



SourceTree 써보기 & 내 컴퓨터에 clone 해보기

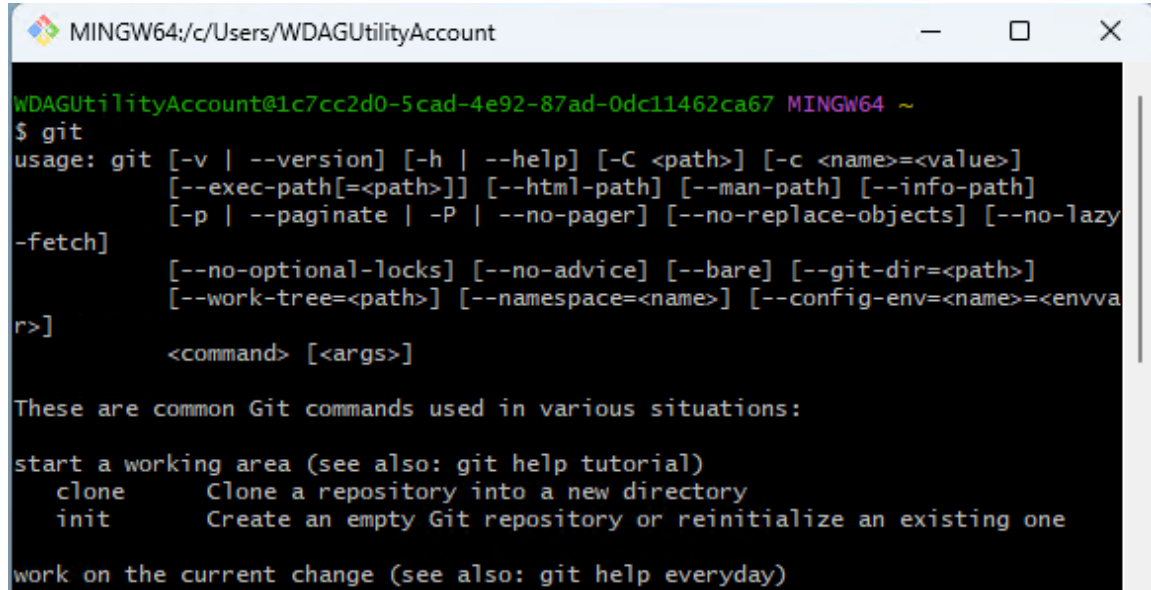
버전 관리의 시작점

- 사용이 어려움
- 그냥 폴더로 관리 하면 안되나?
- 가능함.
- 그런데, 컴퓨터가 해킹 당한다면?!
- 혹은, 컴퓨터가 바뀐다면? (집컴, 회사컴)
- 프로젝트를 쉽게 공유하고 싶다면?
- 여러곳에서 사용가능한 Github이 좋음
- 그러나, 사용방법을 배워야 한다!



자, 이제 시작이야

- Git 레포지토리 관리는 사실 어려움, 왜냐하면...



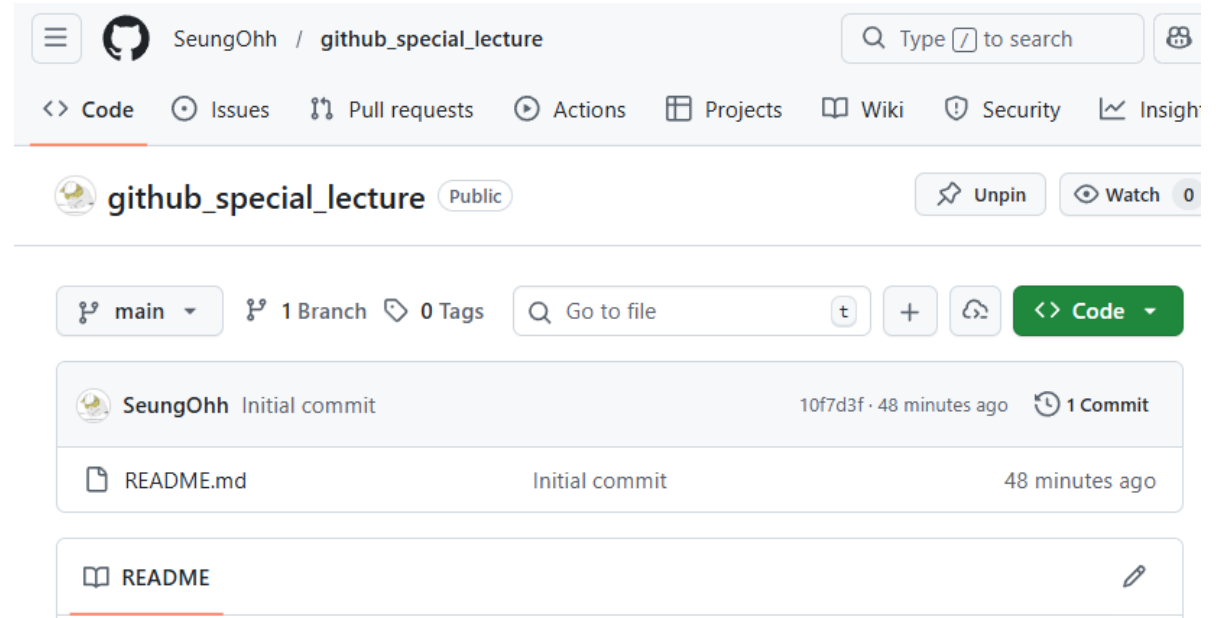
```
MINGW64:/c:/Users/WDAGUtilityAccount
WDAGUtilityAccount@1c7cc2d0-5cad-4e92-87ad-0dc11462ca67 MINGW64 ~
$ git
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
      [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
      [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--no-lazy
      --no-optional-locks] [--no-advice] [--bare] [--git-dir=<path>]
      [--work-tree=<path>] [--namespace=<name>] [--config-env=<name>=<envva
      <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
```

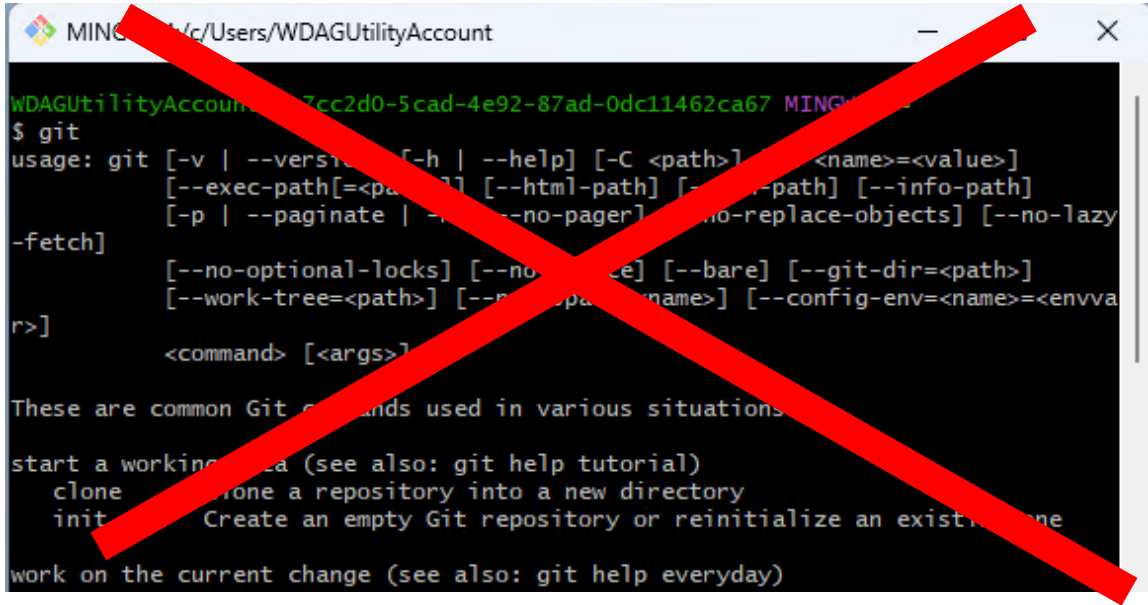
원래, git을 사용하려면 여기서 했어야 함.



웹페이지에서 작업하는건 기능이 부족함.
(단순 수정밖에 안됨)

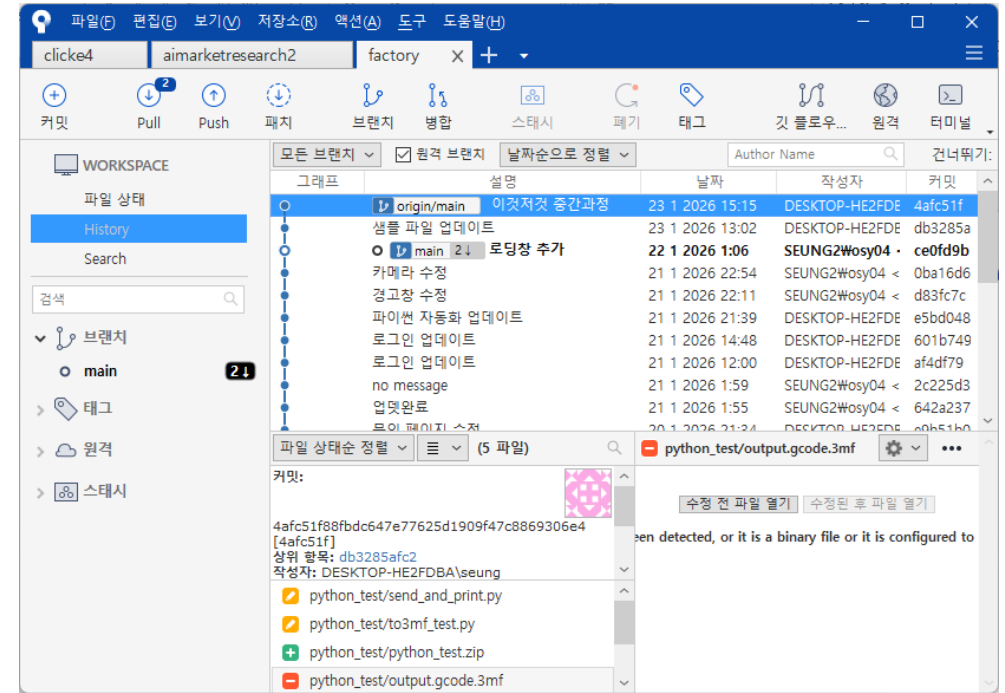
SourceTree, 깃갯프로그램

- Git 레포지토리 관리는 사실 어려움, 왜냐하면...



원래, git을 사용하려면 여기서 했어야 함.

기능의 100%를 사용할 수 있지만, 어려움



sourcetree라고 하는 엄청 좋은 툴을 사용하면 편함!

하지만 기능의 90% 만 사용 가능

SourceTree, 깃갇프로그램

- Sourcetree를 쓰면 편함

나쁜 답변

```
MINGW64/c/Users/WDAGUtilityAccount

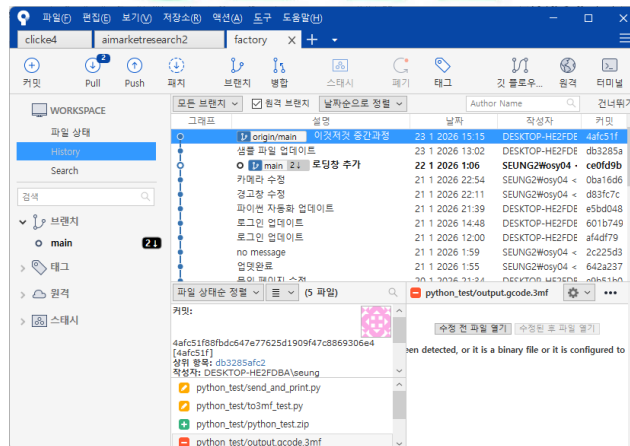
WDAGUtilityAccount@1c7cc2d0-5cad-4e92-87ad-0dc11462ca67 MINGW64 ~
$ git
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
[--exec-path=<path>] [--html-path] [--man-path] [--info-path]
[-p | --paginate | -P | --no-pager] [--no-replace-objects] [--no-lazy
-fetch]
[--no-optional-locks] [--no-advice] [--bare] [--git-dir=<path>]
[--work-tree=<path>] [--namespace=<name>] [--config-env=<name>=<envva
r>]
<command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
clone      Clone a repository into a new directory
init       Create an empty Git repository or reinitialize an existing one
work on the current change (see also: git help everyday)
```

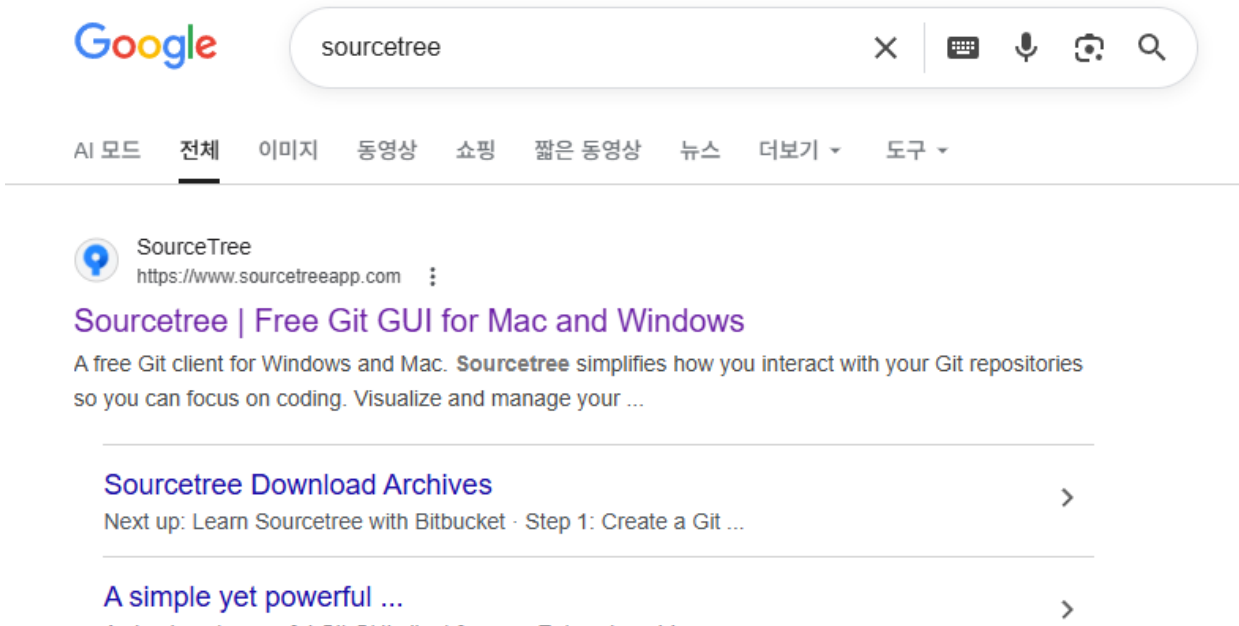
- 시간이 오래 걸림
- 지적허영심을 뽐내는 것 같음
- 지루하고 현학적임
- 논쟁 발생 가능성 100%
- 논쟁에서 밀리면 내 세상이 무너짐

좋은 답변

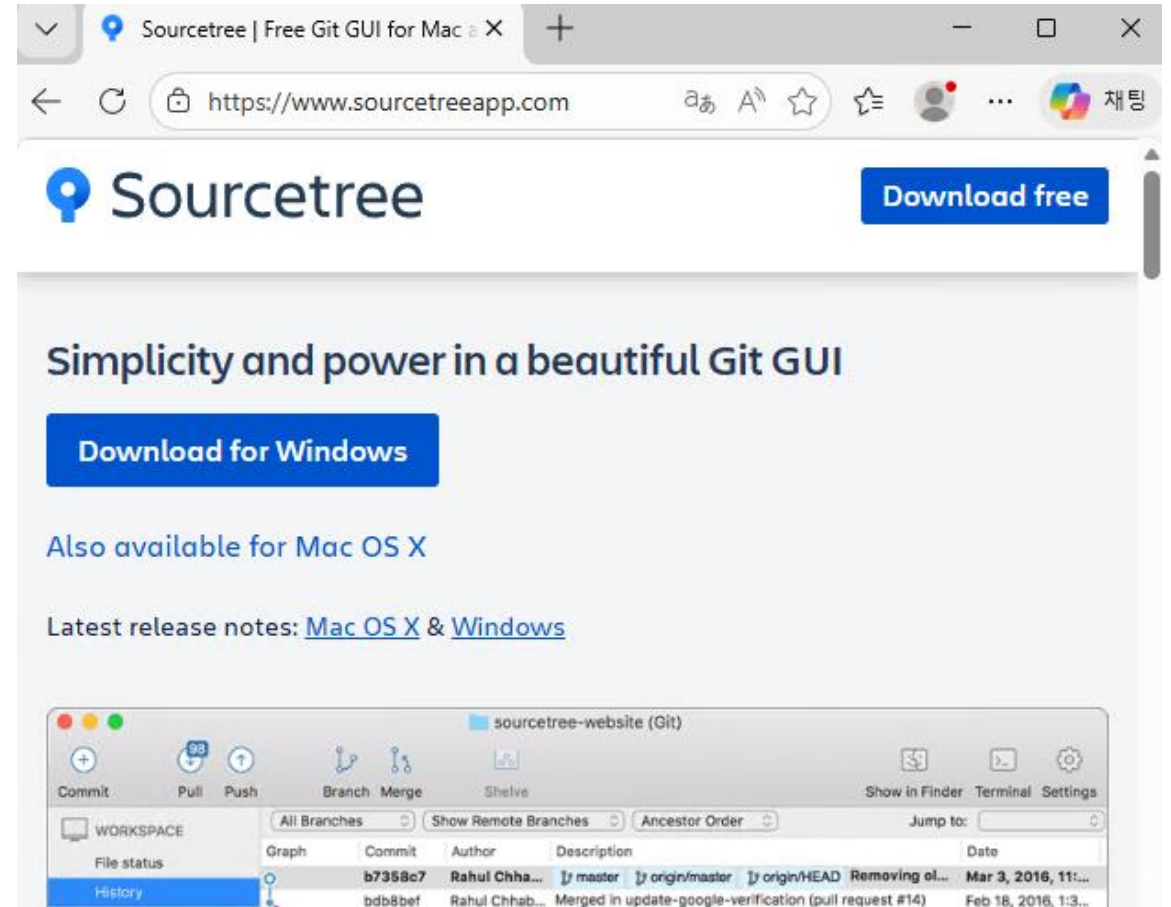


- 금방 끝남
- 오래 기억에 남음
- 팩트임
- 간단하고 사전 지식이 필요X
- 논쟁에서 밀릴 확률 0에 수렴함

SourceTree, 깃프로그램

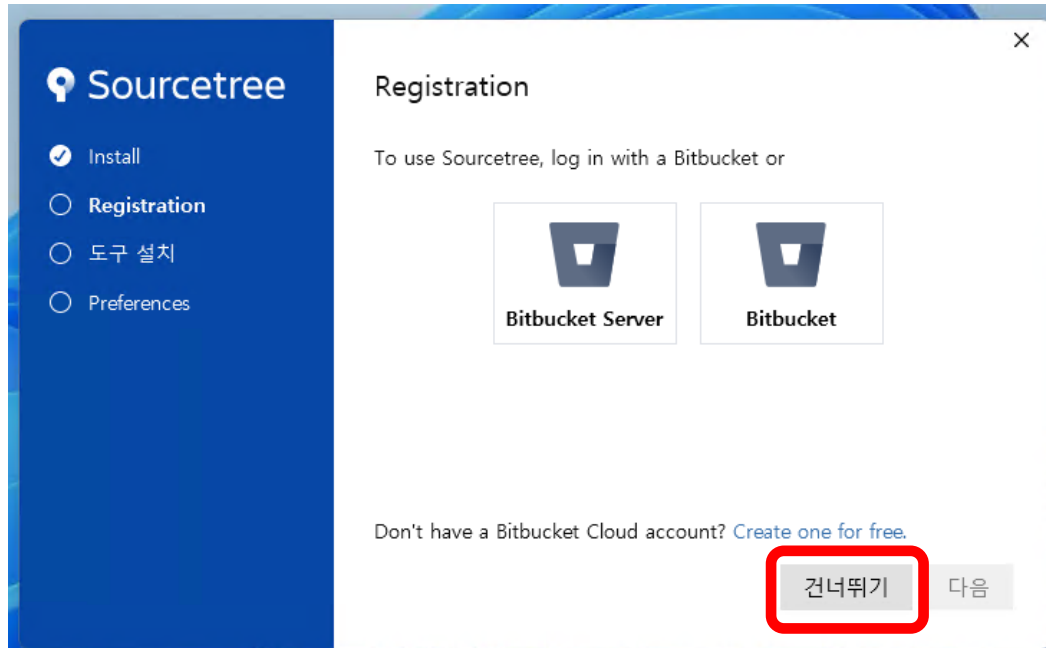


sourcetree 검색하고

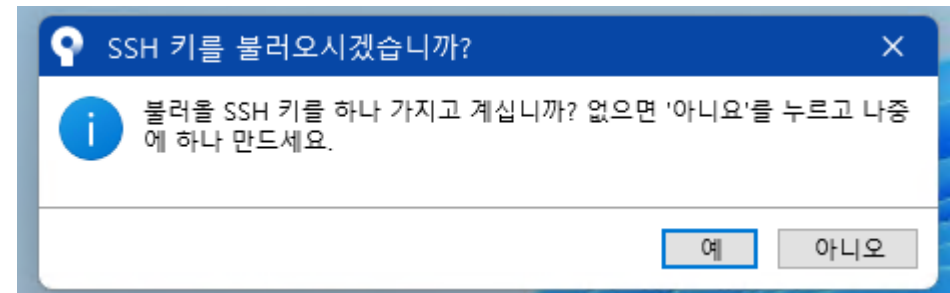


Download for Windows 클릭

SourceTree, 깃갓프로그램



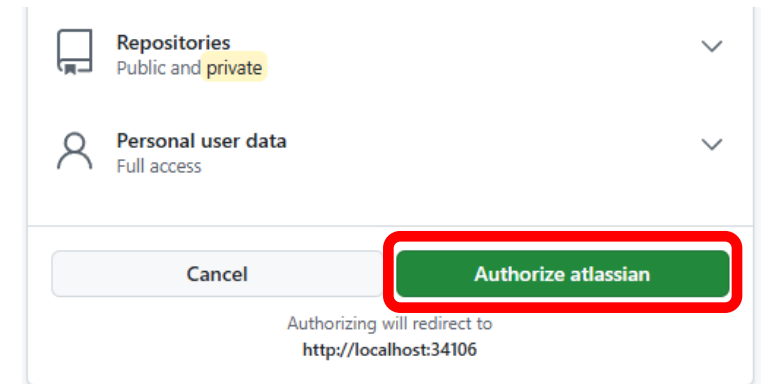
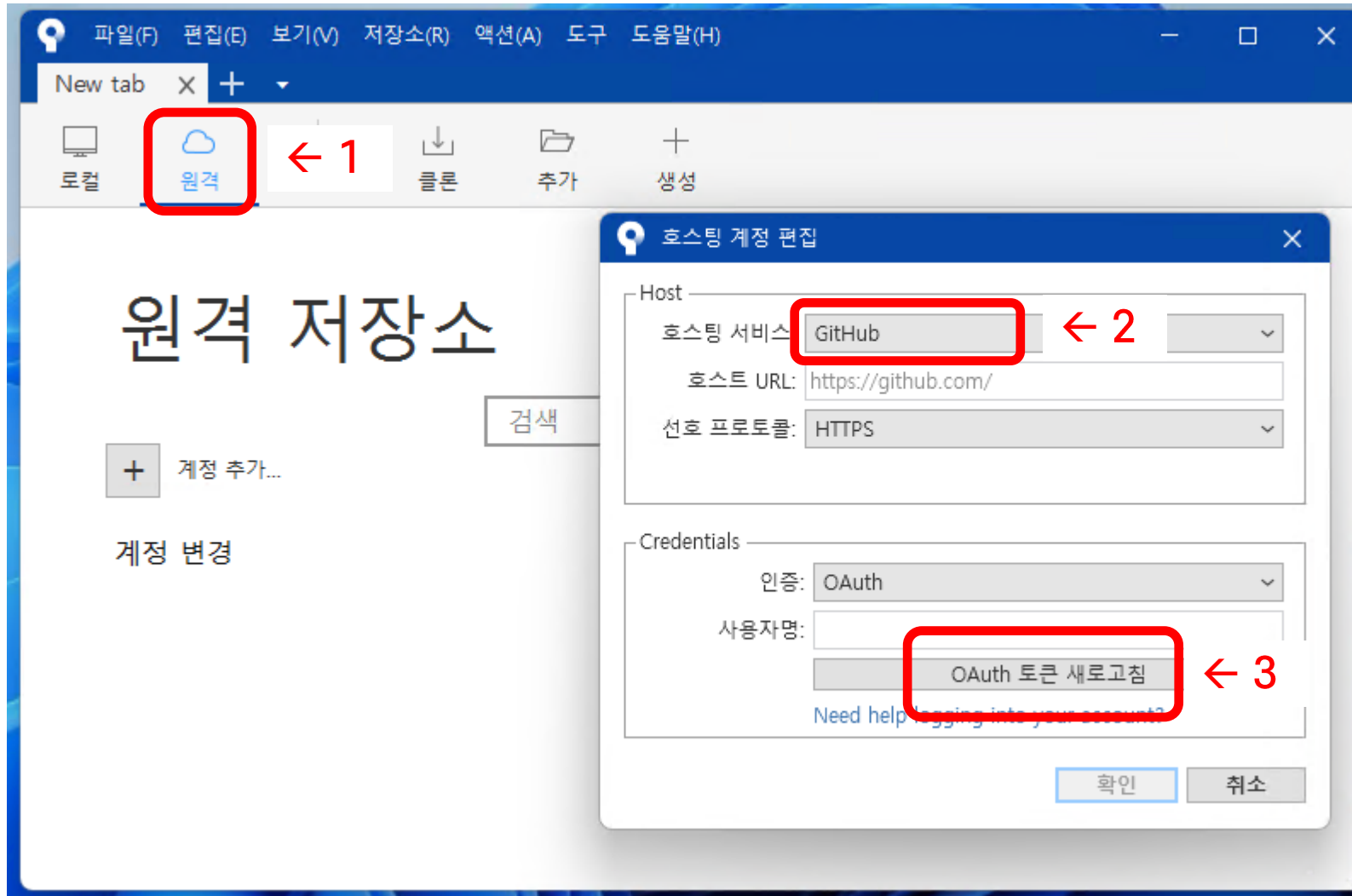
건너뛰기



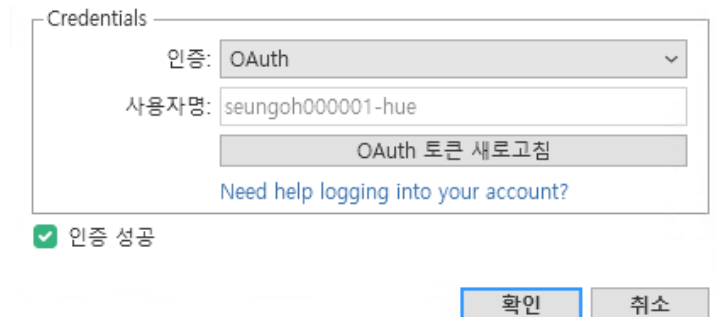
아니오 클릭

SourceTree, 깃갇프로그램

- Sourcetree에서 Github 로그인



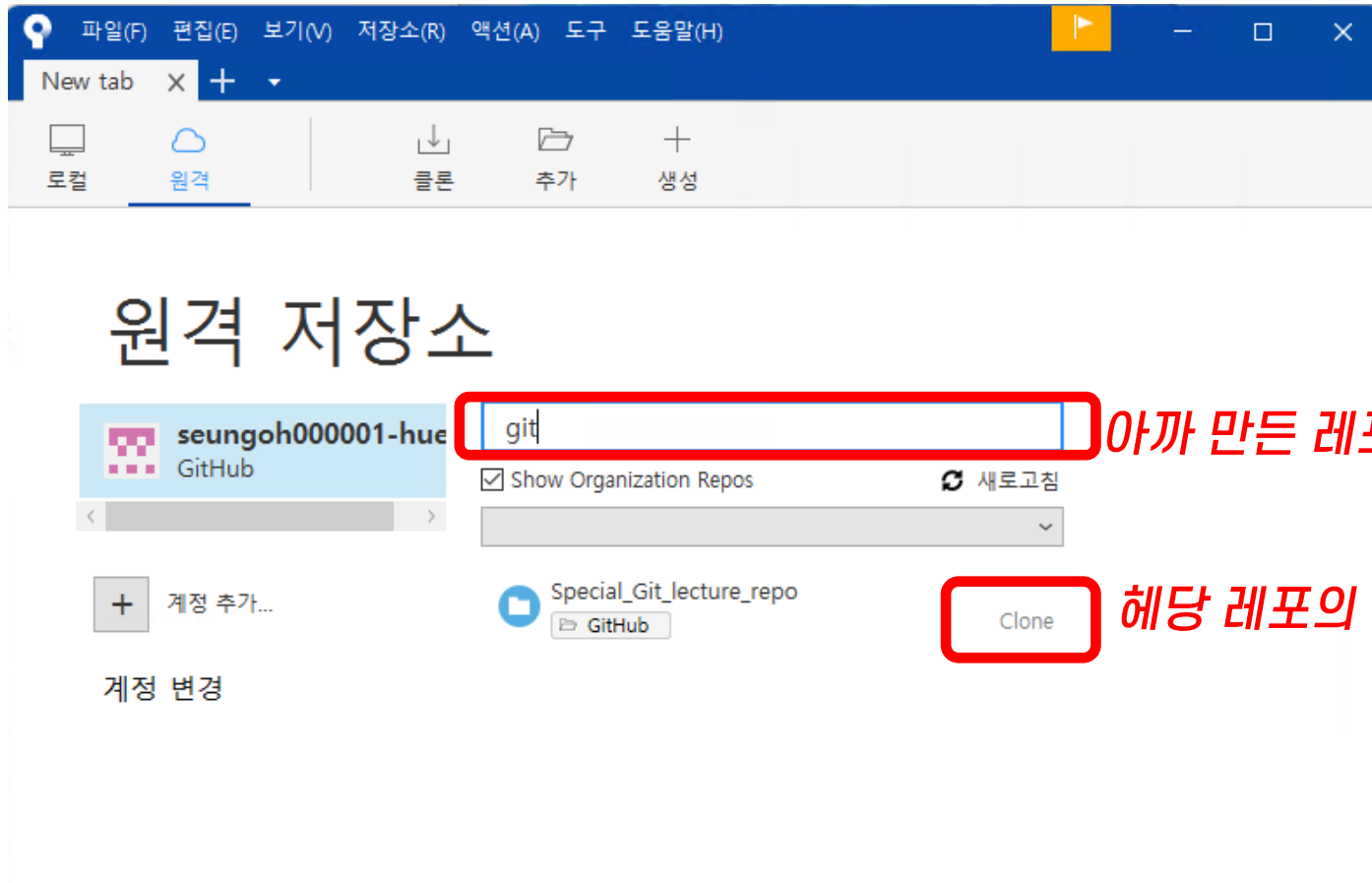
여기를 누르면 완료



로그인 완료!
확인 클릭!

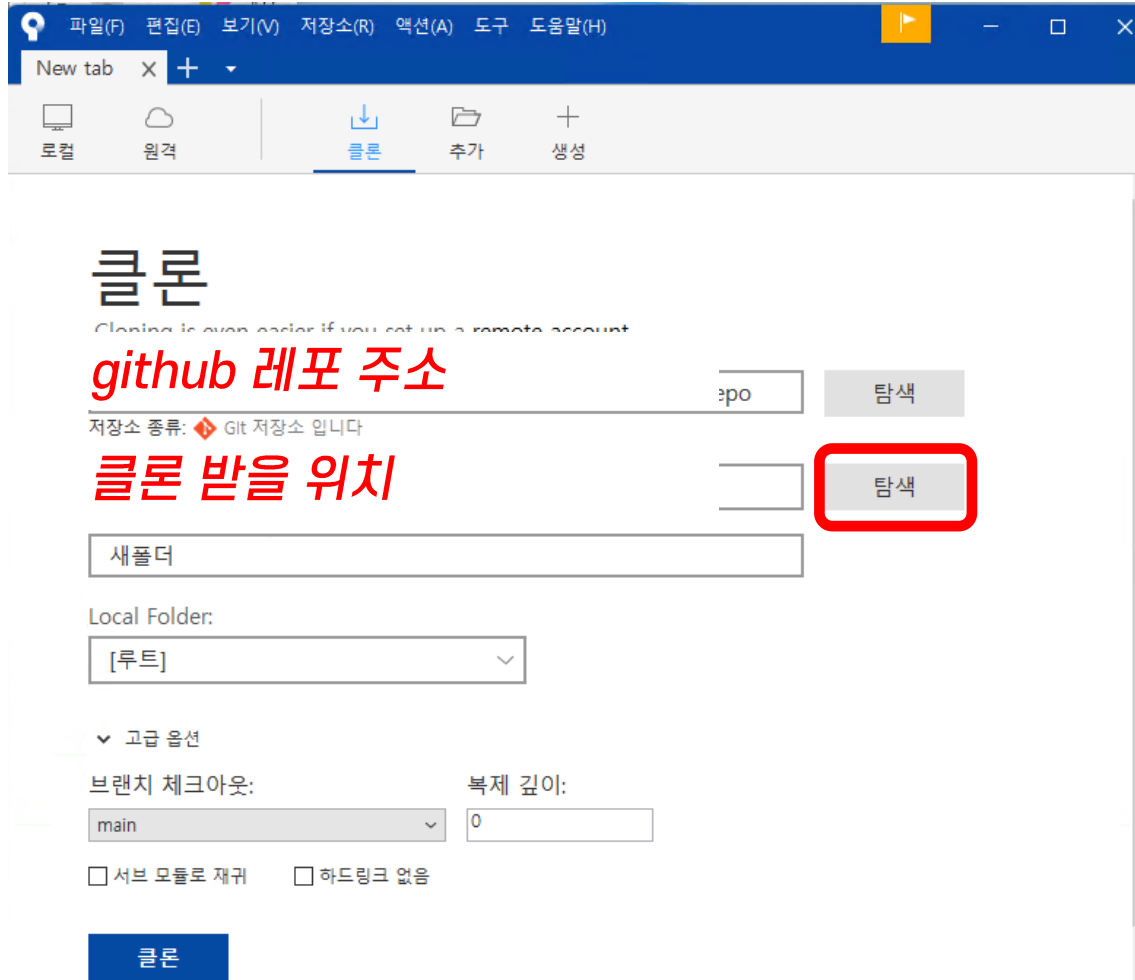
SourceTree, 깃갇프로그램

- 아까 만든 Repo를 불러올 수 있음!



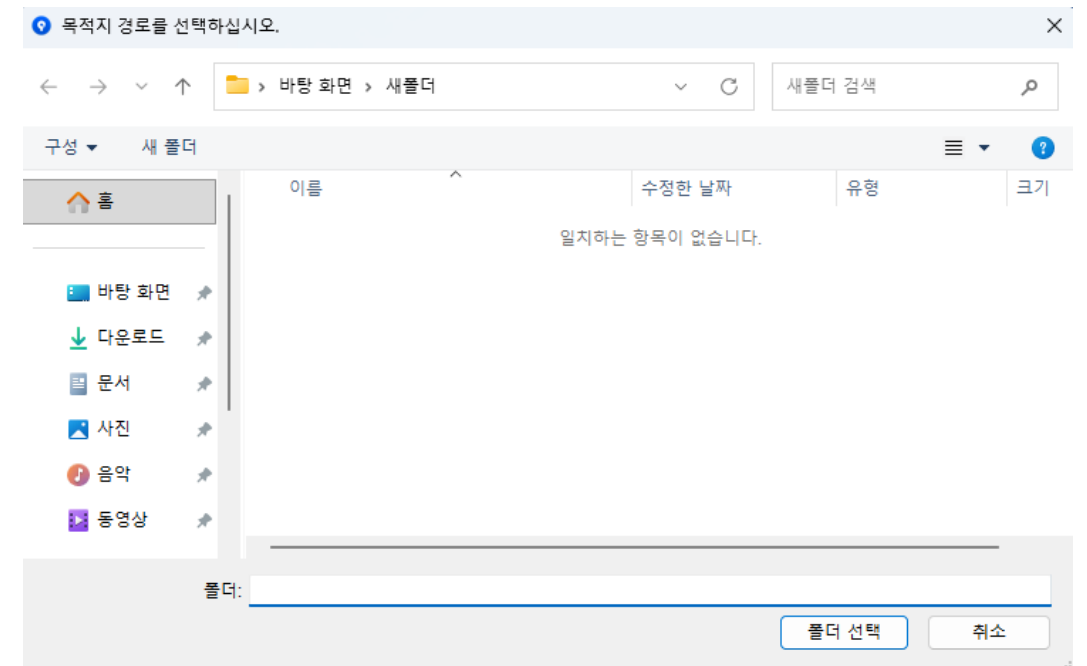
SourceTree, 깃프로그램

- 클론: Repo를 내 컴퓨터에 복사해서 갖고옴



The image shows the 'Clone' dialog box in SourceTree. It has a title bar with standard Windows window controls. Below the title bar is a menu bar with '파일(F)', '편집(E)', '보기(V)', '저장소(R)', '액션(A)', '도구', and '도움말(H)'. Below the menu bar is a toolbar with icons for '로컬', '원격', '클론', '추가', and '생성'. The main area is titled '클론' and contains the following fields and buttons:

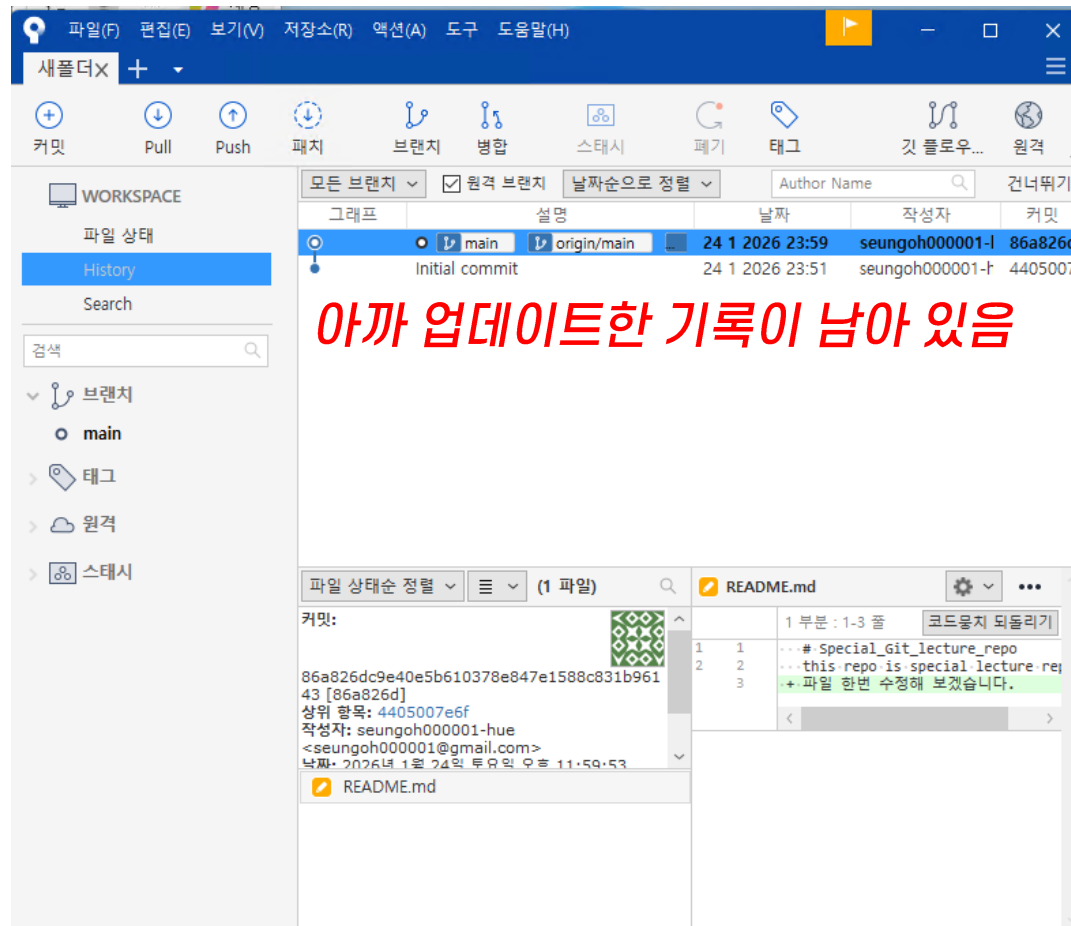
- github 레포 주소**: A text input field with a placeholder 'repo' and a '탐색' (Browse) button.
- 저장소 종류**: A dropdown menu with 'Git 저장소입니다' selected.
- 클론 받을 위치**: A text input field with a placeholder '새폴더' and a '탐색' (Browse) button, which is highlighted with a red rectangle.
- Local Folder**: A dropdown menu with '[루트]' selected.
- 고급 옵션**: A section with a dropdown for '브랜치 체크아웃' (set to 'main') and a text input for '복제 깊이' (set to '0').
- 서브 모듈로 재귀** and **하드링크 없음**: Two checkboxes, both unchecked.
- 클론**: A blue button at the bottom left.



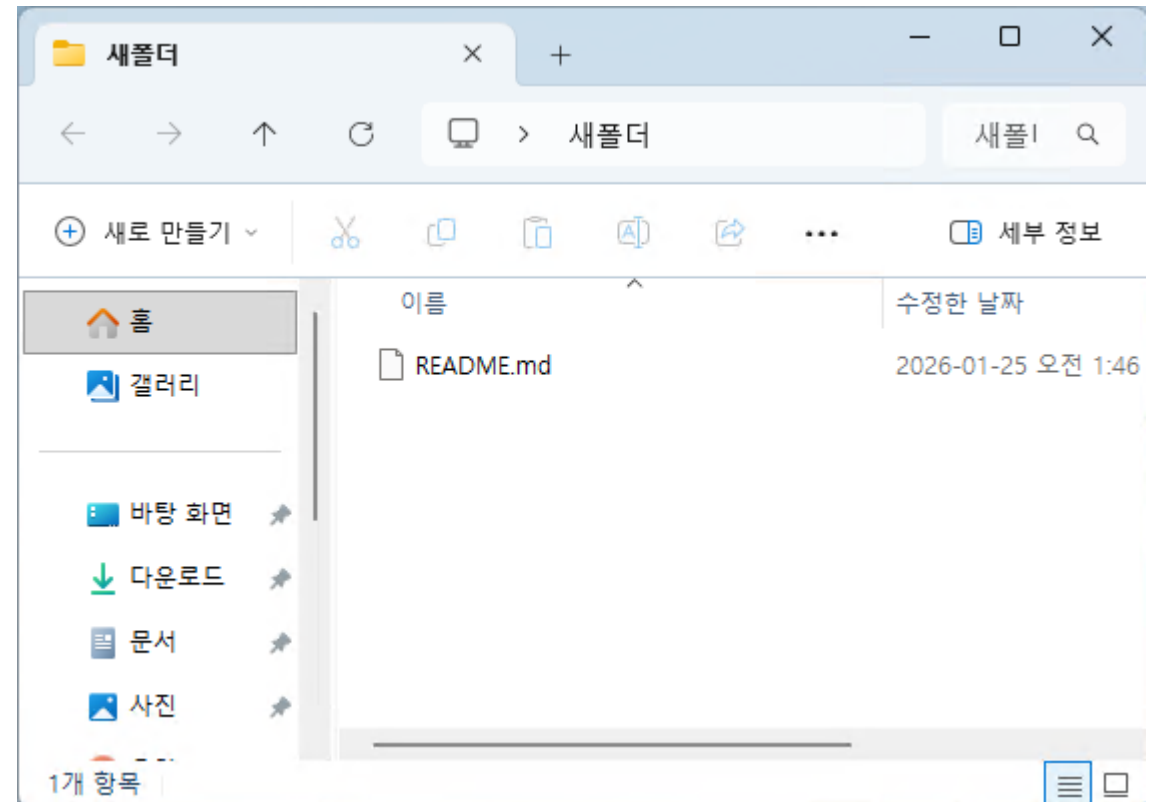
바탕화면에 새폴더 만들어서 폴더 선택

SourceTree, 깃프로그램

- 클론: Repo를 내 컴퓨터에 복사해서 갖고옴



clone된 repo를 볼 수 있음



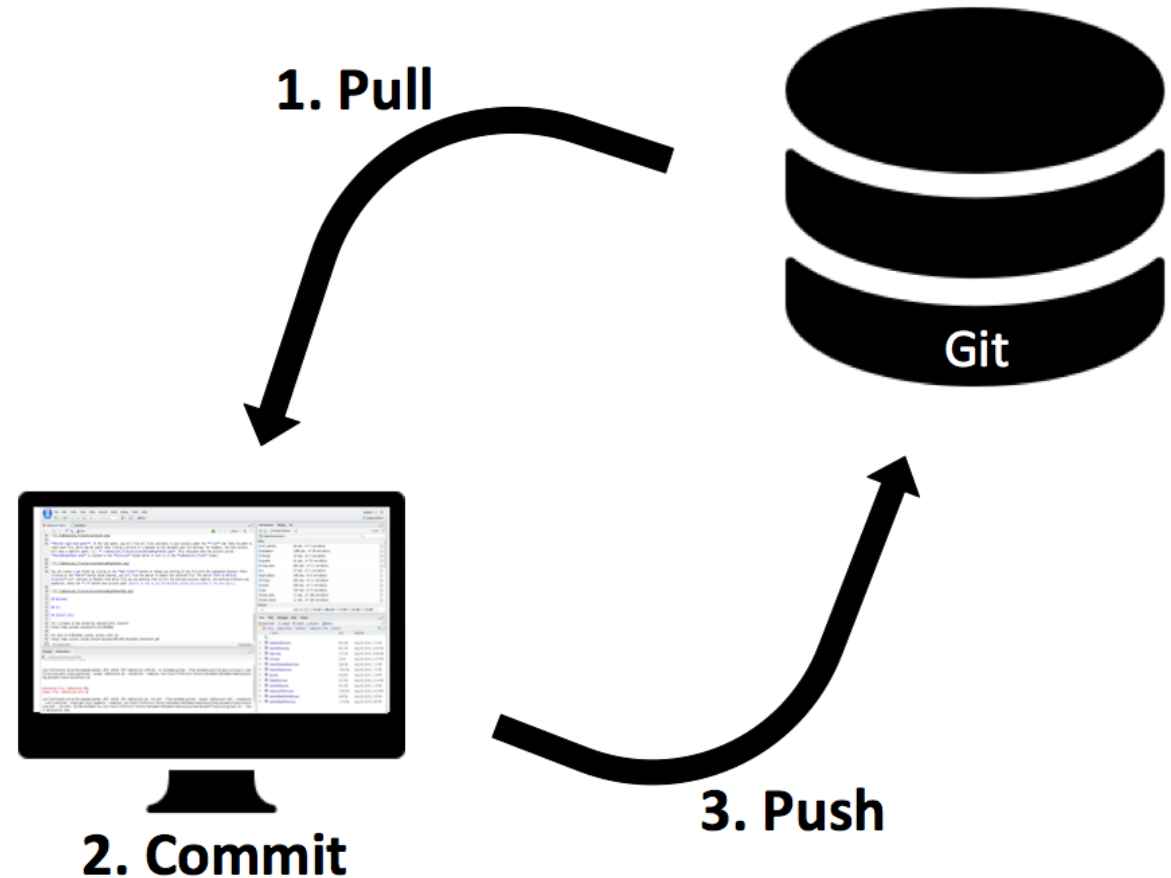
해당 폴더로 가보면 파일도 확인 할 수 있음



Commit & Push

만약 회사에 불이 나면 어쩌지?

- Commit이 뭐지?
 - 내 컴퓨터(로컬)에서 변경된 파일을 내 컴퓨터(로컬)에 기록 하는 것
- Push가 뭐지?
 - commit한 파일을 Github(클라우드)에 올리는 것



만약 회사에 불이 나면 어찌지?

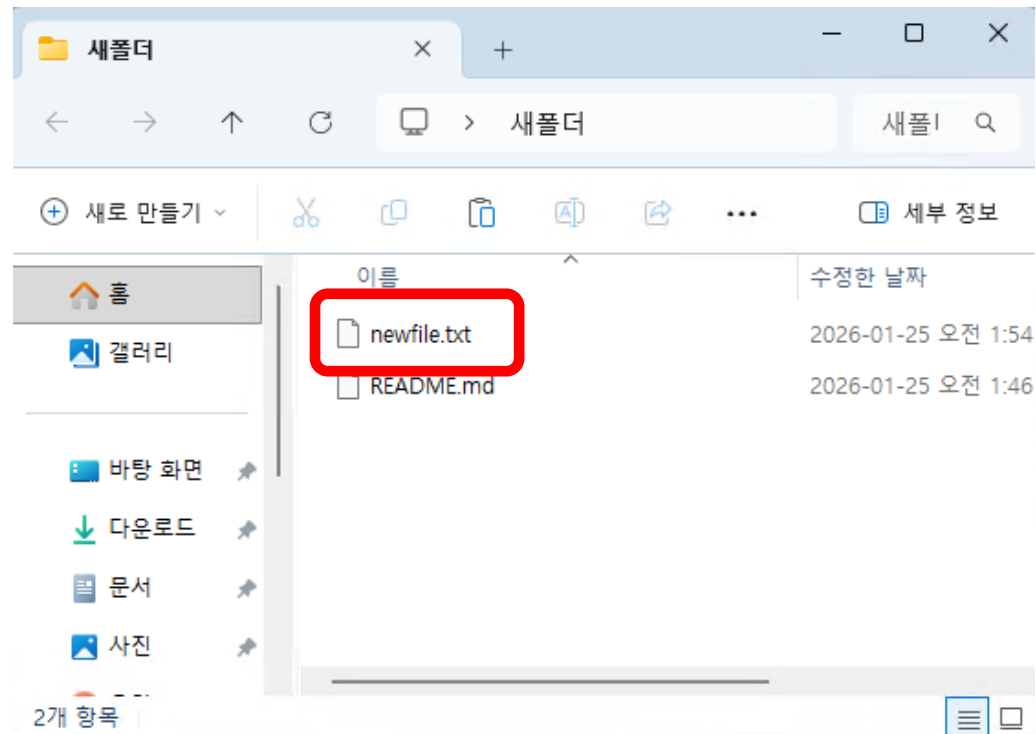
- Commit이 뭐지?
 - 내 컴퓨터(로컬)에서 변경된 파일을
내 컴퓨터(로컬)에 기록 하는 것
- Push가 뭐지?
 - commit한 파일을
Github(클라우드)에 올리는 것



불이 나면, 먼저 커밋 하고, 푸쉬 하고, 건물을 탈출 하세요!

Commit

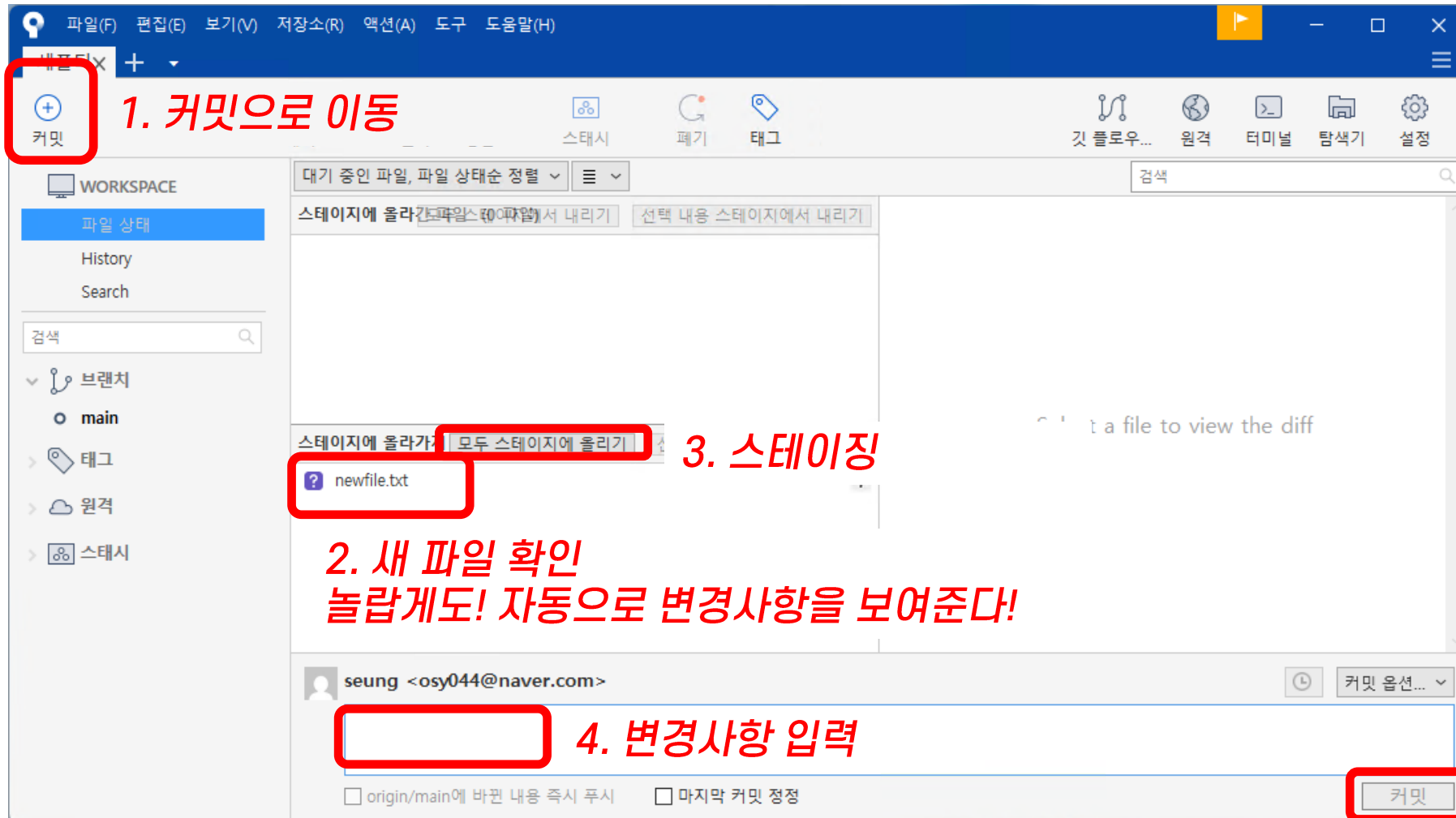
- 이제 폴더에서 새 파일을 추가 해보자!



newfile.txt 추가

Commit

- 추가된 파일 commit 해보기

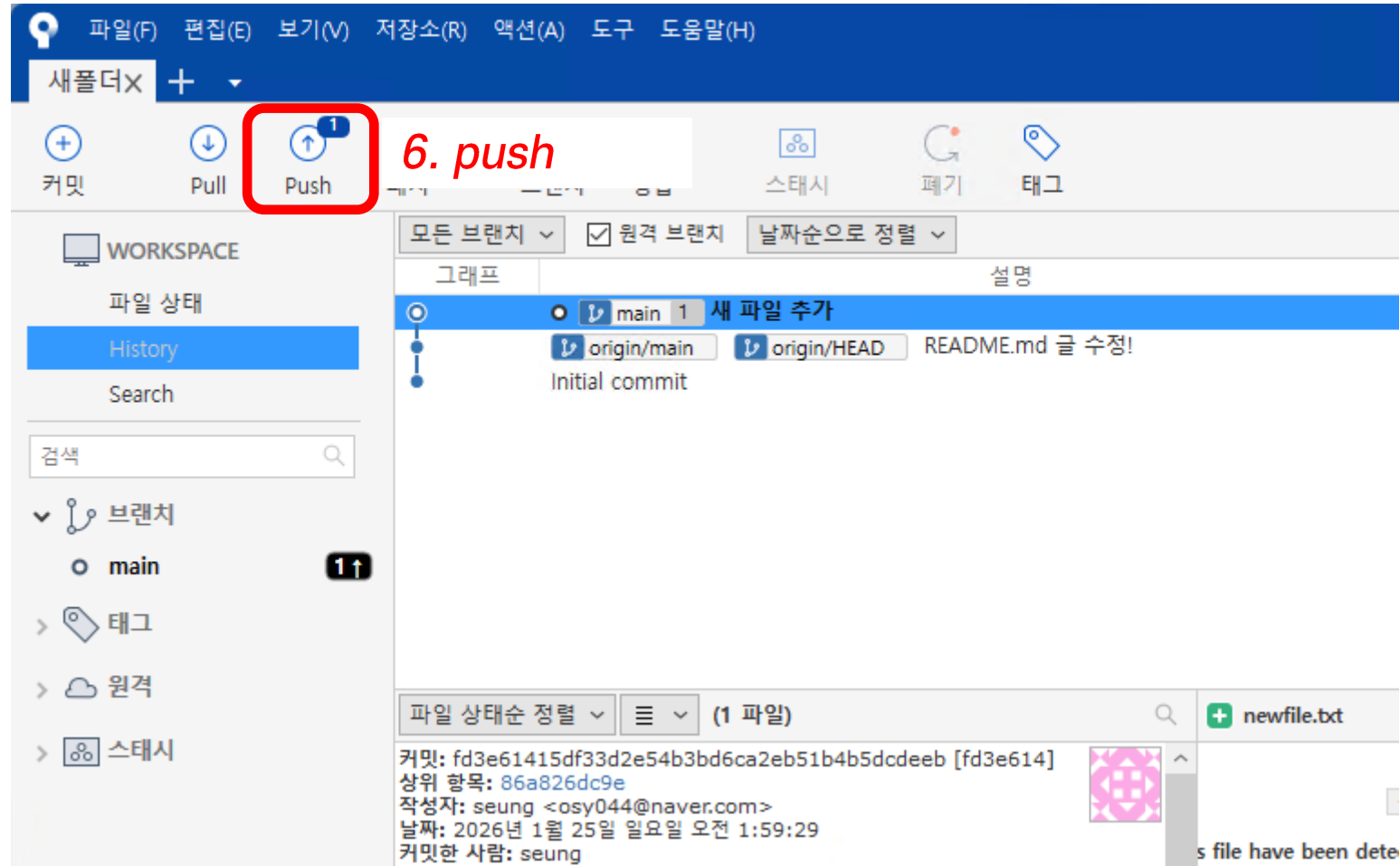


Commit

- Commit을 하더라도 github에 변화가 없음!
- 왜냐하면, 여러분 컴퓨터에 변경사항을 저장한 것이고, 아직 github에는 안올라감

Push

- 추가된 파일 push 해보기



Commit

- Push를 하고나면 변경사항이 github에 반영됨
- 즉, 새로운 파일이 추가됨!

Push

- Push를 하게 되면 최종 파일이 github에 올라가게 됨.
- 그러면 최종본이 바뀌게 됨!
- 즉, 프로젝트가 버전업이 된다!
- push한 것을 다시 되돌릴수는 없다!



아... 실수...망했다...
(잘못된 코드를 push하면 모두가 힘들어짐)



History

과거 시점으로 돌아가기

History

The screenshot shows the Git GUI application. The left sidebar has a 'History' button highlighted with a red box. The main window displays a commit history table with columns for commit hash, date, time, author, and commit message. The 'HEAD' branch is selected, and the 'Initial commit' is visible. A red arrow points to the 'HEAD' branch in the '모든 브랜치' (All Branches) section. Another red arrow points to the 'HEAD' commit in the commit history table. A third red arrow points to the 'HEAD' branch in the '그래프' (Graph) section. A fourth red arrow points to the 'HEAD' commit in the commit history table. A fifth red arrow points to the 'HEAD' commit in the commit history table. A sixth red arrow points to the 'HEAD' commit in the commit history table. A seventh red arrow points to the 'HEAD' commit in the commit history table. A eighth red arrow points to the 'HEAD' commit in the commit history table. A ninth red arrow points to the 'HEAD' commit in the commit history table. A tenth red arrow points to the 'HEAD' commit in the commit history table. A file explorer window is open in the foreground, showing the 'README.md' file. The file's modification date is '2026-01-25 오전 2:04'.

History를 누르면 과거 시점을 볼 수 있다.

혹은 다시 최신 시점으로 이동

원하는 시점을 더블 클릭!

다시 폴더를 보면 파일이 원복되어있음!



Reset (커밋 되돌리기)

Reset

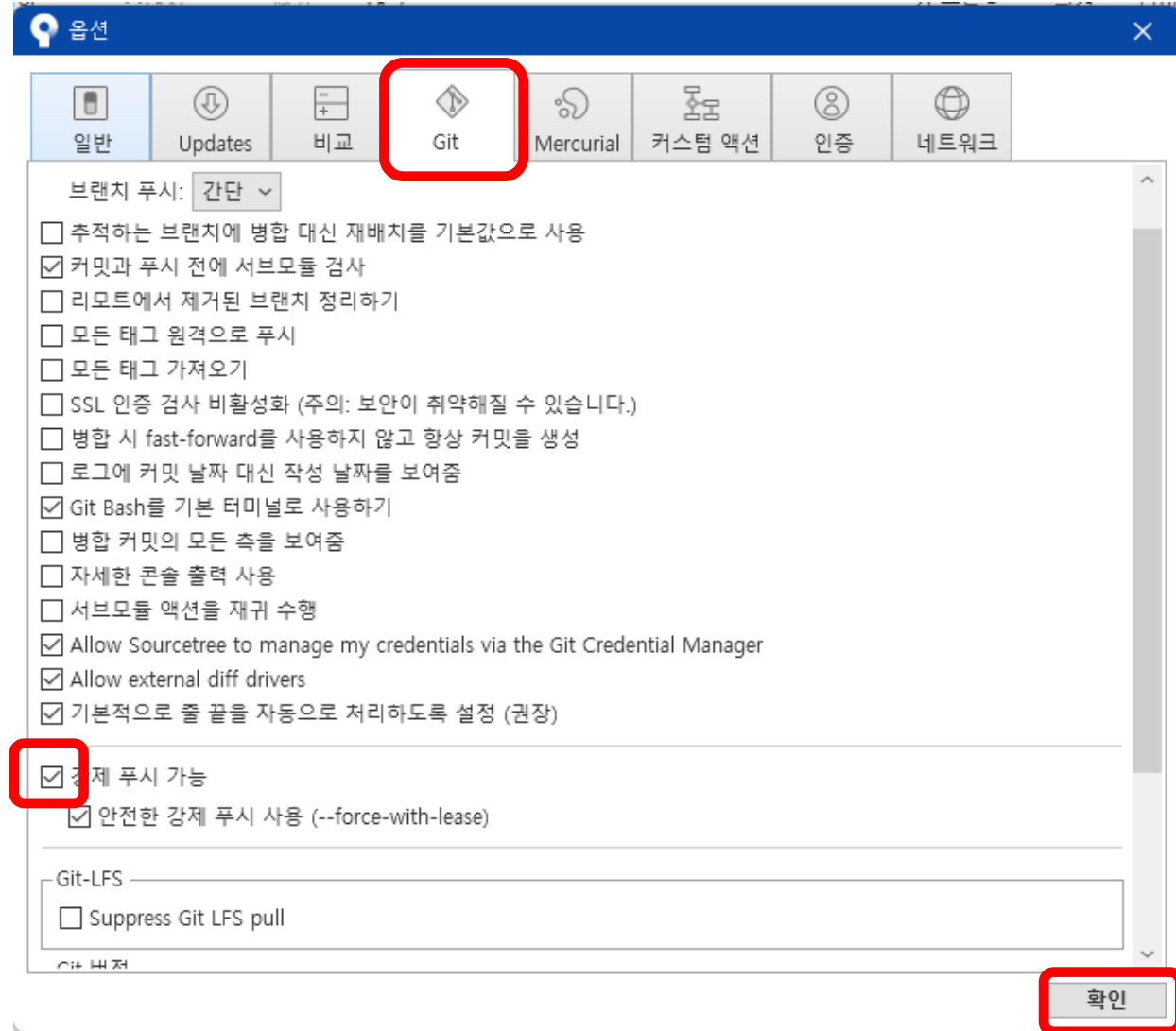
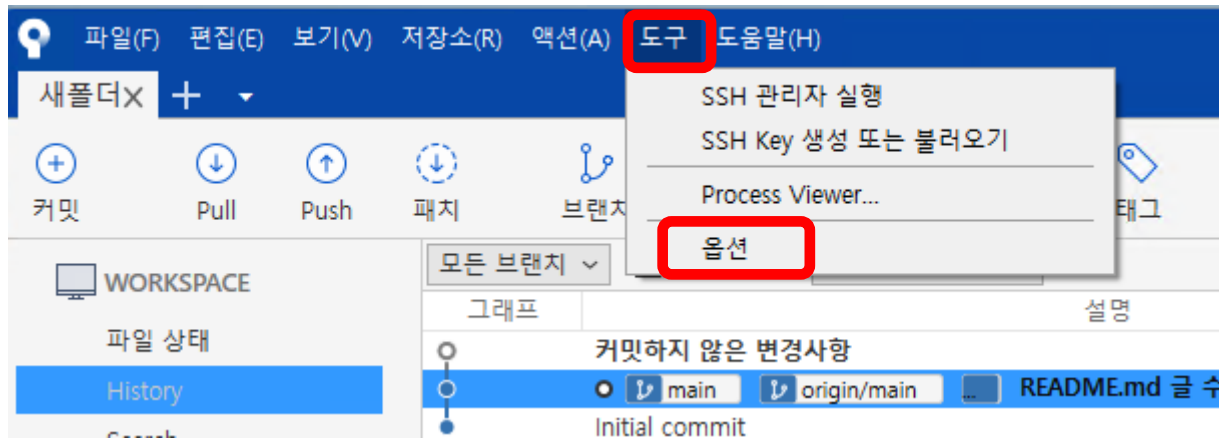
- Reset: 해당 커밋까지 모두 되돌리기
- 실수로 잘못된 파일을 커밋한 경우, 해당 시점까지 되돌리기를 할 수 있다.
- 다만, 해당시점 이후로 commit 했던 파일은 복구가 불가능해진다.



과..과거로 돌아가야해!!

Reset

- Reset은 위험한 기능이기 때문에, 옵션을 바꿔줘야 함



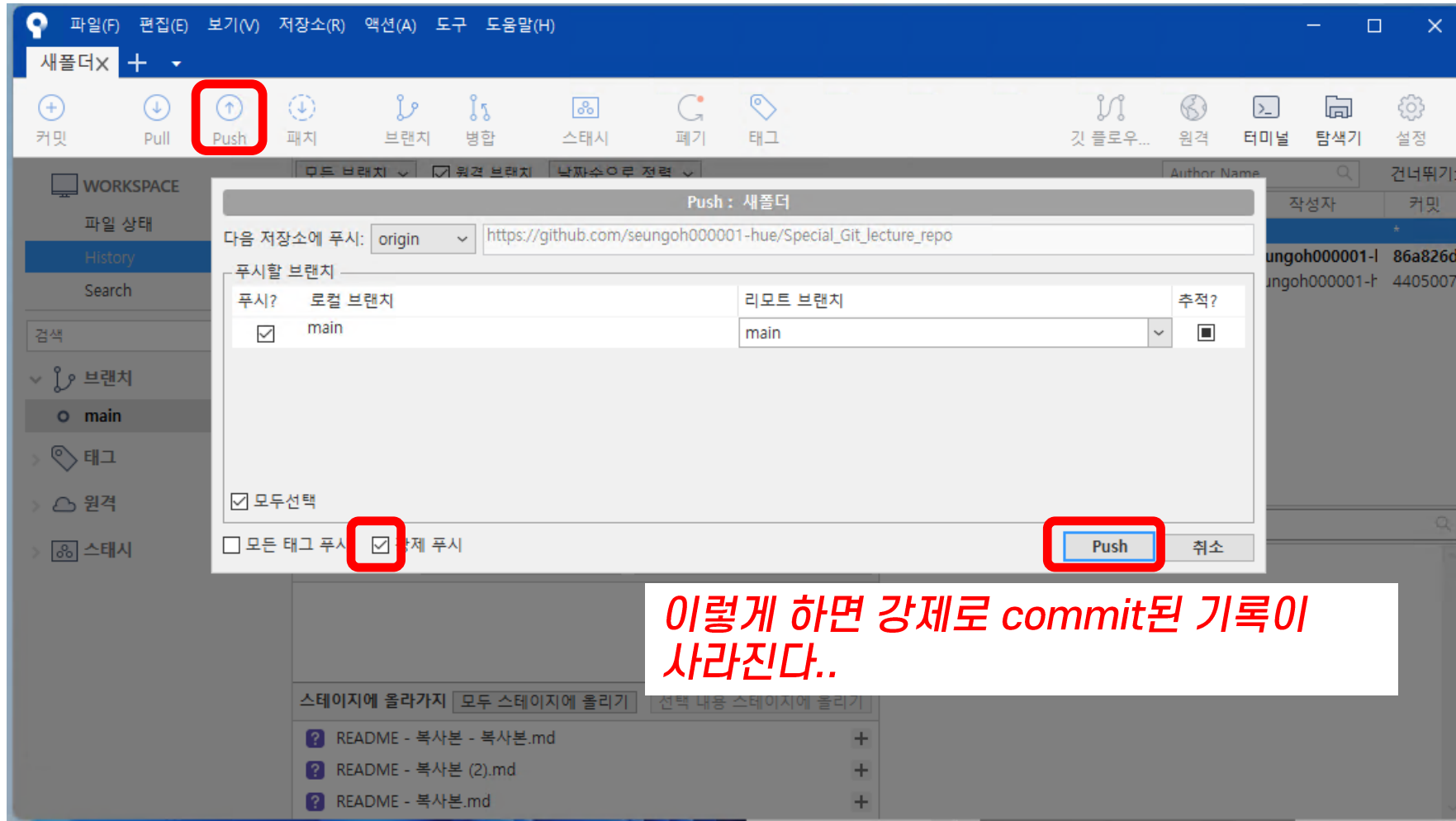
Reset

The screenshot shows the Git GUI interface. The top menu bar includes '파일(F)', '편집(E)', '보기(V)', '저장소(R)', '액션(A)', '도구', and '도움말(H)'. The toolbar contains icons for '커밋', 'Pull', 'Push', '패치', '브랜치', '병합', '스테시', '폐기', '태그', '깃 플로우...', '원격', '터미널', '탐색기', and '설정'. The left sidebar shows 'WORKSPACE' with '파일 상태', 'History', and 'Search'. The main area displays the commit history with columns for commit hash, date, time, author, and commit message. A context menu is open over the commit with hash 86a826d, showing options like '체크아웃...', '병합...', '재배치...', '태그...', '아카이브...', '브랜치...', '86a826d의 자식 커밋을 쌍방향 재배치...', '이 커밋까지 현재 브랜치를 초기화' (highlighted with a red box), '커밋 되돌리기...', '패치 생성...', '처리 픽', 'SHA 값을 클립보드에 복사', '외부 비교도구', and '커스텀 액션'. A red arrow points to the commit 86a826d. A red text overlay says '1. 되돌리고 싶은 커밋을 마우스 우클릭'. Another red text overlay says '이 커밋까지 현재 브랜치를 초기화 클릭'. The bottom right shows a diff view with a green highlight on the line '+ 파일 한번 수정해 보겠습니다.'

1. 되돌리고 싶은 커밋을 마우스 우클릭

이 커밋까지 현재 브랜치를 초기화 클릭

Reset



Revert (특정 커밋 하나만 되돌리기)

Revert

- Revert: 되돌리기
- 실수로 잘못된 파일을 커밋한 경우, 해당 커밋만 다시 되돌리기를 할 수 있다.
- 정확히 말하면, 해당 변경사항만, 다시 원복시키고, 다시 커밋하는것!
- Reset 보다 안전하다.
- 과거 기록을 그대로 남기고 최신 파일에 덮어쓰는것이기 때문
- (하지만 reset이 편하긴 하다..)

Revert

The screenshot shows the Git GUI interface. At the top, the menu bar includes '파일(F)', '편집(E)', '보기(V)', '저장소(R)', '액션(A)', '도구', and '도움말(H)'. Below the menu bar, there's a toolbar with '커밋', 'Pull', and 'Push' buttons. The 'Push' button is highlighted with a red circle and a '1' icon. To the right of the toolbar, there's a red text annotation: '3. push 클릭 → 해당 commit의 수정사항이 다시 덮어쓰기 됨'.

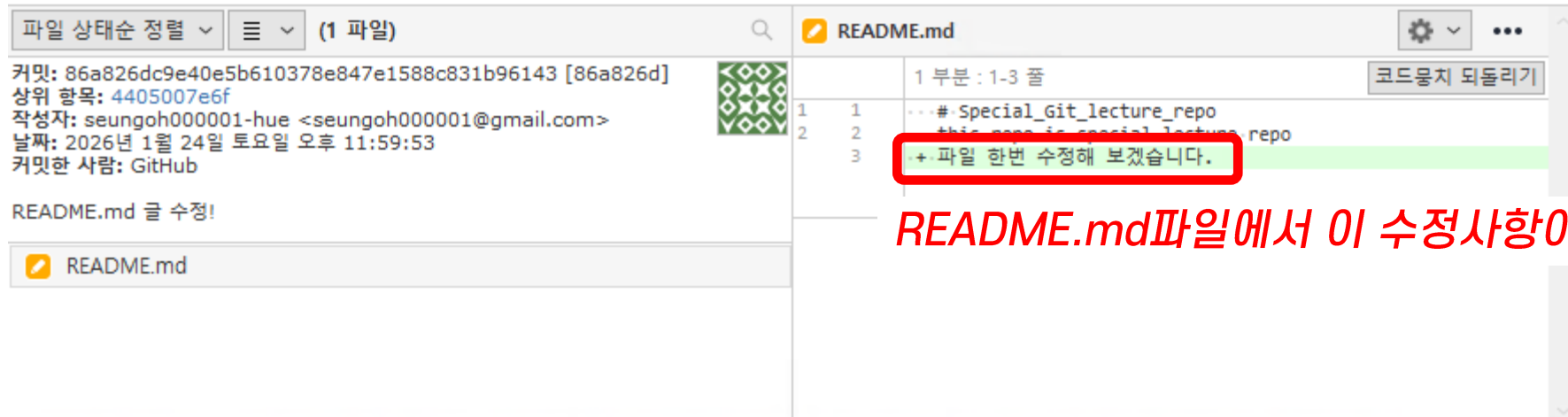
On the left sidebar, the 'WORKSPACE' section shows '파일 상태', 'History', and 'Search'. The '브랜치' section shows 'main' with a '1' icon. The '태그' and '원격' sections are also visible.


The main area shows a commit history graph. The 'main' branch is selected, and the 'Initial commit' is highlighted. A red arrow points to the 'Initial commit' with a red text annotation: '1. 되돌리고 싶은 커밋을 마우스 우클릭 주의** 첫번째 커밋(initial commit)은 안됨!'. A context menu is open over the 'Initial commit', showing options like '재크아웃...', '병합...', '재배치...', '태그...', '아카이브...', '브랜치...', '86a826d의 자식 커밋을 쌍방향 재배치...', '이 커밋까지 현재 브랜치로 초기화', '커밋 되돌리기...', '패시 생성...', '체리 픽', 'SHA 값을 클립보드에 복사', '외부 비교도구', and '커스텀 액션'. The '커밋 되돌리기...' option is highlighted with a red circle and a red text annotation: '2. 커밋 되돌리기 클릭'.

At the bottom, the 'README.md' file is shown with its commit history. The commit hash '86a826dc9e40e5b610378e847e1588c8' is visible, along with the author 'seungoh000001-hue' and the date '2026년 1월 24일 토요일 오후 11:59:53'.

Revert

- 해당 커밋의 수정사항만 원복되었다!

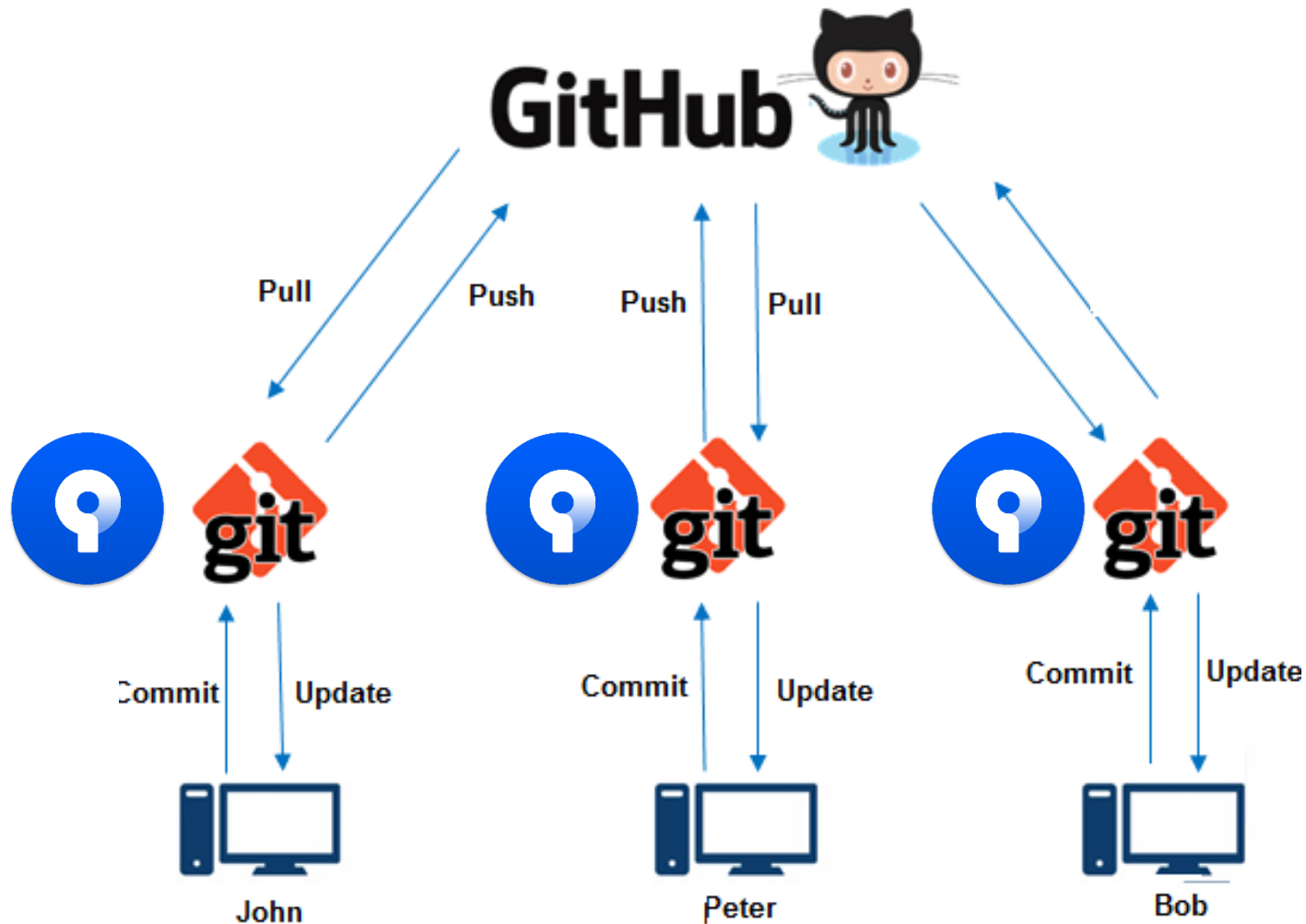




지금까지 한 것 정리

혼자 쓸때는 이정도만 알면 됨

Distributed Version Control System



- Commit
 - 내 컴퓨터에서 변경된 파일을 업데이트(기록) 하는 것
- Push
 - commit한 파일을 Github에 올리는 것
- Pull
 - 내 컴퓨터에 변경사항을 받는 것
- Revert
 - 해당 commit 다시 되돌리기

미션1

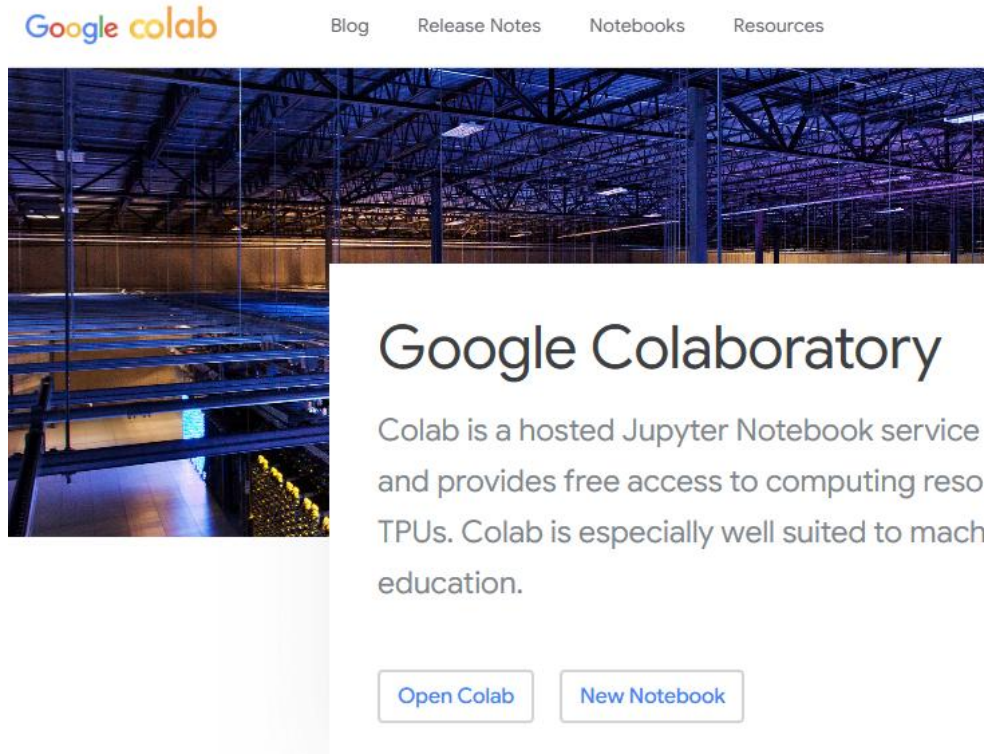
**README.md 파일에 자기 소개
작성해서 commit, push 해보기**

미션2

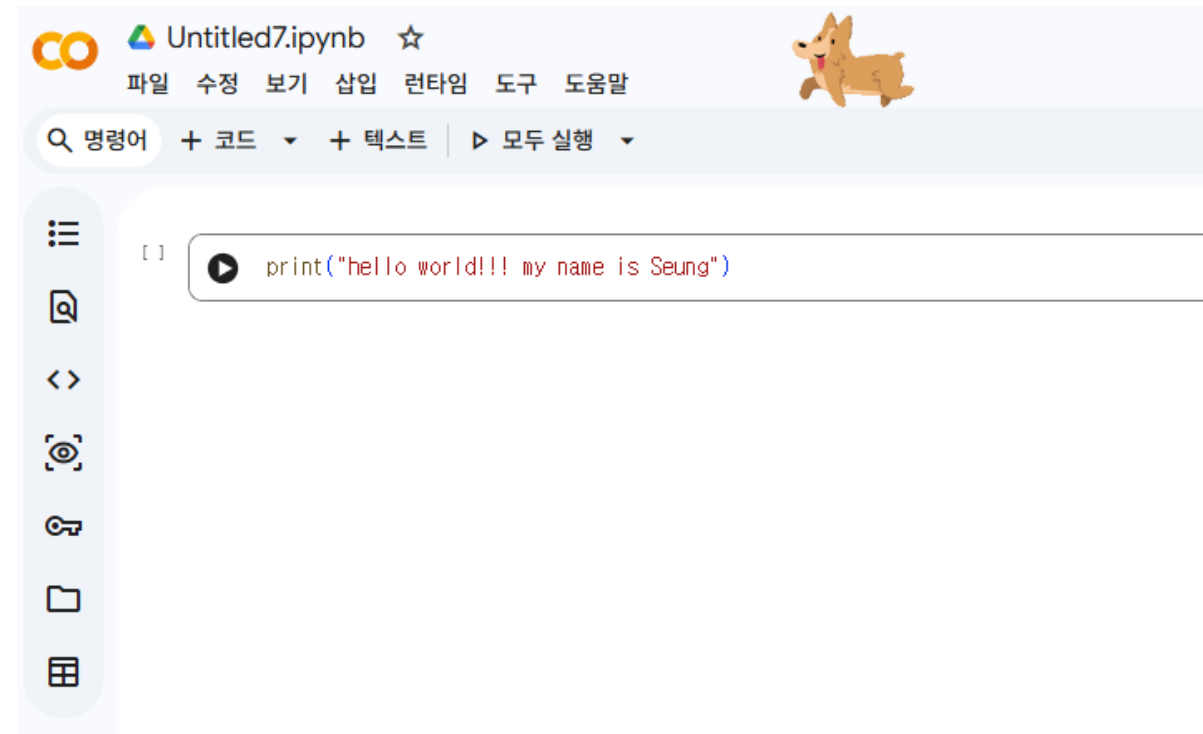
**코랩에서 python 코드 작성해서
내 git repo에 올리기**

Colab

- 구글에서 colab 검색 - new notebook 클릭



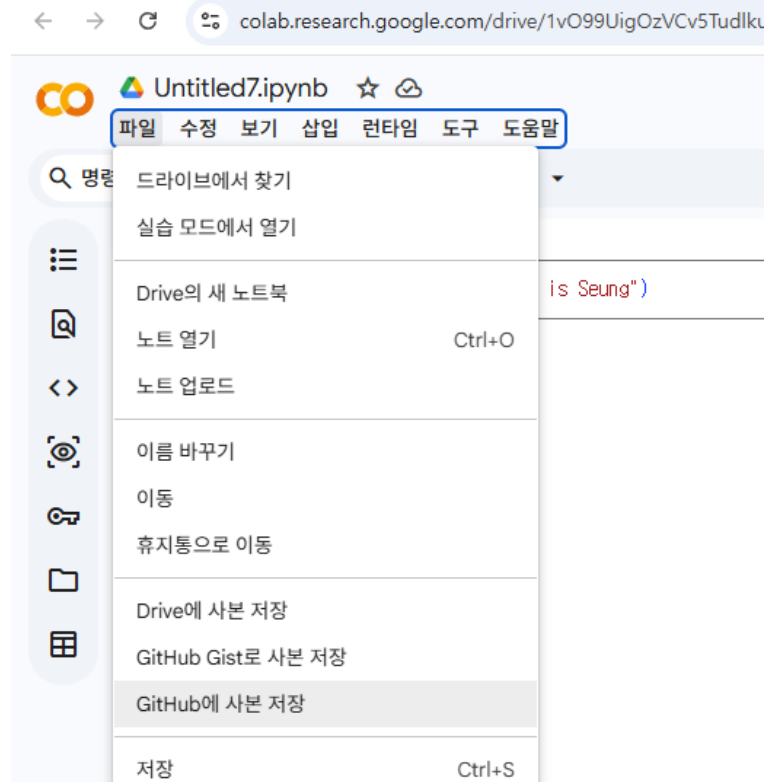
colab 접속



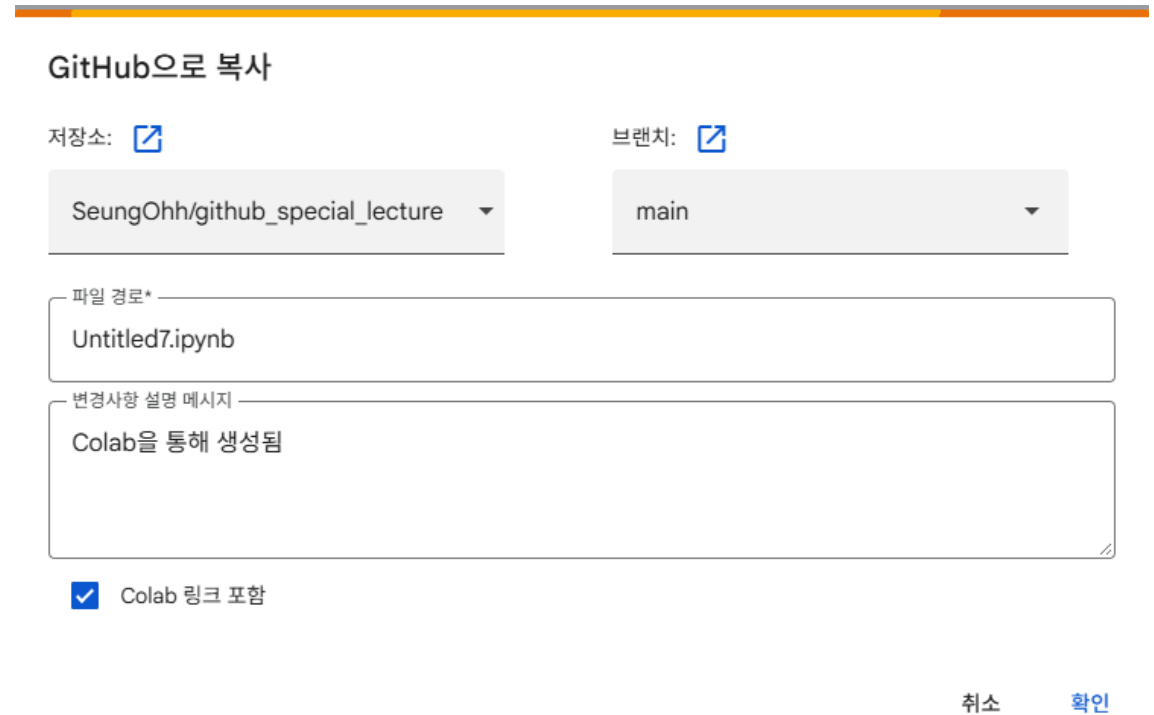
코드 작성

Colab

- 구글에서 colab 검색 - new notebook 클릭




github에 사본 저장







사본 저장

확인!

- 코랩 파일 올라간 것 확인!

 **github_special_lecture** PublicUnpinWatch

main 1 Branch 0 Tagst+Cloud icon<> Code

 SeungOhh Colab을 통해 생성됨	d7264ef · 10 minutes ago	 2 Commits
 README.md	Initial commit	2 hours ago
 Untitled6.ipynb	Colab을 통해 생성됨	10 minutes ago