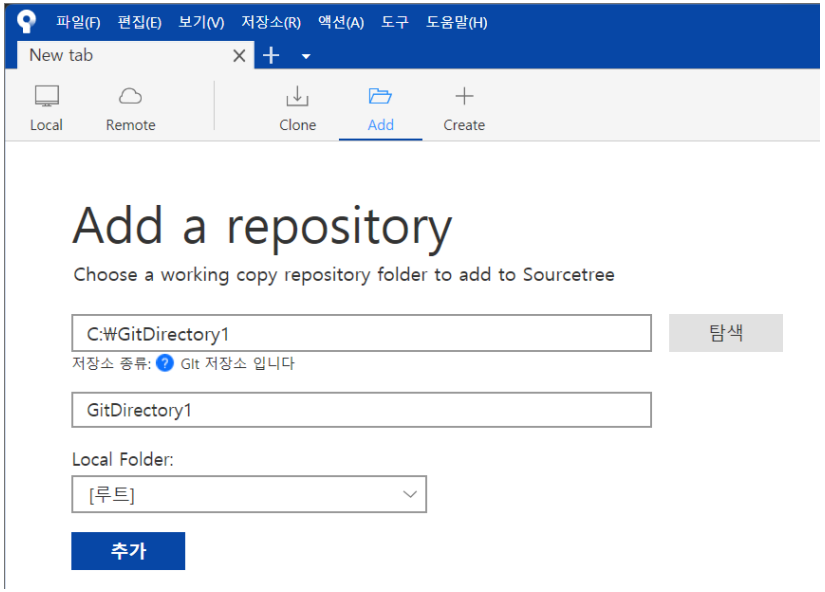
GUI 환경에서의 버전관리2

세종대학교 이은상

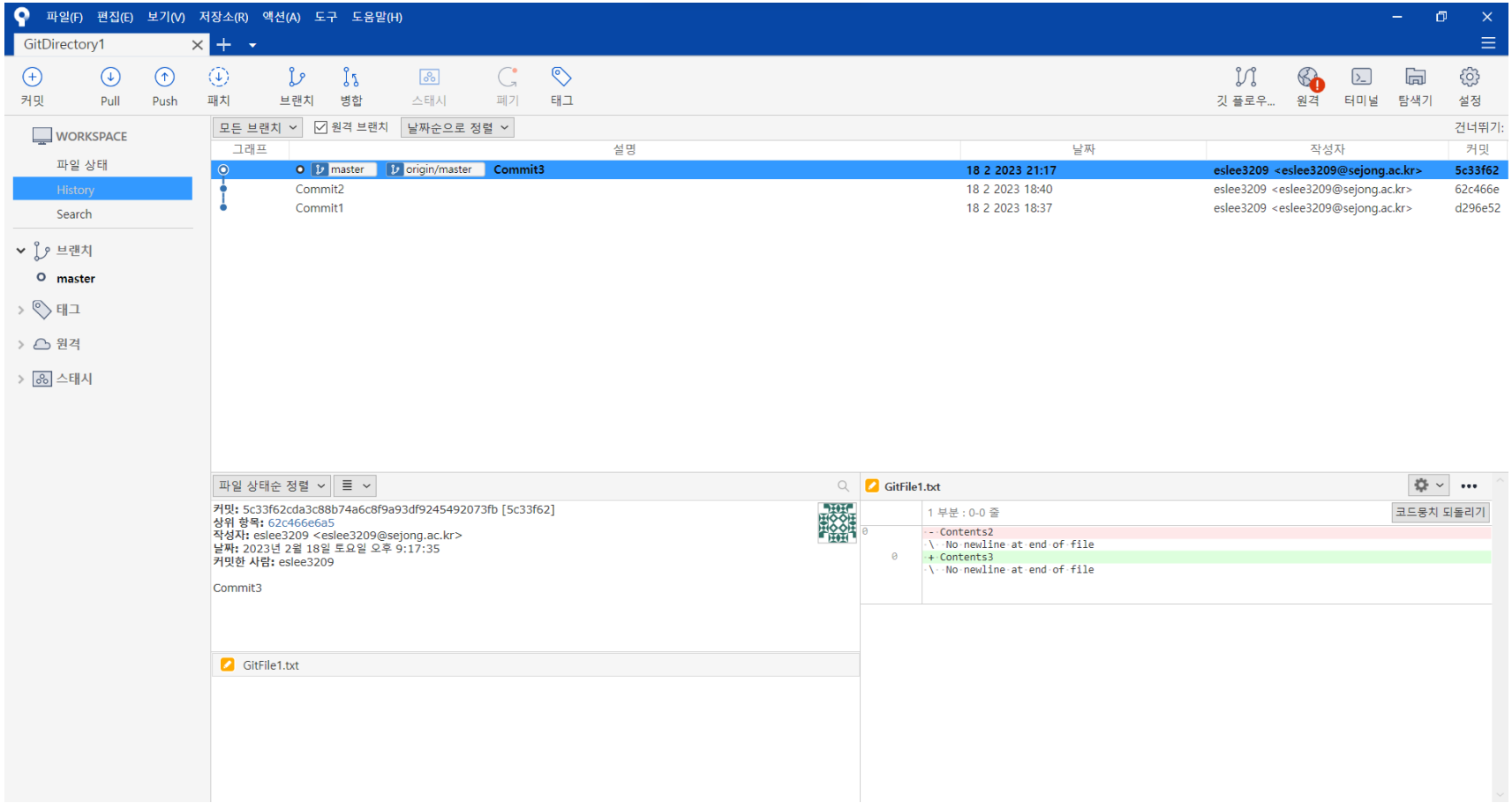
소스트리에 로컬저장소 불러오기

- 소스트리를 실행한 후 상단 탭에서 [Add] 버튼 클릭
- [탐색] 클릭 후 GitDirectory1 폴더를 찾아 [폴더 선택]을 클릭함
- [추가] 버튼을 눌러줌. 그러면 [GitDirectory1] 로컬저장소에서 버전관리를 할 수 있는 새로운 탭이 열림



소스트리에 로컬저장소 불러오기

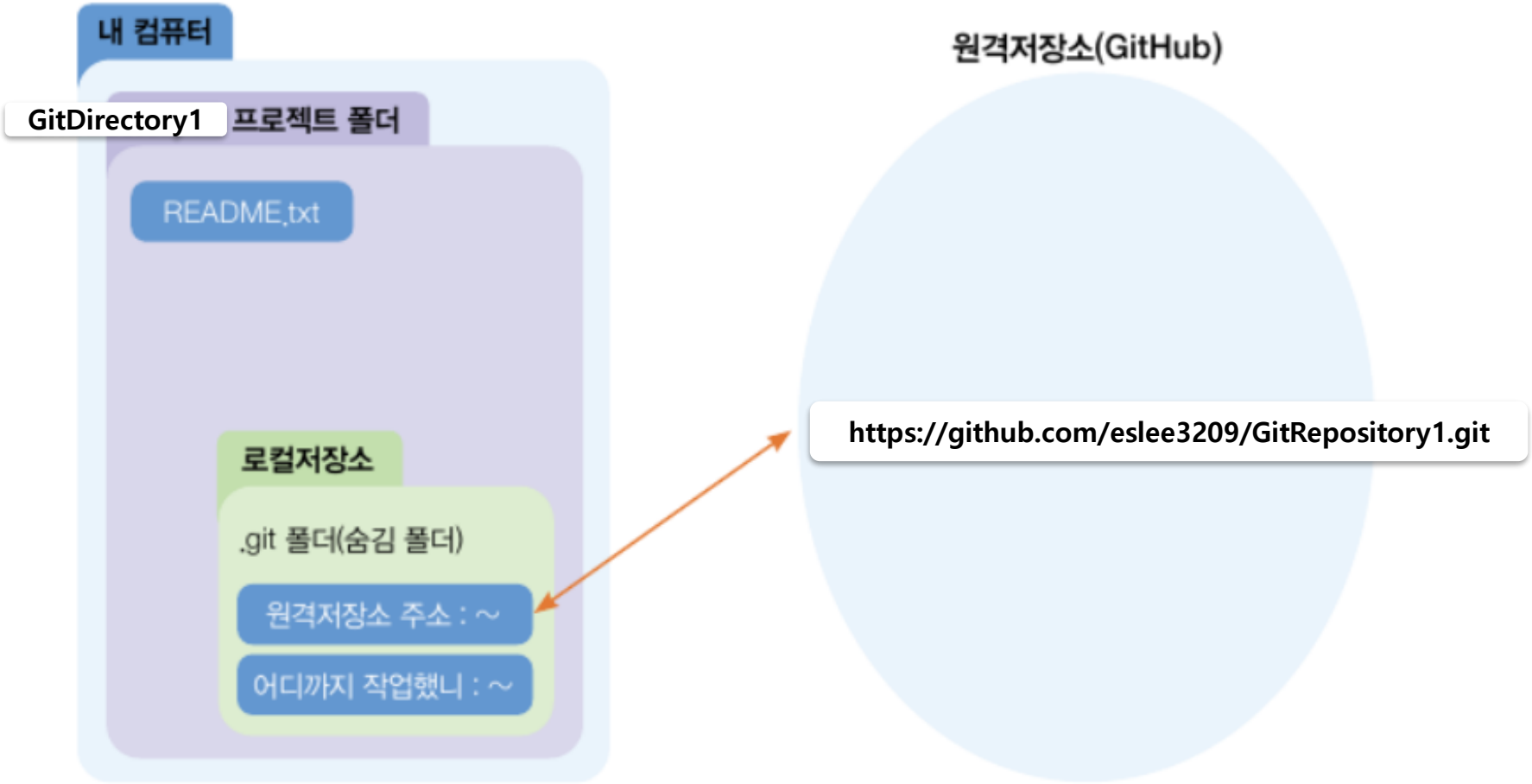
- 커밋한 이력이 그래프로 예쁘게 나옴. 이는 GitDirectory1에 있는 숨김폴더 .git에 저장된 정보 덕분



.git 폴더 개념

- Git으로 생성한 버전들의 정보와 원격저장소 주소 등이 들어있는 숨겨진 폴더임
- 이 .git 폴더를 로컬저장소라고 부름
- 이 폴더에다 버전 관리를 할 수 있음
- CLI에서는 git init 명령어를 통해 만들어지며, GUI 환경에서는 소스트리에서 [Create] 버튼 클릭하여 로컬저장소 생성하면 만들어짐

.git 폴더

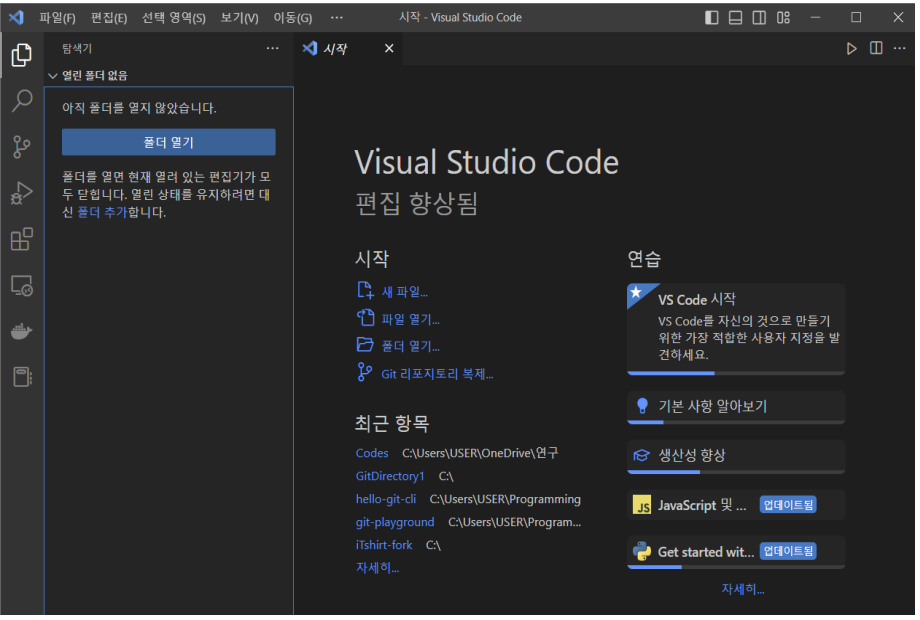

내 컴퓨터
GitDirectory1
프로젝트 폴더
README.txt
로컬저장소
.git 폴더(숨김 폴더)
원격저장소 주소 : ~
어디까지 작업했니 : ~
원격저장소(GitHub)
https://github.com/eslee3209/GitRepository1.git

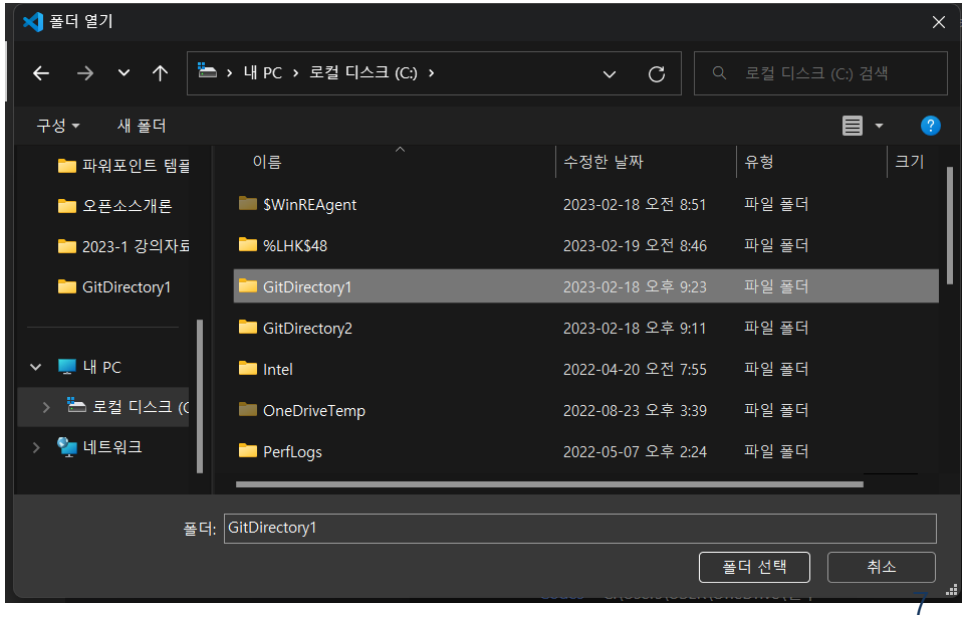
로컬저장소 삭제

- 로컬저장소를 삭제하는 방법은 단순히 git으로 관리하는 폴더를 .git 파일과 함께 삭제하는 것
- 가령 GitDirectory1폴더 전체를 삭제하면 그 안의 .git 파일도 삭제되면서 로컬저장소가 삭제됨
- 만약 CLI 환경이라면 다음 명령어로 가능
  - rm -rf [폴더명]
  - ex) rm -rf GitDirectory1

VSCode에서 폴더 열기

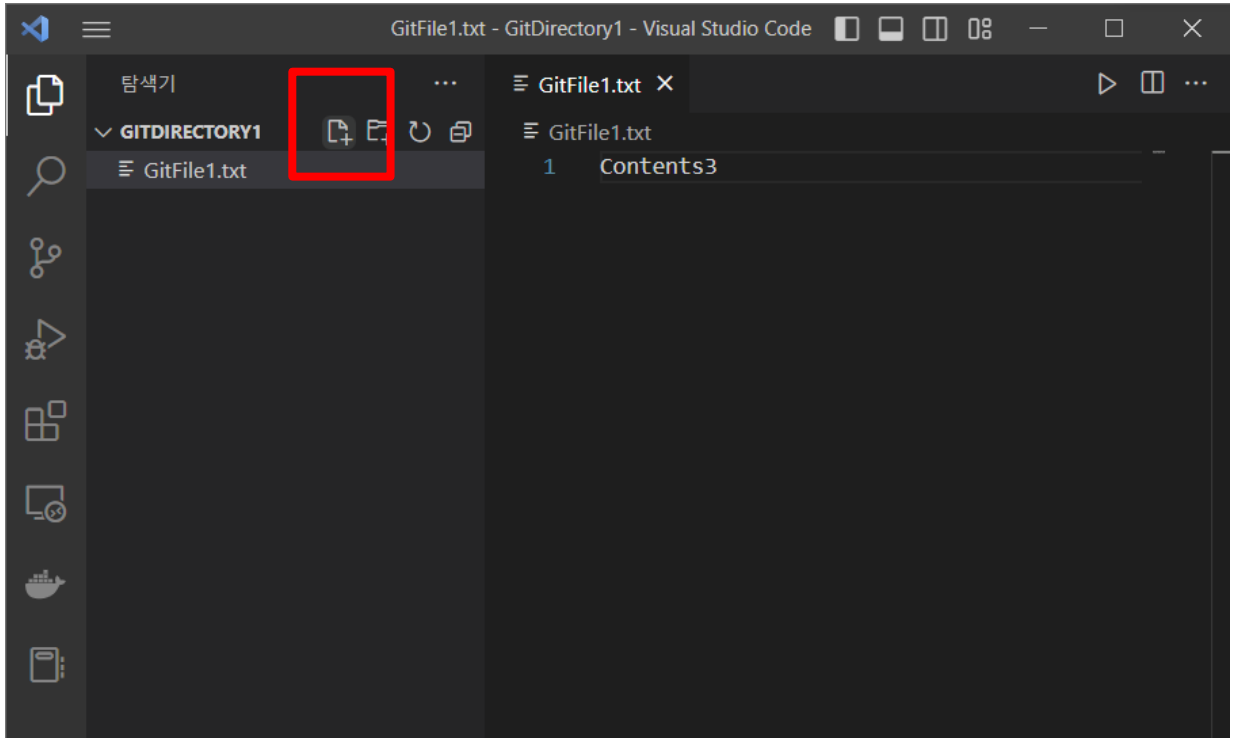
- VSCode에서 [GitDirectory1] 폴더에 가기 위해 왼쪽 탐색기 탭 클릭한 후 [폴더 열기]를 클릭함. [파일]>[폴더 열기]를 통해 폴더를 열 수도 있음
- [GitDirectory1] 폴더를 선택하고 [폴더 선택]을 클릭함. 그러면 GitDirectory1 폴더가 열림





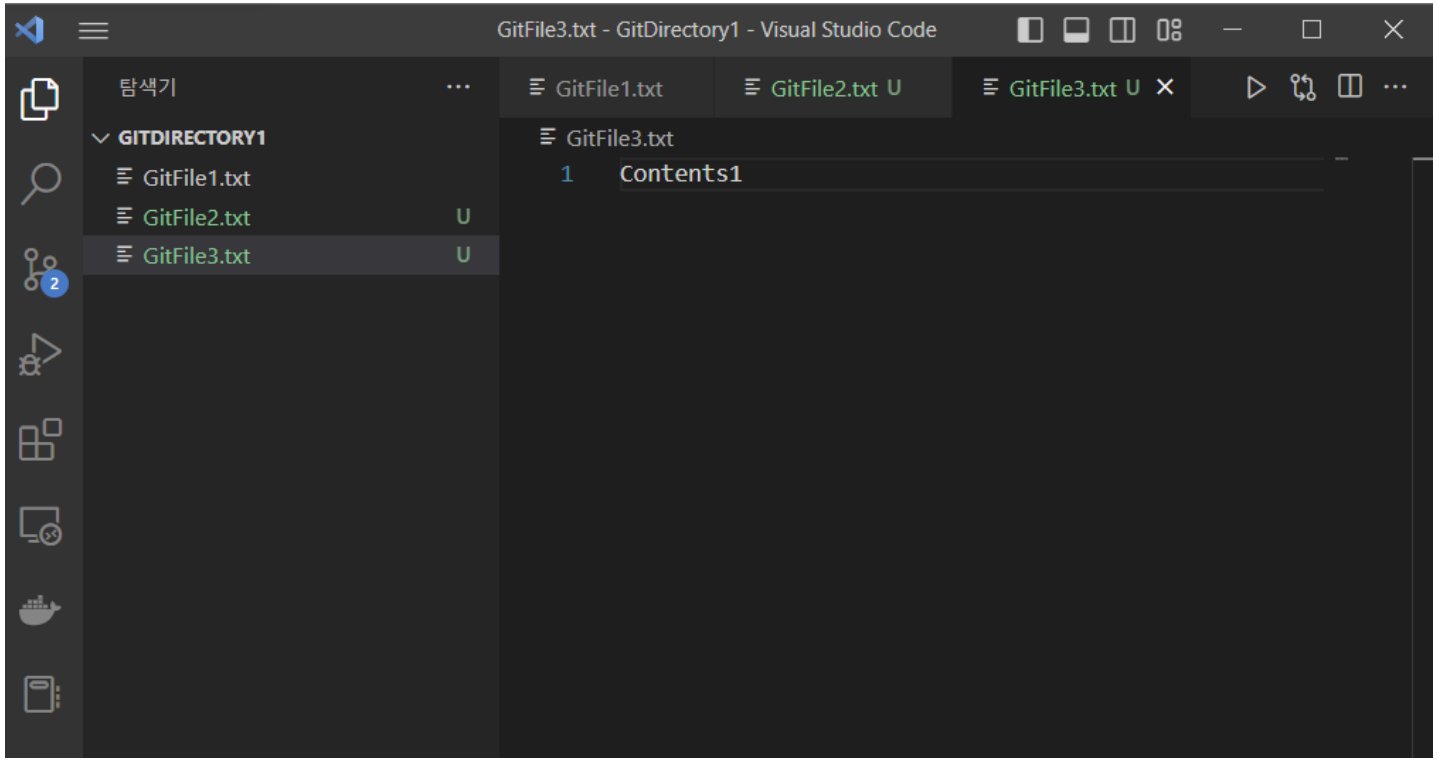
VSCode에서 파일 생성

- [탐색기] 탭에서 [새 파일] 아이콘을 누르면 새 파일을 만들 수 있음. 혹은 메뉴에서 [파일]>[새 파일] 로도 만들 수 있음



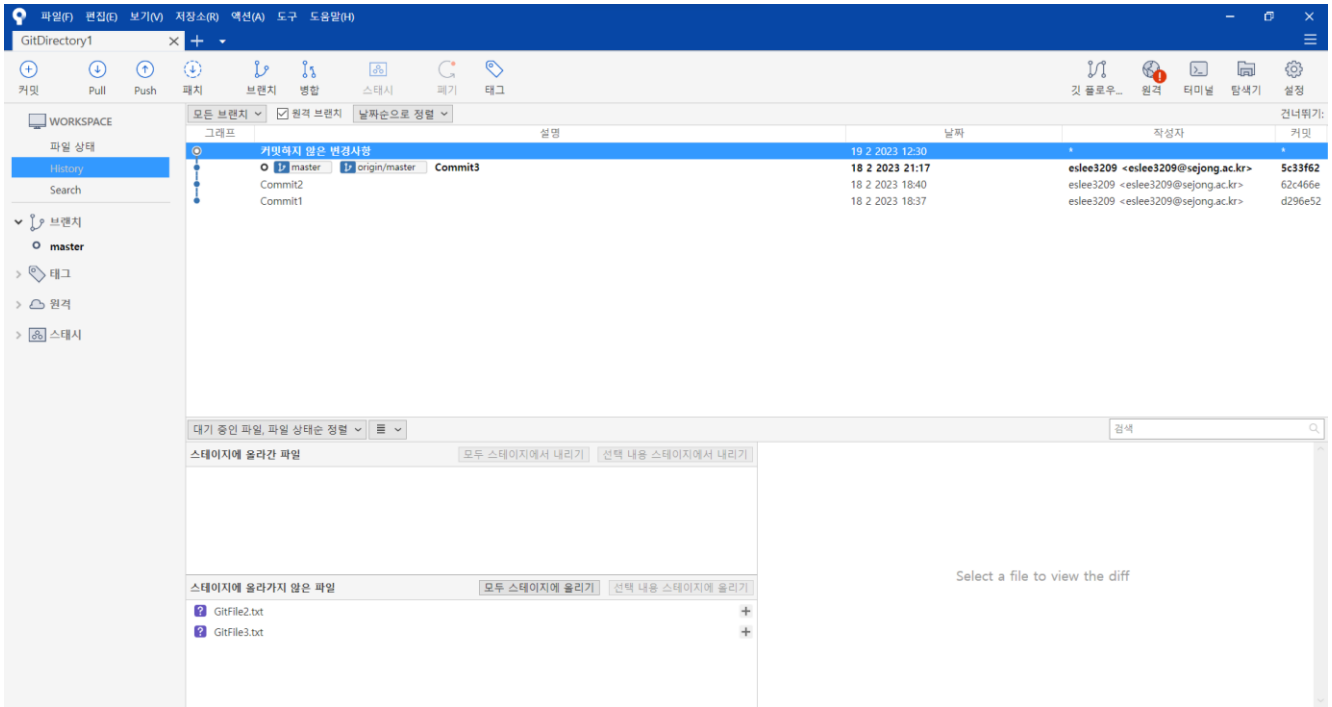
소스트리에서 커밋 실습

- VSCode에서 새로운 파일 2개를 생성함
  - 예) GitFile2.txt 파일을 생성함. 내용은 "Contents1"
  - 예) GitFile3.txt 파일을 생성함. 내용은 "Contents1"
  - 파일 생성은



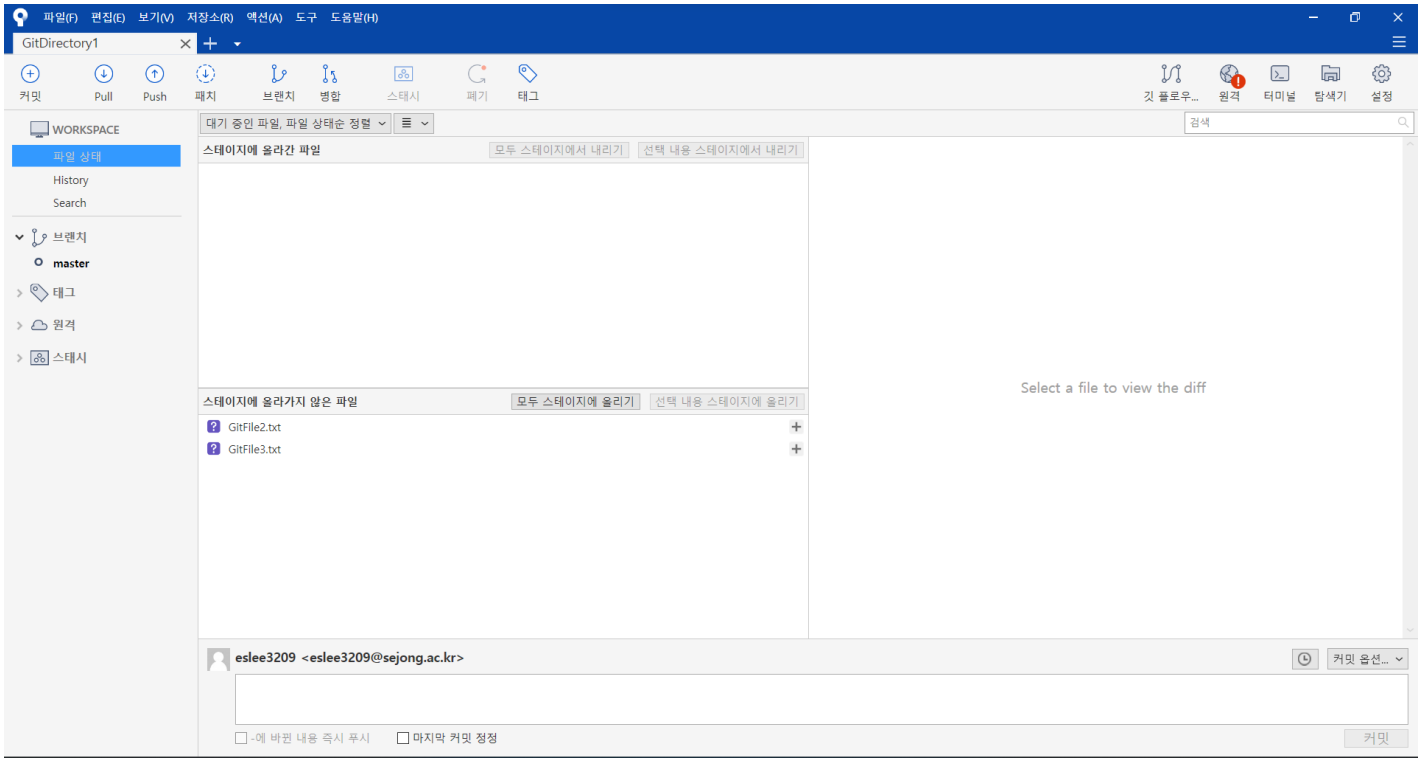
소스트리에서 커밋 실습

- 소스트리에 돌아가보면 아까와 다르게 그래프 최상단에 '커밋하지 않은 변경사항' 텍스트가 보임
- 이 텍스트를 선택하면 소스트리 하단의 [스테이지에 올라가지 않은 파일] 섹션에 방금 만든 파일이 보이는 것을 확인할 수 있음



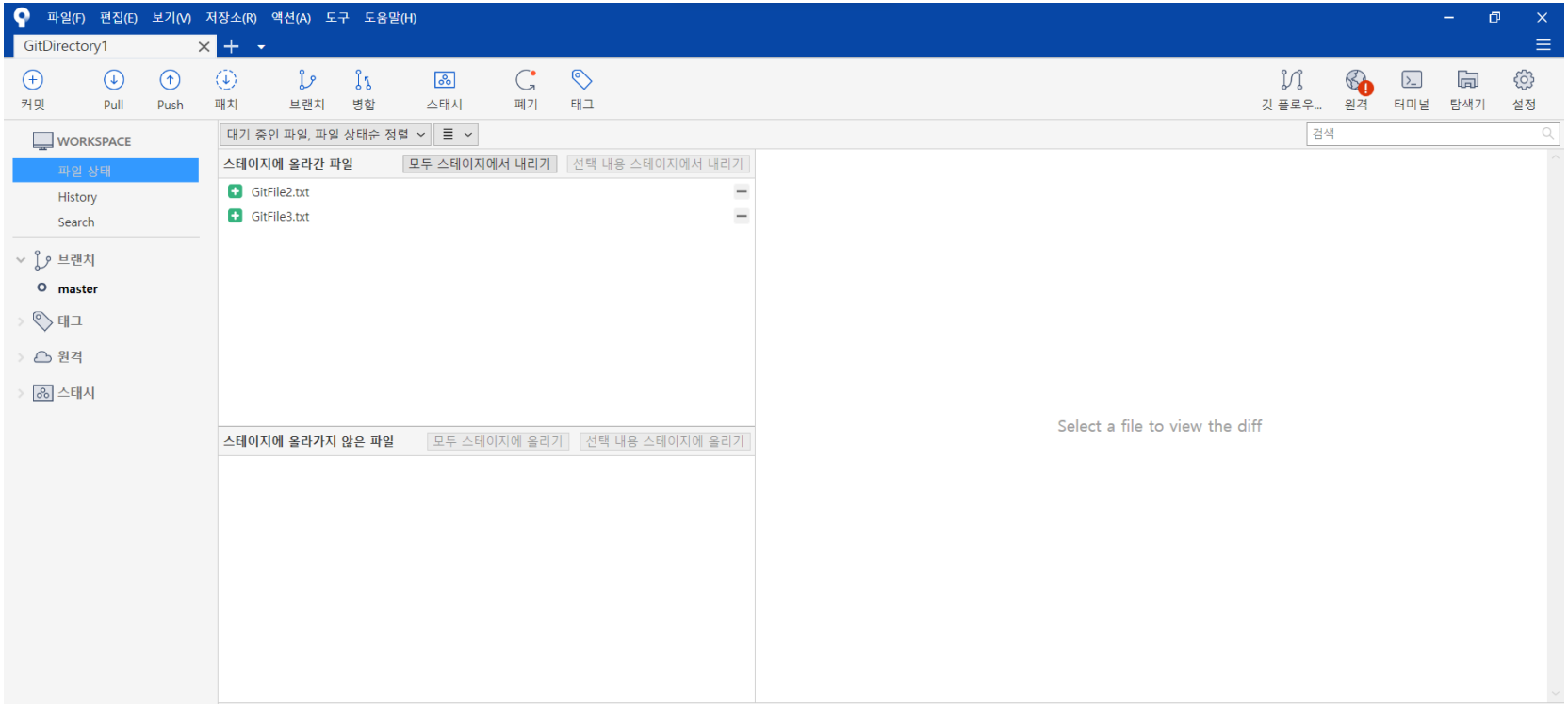
소스트리에서 커밋 실습

- 직전 커밋에 비해 새로 만들었거나 수정하거나 삭제한 파일은 모두 [스테이지에 올라가지 않은 파일]에 보임
- 이제 소스트리 상단 [커밋]을 클릭하면 커밋할 수 있는 뷰로 바뀜



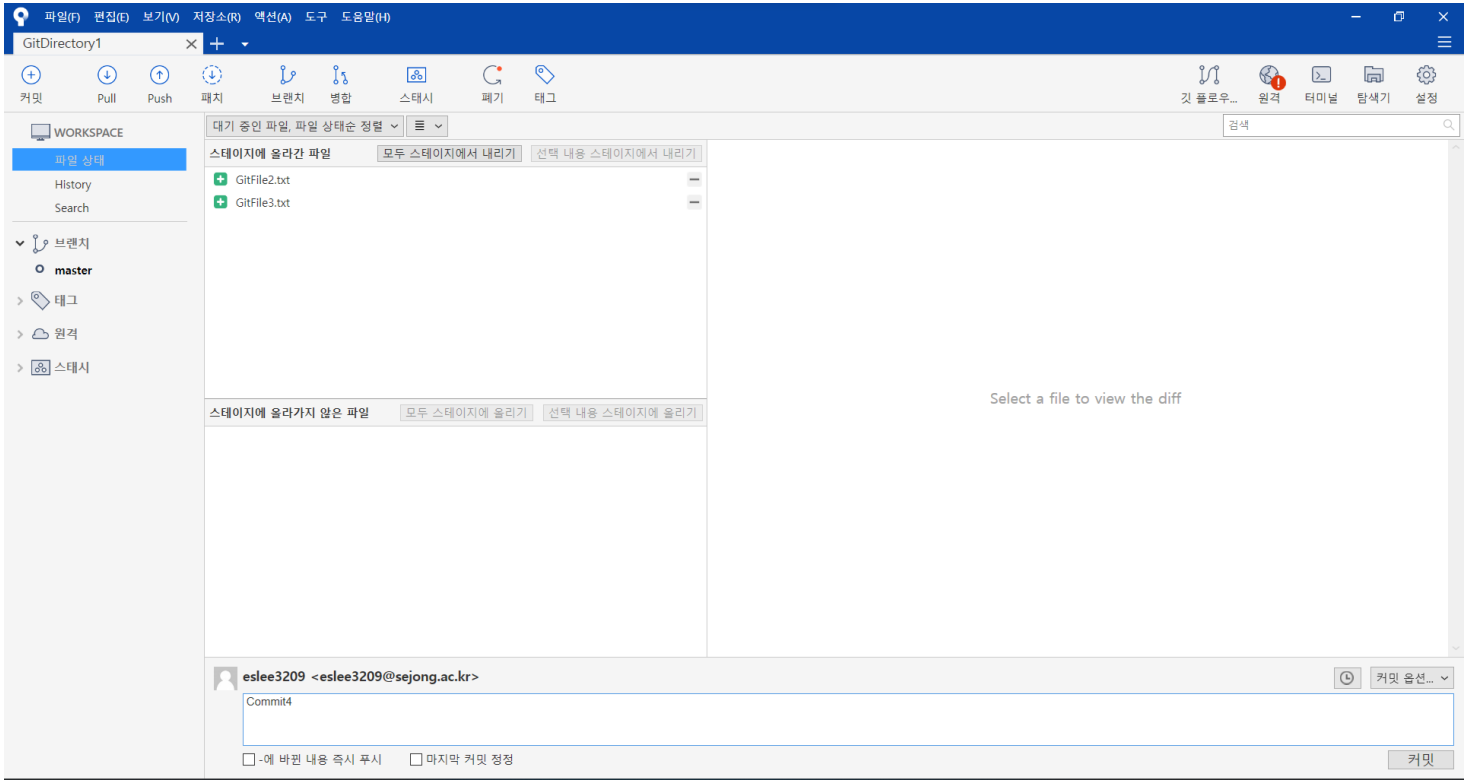
소스트리에서 커밋 실습

- 새로 만든 GitFile2.txt 및 GitFile3.txt 오른쪽 [+] 아이콘을 클릭함. 그러면 이 파일들이 위 섹션인 [스테이지에 올라간 파일]로 올라감. 이는 git add와 동일



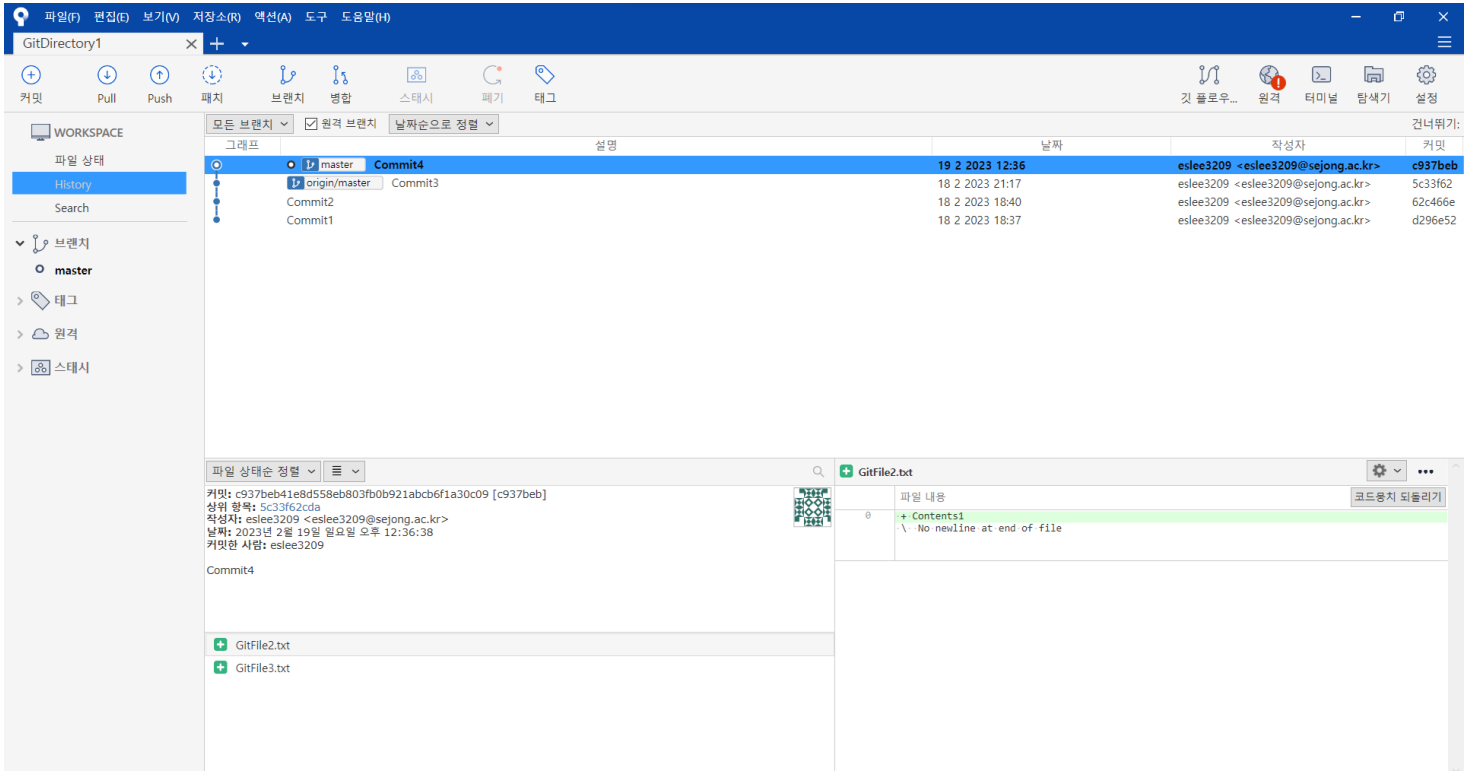
소스트리에서 커밋 실습

- 하단에 커밋 메시지를 입력함. 이는 CLI 환경에서 git commit -m 명령어와 동일함.
  - 예) Commit4로 메시지 입력
- 그 후 [커밋] 버튼을 클릭함. 그러면 커밋이 하나 추가됨



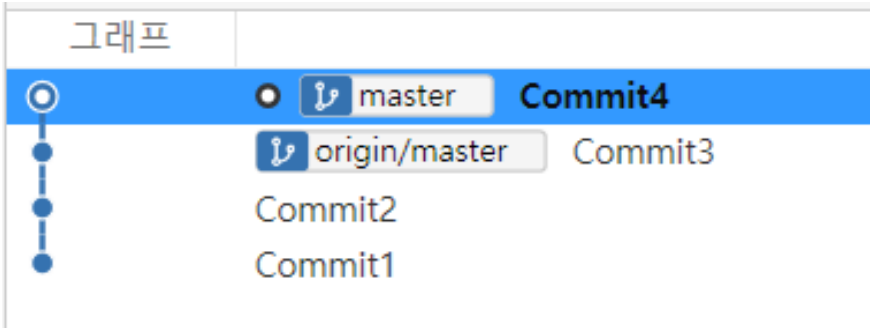
소스트리에서 커밋 실습

- 소스트리에서 왼쪽 [History] 탭을 누르면 아래 그림처럼 새로운 커밋이 추가된 것을 볼 수 있음
- 해당 커밋을 클릭하면 커밋의 상세 설명과 파일에 어떤 변경이 일어났는지도 볼 수 있음



소스트리에서 push 실습

- 소스트리 그래프
  - 아래와 같이 소스트리에서 커밋 그래프를 볼 수 있음
  - master는 현재 로컬저장소에서의 버전을 나타냄
  - origin/master는 원격저장소에서의 버전을 나타냄
  - 즉, 원격저장소의 버전이 로컬저장소 버전에 하나 뒤쳐져있음을 의미함



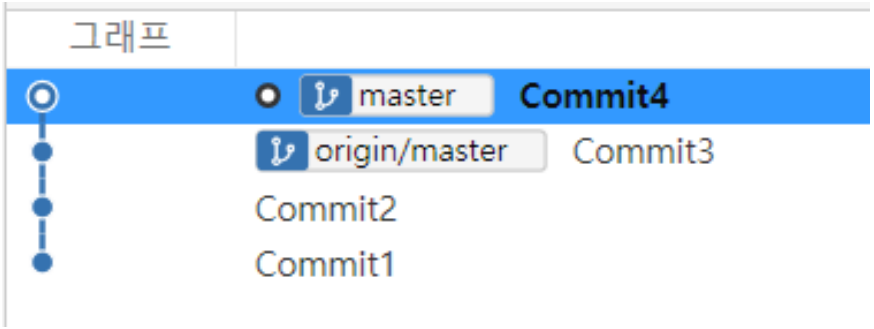
Origin

- origin은 내가 연결한 github 원격저장소의 닉네임임
- CLI 환경에서 아래 명령어로 로컬저장소에게 원격저장소 주소를 알려준 적이 있었음
  - git remote add origin https://github.com/eslee3209/GitRepository1.git
- 이는 origin이란 이름으로 원격저장소를 추가하라는 뜻임
- 만약 아래와 같이 origin 대신 myOrigin으로 바꾼다면 나중에 push할 때 myOrigin 이름을 계속 써야함
  - git remote add myOrigin https://github.com/eslee3209/GitRepository1.git

Master

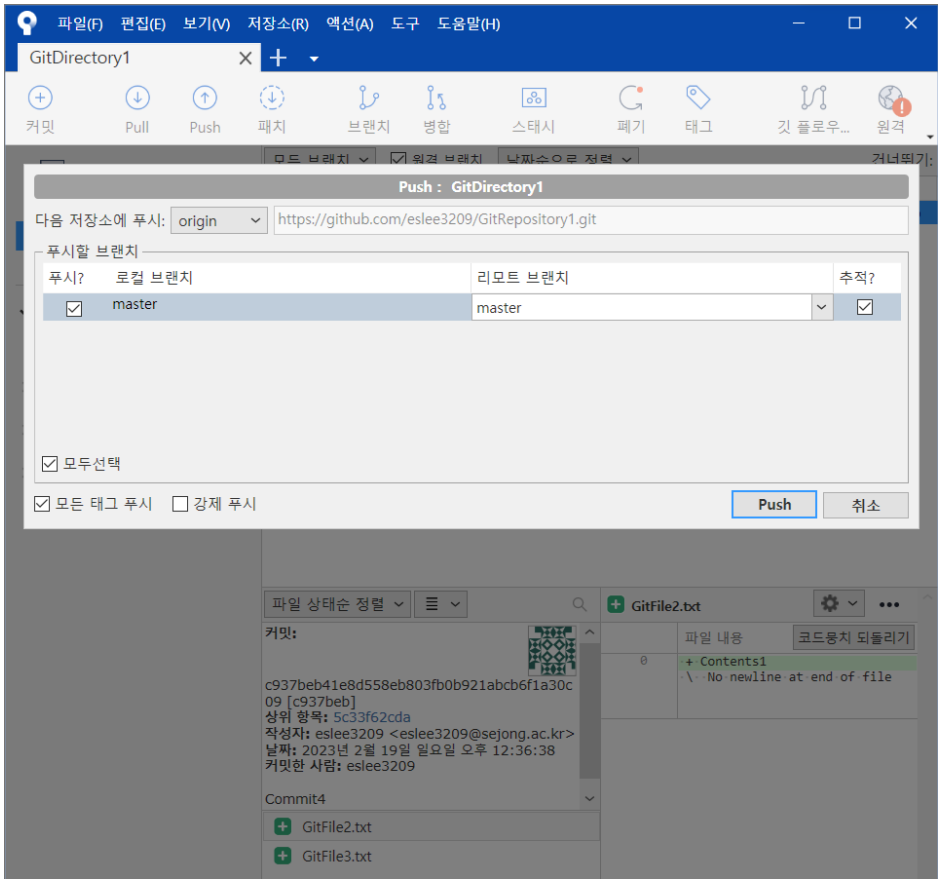
- 소스트리의 [History]에서 그래프를 보면 커밋이 줄줄이 기차처럼 하나의 줄기로 이어져있음. 이러한 줄기를 브랜치(branch)라고 함
- 따로 브랜치를 생성하지 않으면 git은 master라는 기본 줄기에 커밋을 올림. 즉, CLI 환경에서 git init으로 프로젝트를 시작하면 master 브랜치가 기본 브랜치가 됨

*** github에서 직접 프로젝트를 생성한 후 clone을 하면 main 브랜치가 기본 브랜치가 됨



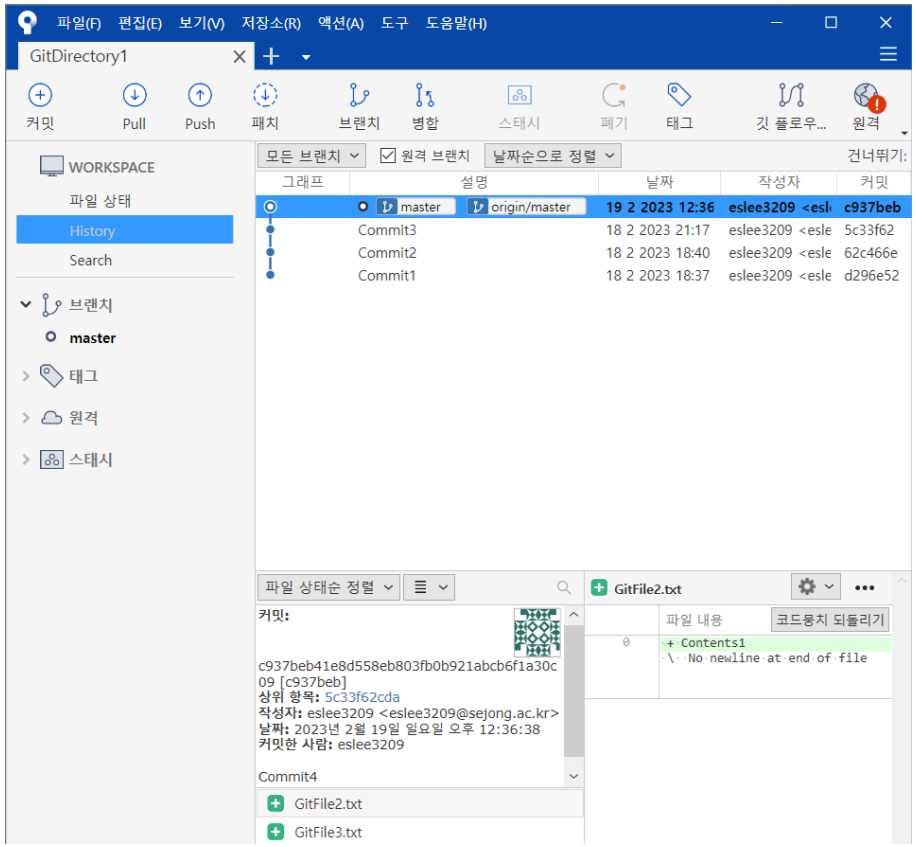
소스트리에서 push 실습

- 이제 새로 만들어진 커밋(Commit4)를 push할 것임
- [Push] 버튼을 클릭하고 push할 브랜치(master) 옆의 체크박스를 체크함. 그 후 [Push]를 클릭함



소스트리에서 push 실습

- 그러면 push할 브랜치의 모든 새 커밋이 원격저장소에 올라감. git push origin master와 동일함
- 아래 그림처럼 origin/master가 가장 최신 커밋을 가리키고 있음



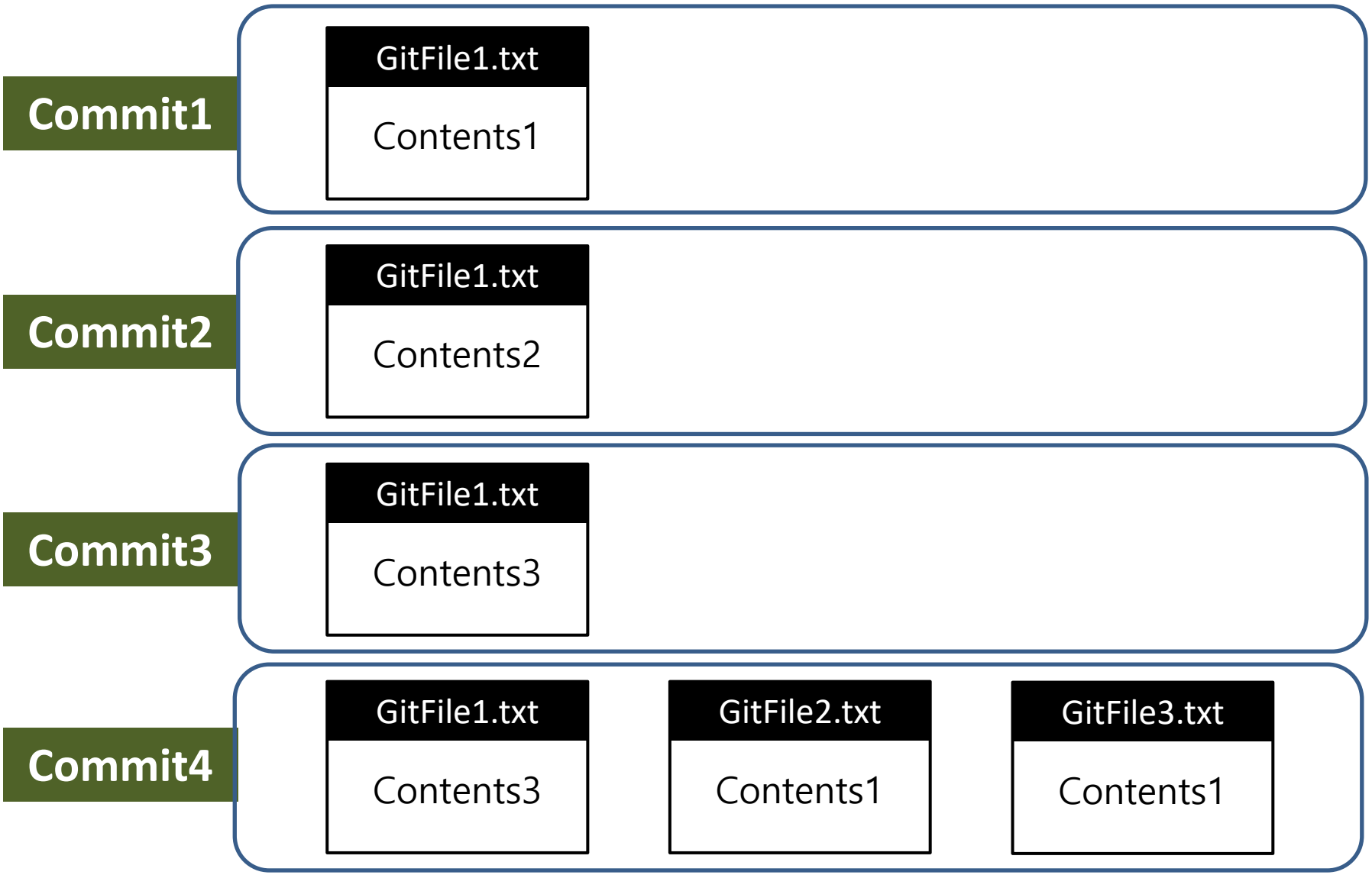
커밋은 스냅샷

- 한 때 개발 업계를 평정했던 SVN(SubVersion)과 같은 버전 관리 시스템과 git의 가장 큰 차이점은 git이 커밋에 바뀐 것만 저장하는 것이 아니라 전체 코드를 저장한다는 점임
- 스테이지에 올라온 것들을 사진 찍어줌

커밋은 스냅샷

- SVN처럼 차이점만 저장한다면 맨 처음부터 거슬러 올라가며 바뀐 점을 모두 반영하여 계산해야함
- 그러나, 스냅샷을 저장하는 git은 계산이 필요없음. 바로 앞 커밋과 비교연산 한번만 하면 됨
- 그리고, 바뀌지 않은 파일은 이전 파일의 링크만 저장하므로 용량도 적고 계산도 필요 없음
- 따라서 git의 명령은 빨리 동작함

커밋은 스냅샷


Commit1
GitFile1.txt
Contents1
Commit2
GitFile1.txt
Contents2
Commit3
GitFile1.txt
Contents3
Commit4
GitFile1.txt
Contents3
GitFile2.txt
Contents1
GitFile3.txt
Contents1

스테이지 개념

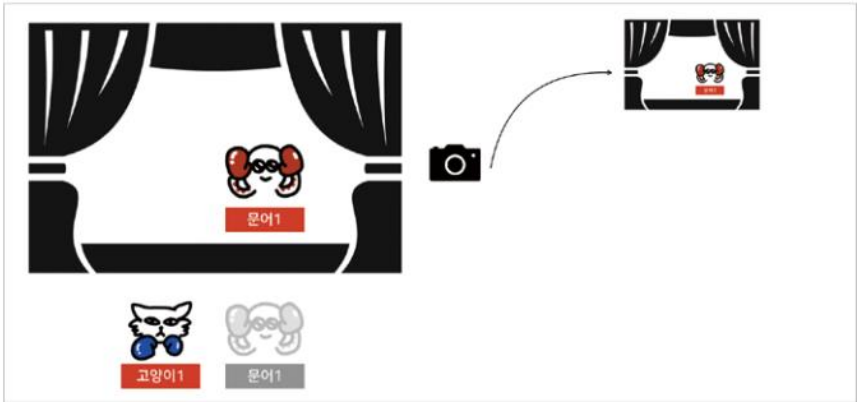
- 스테이지(stage)는 마치 무대에서 사진 찍기 위해 무대 위에 사람들을 올리는 것과 같음
- 스냅샷을 찍을 것, 즉, 커밋에 추가할 파일을 무대 위로 올리는 것임. 스테이지에 올린다고 말함
  - CLI환경에서 git add [파일명]으로 수행 가능함

스테이지 개념

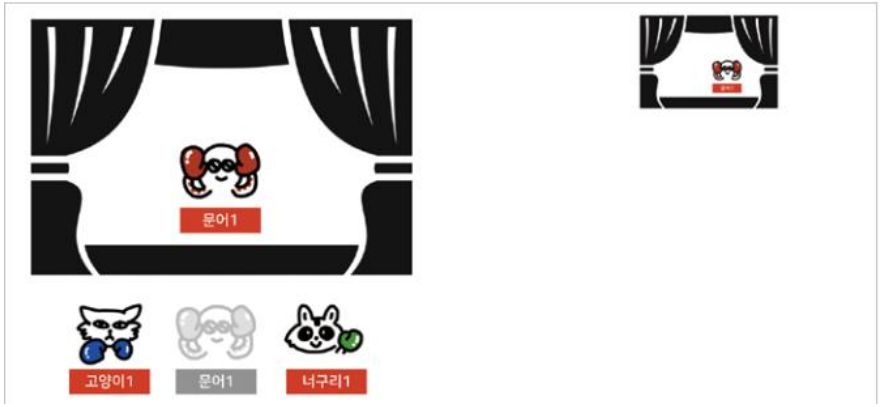
1

2

3

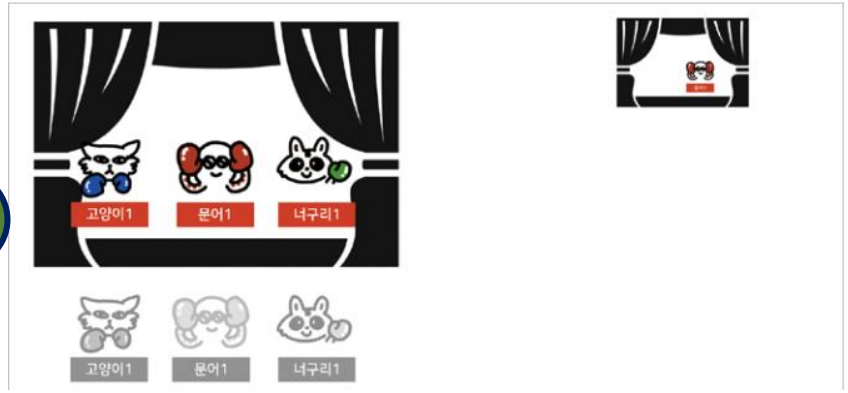


4

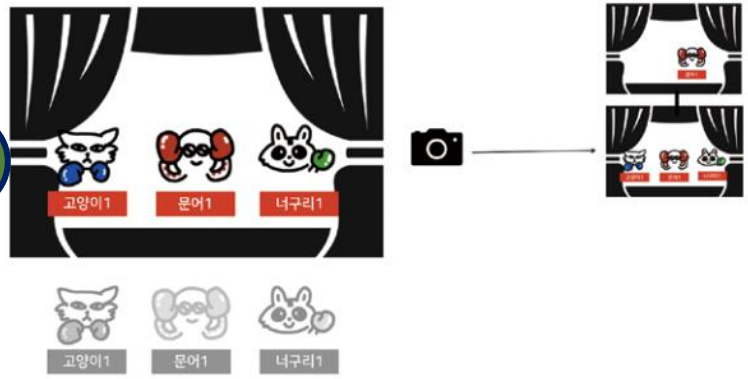


스테이지 개념

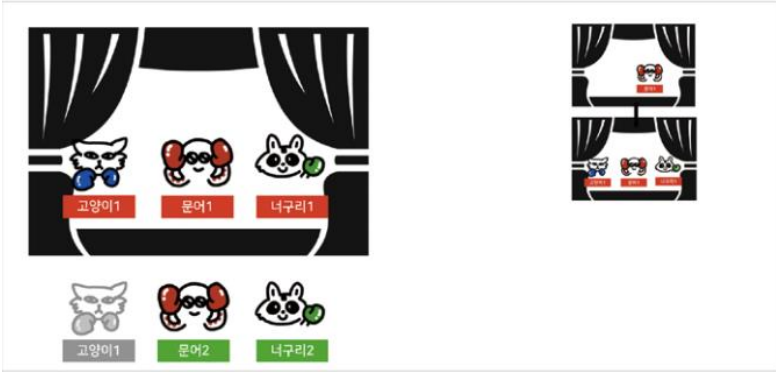
5



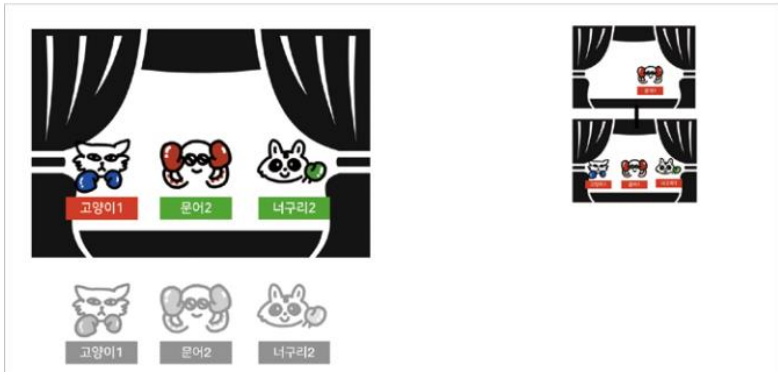
6



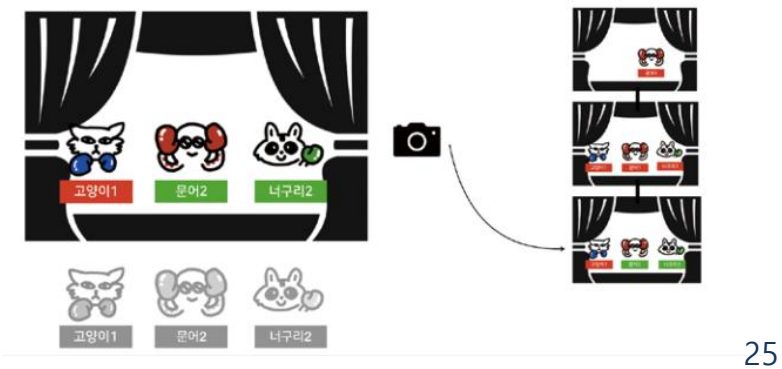
7



8



9



Git으로 관리하는 파일의 상태

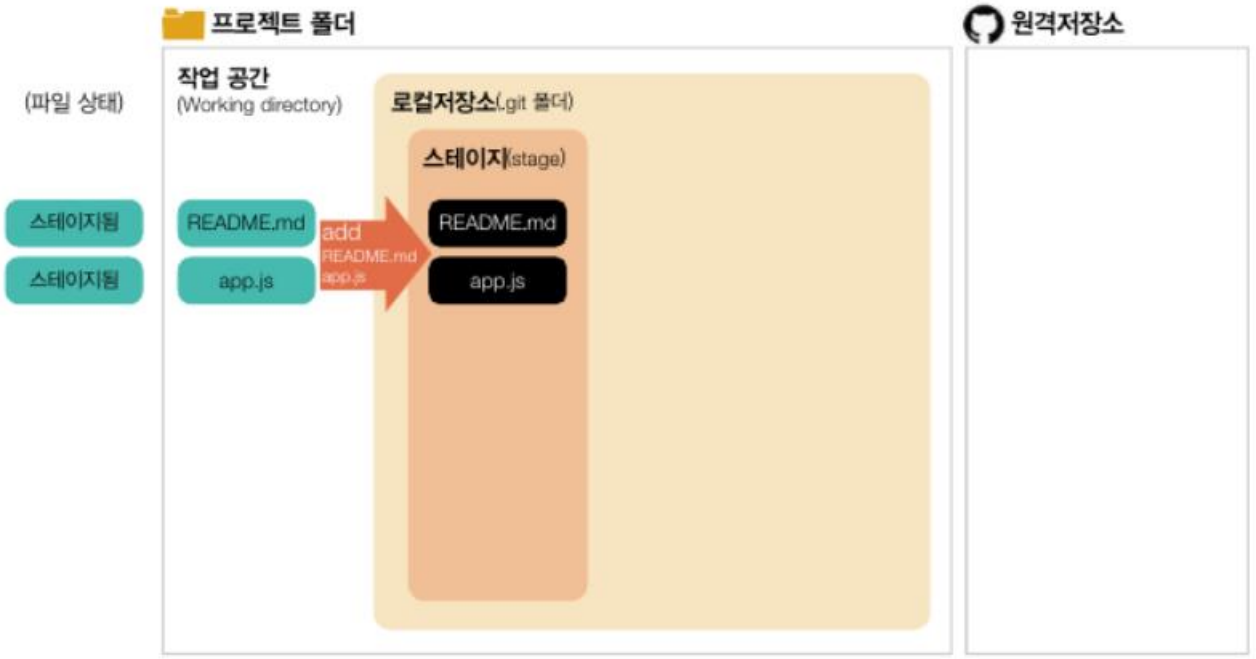
- Git으로 관리하는 파일에는 4가지 상태가 있음
- Untracked
  - 1. 추적 안 됨
- Tracked
  - 2. 수정 없음
  - 3. 수정됨
  - 4. 스테이지됨

Git으로 관리하는 파일의 상태

- 처음 로컬저장소를 생성하고 새 파일(README.md, app.js)을 만들면 이 두 파일은 한번도 커밋되지 않은 파일이기 때문에 '추적 안됨' 파일 상태임

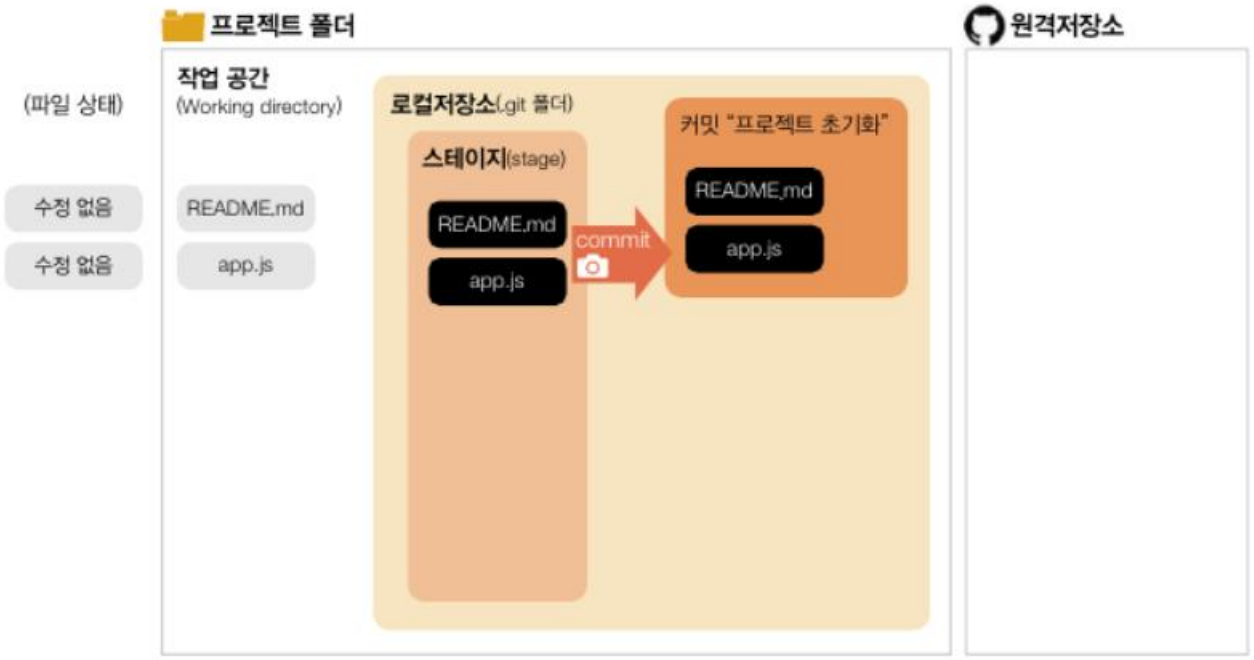
Git으로 관리하는 파일의 상태

- git add 명령어를 통해 두 파일 모두 스테이지에 올림. 파일 상태가 '추적 안됨 ' 에서 '스테이지됨(staged)'로 변경됨



Git으로 관리하는 파일의 상태

- 스테이지에 있는 파일 전체를 commit 명령어를 통해 하나의 스냅샷, 즉 버전으로 만듦. 파일 상태가 '스테이지됨 ' 에서 '수정없음 ' 으로 변경됨.



Git으로 관리하는 파일의 상태

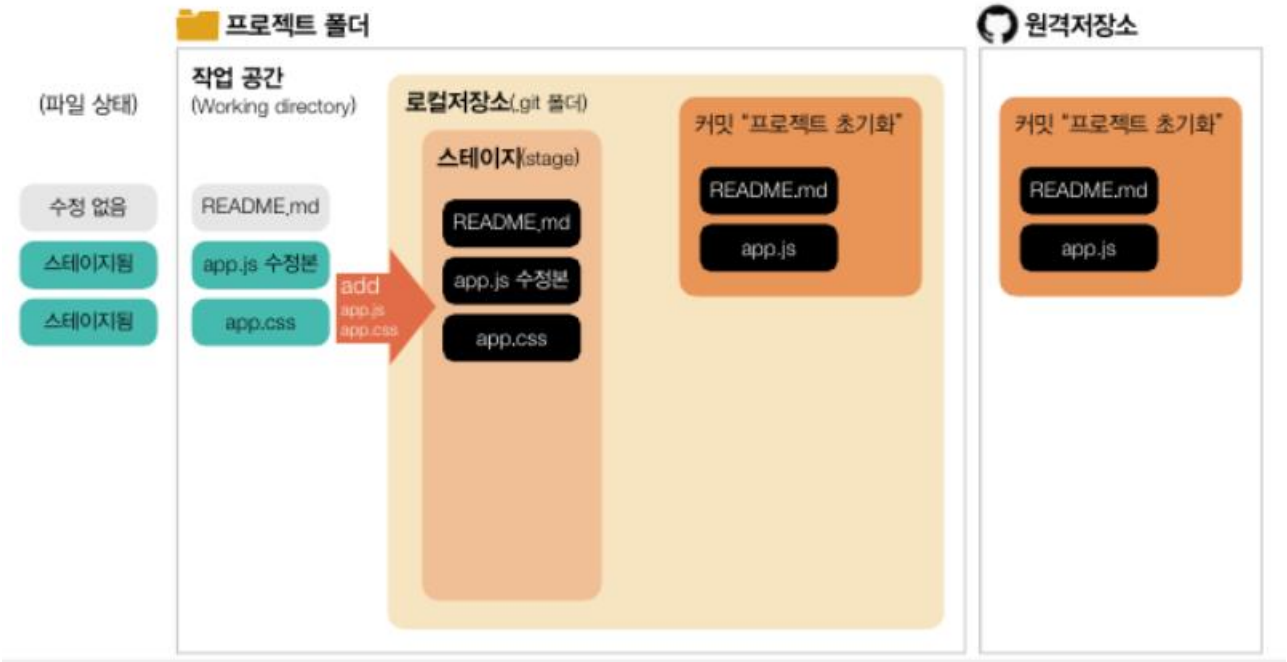
- 커밋을 push 명령을 통해 원격저장소에 push함

Git으로 관리하는 파일의 상태

- app.js를 수정하고 app.css라는 파일을 새로 생성함. 그러면 각각 '수정함 ' 과 '추적 안됨'으로 파일 상태가 됨

Git으로 관리하는 파일의 상태

- '수정 없음' 상태인 README.md 파일은 스테이지로 올릴 수 없음. 정확히는 이미 스테이지에 올라와있음. commit 명령을 통해 나머지 두 파일을 모두 스테이지에 올림.



Git으로 관리하는 파일의 상태

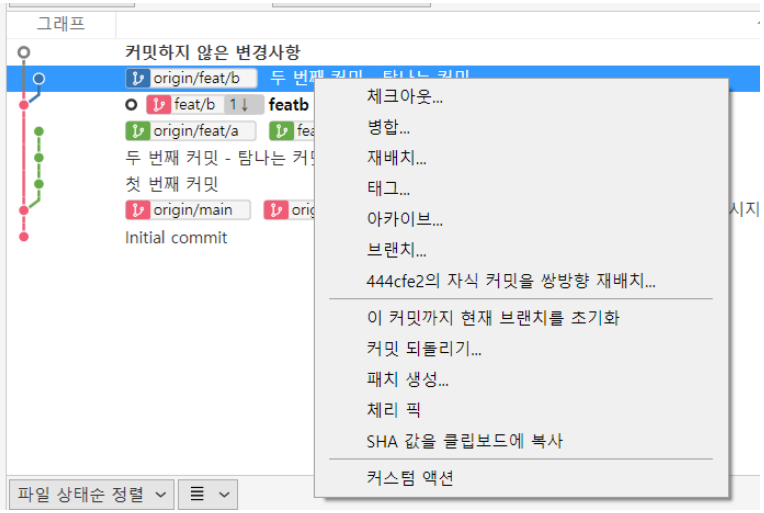
- 커밋을 통해 스냅샷을 만들어줌. 이 커밋은 앞서 만든 커밋인 '프로젝트 초기화 ' 에 연결되어있음. 그래퍼 앞 커밋에 비해 이번 커밋은 app.js가 수정되었고 app.css가 추가되었음을 git이 계산을 통해 알아낼 수 있음

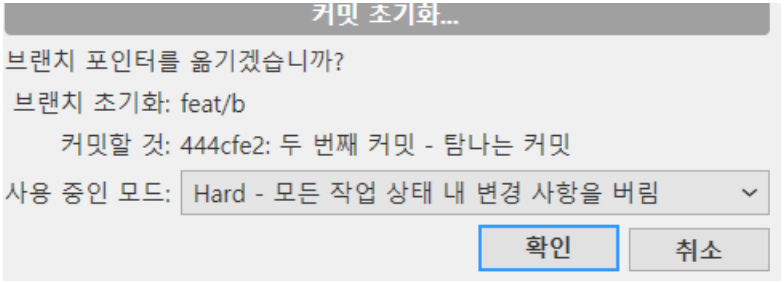
Git으로 관리하는 파일의 상태

- push까지 하면 새로 만든 하나의 커밋 또한 원격저장소에 올라감

소스트리에서 커밋 취소

- 이미 한 커밋을 취소하는 방법은?
- 최근 커밋을 이전 커밋으로 돌리기 위해서는 원하는 커밋에서 마우스 우측 버튼을 누르고 [이 커밋까지 브랜치를 초기화]를 클릭함
- 옵션을 선택해야함. 히스토리를 깔끔하게 돌리기 위해서는 “Hard”를 선택
- 경고창이 떠도 “예 " 를 클릭
- 그러면 과거로 돌아가며 현재 작업하던 변경사항도 없어짐





소스트리에서 커밋 취소

- 원격저장소에까지 push된 커밋을 취소하려면?
- 원하는 옛날 커밋에서 마우스 우측 클릭한 후 [이 커밋까지 현재 브랜치를 초기화] 선택
- Hard 모드 선택
- 그 다음 push를 하되 [강제 푸시]를 체크하고 진행함