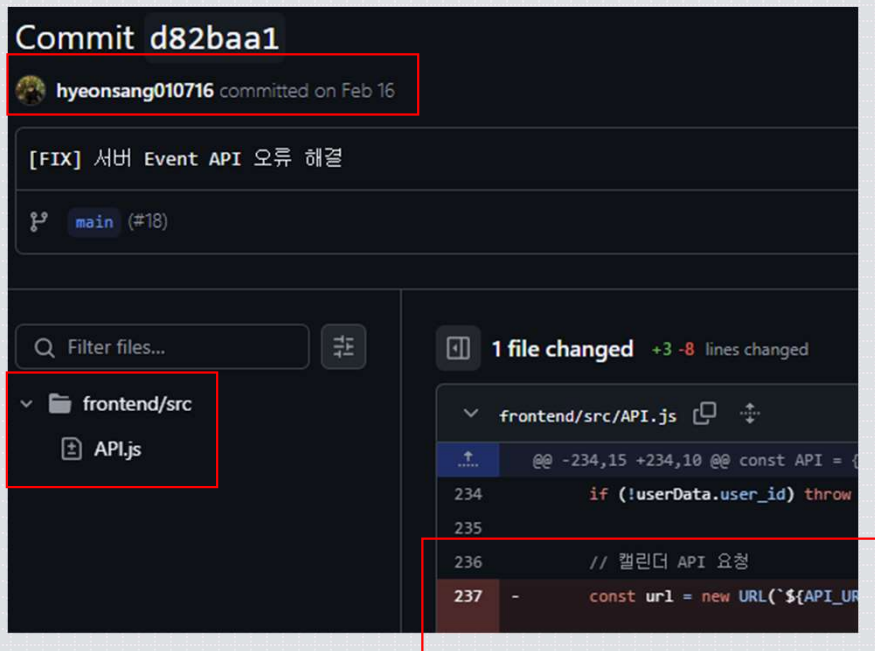


에러가 발생했을 때, 범인 찾기 가능

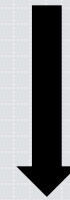
## GitHub 이란?

만약 GitHub을 사용한다면!!!

내 소스 코드가 바뀌었는데 누가 바꿨는지 모름.. (마피아 게임 시작..)



(커밋 상세 페이지 일부분)



경찰 승리

누가, 언제, 무슨 파일에,  
무슨 코드를..

전부 추적 가능

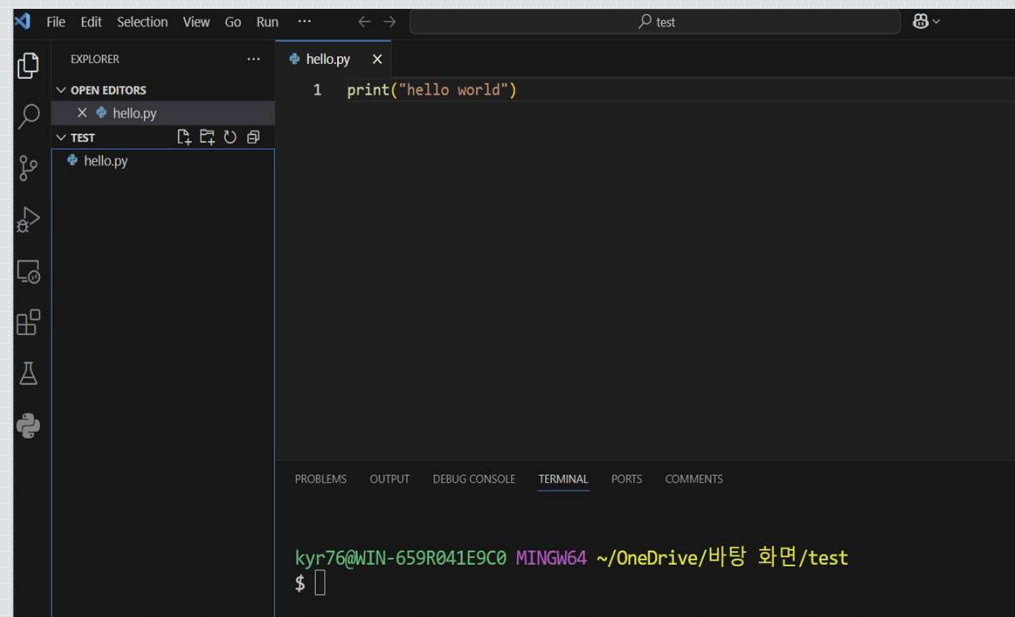
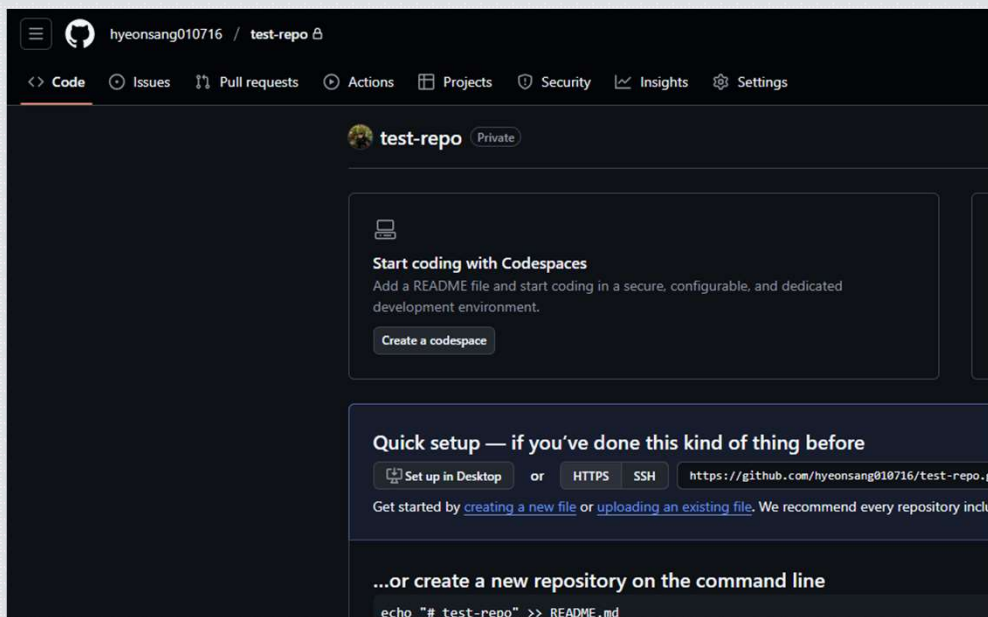


## GitHub 사용하기!

GitHub를 쉽게 이해하려면, 먼저 '**로컬**'과 '**리모트**'가 어떤 의미인지부터 알아두는 게 좋아요!

리모트(인터넷 깃헙)

로컬(내 컴퓨터, Vscode)



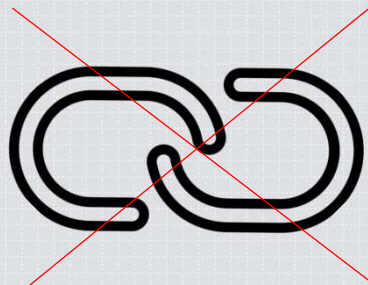




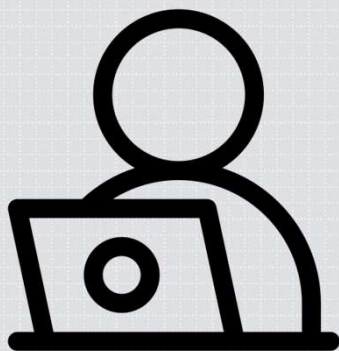
## GitHub 사용하기!

당연! 처음에는 **리모트**와 **로컬**은 연결이 되어있지 않아요

리모트



로컬

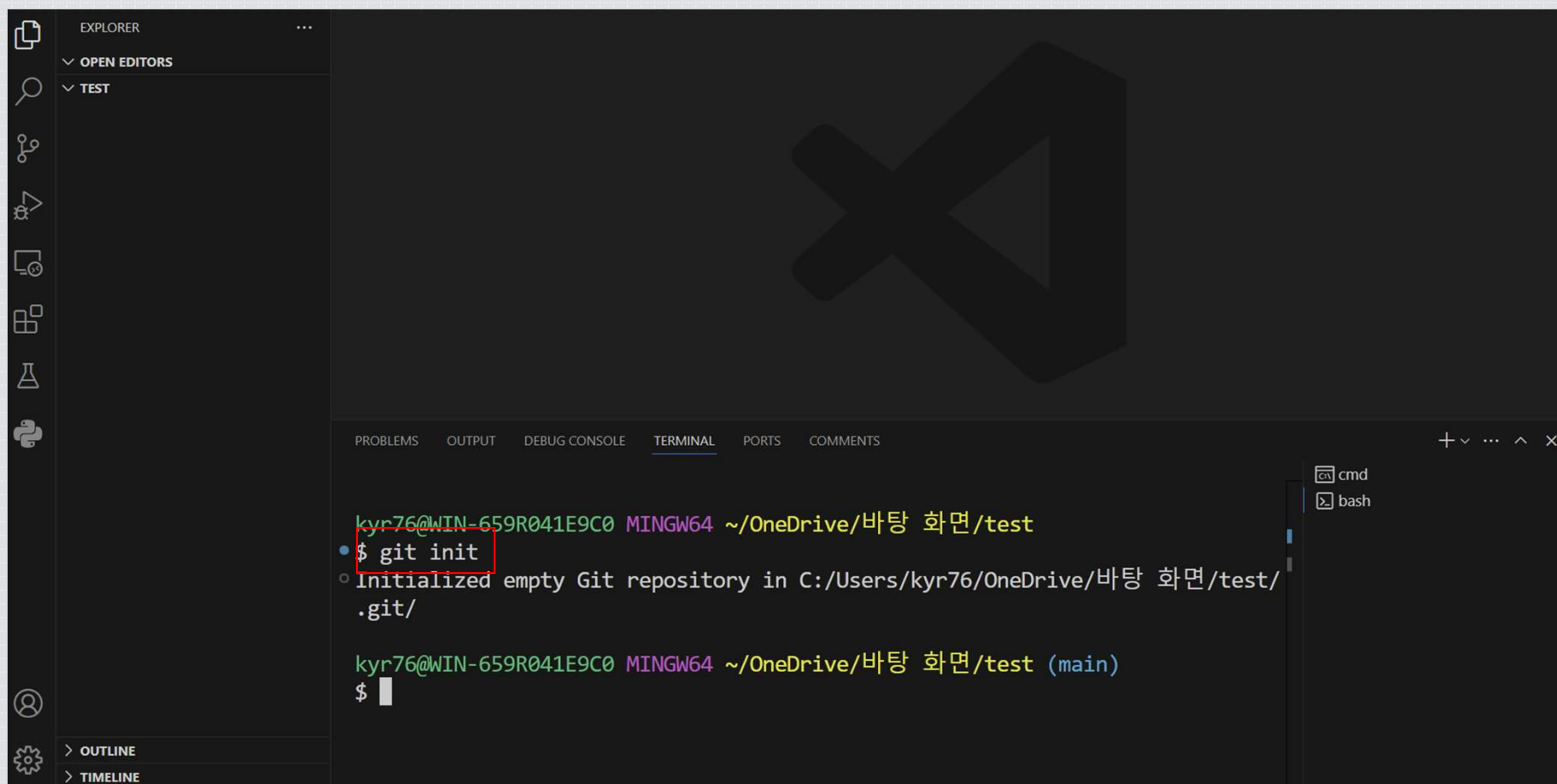


내 소스코드(**로컬**)를  
깃헙(**리모트**)에 올리고 싶은데  
어떻게 올리지..



## GitHub 연결하기!

가장 먼저 깃 명령어를 사용할 준비를 합니다.  
**git init**



The screenshot shows the Visual Studio Code interface with a terminal window open at the bottom. The terminal displays the command `git init` being executed in a Windows command prompt (MINGW64). The output indicates that an empty Git repository has been initialized in the specified directory. The command prompt shows the user is on the `main` branch.

```
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/test
$ git init
Initialized empty Git repository in C:/Users/kyr76/OneDrive/바탕 화면/test/.git/

kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/test (main)
$
```



## GitHub 연결하기!

리모트와 로컬을 연결하기!

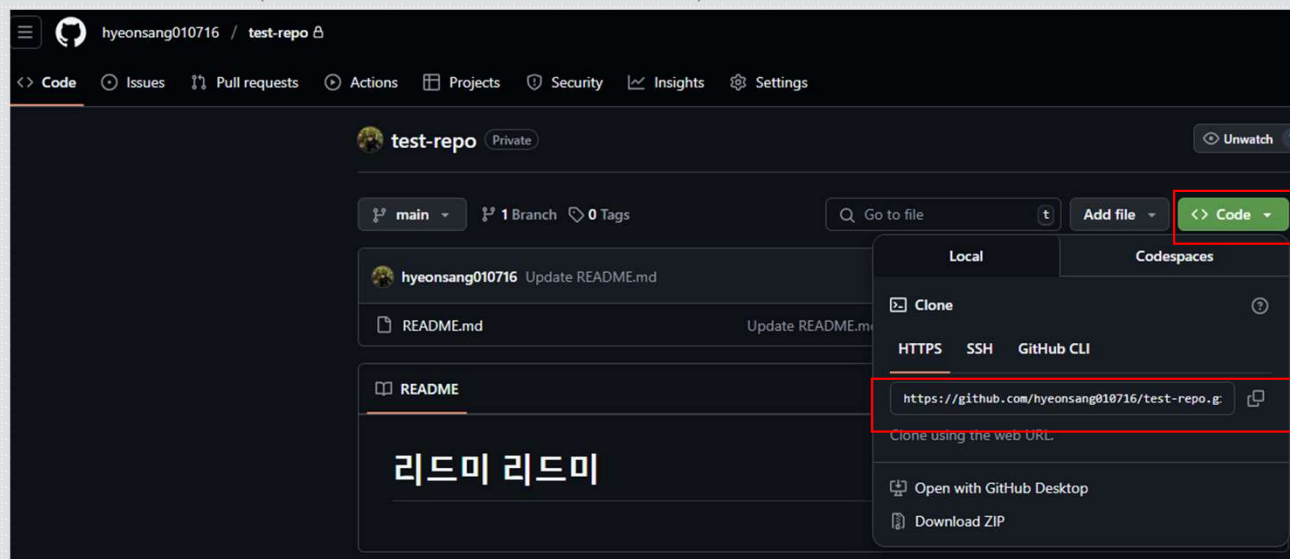
git remote add 리모트별명 리모트주소

예시)

git remote add origin https://github.com/hyeonsang010716/test-repo.git



(리포지토리 처음 만들었을 때 위치)



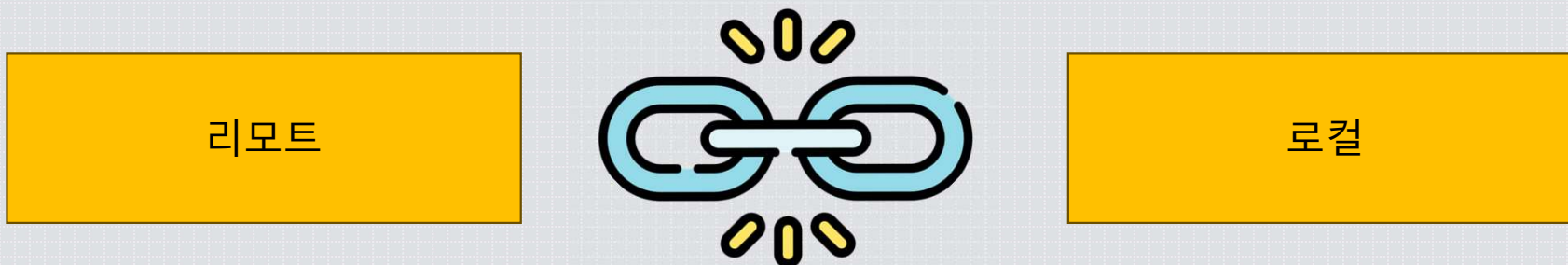
(리포지토리 기본 위치)





## GitHub 연결하기!

리모트와 로컬 연결 완료!



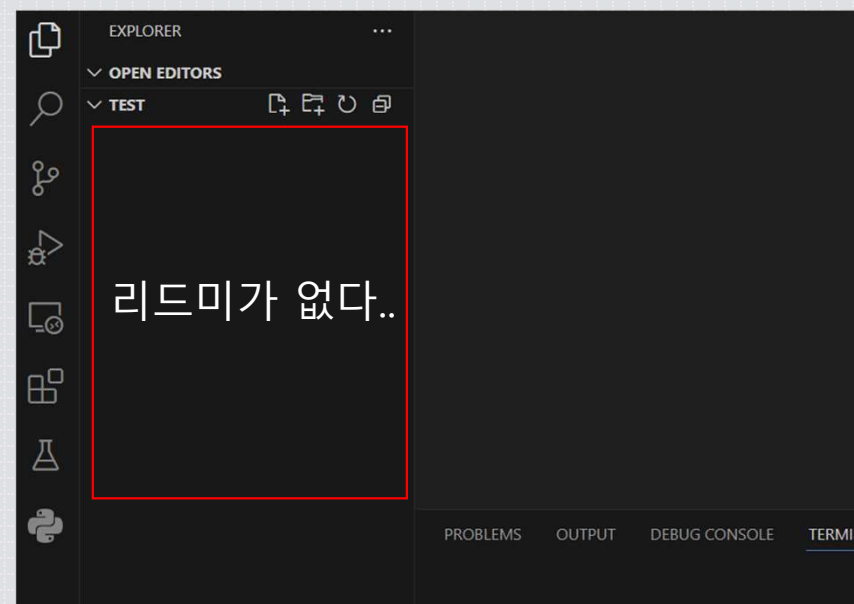
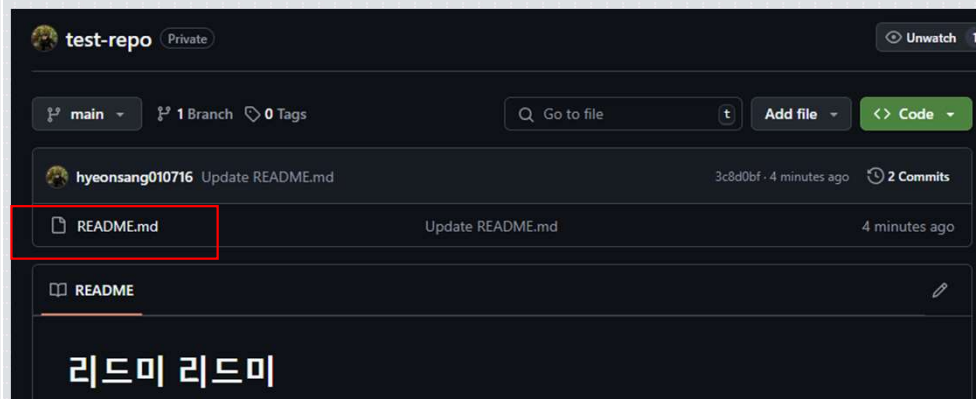
```
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/test (main)
• $ git remote add origin https://github.com/hyeonsang010716/test-repo.git

kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/test (main)
○ $ █
```



## GitHub 연결하기!

주의할 점 아직 연결만 되어있는 상태예요!  
연결 후, main까지 가야 완성입니다!





## GitHub 연결하기!

리모트에 있는 모든 브랜치 내용을 가져와줍니다.  
`git remote update`

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/test (main)

• \$ `git remote update`

remote: Enumerating objects: 6, done.

remote: Counting objects: 100% (6/6), done.

remote: Compressing objects: 100% (2/2), done.

remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)

Unpacking objects: 100% (6/6), 1.70 KiB | 29.00 KiB/s, done.

From <https://github.com/hyeonsang010716/test-repo>

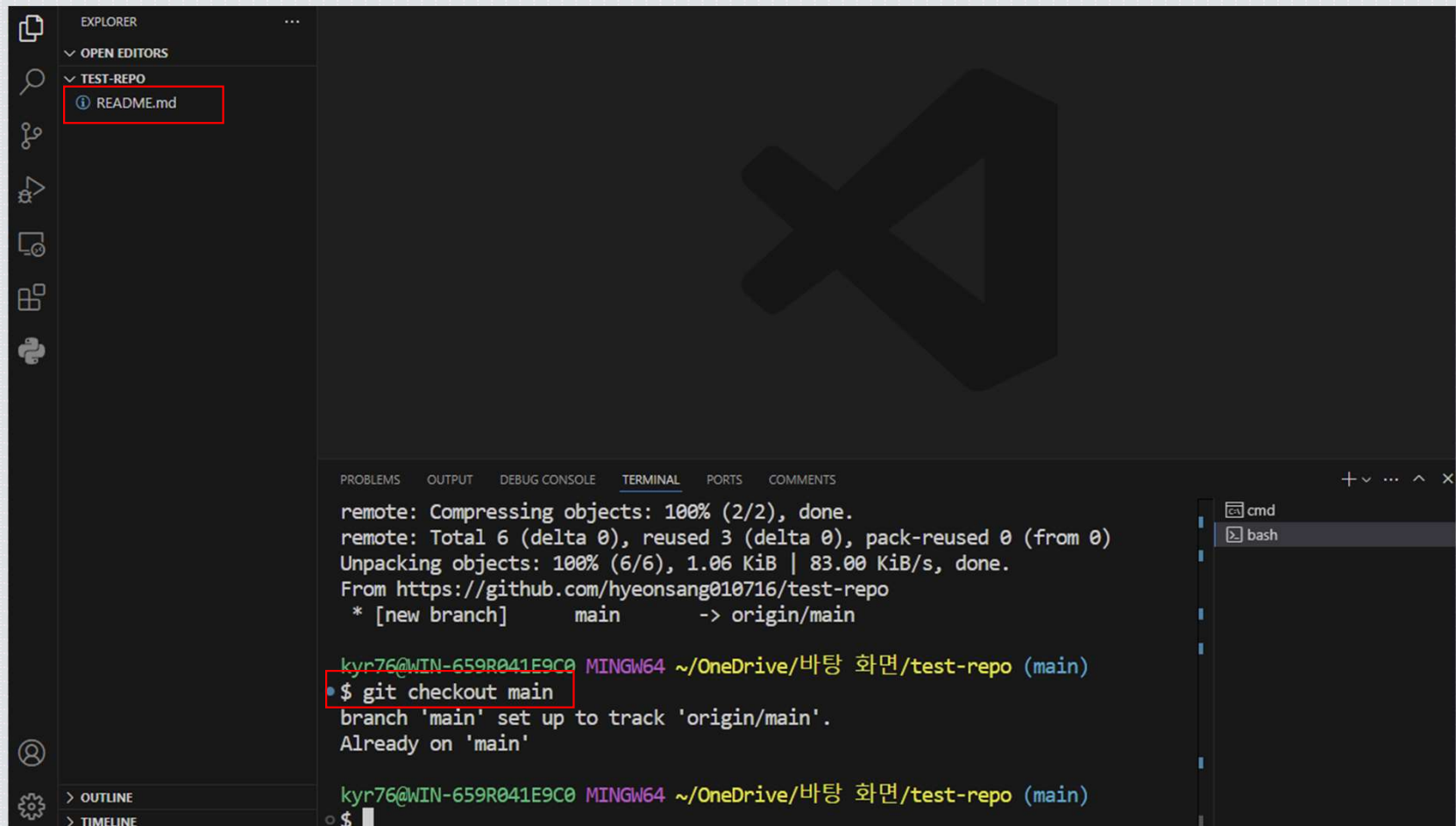
\* [new branch] main -> origin/main



## GitHub 연결하기!

리모트에 있는 모든 브랜치를 가져왔다면,  
마지막으로 main으로 이동해줍니다.

**git checkout main**



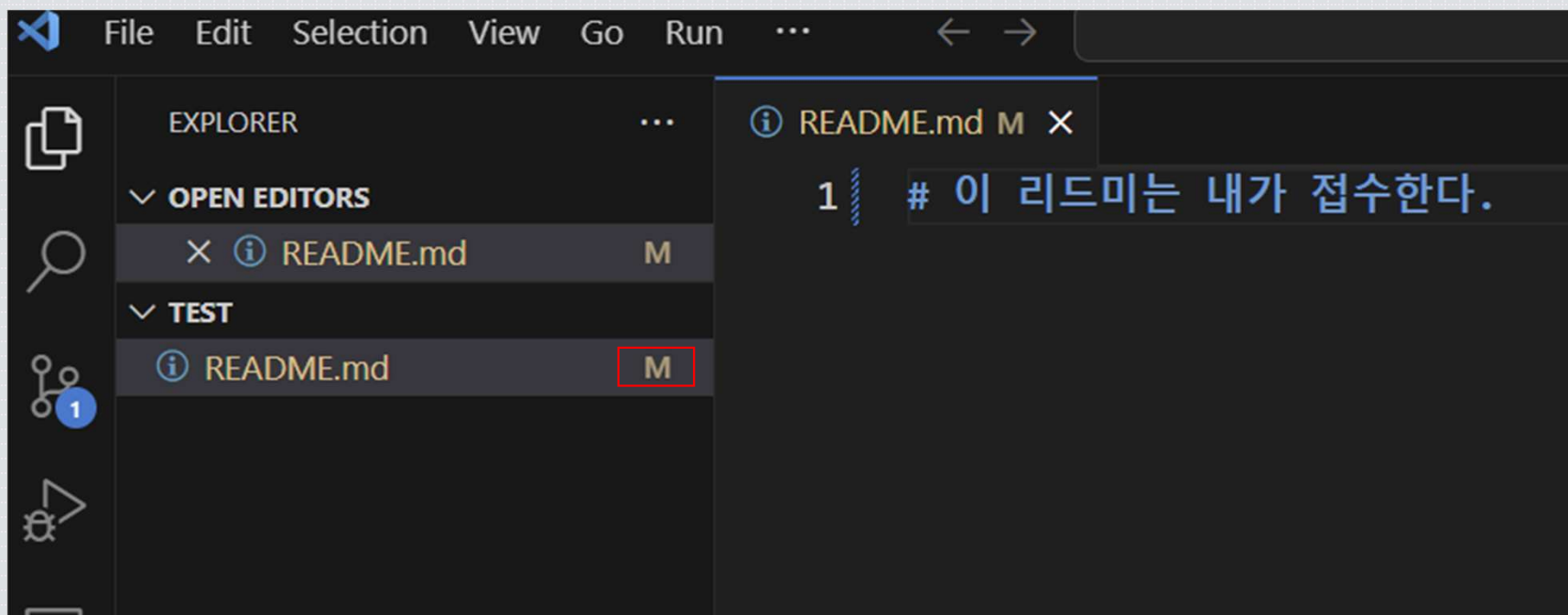
The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar shows a file named 'README.md' under the 'TEST-REPO' folder, which is highlighted with a red box. The main editor area is currently blank, displaying a large, faint GitHub logo watermark. At the bottom, the Terminal panel is open, showing the output of a git pull command and the execution of 'git checkout main'. The command and its output are highlighted with a red box. The terminal text is as follows:

```
remote: Compressing objects: 100% (2/2), done.  
remote: Total 6 (delta 0), reused 3 (delta 0), pack-reused 0 (from 0)  
Unpacking objects: 100% (6/6), 1.06 KiB | 83.00 KiB/s, done.  
From https://github.com/hyeonsang010716/test-repo  
* [new branch]      main      -> origin/main  
  
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/test-repo (main)  
• $ git checkout main  
branch 'main' set up to track 'origin/main'.  
Already on 'main'  
  
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/test-repo (main)  
$
```



## GitHub 파일 업로드 하기

파일을 수정하면 아래와 같이 M이 뜹니다.



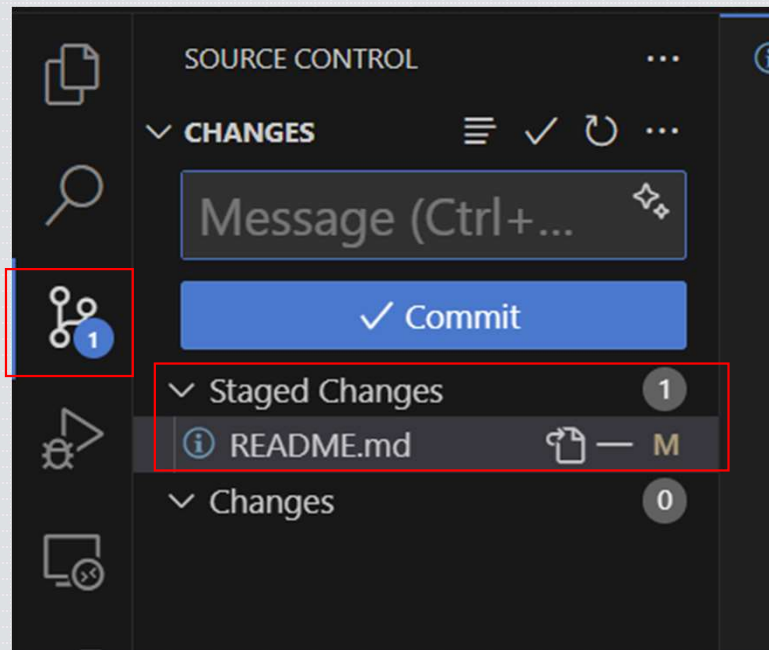
## GitHub 파일 업로드 하기

수정된 파일을 업로드 하기 위해선, 먼저 스테이징을 해야 해요!

git add 파일이름

git add README.md

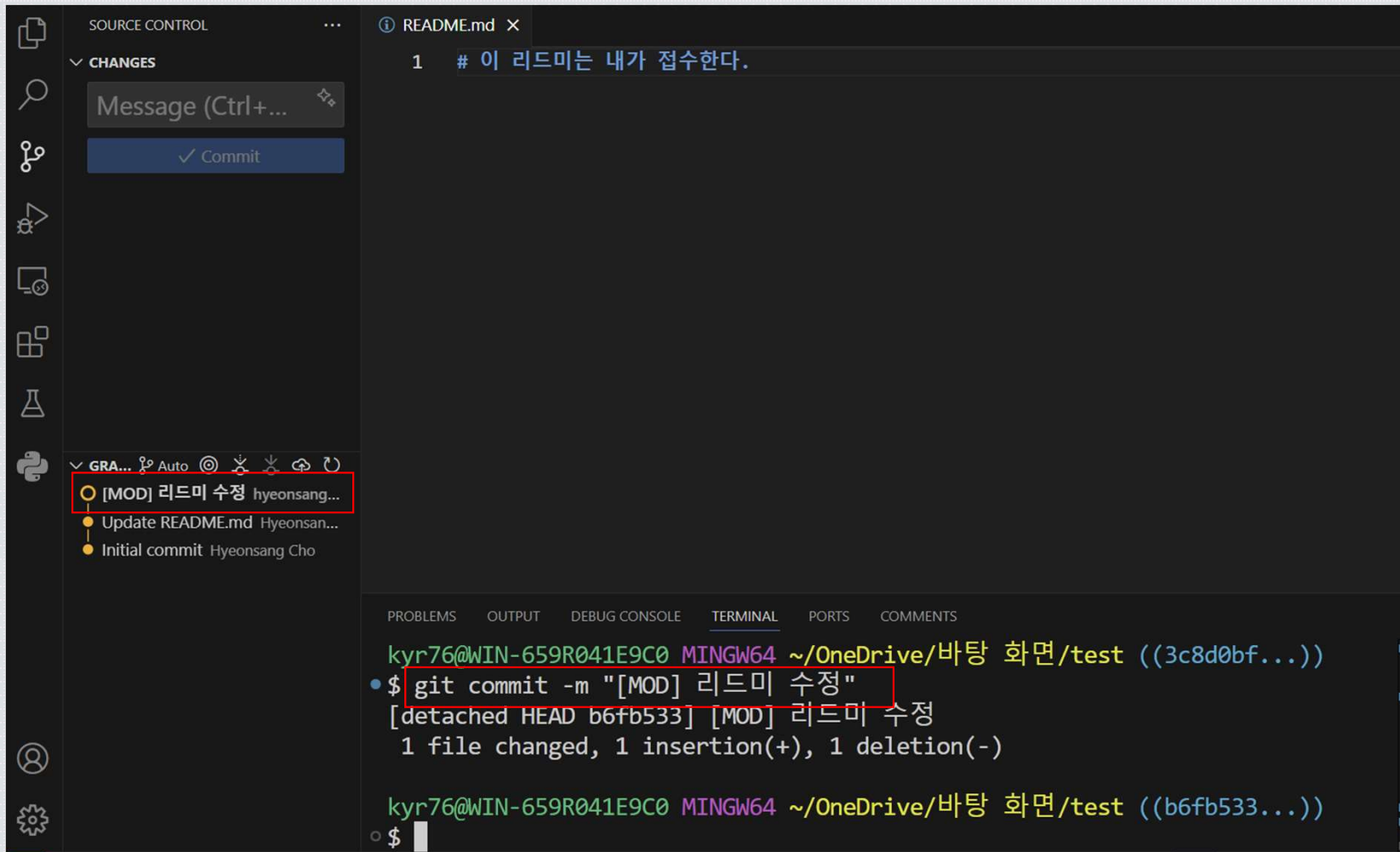
```
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/test ((3c8d0bf...))  
$ git add README.md
```



Source Control 페이지에서  
스테이징 상태를 확인할 수 있어요!

## GitHub 파일 업로드 하기

스테이징이 되었다면 커밋을 해야 해요.  
`git commit -m "메시지"`

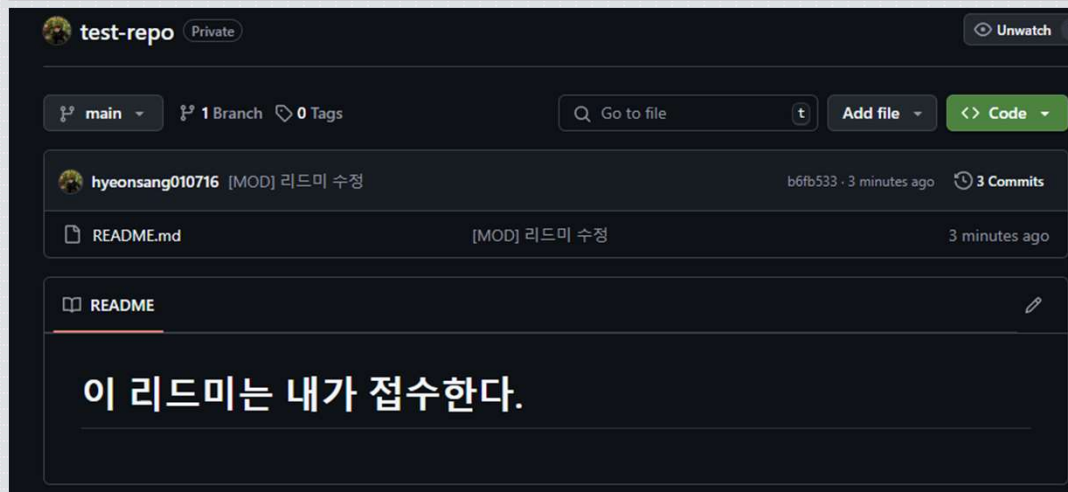




## GitHub 파일 업로드 하기

이제 마지막으로 푸시를 하면 끝!  
`git push`

```
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/test ((b6fb533...))
$ git push origin HEAD:main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 317 bytes | 317.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/hyeonsang010716/test-repo.git
 3c8d0bf..b6fb533 HEAD -> main
```



`git push origin HEAD:main` 으로도 가능

만약 처음 브랜치를 만들고, 푸시할 때는 아래 명령어 필요  
`git push --set-upstream origin main`



## 1주차 실습 과제

1. 깃헙 아이디로 폴더 만들기
2. 만든 폴더 안에 README.md 만들기
3. README.md에 자신의 이름 적기
4. main에 올리기

### [주의할 점]

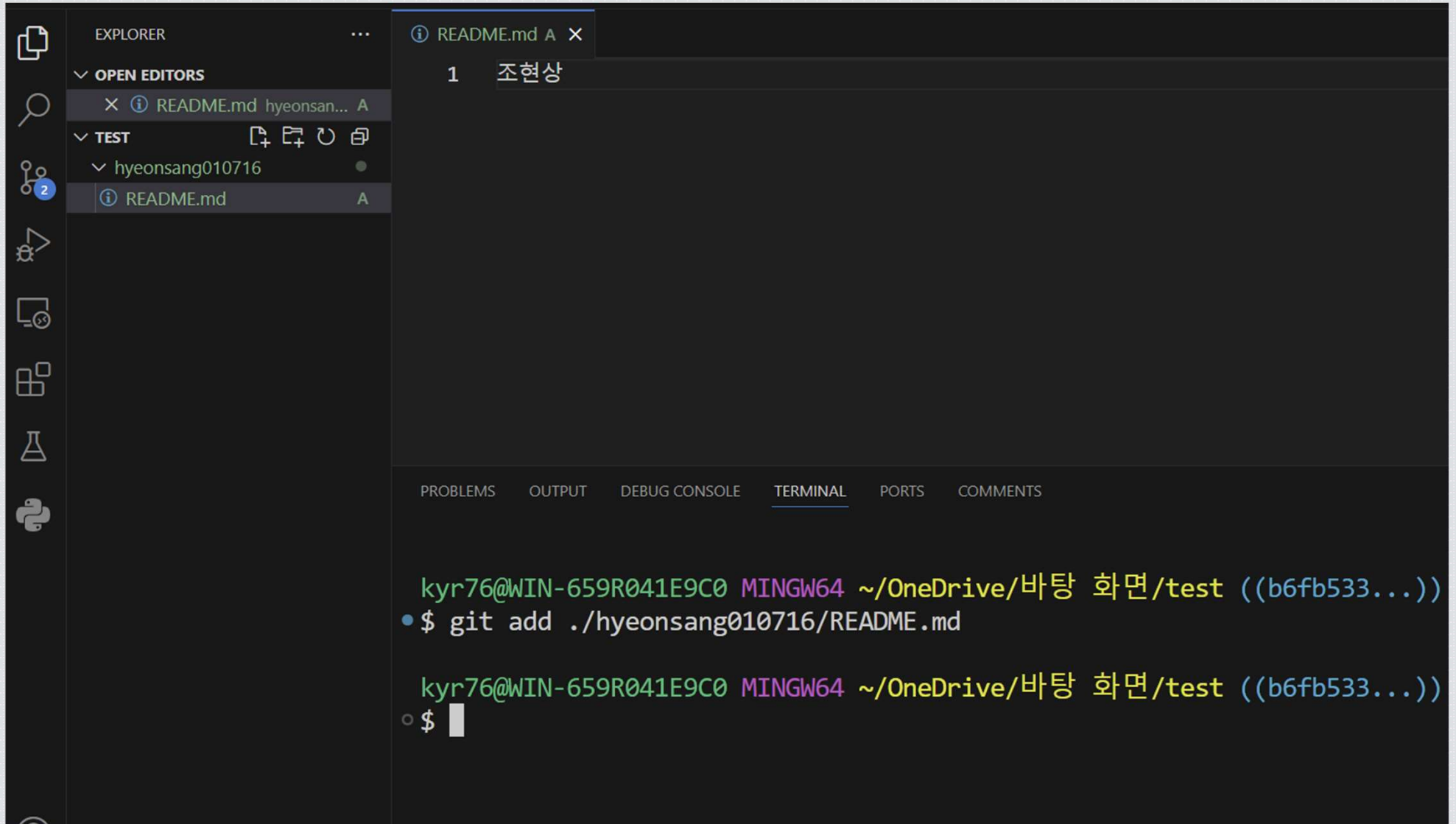
아직 깃헙 충돌을 배우지 않았기 때문에  
push 를 하기 전, git pull origin main 을 하고 푸쉬 하기

Commit은 [FEAT] 깃헙 아이디 README 생성으로 통일하기!  
EX)

“[FEAT] hyeonsang010716 README 생성”

**3월 30일 23:59 까지**

## 1주차 실습 과제

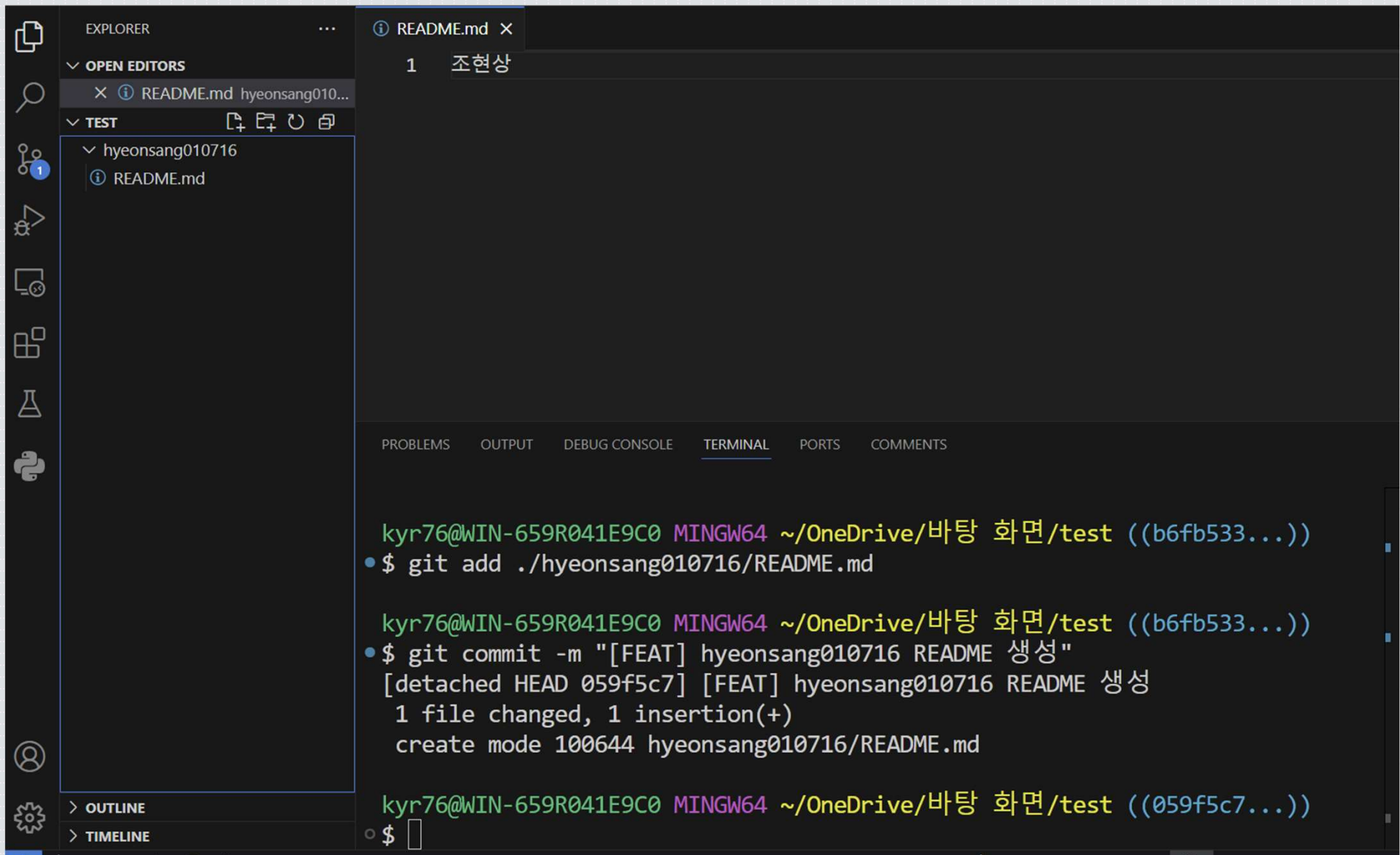


The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar shows a project named 'TEST' with a subdirectory 'hyeonsang010716' containing a 'README.md' file. The main editor area displays the 'README.md' file with the text '1 조현상'. At the bottom, the TERMINAL panel is active, showing a Windows command prompt session. The prompt is 'kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/test ((b6fb533...))'. The user has entered the command '\$ git add ./hyeonsang010716/README.md' and is waiting for a response.

```
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/test ((b6fb533...))
• $ git add ./hyeonsang010716/README.md

kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/test ((b6fb533...))
◦ $
```

## 1주차 실습 과제



The screenshot shows the Visual Studio Code interface. On the left, the Explorer view displays a file tree for a repository named 'hyeonsang010716', containing a 'README.md' file. The main editor area shows the 'README.md' file with the text '1 조현상'. At the bottom, the Terminal panel is active, showing a series of git commands and their output in a Windows command prompt environment.

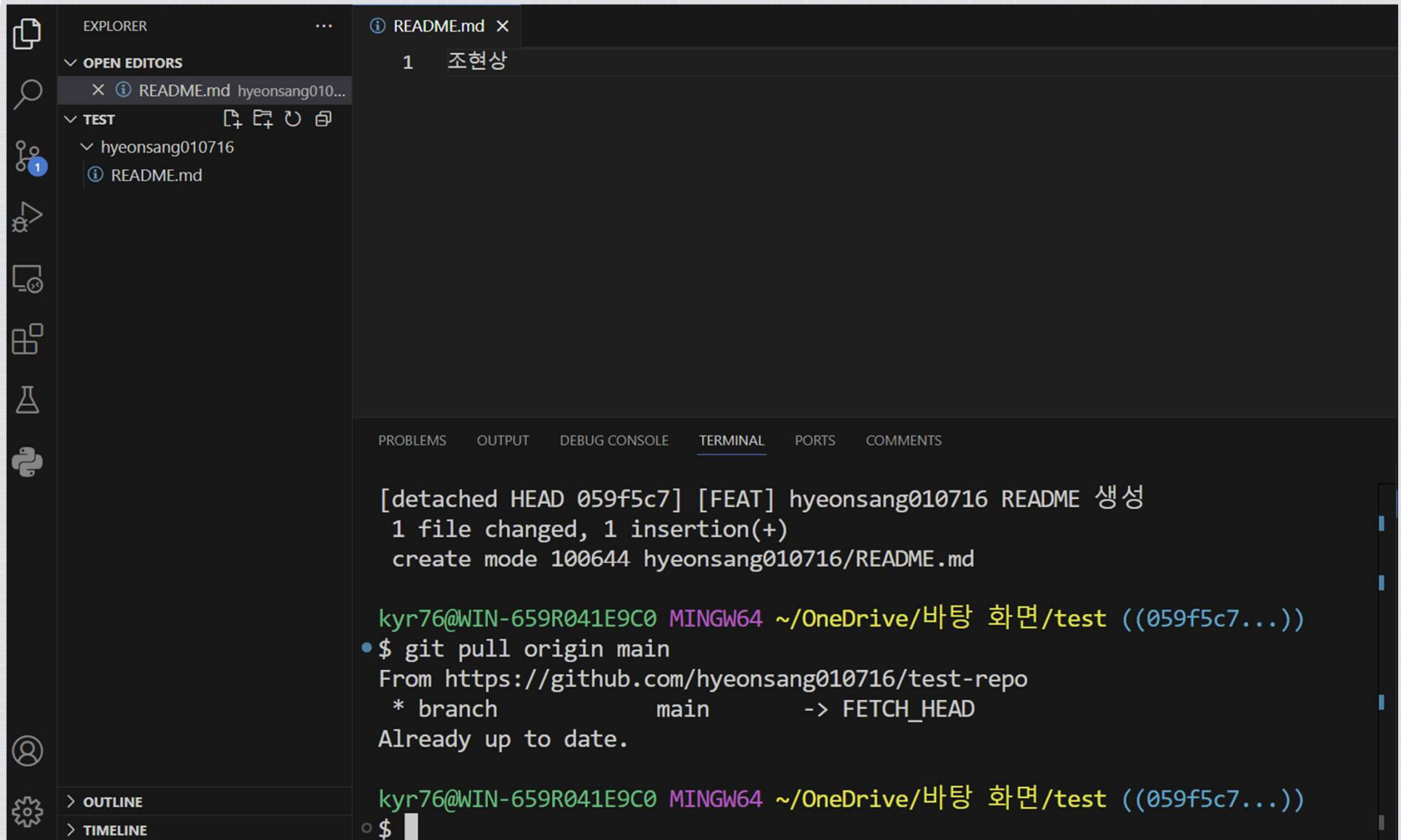
```
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/test ((b6fb533...))
• $ git add ./hyeonsang010716/README.md

kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/test ((b6fb533...))
• $ git commit -m "[FEAT] hyeonsang010716 README 생성"
[detached HEAD 059f5c7] [FEAT] hyeonsang010716 README 생성
1 file changed, 1 insertion(+)
create mode 100644 hyeonsang010716/README.md

kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/test ((059f5c7...))
○ $
```



## 1주차 실습 과제



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar shows a project structure with a folder named 'TEST' containing a file 'README.md'. The Open Editors sidebar shows the 'README.md' file is open. The main editor area displays the content of 'README.md', which is '1 조현상'. Below the editor, the TERMINAL panel is active, showing the output of a git command and the prompt for the next command.

EXPLORER

OPEN EDITORS

TEST

hyeonsang010716

README.md

README.md

1 조현상

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

```
[detached HEAD 059f5c7] [FEAT] hyeonsang010716 README 생성
1 file changed, 1 insertion(+)
create mode 100644 hyeonsang010716/README.md

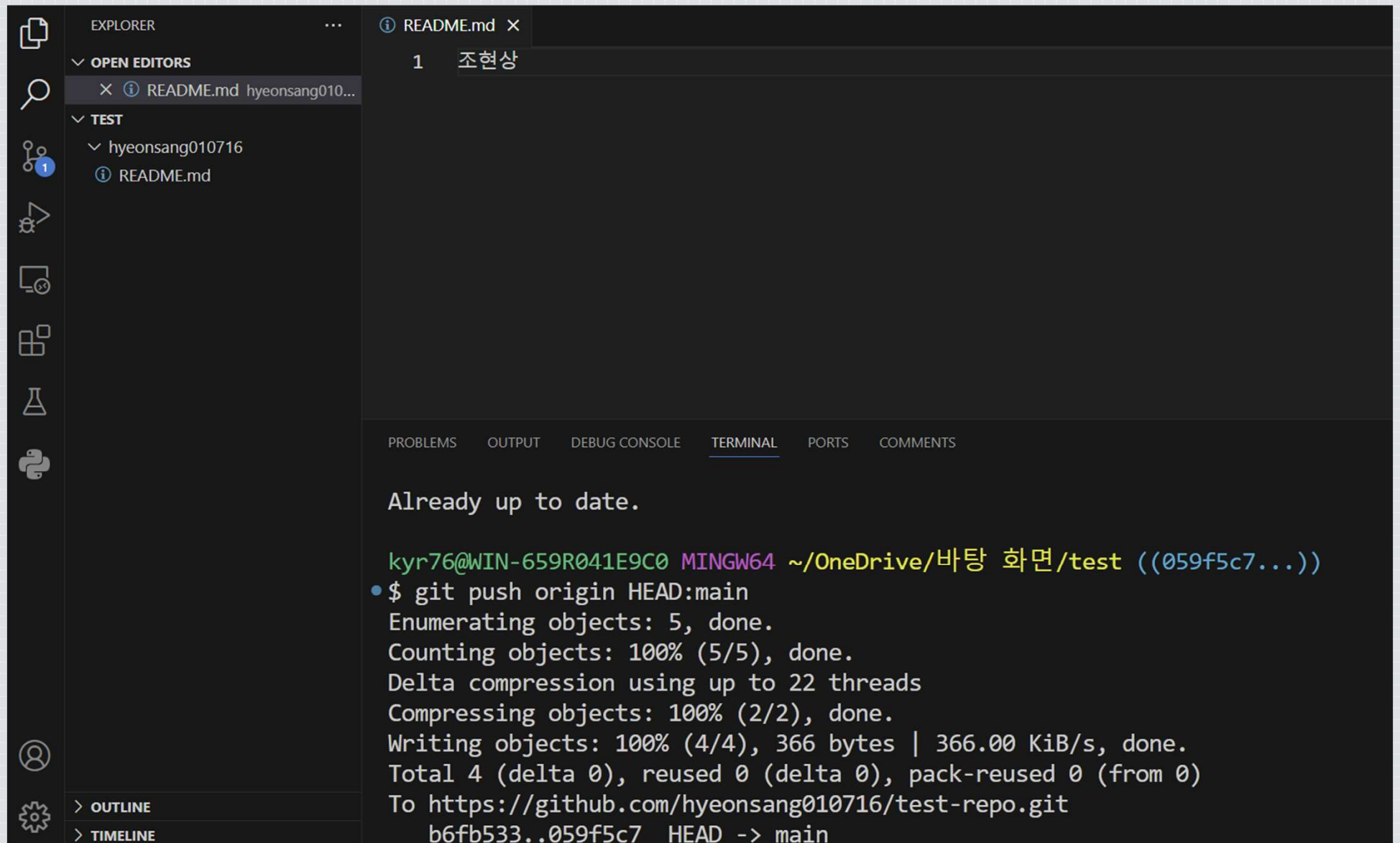
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/test ((059f5c7...))
• $ git pull origin main
From https://github.com/hyeonsang010716/test-repo
* branch          main          -> FETCH_HEAD
Already up to date.

kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/test ((059f5c7...))
$
```

> OUTLINE

> TIMELINE

## 1주차 실습 과제



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar shows a project named 'hyeonsang010716' with a 'README.md' file. The main editor area shows the 'README.md' file with the text '1 조현상'. At the bottom, the Terminal panel is active, displaying the output of a 'git push' command.

EXPLORER

OPEN EDITORS

README.md hyeonsang010...

TEST

hyeonsang010716

README.md

1 조현상

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

Already up to date.

```
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/test ((059f5c7...))
• $ git push origin HEAD:main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 22 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 366 bytes | 366.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/hyeonsang010716/test-repo.git
b6fb533..059f5c7 HEAD -> main
```

## 1주차 실습 과제

hyeonsang010716 / test-repo

<> Code Issues Pull requests Actions Projects Security Insights Settings

Files

main

Go to file

hyeonsang010716

README.md

README.md

test-repo / hyeonsang010716

hyeonsang010716 [FEAT] hyeonsang010716 README 생성

Name	Last commit message
..	
README.md	[FEAT] hyeonsang010716 README 생성

README.md

조현상



언제든지 자유롭게 물어보기!

<> Code **Issues** Pull requests Actions Projects Wiki Security Insights Settings

is:issue state:open

☐ Open 0 ☐ Closed 0

Author ▾ Labels ▾ Projects ▾ Milestones ▾ Assignees ▾ Types ▾ Newest ▾

**No results**  
Try adjusting your search filters.

질문이 생기면 주저 말고 GitHub Issue에 남겨주세요 ;)