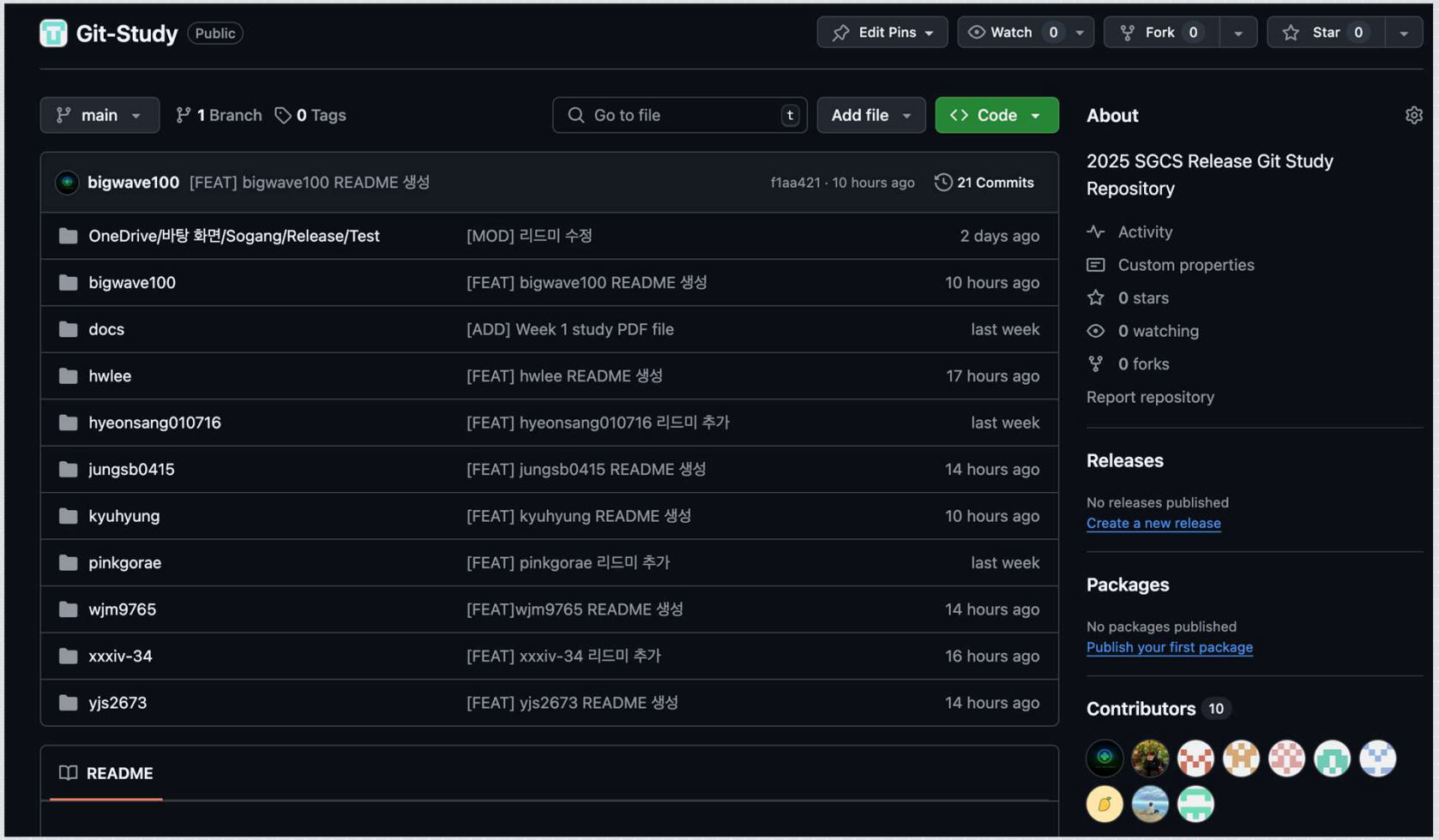
2025 깃헙 스터디

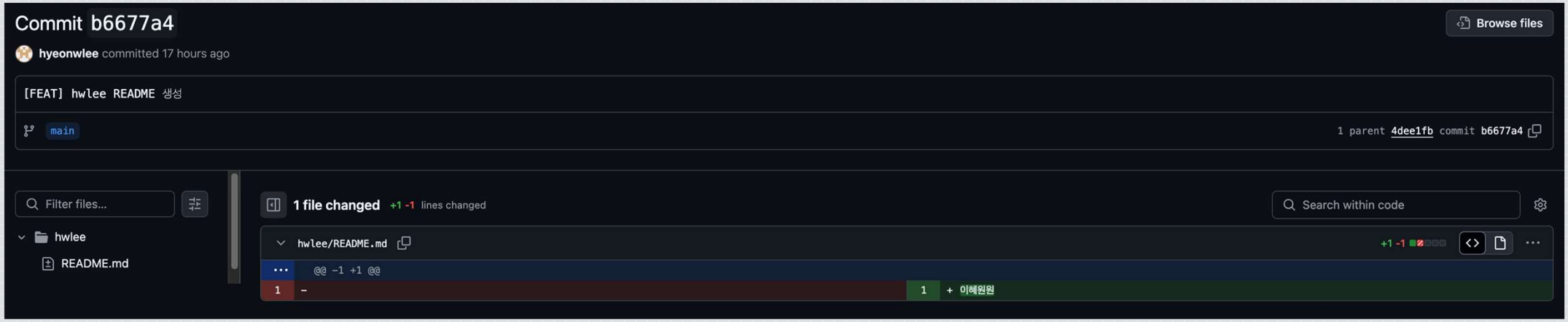
서강대학교 Release 학회

수고하셨습니다~!



모두 1주차 실습하시느라 고생하셨어요~!

피드백



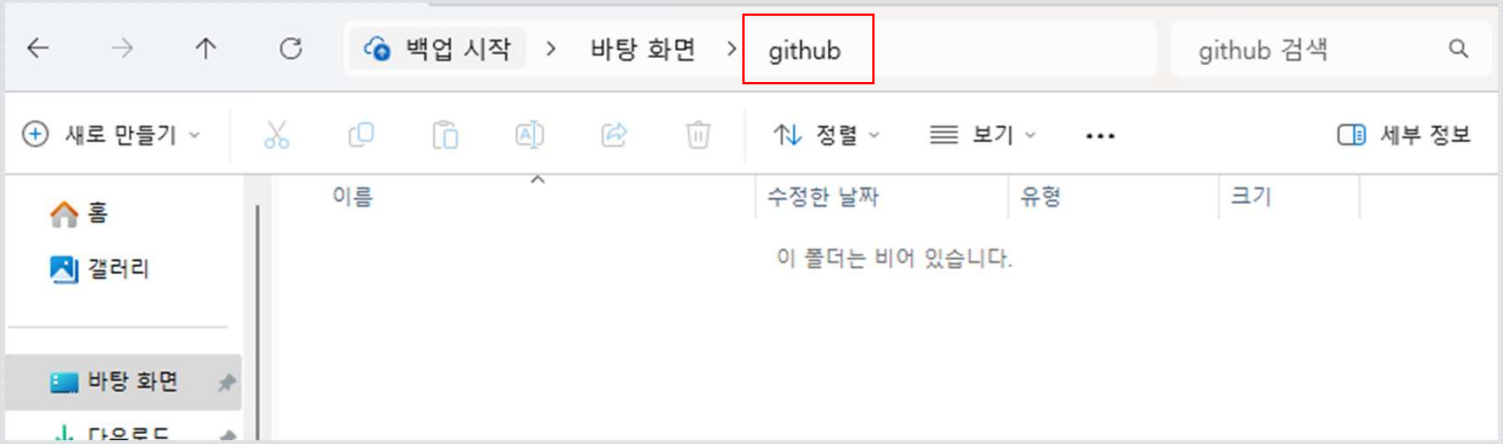
윈도우 한글 두 번 입력 에러는 아래 방법으로 해결할 수 있어요!
-> https://blog.naver.com/yiyouya/223470446020

피드백

OneDrive/바탕 화면/Sogang/Release/Test [MOD] 리드미 수정 2 days ago

어떤 일이 있었던 거죠!!😂

깃을 사용할 때는 리프 디렉토리를 만들고 해당 위치에서 터미널을 사용해야 해요!



피드백

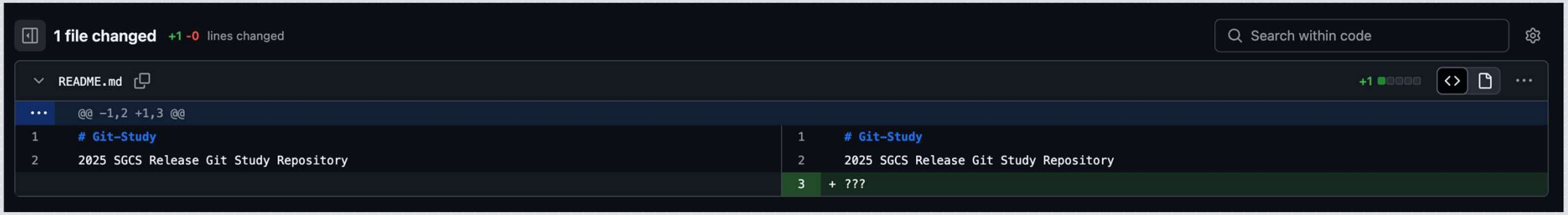
```
hyeonsang@johyeonsang-ui-MacBookPro git-study % git remote add origin https://github.com/SGCS-Release-Git-Project/Git-Study.git
error: remote origin already exists.
hyeonsang@johyeonsang-ui-MacBookPro git-study %
```

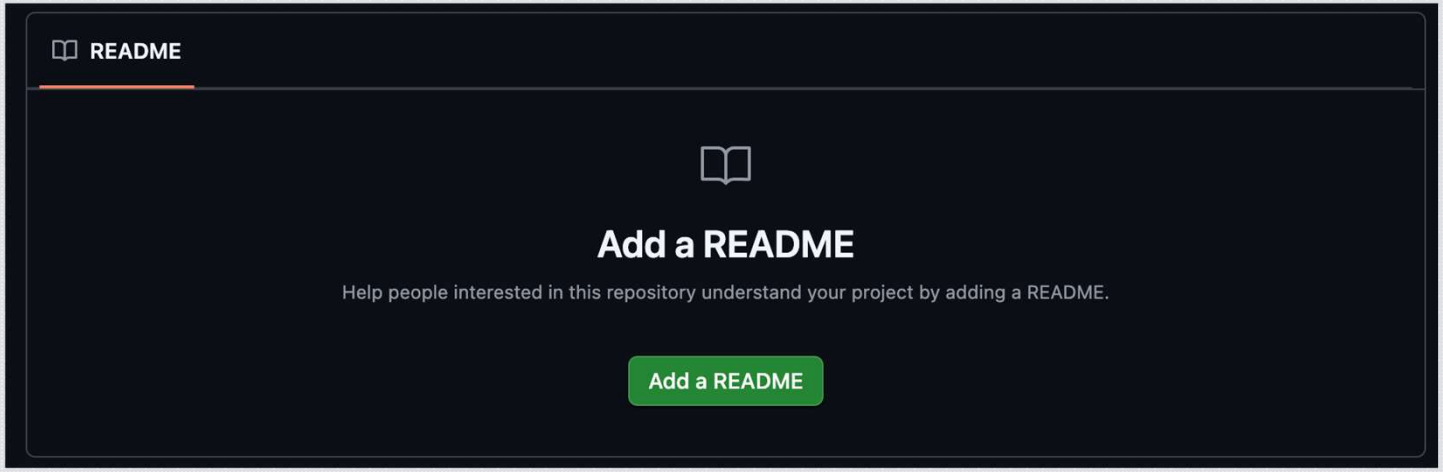
만약 git remote add origin 잘못된 URL 입력으로 꼬였을 경우!

git remote rm origin # origin 삭제함
또는
rm –rf .git # git을 삭제함

방법으로 해결할 수 있음!

문제점





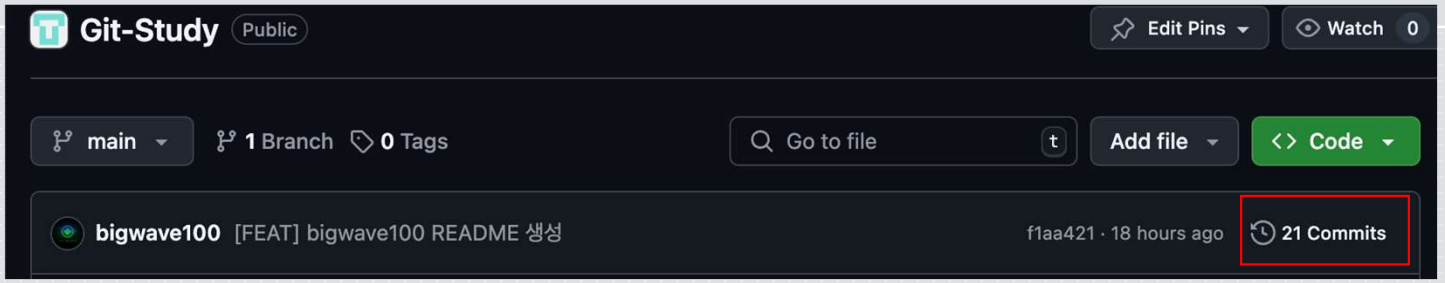
건드리면 안 되는 파일 수정

이외에도..

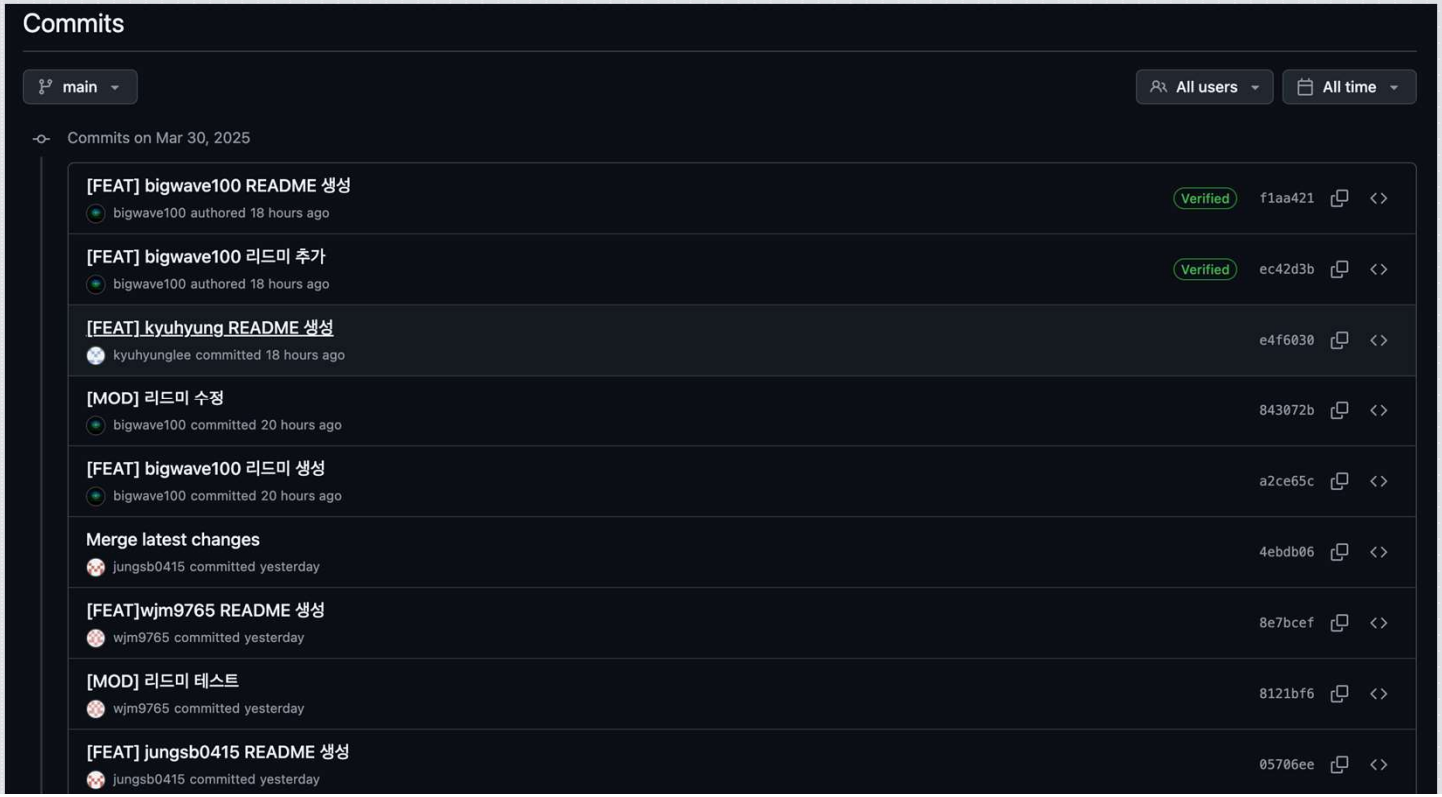
- 왜 git pull origin main 을 사용해야 했을까!!

컨트롤 + s 를 안 해서, 에러를 겪으신 분들은 아래 VSCode 자동 저장 참고하세요!
(https://hianna.tistory.com/354)

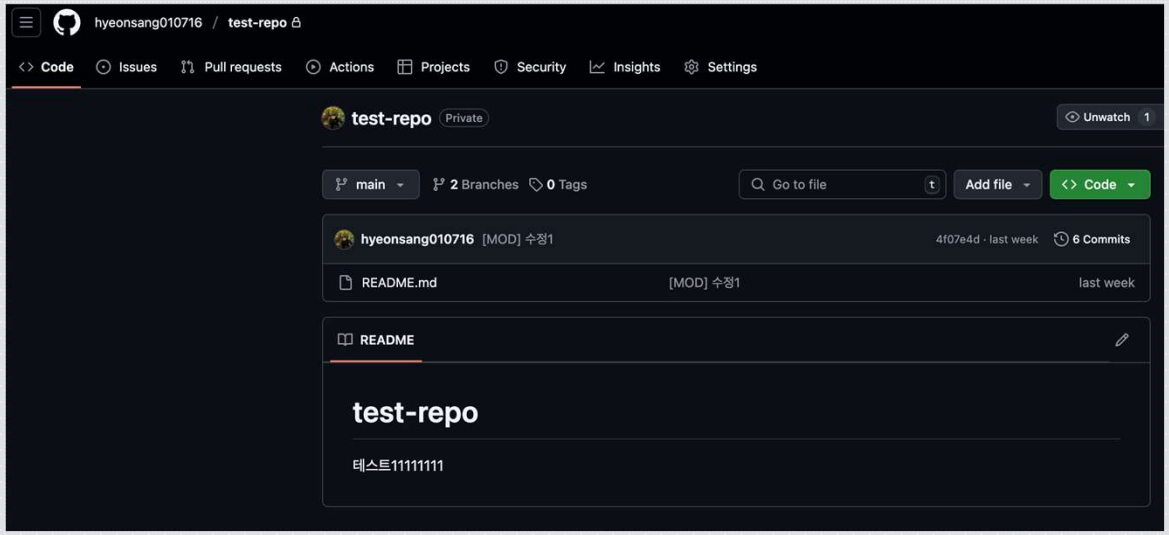
깃헙 히스토리



깃헙에는 히스토리가 존재하고 매우 중요합니다!

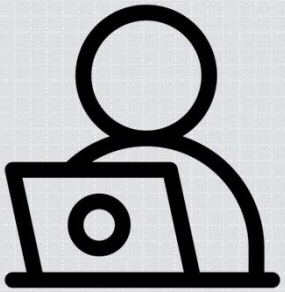


깃헙 히스토리 예제

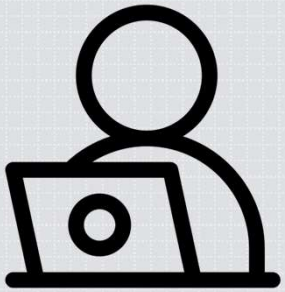


히스토리

<시나리오>
A개발자와 B개발자가 리모트 main 브런치를 로컬에 가져와서 개발할 예정!

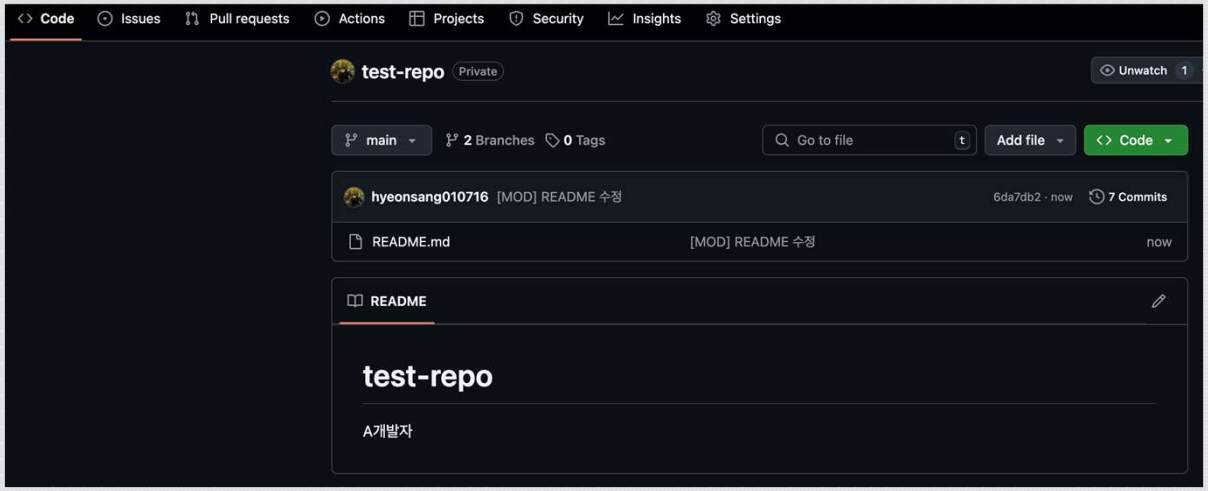


A개발자



B개발자

깃헙 히스토리 예제

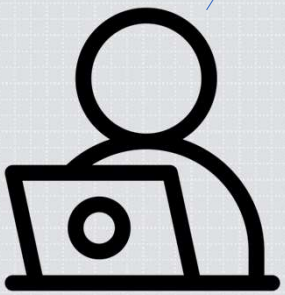


히스토리

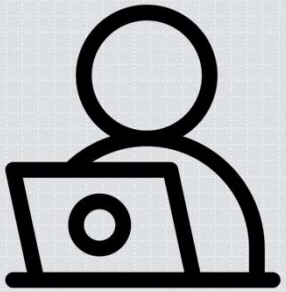
[MOD] 수정1 B

[MOD] README 수정 A

<시나리오>
A 개발자가 리모트를 수정해서 푸쉬 함.

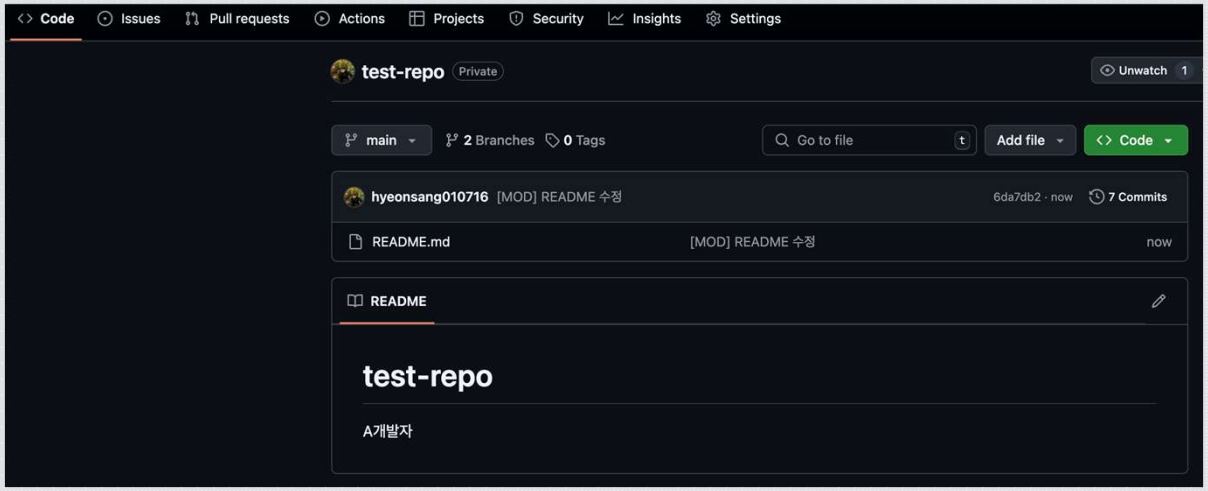


A개발자



B개발자

깃헙 히스토리 예제

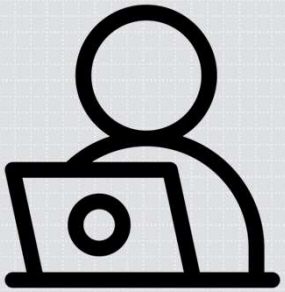


히스토리

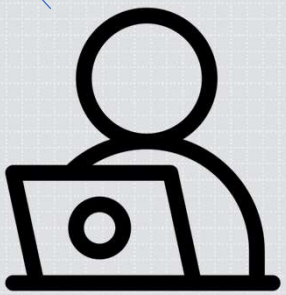
[MOD] 수정1 B

[MOD] README 수정 A

<시나리오>
그 다음 B개발자가 개발을 하고, 푸쉬를 할 거임

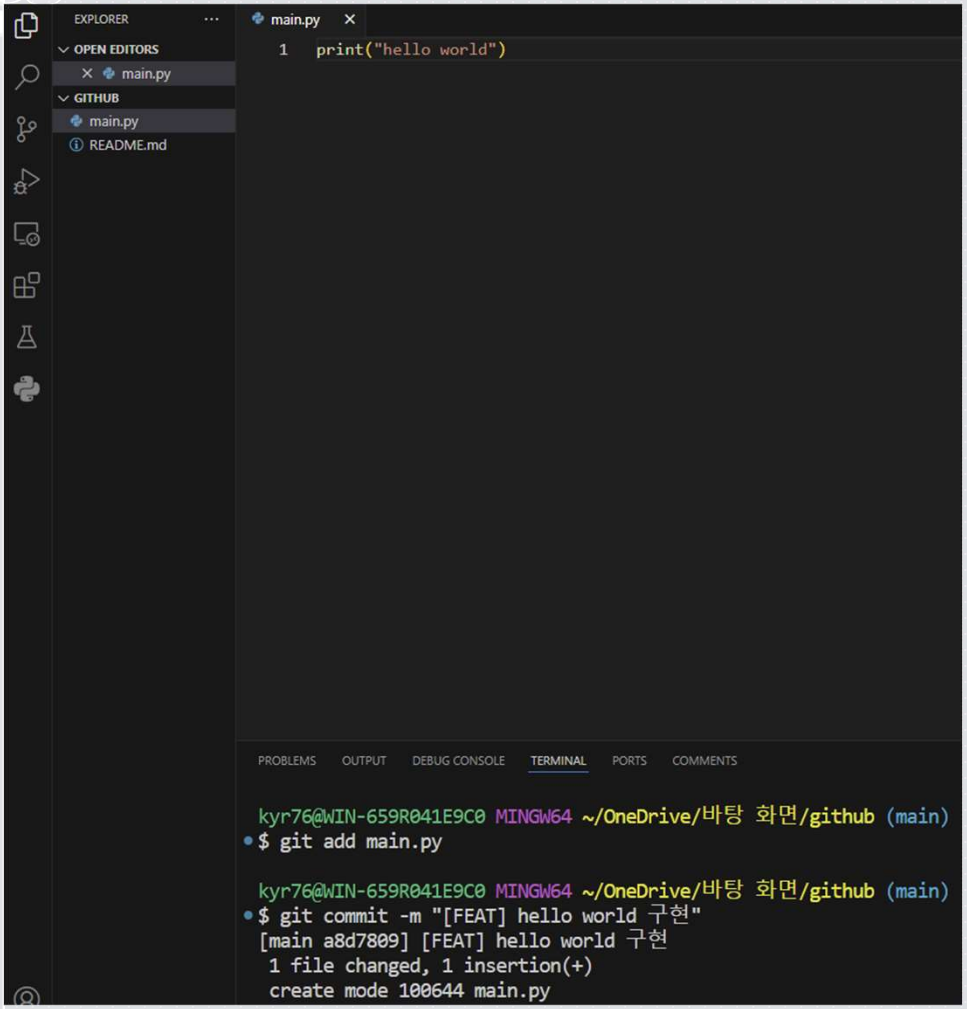


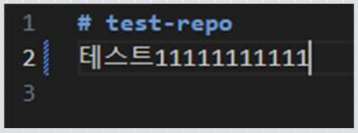
A개발자

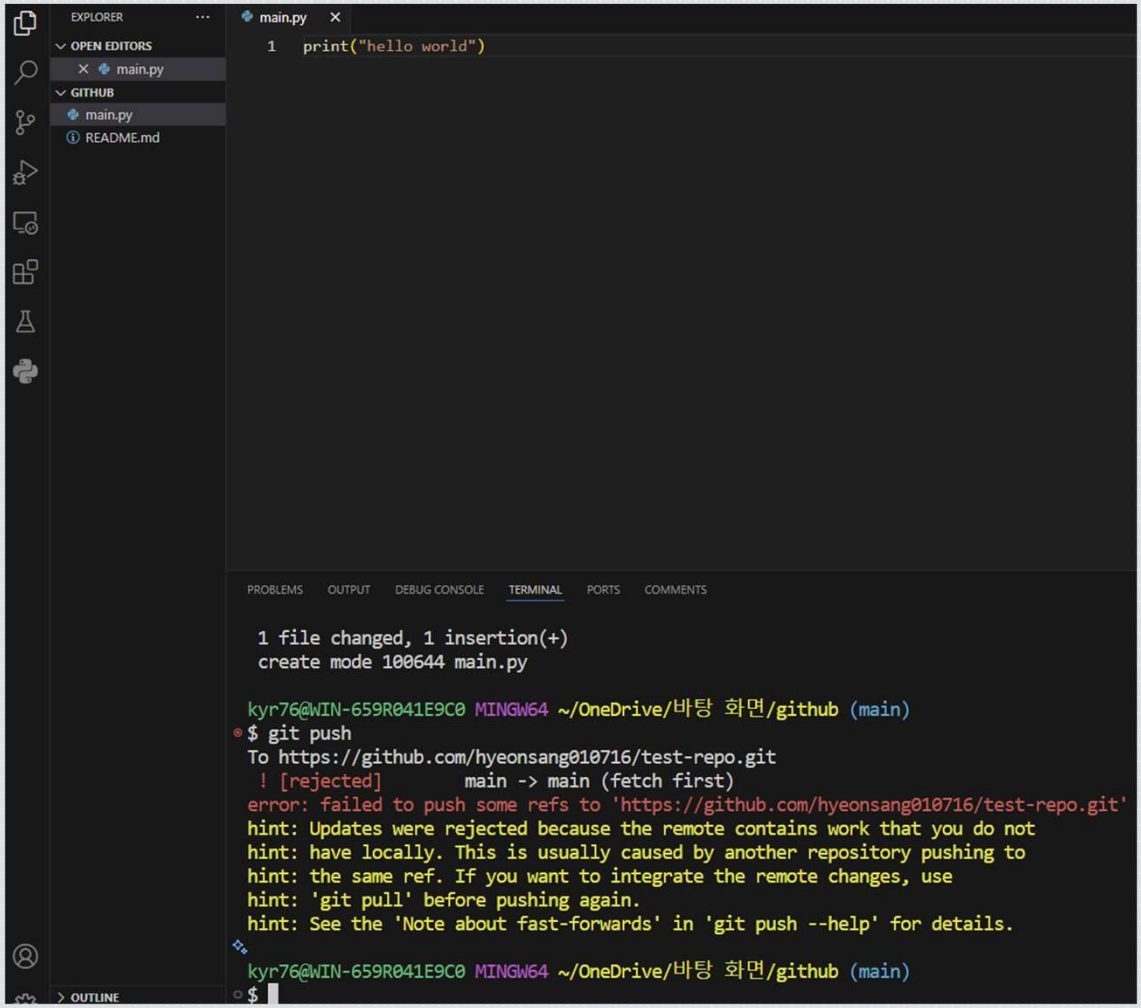


B개발자

깃헙 히스토리 예제

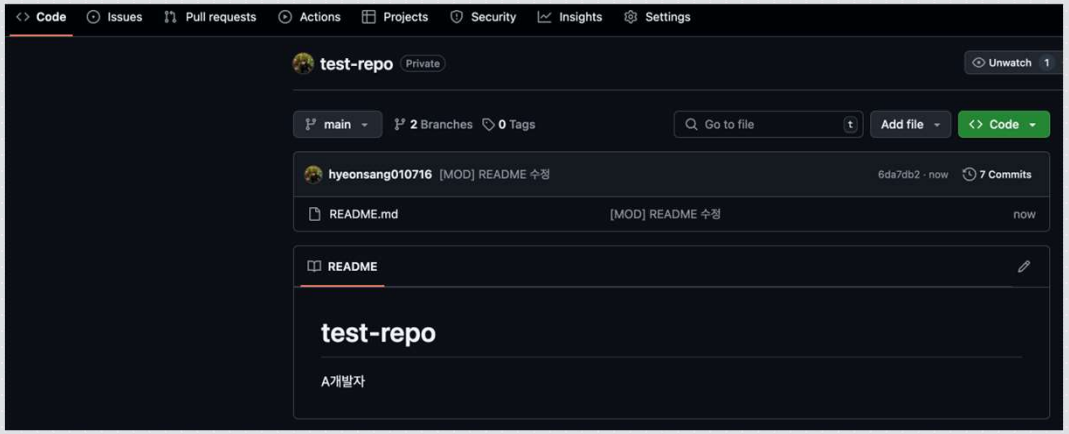






하지만 충돌 발생!

깃헙 히스토리 예제



히스토리

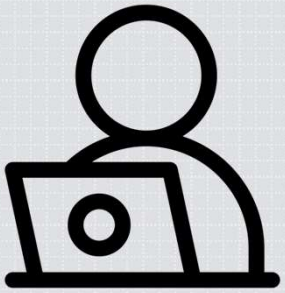
[MOD] 수정1 B

[MOD] README 수정 A

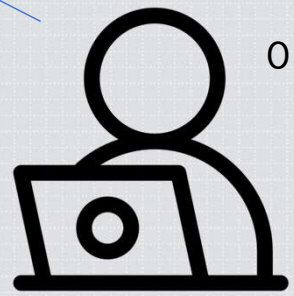
충돌

<시나리오>
B 개발자가 리모트를 수정해서 푸쉬 함.
하지만 충돌!

이유) 최신 히스토리가 아니라서!

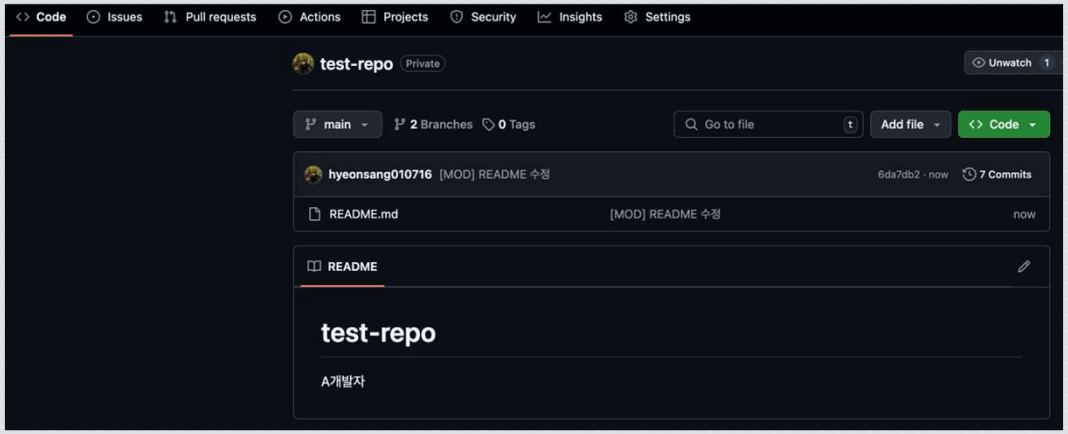


A개발자



B개발자

깃헙 히스토리 예제



히스토리

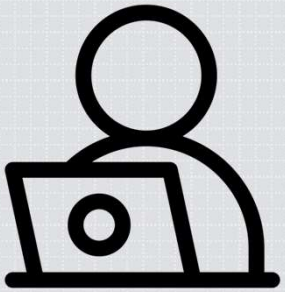
[MOD] 수정1 B

[MOD] README 수정 A

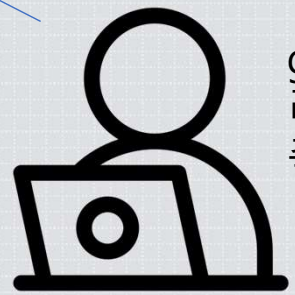
충돌

충돌을 해결하기 위해선, 히스토리 재정리가 필요함!

히스토리 충돌이 일어날 때, 필요한 명령어 : **git pull**



A개발자

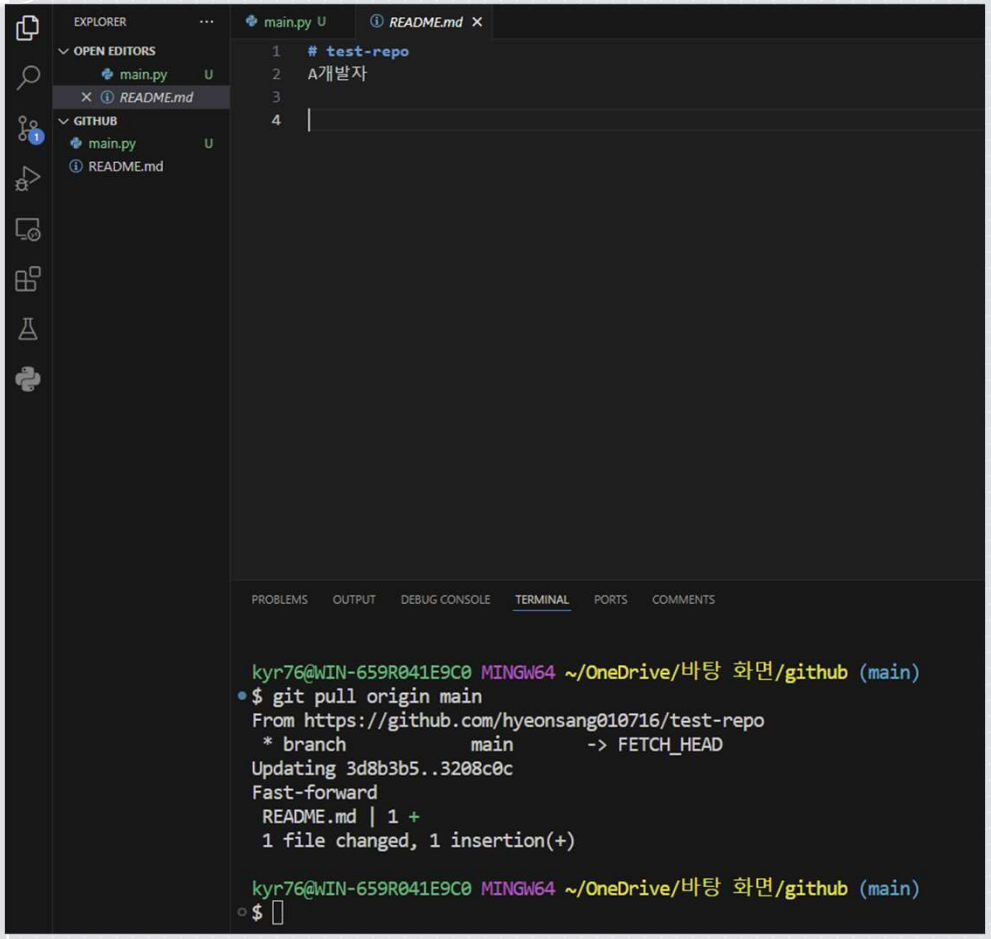


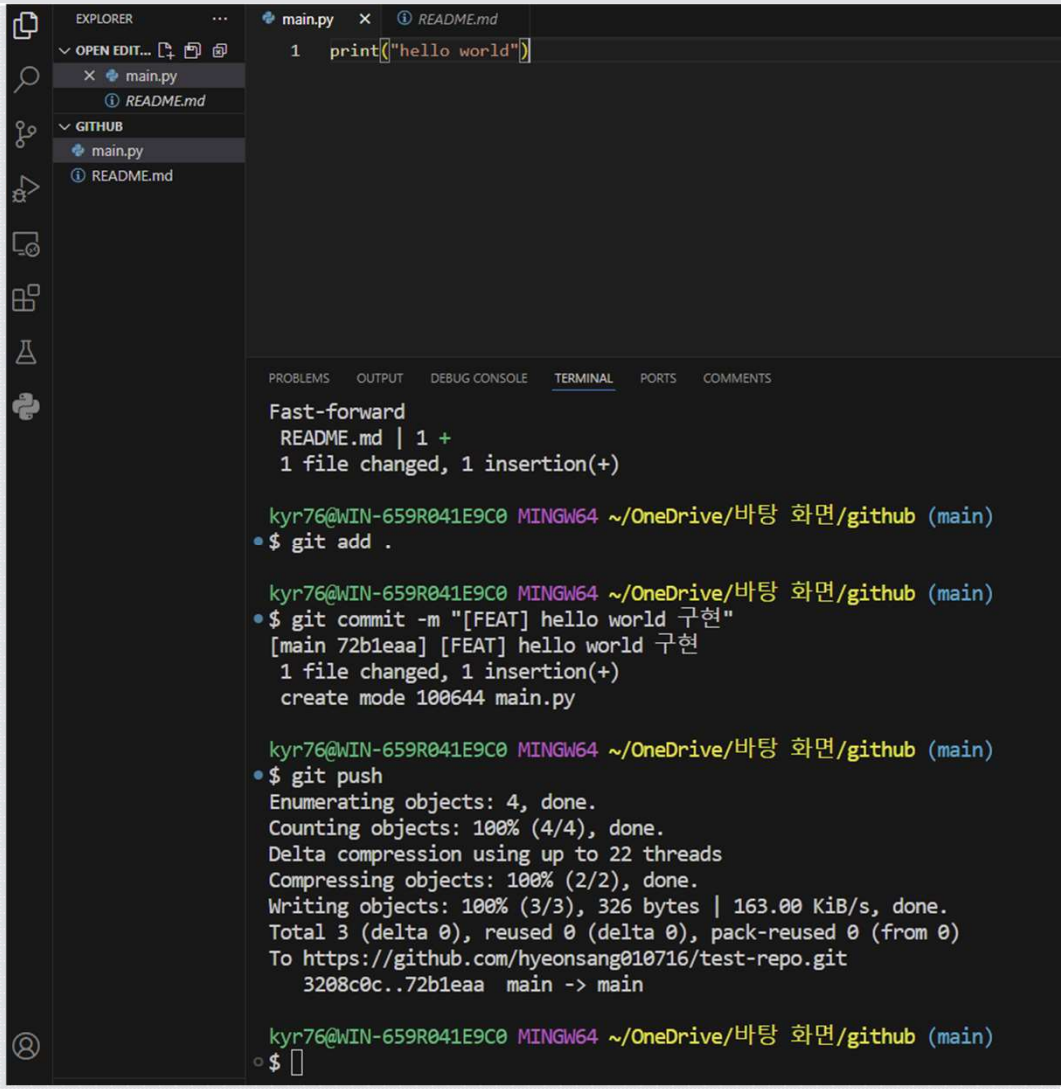
B개발자

git pull origin main
리모트 main 브런치에 있는 변경 사항을 가져옴
즉 업데이트 된 히스토리를 가져 온다.

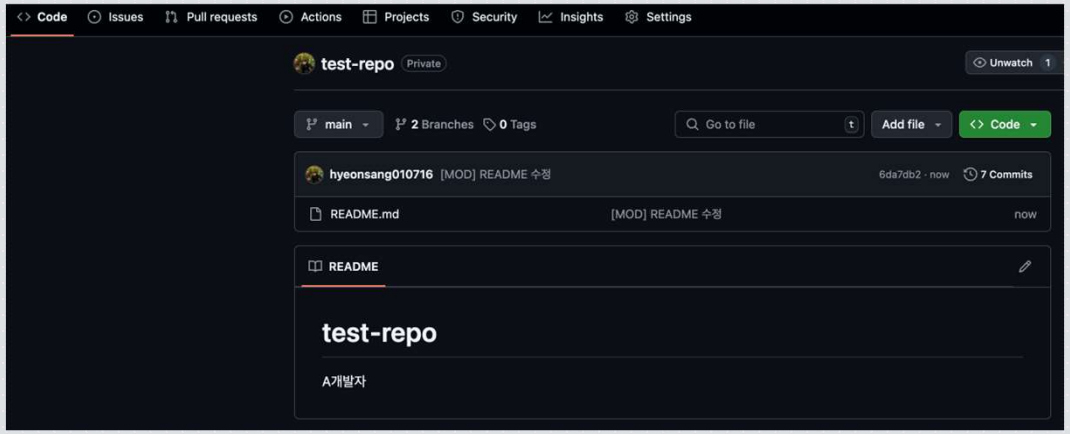
만약 B개발자가 push를 하기 전, pull을 먼저 했다면 어땠을까?

깃헙 히스토리 예제


EXPLORER
OPEN EDITORS
main.py U
README.md
GITHUB
main.py U
README.md
main.py U
README.md
1 # test-repo
2 A개발자
3
4
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/github (main)
$ git pull origin main
From https://github.com/hyeonsang010716/test-repo
* branch main -> FETCH_HEAD
Updating 3d8b3b5..3208c0c
Fast-forward
README.md | 1 +
1 file changed, 1 insertion(+)
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/github (main)
$


EXPLORER
OPEN EDIT...
main.py
README.md
GITHUB
main.py
README.md
main.py
README.md
1 print("hello world")
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
Fast-forward
README.md | 1 +
1 file changed, 1 insertion(+)
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/github (main)
$ git add .
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/github (main)
$ git commit -m "[FEAT] hello world 구현"
[main 72b1eaa] [FEAT] hello world 구현
1 file changed, 1 insertion(+)
create mode 100644 main.py
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/github (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 22 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 326 bytes | 163.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/hyeonsang010716/test-repo.git
3208c0c..72b1eaa main -> main
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/github (main)
$

깃헙 히스토리 예제



히스토리

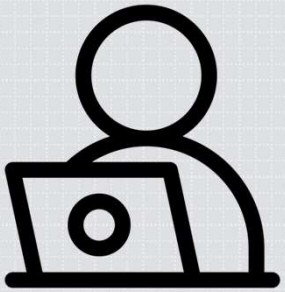
[MOD] 수정1

[MOD] README 수정

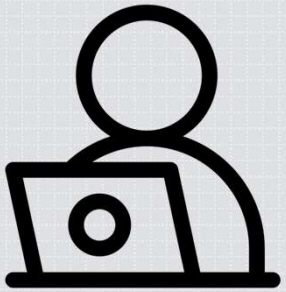
A

B

pull

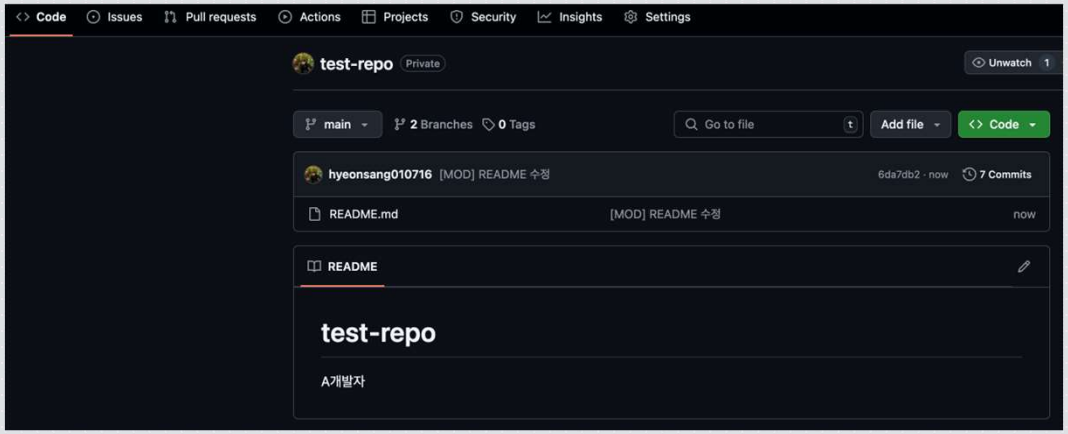


A개발자



B개발자

깃헙 히스토리 예제



히스토리

[MOD] 수정1

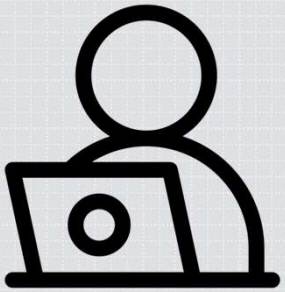
[MOD] README 수정

A

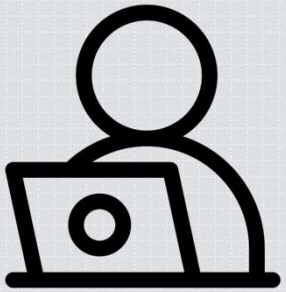
[FEAT] hello ~

B

push

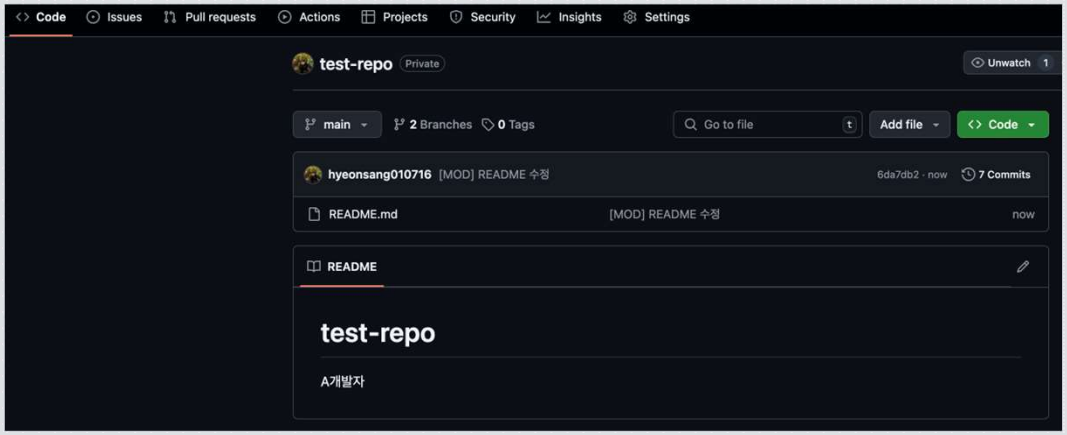


A개발자

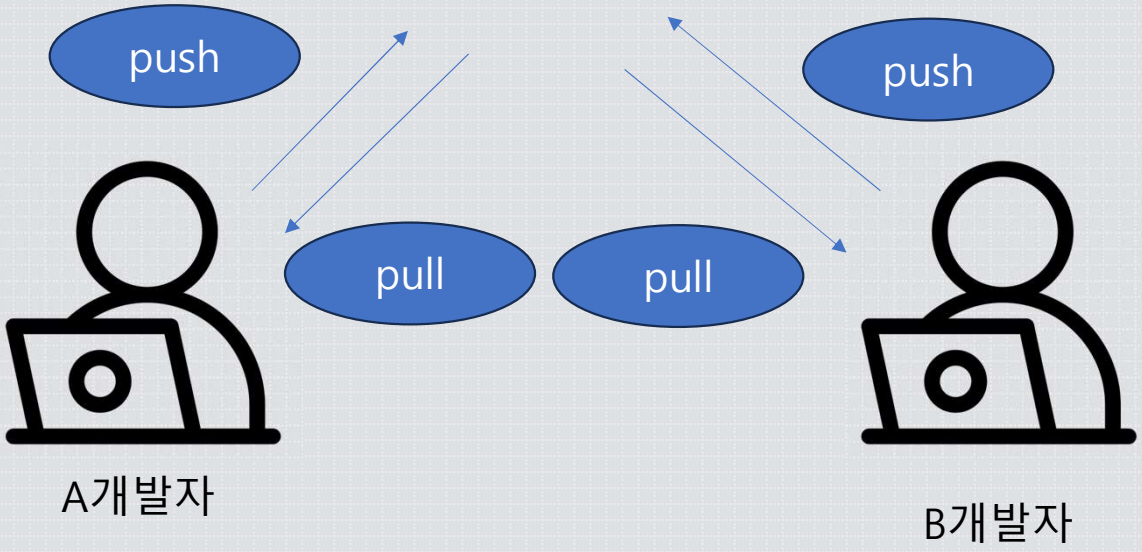


B개발자

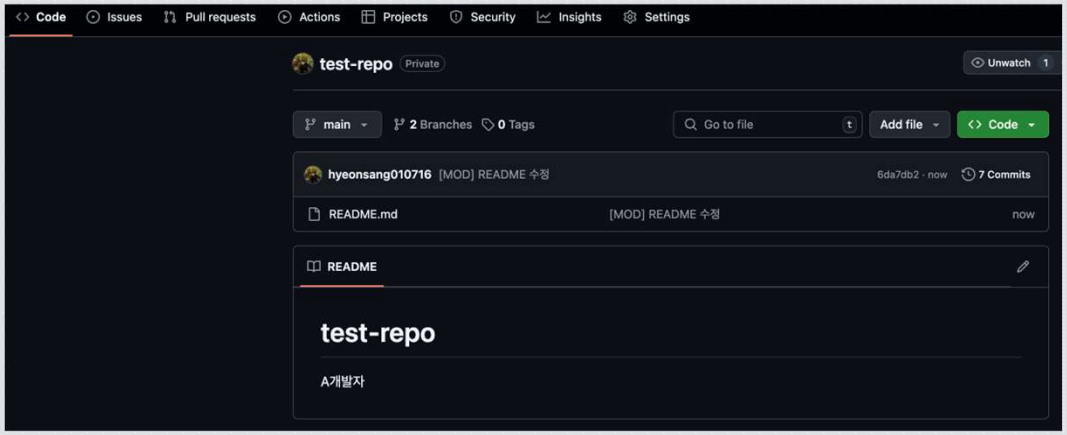
깃헙 히스토리 예제



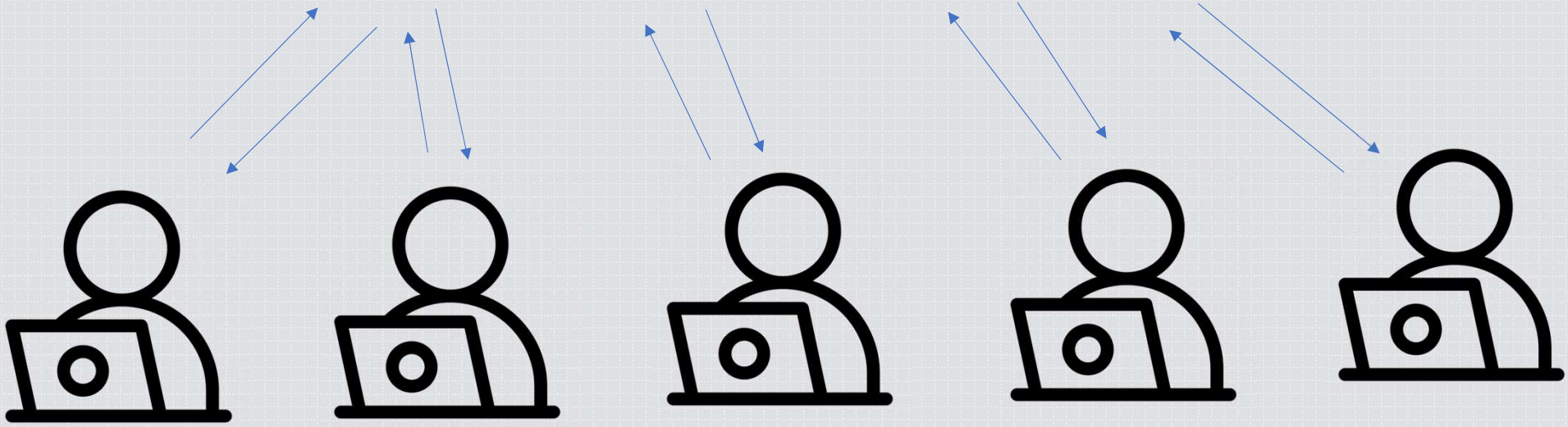
그렇다면 계속 pull, push , pull, push... 해야할까..?



깃헙 히스토리 예제

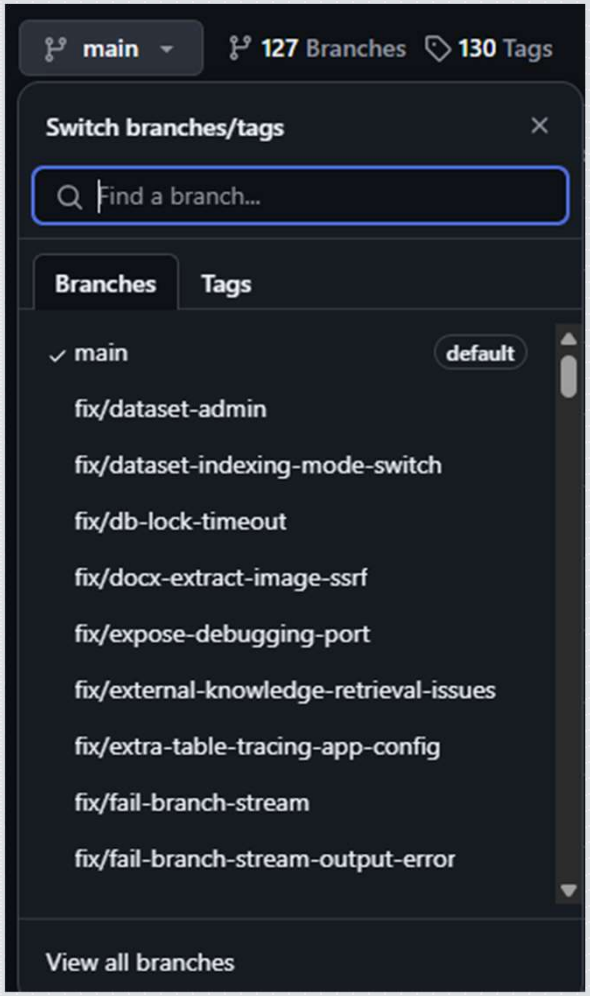


N명이라면..?



깃헙 브런치 예제

해당 문제 때문에, 하나의 브런치로 협업을 하지 않음!

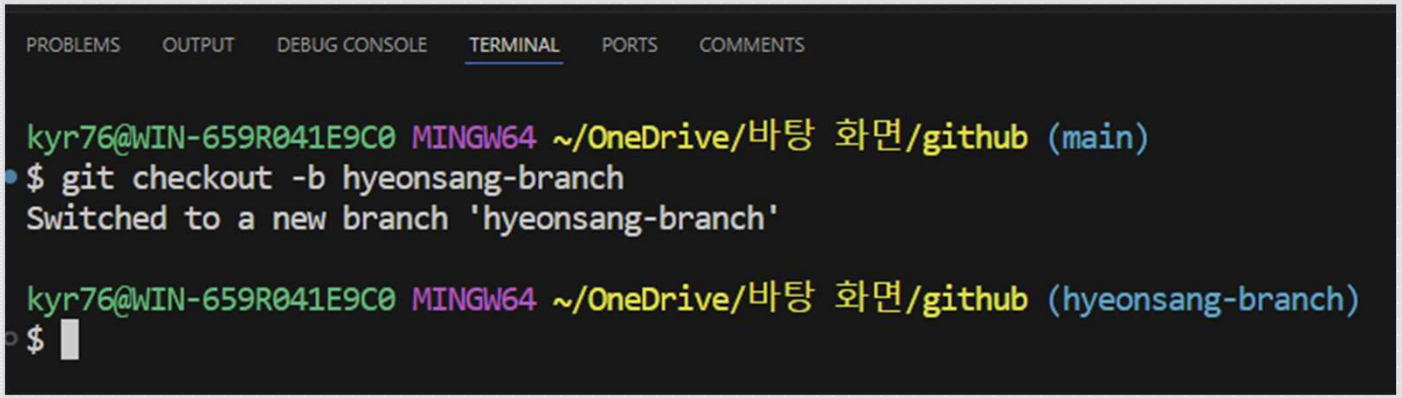


기능마다 브런치를 나누고, 최대한 효율적인 작업을 진행 함.

그러면 왜 여러 브런치로 나눠서 작업을 하면 효과 적일까?

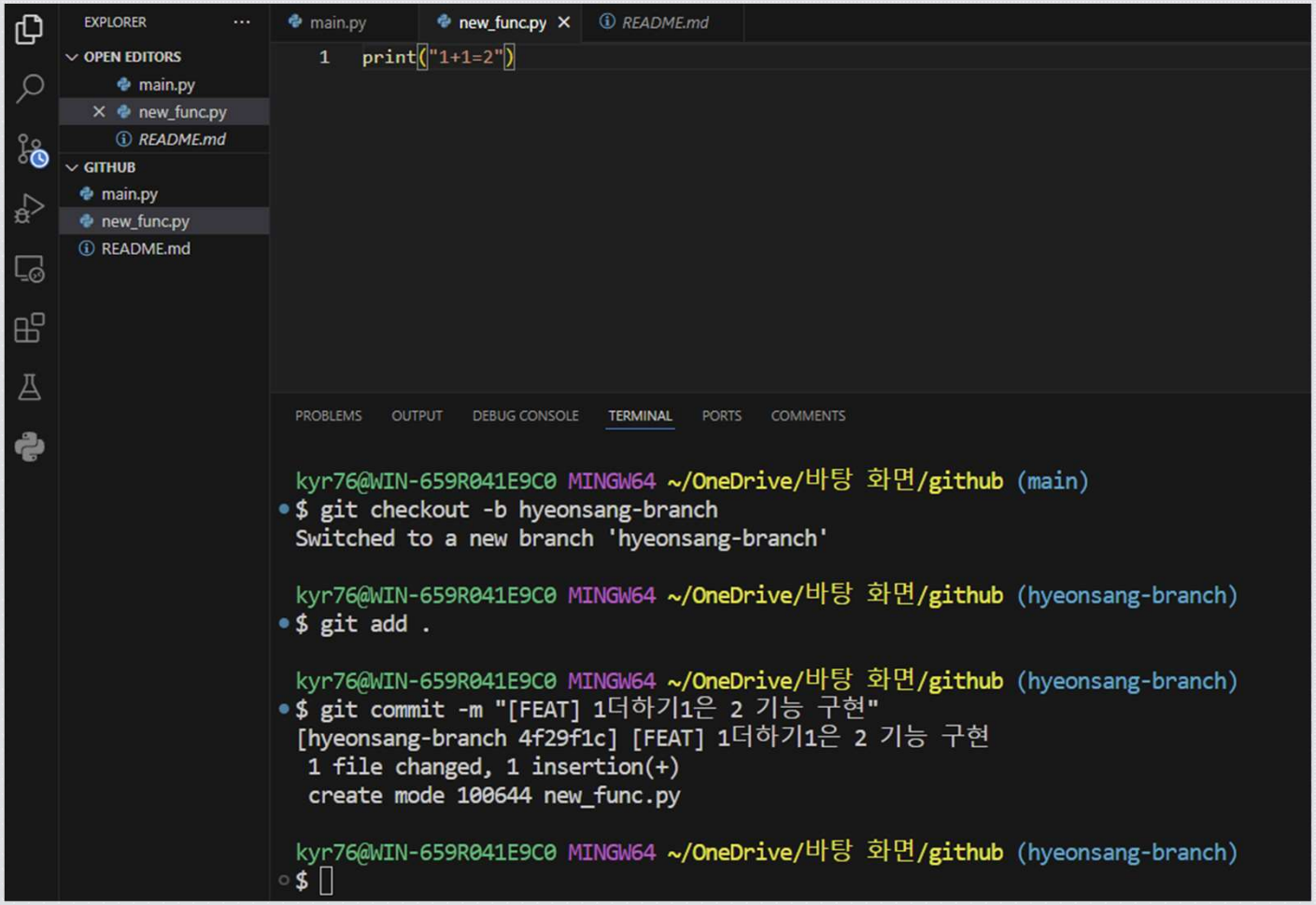
깃헙 브런치 예제

브런치가 효과적인 이유를 보기 전, 브런치를 어떻게 만드는지 알아봅시다.



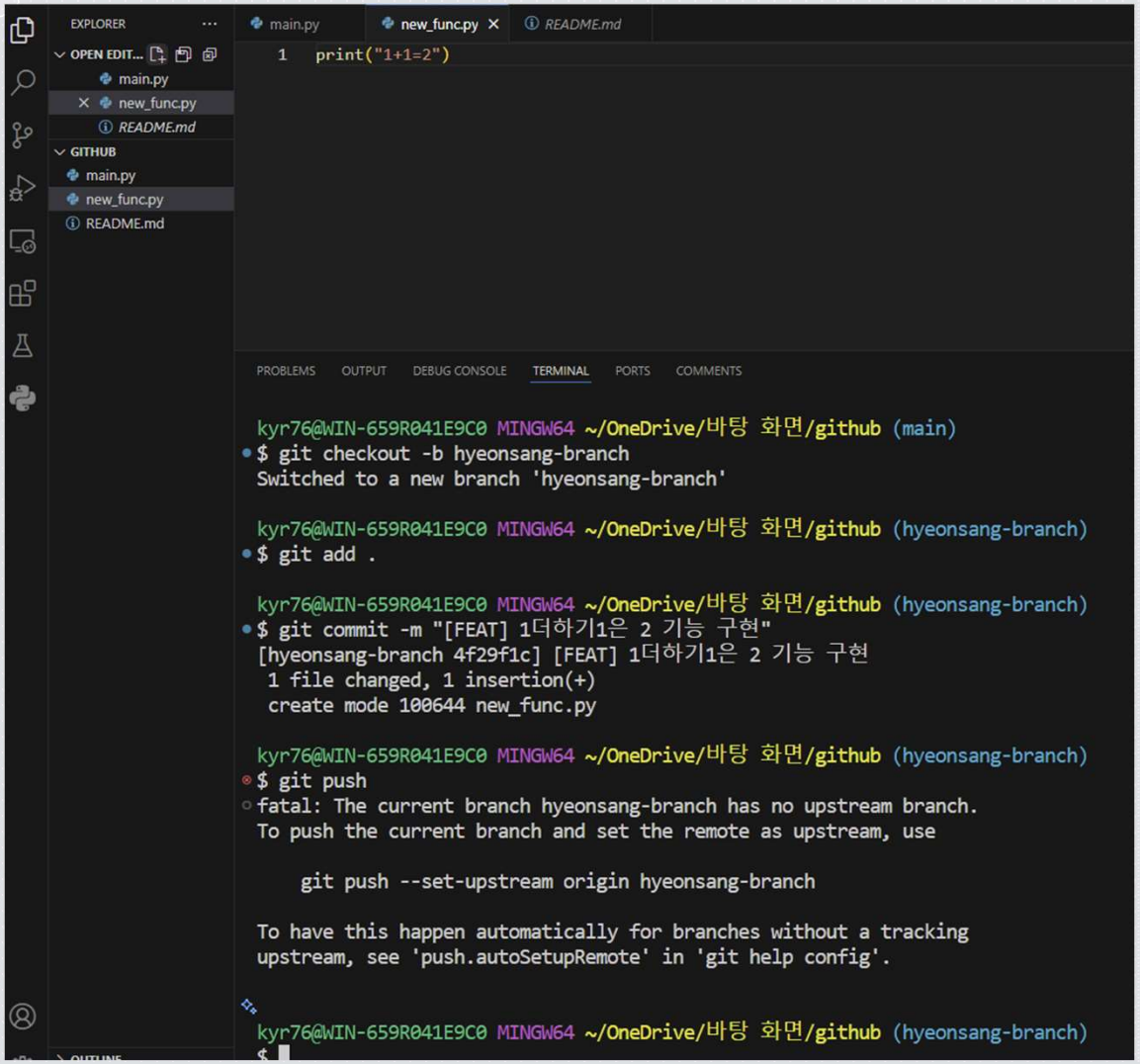
git checkout –b 브런치이름 : 새로운 브런치를 만드는 명령어!
(주의할 점 로컬에만 브런치가 만들어진 것! 리모트에도 만들어졌다고 생각하면 안 됩니다.)

깃헙 브런치 예제



이전과 동일하게, add -> commit 까지 진행합니다.

깃헙 브런치 예제



한 가지 다른 점은
git push를 했을 때
git push --set-upstream origin 브런치이름
이 등장합니다.

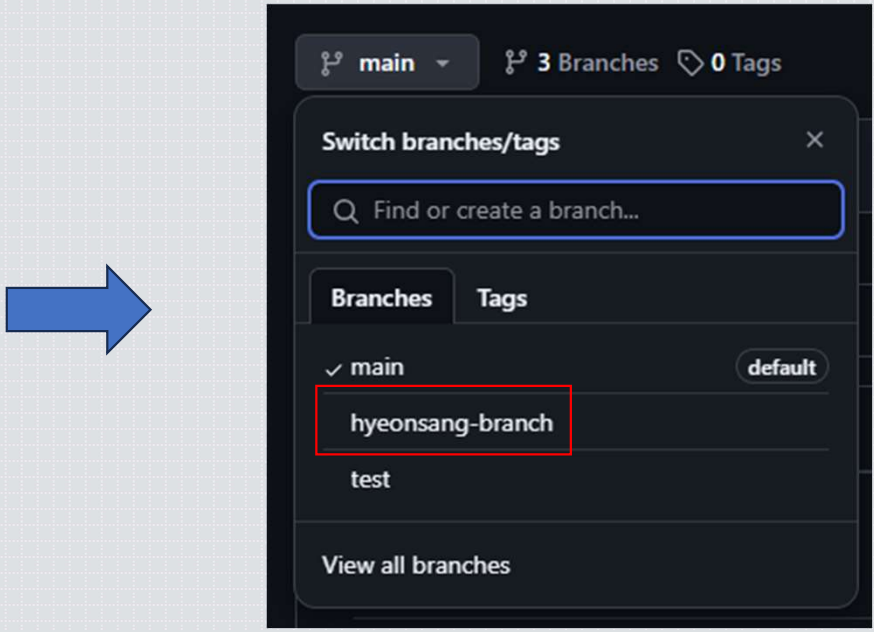
이 명령어는 해당 로컬에서 작업한 브런치를 리모트에 생성하고, 리모트와 로컬을 연결 시킬까요?

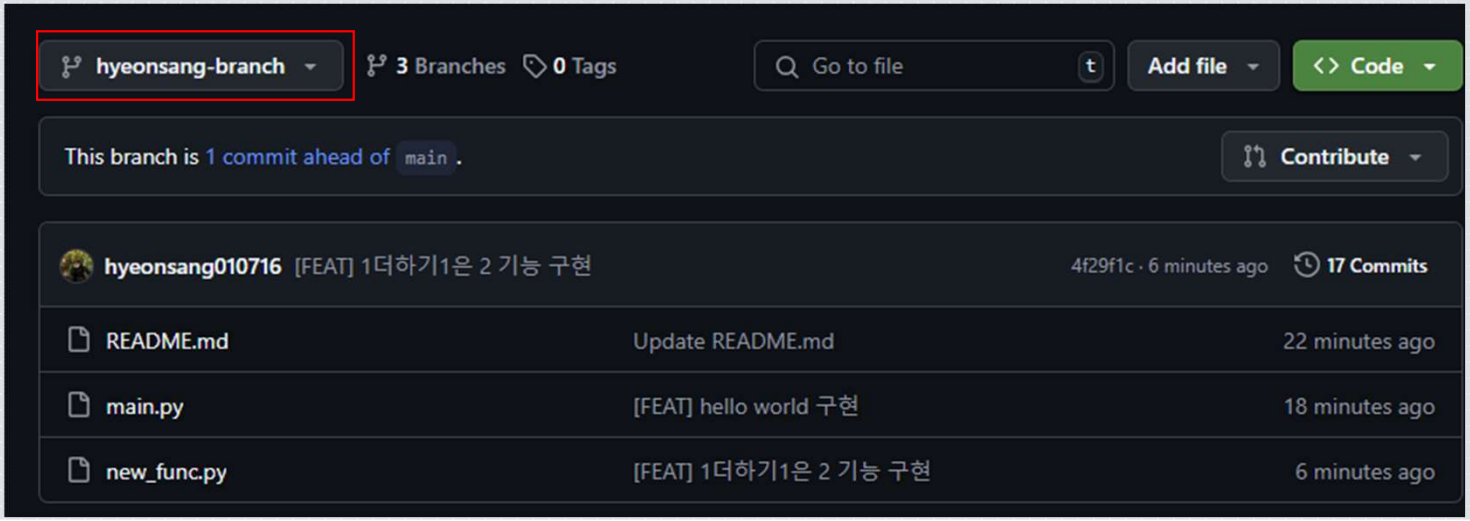
라는 의미입니다.

git push --set-upstream origin 브런치이름
를 한 번 하게 된다면
그 다음부터 git push만 해도 푸쉬 됨!

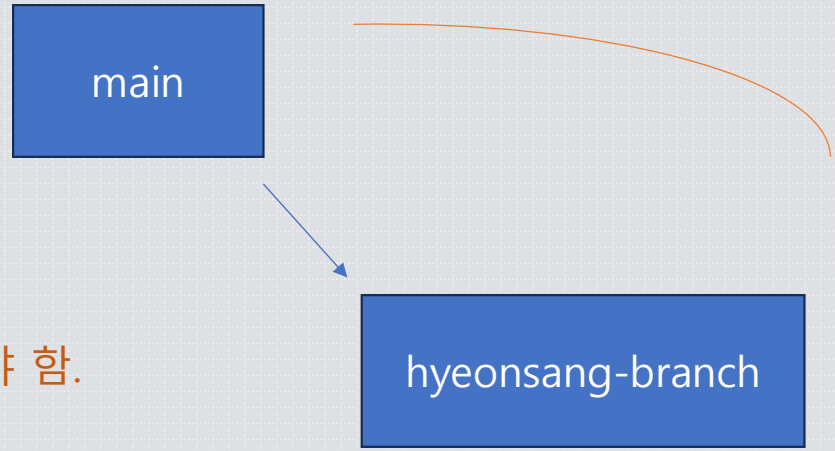
깃헙 브런치 예제

```
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/github (hyeonsang-branch)
$ git push --set-upstream origin hyeonsang-branch
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 22 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 373 bytes | 373.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'hyeonsang-branch' on GitHub by visiting:
remote:      https://github.com/hyeonsang010716/test-repo/pull/new/hyeonsang-branch
remote:
To https://github.com/hyeonsang010716/test-repo.git
 * [new branch]      hyeonsang-branch -> hyeonsang-branch
branch 'hyeonsang-branch' set up to track 'origin/hyeonsang-branch'.
```





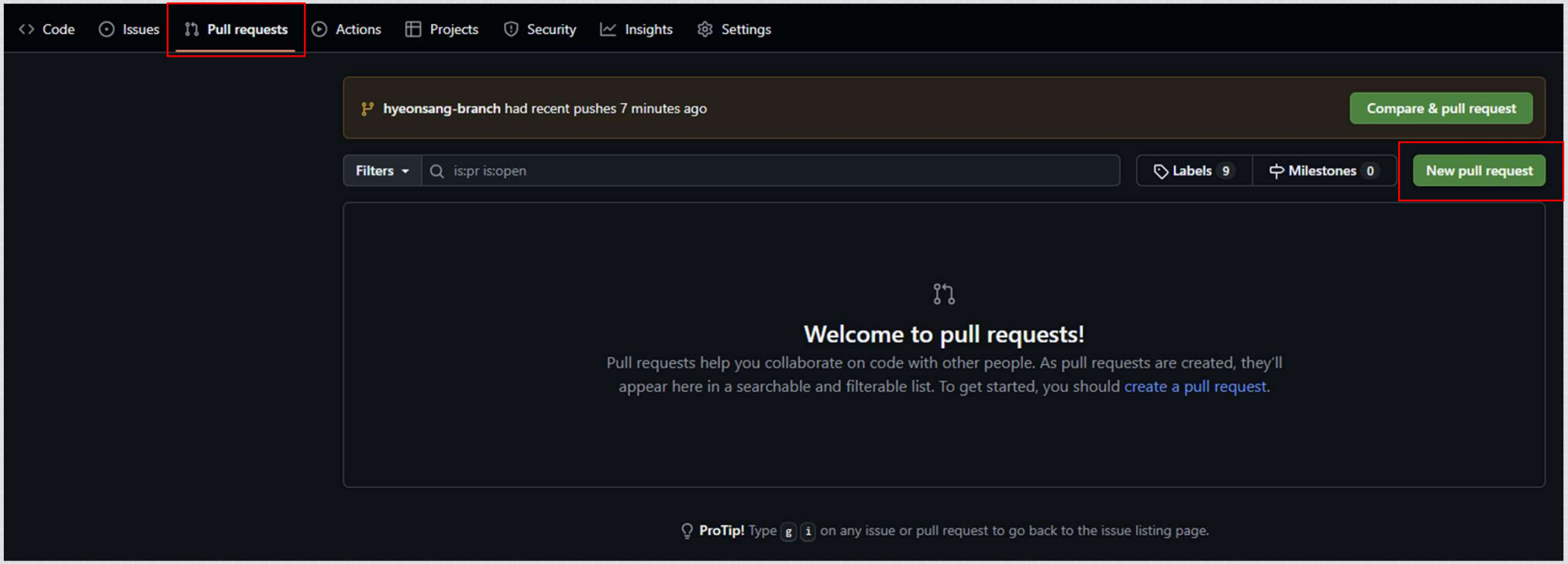
깃헙 브런치 예제



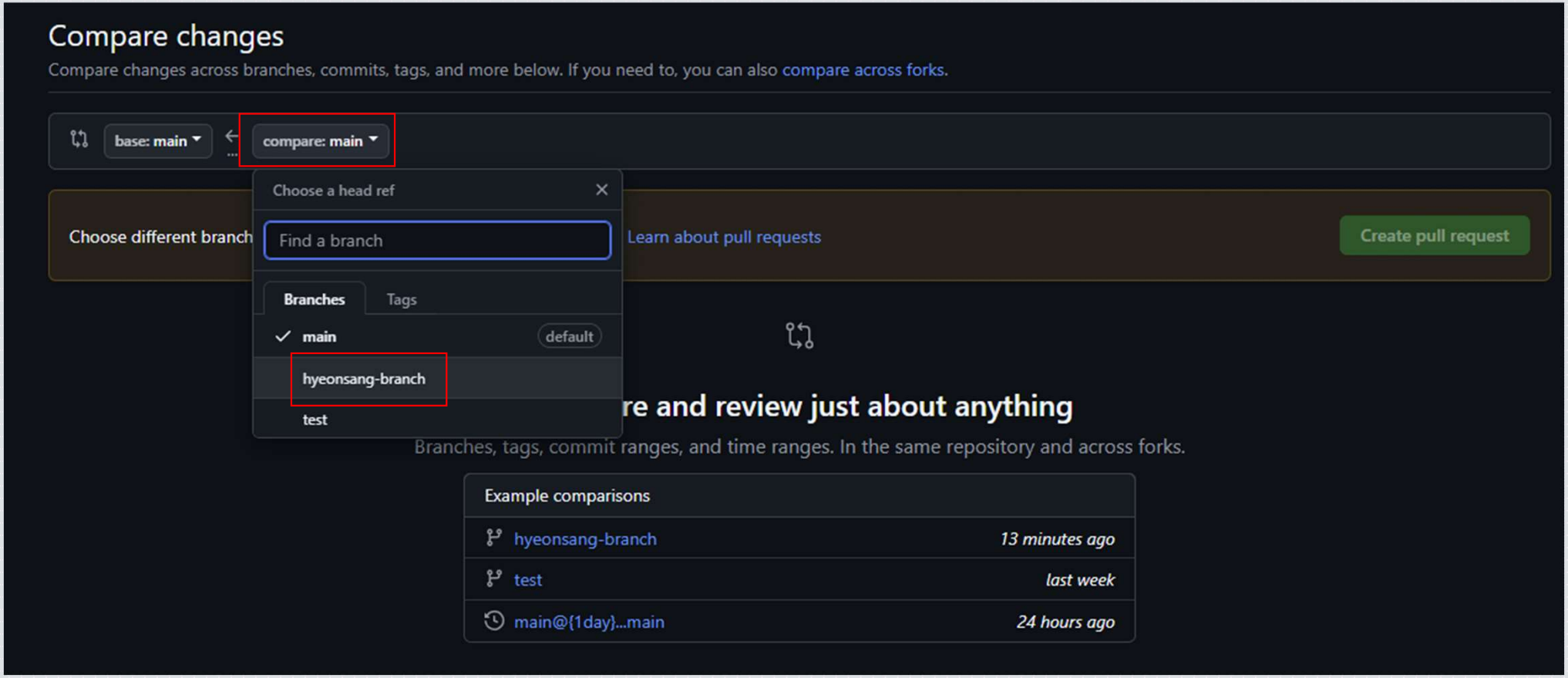
현재 1+1=2 기능을 hyeonsang-branch로 구현한 상태.

그 다음 해줘야 할 작업은 hyeonsang-branch를 main에 합쳐야 함.
(pull request (풀리퀘스트) : 브런치를 합친다)

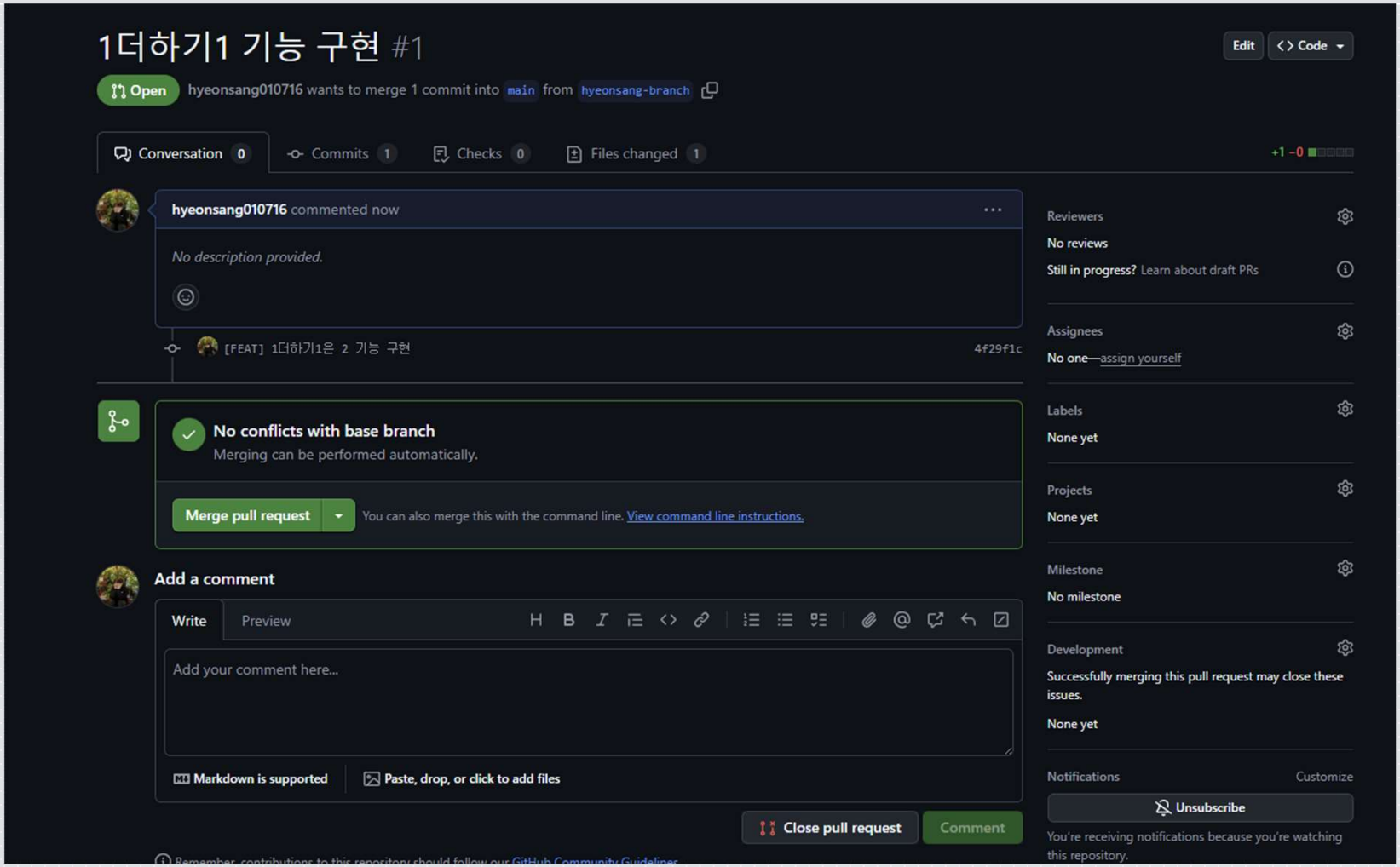
깃헙 브런치 예제


Code
Issues
Pull requests
Actions
Projects
Security
Insights
Settings
hyeonsang-branch had recent pushes 7 minutes ago
Compare & pull request
Filters
is:pr is:open
Labels 9
Milestones 0
New pull request
Welcome to pull requests!
Pull requests help you collaborate on code with other people. As pull requests are created, they'll appear here in a searchable and filterable list. To get started, you should create a pull request.
ProTip! Type g i on any issue or pull request to go back to the issue listing page.

깃헙 브런치 예제

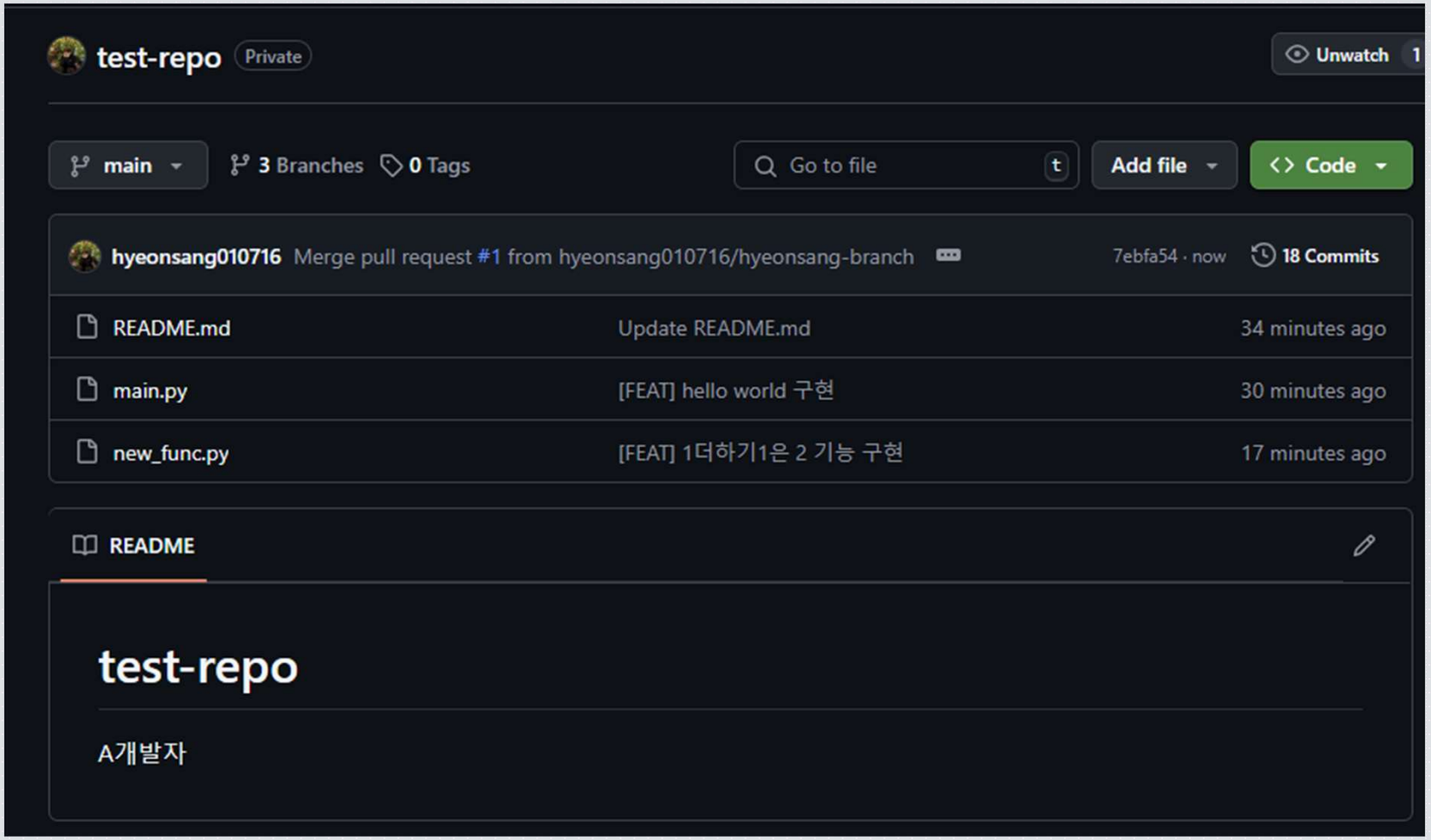

Compare changes
Compare changes across branches, commits, tags, and more below. If you need to, you can also compare across forks.
base: main
compare: main
Choose a head ref
Find a branch
Branches
Tags
main
default
hyeonsang-branch
test
Choose different branch
Learn about pull requests
Create pull request
re and review just about anything
Branches, tags, commit ranges, and time ranges. In the same repository and across forks.
Example comparisons
hyeonsang-branch
13 minutes ago
test
last week
main@{1day}...main
24 hours ago

깃헙 브런치 예제



여기까지 뜨면 성공!
(merge pull request는 팀장 권한이어서 위처럼 안 나올 겁니다!)

깃헙 브런치 예제



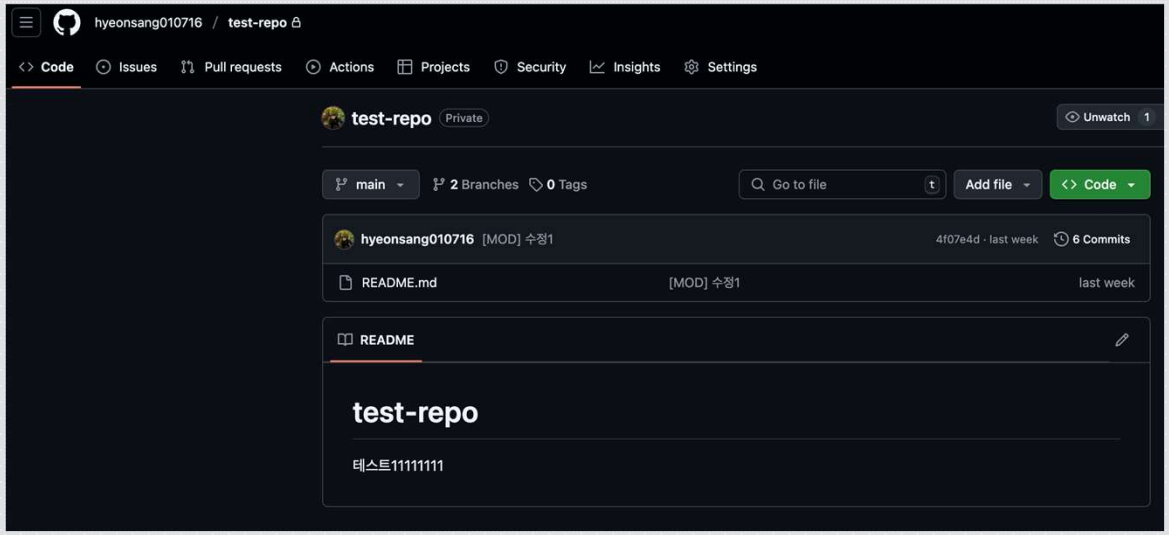
팀장이 수락을 했다면 main에 위처럼 반영 됨!

깃헙 브런치 예제

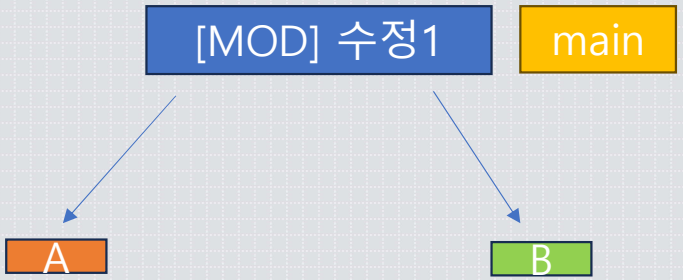
자.. 그래서 왜 branch를 나눠서 관리를 하면 좋을까..

1. main에 합치기 전, 코드 리뷰를 할 수 있음.
    1. 어떤 파일을 수정했는지
    2. 커밋은 잘했는지
    3. 누가 무슨 코드를 수정했고, 언제 수정했는지 다 알 수 있음!
    4. 팀장 개발자가 마음에 안 들면 무한 수정 시작..
2. 하지만 제일 좋은 장점은 히스토리 충돌이 사라짐

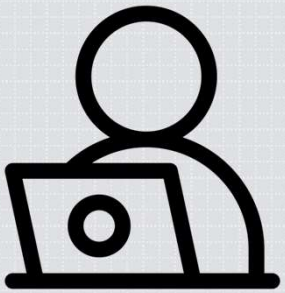
깃헙 브런치 예제



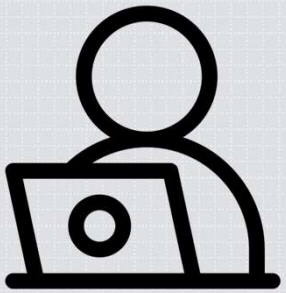
히스토리



<시나리오>
A개발자와 B개발자가 동시에 리모트 main 브런치에서 각자 브런치를 생성하고 기능 구현을 시작함

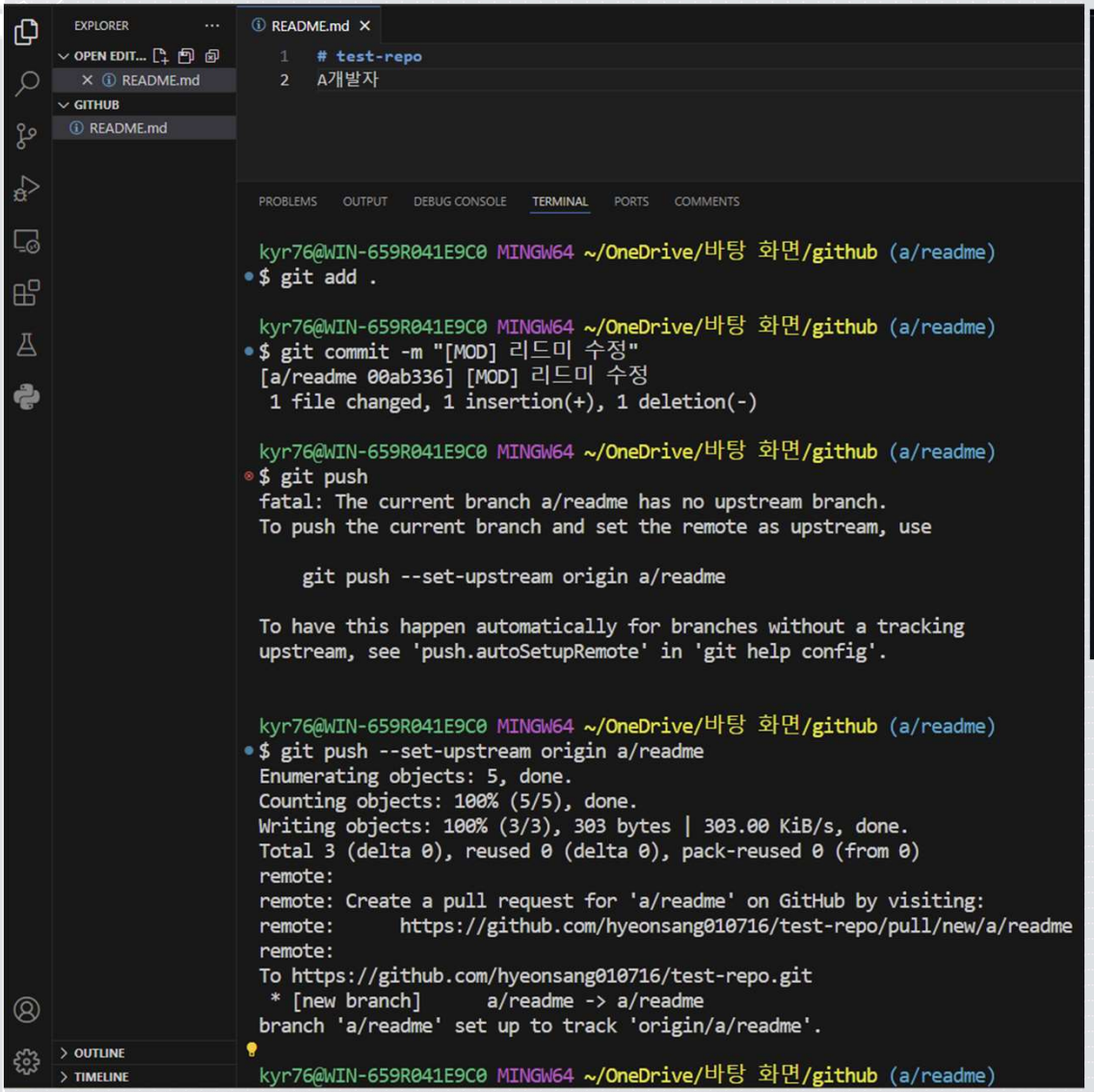


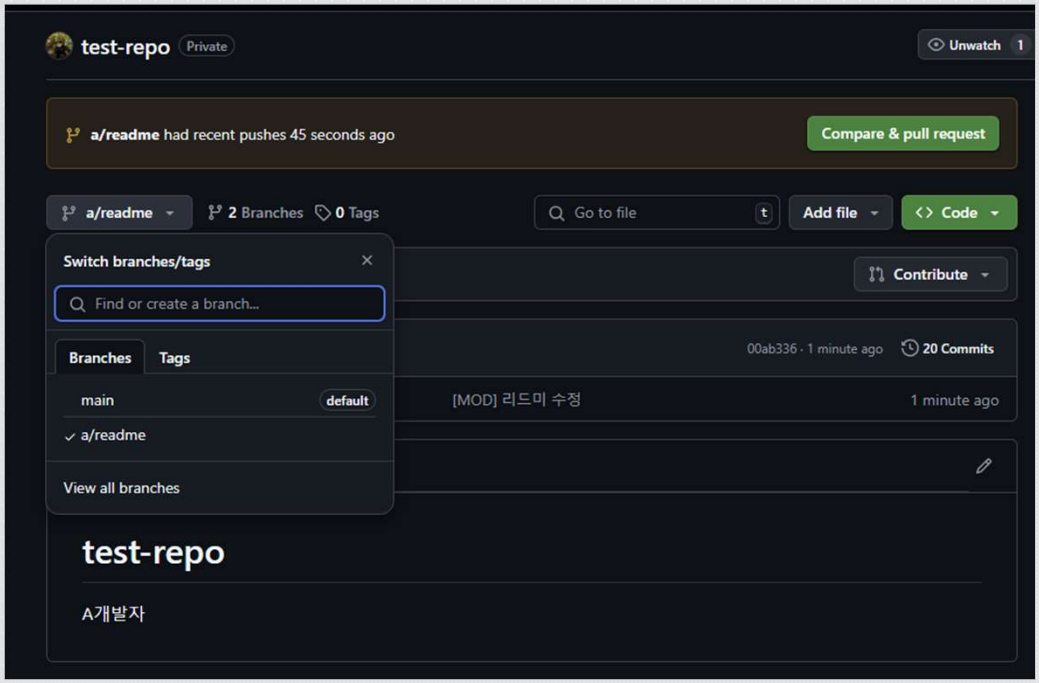
A개발자

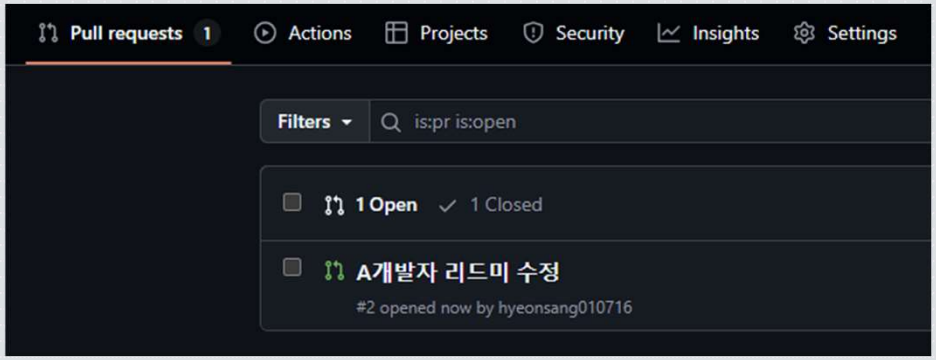


B개발자

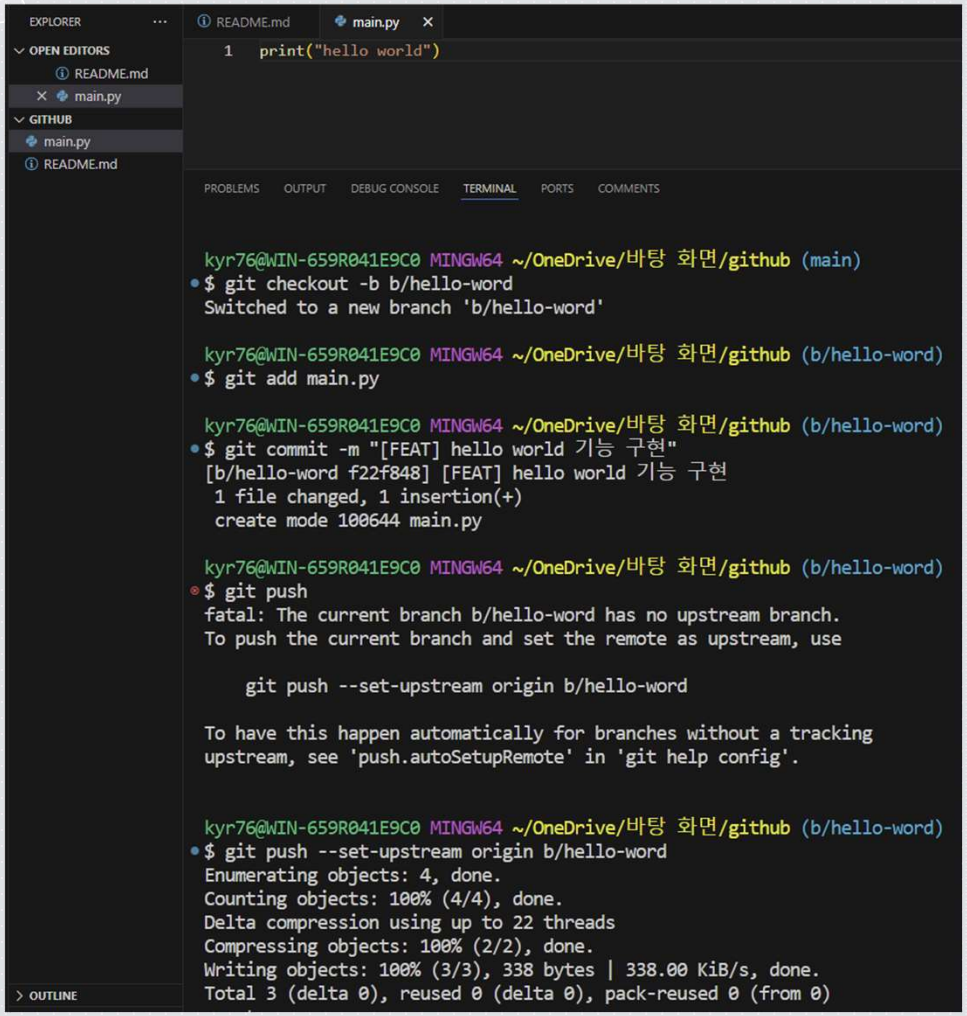
깃헙 브런치 예제

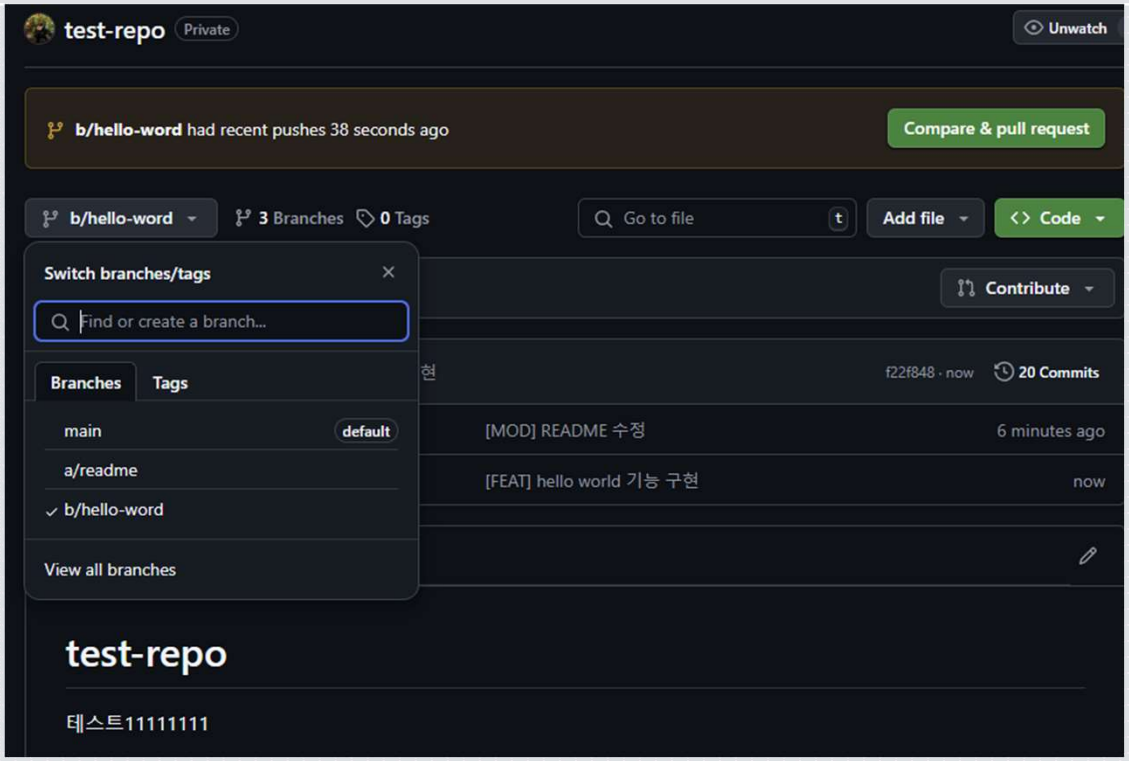

EXPLORER
OPEN EDIT...
README.md
GITHUB
README.md
README.md
1 # test-repo
2 A개발자
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/github (a/readme)
$ git add .
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/github (a/readme)
$ git commit -m "[MOD] 리드미 수정"
[a/readme 00ab336] [MOD] 리드미 수정
1 file changed, 1 insertion(+), 1 deletion(-)
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/github (a/readme)
$ git push
fatal: The current branch a/readme has no upstream branch.
To push the current branch and set the remote as upstream, use
git push --set-upstream origin a/readme
To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/github (a/readme)
$ git push --set-upstream origin a/readme
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 303 bytes | 303.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'a/readme' on GitHub by visiting:
remote: https://github.com/hyeonsang010716/test-repo/pull/new/a/readme
remote:
To https://github.com/hyeonsang010716/test-repo.git
* [new branch] a/readme -> a/readme
branch 'a/readme' set up to track 'origin/a/readme'.
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/github (a/readme)
OUTLINE
TIMELINE

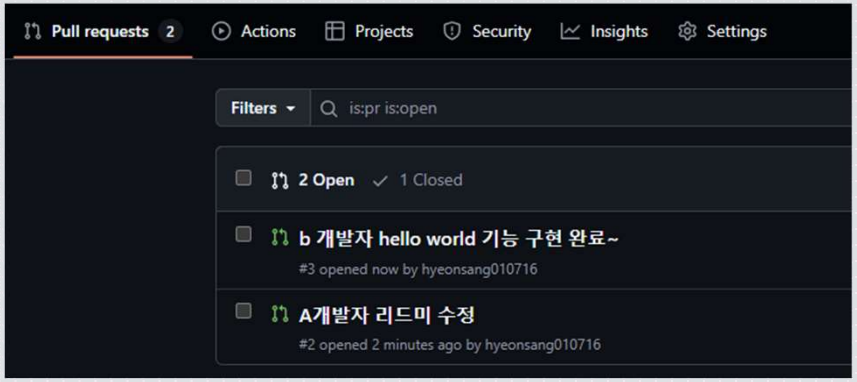

test-repo
Private
Unwatch 1
a/readme had recent pushes 45 seconds ago
Compare & pull request
a/readme
2 Branches
0 Tags
Go to file
t
Add file
Code
Switch branches/tags
Find or create a branch...
Branches
Tags
main
default
a/readme
View all branches
Contribute
00ab336 · 1 minute ago
20 Commits
[MOD] 리드미 수정
1 minute ago
test-repo
A개발자


Pull requests 1
Actions
Projects
Security
Insights
Settings
Filters
is:pr is:open
1 Open
1 Closed
A개발자 리드미 수정
#2 opened now by hyeonsang010716

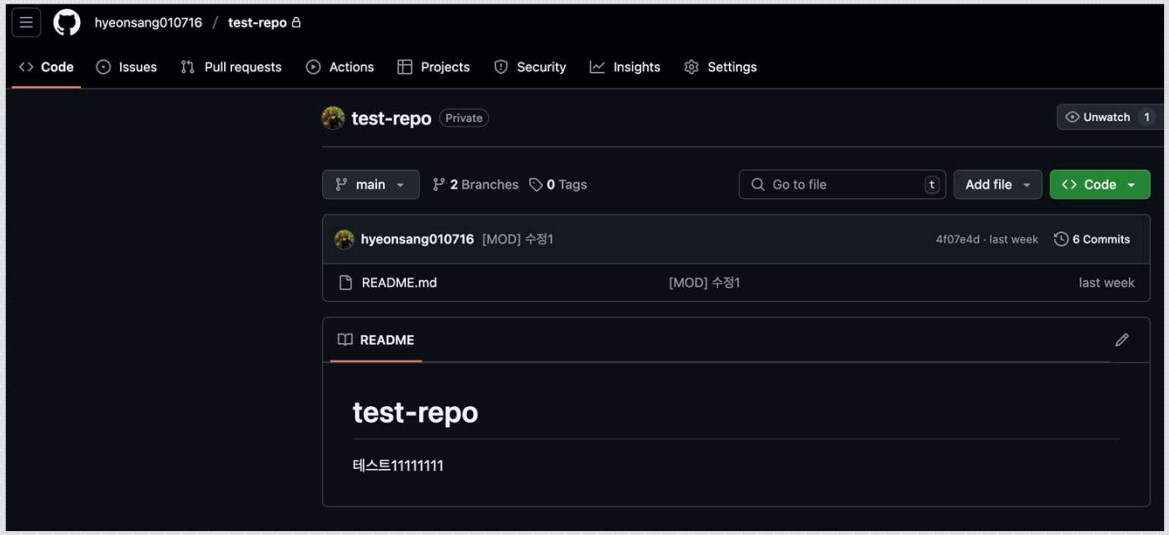
깃헙 브런치 예제


EXPLORER
OPEN EDITORS
README.md
main.py
GITHUB
main.py
README.md
OUTLINE
README.md
main.py
1 print("hello world")
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/github (main)
$ git checkout -b b/hello-word
Switched to a new branch 'b/hello-word'
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/github (b/hello-word)
$ git add main.py
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/github (b/hello-word)
$ git commit -m "[FEAT] hello world 기능 구현"
[b/hello-word f22f848] [FEAT] hello world 기능 구현
1 file changed, 1 insertion(+)
create mode 100644 main.py
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/github (b/hello-word)
$ git push
fatal: The current branch b/hello-word has no upstream branch.
To push the current branch and set the remote as upstream, use
git push --set-upstream origin b/hello-word
To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/github (b/hello-word)
$ git push --set-upstream origin b/hello-word
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 22 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 338 bytes | 338.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)

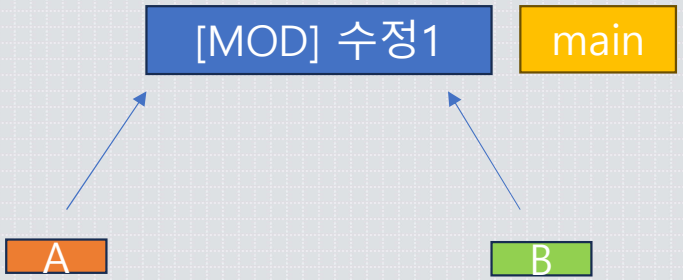

test-repo
Private
Unwatch
b/hello-word had recent pushes 38 seconds ago
Compare & pull request
b/hello-word
3 Branches
0 Tags
Go to file
t
Add file
Code
Switch branches/tags
Find or create a branch...
Branches
Tags
main
default
a/readme
b/hello-word
View all branches
Contribute
f22f848 · now
20 Commits
[MOD] README 수정
6 minutes ago
[FEAT] hello world 기능 구현
now
test-repo
테스트11111111


Pull requests 2
Actions
Projects
Security
Insights
Settings
Filters
is:pr is:open
2 Open
1 Closed
b 개발자 hello world 기능 구현 완료~
#3 opened now by hyeonsang010716
A개발자 리드미 수정
#2 opened 2 minutes ago by hyeonsang010716

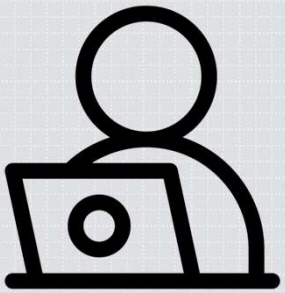
깃헙 브런치 예제



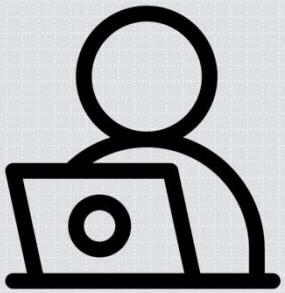
히스토리



<시나리오>
동시에 기능 구현 완료를 해서, 풀리퀘스트를 보냄

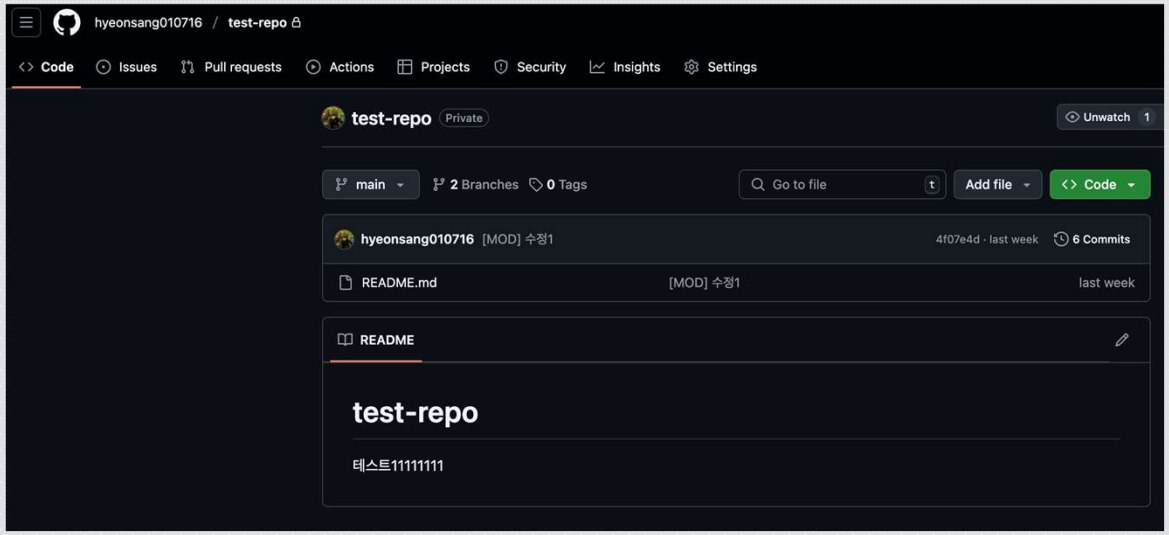


A개발자

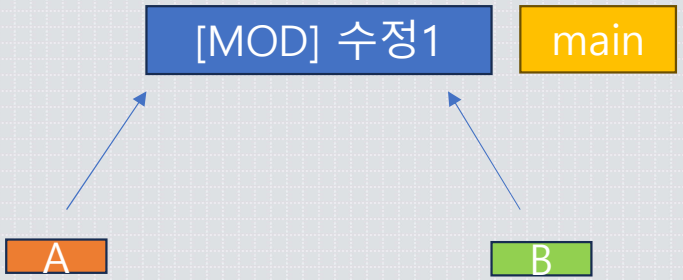


B개발자

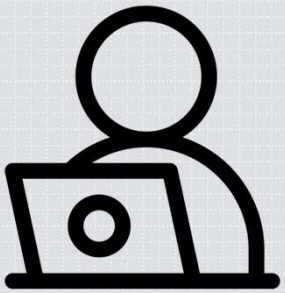
깃헙 브런치 예제



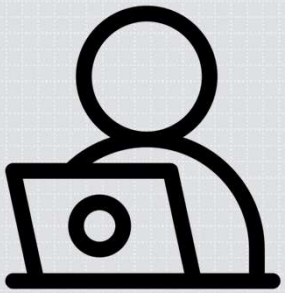
히스토리



<시나리오>
동시에 기능 구현 완료를 해서, 풀리퀘스트를 보냄

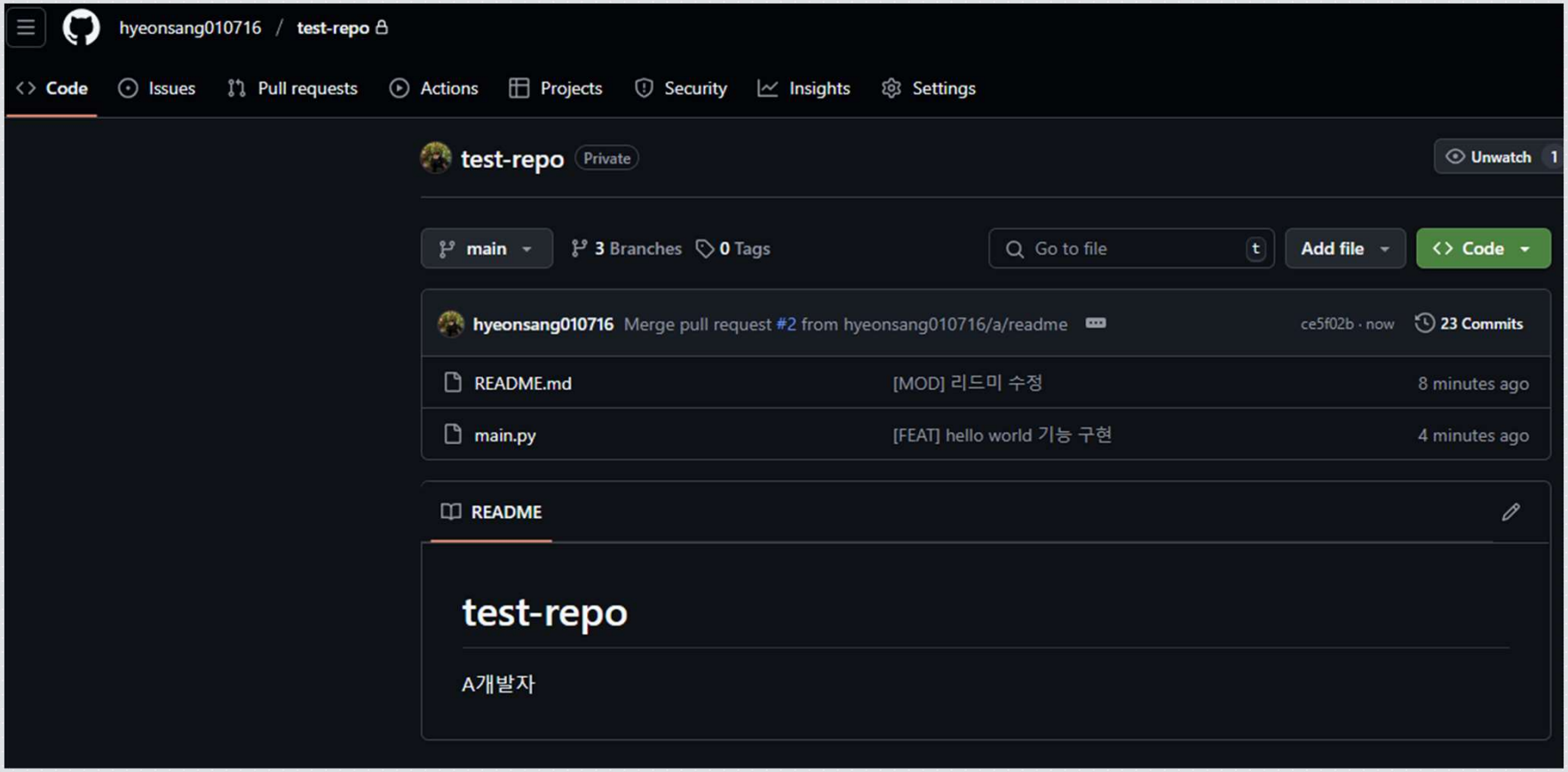


A개발자



B개발자

깃헙 브런치 예제



동시에 풀리퀘스트를 보냈지만,
팀장 개발자가 merge만 하면 충돌X 반영 완료!

이번 주 실습!

1. **feat/깃헙아이디-main** 으로 브런치 만들기 ex) feat/hyeonsang010716-main
2. 자신의 디렉토리 (이전 실습으로 만든 디렉토리)에 **main.py** 생성
3. git add 한 후, git commit -m **"[FEAT] main 생성"** 으로 커밋하고 푸쉬하기
4. 풀리퀘스트까지 보내면 실습 끝!

이번 주 실습!
(먼저 git pull origin main을 해서 업데이트 된 main을 가져와주세요!)

1. **feat/깃헙아이디-main** 으로 브런치 만들기 ex) feat/hyeonsang010716-main

2. 자신의 디렉토리 (이전 실습으로 만든 디렉토리)에 **main.py** 생성

3. git add 한 후, git commit -m **"[FEAT] main 생성"** 으로 커밋하고 푸쉬하기

4. 풀리퀘스트까지 보내면 실습 끝!

이번 주 실습!
(먼저 git pull origin main을 해서 업데이트 된 main을 가져와주세요!)

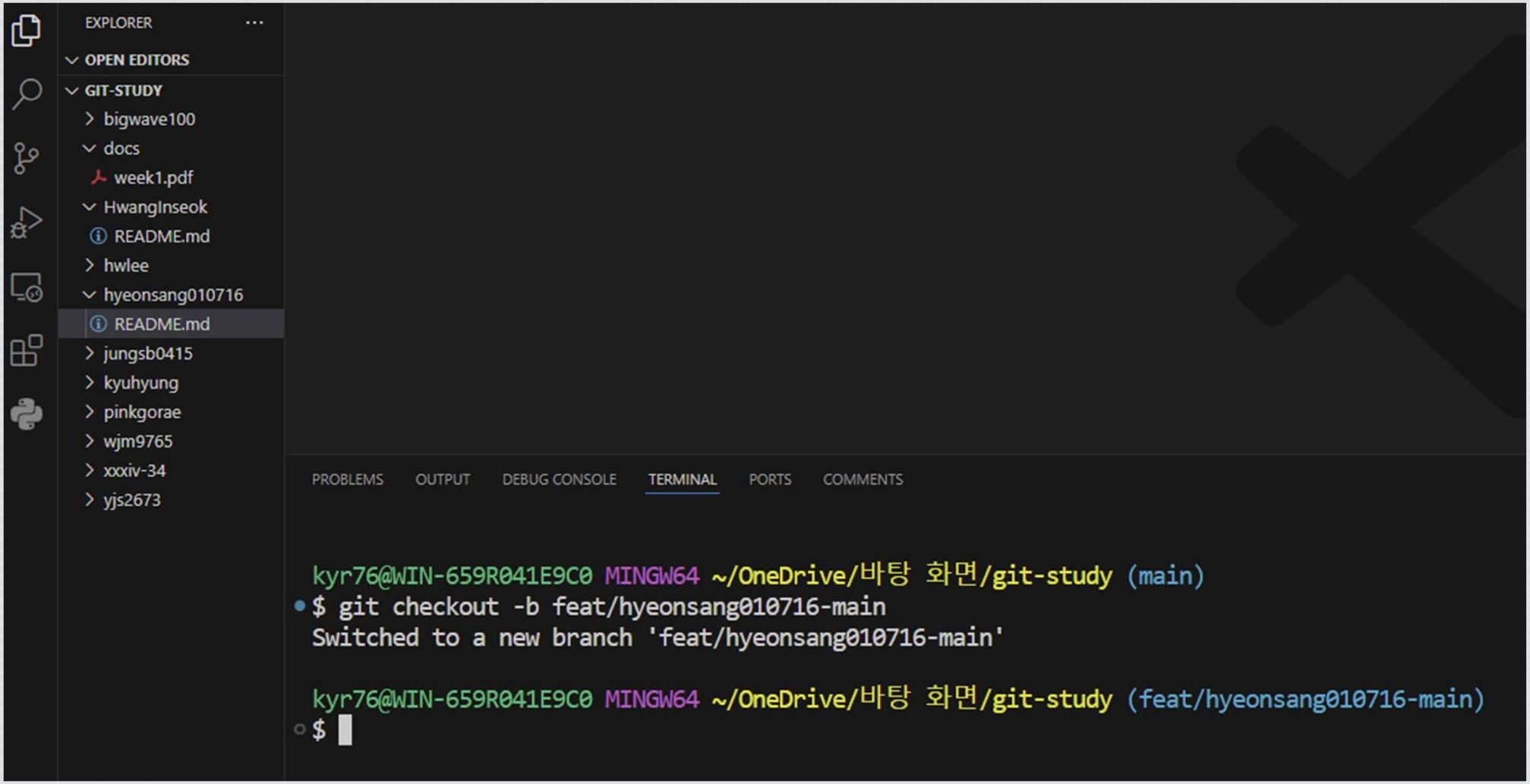
1. **feat/깃헙아이디-main** 으로 브런치 만들기 ex) feat/hyeonsang010716-main

2. 자신의 디렉토리 (이전 실습으로 만든 디렉토리)에 **main.py** 생성

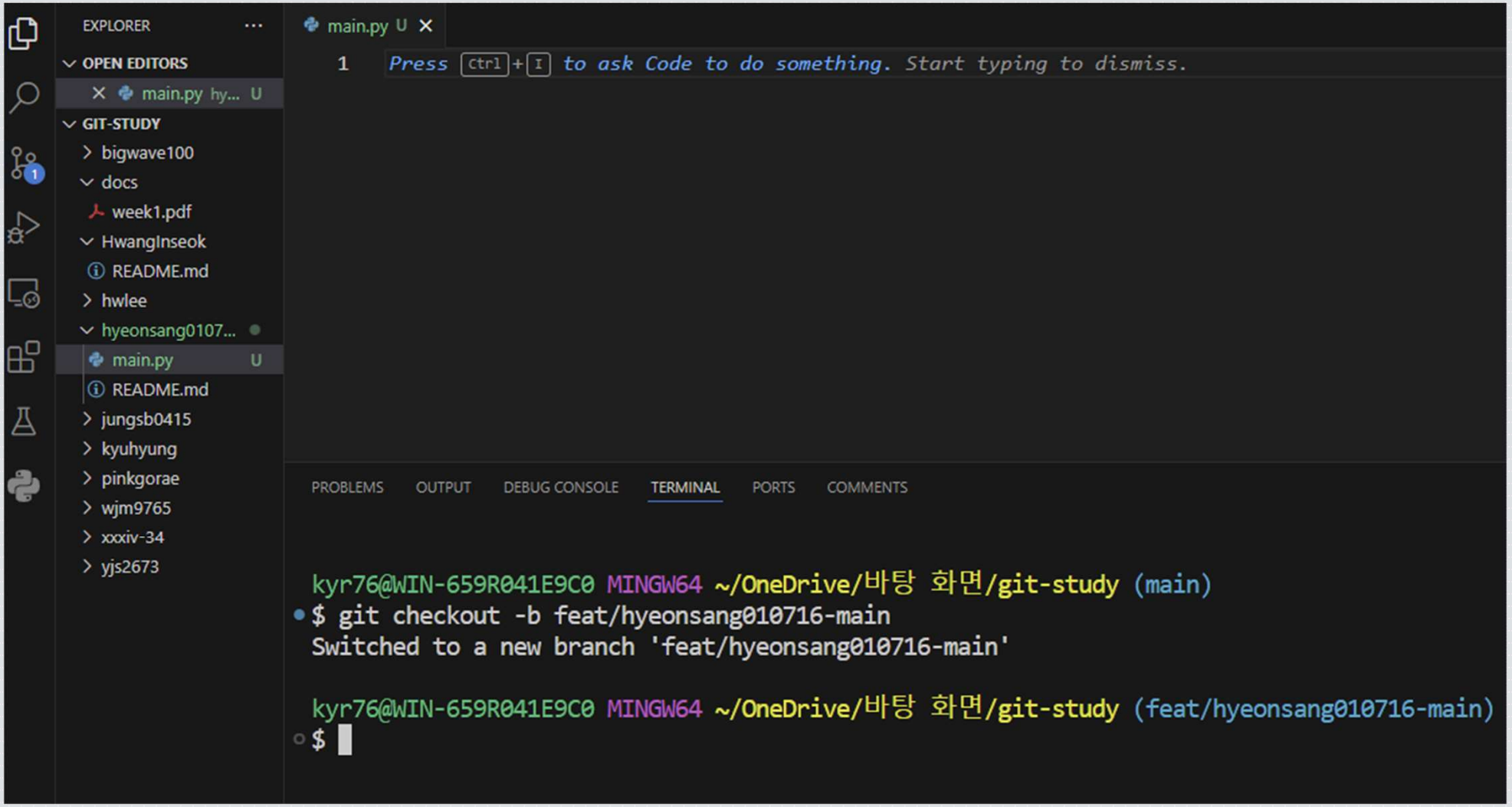
3. git add 한 후, git commit -m **"[FEAT] main 생성"** 으로 커밋하고 푸쉬하기

4. 풀리퀘스트까지 보내면 실습 끝!

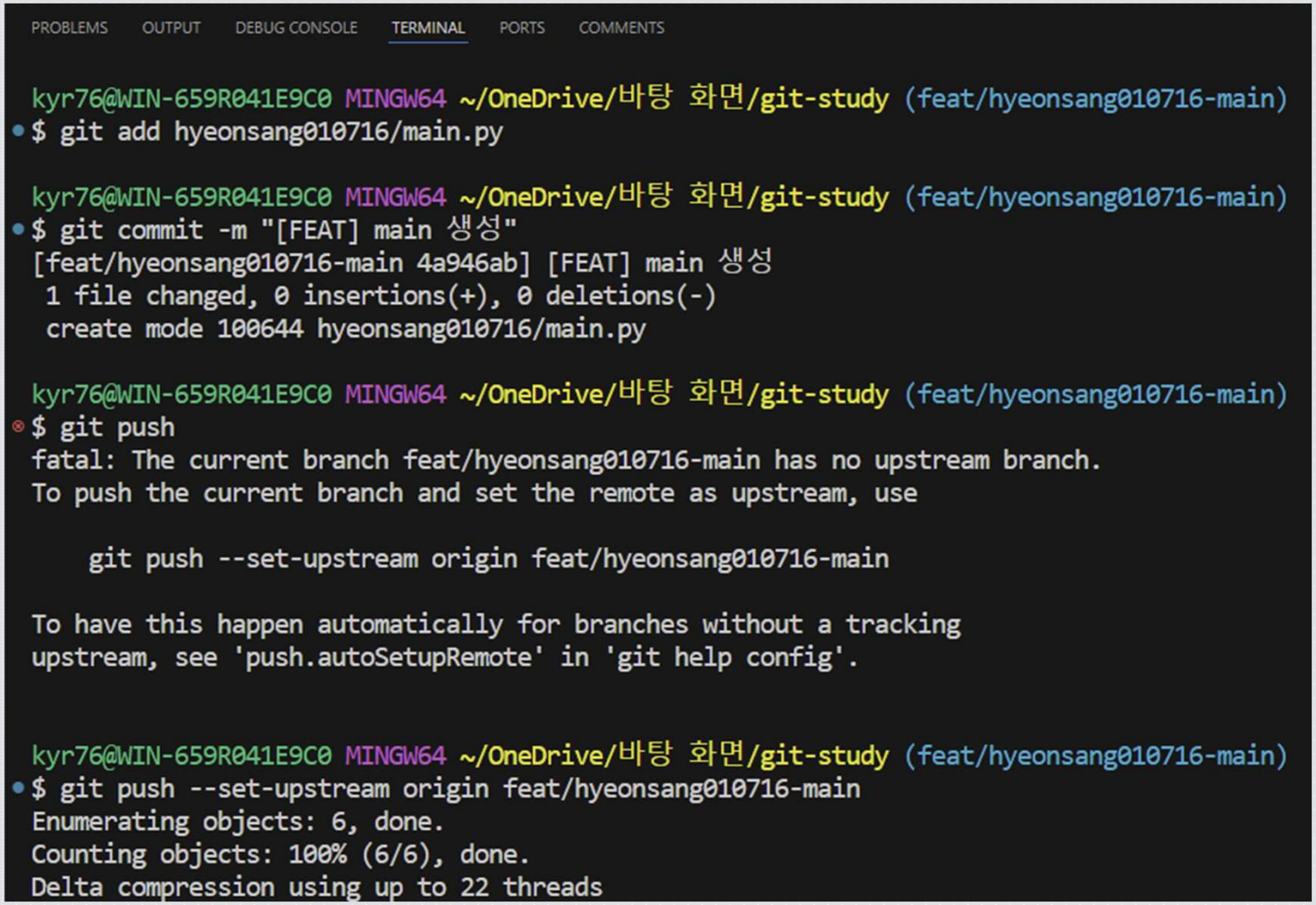
깃헙 실습


EXPLORER
OPEN EDITORS
GIT-STUDY
bigwave100
docs
week1.pdf
HwangInseok
README.md
hwlee
hyeonsang010716
README.md
jungsb0415
kyuhyung
pinkgorae
wjm9765
xxxiv-34
yjs2673
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
COMMENTS
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/git-study (main)
$ git checkout -b feat/hyeonsang010716-main
Switched to a new branch 'feat/hyeonsang010716-main'
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/git-study (feat/hyeonsang010716-main)
$

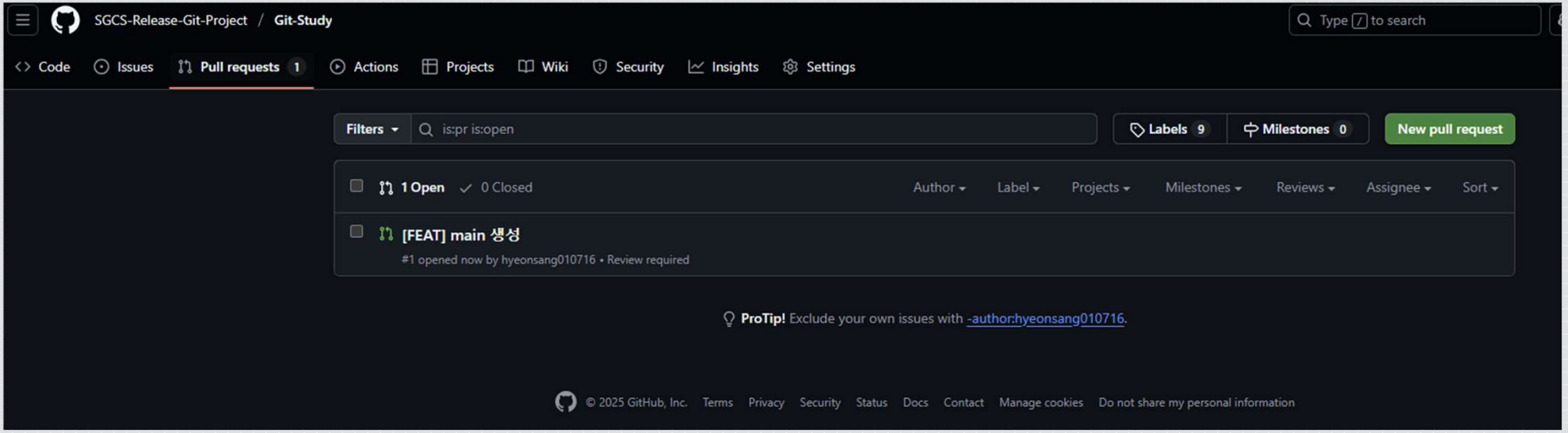
깃헙 실습


EXPLORER
main.py U
1 Press Ctrl + I to ask Code to do something. Start typing to dismiss.
OPEN EDITORS
main.py hy... U
GIT-STUDY
bigwave100
docs
week1.pdf
HwangInseok
README.md
hwlee
hyeonsang0107...
main.py U
README.md
jungsb0415
kyuhyung
pinkgorae
wjm9765
xxxiv-34
yjs2673
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
COMMENTS
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/git-study (main)
$ git checkout -b feat/hyeonsang010716-main
Switched to a new branch 'feat/hyeonsang010716-main'
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/git-study (feat/hyeonsang010716-main)
$

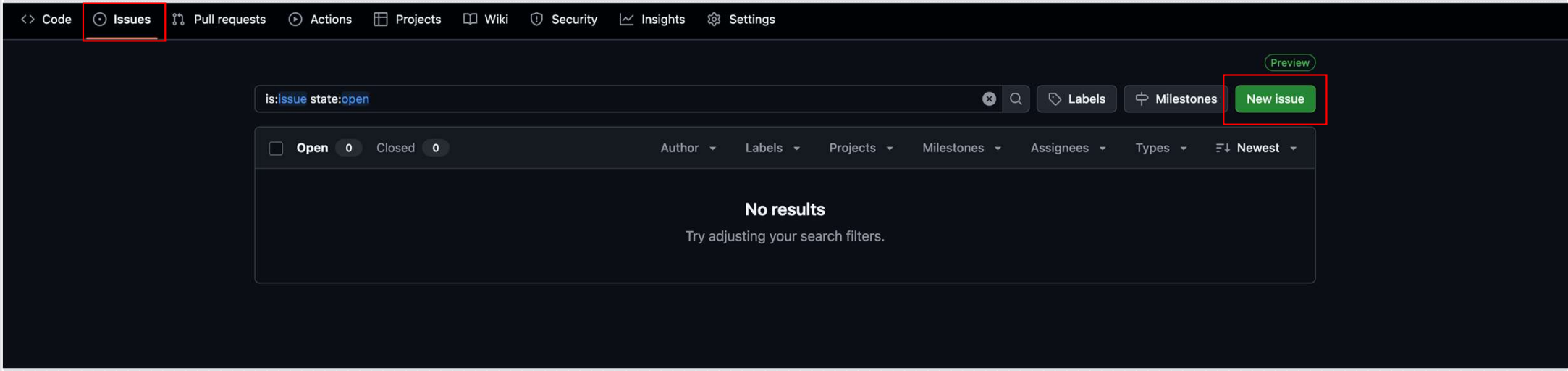
깃헙 실습


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
COMMENTS
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/git-study (feat/hyeonsang010716-main)
$ git add hyeonsang010716/main.py
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/git-study (feat/hyeonsang010716-main)
$ git commit -m "[FEAT] main 생성"
[feat/hyeonsang010716-main 4a946ab] [FEAT] main 생성
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 hyeonsang010716/main.py
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/git-study (feat/hyeonsang010716-main)
$ git push
fatal: The current branch feat/hyeonsang010716-main has no upstream branch.
To push the current branch and set the remote as upstream, use
git push --set-upstream origin feat/hyeonsang010716-main
To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.
kyr76@WIN-659R041E9C0 MINGW64 ~/OneDrive/바탕 화면/git-study (feat/hyeonsang010716-main)
$ git push --set-upstream origin feat/hyeonsang010716-main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 22 threads

깃헙 실습


SGCS-Release-Git-Project / Git-Study
Type / to search
Code
Issues
Pull requests 1
Actions
Projects
Wiki
Security
Insights
Settings
Filters
is:pr is:open
Labels 9
Milestones 0
New pull request
1 Open
0 Closed
Author
Label
Projects
Milestones
Reviews
Assignee
Sort
[FEAT] main 생성
#1 opened now by hyeonsang010716 • Review required
ProTip! Exclude your own issues with -author:hyeonsang010716.
© 2025 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact
Manage cookies
Do not share my personal information

언제든지 자유롭게 물어보기!



질문이 생기면 주저 말고 GitHub Issue에 남겨주세요 ;)