



Git 정복하길! - 기본



본 자료는 스파르타코딩클럽 핵심 쪽쪽 Git 강의에서 발췌한 강의 자료입니다.

[가이드 목표]

1. 나 혼자 사용할 수 있는 Git 프로젝트를 만들어 본다 - commit, pull, push
2. 프로그래밍을 배울 때 마음가짐을 탑재한다.

[목차]

- [01. Git은 뭐고, Github은 무엇인가요?](#)
- [02. 버전관리와 commit - 개념tap재](#)
- [03. 버전관리와 commit - 실습 01](#)
- [04. 버전관리와 commit - 실습 02](#)
- [05. 버전관리와 commit - 정리](#)
- [06. 원격 repo 사용하기 - 개념tap재](#)
- [07. 원격 repo 사용하기 - 실습](#)
- [08. 원격 repo 사용하기 - 정리](#)

01. Git은 뭐고, Github은 무엇인가요?

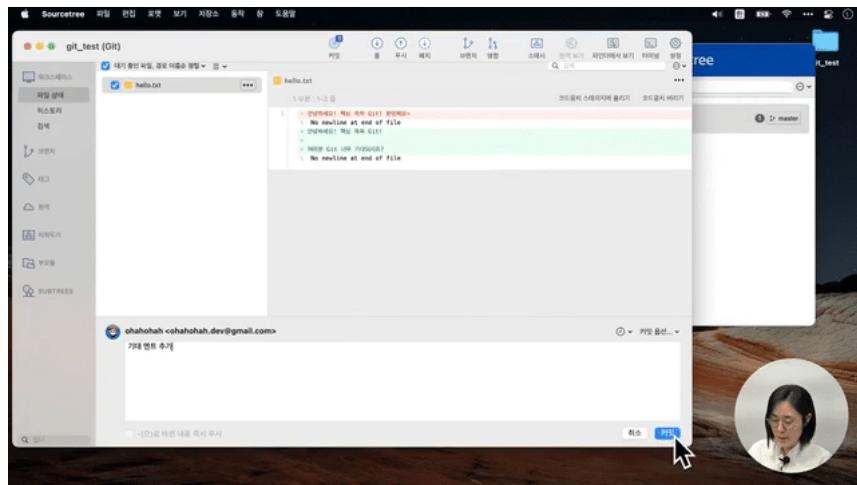


git이 무엇인지 살펴볼까요? 핵심 원리를 이해하고 있으면 문제를 해결할 때 도움이 되어요.

▼ 8) 배우고 나면 할 수 있는 것 먼저 보여드릴게요!



지금은 뭐가 뭔지 몰라도 괜찮아요! 강의 영상을 참고해주세요!



▼ 9) Git으로 무엇을 할 수 있을까요?

- Git은 프로젝트의 버전 관리를 위한 도구입니다. 자세한 방법은 다음 챕터에서 배워볼게요! 먼저 어떤 걸 할 수 있는지부터 봅시다!

1. 버전 관리를 할 수 있어요!

Name	Date Modified
1분기보고서.ppt	Today 7:30 PM
1분기보고서_컨펌버전.ppt	Today 7:30 PM
1분기보고서_최종.ppt	Today 7:30 PM
1분기보고서_최종_최종.ppt	Today 7:31 PM
1분기보고서_최종_최종_최종.ppt	Today 7:31 PM
1분기보고서_진짜최종.ppt	Today 7:31 PM
1분기보고서_발표본.ppt	Today 7:32 PM
1분기보고서_발표본_최종.ppt	Today 7:32 PM

- 더 이상 이런 파일은 그만! 하나의 파일로도 버전관리를 할 수 있게 도와줘요.
- Git을 사용하면 무슨 작업을 했는지도 히스토리도 한 눈에 볼 수 있답니다.

대한민국 헌법 제10호 - 1987.10.29 전부개정/ 6월 민주화 항쟁
ohahohah committed 3 days ago

대한민국 헌법 제9호 - 1980.10.27 전부개정/ 10.26 사건, 12.12 군사반란
ohahohah committed 3 days ago

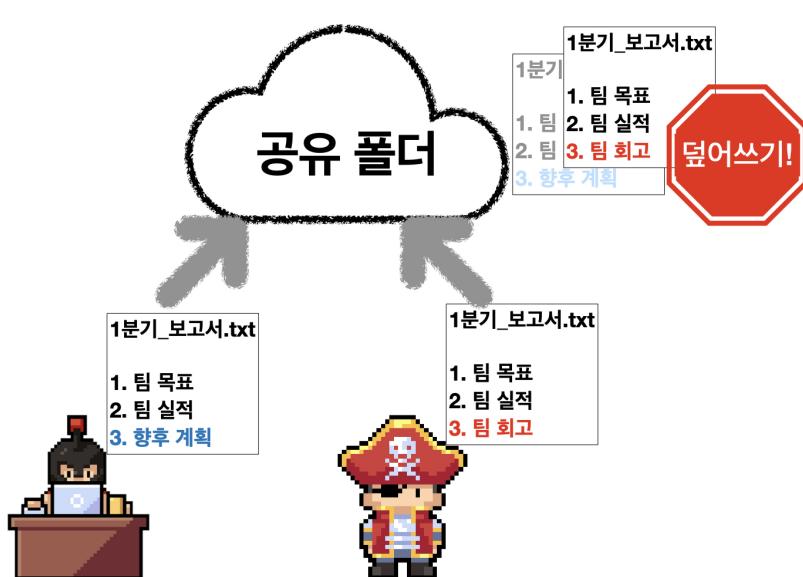
대한민국 헌법 제8호 - 1972.12.27 전부개정/ 10월 유신
ohahohah committed 3 days ago

대한민국 헌법 제7호 - 1969.10.21 일부개정
ohahohah committed 3 days ago

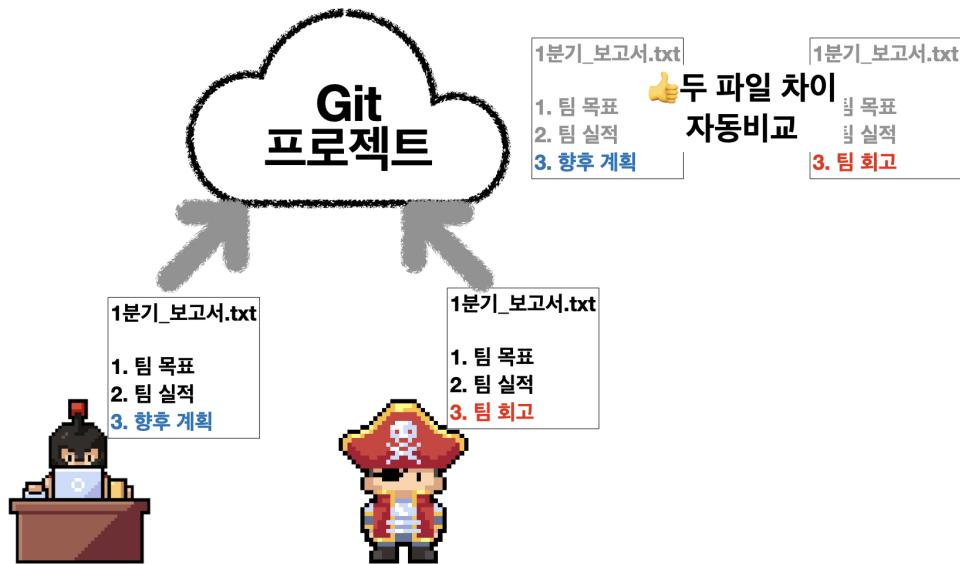
대한민국 헌법 개정내역 <https://github.com/ohahohah/constitution-of-republic-of-korea/commits/main>

2. 작업 단위 나누기

- 프로그래밍 하다보면 분명 아끼는 되었는데 지금 코드 고치니까 프로젝트가 동작 안하는 순간이 있어요. 기능을 완성할 때마다 작업 내역을 저장하면 어떤 부분을 만들 때 에러가 발생했는지 쉽게 파악할 수 있어요.
- 협업해서 하나의 프로젝트를 만드는데 유용해요.
- 프로젝트를 나누어서 작업하고 하나로 합치는 것이 편합니다. 누가, 언제, 어떤 부분을 수정했는지를 한 눈에 파악할 수 있어요.
- 만약 Git을 사용하지 않고 프로젝트 파일을 덮어쓰는 형태로 관리한다면 아래 같은 경우가 발생할 수 있겠죠! 다른 사람이 작업한 내용을 내 파일로 덮어써버리는 비극이 발생할 수 있어요 😱



- Git을 사용하면 같은 파일명의 내용이 어떤 부분이 다른지를 자동으로 비교하고, 어떤 것을 반영할지 선택할 수 있어요.



👉 Git 으로 모든 파일의 내용이 자동 비교가 되나요?
기본 설정으로는 코드(Python, HTML, JavaScript, Java,...) text 파일, markdown파일(text 파일의 일종), CSV 파일 등이 가능합니다!

이미지 파일, Word 파일, PDF 파일, 엑셀 파일은 여러가지 설정을 해주어야 가능하답니다. 기본 설정으로 이런 파일들은 파일의 크기가 변했구나! 만 알 수 있어요~

▼ 10) Github 으로 무엇을 할 수 있을까요?

- Git 과 Github 은 다릅니다! Github 은 Git 원격 저장소 + Git 으로 할 수 있는 커뮤니티 기능 서비스입니다.
- 즉, Github 은 Git 으로 된 프로젝트 저장 공간을 제공하고, Git 편하게 사용하기 위한 여러가지 부가기능을 가지고 있어요. Git 이 협업할 때 필수! 라고 했었죠? Github 에는 협업하기 위한 기능들을 가지고 있어요. 마치 개발자들의 SNS 같답니다.
- Github 외에도 Git 프로젝트 저장소 + 프로젝트 관리하는 기능을 제공하는 곳으로는 대표적으로 Gitlab, bitbucket 등의 서비스 가 있어요.

1. 인터넷으로 연결되어있는 프로젝트 저장소

- 컴퓨터에 있는 내 Git 프로젝트를 저장하기! 이어지는 시간에 곧! 배울 꺼예요.

About
김치찌개 저장소

Releases
No releases published
[Create a new release](#)

Packages
No packages published
[Publish your first package](#)

- 개발해야할 기능들을 관리하기

The screenshot shows a GitHub repository's Issues page. At the top, there are navigation links: Code, Issues 16 (which is highlighted), Pull requests, Actions, Projects 1, Wiki, Security, Insights, and Settings. Below the navigation is a search bar with the query "is:issue is:open". To the right of the search bar are buttons for Labels (12), Milestones (0), and New issue. A dropdown menu labeled "Filters" is open, showing the current filter "is:issue is:open". The main area displays 16 open issues, each with a checkbox, a title, a description, and a timestamp. The issues are categorized by labels: enhancement, security, test, and documentation. Some issues have a small icon next to them.

Issue #	Title	Description	Label	Comments
#23	목표 미달성시 공약지기라고 슬랙 메시지 엔션	#23 opened 14 days ago by ohahohah		
#22	github 조회 event api 가 나은지 확인해보자	enhancement		
#21	github 내용 가져오기 외부 패키지 사용으로 변경			
#20	실행환경 docker 로 만들기	enhancement		
#19	웹 페이지 보안 추가	security		
#18	개인 repo 로 fork 해 개발			
#17	화면 error 발생 log 처리	test		1
#16	demo 페이지 만들기	documentation		

2. 개발자들의 커뮤니티

- 다른 사람 공개 프로젝트 구경하고, 내가 관심있는 주제/ 프로젝트 소식 받아볼 수 있어요.

Here's what we found based on your interests...

Based on topics you've starred

[scalastic](#)

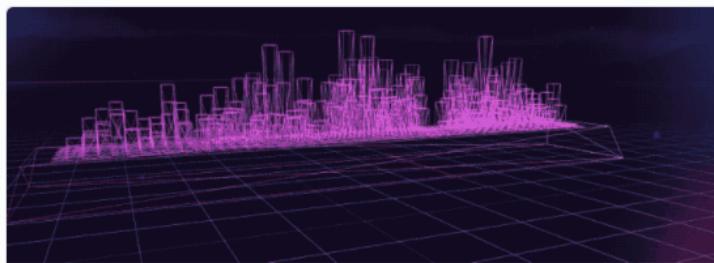
[Star](#)

There are 1 public repositories matching this topic

[scalastic / scalastic.github.io](#)

Code source du site web de la société Scalastic

[See more matching repositories](#)



Staff pick

[GitHub Skyline](#)

View a 3D model of your GitHub contribution graph. Share it, print it, and more!



<https://github.com/explore>

Trending repositories today

[rms-support-letter / rms-support-letter.github.io](#)

An open letter in support of Richard Matthew Stallman being reinstated by the Free Software Foundation

[rails / rails](#)

48k

Ruby on Rails

[ClearURLs / Addon](#)

719

ClearURLs is an add-on based on the new WebExtensions technology and will automatically remove tracking elements from URLs to help protec...

[alexgurr / react-coding-challenges](#)

640

A series of ReactJS coding challenges with a variety of difficulties.

[See more trending repositories >](#)

Trending developers



[Matt \(IPv4\) Cowley](#)

MattIPv4

[DNS-over-Discord](#)



[Stefan Prodan](#)

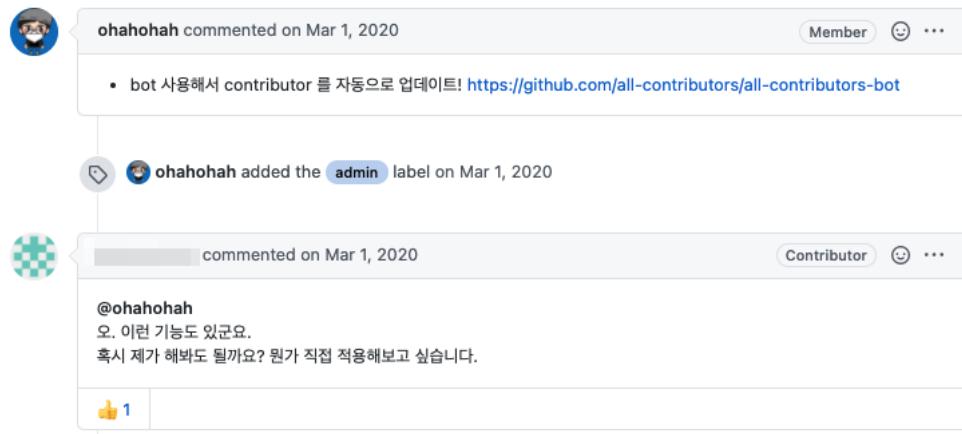
stefanprodan

[podinfo](#)

- 프로젝트 함께 만드는데 참여하는 것 즉 '프로젝트에 기여하기(contribution)' 하기 위한 여러 기능도 제공합니다. 이 부분 버그(프로그램 오류, 오작동)가 있어요! 알리고 프로젝트를 개선시키려면 어떻게 필요할까? 토의할 수도 있어요.

contributor 를 README.md 에 노출합시다! #19

 ohahohah opened this issue on Mar 1, 2020 · 9 comments



ohahohah commented on Mar 1, 2020

Member

• bot 사용해서 contributor 를 자동으로 업데이트! <https://github.com/all-contributors/all-contributors-bot>

ohahohah added the admin label on Mar 1, 2020

commented on Mar 1, 2020

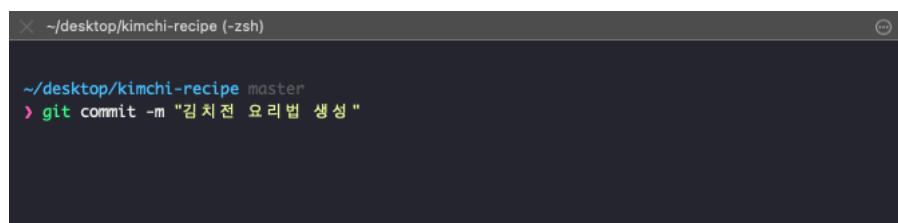
Contributor

@ohahohah
오. 이런 기능도 있군요.
혹시 제가 해봐도 될까요? 누가 직접 적용해보고 싶습니다.

1

▼ 11) sourcetree 는 뭐죠?

- sourcetree 는 Git 을 쉽게 사용할 수 있는 도구입니다. 워드 파일을 편집할 때 MS office 를 사용하는 것처럼요!
- 🔥 우리가 sourcetree 로 배우는 이유!
 1. 전 세계에서 가장 많이 사용되는 Git 도구 중에 하나입니다. 스스로 공부할 때도 참고할 만한 자료가 많습니다.
 2. 초심자부터 중급자까지 사용할 수 있도록 여러가지 기능을 제공합니다. 한 번 배워두면 편하겠죠?
 3. Git 사용 패턴에 집중하기 위해서!
 - 우리의 목표는 Git 사용 패턴을 이해하고 익숙해지는 것! 사용법 암기가 아니랍니다. 명령어 사용하는 것까지 한 번에 둘 다 배우면 우리가 진짜로 배우고 싶은 Git 사용에 집중하지 못할 수 있어요!
- Git 은 여러가지 방법으로 사용할 수 있어요. 아래처럼 터미널에 명령어를 직접 입력할 수도 있죠(이런 방법을 CLI 라고 불러요). 3주차에 다른 Git 사용 도구를 살짝 구경할 꺼에요~!



▼ 12) 개발자 / IT 조직에서는 Git과 Github 을 이렇게 사용해요.

 Git 만, Github 만 따로 사용 🤷!
Git + Github 혹은 Git + 프로젝트 관리 도구 를 조합해서 사용하는 경우가 대부분이에요!

- 대부분의 회사에서 Git 과 같은 버전관리 툴을 필수적으로 사용합니다.
- 프로젝트 작업내역 관리하기
- 여러 개발자들이 하나의 제품을 만들 때 각자 작업한 코드를 합치기
- 버그 리포트 받고 해결하기
 - 작업내역을 확인해서 어제까지는 분명 잘 되었는데 오늘 작업하니까 에러가 나네? 아하, 그럼 오늘 작업한 내용 중에 버그를 만드는 코드가 있겠구나. 어떤 게 어제 작업 내용이고 어떤 것이 오늘 작업 내용이지 확인하자. 필요하다면 어제 만든 버전으로 되돌리자.

- Github 에 있는 저장소를 웹 사이트로 만들기
- ▼ 13) [💡 배웠으면 써먹자] - Github 프로젝트 구경하자
- Github 에 있는 프로젝트 구경해보세요. 아래를 클릭!
- ▼ [코드스니펫] Github 대한민국 헌법 개정이력 프로젝트
<https://github.com/ohahohah/constitution-of-republic-of-korea>
- ▼ [코드스니펫] Github 프로젝트 - 모던 자바스크립트 튜토리얼 번역
<https://github.com/javascript-tutorial/ko.javascript.info>
- ▼ [코드스니펫] Github 프로젝트 - 주니어 개발자를 위한 취업 정보
<https://github.com/jojoldu/junior-recruit-scheduler>
- ▼ [코드스니펫] Github 프로젝트 - 개발자 채용 가이드북
<https://github.com/innovationacademy-kr/tech-hr>
- ▼ [코드스니펫] Github 프로젝트 - 파이콘 한국(파이썬 개발 컨퍼런스) 녹화 영상 자막
<https://github.com/pythonkr/pyconkr-script>
- ▼ [코드스니펫] Github 프로젝트 - 타코 요리법 모음
<https://github.com/sinker/tacofancy>.

02. 버전관리와 commit - 개념탐재



하나 하나 사용방법을 외우는게 아니라 어떻게 사용하는지 흐름에 집중합시다!
 앞으로 개념탐재 - 실습 - 정리하기 순으로 배워나갑니다!

- ▼ 14) Git 은 어떻게 버전관리를 할까?
- Git은 버전관리 도구라고 했습니다. 그렇다면 버전을 어떻게 관리한다는 것일까요?
 - 버전1과 버전2가 다른지 어떻게 알 수 있을까요? 누가, 언제, 해당 버전의 프로젝트의 파일이 무엇인지 정보가 있으면 되지 않나요?
 - 그럼 이렇게 버전을 만들어주려면 아래 그림처럼 버전1, 버전2, 버전3 처럼 해당 버전의 파일들을 하나하나 다 따로 만들어주어야 할까요?

Name	Date Modified
대한민국헌법_제1호_버전1.txt	Today 9:53 PM
대한민국헌법_제2호_버전2.txt	Today 9:53 PM
대한민국헌법_제3호_버전3.txt	Today 9:54 PM
대한민국헌법_제4호_버전4.txt	Today 9:54 PM
대한민국헌법_제5호_버전5.txt	Today 9:54 PM

- 아니요! Git 에서는 놀랍게도 버전별로 만들어줄 필요없이 중간중간 Git 을 사용해 현재 프로젝트의 상태만 저장해주면 됩니다. 여러분들이 파일 저장 버튼을 누르는 것처럼요!

	ohahohah 대한민국 헌법 제10호 - 1987.10.29 전부개정/ 6월 민주화 항쟁	ed51228 3 days ago	14 commits
	LICENSE	Initial commit	4 days ago
	README.md	Initial commit	4 days ago
	대한민국 헌법 제10호 - 1987.10.29 전부개정/ 6월 민주화 항쟁	3 days ago	
	대한민국 헌법 제10호 - 1987.10.29 전부개정/ 6월 민주화 항쟁	3 days ago	

- Git에서는 '누가, 언제, 현재 프로젝트의 상태가 어떤지(현재 파일 내용들)' 세 가지 정보를 포함해 작업내역을 관리합니다. 이렇게 현재 프로젝트 상태를 저장한 것을 **commit(커밋)** 이라고 표현해요. commit 이 무엇인지는 곧 아래에서 배워보겠습니다.

▼ 15) commit(커밋)은 무엇일까?

- Git은 commit(커밋)을 통해 '현재 프로젝트의 상태'를 저장하고 조회합니다.
- 여러분들이 '파일 저장' 버튼을 누르면 현재 상태의 파일이 저장되는 것처럼 현재 프로젝트의 상태를 저장할 수 있어요. 정확히는 snapshot(스냅샷) 즉, 찰칵 사진을 찍는 것 📸 처럼 현재 프로젝트의 전체 상태를 포착하는 거예요.

▼ [코드스니펫] commit 내용 페이지 링크

<https://github.com/ohahohah/constitution-of-republic-of-korea/commit/4fce742d1e851840c600ff316adfa335f986841>

대한민국 헌법 - 1960.6.15 일부개정/ 4.19 혁명

main
누가
언제
ohahohah committed 3 days ago

Showing 2 changed files with 191 additions and 131 deletions.

어떤 내용이 변경되었는지

현재 버전의 파일

File	Additions	Deletions
constitution-amendment.txt	19	131
constitution.txt	303	19

1 대한민국헌법
2 - [시행 1954. 11. 29] [헌법 제3호, 1954. 11. 29, 일부개정]
2 + [시행 1960. 6. 15] [헌법 제4호, 1960. 6. 15, 일부개정]
3
4
5 전문
@@ -29,11 +29,13 @@
29 제9조 모든 국민은 신체의 자유를 가진다. 법률에 의하지 아니하고는 체포, 구금, 수색, 심문, 처벌과 강제노역을 받지 아니한다.
30 체포, 구금, 수색에는 법관의 영장이 있어야 한다. 단, 범죄의 현행 범인의 도피 또는 증거인멸의 염려가 있을 때에는 수사기관은 법률의 정하는 바에 의하여 사후에 영장의 교부를 청구할 권리가 보장된다.
31 누구든지 체포, 구금을 받은 때에는 즉시 변호인의 조력을 받을 권리와 그 당부의 심사를 법원에 청구할 권리가 보장된다.
32 - 제10조 모든 국민은 법률에 의하지 아니하고는 거주와 이전의 자유를 제한받지 아니하여 주거의 침입 또는 수색을 받지 아니한다.
33 - 제11조 모든 국민은 법률에 의하지 아니하고는 통신의 비밀을 침해받지 아니한다.
32 + 제10조 모든 국민은 거주와 이전의 자유를 제한받지 아니하여 주거의 침입 또는 수색을 받지 아니한다.<개정 1960. 6. 15.>
33 + 제11조 모든 국민은 통신의 비밀을 침해받지 아니한다.<개정 1960. 6. 15.>
34 제12조 모든 국민은 산업과 양심의 자유를 가진다.
35 국교는 존재하지 아니하여 종교는 정치로부터 분리된다.

- commit을 하는 순간 현재 프로젝트의 파일 내용, 언제, 누가 저장했는지 정보가 남습니다. 이전 commit의 프로젝트 상태와 현재 프로젝트 상태의 차이를 자동으로 알려줍니다. 빨간색이 삭제된 내용, + 초록색이 추가된 내용이에요.
- commit들은 언제 했는지 정보도 포함하고 있으니까 순서대로 보면 그 자체가 히스토리(history)가 되겠지요? 이것을 commit history 또는 commit log(로그, 기록)라고 합니다!

▼ [코드스니펫] commit history 보기 링크

<https://github.com/ohahohah/constitution-of-republic-of-korea/commits/main>

Commits on Mar 23, 2021		
대한민국 헌법 제10호 - 1987.10.29 전부개정/ 6월 민주화 항쟁	ed51228	🔗
ohahohah committed 3 days ago		
대한민국 헌법 제9호 - 1980.10.27 전부개정/ 10.26 사건, 12.12 군사반란	34a6869	🔗
ohahohah committed 3 days ago		
대한민국 헌법 제8호 - 1972.12.27 전부개정/ 10월 유신	f4b4ec3	🔗
ohahohah committed 3 days ago		
대한민국 헌법 제7호 - 1969.10.21 일부개정	fec8b70	🔗
ohahohah committed 3 days ago		
대한민국 헌법 제6호 - 1962.12.26 전부개정 / 5.16 군사정변	a2f9dc0	🔗
ohahohah committed 3 days ago		
대한민국 헌법 제 5호 - 1960.11.29 일부개정	42c1b08	🔗
ohahohah committed 3 days ago		
대한민국 헌법 - 1960.6.15 일부개정/ 4.19 혁명	4fce74	🔗
ohahohah committed 3 days ago		

▼ 16) 개념 중간 정리 - 버전관리, Commit

- 버전 관리 : 누가, 언제, 현재 프로젝트의 내용이 어떤지 정보를 남긴다는 것
- commit : 현재 프로젝트의 상태를 저장한다. Git 이 이전 commit(이전에 저장한 프로젝트의 상태)를 알고 있으므로 자동으로 어떤 부분이 바뀌었는지 알려준다. 누가, 언제, 어떤 부분을 바꾸었는지 확인해볼 수 있다.

👉 commit 은 **현재 프로젝트의 상태를 저장하는 것**이라는 것을 기억하세요! 파일의 어떤 부분이 변경되었는지를 저장하는 것 이 아니랍니다

▼ 17) [💡 배웠으면 써먹자] - commit 페이지 확인하기

- Q. 아래 대한민국 헌법 commit history 페이지에서 [대한민국 헌법 제10호](#) 가 저장되어있는 commit 페이지로 들어가보세요.

▼ [코드스니펫] 퀴즈 - 대한민국 헌법 commit history 페이지

<https://github.com/ohahohah/constitution-of-republic-of-korea/commits/main>

- A. 정답

▼ [코드스니펫] 정답- 대한민국 헌법 commit history 페이지

<https://github.com/ohahohah/constitution-of-republic-of-korea/commit/ed512280b49c566de2749f3abd28b9d11b22c158>

03. 버전관리와 commit - 실습 01



앞으로 우리는 김치 요리법 🥕 을 모으는 프로젝트를 Git 을 사용해 관리할 꺼예요.

여러분들이 내 코드를 관리하는 프로젝트를 만들고 싶다면 코드파일을 넣어주면 되겠죠? 원리는 똑같습니다.

▼ 18) 프로젝트 만들기



프로그래밍할 때 파일/폴더명은

- 영어로, 특수문자 없이(띄어쓰기도 안돼요!) 만들어주세요. 단어를 연결할 때는 _ 나 . 둘 중 하나를 사용하도록 합시다! 컴퓨터가 다른 문자가 섞이면 잘 못 알아들을 수 있어요.

2. 내용(데이터)를 제대로 나타내주는 이름을 지어주세요!

파일/폴더에 담긴 내용은 곧 데이터입니다. 데이터를 제대로 관리하는 첫 걸음은 제대로 이름 붙여주는 것이에요. (예.

11111.txt 🧑 / project_list.txt, commit_practice.py 🧑)

예를 들면, 프로젝트 이름을 git-tutorial, taco-recipe, constitution-of-republic-of-korea 로 짓는 것처럼요!

- 바탕화면에 `kimchi-recipe` 라는 폴더를 만들어주세요. 앞으로 사용할 우리의 실습 프로젝트 폴더입니다.

- `kimchi-recipe` 폴더 안에 `jeon.txt` 라는 파일을 만듭시다. 메모장을 켜서 아래 내용을 적은 후 파일을 저장해주세요.

▼ [코드스니펫] windows에서 txt 파일 만드는 방법 링크

<https://blog.naver.com/PostView.nhn?blogId=hanasilver&logNo=220703151064&redirect=Dlog&widgetTypeCall=true&topReferer=https%3A%2F%2Fwww>

▼ [코드스니펫] mac에서 txt 파일 만드는 방법 링크

<https://ngnl.tistory.com/15>

▼ [코드스니펫] 김치전 만드는 방법

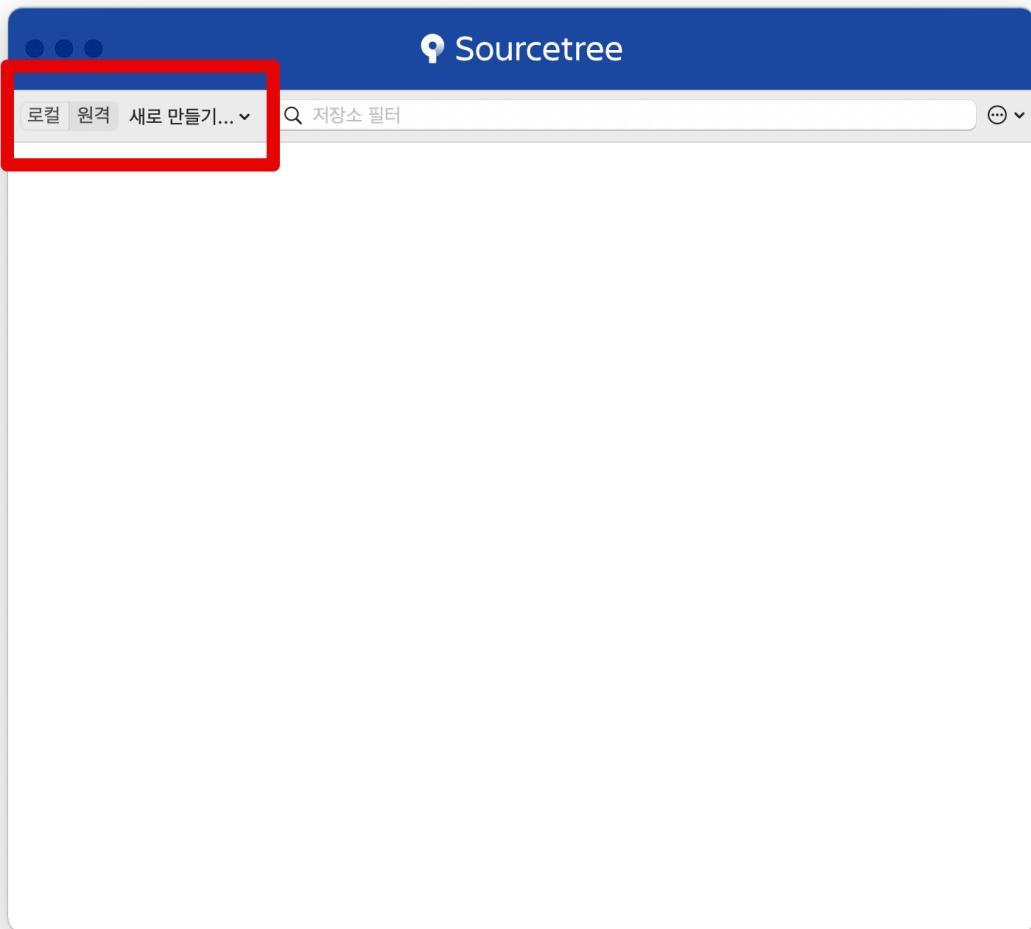
```
김치전 만드는 방법
1. 김치를 송송 썬다.
2. 부침개가루를 물에 섞는다.
3. 1번의 김치와 2번의 부침개 반죽을 섞는다.
```

▼ 19) 만든 프로젝트를 Git이 관리하는 폴더로 만들기

- Git을 사용하기 위해서 Sourcetree를 사용하겠습니다.

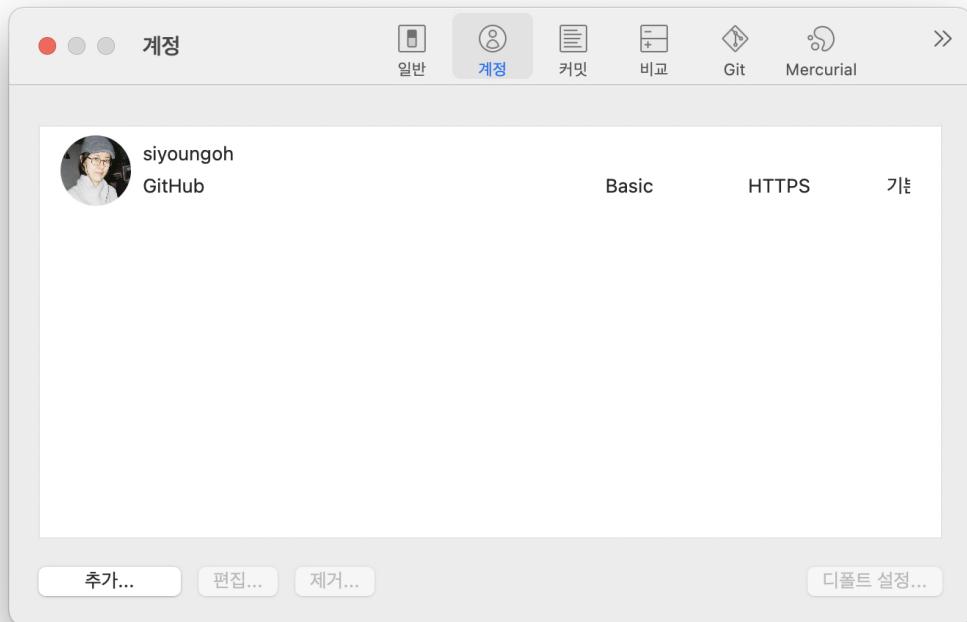
▼ sourcetree 설치 제대로 되어있는지 확인하기

- 한글로 잘 보이는지 확인해주세요. Git 용어에 익숙하다면 영어로 사용해도 상관없습니다! 아래처럼 메뉴들이 한글로 보이면 됩니다.



- Github에 잘 로그인 되었는지 확인

- 설정 - 계정

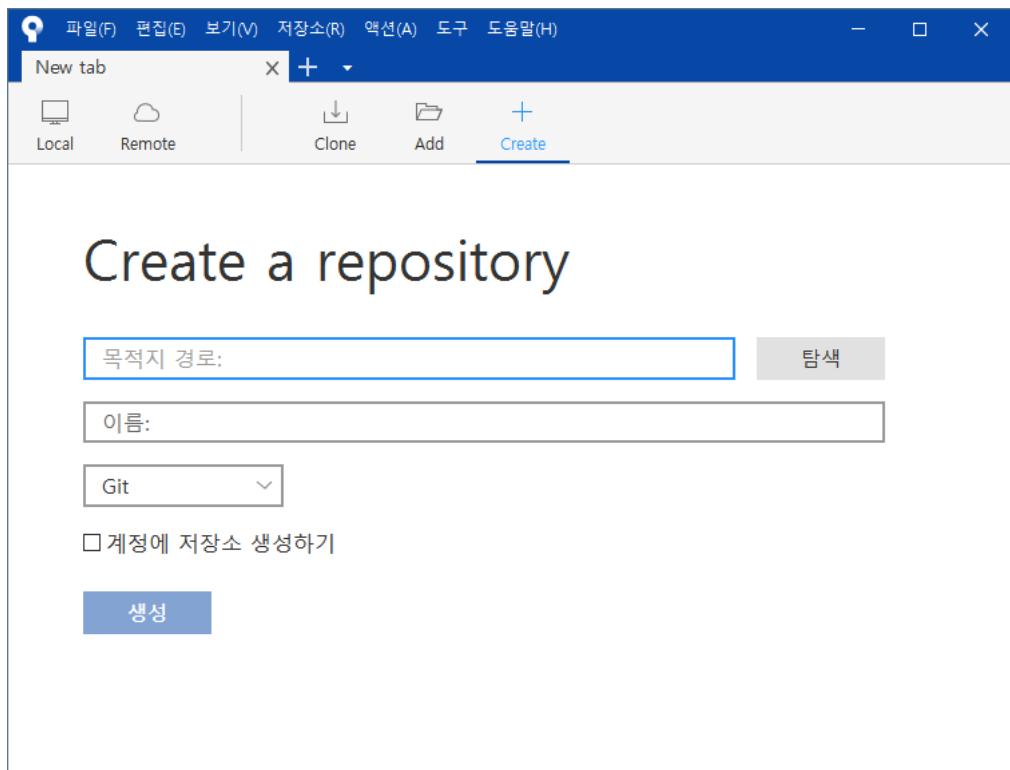


▼ [코드스니펫] 사진 설치 - sourcetree, Github 설정하기

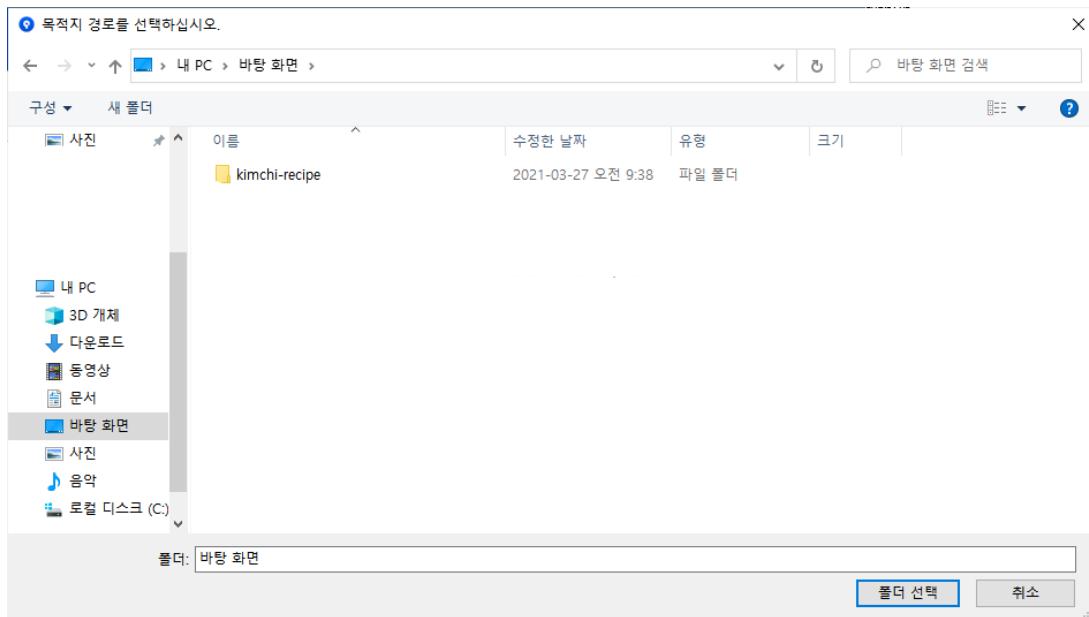
<https://www.notion.so/teamsparta/Git-0-a9bfcc232d244e52bedebbc5ce583a1c#e47f295b04384647995862a9302a8c2e>

▼ windows에서 Git 관리 폴더 설정하기

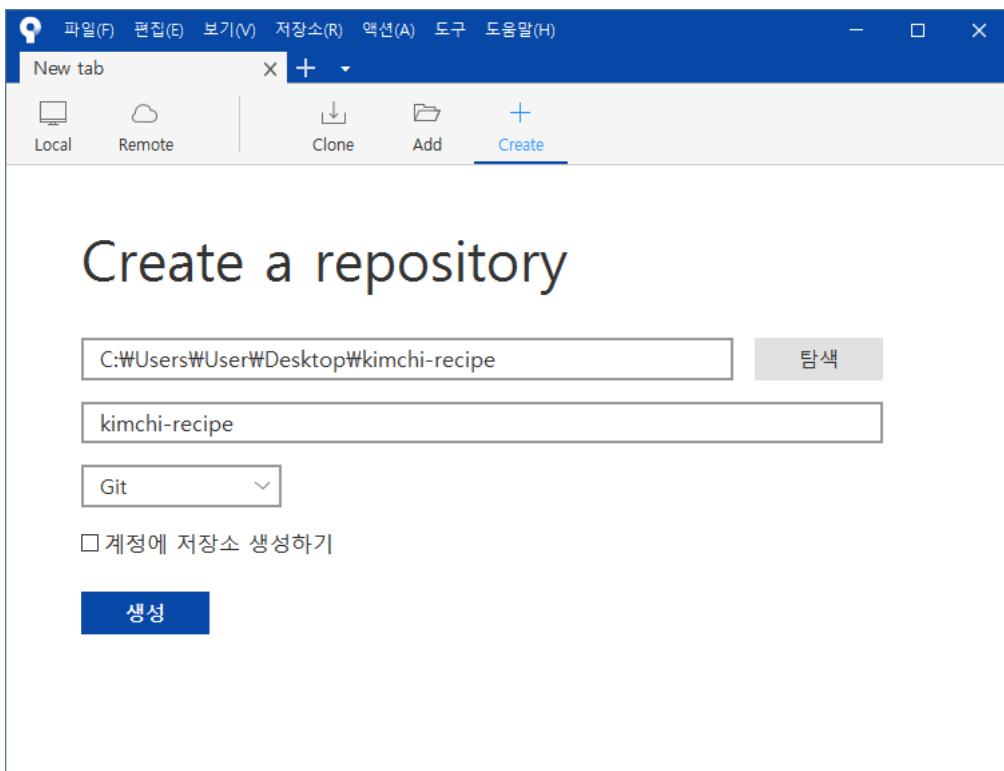
1. sourcetree 를 켜고 Create



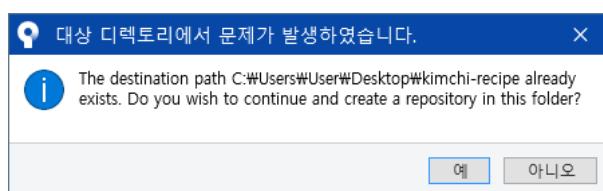
2. 탐색을 누르고 뜨는 화면에서 바탕화면에 만든 `kimchi-recipe` 클릭하고 '폴더 선택' 버튼을 눌러주세요.



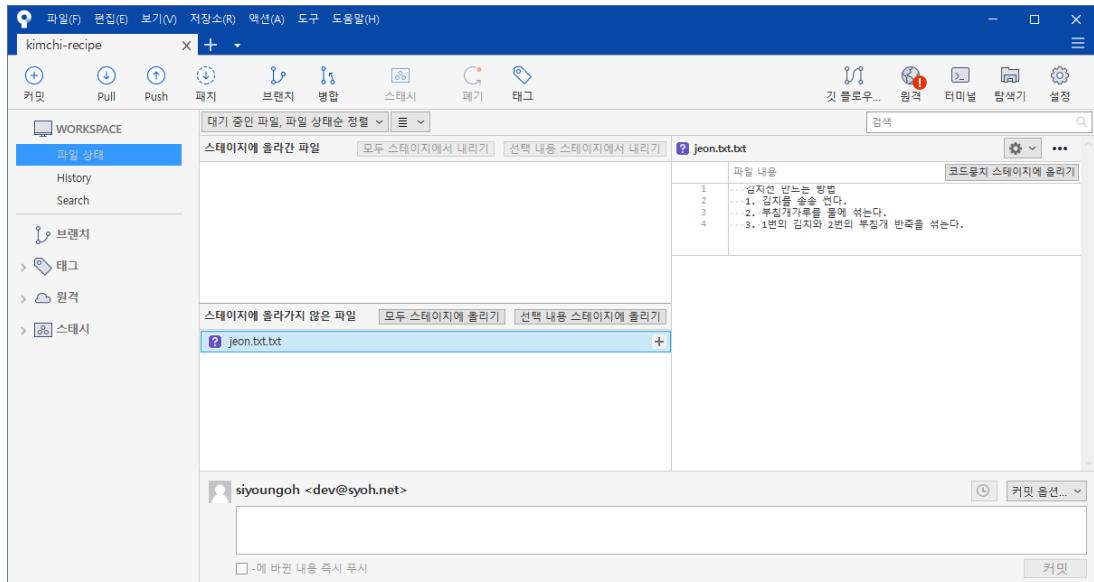
- 아래처럼 내용이 잘 채워졌다면 '생성을' 눌러주세요.



- 아래처럼 팝업이 뜨면 예 를 눌러주세요. 놀라지 마세요! 이미 있는 폴더를 git 폴더로 만들 건지 다시 한번 확인하는 것 뿐이에요.



3. 아래처럼 보이면 잘 된 거에요!



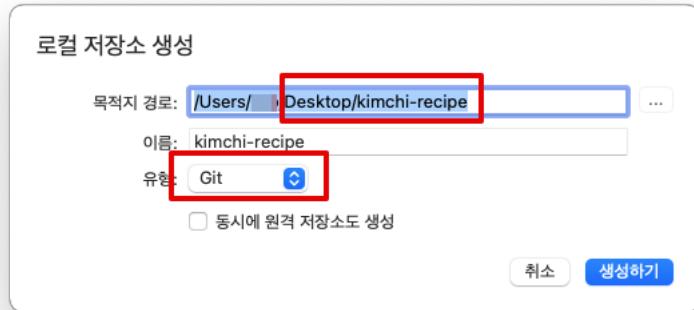
👏 오늘도 설정 아주 잘하셨어요! 이렇게 프로젝트를 git 프로젝트로 설정하는 것을 git 초기화(initialize, init) 이라고 불러요!

▼ mac에서 Git 관리 풀더 설정하기

1. sourcetree 를 켜고 새로 만들기 - 로컬 저장소 추가하기 클릭



2. 목적지 경로 옆에 ... 을 클릭해서 바탕화면에 있는 `kimchi-recipe` 폴더를 선택해주세요. 이름과 유형을 아래처럼 적어주고 '생성하기' 버튼 클릭

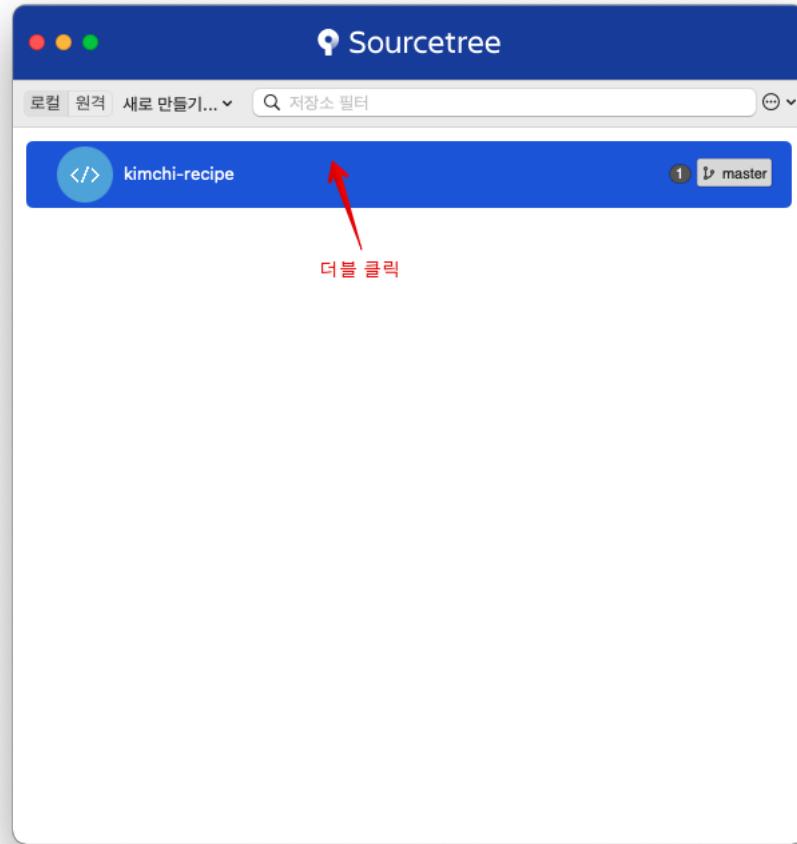


- 아래처럼 팝업이 뜨면 OK 를 누르면 됩니다. (컴퓨터 설정이 한글로 되어있다면 한글 메시지가 뜨겠지요? 😊 오른쪽 버튼을 누르시면 됩니다)

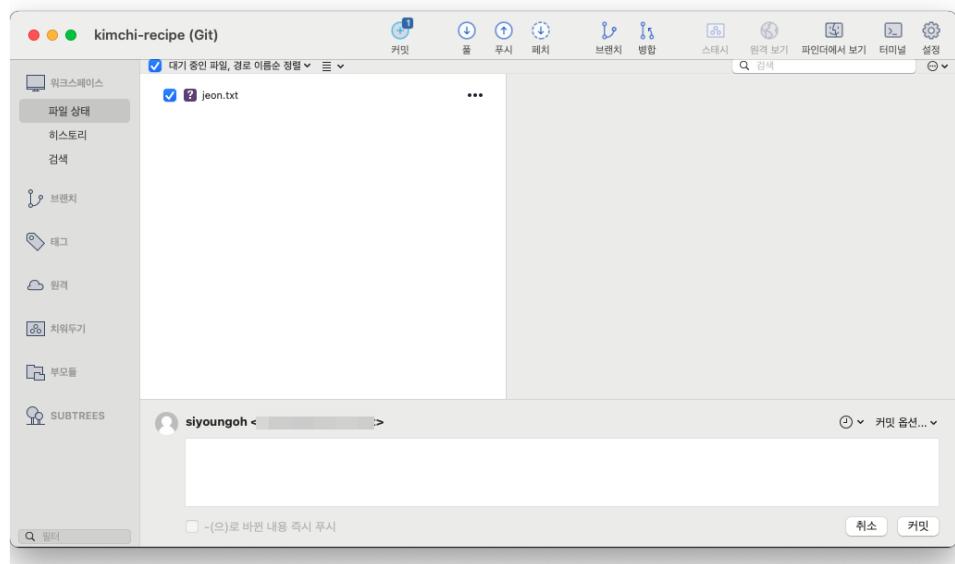


3. 로컬 탭을 누르고 - 하단 리포지토리 이름을 더블 클릭해서 프로젝트 관리하는 창으로 들어가세요.





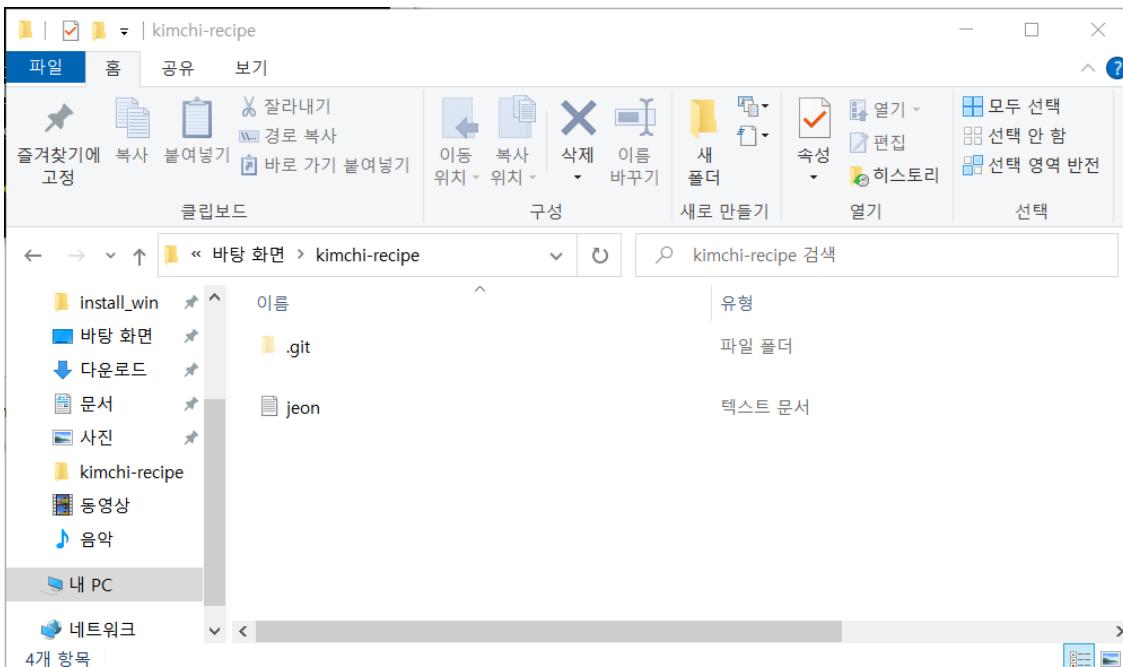
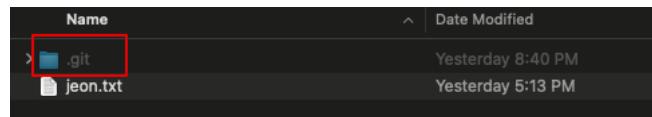
- 아래처럼 보이면 잘 된 거에요!





오늘도 설정 아주 잘하셨어요! 이렇게 프로젝트를 git 프로젝트로 설정하는 것을 git 초기화(initialize, init)이라고 불러요!

- 이제 Git을 사용할 수 있는 준비가 끝났습니다! 파일 탐색기(mac은 Finder)에서 아까 만든 프로젝트 폴더(바탕화면-kimchi-recipe)에 들어가면 git 설정파일인 `.git` 폴더가 생겨있을 거예요!



- 만약 보이지 않는다면 숨김파일이라 컴퓨터 설정에서 보이지 않게 해둔 거랍니다! 아래 설정을 따라해주세요!

▼ [코드스니펫] windows8 이상에서 숨김파일 보기

<https://support.microsoft.com/ko-kr/windows/windows-10에서-숨김-파일-및-폴더-보기-97fbca472-c603-9d90-91d0-1166d1d9f4b5>

▼ [코드in스니펫] windows7에서 숨김파일 보기

<https://helpx.adobe.com/x-productkb/global/show-hidden-files-folders-extensions.html>

▼ mac에서 숨김파일 보기

- command + shift + . 을 누를 때마다 숨김 파일이 보였다가 보이지 않았다가 합니다.



컴퓨터에 있는 프로젝트를 Git이 관리하는 프로젝트로 만들 수 있습니다. 앞으로 Git으로 관리할꺼야! 하고 설정해주면 됩니다. 이 작업을 git 초기화(git initialize)한다고 표현합니다.

▼ 20) Git 프로젝트 없애는 방법 - 예러나면 참고!

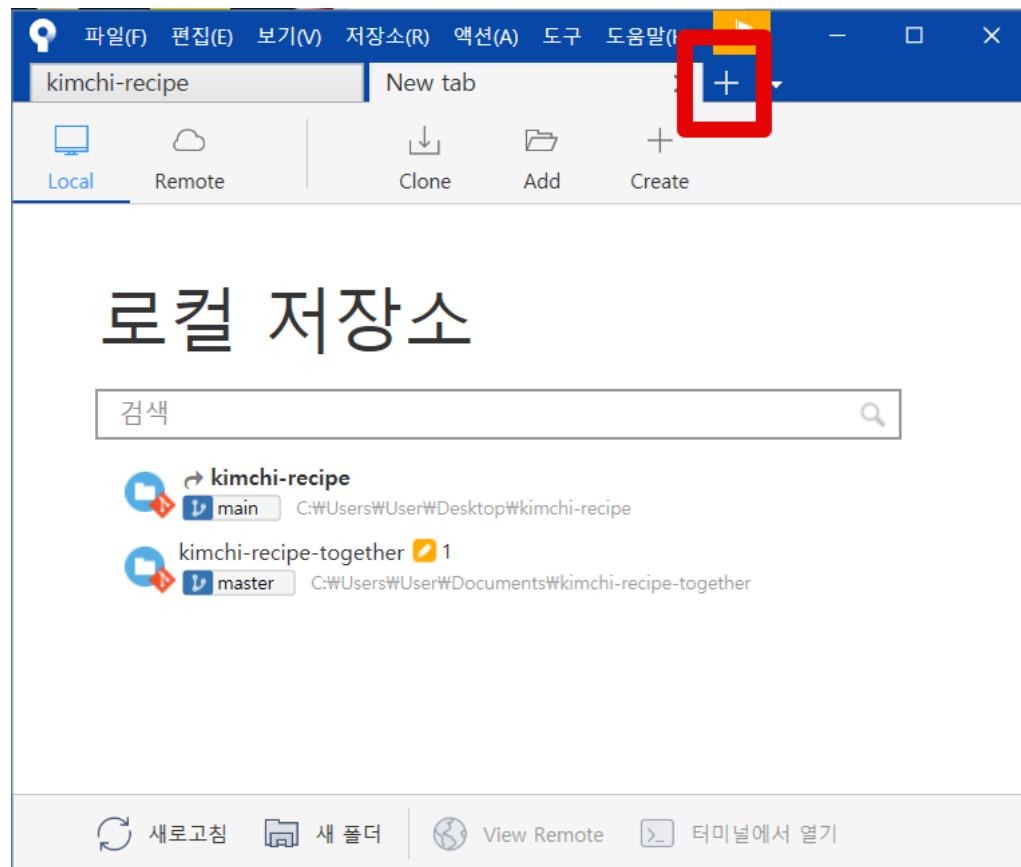


맘껏 실험해보기 위해 설정을 처음으로 깔끔하게 되돌리는 방법을 알려드릴게요!
뭔가 잘 동작 안 된다면 없애고 새 마음으로 시작하는 것도 방법이에요!

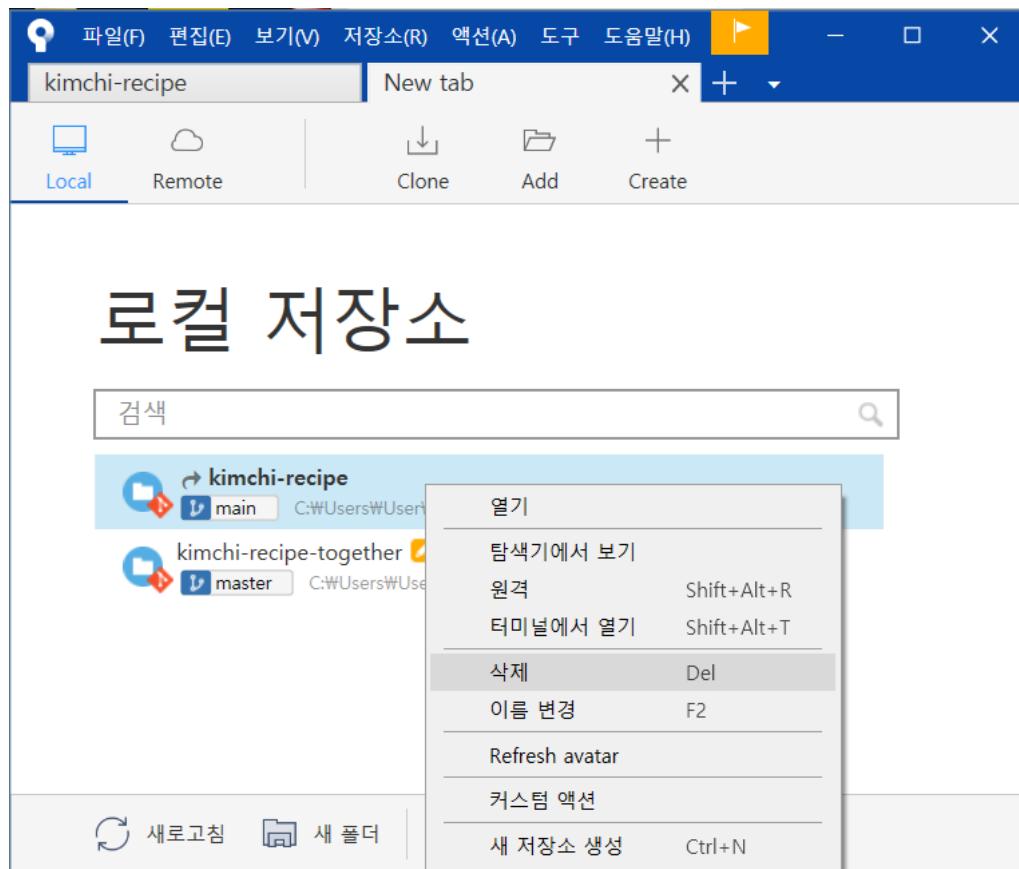
단, 언제나 무조건 다 지우고 처음부터 만들 수는 없겠죠?
다음 수업에서는 에러를 해결하기 위한 첫 단계! 기록하고 해결하는 팁도 알려드릴게요!

▼ Windows에서 삭제하기

1. 를 눌러서 New Tab 을 열어주세요.



2. 삭제하려는 저장소를 선택한 후 우클릭 - 삭제를 누르면 팝업이 뜹니다.

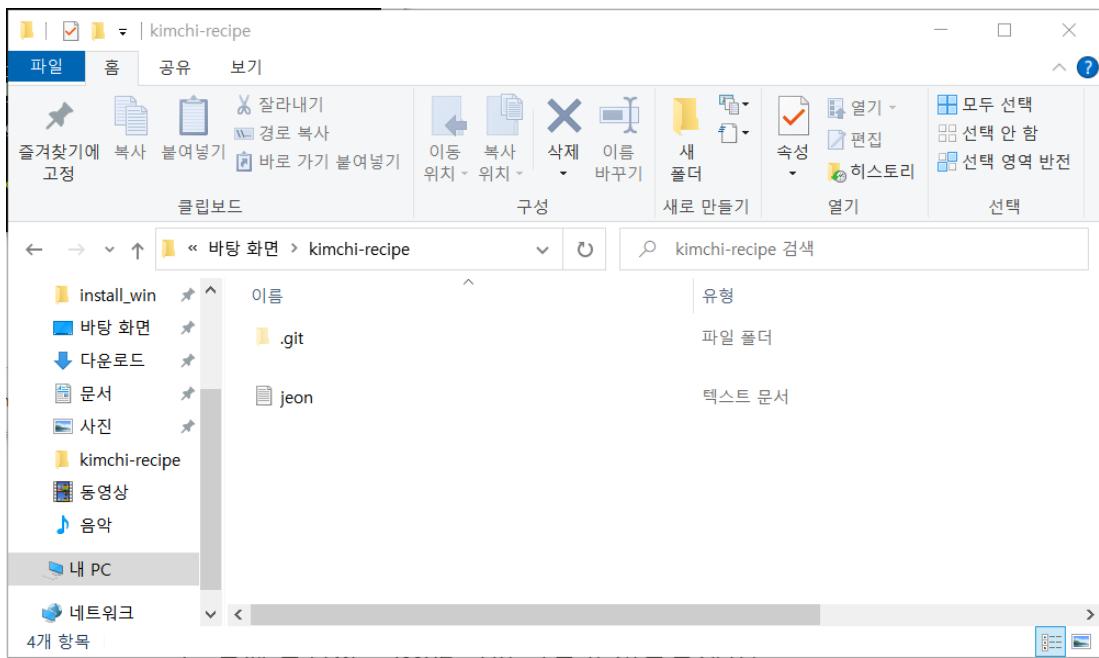


3. sourcetree 에서만 안 보이게 하려면 [북마크를 제거하세요](#) 를
컴퓨터에서까지 삭제하려면 [디스크에 있는 저장소를 삭제하세요](#) 를 눌러주세요.

디스크에 있는 저장소를 삭제하세요 를 누르면 내 컴퓨터에서도 삭제가 됩니다! git 설정이 꼬여서 git 설정만 처음부터 해보고 싶은 거라면 다음 단계를 참고!

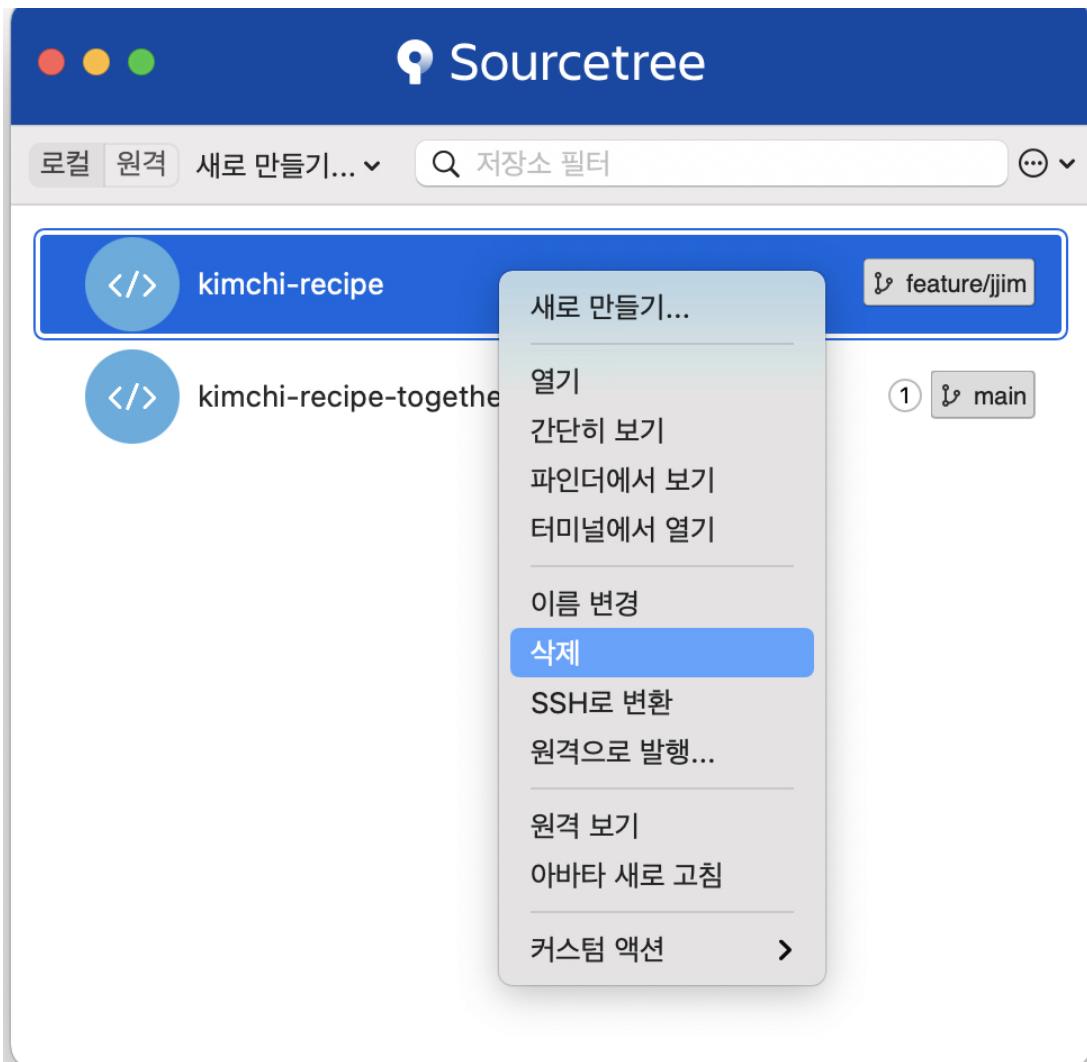


4. git 설정이 잘 되지 않아서 프로젝트 파일은 현재 상태 그대로 두고, git 설정만 처음부터 되돌리고 싶은 거라면!
- 내 컴퓨터 프로젝트 폴더 내 `.git` 폴더를 지워주면 됩니다. 이 경우, git 프로젝트가 아닌 그냥 일반 프로젝트가 되었다고 생각하면 됩니다.
다시 git 프로젝트로 만드는 git 초기화(initialize)부터 해주면 되겠죠?



▼ mac에서 삭제하기

1. 삭제하고 싶은 프로젝트를 선택하고 우클릭 - 삭제 클릭

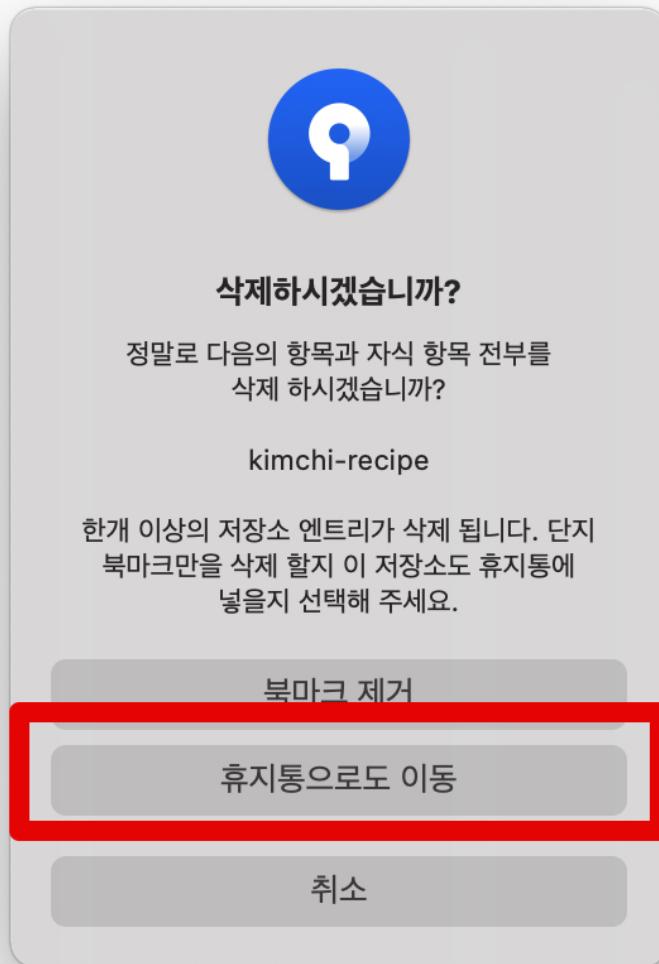


2. 휴지통으로 이동 : 컴퓨터에서도 파일을 삭제하기를 선택하면, 내 컴퓨터에서도 파일이 삭제됩니다. 만약, git 설정이 꼬여서 삭제하고 싶은 거라면 다음 단계를 참고해주세요!

북마크 제거 하면 sourcetree 에서만 보이지 않아요. 프로젝트 파일은 컴퓨터에 남아있어요. 즉, Git 의 정보도 그대로 남아있어요.



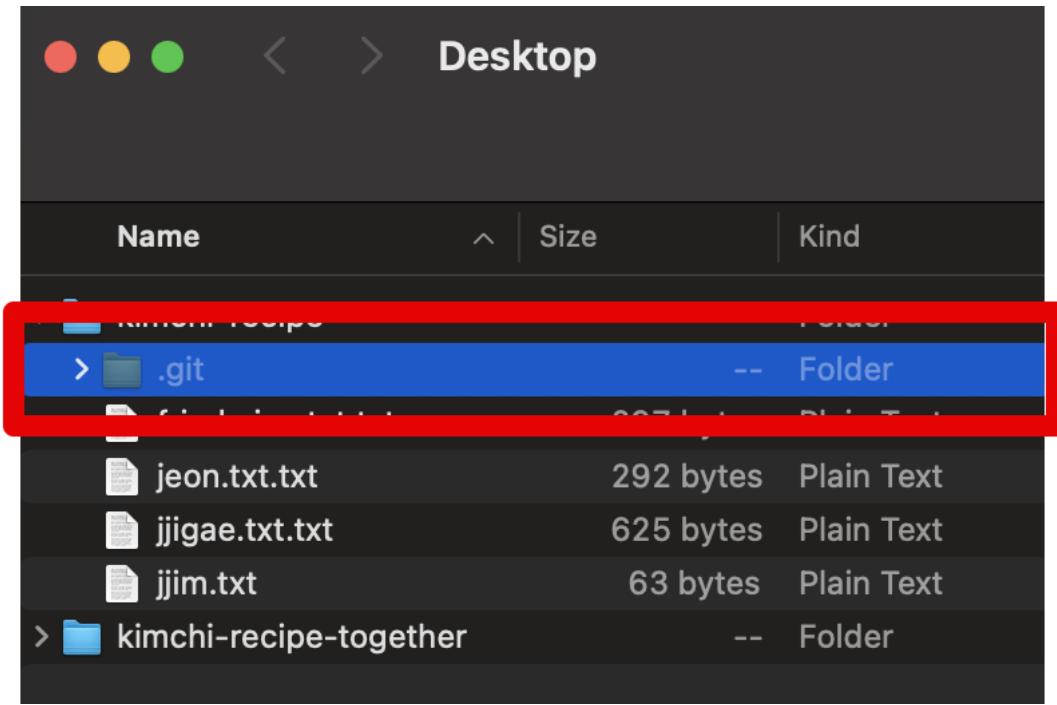
휴지통으로 이동을 누르면 내 컴퓨터에서도 삭제가 됩니다! git 설정이 꼬여서 git 설정만 처음부터 해보고 싶은 거라면 다음 단계를 참고!



3. 한 걸음 더!

git 설정이 잘 되지 않아서 프로젝트 파일은 현재 상태 그대로 두고, git 설정만 처음부터 되돌리고 싶은 거라면 내 컴퓨터 프로젝트 폴더 내 `.git` 폴더를 지워주면 됩니다.

- 이 경우, git 프로젝트가 아닌 그냥 일반 프로젝트가 되었다고 생각하면 됩니다.
다시 git 프로젝트로 만드는 git 초기화(initialize)부터 해주면 되겠죠?

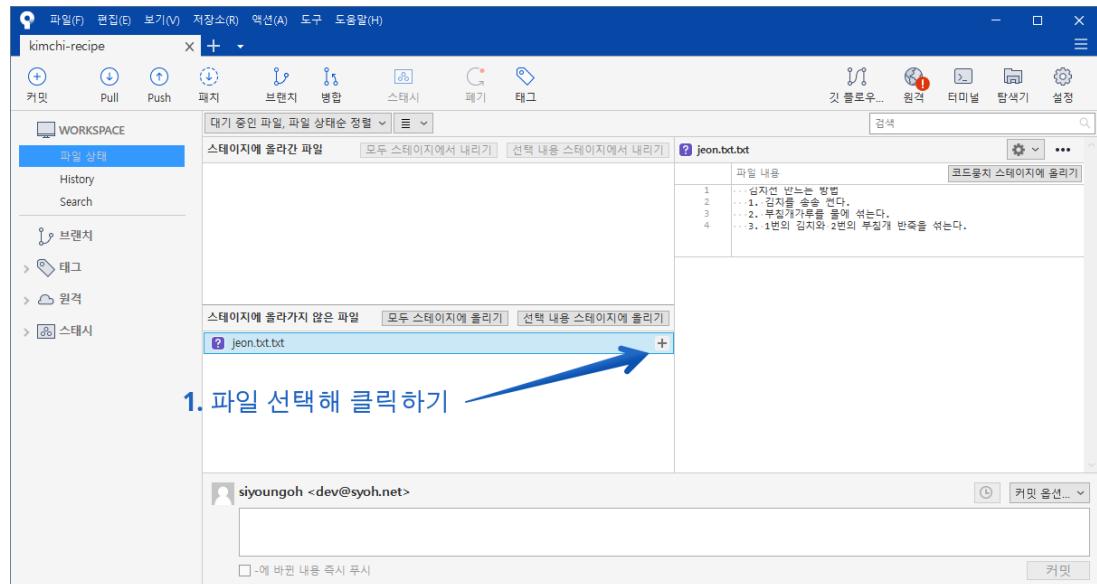


▼ 21) 첫 commit 하기

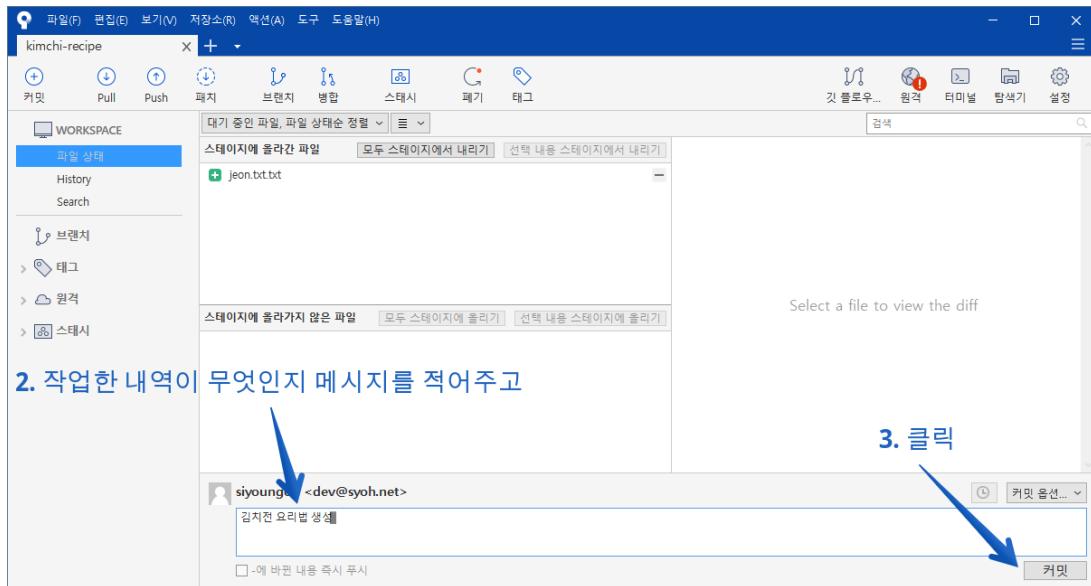
commit은 현재 프로젝트 상태를 📸 찰칵 사진찍어서 저장하는 것!
프로젝트 상태를 포함해서 누가, 언제, 작업내역 메시지(commit 메시지라고 해요) 정보를 포함하고 있어요.

▼ windows에서 commit 하기

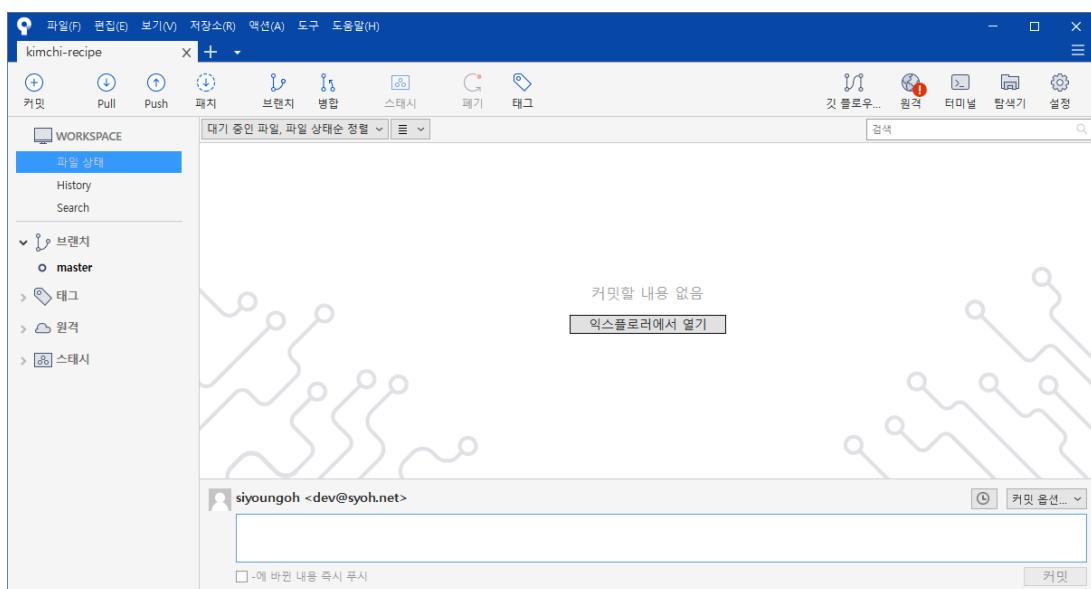
1. sourcetree를 가보면 아래처럼 현재 프로젝트의 상태가 보일 꺼예요. 변경된 내역이 보이네요! 파일 옆 + 버튼을 눌러주세요.



2. 를 클릭했더니 위로 올라갔네요! 아래 순서대로 적어주고 커밋 버튼을 클릭!



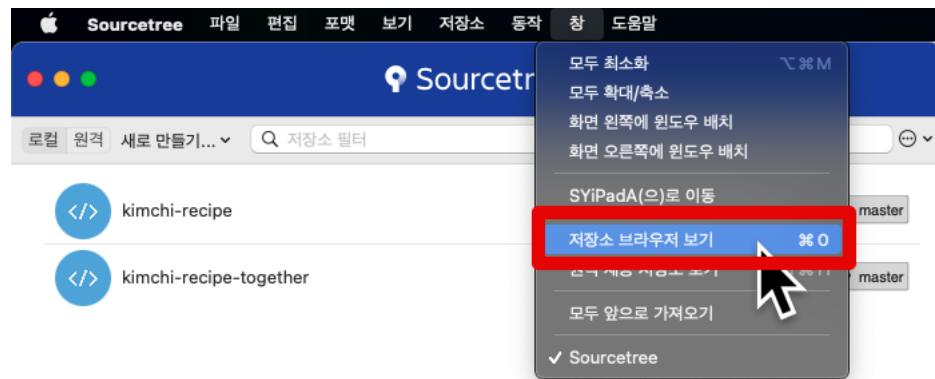
- 다시 이 상태로 돌아오면 현재 프로젝트의 상태가 저장된 것입니다! 첫 commit 을 축하해요! 여기까지 잘 따라온 나를 위해 박수 쳐 주세요! 🎉🎉🎉



▼ mac에서 commit 하기

▼ 참고. 만약 아래와 같은 창이 바로 보이지 않는다면

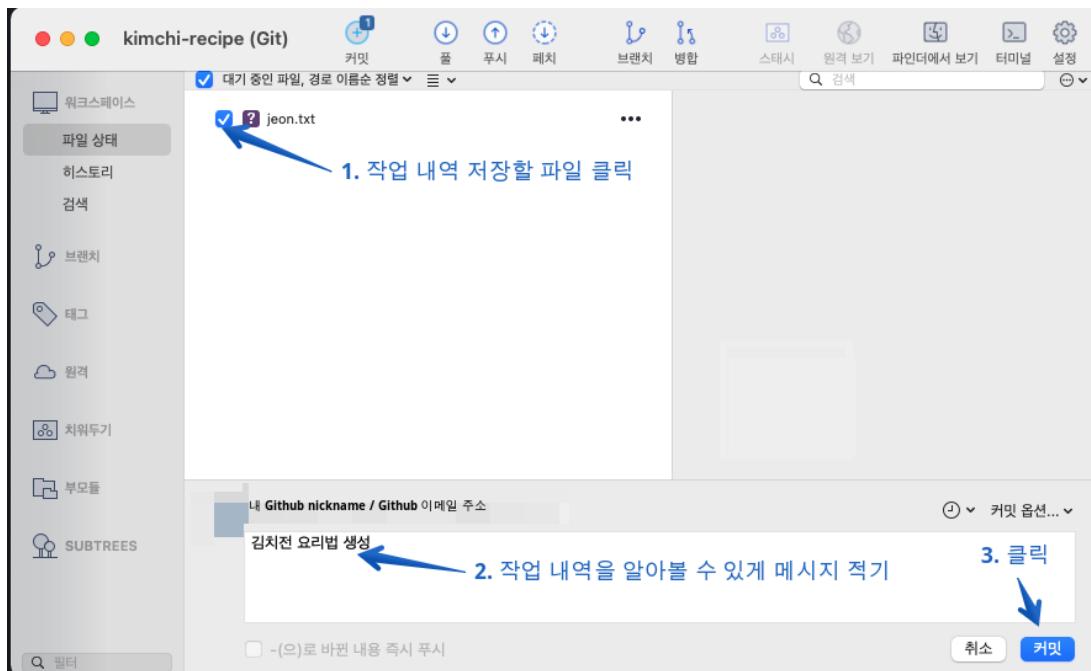
1. 상단 메뉴바 - 창 - 저장소 브라우저 보기 를 클릭하면 아래 창이 됩니다.



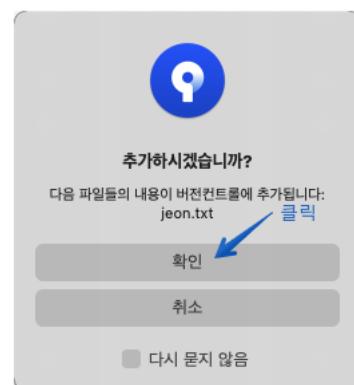
2. 여기서 로컬 탭 누르고, kimchi-recipe(내가 만든 repo 이름) 더블 클릭!



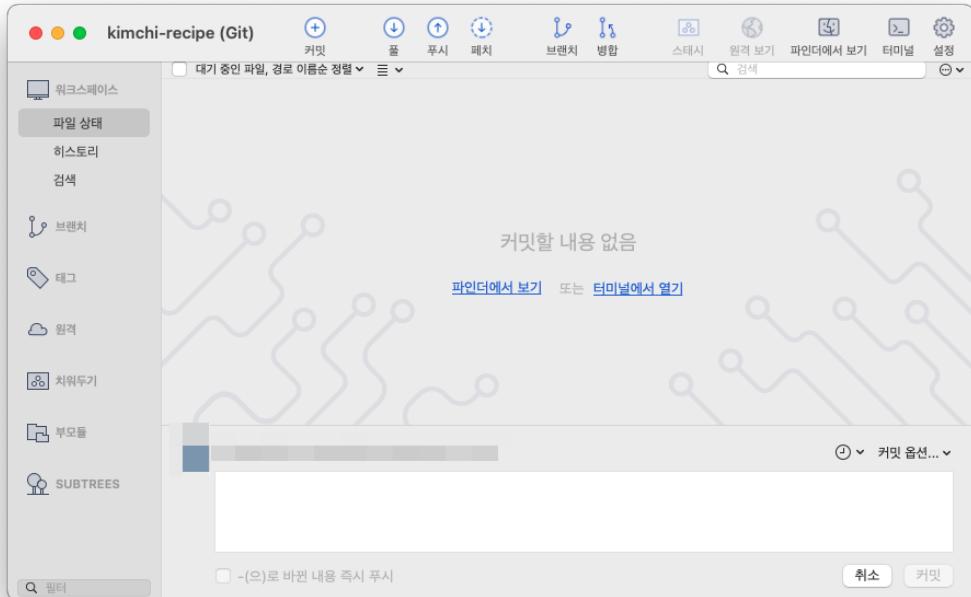
1. sourcetree 를 가보면 아래처럼 현재 프로젝트의 상태가 보일 꺼에요. 각 항목을 적고 커밋 을 눌러줍시다.



- 이런 팝업이 뜬다면 확인을 눌러주세요. 전에 commit 하지 않았던 새로운 파일을 추가한다는 뜻이에요.



- 다시 이 상태로 돌아오면 현재 프로젝트의 상태가 저장된 것입니다! 첫 commit 을 축하해요! 여기까지 잘 따라온 나를 위해 박수 쳐 주세요! 🥁🥁🥁



👉 이제 작업 내역이 내 컴퓨터에 저장되었어요!
누구나 인터넷에서 내 코드를 볼 수 있게 Github을 사용하는 건 조금 이따가 같이 배워볼게요~!

▼ 22) commit 한 번 더 하기

- 사용하면 할 수록 익숙해집니다! 한 번 더 commit 해보죠!

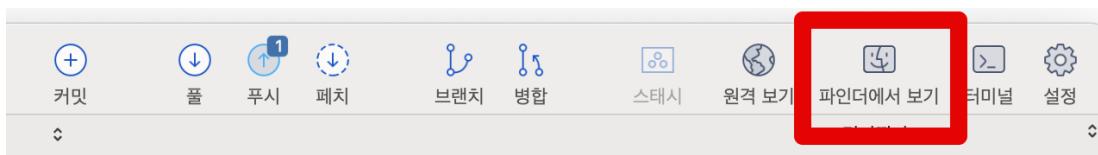
1. 파일 설정

- commit (현재 프로젝트의 상태 저장)을 하기 위해서 파일을 수정해줍시다. 프로젝트가 변경된 내용이 있어야 commit을 할 수 있으니까요!
- 생각해보니 김치전을 굽는 방법을 적지 않았네요. jeon.txt 파일을 아래처럼 ④ 항목을 추가해볼게요.

▼ [코드스니펫] 커밋 한 번 더 하기

4. 부침개 반죽을 후라이팬에 올려 굽는다.

- 👉 꿀팁! sourcetree 우측 상단을 보면 [파인더에서 보기](#) (mac), [탐색기](#) (windows)를 누르면 프로젝트 폴더가 열립니다!

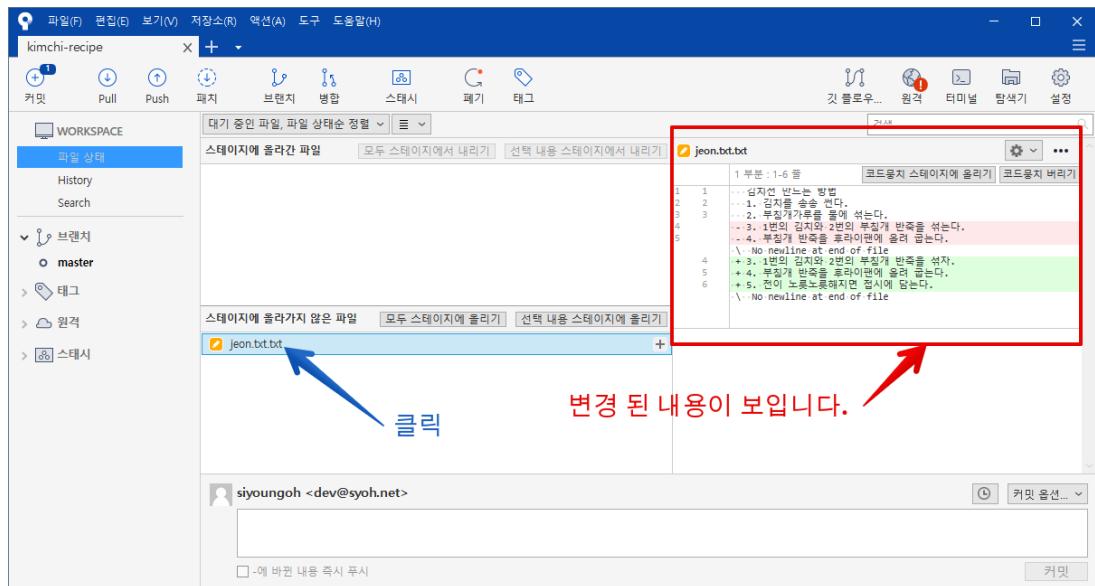


2. 프로젝트 상태 확인하기

- 다시 sourcetree 를 가보죠. 세상에나! 어떤 부분이 변경되었는지 자동으로 보이네요!
- 아하! 이전 commit에 저장되어있는 프로젝트 상태와 현재 프로젝트 상태를 비교해서 보여준다는 게 이런 의미였군요! - **빨간색**으로 되어있는 부분이 삭제된 부분이고, **+ 초록색**으로 되어있는 부분이 추가된 부분이네요!

👉 그럼 commit을 잘 사용하면 프로젝트에서 이전 내용과 어떤 부분이 바뀌었는지도 알 수 있겠네요!

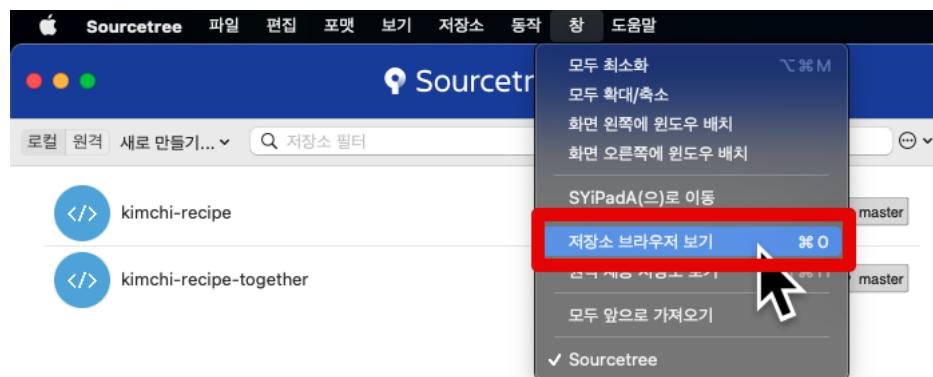
◦ windows



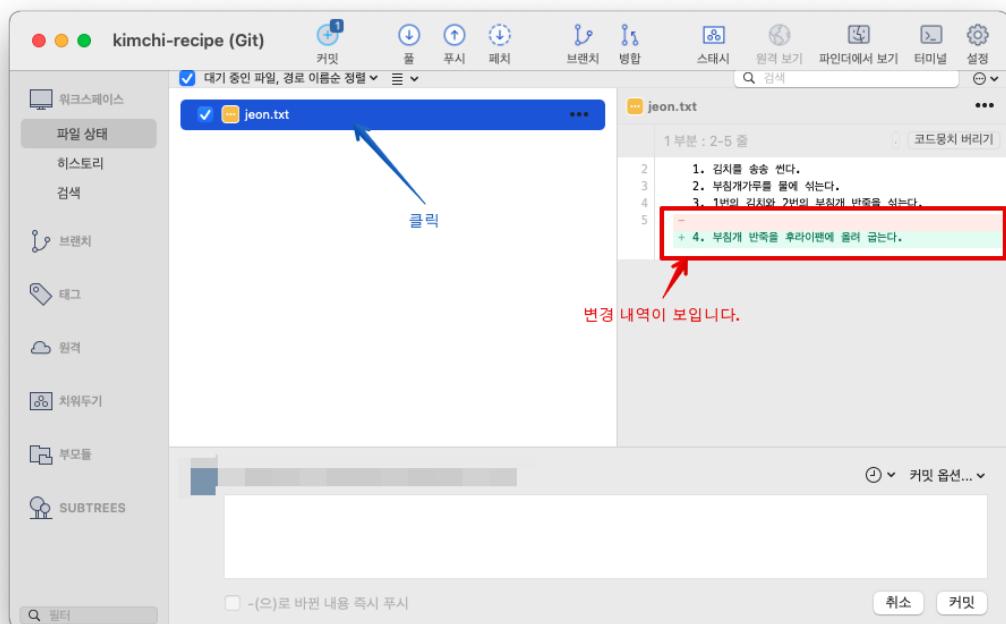
◦ mac

▼ 참고. 만약 아래와 같은 창이 바로 보이지 않는다면

- 상단 메뉴바 - 창 - 저장소 브라우저 보기 를 클릭하면 아래 창이 뜹니다.



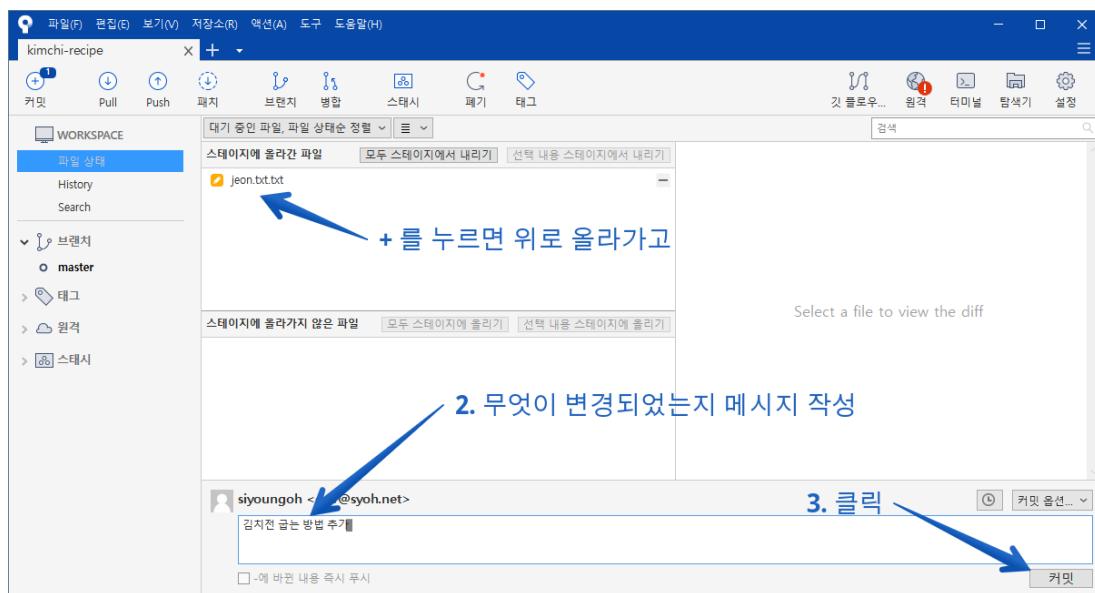
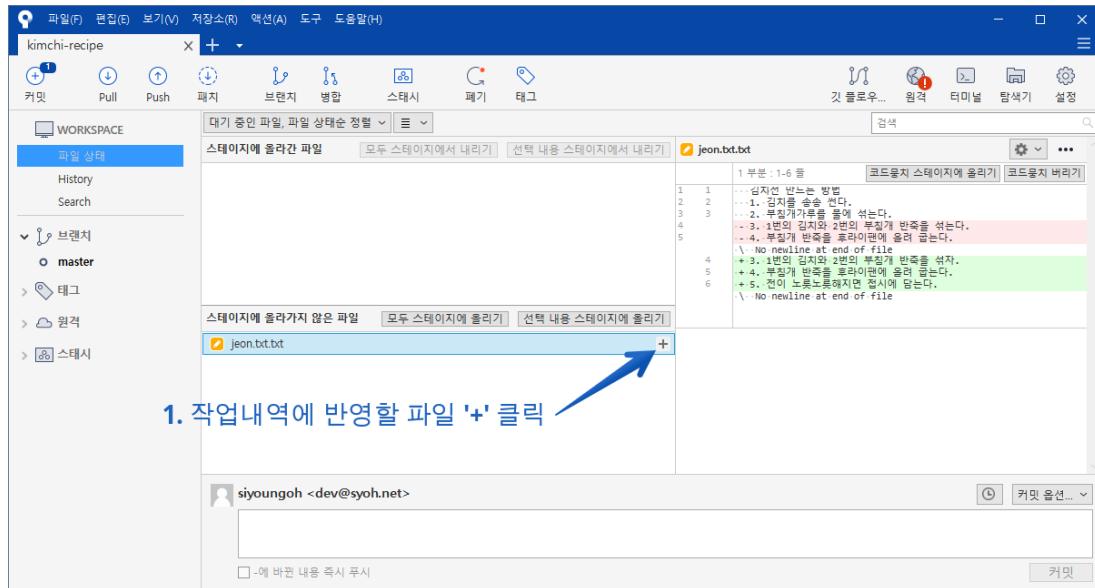
- 여기서 로컬 탭 누르고, kimchi-recipe(내가 만든 repo 이름) 더블 클릭!



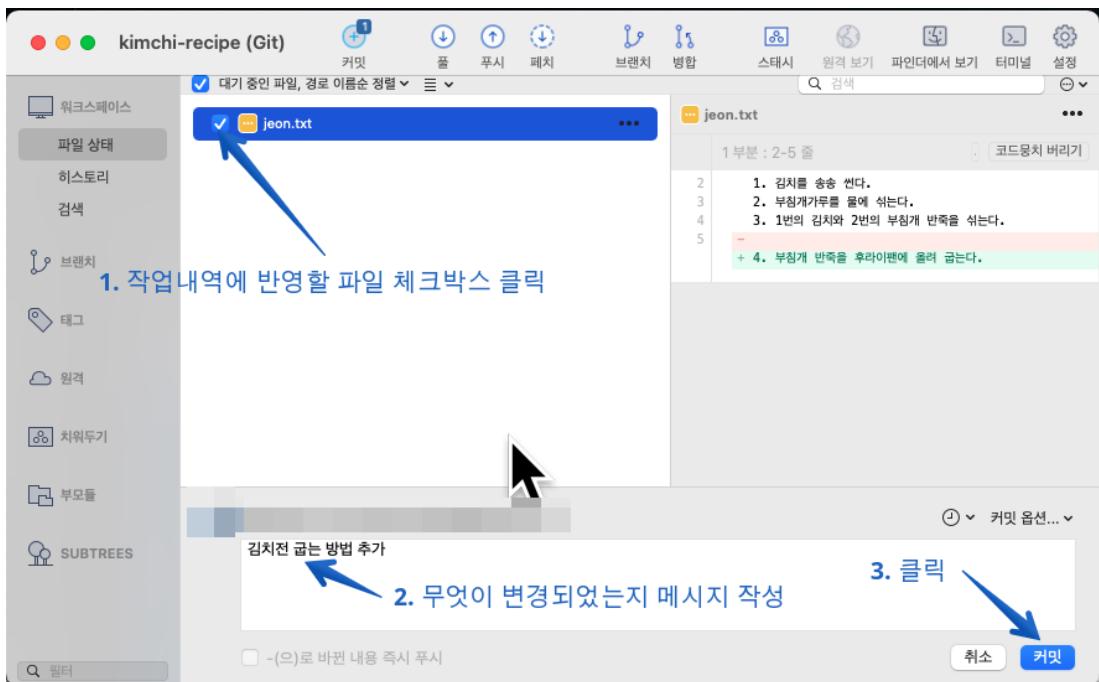
3. commit 하기

- 그럼 다시 현재 프로젝트 상태를 찰칵 📸 commit 해보죠!

- windows



- mac



04. 버전관리와 commit - 실습 02

▼ 23) commit 한 번 더! - 여러 파일 수정

- 이번엔 여러 파일을 한번에 commit 해보겠습니다. 파일을 하나 수정하는 것과 여러 개 수정하는 것 모두 commit 하는 방법은 같습니다!

1. 파일 수정하기

- 김치전 요리법을 업데이트 할게요. `jeon.txt` 파일을 아래처럼 3~5 항목을 수정해주세요.

▼ [코드스니펫] jeon.txt

```

3. 1번의 김치와 2번의 부침개 반죽을 섞자.
4. 부침개 반죽을 후라이팬에 올려 굽는다.
5. 전이 노릇노릇해지면 접시에 담는다.

```

- 그리고 김치볶음밥과 김치찌개 요리법을 저장할 텍스트 파일을 만들고 각각 파일에 아래처럼 적어주세요.

▼ [코드스니펫] fried-rice.txt

```
김치 볶음밥 만드는 방법
```

▼ [코드스니펫] jjigae.txt

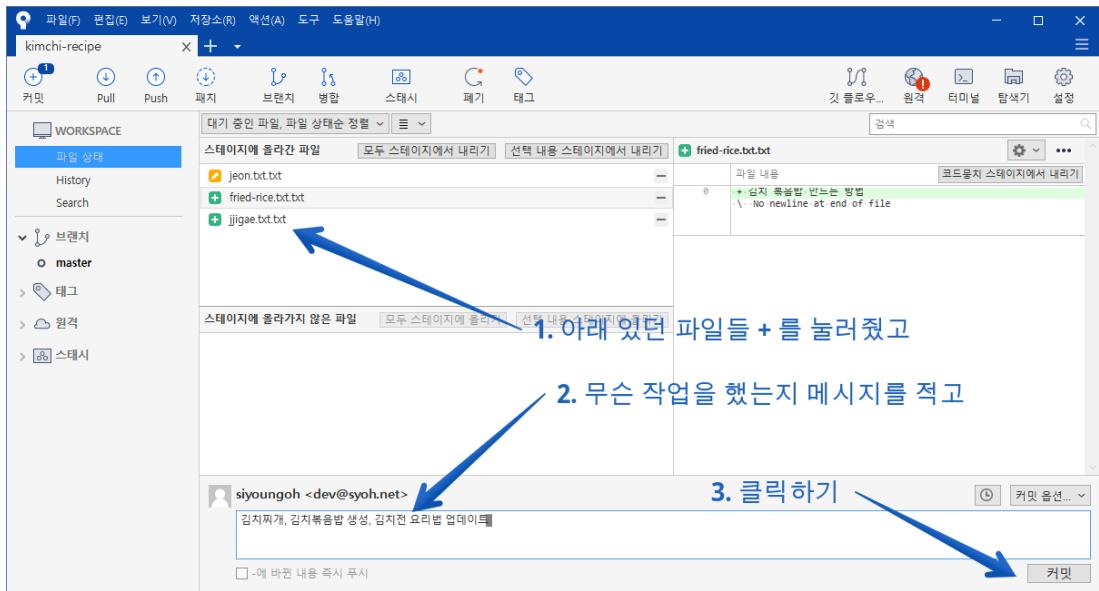
```
김치찌개 만드는 방법
```

2. 프로젝트 상태 확인하기

- 다시 sourcetree 를 켜보죠.
- 세상에나! 파일을 새롭게 만들어도, 여러 파일을 수정해도 어떤 부분이 변경되었는지 보이죠!

3. commit 하기 - 프로젝트 상태 저장

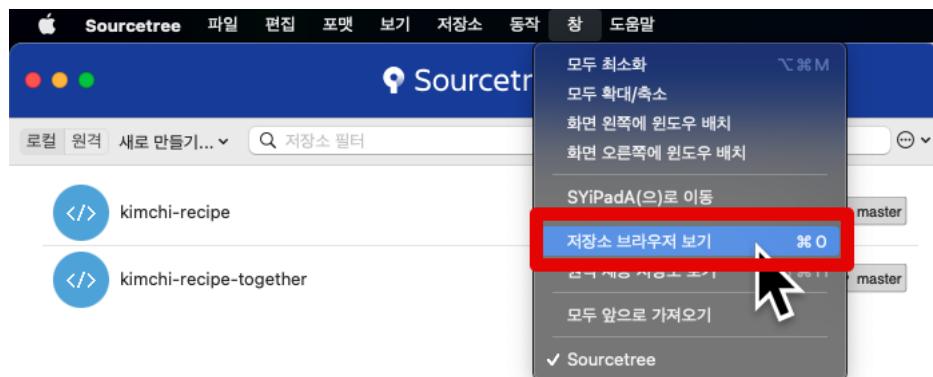
- 이제 또 commit 을 해보겠습니다!
- windows



- mac

▼ 참고. 만약 아래와 같은 창이 바로 보이지 않는다면

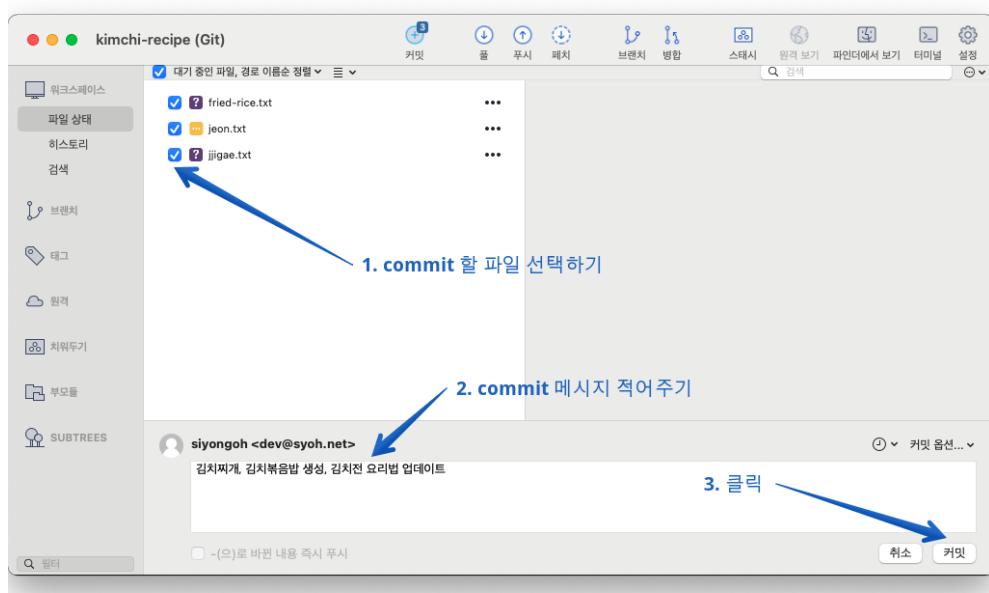
1. 상단 메뉴바 - 창 - 저장소 브라우저 보기 를 클릭하면 아래 창이 됩니다.



2. 여기서 로컬 탭 누르고, kimchi-recipe(내가 만든 repo 이름) 더블 클릭!



- 아까처럼 추가하시겠습니까? 아까처럼 파일을 추가하겠냐고 묻는 팝업창이 뜨면 '확인'을 눌러주세요!





하단에 'commit 할 때 적는 메시지'를 **commit 메시지**라고 부릅니다.
commit 메시지는 꼭 해당 commit의 내용이 무엇인지 알 수 있도록 잘 적어주어야해요!
 그래야 나중에 작업내역을 파악할 수 있습니다.

▼ 24) commit 한 번 더! - 파일 중 하나만 수정

1. 파일 수정하기

- 이번에는 **fried-rice.txt** 파일만 아래처럼 업데이트 해보죠!

▼ [코드스니펫] fried-rice.txt 업데이트

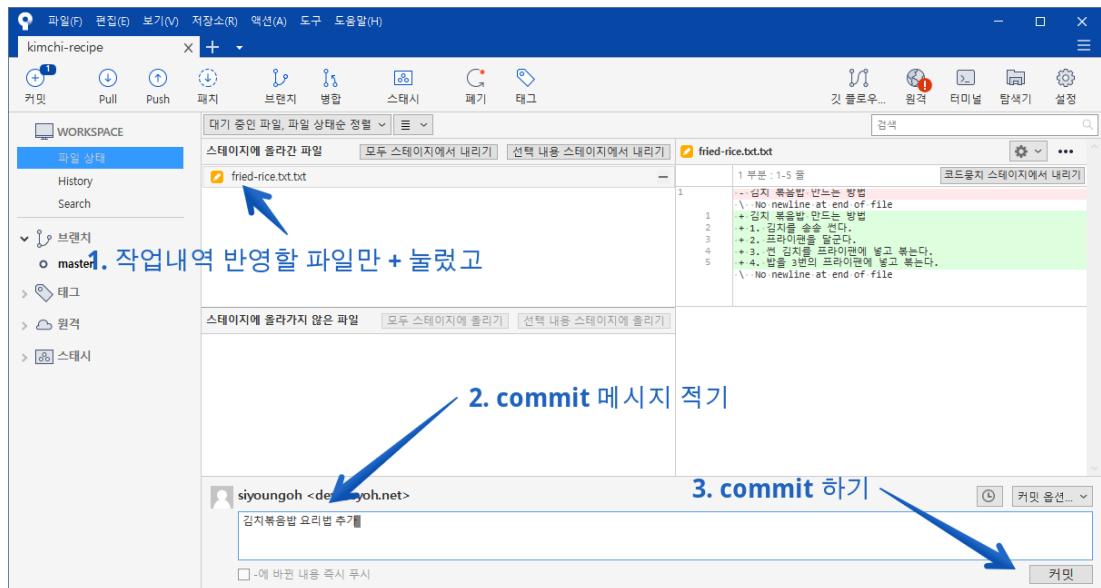
```
김치 볶음밥 만드는 방법
1. 김치를 송송 썬다.
2. 프라이팬을 달군다.
3. 썬 김치를 프라이팬에 넣고 볶는다.
4. 밥을 3번의 프라이팬에 넣고 볶는다.
```

2. 프로젝트 상태 확인

- 이제 sourcetree 를 다시 켜볼까요? 어떻게 보일지 짐작가시죠? 😊
- 이번에도 변경된 내역이 보이네요! 변경되지 않은 **jeon.txt** 파일과 **jjigae.txt** 파일은 이전 commit과 내용이 바뀌지 않았기 때문에 현재 프로젝트 상태를 저장하지 않습니다! Git 이 참 똑똑하죠~!

3. commit 하기

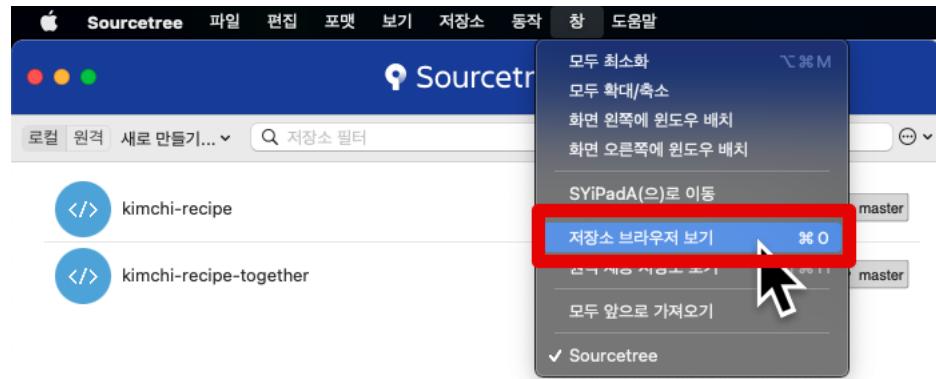
- windows



- mac

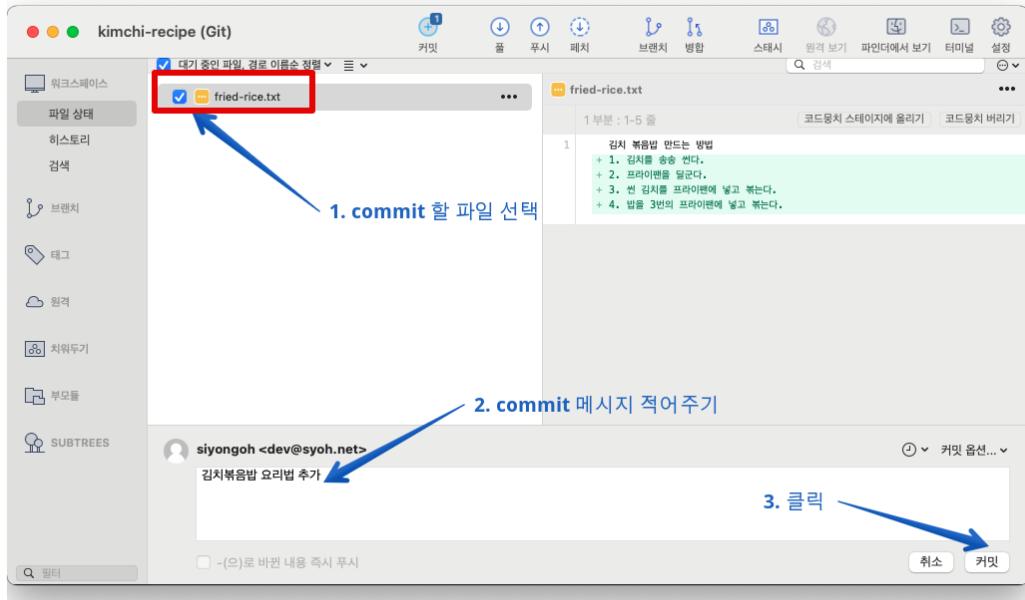
▼ 참고. 만약 아래와 같은 창이 바로 보이지 않는다면

1. 상단 메뉴바 - 창 - 저장소 브라우저 보기 를 클릭하면 아래 창이 됩니다.



2. 여기서 로컬 탭 누르고, kimchi-recipe(내가 만든 repo 이름) 더블 클릭!





▼ 25) 마지막 한 번 더 commit! - 수정된 파일 중에 일부만 commit

- 수정된 파일 중에 내가 작업내역으로 저장하고 싶은 것만 선택해서 commit해보겠습니다.

1. 파일 수정하기

- `fried-rice.txt` 파일에 분홍색으로 표시된 5 항목을 추가해주세요.

▼ [코드스니펫] 21 - fried-rice.txt

```
5. 적당히 볶아졌으면 참기름을 뿌리고 그릇에 옮겨담는다.
```

- `jjigae.txt` 파일에 분홍색으로 표시된 1~2 항목을 추가해주세요.

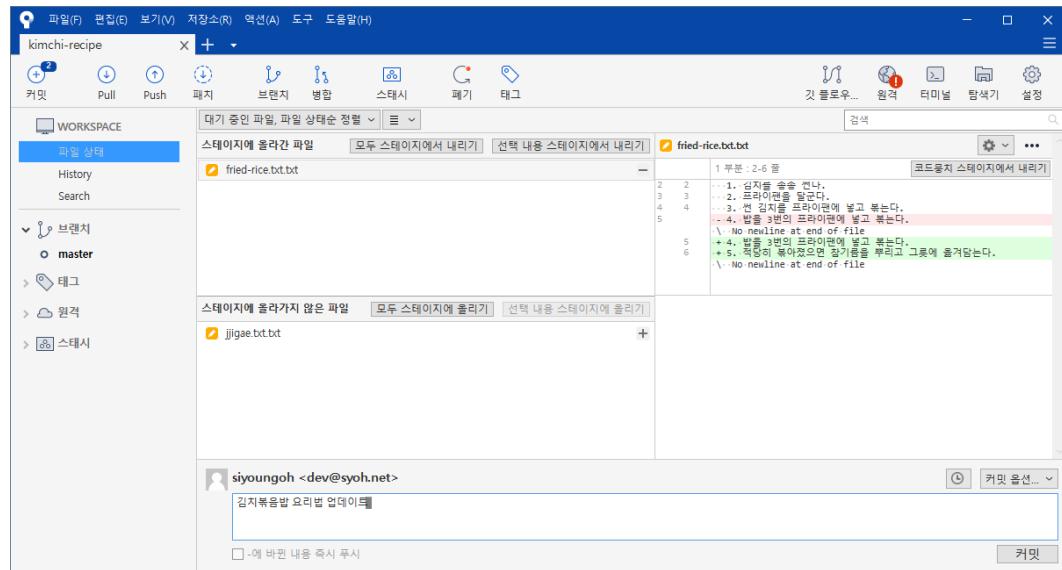
▼ [코드스니펫] 21 - jjigae.txt

```
1. 배추 김치를 얹을만한 크기로 썬다.  
2. 냄비에 물을 끓인다.
```

2. 프로젝트 상태 확인하기

- 이제 sourcetree 를 다시 켜볼까요? 이번에도 변경된 내역이 보이네요! `jeon.txt` 파일은 이전 commit 과 내용이 바뀌지 않았기 때문에 현재 프로젝트 상태를 저장하지 않습니다.

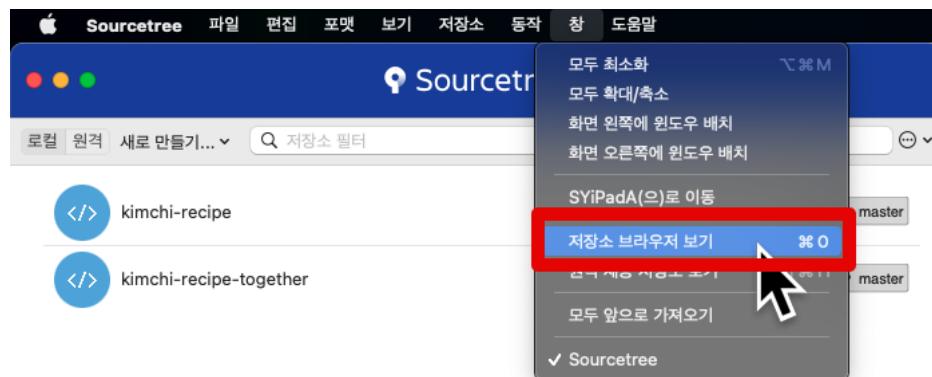
- windows



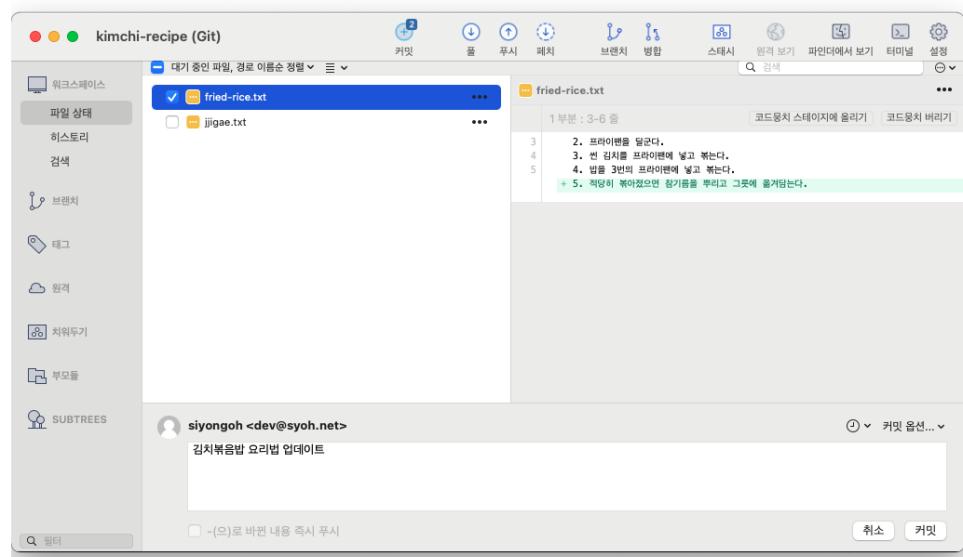
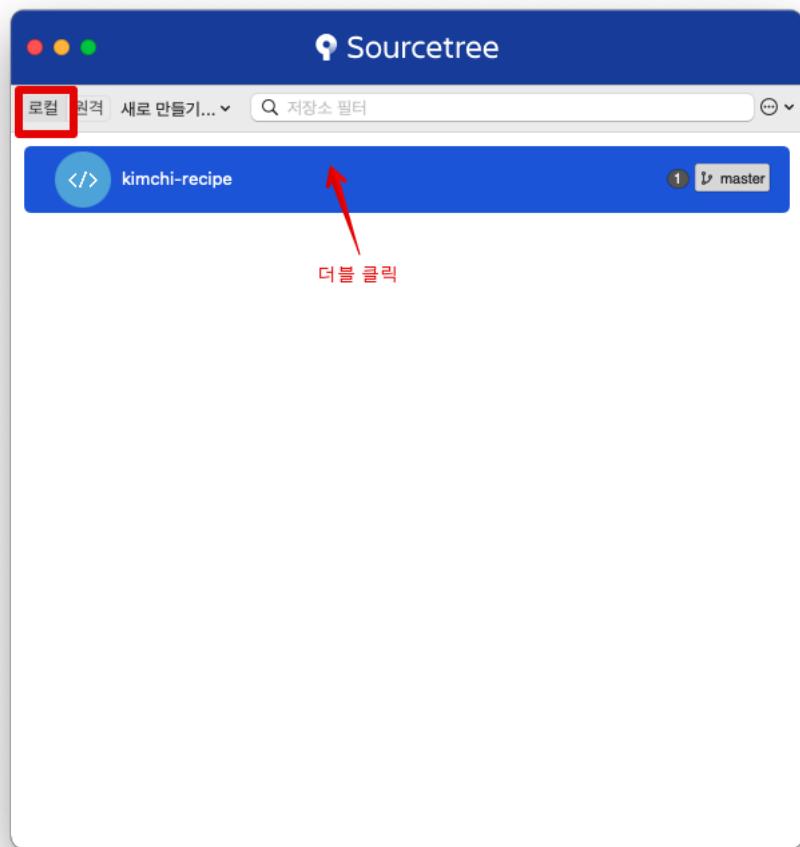
o mac

▼ 참고. 만약 아래와 같은 창이 바로 보이지 않는다면

1. 상단 메뉴바 - 창 - 저장소 브라우저 보기 를 클릭하면 아래 창이 뜹니다.

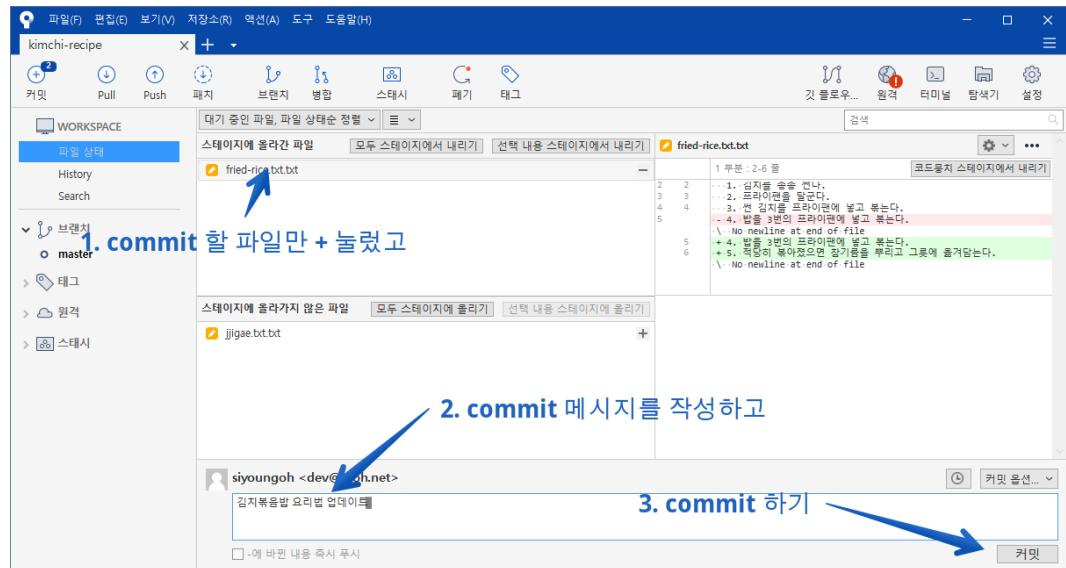


2. 여기서 로컬 탭 누르고, kimchi-recipe(내가 만든 repo 이름) 더블 클릭!



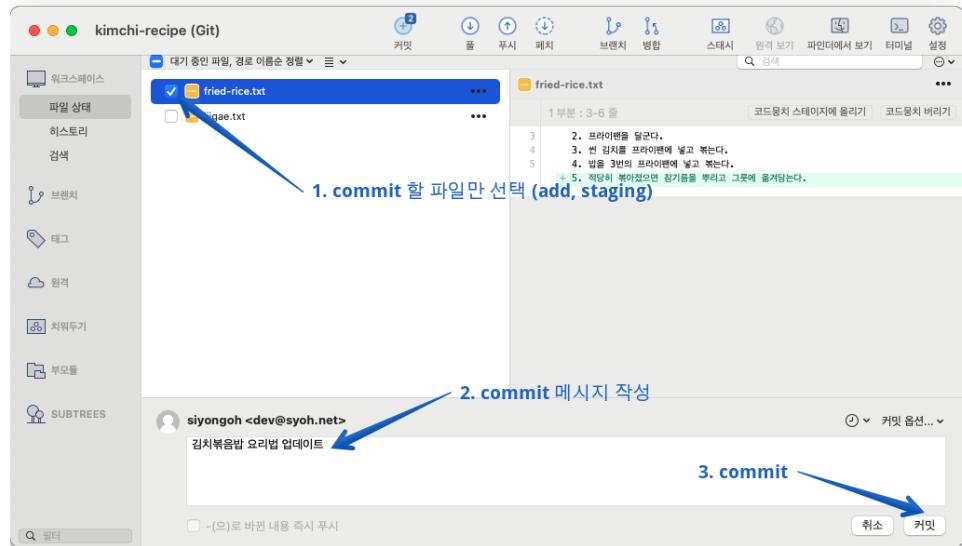
3. commit 하기

- 앗! 근데 아직 김치찌개 요리법은 완성되지 않았네요. 그럼 이번 commit 은 **김치 볶음밥 요리법** 만 반영해보죠! 김치찌개 요리법 파일인 **jjigae.txt** 는 빼고 commit 해봅시다.
 - windows



- o mac

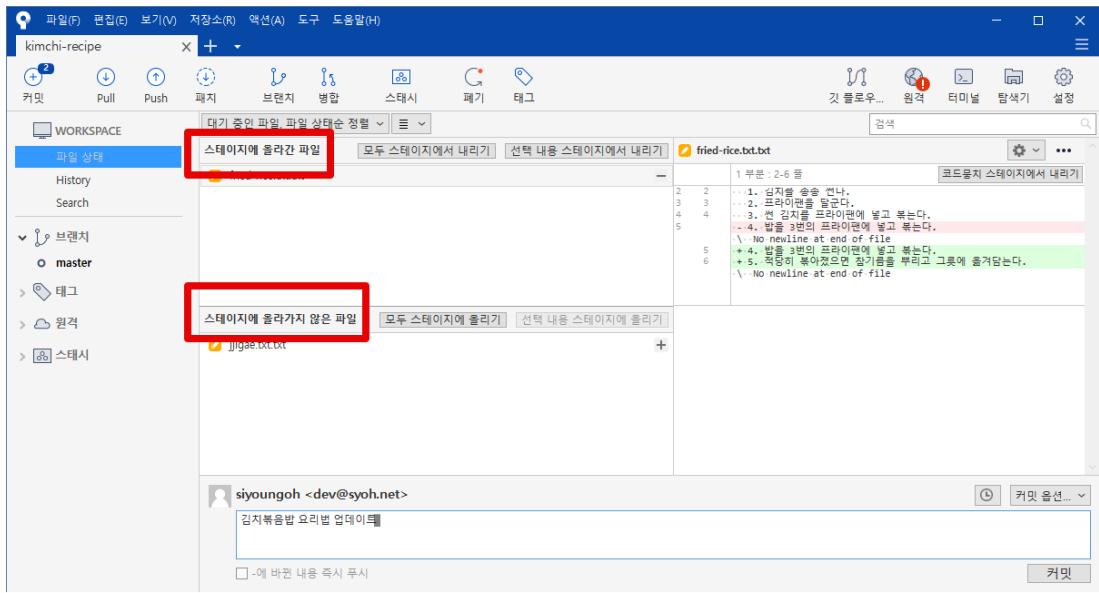
- 내가 저장하기 원하는 파일만 체크해주면 됩니다.



- 이렇게 내가 commit 하기 위한 파일만 선택하는 것을 `add` (또는 `staging`, 스테이징) 이라고 합니다.

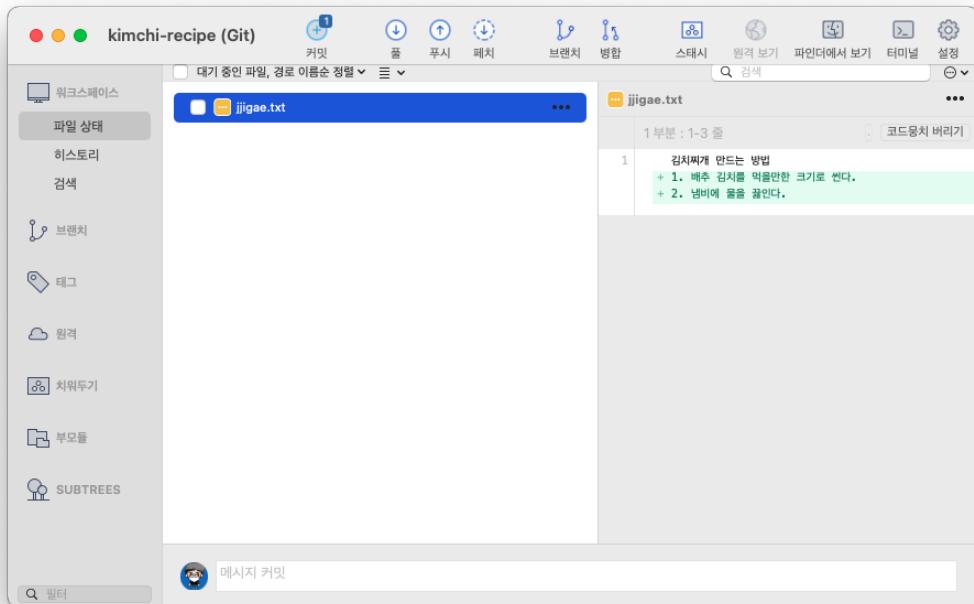
- o windows

- windows 버전 sourcetree 에는 '스테이지' 라고 적혀있죠! 무대(stage)에 올리는 것처럼 내가 commit 하기 원하는 파일만을 선택해 add 하는 과정이구나! 라고 이해하시면 됩니다.



- 아래처럼 commit 한 후에도 `jjigae.txt` 파일은 commit 되지 않은 상태로 남아있게 됩니다.

o mac



👉 `add(staging)` 를 사용하면, 컴퓨터에서 여러 파일을 수정했어도 '기능 A 수정'에 관련된 파일만 골라서 commit 할 수 있어요. 작업내역을 깔끔하게 관리하기 위해서는 꼭 아래처럼 commit 해주세요!

- 내가 기록할 작업 내역이 무엇인지 생각하고,
- 관련된 파일만 add 해서
- 작업내역을 나타내는 commit 메시지 적기!

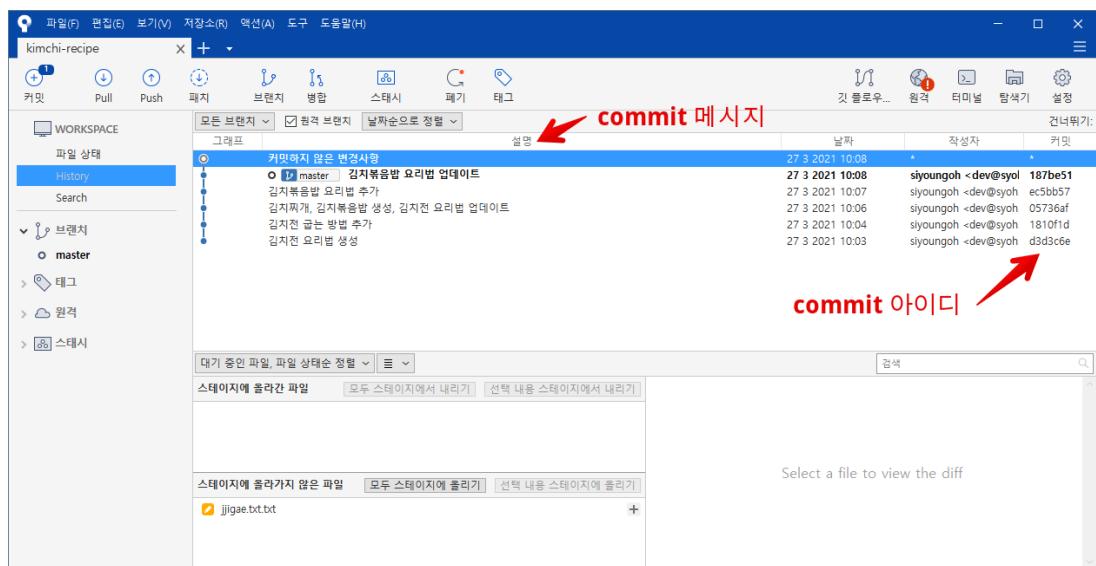
▼ 26) 지금까지 한 commit 내역 보기 - commit history, commit id

- 현재 commit 까지 Git 에 기록된 상태 즉, commit 한 내역을 한 눈에 볼 수 있습니다. 이것을 commit history(커밋 히스토리)라고 합니다. 말 그대로 commit 을 해온 역사 📜입니다!

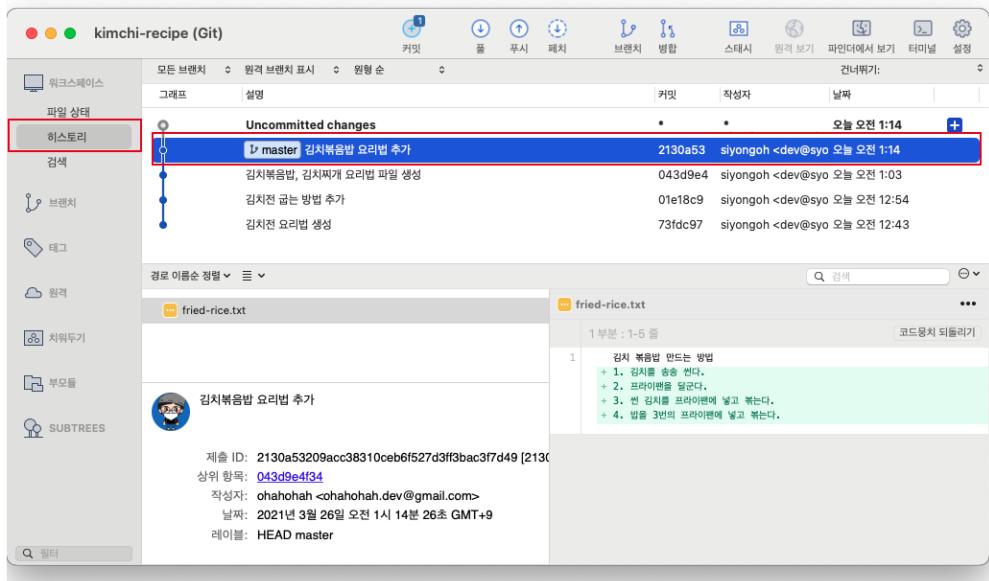
- sourcetree 에 가서 왼쪽 메뉴에서 히스토리를 클릭하세요. 와! 그동안 했던 commit 내역들을 쭉 확인할 수 있네요!
 - 화면을 왼쪽에서 오른쪽 항목으로 따라가면서 살펴보죠!
 - 가장 위 항목이 최근 commit 이군요! 오 commit 들끼리 그래프로 연결도 되어있네요. (이 내용이 무엇을 뜻하는지는 2주차에 배웁니다 😊)
 - 설명 항목에는 내가 작성한 commit message 가 보여요! 아하! 작업을 어떻게 해왔는지를 commit message 로 확인할 수 있군요! 해당 commit 메시지를 클릭하면 하단에 자세한 내용도 보이네요.
 - 날짜, 작성자 는 commit 한 일시와 작성자를 뜻하고요.
 - 맨 위에 Uncommitted changes(커밋되지 않은 변경사항) 는 아직 커밋되지 않는 변경사항이 있다고 알려주는 것입니다. commit 할게 남아있어~ 하고 알려주는 거예요.
 - 커밋 이라고 적힌 부분(아래 그림의 2130a53, 043d9e4 등)은 해당 commit 에 붙여주는 유일한 값이에요. 이것을 commit id(커밋 아이디) 라고 합니다. commit 을 구분하기 위해 git 이 붙여주는 아이디라고 생각하면 됩니다. 사용자를 구분하기 위해서 사용자 아이디가 있는 것처럼요!

commit id 는 나중에 커밋을 관리하고 되돌릴 때 사용하는 중요한 정보예요!
앞으로 이 commit id 를 사용해 작업하는 방법을 배워볼 거예요!
각 commit 을 관리하기 위해 id 가 부여된다! 체크 ✓

- windows



- mac



- commit은 아래 정보를 포함합니다. 외우지 않아도 됩니다! commit을 구분하고 잘 사용하기 위한 정보는 이런 게 있구나~ 하고 넘어가면 됩니다!
 - commit id : commit을 구분하기 위한 유일한 값.
 - 작업 일자 (날짜와 시간)
 - 작업한 사람(작성자 author)
 - 작업 내역 (commit 메시지)
 - 작업 내역의 순서 : 해당 commit의 직전 commit이 무엇인지 정보

05. 버전관리와 commit - 정리



정리하기 보기 전에 먼저! 스스로 배운 내용을 정리하고 실습해보세요!

몰라도 괜찮고 틀려도 괜찮아요! 내가 모르는 것이 뭔지 알아야 제대로 복습할 수 있습니다. 스스로 해보지 않고 눈으로 보는 건 배우는 게 없어요. 연필 들고 직접 적어보세요!

▼ 27) [💡 배웠으면 써먹자] - 개념 적어보기

- 종이와 펜을 준비해서 직접 써보세요. 머리로 말고 직접 글로 적어 주세요. 그래야 내가 제대로 아는지 정확히 확인할 수 있어요. 수업자료를 확인하기 전에 스스로 먼저 적어보세요!



잠깐, 이건 암기력 테스트가 아니에요. 스스로 개념을 정리해야 머리 속에 남습니다.

모르면 알아가면 되죠! 수업 자료 어디 안 도망가요! 📚🏃‍♂️💡 다시 찾아보면 됩니다. 괜찮아요.

내가 모르는 부분을 알아야 그 부분을 집중적으로 복습할 수 있습니다.

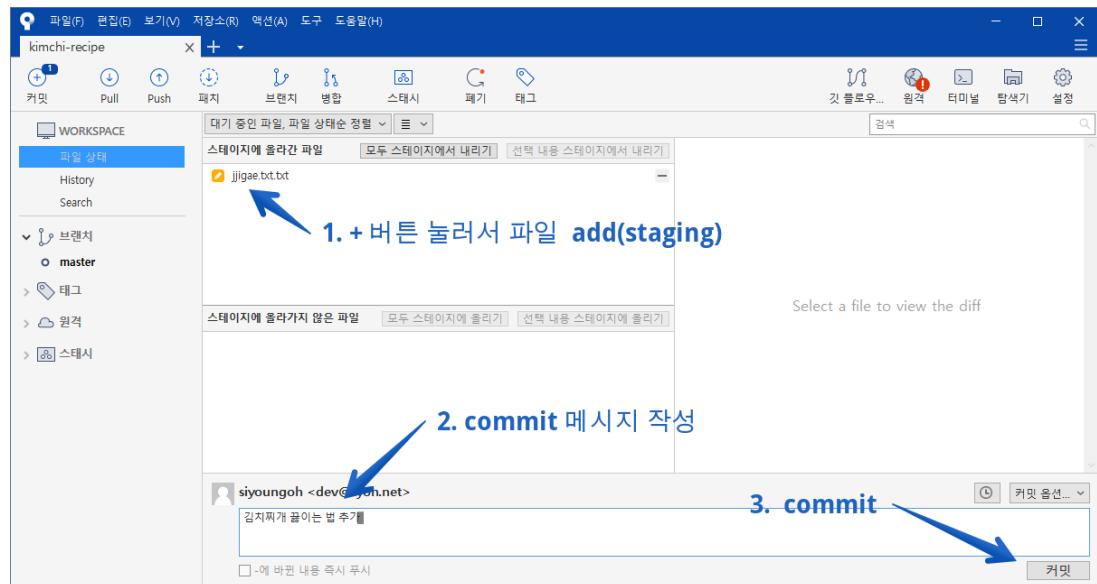
- 버전관리를 한다는 건 어떤 의미일까?
 - 작업내역 단위인 commit에는 어떤 정보가 포함되어 있어야 잘 버전관리를 할 수 있을까?
 - 지금까지 우리가 실습은 어떤 순서로 했었지?
- 위 내용을 스스로 적었으면 25) 정리하기 를 확인하며 다시 내용을 정리해보세요.

▼ 28) [💡 배웠으면 써먹자] - commit 혼자 해보기

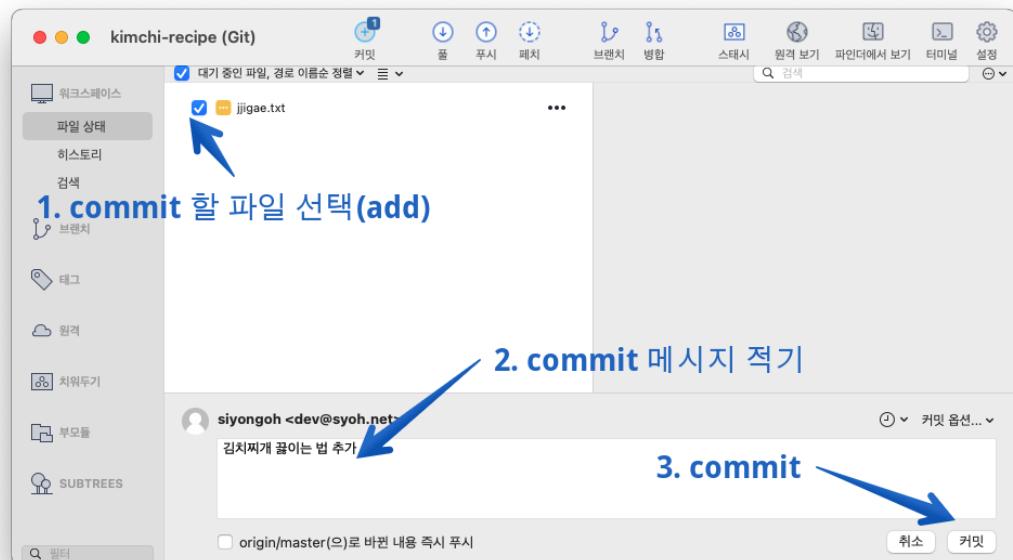
- 아까 commit하지 않은 김치찌개 요리법 파일 jjigae.txt를 commit해보세요.

▼ A. 이미지로 정답 확인하기

- windows



- mac



👉 다음 과정 하기 전에 꼭 해주세요~! 그렇지 않으면 에러가 발생할 수 있어요!

▼ 29) 정리하기

- 버전관리를 한다는 것은 프로젝트 상태가 변경되는 정보를 알고 있다는 것입니다. Git 은 commit 을 사용해서 버전이 달라지는 것을 관리합니다.
- 컴퓨터에 있는 프로젝트를 Git 이 관리하는 프로젝트로 만들 수 있습니다. 앞으로 Git 으로 관리할꺼야! 하고 설정해주면 됩니다. 이 작업을 git 초기화(git initialize)한다고 표현합니다.
- 현재 프로젝트의 상태를 찰칵 📸 저장하는 것을 commit 이라고 합니다.
- commit 에는 아래를 포함합니다.
 - 누가(author), 언제 commit 했는지의 정보와 프로젝트 변경 내용
 - 작업내역이 어떤 것인지 알아볼 수 있게 적는 메시지를 'commit 메시지'라고 합니다.
- commit 에 반영할지 안할지는 파일 단위로 선택할 수 있습니다. commit 에 반영할 파일을 선택하는 것을 add (혹은 staging, 스테이징) 이라고 합니다.
- commit 한 기록은 history 로 볼 수 있습니다.
- 지금까지 우리가 한 작업은 'git 초기화하기(initialize) - add(staging) - commit' 입니다.
 - git 초기화는 처음에 단 한번만 해 주면 됩니다. 작업 내역을 저장하기 위해서는 add - commit 만 하면 됩니다.

▼ 23) [배웠으면 써먹자] - 개념 적어보기 정답 예시

1. 버전관리를 한다는 건 어떤 의미일까?

- 프로젝트 상태가 변경되는 정보를 알고 있다는 것입니다. Git 은 누가, 언제, 해당 시점의 프로젝트 상태를 기록해두기 때문에 버전 관리가 가능합니다.

2. 작업내역 단위인 commit 에는 어떤 정보가 포함되어 있어야 잘 버전관리를 할 수 있을까?

- 누가, 언제, 무엇을 했는지가 필요하겠죠? 어떤 작업을 했는지 알려주는 작업 내역인 commit 메시지, 누가(author), commit 한 시간이 필요해요.
- 그리고 commit 을 편하게 관리하기 위한 commit 아이디! commit 아이디는 사용자 아이디처럼 commit 에 부여된 유일한 값!

3. 지금까지 우리가 실습은 어떤 순서로 했었지?

- 기존의 프로젝트를 git 프로젝트로 만든다. - git 초기화(git initialize)
- commit 할 파일들을 add(staging) 하고 commit 메시지 작성해서 commit!

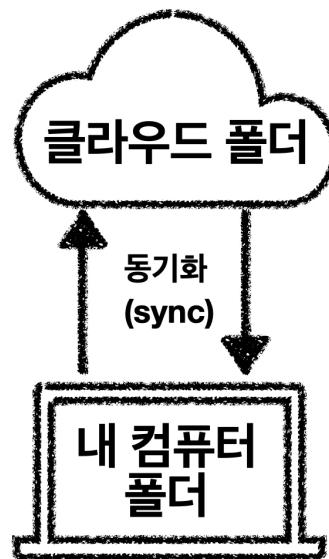
06. 원격 repo 사용하기 - 개념탐재

▼ 30) 원격 repo와 로컬 repo 가 뭐예요?

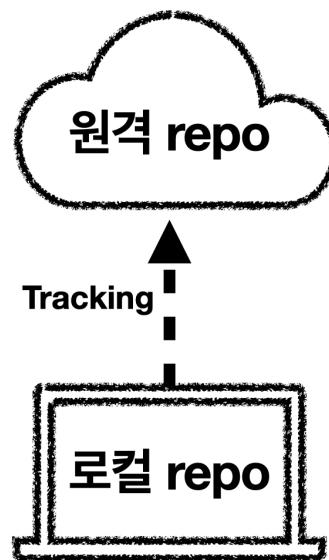
- 내 컴퓨터에 저장된 Git 프로젝트는 오직 이 컴퓨터에서 나만 볼 수 있는 걸까요? 아닙니다! 다른 사람과 공유하고 다른 사람도 볼 수 있어야 Git 으로 협업할 수 있겠죠!
- 'Git으로 관리되는 프로젝트' 를 Git 에서는 repo(리포, repository 리포지토리의 약자) 라고 부릅니다.
- 내 컴퓨터에 저장되어있는 리포지토리를 로컬 repo(local repository) 라고 합니다. Github 처럼 다른 곳에서 접속할 수 있는 공간에 저장되어있는 것을 원격 repo(remote repository) 라고 합니다.
- Github은 원격 repo 가 저장되어있고 + 개발자 커뮤니티 기능을 하는 서비스입니다. 우리는 Github 을 사용해서 원격 repo 를 만들고 관리해볼게요!

▼ 31) 원격 repo 와 로컬 repo 같이 사용한다는 것은?

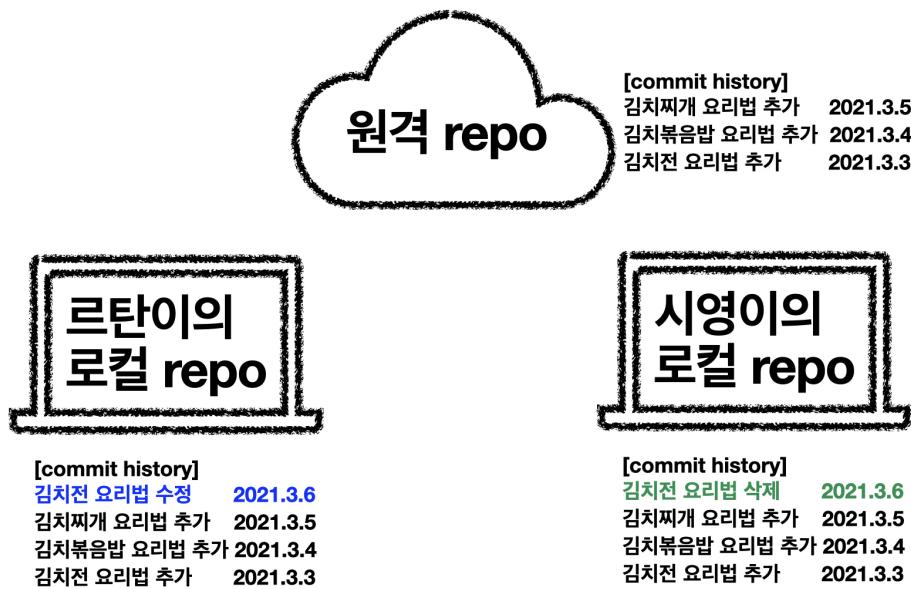
- 먼저 one drive, google drive 같은 클라우드 서비스를 사용하는 걸 생각해볼게요. 클라우드에 있는 폴더를 내 컴퓨터에 있는 폴더를 연결해서 동기화해두면 내 컴퓨터 폴더의 내용이 자동으로 클라우드 폴더와 똑같아지죠?



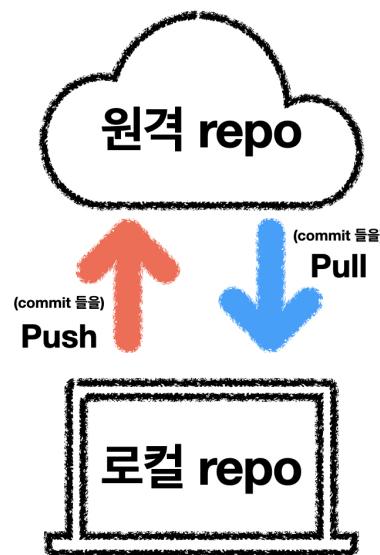
- Git 도 클라우드 서비스로 두 군데의 내용을 동기화한 것처럼 원격 repo와 로컬 repo 를 연결시켜서 내용을 반영시킬 수 있어요. 로컬 repo 가 원격 repo 를 연결하는 것을 **추적(Tracking, 트랙킹 / branch tracking)** 이라고 해요.
 - 로컬 repo 만이 내가 어떤 원격 repo 와 연결되어있는지를 알고 있어요. 원격 repo 는 내가 어떤 로컬 repo 와 연결되어있는지 정보를 가지고 있지 않아요. 언제나 로컬 repo 를 기준으로 동작을 이해해주세요!



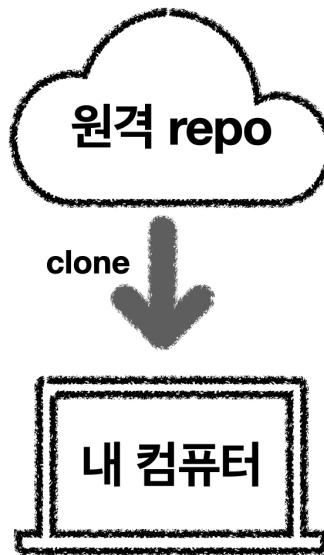
- 단! 클라우드 서비스와 다르게 작업내역 즉, commit 을 자동으로 반영하지 않습니다! 내가 원하는 대로 어디 commit 까지만 반영 할지를 수동으로 설정할 수 있게 해서 프로젝트를 더 잘 관리하기 위해서입니다!
 - 협업할 때 특히 commit 을 원하는만큼 반영하는 것이 필요해요! 만약 commit 이 자동으로 반영된다면 아래처럼 같이 협업할 때 누구는 요리법 수정을 했고, 누구는 요리법 삭제를 했는데 둘 중에 어떤 내용을 반영해야할지 난감하겠죠?
 - 2주차에서 자세하게 이렇게 협업하는 내용에 대해서 배울 꺼에요. commit 을 수동으로 반영해야한다! 만 기억해주세요 😊



- 로컬 repo 의 commit 들을 원격 repo 에 반영하는 것을 **push(푸쉬)**이라고 해요. commit들을 밀어넣기!
- 원격 repo 의 commit 들을 로컬 repo 에 반영하는 것을 **pull(풀)** 이라고 해요. commit들을 땡겨오기!
- 로컬 repo 를 기준으로 생각하면 되겠죠? 나(로컬 repo)의 내용을 보내주는 거니까 push! 나(로컬 repo)로 내용을 땡겨오는 거니까 pull!



- 원격 repo 를 내 컴퓨터에서도 사용할 수 있도록 가져올 수도 있어요. 일종의 초기 다운로드라고 생각하면 됩니다. 이걸 **clone(클론, 복제)** 라고 해요.



07. 원격 repo 사용하기 - 실습

▼ 32) tracking- 로컬 repo 와 원격 repo 연결하기

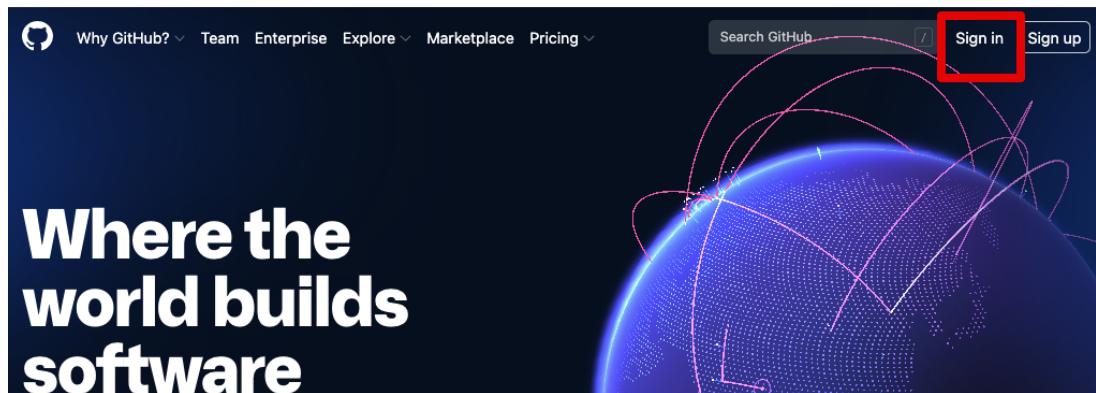
- 원격 repo 가 없으므로 Github 에 원격 repo 를 만들고, 내 컴퓨터에 있는 로컬 repo 와 연결시켜볼게요!

▼ 1. Github 에 접속하기

- Github 사이트에 접속해서 상단 sign in 을 눌러서 로그인해주세요.

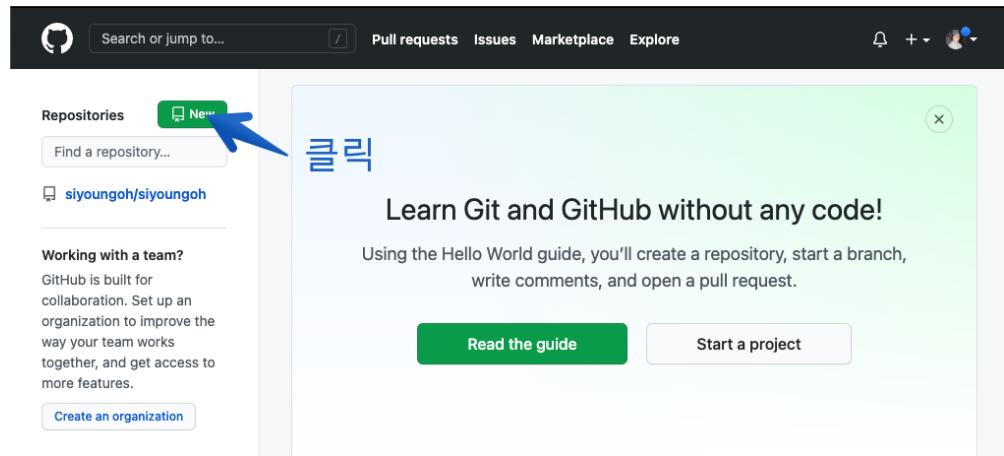
▼ [코드스니펫] Github 링크

<https://github.com>



▼ 2. repo 만들기

- 로그인 후 뜨는 페이지에서 초록색 new 버튼을 눌러서 repo를 만들어보겠습니다.



- 프로젝트 설명 페이지, 라이센스 등 여러 설정을 할 수 있지만 일단은 가장 간단한 형태로 만들어볼게요!
 - repository name (repo 이름) : kimchi-recipe
 - Description : 김치요리 저장소

 repo 이름과 description 은 프로젝트가 무엇인지 나타낼 수 있게 잘 적어주어야합니다! 정보를 잘 표현할 수 있는 이름짓기(naming)은 프로그래밍에서 중요해요!

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

1. 이름 입력

Owner *  siyoungoh	Repository name * <input style="width: 150px; border: 1px solid #ccc; border-radius: 5px; padding: 2px 5px;" type="text" value="kimchi-recipe"/> ✓
--	--

Great repository names are short and memorable. Need inspiration? How about [animated-octo-enigma](#)?

2. 어떤 리포지토리인지 설명

김치요리 저장소

3. 공개로 설정

 Public
Anyone on the internet can see this repository. You choose who can commit.

 Private
You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

Add a README file
 This is where you can write a long description for your project. [Learn more](#).

Add .gitignore
 Choose which files not to track from a list of templates. [Learn more](#).

Choose a license
 A license tells others what they can and can't do with your code. [Learn more](#).

4. 클릭

[Create repository](#)

- 아래처럼 뜨면 repo 가 잘 만들어진겁니다! 빨간 부분이 내 원격 repo 의 주소입니다. 버튼을 클릭하면 주소가 복사됩니다.

- 아직 창을 닫지 마세요. 이따가 설정에 원격 repo 주소를 사용해야해요.

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH **https://github.com/siyoungoh/kimchi-recipe.git** ⌂

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# kimchi-recipe" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/siyoungoh/kimchi-recipe.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/siyoungoh/kimchi-recipe.git
git branch -M main
git push -u origin main
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code ⌂

💡 **ProTip!** Use the URL for this page when adding GitHub as a remote.

▼ 3. Tracking 하기 Github 에 있는 repository 와 내 컴퓨터에 만들어놓은 repository 연결하기

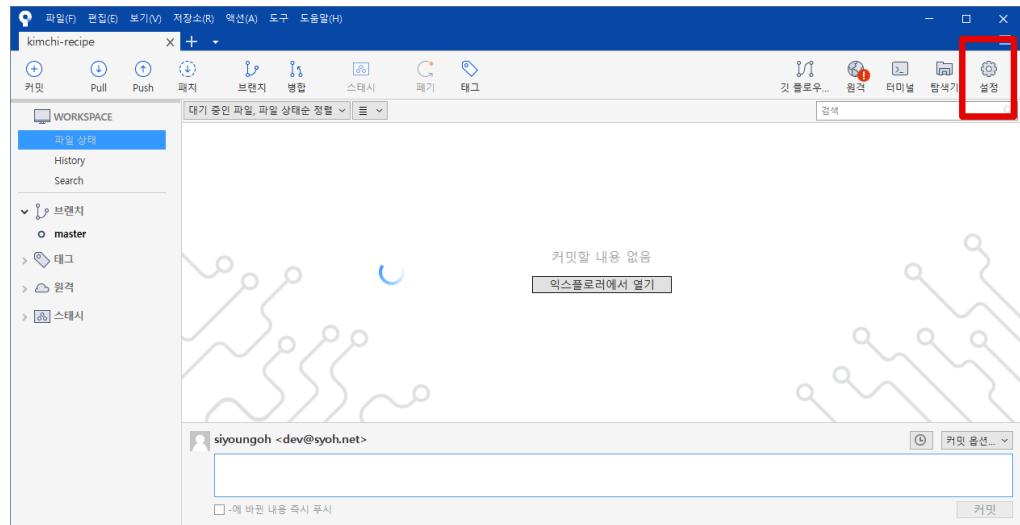
- 방금 만든 Github 원격 repo 와 내 컴퓨터에 저장되어있는 로컬 repo 를 연결하겠습니다. 두 repo 를 연결해두면 로컬 repo 의 commit 들과 원격 repo commit들을 쉽게 동기화 시킬 수 있습니다.



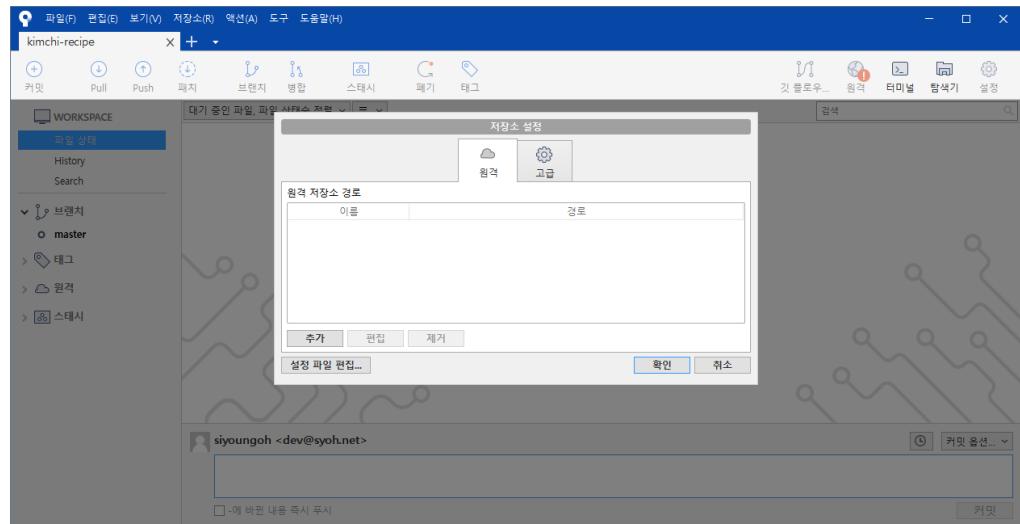
로컬 repo 가 원격 repo 를 추적(tracking) 하는 겁니다! 연결되어있다는 정보는 로컬 repo 에만 있습니다. 언제나 로컬 repo 를 기준으로 동작을 이해해주세요!

▼ Windows 에서 설정하기

- sourcetree 에 들어가서 `kimchi-recipe` 창에서 `설정` 버튼을 눌러주세요

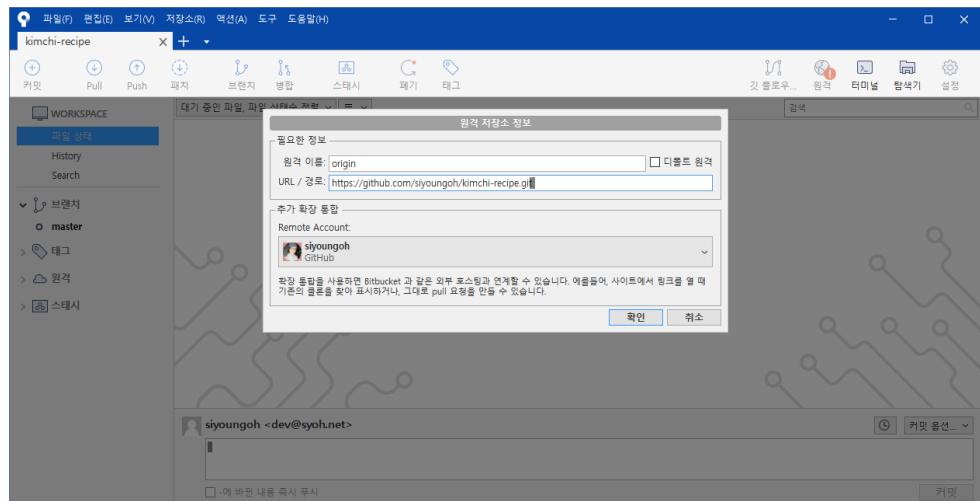


- 설정창이 뜨면 원격 - 추가를 누르세요.



- 아래처럼 적어주세요.

- 원격 이름 : origin 을 적어주세요. origin 은 원격에 연결하는 저장소를 말할 때 일컫는 이름이에요. 아하 보통 이렇게 쓰는구나 하고 넘어가시면 됩니다!

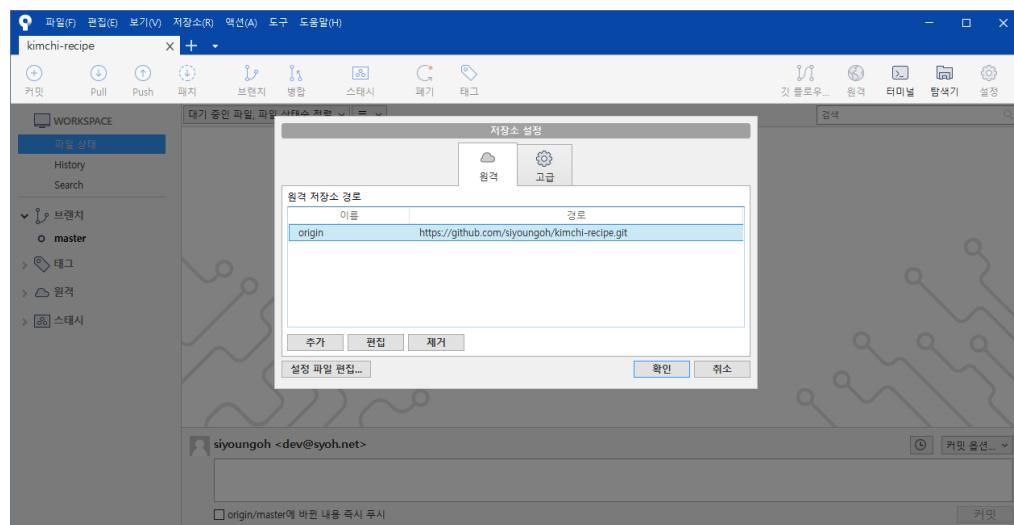


- url/ 경로 : 아까 만들어주었던 github 의 주소를 넣어주세요.

▼ 참고 - 내 Github repo url 주소 확인하기

The screenshot shows a GitHub repository page for 'siyoungoh/kimchi-recipe'. At the top, there's a navigation bar with tabs for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and more. Below the navigation bar, the repository name 'siyoungoh/kimchi-recipe' is displayed. In the center, there's a 'Quick setup' section with a red box highlighting the URL input field. The URL 'https://github.com/siyoungoh/kimchi-recipe.git' is entered in this field. A blue arrow points from the right towards the URL field. Below the URL field, there's a note: 'Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and more.' To the right of the URL field, there are icons for copy and open in new tab.

- 앞에 설정이 잘 되었다면 아래처럼 보일 꺼에요. 여러분들은 siyoungoh 대신 여러분들 github 이름이 보이겠죠? 여기에서 확인 버튼을 누르면 됩니다.

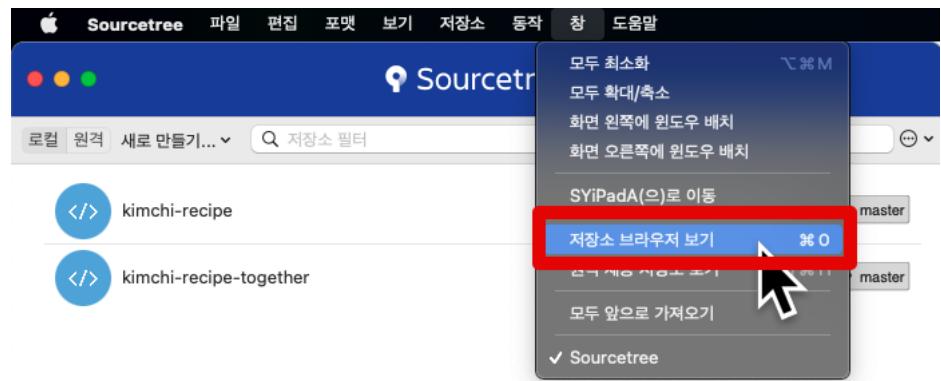


▼ mac에서 설정하기

- sourcetree 에 들어가서 `kimchi-recipe` 창에서 설정 버튼을 눌러주세요.

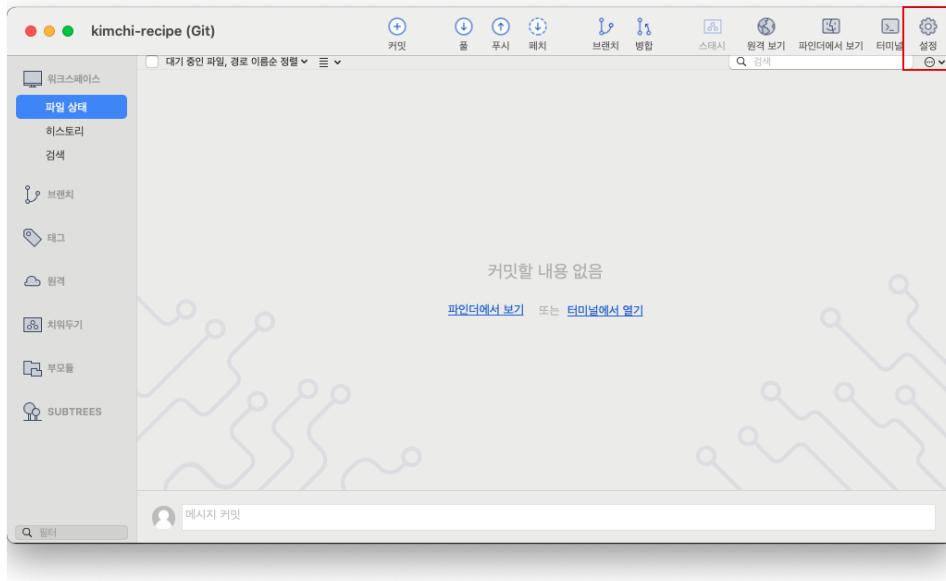
▼ 참고. 만약 아래와 같은 창이 바로 보이지 않는다면

1. 상단 메뉴바 - 창 - 저장소 브라우저 보기 를 클릭하면 아래 창이 뜹니다.

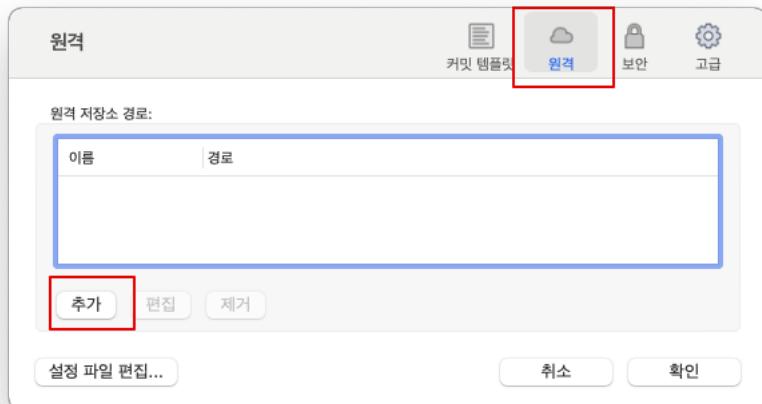


2. 여기서 로컬 탭 누르고, kimchi-recipe(내가 만든 repo 이름) 더블 클릭!





- 설정창이 뜨면 원격 - 추가를 누르세요.

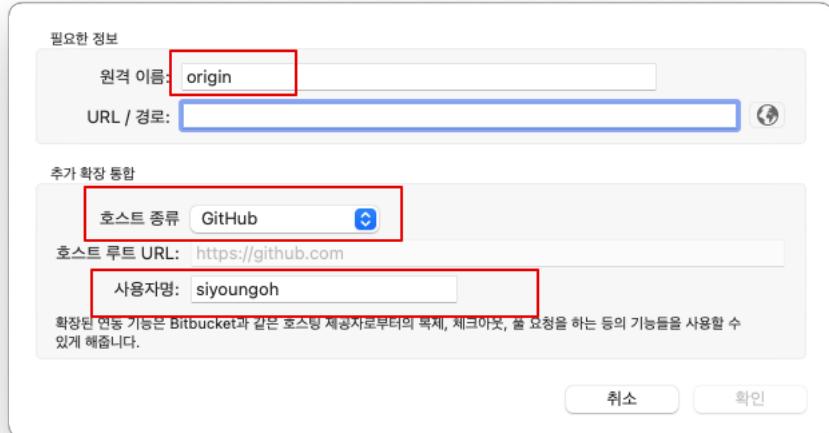


- 아래처럼 적어주세요. 사용자명은 내 github 사용자명을 적어주세요.
 - origin 은 원격에 연결하는 저장소를 말할 때 일컫는 이름이에요. Git 의 기본 설정된 이름이랍니다. 아하 보통 이렇게 쓰는구나 하고 넘어가시면 됩니다!

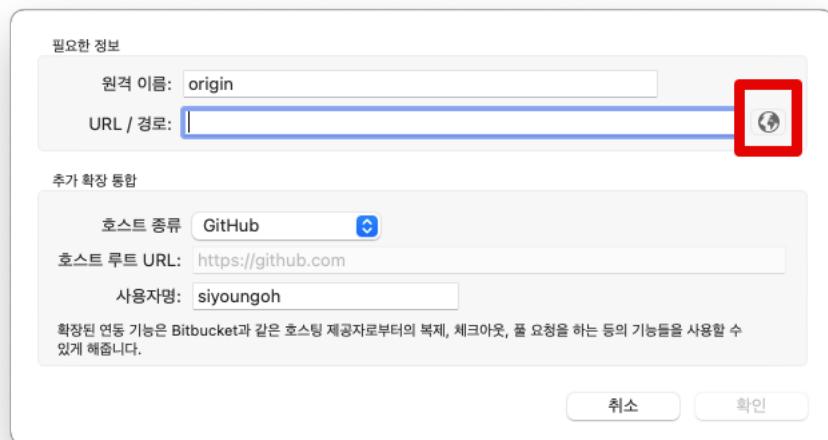
▼ 왜 origin 인지 너무너무 궁금하면 열어보세요!

밑에 내용이 잘 이해 안 가도 괜찮아요!
김치가 몇 세기에 만들어졌는지 아는 것처럼 아주아주 부가적인 내용이라고 생각하면 됩니다!

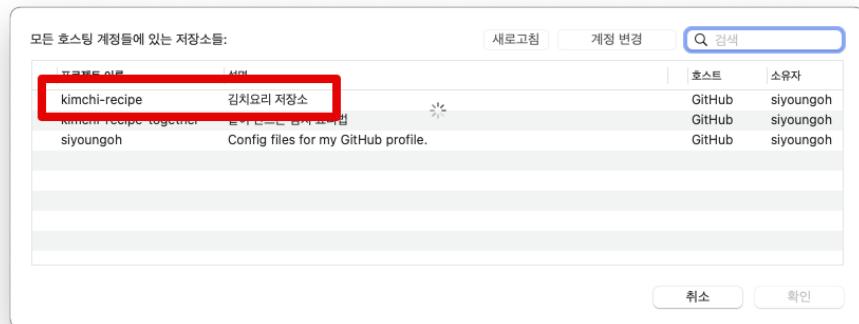
- 나중에 배우게 될 clone (원격 repo 복제 해오는 것) 에서 생겨난 말이에요. 하나의 local repo 에 여러 원격 repo 를 연결시킬 수 있어요. 그 중에서 기준으로 clone 해온 원격 repo를 구분하기 위해서 'originally cloned 된 원격 repo' 를 줄여서 origin 이라고 부릅니다. 여러 repo를 연결해서 사용하는 건 아주 나중에 계속 쓰다보면 사용하는 경우가 있어요. 통상적으로 하나의 원격 repo 만 연결해서 쓰는 경우가 더 많답니다!



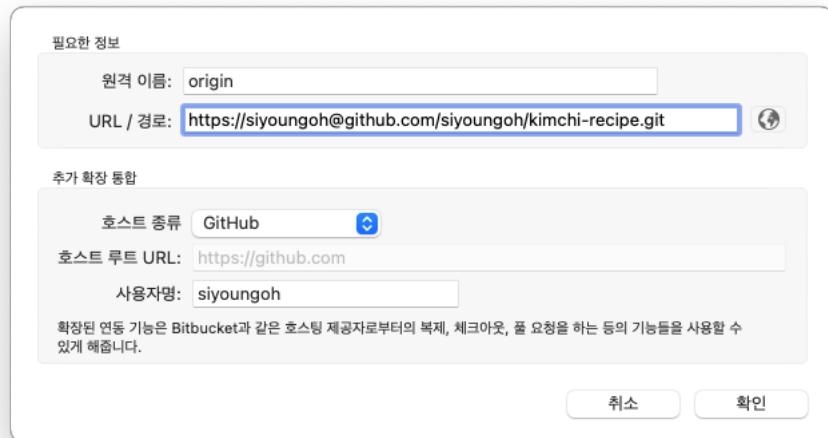
- URL/경로 옆의 지구 모양 버튼을 눌러주세요.



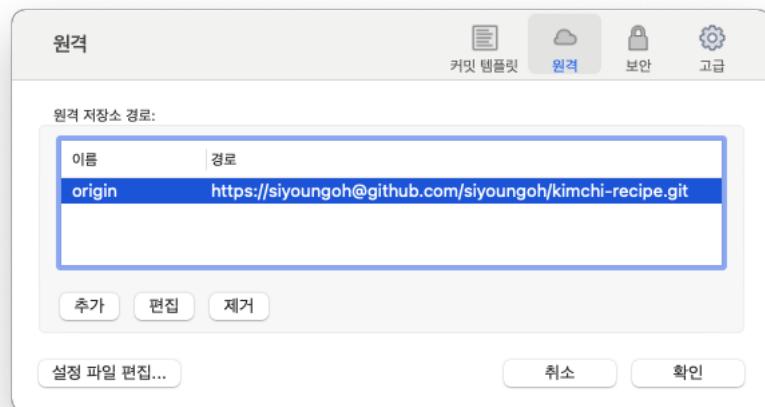
- 뜨는 창에서 kimchi-recipe 를 선택해주세요.



- 아래처럼 모든 정보가 채워졌으면 확인 버튼을 눌러주세요.



- 앞에 설정이 잘 되었다면 아래처럼 보일 거예요. 여러분들은 siyoungoh 대신 여러분들 github 이름이 보이겠죠? 여기에서 확인 버튼을 누르면 됩니다.



▼ 4. 로컬 repo 의 설정 바꿔주기 - 브랜치 이름 바꾸기

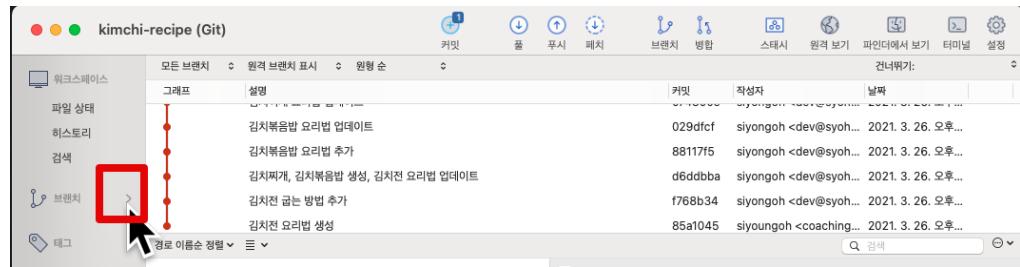


Github 의 기본 설정과 맞춰주기 위해서 먼저 설정을 하나 바꿔주고 갈게요!
여기서 이야기하는 '브랜치'는 2주차에 곧 배울 거예요.

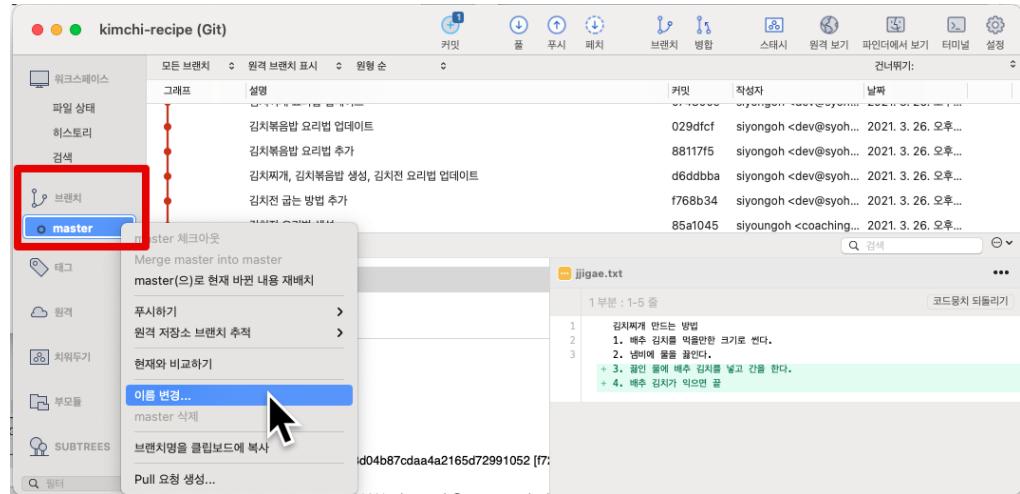
▼ Windows

▼ mac

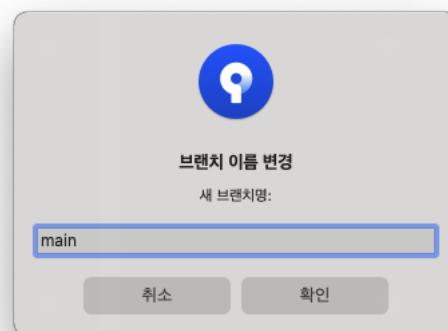
1. 브랜치 항목 옆에 살짝 커서를 가져다 대면 > 표기가 나타납니다. > 를 클릭하면 master 가 보입니다.



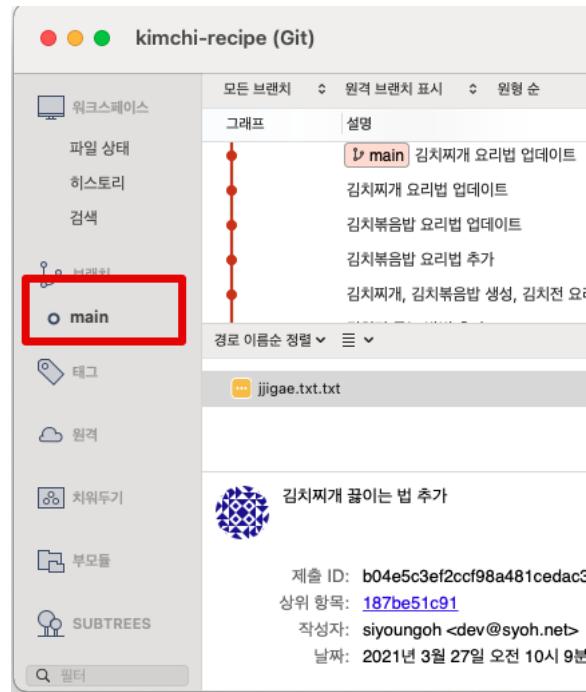
2. master에서 우클릭 - 이름 변경을 눌러주세요.



3. 브랜치 이름 변경을 `main` 으로 바꿔서 적어준 후 확인을 눌러주세요.



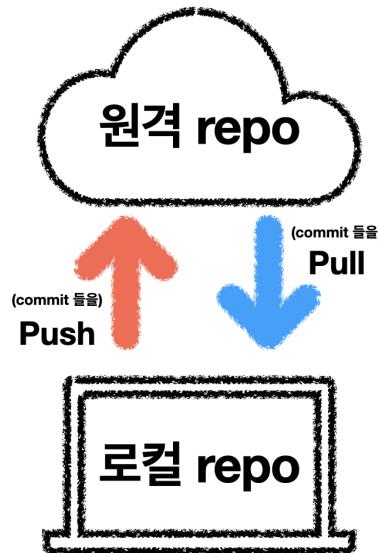
4. 아래 빨간 네모처럼 보면 이름이 잘 변경된 것입니다.



▼ 33) push - 로컬 repo 의 commit 들을 원격 repo commit들에 합치기

▼ 34) pull - 원격 repo 의 commit 들을 로컬 repo commit들에 합치기

- 원격 repo 의 commit 들을 로컬 repo commit들 내역에 합치는 것을 **pull(풀)** 이라고 해요. commit들을 땡겨오기! 마치 commit 내역들 다운로드 같네요!



▼ 1. 원격 repo 에 commit 만들기

- 로컬 repo 에 반영할 commit 이 없으니까 원격 repo를 수정해서 원격 repo에 commit 을 만들어줄게요. 로컬 repository 와 다른 commit 이 있어야 그 내용을 반영할 수 있겠죠? commit 이 원격 repo 와 로컬 repo 가 모두 같으면 가져올 게 없잖아요 그쵸? 😊
 - github 페이지에 가서 jjigae.txt 를 바꾸고 commit 해볼게요.
1. jjigae.txt 를 클릭해서 연필모양 버튼(수정하기) 를 눌러주세요.

siyoungoh / kimchi-recipe

Code Issues Pull requests Actions Projects Wiki Security Insights ...

master 1 branch 0 tags Go to file Add file Code

About 김치요리 저장소

Releases No releases published Create a new release

Packages No packages published Publish your first package

Help people interested in this repository understand your project by adding a README. Add a README

siyoungoh / kimchi-recipe

Code Issues Pull requests Actions Projects Wiki Security Insights ...

master Go to file ...

kimchi-recipe / jjigae.txt

siyoungoh 김치찌개 요리법 업데이트 Latest commit 074800e 1 hour ago History

1 contributor

3 lines (3 sloc) 112 Bytes Raw Blame ⌂ ⌂

```

1 김치찌개 만드는 방법
2 1. 배추 김치를 먹을만한 크기로 썬다.
3 2. 냄비에 물을 끓인다.
4 3. 끓인 물에 배추 김치를 넣고 간을 한다.
5 4. 배추 김치가 익으면 끝

```

2. 편집창에서 아래처럼 3,4 항목을 추가해주겠습니다.

▼ [코드스니펫] pull - 김치찌개

```

김치찌개 만드는 방법
1. 배추 김치를 먹을만한 크기로 썬다.
2. 냄비에 물을 끓인다.
3. 끓인 물에 배추 김치를 넣고 간을 한다.
4. 배추 김치가 익으면 끝

```

siyoungoh / kimchi-recipe

Code Issues Pull requests Actions Projects Wiki Security Insights ...

kimchi-recipe / jjigae.txt in master Cancel Changes

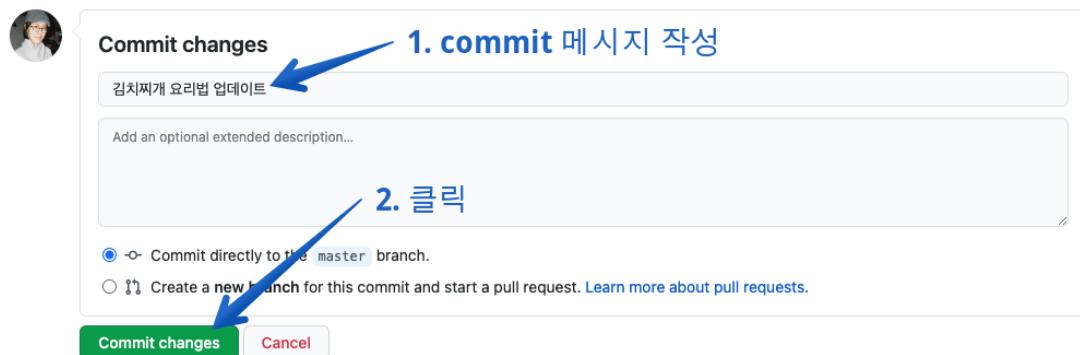
Edit file Preview changes Spaces 2 Soft wrap

```

1 김치찌개 만드는 방법
2 1. 배추 김치를 먹을만한 크기로 썬다.
3 2. 냄비에 물을 끓인다.
4 3. 끓인 물에 배추 김치를 넣고 간을 한다.
5 4. 배추 김치가 익으면 끝

```

3. commit 메시지를 잘 적어주고 commit 버튼 누리기



4. commit 이 잘 반영되었네요!

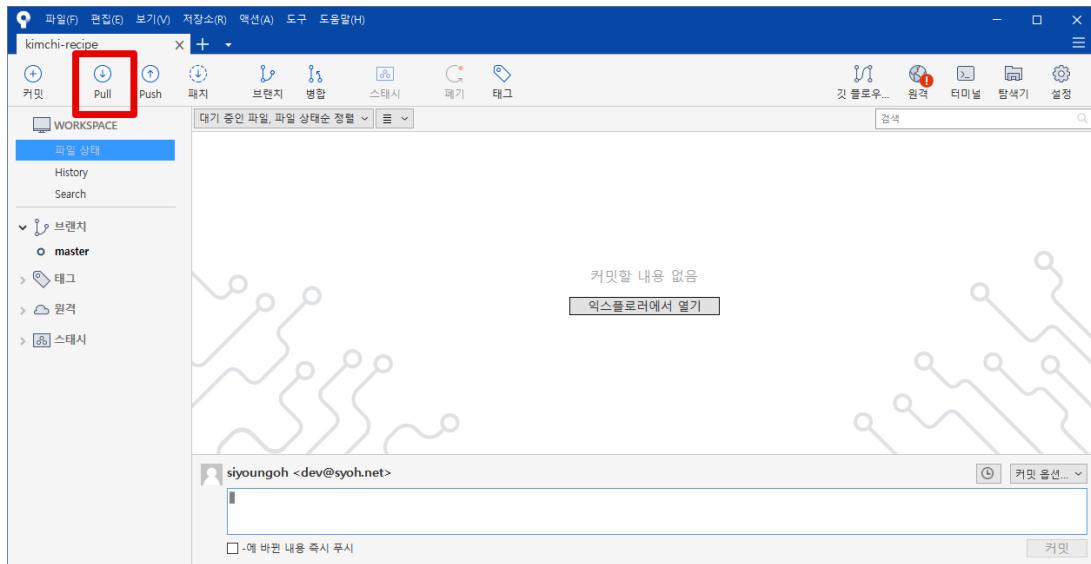
The screenshot shows a GitHub repository page for 'siyoungoh / kimchi-recipe'. The 'Code' tab is selected. At the top, there are buttons for 'Issues', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', and 'Insights'. Below these are buttons for 'Unwatch', 'Star', 'Fork', and a '...'. The main content area shows a file named 'kimchi-recipe / jjigae.txt'. A red box highlights the commit message '김치찌개 요리법 업데이트' in the commit history. The commit history also shows 'Latest commit f72a5e0 now' and a 'History' link. The commit details show 5 lines (5 sloc) and 203 Bytes. The commit message itself is listed as follows:

```
1 김치찌개 만드는 방법
2 1. 배추 김치를 먹을만한 크기로 썬다.
3 2. 냄비에 물을 끓인다.
4 3. 끓인 물에 배추 김치를 넣고 간을 한다.
5 4. 배추 김치가 익으면 끌
```

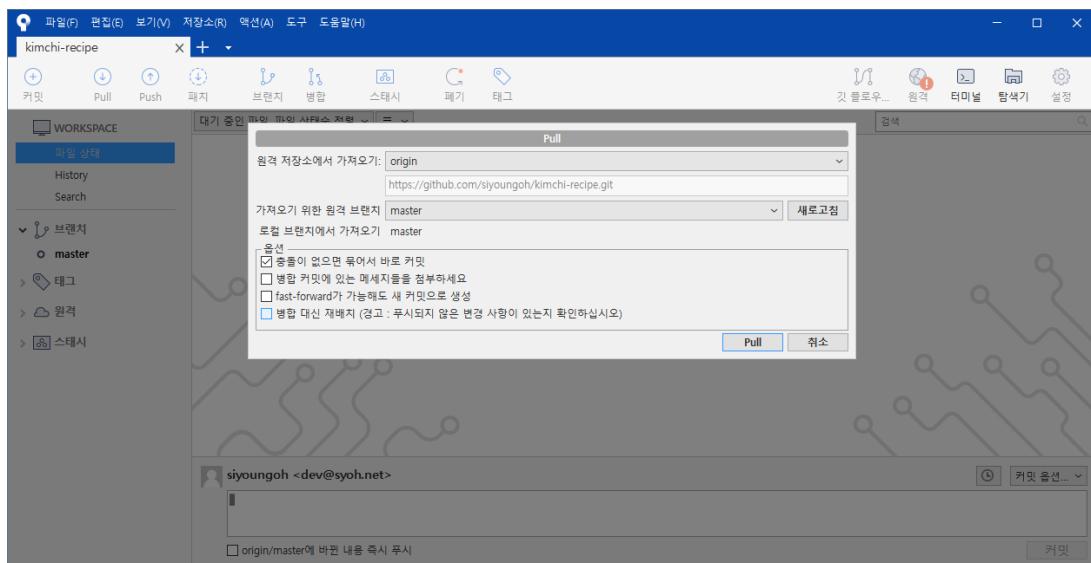
▼ 2. sourcetree 로 pull 하기

▼ windows

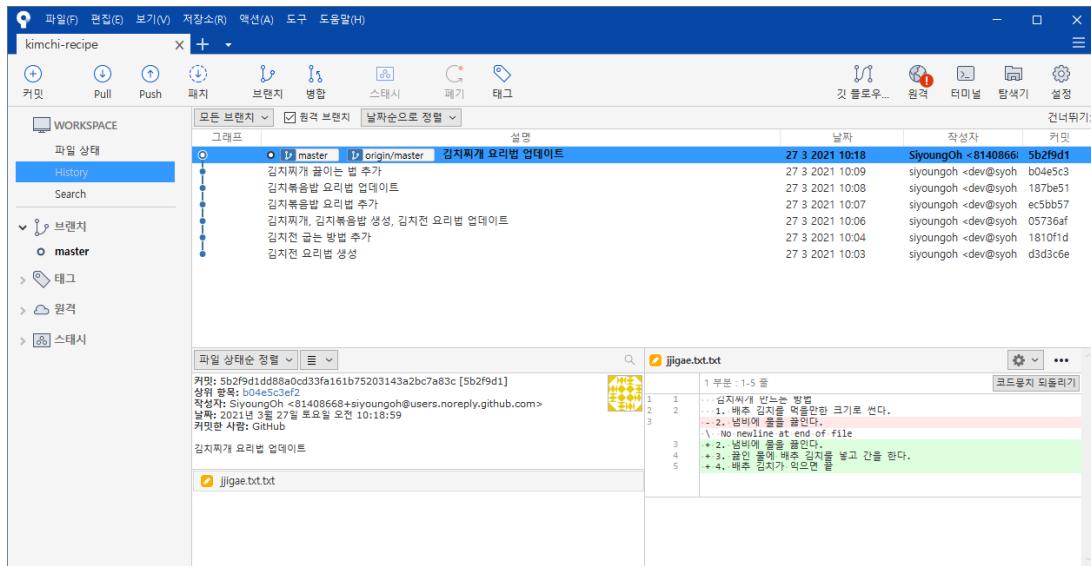
- pull 클릭



- 여러 옵션이 있지만 아래처럼 기본적인 옵션으로 둔 다음 pull 버튼 클릭



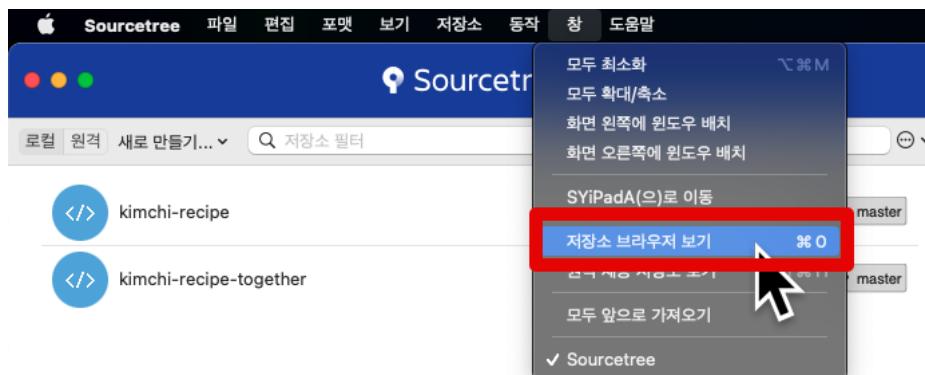
- 아래처럼 pull 해온 commit 내역도 history 에 보입니다.



▼ mac

▼ 참고. 만약 아래와 같은 창이 바로 보이지 않는다면

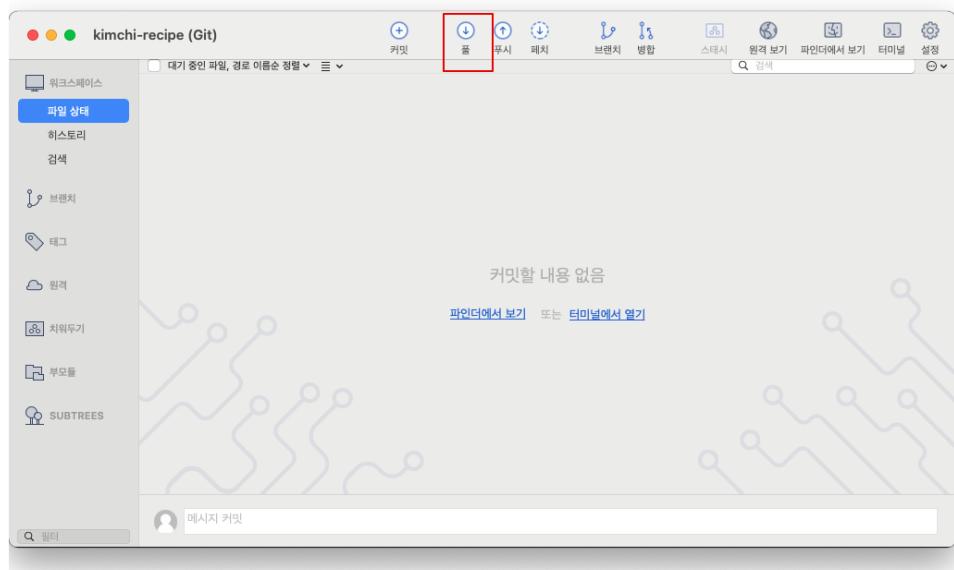
1. 상단 메뉴바 - 창 - 저장소 브라우저 보기 를 클릭하면 아래 창이 뜹니다.



2. 여기서 로컬 탭 누르고, kimchi-recipe(내가 만든 repo 이름) 더블 클릭!



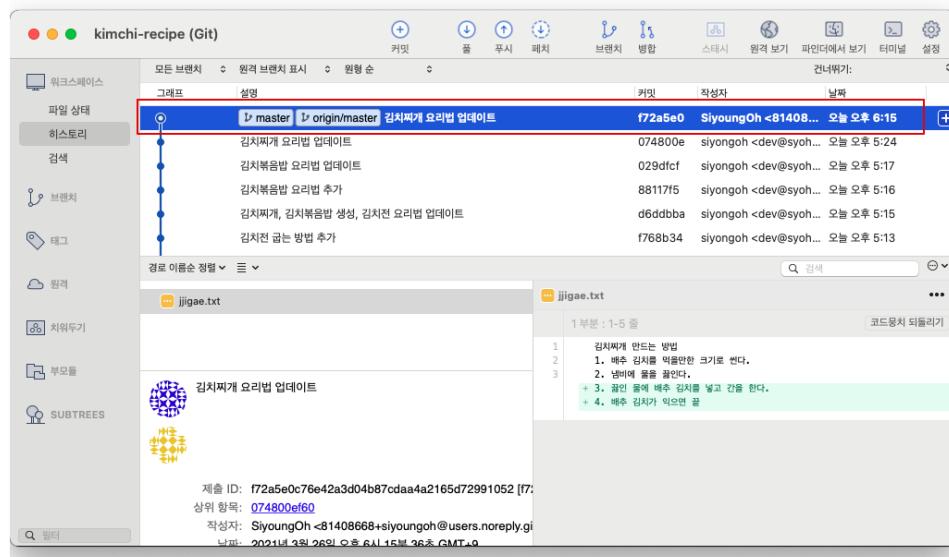
- 풀 버튼을 눌러주세요.



- 여러가지 옵션이 있지만 기본 옵션을 사용하겠습니다. 아래처럼 옵션을 선택하고 확인 버튼을 눌러주세요.



- commit 을 잘 pull 했는지 확인하기 위해 히스토리 버튼을 누르면 와! 원격에 있는 commit 이 생겼네요!



▼ 35) 원격 repo Github 에서 없애는 방법 - 에러나면 참고!



맘껏 실험해보기 위해 설정을 처음으로 깔끔하게 되돌리는 방법을 알려드릴게요!
뭔가 잘 동작 안 된다면 없애고 새 마음으로 시작하는 것도 방법이에요!

단, 바로 지우기 전에 어떤 것이 잘 동작되지 않는지 에러노트! 꼭! 써보세요!

1. Github 에 로그인 한 후 repo 페이지로 가서 Setting 을 누릅니다.

2. options 를 선택한 후 맨 마지막으로 스크롤을 내립니다.

3. Danger Zone 에 있는 [Delete this repository](#) (이 repository 지우기) 클릭!

Danger Zone

Change repository visibility
This repository is currently public. [Change visibility](#)

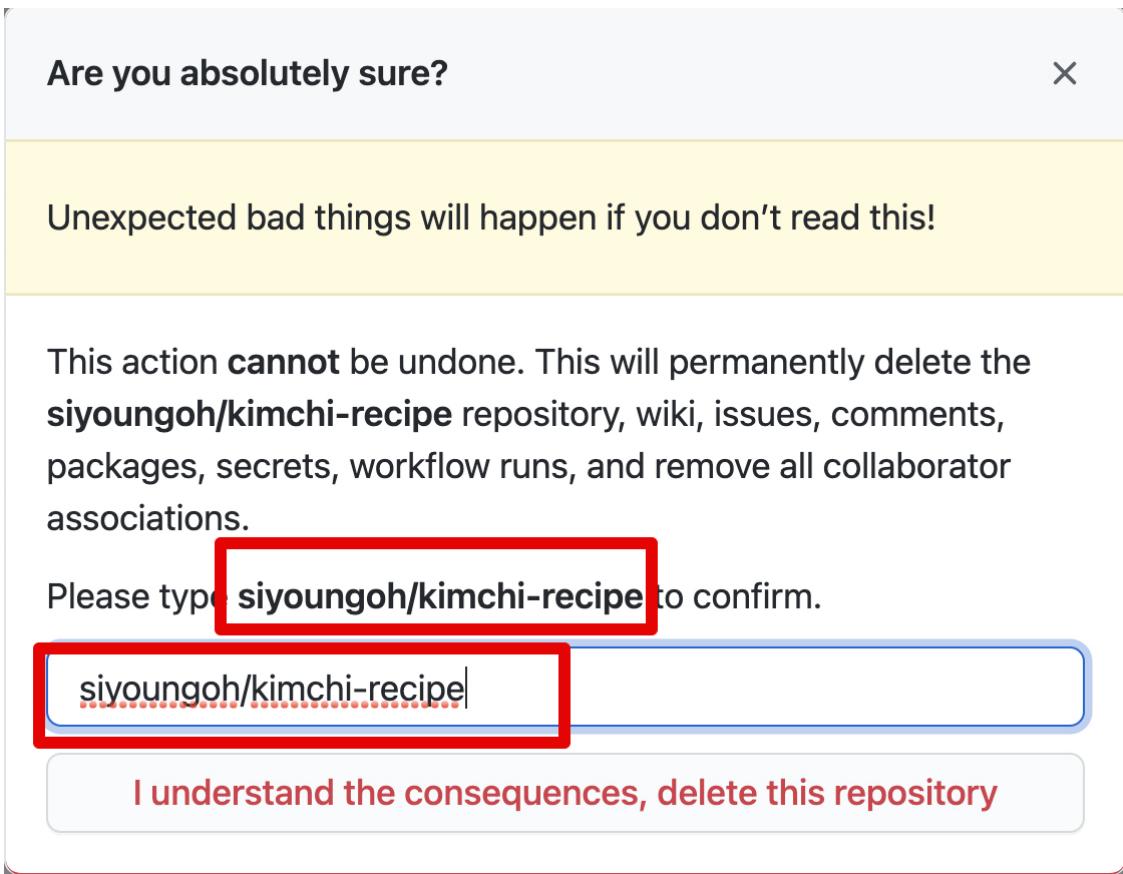
Transfer ownership
Transfer this repository to another user or to an organization where you have the ability to create repositories. [Transfer](#)

Archive this repository
Mark this repository as archived and read-only. [Archive this repository](#)

Delete this repository
Once you delete a repository, there is no going back. Please be certain. [Delete this repository](#)

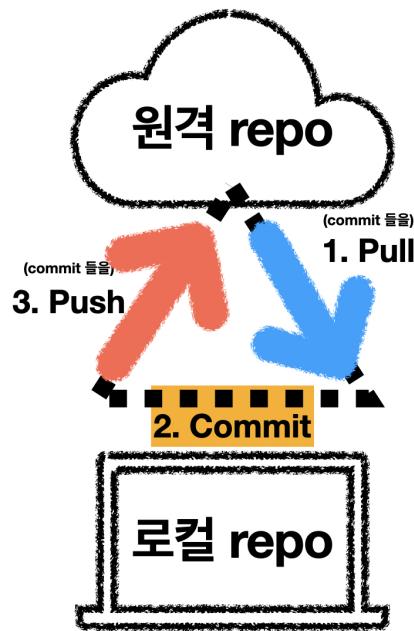
4. 정말로 지울 것인지 확인하는 창이 뜹니다. 절대 복원할 수 없으니 꼭 주의하세요!

- 하단 빈 칸에 `user이름/repository명` 형식으로 그대로 적어주세요.
그런 다음 I understand the consequences, ~ 버튼을 누르면 repo 가 삭제됩니다.



▼ 36) 초심자를 위한 꿀 패턴! pull - commit - push

- Git과 초면인 여러분들을 위한 팁!
혼자 Git 프로젝트 작업을 할 때는 `pull` -> `로컬 repo commit` -> `push` 순서로 하면 좋아요.
- 원격 repo 와 로컬 repo 에서 같은 파일을 수정하면 Git 이 같은 파일을 수정했는데 내가 어떤 파일을 최종으로 할까? 라고 확인 메시지를 줍니다. (정확한 내용은 곧 배울꺼에요!) 이런 것을 바로 **충돌(conflict)**이라고 표현해요.
- 충돌을 피하기 위해서는 아래 순서를 따라주는 게 편해요. 같은 파일을 동시에 수정해버리면 충돌이 나니까 두 repo 의 상태를 똑같이 맞춰준 후에 변경작업을 해주는 거예요.
 - 원격 repo 와 로컬 repo 의 상태를 똑같이 맞춰주기, 즉 로컬 repo 에 원격 repo 작업내역 가져오기 (`pull`)
 - 로컬 repo 의 작업 내용을 저장하고 (`commit`)
 - 원격 repo 에 로컬 repo 내용을 반영 (`push`)
- 원격 repo 에 변경사항이 생겼다! 하면 먼저 `pull` 하고 로컬 repo 에서 작업하면 많은 경우 충돌을 피할 수 있어요.

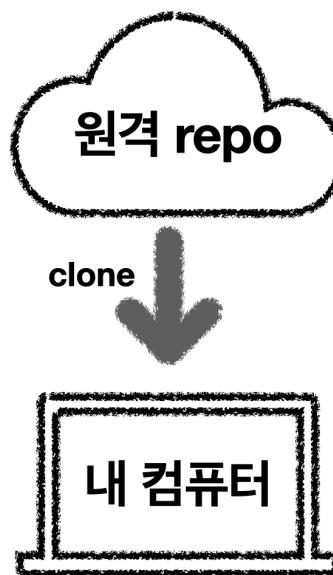


언제나 100% 통하는 패턴은 아니에요!

경우에 따라 다른 패턴을 사용하기도 하니 대부분의 경우에 통하는 초심자를 위한 팁! 정도로 기억해두세요. 😊

▼ 37) clone - 원격 repo 를 내 컴퓨터에 가져오기

- 원격 repo를 내 컴퓨터에 가져올 수는 없을까요?
 - A 컴퓨터에서 작업한 걸 github에 올리고 B 컴퓨터에서 내용을 보고 싶을 때
 - 다른 사람의 repo를 나도 다운로드 받아서 보고 싶을 때가 있겠죠!
 - 이때 사용하는 것이 clone입니다. 물론! 복제하다라는 뜻이죠. repo를 내 컴퓨터에 복제해오는 겁니다.
- url을 통해서 원격 repo에 접근할 수 있습니다. repo url을 사용해 repo를 clone 해올게요!



1. 먼저 내 컴퓨터 바탕화면에 `kimchi-recipe-together` 이름의 폴더를 만들어주세요. 원격 repo 내용을 저장할 폴더입니다.
2. 복제할 repo의 url 복사해오기

- 아래 url에 접속해서 repo url을 복사해오겠습니다.

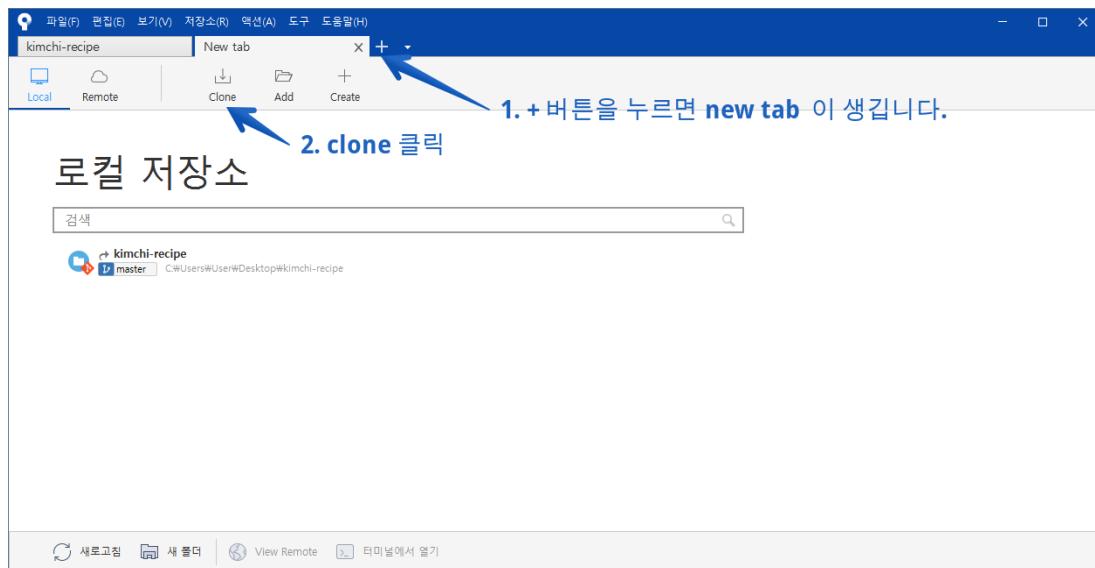
▼ [코드스니펫] clone 할 repo 링크

<https://github.com/siyoungoh/kimchi-recipe-together>

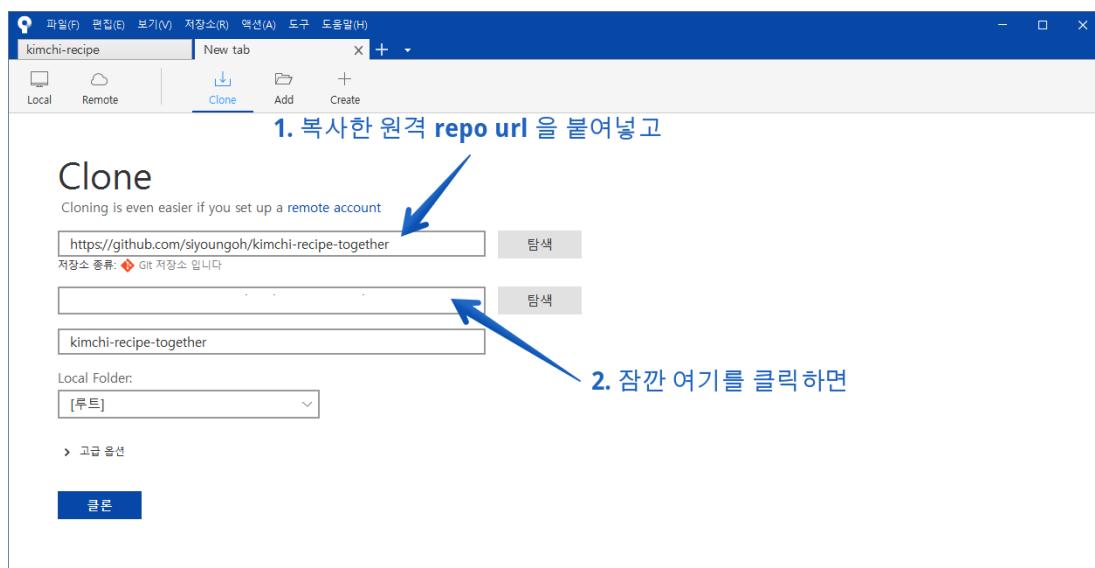
3. sourcetree로 사용해서 clone하기

▼ windows

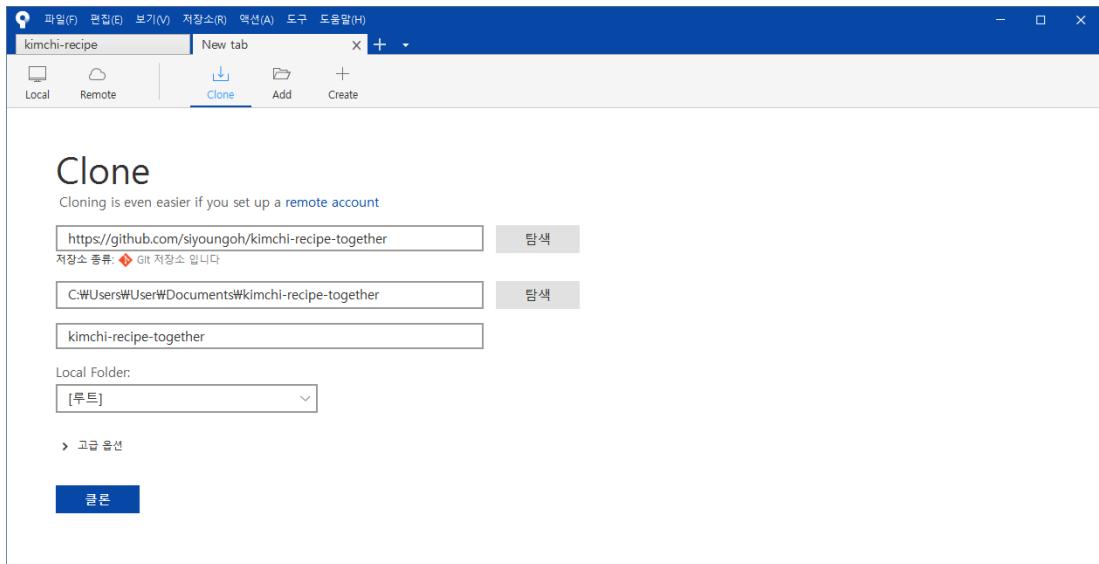
- + 버튼을 누르면 new tab이 생깁니다. 여기서 clone을 눌러주세요.



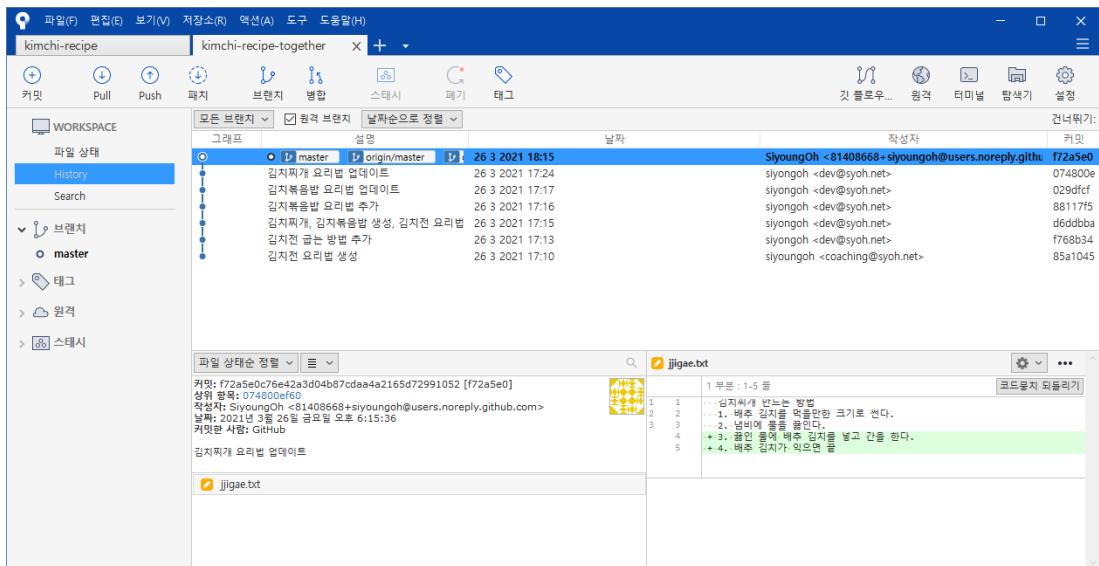
- 첫번째 칸에 원격 repo url을 붙여넣고 두번째 칸을 잠깐 클릭하면



- 두번째 칸도 자동으로 채워집니다. 기본 설정은 Documents 밑에 폴더가 생깁니다.



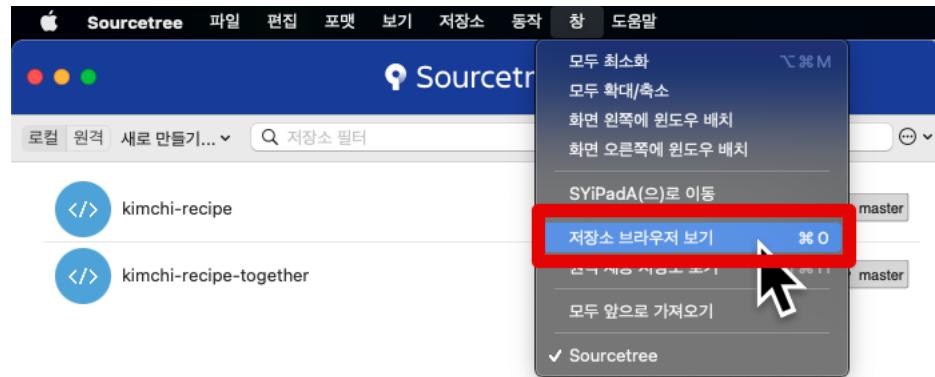
4. 아래처럼 보면 잘 clone 해온 것입니다.



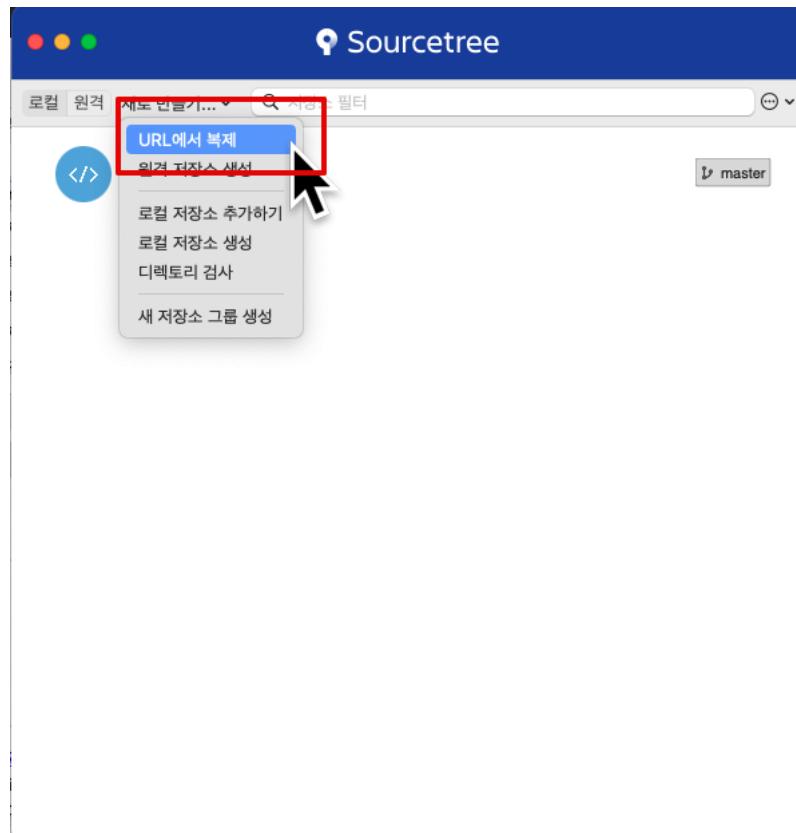
▼ mac

▼ 참고. 만약 아래와 같은 창이 바로 보이지 않는다면

- 상단 메뉴바 - 창 - 저장소 브라우저 보기 를 클릭하면 아래 창이 됩니다.



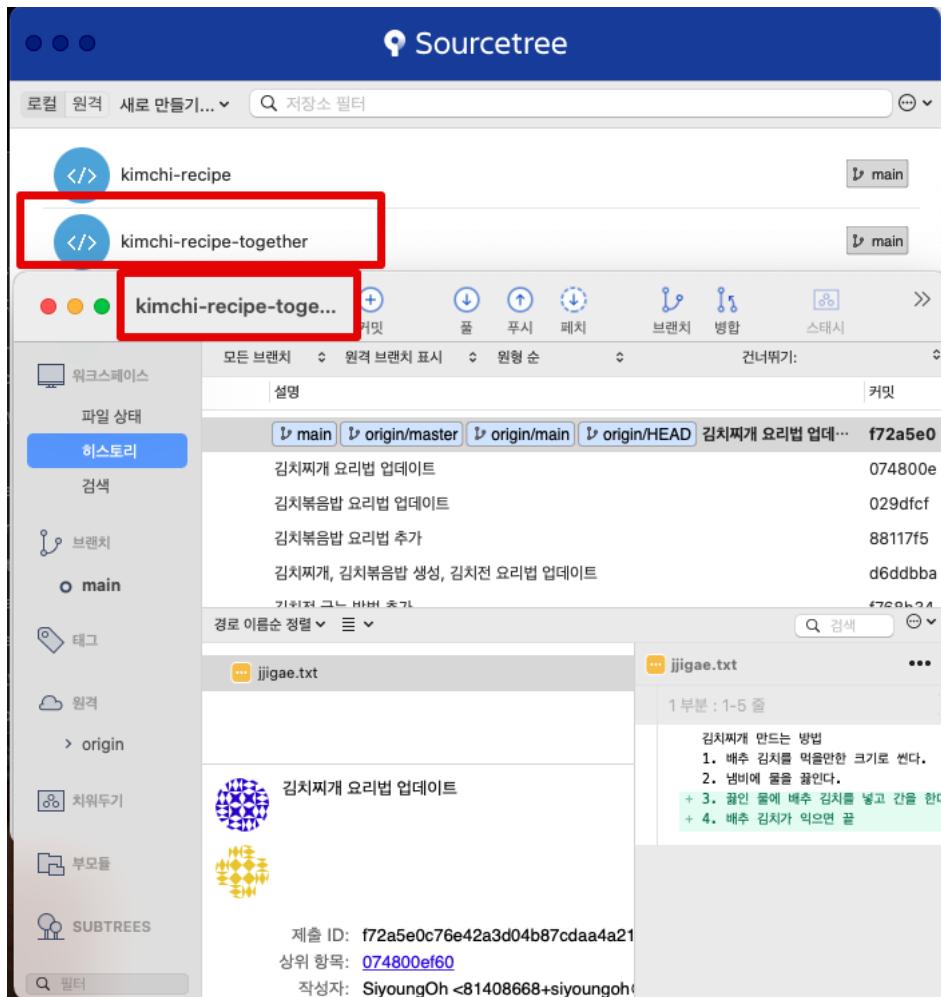
- 새로 만들기 - url에서 복제



- 아래처럼 입력한 후 클론 버튼 클릭



- 아래처럼 repo 를 잘 가져왔네요! 공개 repo url 을 사용해 이런 식으로 clone 해 올 수 있습니다.



공개 repo 만 clone 해 올 수 있을까요?

비공개 repo 라도 권한이 있다면(해당 repo의 주인인 경우 등) clone 해올 수 있습니다.

구글 드라이브, 네이버 클라우드처럼 비공개 파일은 주인이나 공유 권한이 있는 사람만 접근할 수 있는 것과 비슷합니다.

정리하자면, 공개 repo 인 경우는 누구나 clone 할 수 있고, 비공개 repo 인 경우 repo 에 접근할 수 있는 권한이 있는 경우에만 clone 할 수 있습니다.

08. 원격 repo 사용하기 - 정리

▼ 38) [🍺 배웠으면 써먹자] - 개념 적어보기



종이와 펜을 준비해서 직접 써보세요. 머리로 말고 직접 글로 적어 주세요.

그래야 내가 제대로 아는지 정확히 확인할 수 있어요.

수업자료를 확인하기 전에 스스로 먼저 적어보기 약속 👉

1. 원격 repo 와 로컬 repo 를 연결해서 내용을 반영하고 싶을 땐 어떤 방법을 써야할까?
 2. 원격 repo 와 로컬 repo 는 왜 따로 있을까?
 3. push 와 pull 의 개념을 원격 repo와 로컬 repo 를 포함해 그림으로 그려보세요.
- 위 내용을 스스로 적었으면 36)정리하기 를 확인하며 다시 내용을 정확히 확인해보세요.

▼ 39) [💡 배웠으면 써먹자]- push 하기, pull 스스로 해보기



다시 한 번 확인!

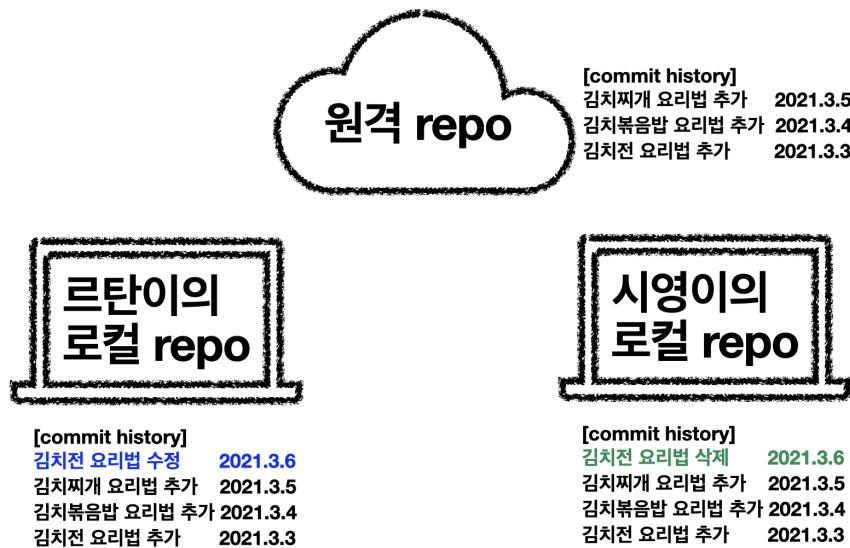
만약 pull 과 push 가 잘 안 된다면, 원격 repo 와 로컬 repo 에서 같은 파일을 수정해서 충돌(conflict)이 발생한 것일 수 있어요!

충돌(conflict) 은 나쁜 게 아니라 원격 repo와 로컬 repo 의 파일 변경사항이 겹친다는 것을 알려주는 것입니다. 충돌을 수정하는 방법은 앞으로 배울 꺼니까 걱정하지 마세요! 😊

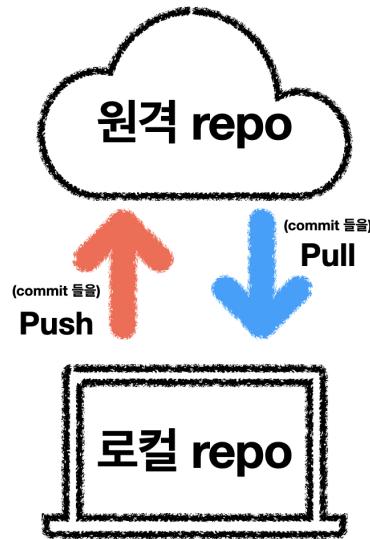
▼ 40) 정리하기

▼ 34) 개념 적어보기 질문 정답 예시 확인

1. 원격 repo 와 로컬 repo 를 연결해서 내용을 반영하고 싶을 땐 어떤 방법을 써야할까?
 - 로컬 repo 가 없고 원격 repo 에 있는 내용을 가져오고 싶을 때는 clone 한다.
 - 원격 repo 와 로컬 repo 둘 다 있고 두 개를 연결하고 싶을 때는 로컬 repo 가 원격 repo 를 tracking 하도록 설정한다.
2. 원격 repo 와 로컬 repo 는 왜 따로 있을까?
 - 협업할 때, 혹은 여러 컴퓨터를 사용한다면 하나의 원격 repo 에 여러 로컬 repo 를 연결시킬 수 있다. 하나의 프로젝트를 동시에 작업하는게 가능해진다.



3. push 와 pull 의 개념을 원격 repo와 로컬 repo 를 포함해 그림으로 그려보세요.



▼ 41) [💡 배웠으면 써먹자] - 이 가이드에서 배운 내용 그림으로 그려보기

- 종이와 연필을 꺼내서 내가 이해한만큼 Git의 동작을 아래 키워드를 포함해 그림으로 그려보세요.
- 처음에는 수업자료를 보지 않고 내 머리속에 있는 내용만으로 그림을 그려보세요.
두번째는 수업자료의 각 25) 정리하기, 36) 정리하기 부분을 보고 나서 다시 그려보세요!
세번째는 아래 내용을 보고 내 개념 지도에서 빠진 부분을 포함해서 다시 그려보세요!

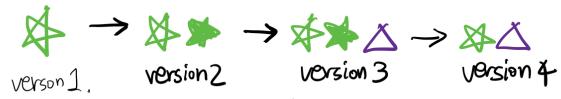
▼ 키워드

- 버전 관리
- git 초기화 (initialize, init)
- add / staging
- commit (커밋)
- commit 내역(history)
- push
- pull
- clone
- tracking

▼ 내가 그리는 개념지도 예시

버전 관리

↳ 언제 무엇이 어떻게 바뀌었는지 상태를 정리하는 것



↳ 각 version이 상태에 정보를 가지고 있으면 된다.

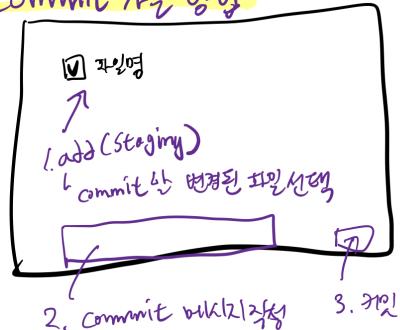
Commit

↳ 뭔가 바뀌었는지 project의 상태를 저장 Camera 촬영 추가!

Git repo로 만들기



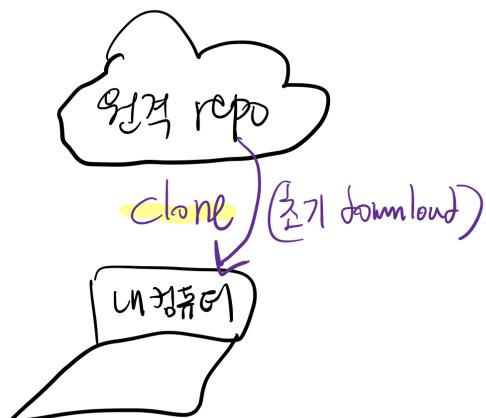
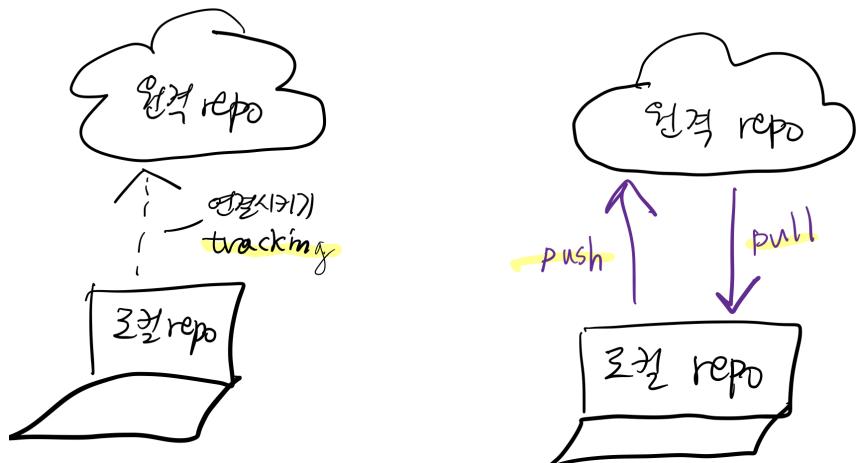
Commit 하는 방법



Commit history

Commit을 내역





👉 꼭! 머리 속에 있는 내용을 먼저 그려보고 나서 수업자료를 보면서 정리하세요!

내가 모르는 것이 무엇인지 알아야 모르는 부분에 집중해서 효율적으로 학습할 수 있어요.
지금은 모르는게 당연합니다. 몰라도 괜찮아요! 아까 배웠는데 바로 기억하면 완전 천재죠!

▼ 42) 총 정리 개념

- **버전 관리:** 프로젝트 상태가 변경되는 정보를 알고 있다는 것입니다. Git은 가장 널리 쓰이는 버전관리 도구 중에 하나로 commit을 사용해서 버전이 달라지는 것을 관리합니다.
- **git 초기화(git initialize):** 컴퓨터에 있는 프로젝트를 Git이 관리하는 프로젝트로 만들 수 있습니다. 앞으로 Git으로 관리할꺼야! 하고 설정해주면 됩니다. 이 작업을 한다고 표현합니다.
- **commit :** 현재 프로젝트의 상태를 찰칵 📸 저장하는 것을 이라고 합니다.
 - 누가(author), 언제 commit 했는지의 정보와 프로젝트 변경 내용
 - 작업내역이 어떤 것인지 알아볼 수 있게 적는 메시지를 'commit 메시지'라고 합니다.

▼ [코드스니펫] commit 내용 페이지 링크

<https://github.com/ohahohah/constitution-of-republic-of-korea/commit/4fce742d1e851840c600ff316adfa335f986841>

대한민국 헌법 - 1960.6.15 일부개정/ 4.19 혁명
main 누가 언제
ohahohah committed 3 days ago

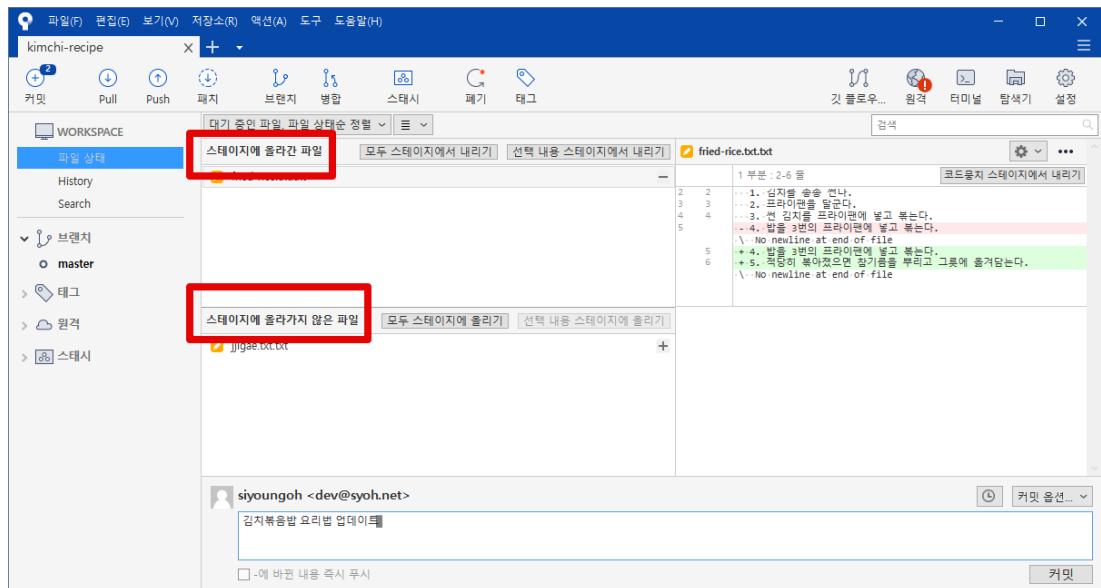
Showing 2 changed files with 191 additions and 131 deletions.

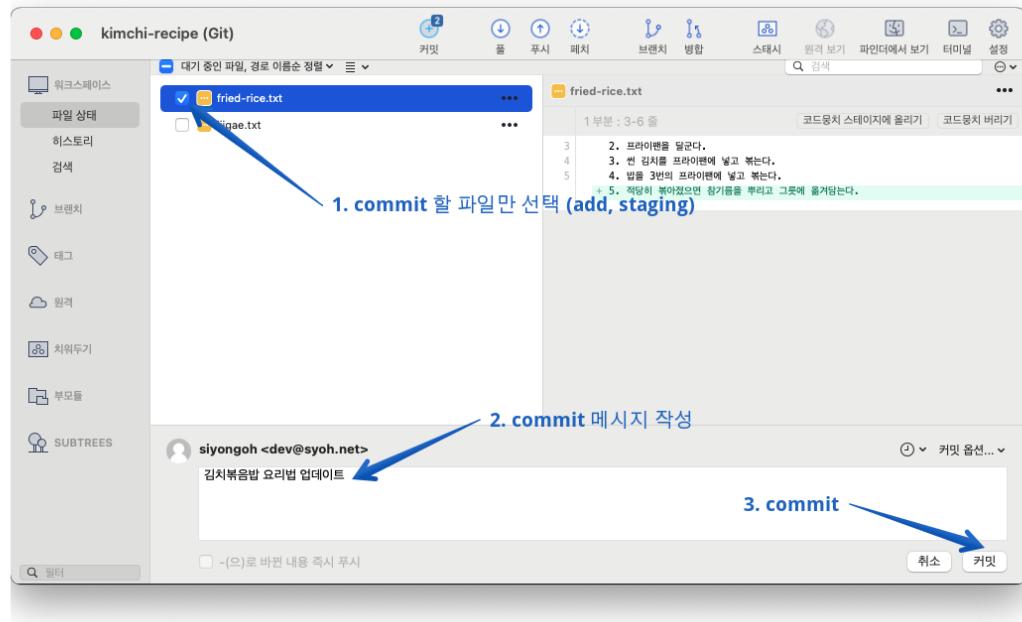
宪法修正案.txt
宪法.txt

@@ -29,11 +29,13 @@

제9조 모든 국민은 신체의 자유를 가진다. 법률에 의하지 아니하고는 체포, 구금, 수색, 심문, 처벌과 강제노역을 받지 아니한다.
제10조 모든 국민은 법률에 의하지 아니하고는 거주와 이전의 자유를 제한받지 아니하며 주거의 침입 또는 수색을 받지 아니한다.
제11조 모든 국민은 법률에 의하지 아니하고는 통신의 비밀을 침해받지 아니한다.
제12조 모든 국민은 신앙과 양심의 자유를 가진다.
국교는 존재하지 아니하며 종교는 정치로부터 분리된다.

- add (혹은 staging, 스테이징) : commit에 반영할지 안할지는 파일 단위로 선택할 수 있습니다. commit에 반영할 파일을 선택하는 것



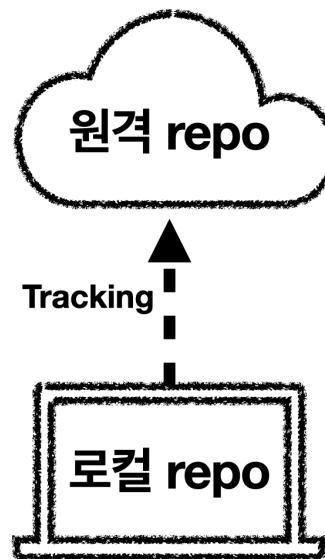


- commit history : commit 한 순서대로 리스트. 역사!

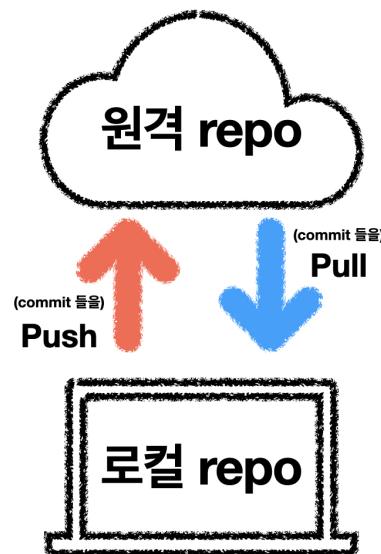
Commits on Mar 23, 2021

- 대한민국 헌법 제10호 - 1987.10.29 전부개정/ 6월 민주화 항쟁 (ed51228)
- 대한민국 헌법 제9호 - 1980.10.27 전부개정/ 10.26 사건, 12.12 군사반란 (34a6869)
- 대한민국 헌법 제8호 - 1972.12.27 전부개정/ 10월 유신 (f4b4ec3)
- 대한민국 헌법 제7호 - 1969.10.21 일부개정 (fec8b70)
- 대한민국 헌법 제6호 - 1962.12.26 전부개정 / 5.16 군사경변 (a2f9dc0)
- 대한민국 헌법 제 5호 - 1960.11.29 일부개정 (42c1b08)
- 대한민국 헌법 - 1960.6.15 일부개정/ 4.19 혁명 (4fce74)

- 지금까지 우리가 한 작업은 'git 초기화하기(initialize) - add(staging) - commit' 입니다.
 - git 초기화는 처음에 단 한번만 해 주면 됩니다. 작업 내역을 저장하기 위해서는 **add - commit** 만 하면 됩니다.
- repo** : 'Git으로 관리되는 프로젝트'를 Git에서는 **repo(리포, repository 리포지토리의 약자)**라고 부릅니다. 내 컴퓨터에 저장되어있는 리포지토리를 **로컬 repo(local repository)**라고 합니다. Github처럼 다른 곳에서 접속할 수 있는 공간에 저장되어있는 것을 **원격 repo(remote repository)**라고 합니다.
- Tracking(추적) : 로컬 repo 와 원격 repo 를 연결한다!



- push : 로컬 repo 의 commit 들을 원격 repo 에 반영하기(push)! 밀어넣기. 원격 repo 에 없는 즉, 새로운 commit 내역을 모두 원격 repo 에 한 번에 반영합니다.
- pull : 원격 repo 의 commit 들을 로컬 repo 로 반영하기(pull)! 땡겨오기. 로컬 repo 에 없는 즉, 새로운 commit 내역을 모두 로컬 repo 에 한 번에 반영합니다.



- clone : 원격 repo 를 내 컴퓨터에 가져와서 초기 repo 세팅하는 것을 clone(복제하기)!

