

GUI 환경에서의 태그, release, fork

세종대학교 이은상

버전




- 프로그램을 출시할 때, 그리고 업그레이드할 때 이를 만든 회사에서는 버전을 명시함
 - 예) 포토샵 7.0, 아이폰14 등과 비슷함
- 의미있는 특정 시점의 맥락을 말하는 것임
- 버전을 올리는 것은 크게 메이저 업그레이드와 마이너 업그레이드로 나뉨
- 사용자들이 크게 느낄 변화를 적용했을 때 보통 메이저 버전을 올림
 - v2.x -> v3.x
- 작은 변화 등이 생겼을 땐 마이너 버전을 올림
 - v2.3 -> v2.4

버전

- Node.js 다운로드 페이지에 들어가면 18.14.1 버전과 19.6.1 버전을 제공함
- 가장 최신 버전은 19.6.1 버전이지만 안정적인 버전은 18.14.1 버전임
- LTS(long time support)는 장기 지원 버전의 약자인데 일반적인 버전보다 장기간에 걸쳐 지원하도록 특별히 만들어진 버전임
- 18.14.1을 보면 메이저 버전인 18, 마이너 버전인 1, 그리고 세 번째 버전이 있음. 이는 메인터넌스 버전으로 버그나 유지보수 등 작은 수정이 들어갔을 때 바꿈




다운로드
최신 LTS 버전: 18.14.1 (includes npm 9.3.1)

플랫폼에 맞게 미리 빌드된 Node.js 인스톨러나 소스코드를 다운받아서 바로 개발을 시작하세요.

LTS 대다수 사용자에게 추천	현재 버전 최신 기능	
 Windows Installer node-v18.14.1-x64.msi	 macOS Installer node-v18.14.1.pkg	 Source Code node-v18.14.1.tar.gz

다운로드
최신 현재 버전: 19.6.1 (includes npm 9.4.0)

플랫폼에 맞게 미리 빌드된 Node.js 인스톨러나 소스코드를 다운받아서 바로 개발을 시작하세요.

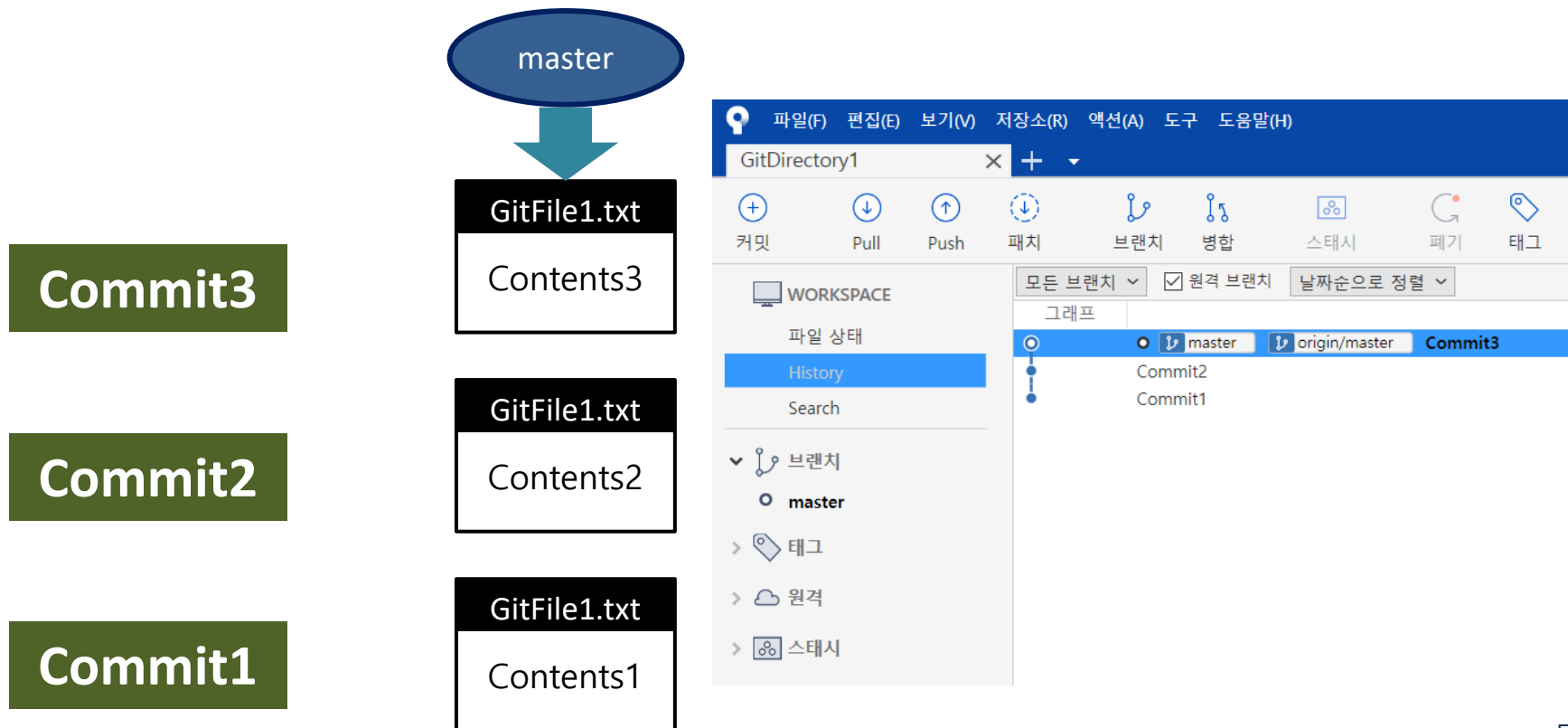
LTS 대다수 사용자에게 추천	현재 버전 최신 기능	
 Windows Installer node-v19.6.1-x64.msi	 macOS Installer node-v19.6.1.pkg	 Source Code node-v19.6.1.tar.gz

Release

- 프로그램을 출시하는 것을 release라고 함
- master 브랜치를 서버에 올려 사용자가 쓸 수 있도록 배포하고 현재 코드 상태를 v1.0.0이라고 기록한다고 함
- 이를 태그(tag)를 통해 간단히 표시할 수 있음. 특정 커밋에 포스트잇을 붙이는 느낌
- 브랜치는 특정 커밋을 가리키는 포인터라고 했는데 태그도 그러함

태그 실습

- 실습을 아래와 같은 상황에서 시작함

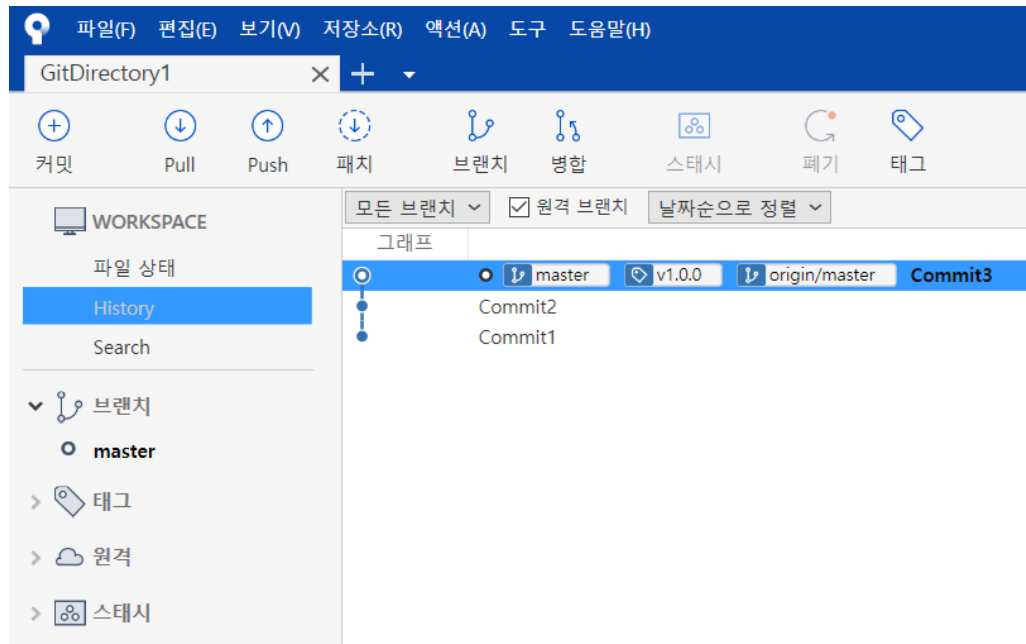


태그 실습

- master 브랜치 최근 커밋(Commit3)에 'v1.0.0'이라는 태그를 달 것임
- master 브랜치에서 소스트리 상단의 [태그] 아이콘 클릭
- 팝업 창이 뜨면 태그 이름을 'v1.0.0' 이라고 적고 [태그 추가] 버튼을 클릭함

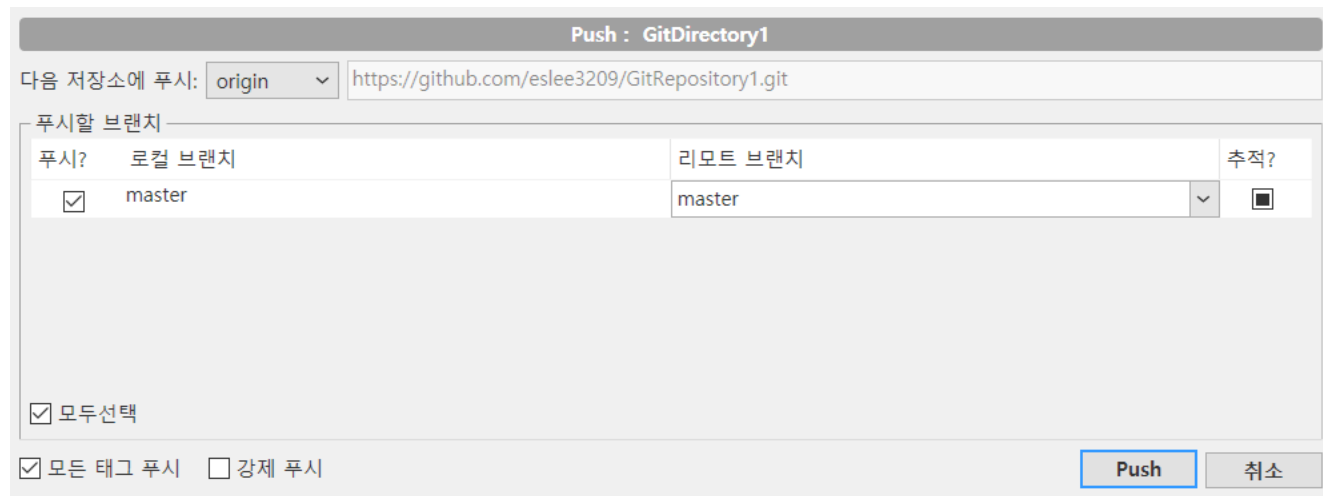
태그 실습

- 그러면 master 브랜치 라벨 옆에 [v1.0.0] 라벨이 새로 붙은 것을 확인할 수 있음
- 브랜치와 비슷하게 생김. 둘 다 커밋을 가리키는 가벼운 포인터이기 때문임



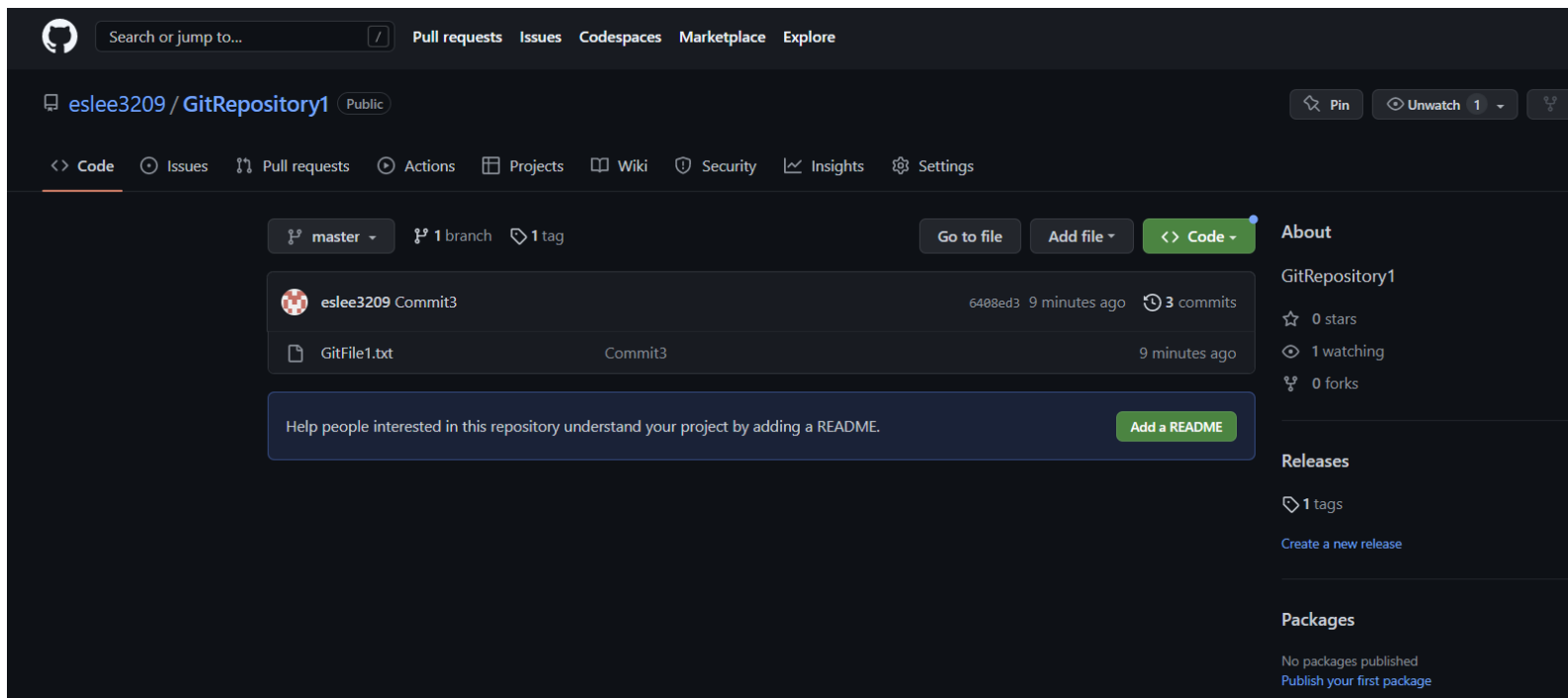
태그 실습

- 태그도 브랜치처럼 push를 해주어야 원격저장소에서 볼 수 있음
- [Push] 눌러 팝업 창 띄우고 하단에 [모든 태그 표시]라는 체크박스를 체크하고 [Push]를 클릭함
- 그러면 태그가 원격저장소에도 반영됨



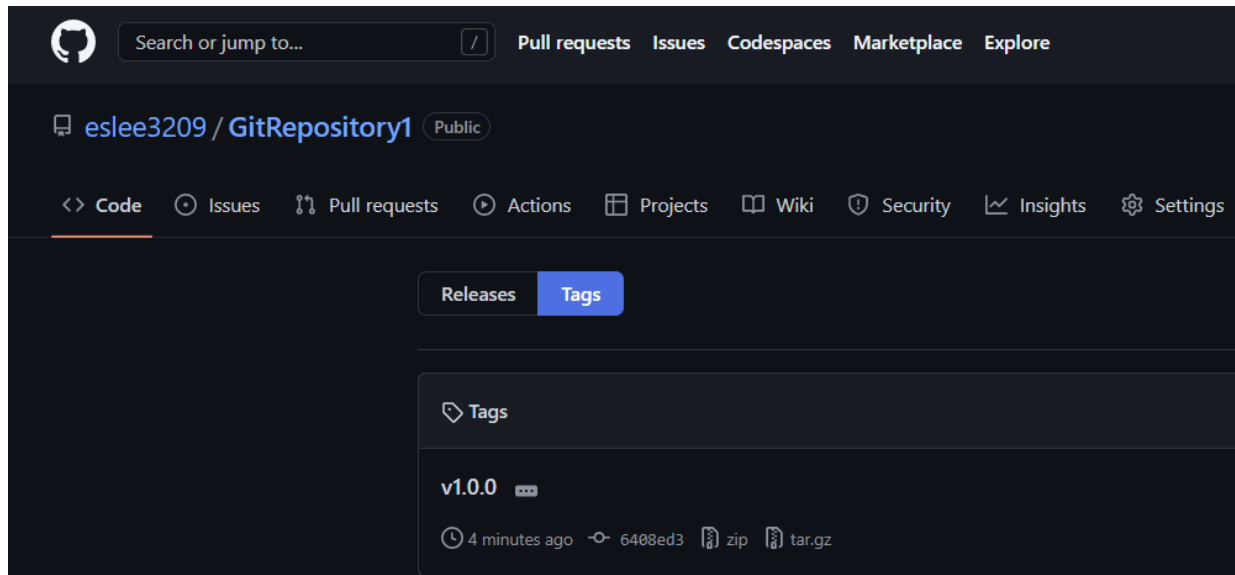
태그 실습

- GitHub에서 [GitRepository1] 원격저장소에서 브랜치명 옆을 보면 "1 tag"라고 표시된 탭이 있음. 혹은 우측에도 "1 tags"가 써있는 것을 볼 수 있음



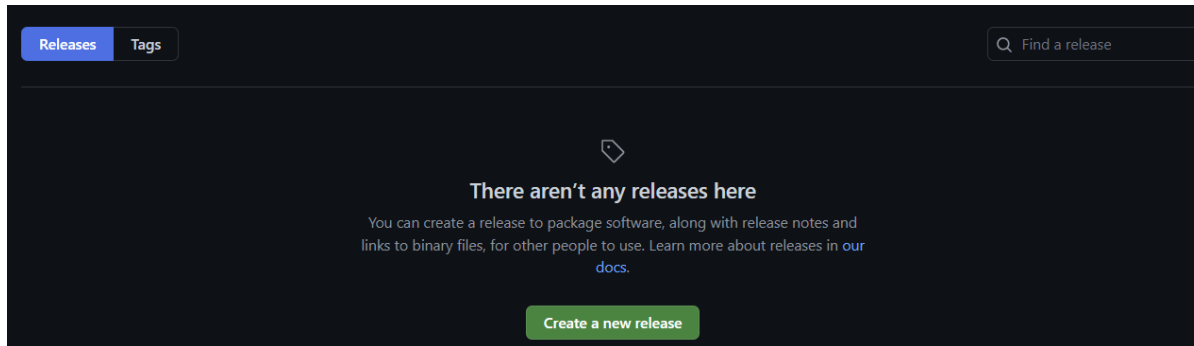
태그 실습

- "1 tag"를 클릭함
- 그러면 방금 만든 태그(v1.0.0)이 있음
- [zip] 아이콘을 누르면 해당 태그가 가리키는 버전을 압축파일로 내려받을 수 있음



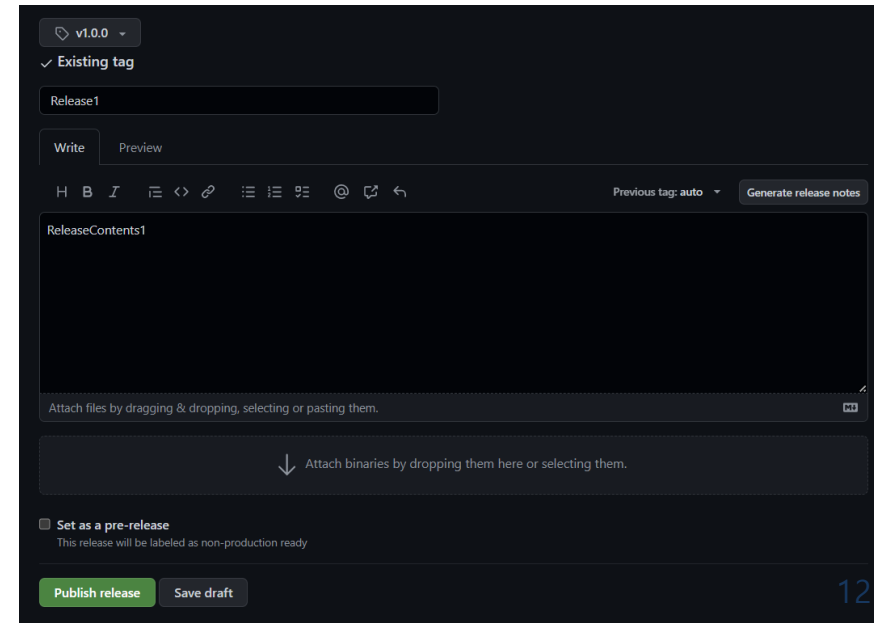
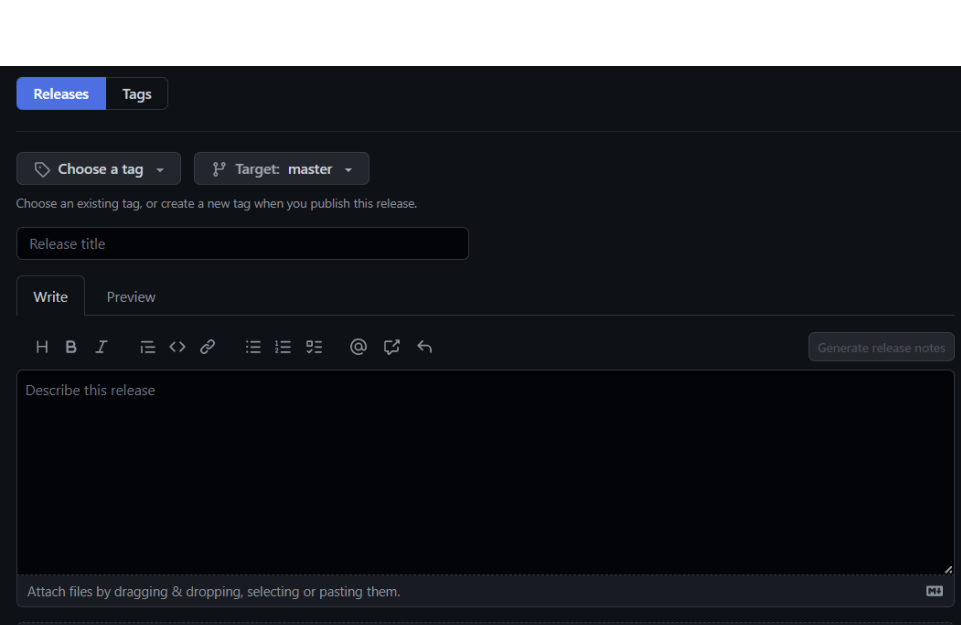
Release 실습

- 태그가 성공적으로 만들어졌으면 태그를 사용하여 release를 할 수 있음
- [GitRepository1] 원격저장소 우측에서 "Release"를 클릭 함
- 아직 release가 없다고 뜸. [Create a new release] 클릭



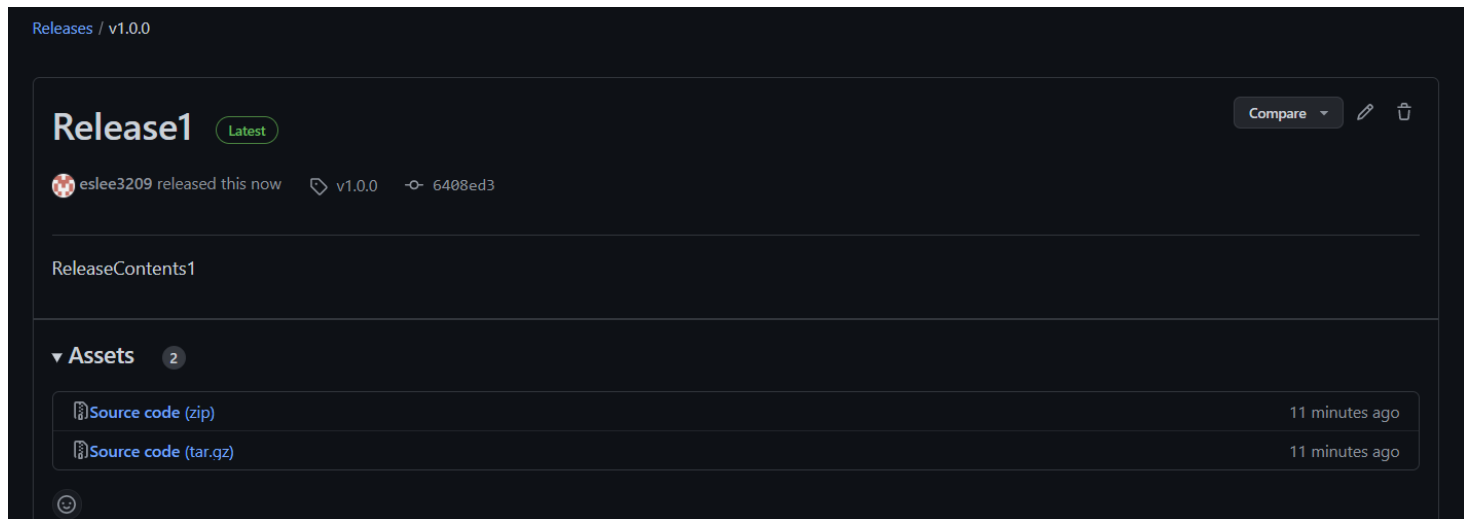
Release 실습

- 만들어두었던 태그를 선택하고 제목과 release 설명을
쓰
– 예) 제목: Release1
– release 설명: ReleaseContents1



Release 실습

- 그러면 성공적으로 release가 됨

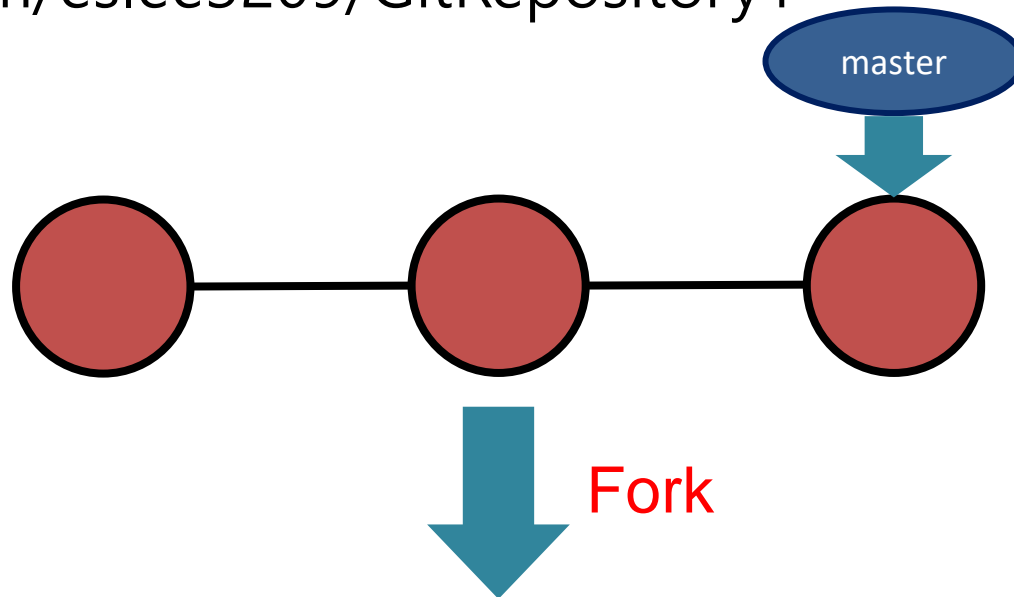


Fork

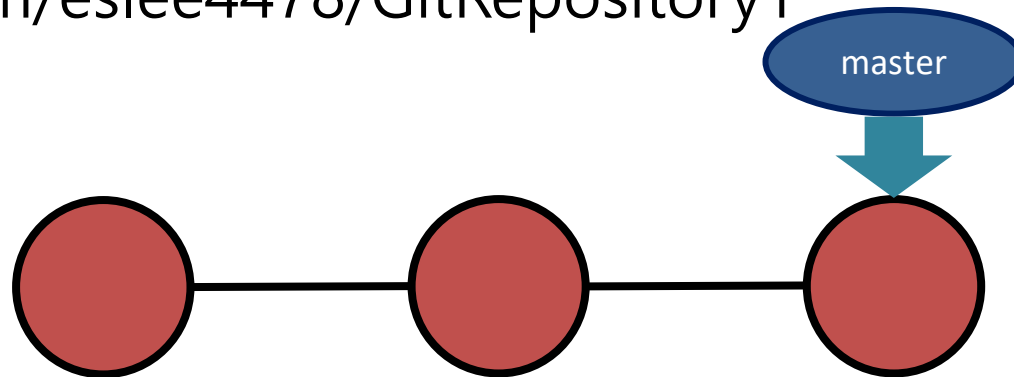
- 지금까지는 여러 사람이 하나의 원격저장소에 대해 각자 브랜치를 만들고 합치면서 협업하는 경우를 다룸
- 그러나 원격저장소 주인(Alice) 다른 사람(Bbob)에게 이 원격저장소에 바로 커밋을 올릴 권한을 주기 어려운 경우가 있음. Alice가 관리하는 원격저장소를 특별히 "원본 저장소"라고 부르기도 함
- 따라서 Bob은 원본저장소를 복사해 본인의 github에 새로운 원격저장소를 만들고 이곳에 커밋을 올림
- 새 원격저장소는 Bob만 사용하는 것이므로 온갖 실험적인 커밋을 올릴 수 있음
- 이렇게 남의 원본저장소를 내 계정의 원격저장소로 복사해오는 명령어는 fork임

Fork

- github.com/eslee3209/GitRepository1



- github.com/eslee4478/GitRepository1



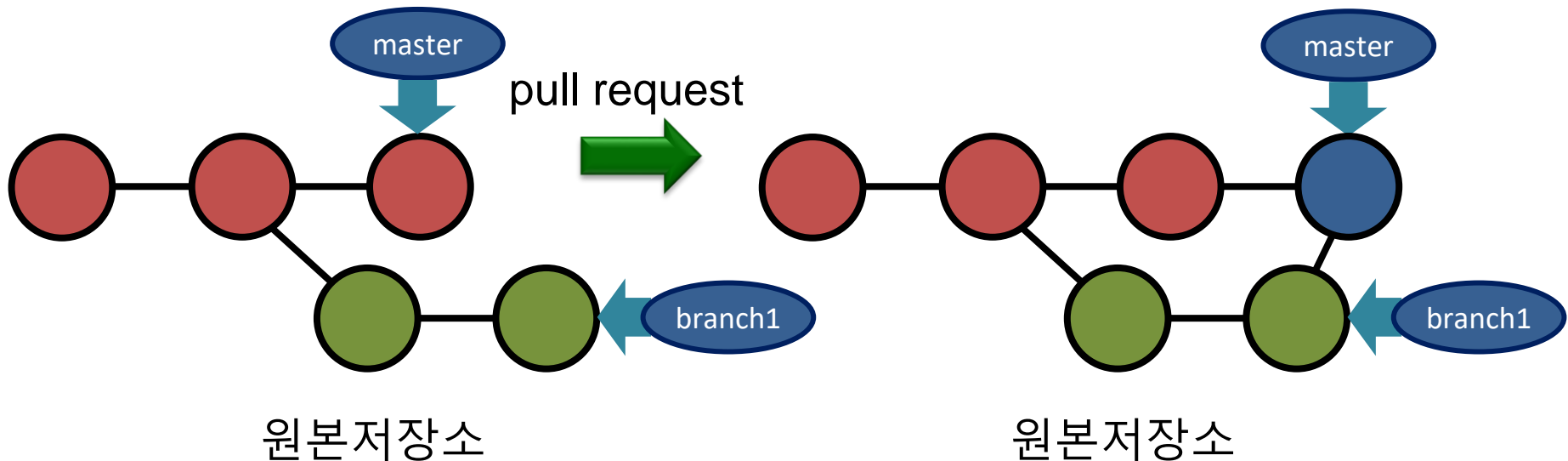
단순 복사

Pull Requests 종류

- Pull requests는 두 가지 종류로 분류될 수 있음
- 브랜치
 - 두 협력자가 동일한 원격저장소에서 각자 다른 브랜치를 만들어 작업을 수행함
 - 이 경우 pull requests는 한 사람이 자신의 브랜치(예, branch1)에서 작업이 끝나 해당 브랜치를 master 브랜치로 병합해도 될지 다른 협력자에게 확인을 부탁하는 메시지를 보내는 것임
- Fork
 - 원본저장소 소유자가 아닌 협력자가 먼저 원본저장소를 fork하여 자신의 원격저장소에서 작업함
 - 이 경우 pull requests는 협력자가 작업이 끝난 것을 원본저장소 master 브랜치에 합쳐도 될지 원본저장소 소유자에게 승인을 부탁하는 메시지를 보내는 것임

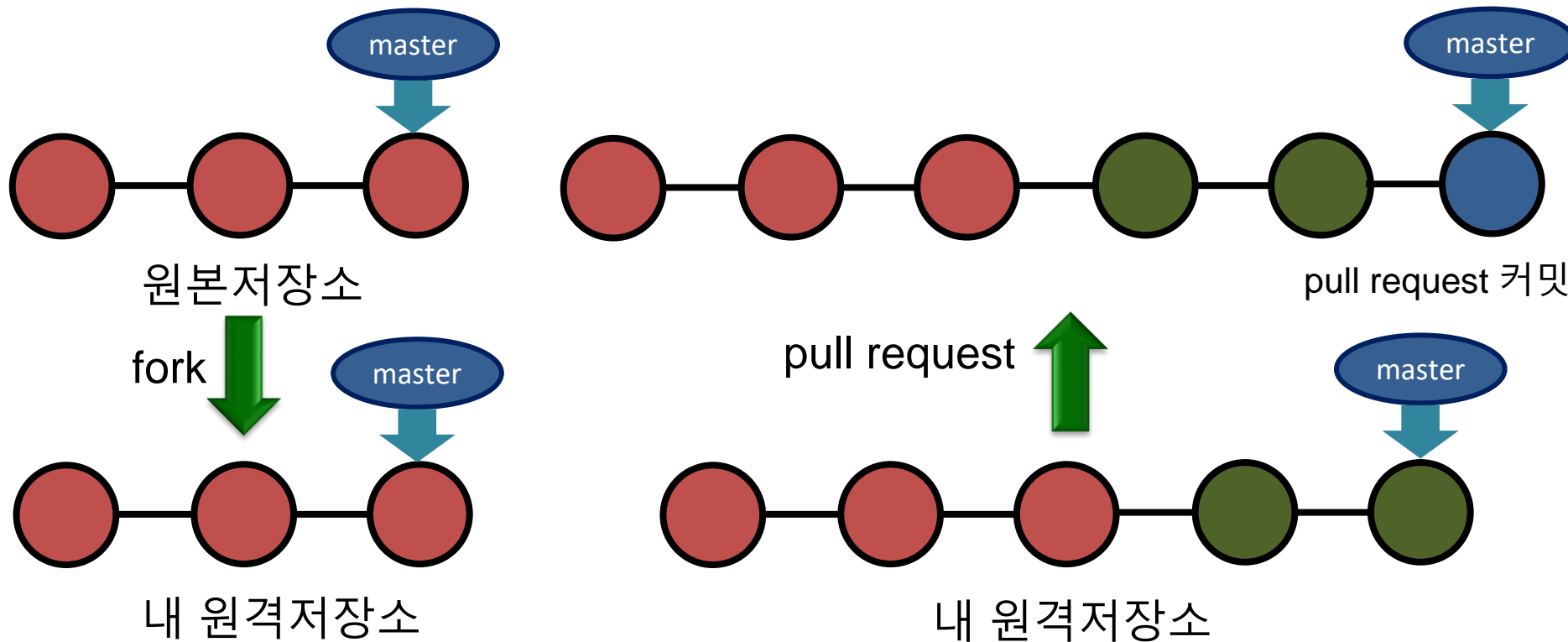
브랜치 v.s. fork

- 협업을 위해 브랜치를 만들어 작업하고 pull request하는 것과 fork를 한 후 작업하고 pull request하는 것은 비슷하지만 차이점이 있음
- 먼저 브랜치로 협업을 하는 경우 아래 그림처럼 브랜치를 생성하여 작업한 후에 pull request를 하여 master와 병합하는 형태임



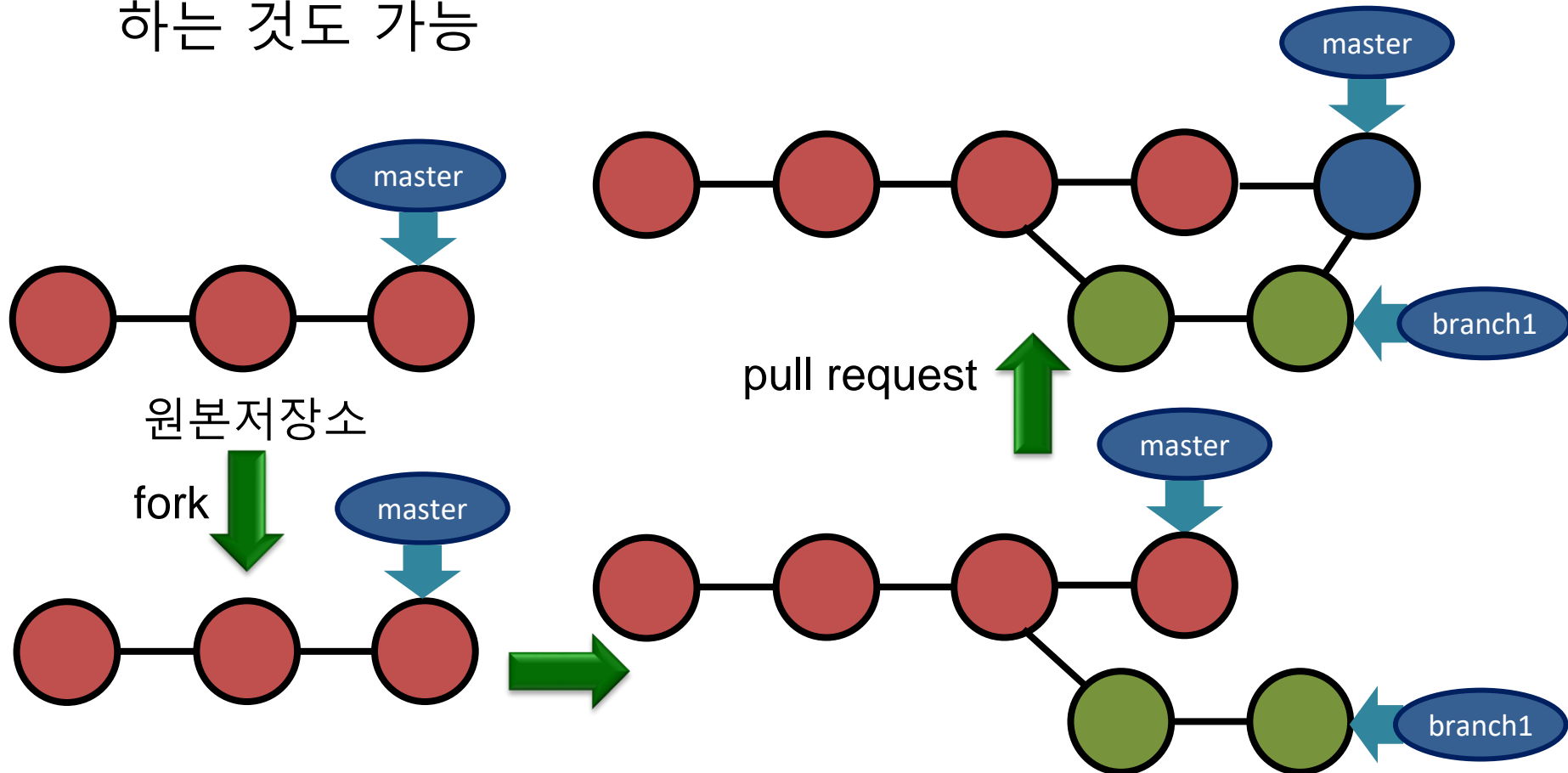
브랜치 v.s. fork

- fork로 협업하는 경우 원본저장소를 자신의 원격저장소로 복사(fork)한 후 변경하려는 브랜치(master)에 작업을 하여 커밋한 후 pull request하여 원본저장소에 반영하는 것임



브랜치 v.s. fork

- fork를 한 후 새로 브랜치(branch1)을 만들어 작업한 후 원본저장소의 master 브랜치에 합치도록 pull requests 하는 것도 가능



브랜치 v.s. fork

	의의	편리한 점	불편한 점
브랜치	하나의 원본저장소에서 분기를 나눈다.	하나의 원본저장소에서 코드 커밋 이력을 편하게 볼 수 있다.	다수의 사용자가 다수의 브랜치를 만들면 관리하기 힘들다.
포크	여러 원격저장소를 만들어 분기를 나눈다.	원본저장소에 영향을 미치지 않으므로 원격저장소에서 마음껏 코드를 수정할 수 있다.	원본저장소의 이력을 보려면 따로 주소를 추가해야 한다

- 개발자가 많아지면 브랜치로 관리하고 어려워짐
- 예를 들어, 메타의 오픈소스 리액트는 contributor가 1200명이 넘음 (2019년 1월 기준)

Fork 실습

- Fork 실습을 위해 두 계정이 필요함
- GitHub에서 새 계정을 가입함
- 참고로 gmail의 + 기능을 사용하면 이메일 주소 하나로 마치 여러 개의 이메일 주소인 것처럼 GitHub에서 사용할 수 있음
- 예) opensource@gmail.com 및 opensource+2@gmail.com



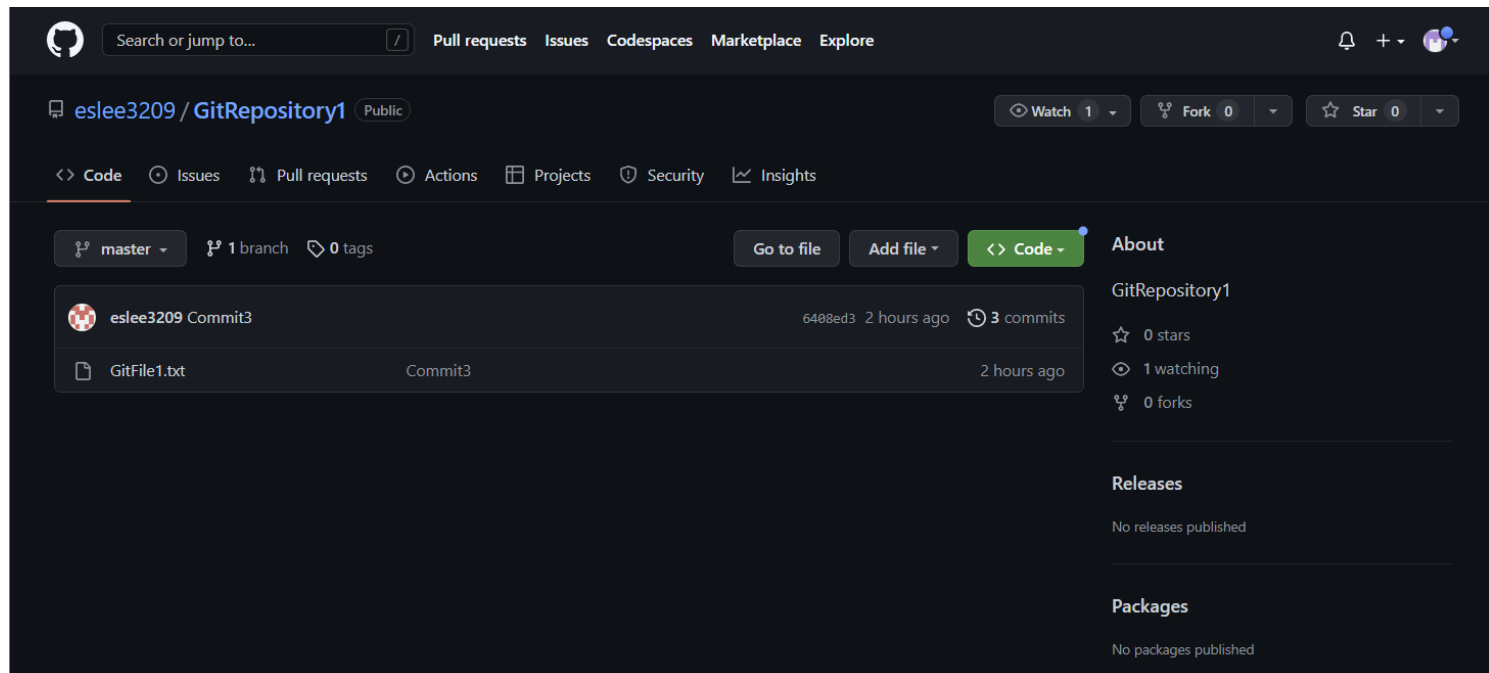
하나의 메일 주소를 여러 개처럼 사용하기

Gmail의 + 기능을 사용하면 이메일 주소 하나로 마치 여러 개의 이메일 주소인 것처럼 GitHub에서 사용할 수 있습니다. 내가 만약 test@gmail.com 주소를 가지고 있다면 test+sub@gmail.com, 혹은 test+1@gmail.com 처럼 +와 함께 뒤에 아무 글자를 붙여서 가입해도 모든 메일은 test@gmail.com으로 오는 편리한 기능입니다. 다중 이메일 가입을 막는 사이트에서 여러 계정을 만들 때 유용합니다. 계정 생성이 완료되었다면 까먹지 않도록 아래 표에 적어두세요.

GitHub 계정	필자	독자
고양이	Cat-Hanbit hello.git.github@gmail.com	
너구리	Raccoon-Hanbit hello.git.github+2@gmail.com	

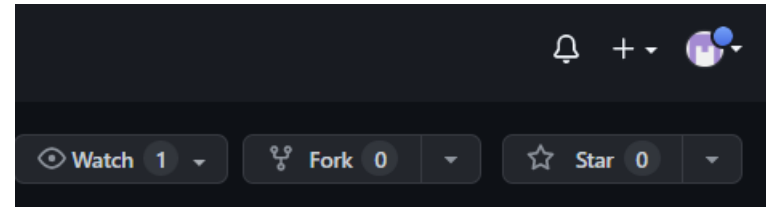
Fork 실습

- 새 계정으로 로그인함
 - 예) 기존 유저이름: eslee3209, 새 유저이름: eslee4478
- 원본저장소의 주소로 들어감
 - 예) <https://github.com/eslee3209/GitRepository1.git>
 - 아래 그림은 eslee4478 계정으로 eslee3209 계정의 GitRepository1 원본저장소로 들어간 것임



Fork 실습

- 우측 상단에 Fork를 클릭함
- 원격저장소 이름 및 description을 적을 수 있음. 원본저장소와 둘 다 동일하게 GitRepository1으로 함



Create a new fork

A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Owner * Repository name *

eslee4478 / GitRepository1 ✓

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

GitRepository1

☒ Copy the master branch only

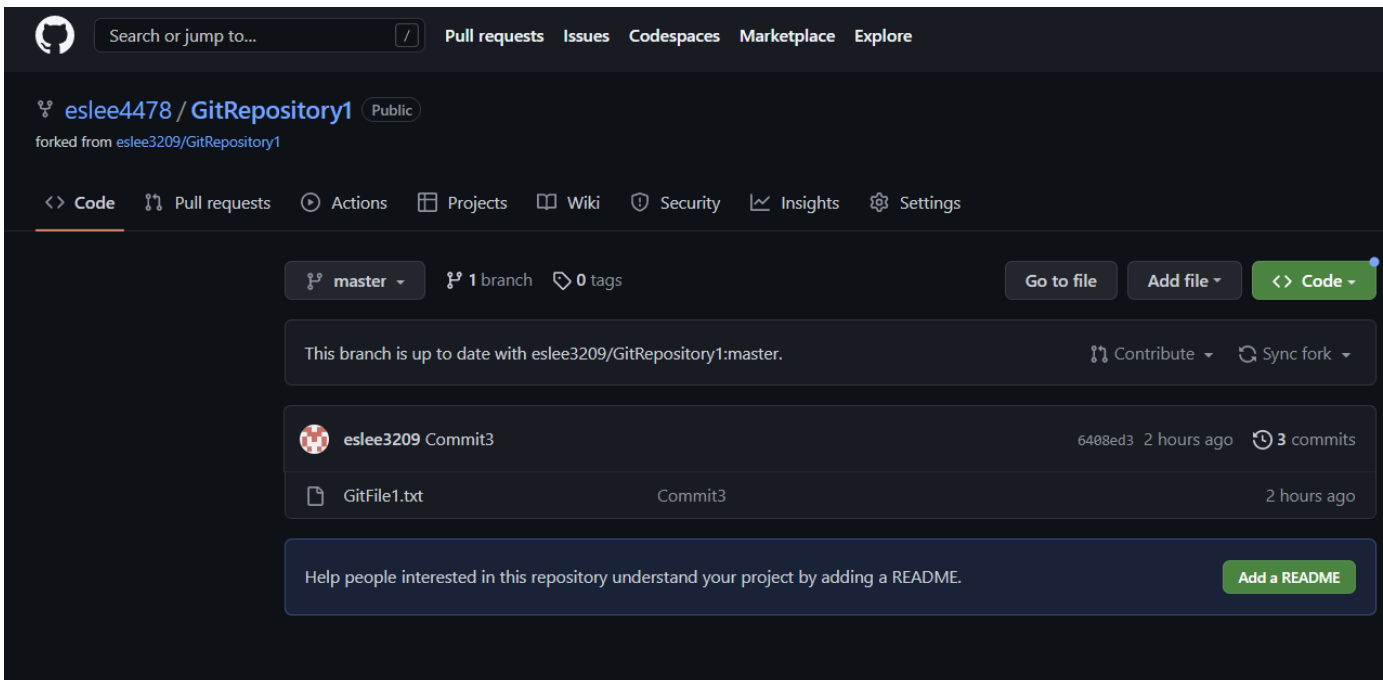
Contribute back to eslee3209/GitRepository1 by adding your own branch. [Learn more.](#)

i You are creating a fork in your personal account.

Create fork

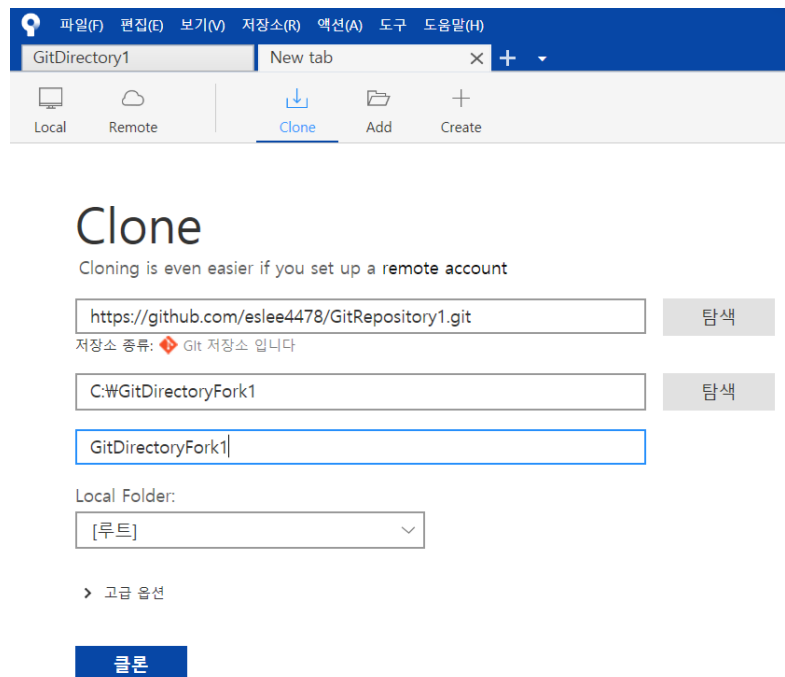
Fork 실습

- 그러면 fork가 성공적으로 이루어지며 eslee3209/GitRepository1 저장소에서 eslee4478/GitRepository1 저장소로 이동함
- "forked from eslee3209/GitRepository1"라고 fork된 표 시도 있음



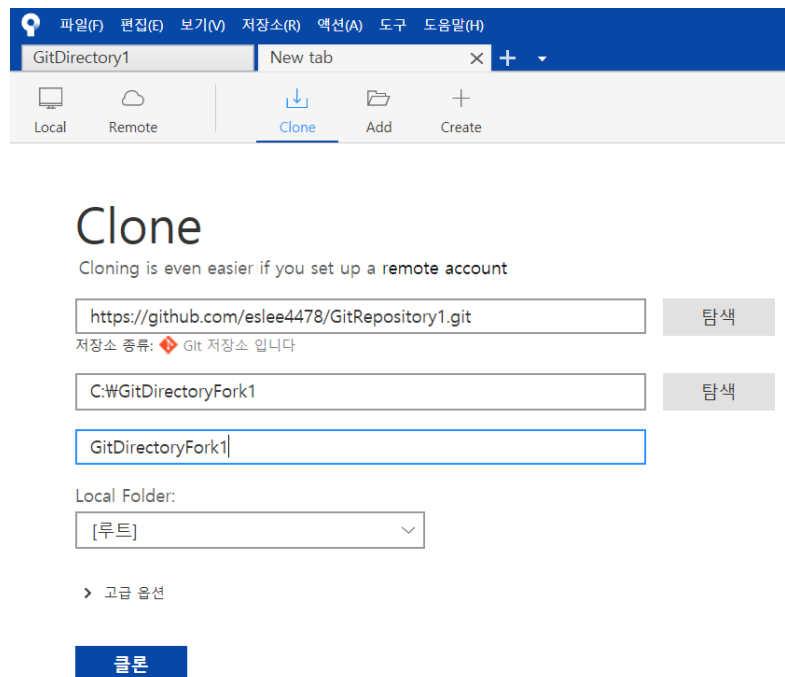
Fork 실습

- 이제 fork되어 새롭게 만들어진 원격저장소를 소스트리로 로컬 컴퓨터에 받아올 것임
- 먼저 fork한 원격저장소에서 원격저장소 주소를 복사함
 - 예) `https://github.com/eslee4478/GitRepository1.git`
- 소스트리에서 새로운 탭을 만들어 [Clone]을 클릭함



Fork 실습


- 원본저장소 주소와 클론할 폴더를 선택함.
 - 예) 클론할 폴더: C:\GitDirectoryFork1



The screenshot shows the GitHub 'Clone' dialog in a web browser. The browser's address bar shows 'GitDirectory1' and the page title is 'New tab'. The browser's menu bar includes '파일(F)', '편집(E)', '보기(V)', '저장소(R)', '액션(A)', '도구', and '도움말(H)'. The browser's toolbar shows 'Local', 'Remote', 'Clone', 'Add', and 'Create'. The 'Clone' dialog has a title 'Clone' and a subtitle 'Cloning is even easier if you set up a remote account'. It contains three input fields: the first for the repository URL 'https://github.com/eslee4478/GitRepository1.git', the second for the local path 'C:\GitDirectoryFork1', and the third for the local folder name 'GitDirectoryFork1'. Below these fields is a 'Local Folder:' label and a dropdown menu showing '[루트]'. At the bottom, there is a link '> 고급 옵션' and a blue '클론' button.

Clone

Cloning is even easier if you set up a remote account

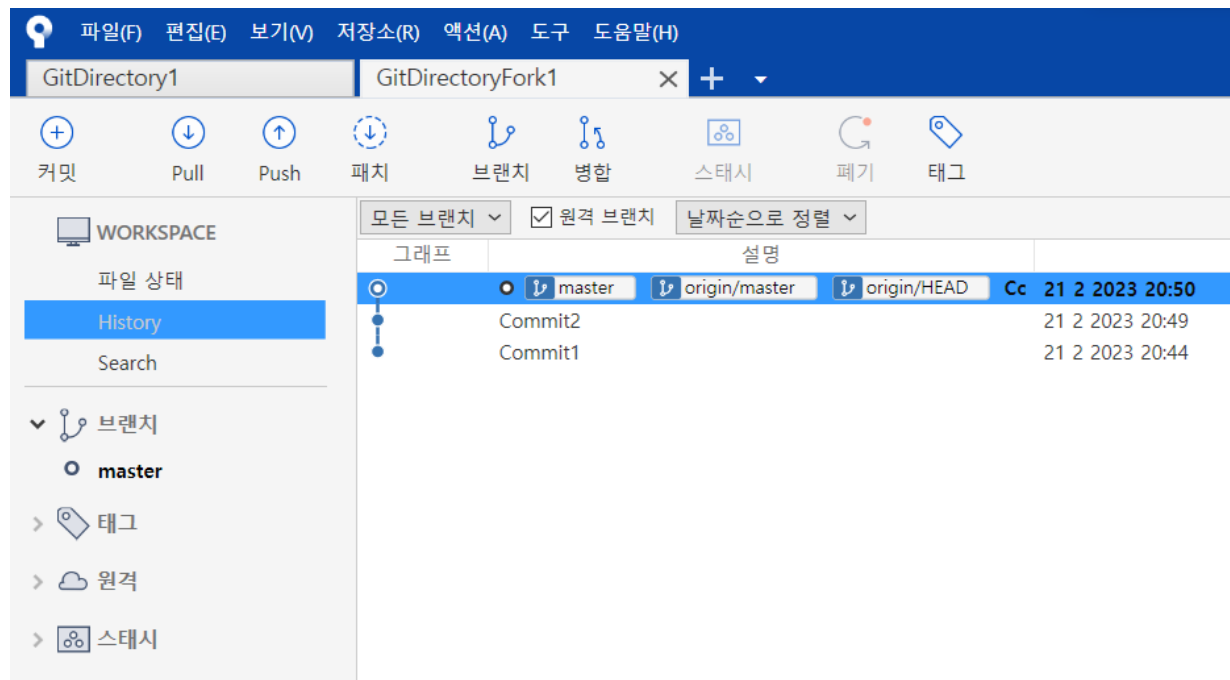
저장소 종류:  Git 저장소입니다

Local Folder:

> 고급 옵션

Fork 실습

- 아래와 같이 성공적으로 이루어짐
- 이제, 다른 계정(eslee4478)로 여기서 작업한 후 pull request를 하고자 함
- 먼저, 소스트리에서 github 다른 계정에 로그인해야함

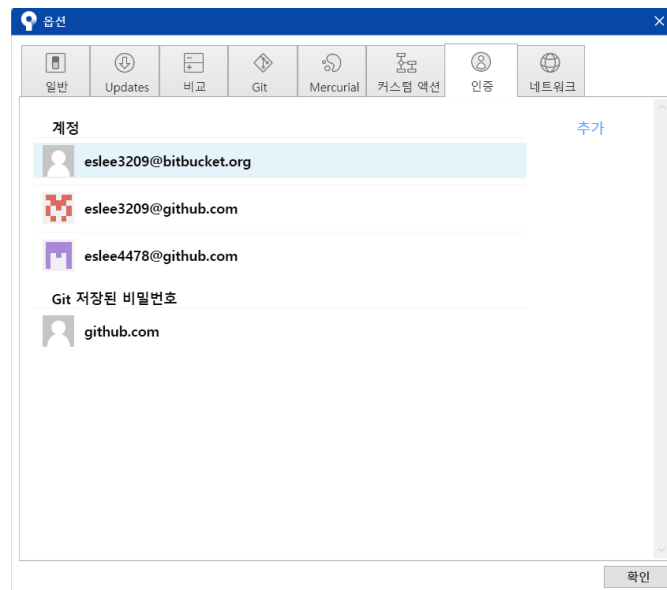


소스트리에서 GitHub 계정 추가

- 도구>옵션에 들어감
 - [인증] 탭에 들어가면 계정을 관리할 수 있음
 - [추가] 클릭하면 새 계정 추가 가능함
 - 옵션은 다음과 같이 함
 - 호스팅 서비스: GitHub
 - 선호 프로토콜: HTTPS
 - 인증: OAuth 혹은 Basic
- *** OAuth는 웹 브라우저를 띄워 로그인하는 방식. Basic은 github 닉네임과 암호를 입력해서 로그인하는 방식

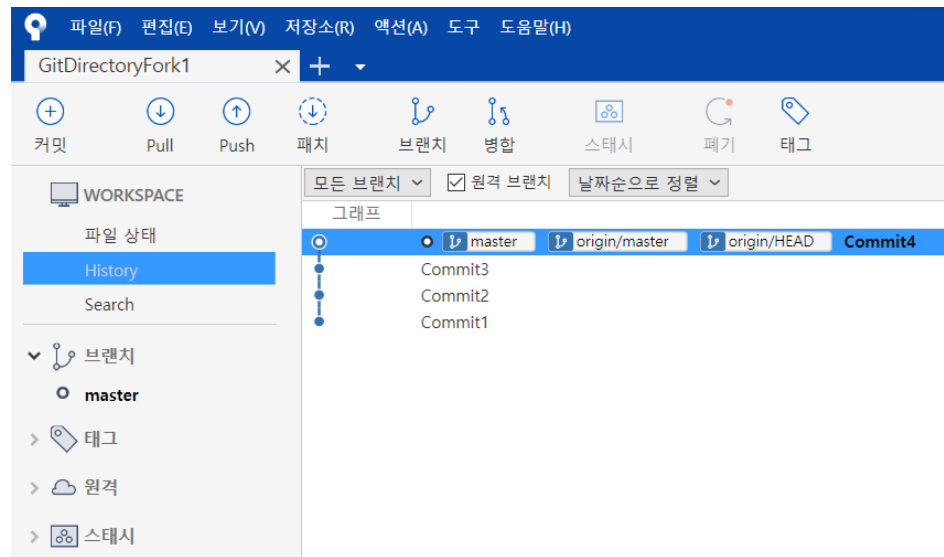
Fork 실습

- 그러면 계정이 추가됨 (eslee4478)
- 이 인증 화면에서 원하는 계정에서 "설정 초기화"를 누름
. 그러면 그 계정이 기본 로그인 계정이 됨



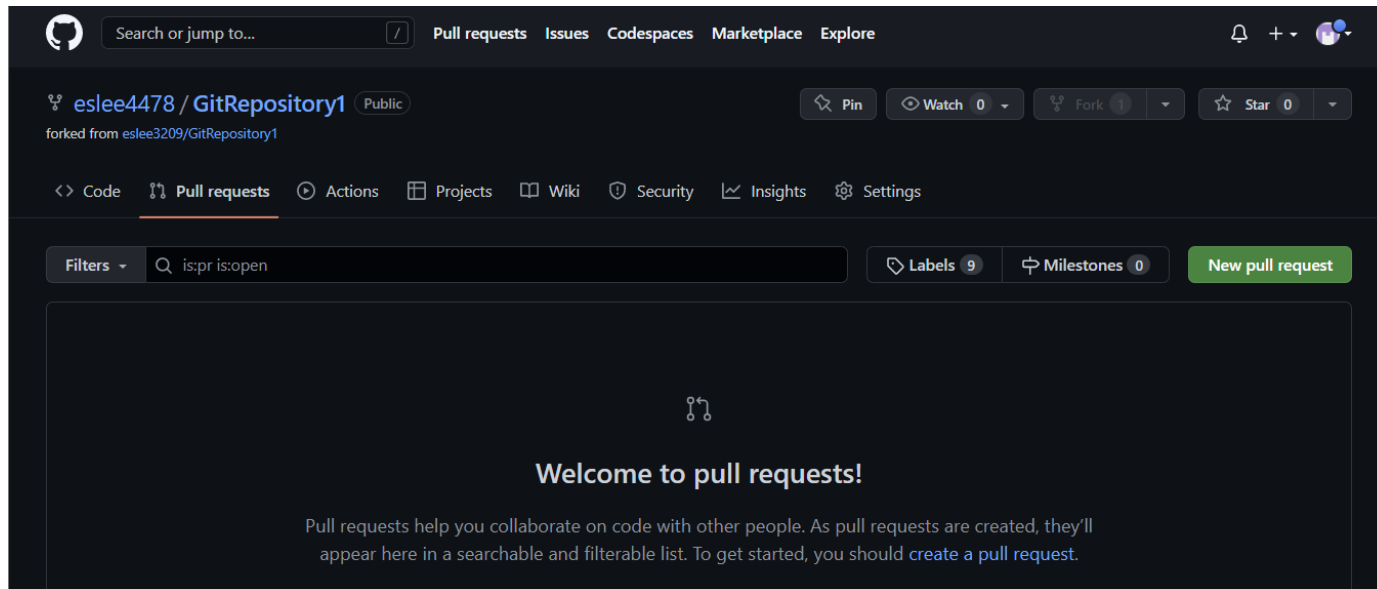
Fork 실습

- 이제, GitFile1.txt의 내용을 Contents4로 수정함
- 그리고 eslee4478 계정으로 커밋 및 push를 수행함
- eslee4478 계정의 GitRepository1 원격저장소에도 커밋이 잘 반영된 것을 알 수 있음



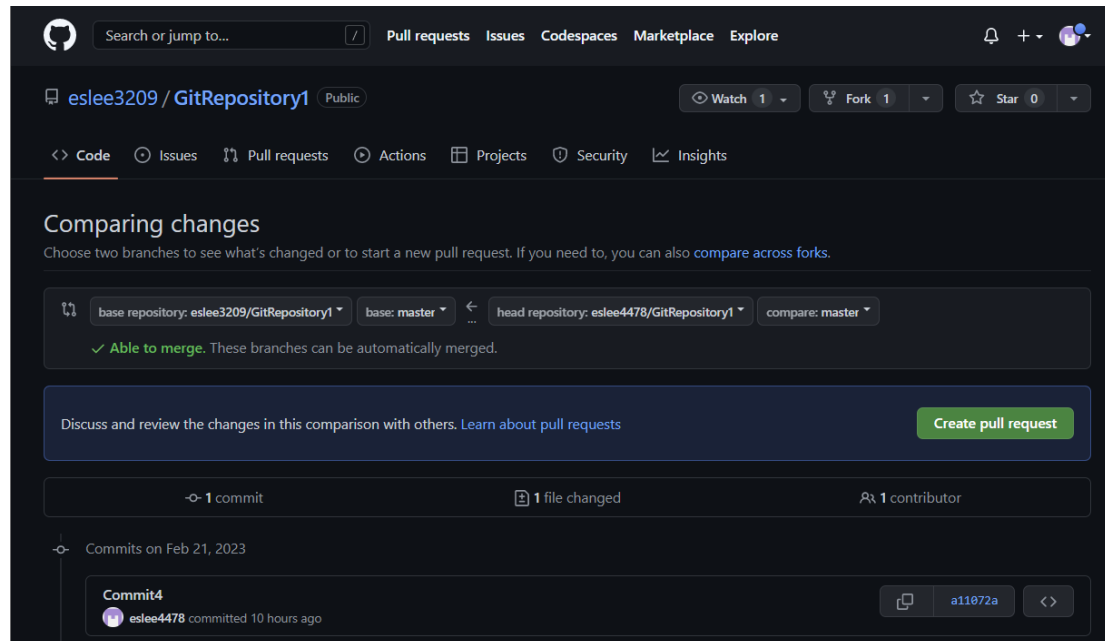
Pull Requests 실습 (Fork)

- 원본저장소와 다른 계정(eslee4478)으로 GitHub에 로그인함
- 최근 푸시한 커밋(Commit4)가 있는데 이 변경사항을 원본저장소 소유주(eslee3209)에게 알려주고 병합을 요청해야함
- [pull requests] 탭에서 [New pull request]를 클릭함



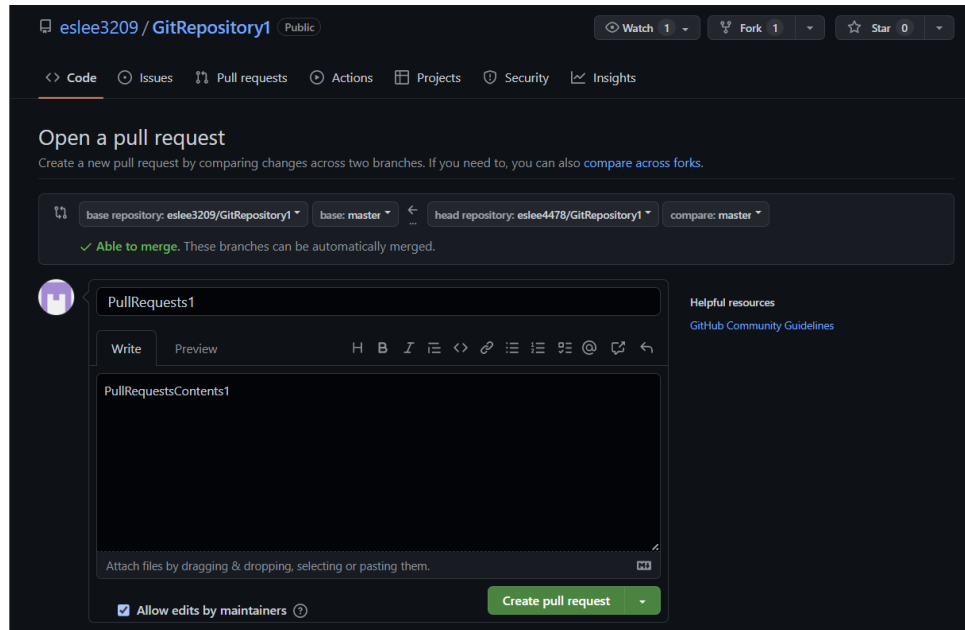
Pull Requests 실습 (Fork)

- [head repository]에 내가 포크한 원격저장소가, [base repository]에 원본저장소가 보여지면 성공임. head의 변경사항을 base에 합치려는 상황임
- Able to merge라고 쓰여진 초록색 텍스트는 충돌 없이 바로 병합 가능하다는 메시지임



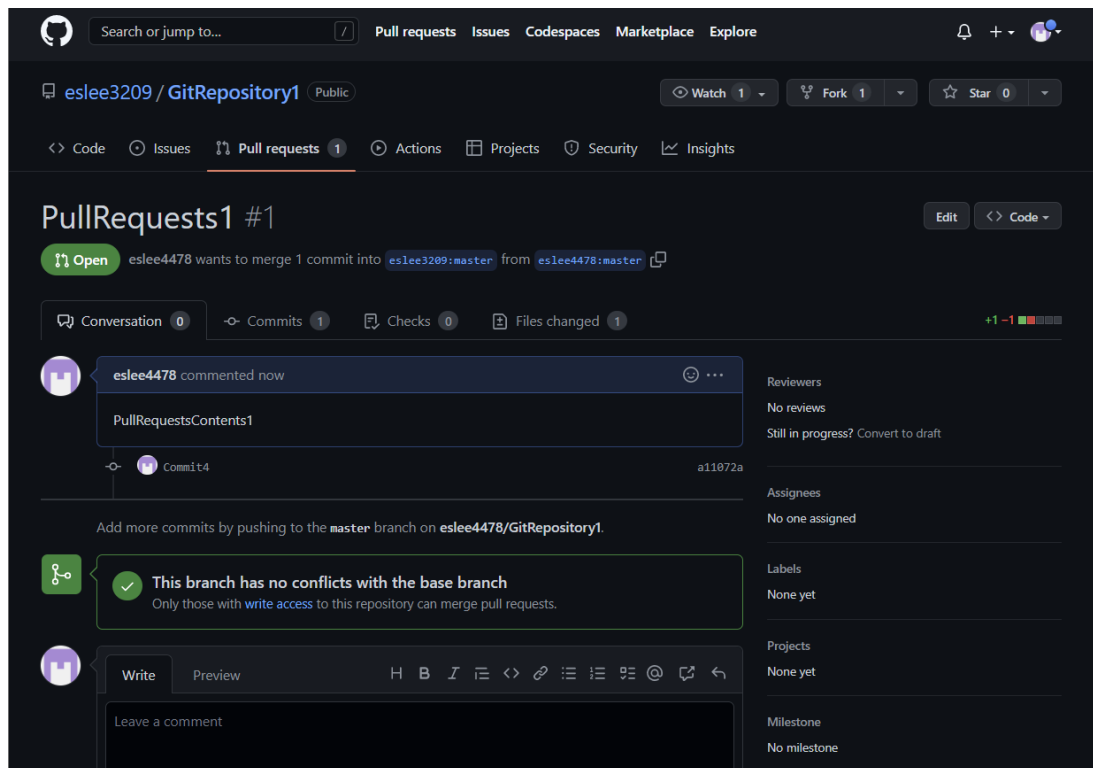
Pull Requests 실습 (Fork)

- 이제 [Create pull requests]를 클릭함
- pull requests에 대한 설명을 적음
 - 예) pull requests 이름: PullRequests1
 - pull requests 내용: PullRequestsContents1
- 만약 실제 디자인이나 UI에 변경이 일어났다면 스크린샷을 찍어 첨부하는 것도 좋음



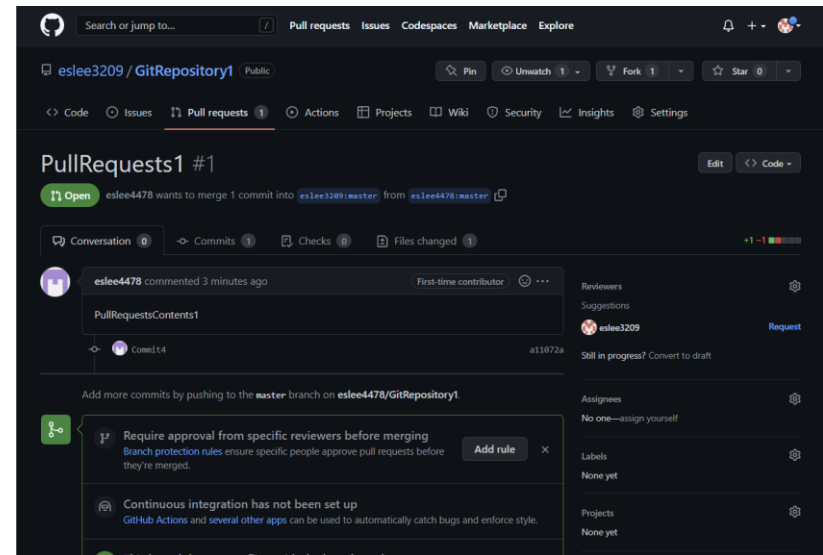
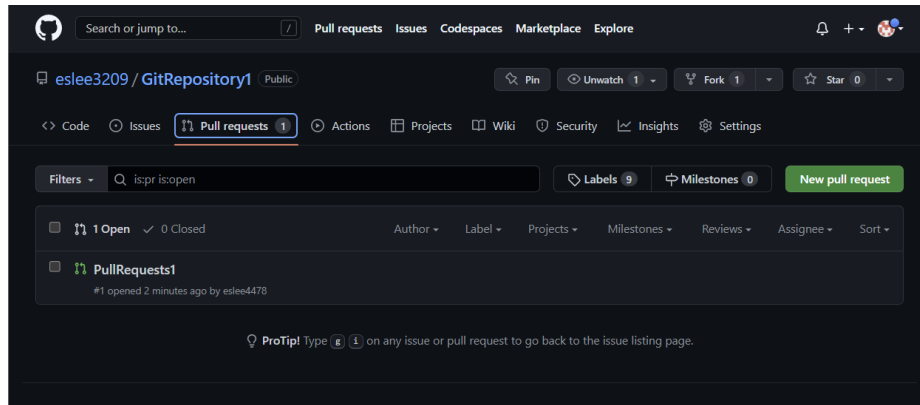
Pull Requests 실습 (Fork)

- [Create pull requests]를 클릭함
- 그러면 pull request가 만들어지고 이제 원본저장소 소유주(eslee3209)가 pull request를 승인하고 병합을 하기만 하면 됨



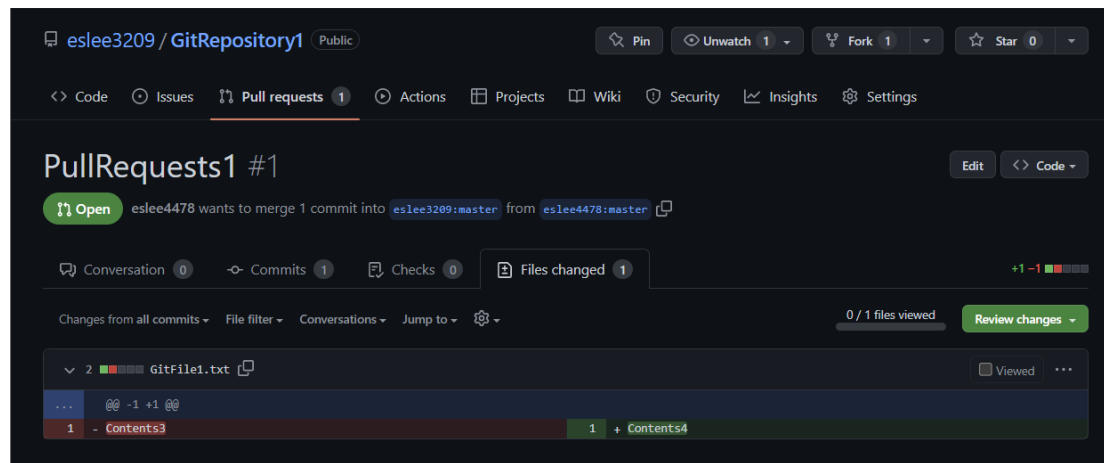
Pull Requests 실습 (Fork)

- 이제 원본저장소 주인 계정으로 GitHub 로그인하여 pull request를 확인함. pull requests 탭에 들어가면 뜬 것을 볼 수 있음
 - 예) eslee3209 계정으로 로그인하여 GitRepository1 원격저장소에 들어감
- 뜬 문구(PullRequests1)를 클릭하면 내용을 볼 수 있음



Pull Requests 실습 (Fork)

- [File changed] 탭을 클릭하면 어떤 새로운 코드가 이 pull request에 담겨있는지 확인할 수 있음
- 변경된 코드의 왼쪽 [+] 버튼을 클릭하면 코드 라인별 댓글을 달 수도 있음
- 여기에 수정사항 제안하거나 질문할 수 있음



Pull Requests 실습 (Fork)

- 코드 리뷰를 끝내고 우측 상단 [Review changes]을 클릭하면 [Write] 창이 열림
- 그냥 댓글만 달고 싶으면 [Comment]를, 댓글을 달고 코드가 좋아 바로 병합해도 될 것 같으면 [Approve]를, 수정을 요청하고 싶으면 [Request changes]을 선택함
- 본 실습에서는 승인한다고 함. [Approve]를 선택하고 [Submit review]를 클릭함

Finish your review

Write Preview

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

☐ Comment
Submit general feedback without explicit approval.

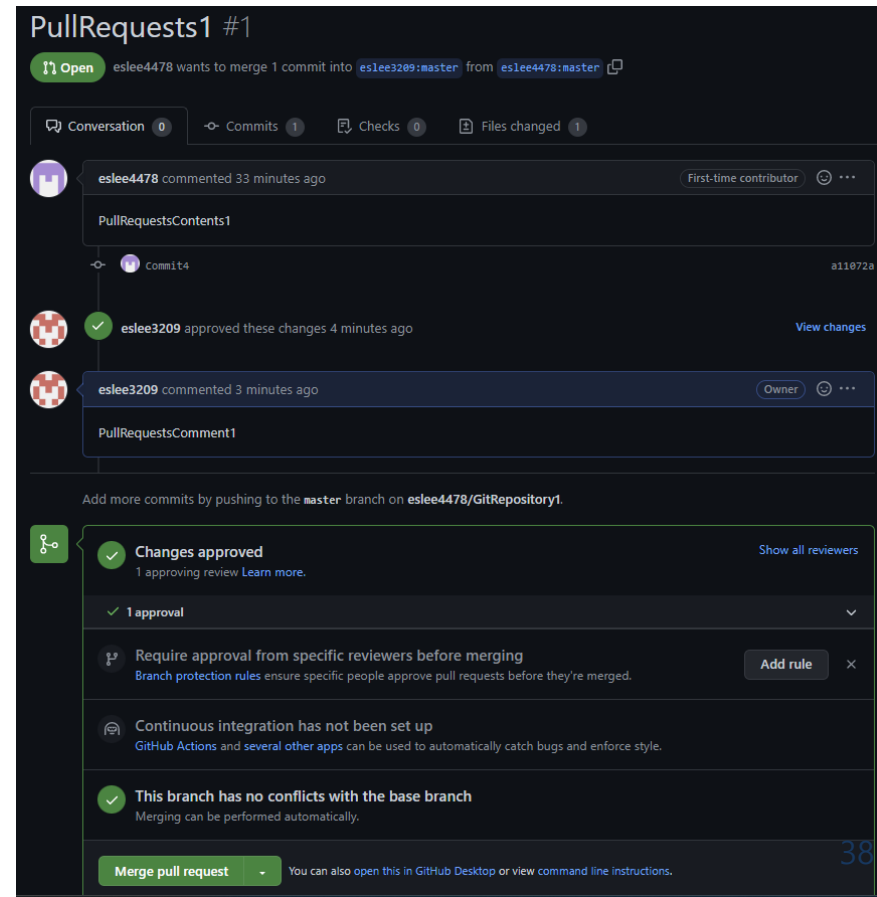
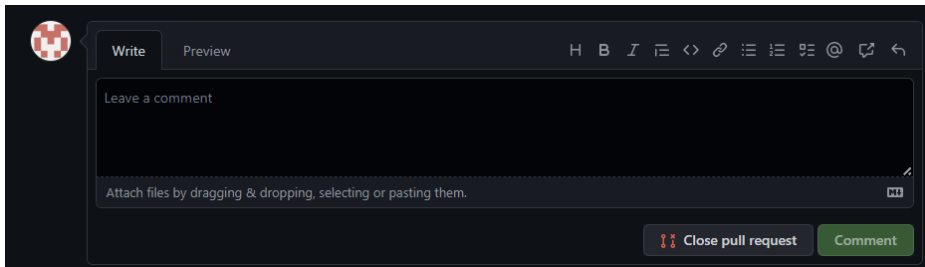
☒ Approve
Submit feedback and approve merging these changes.

☐ Request changes
Submit feedback that must be addressed before merging.

Submit review

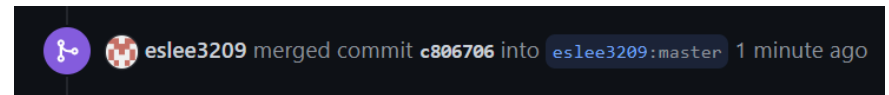
Pull Requests 실습 (Fork)

- 아래 Write라고 되어있는 부분에서는 코멘트를 남길 수 있음
 - 예) eslee3209 (원본저장소 주인) 계정으로 코멘트를 "PullRequestsComment1"이라고 남김



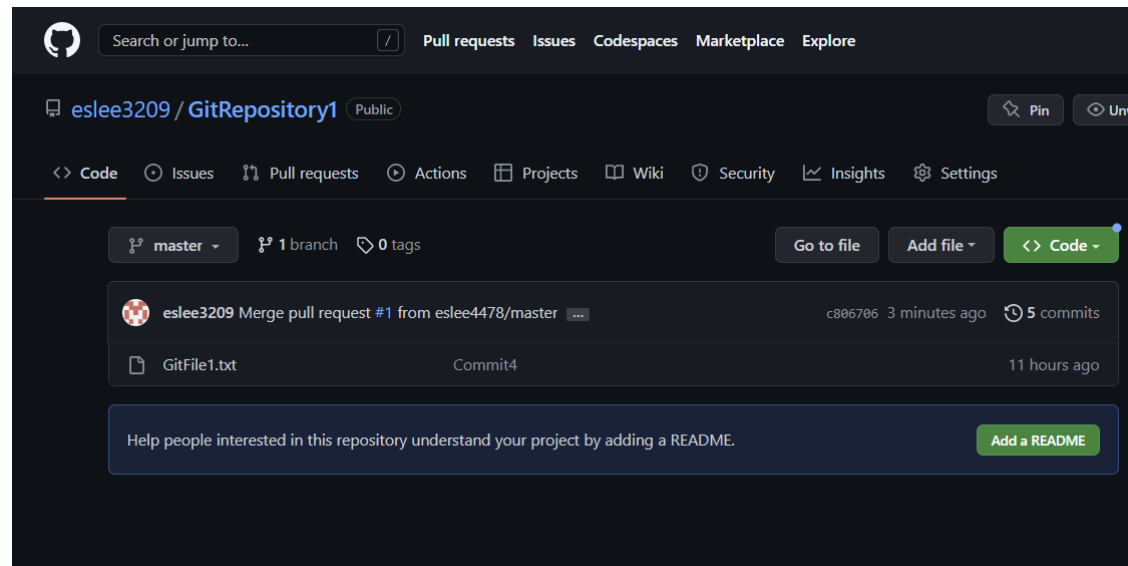
Pull Requests 실습 (Fork)

- 병합은 원본저장소 주인만 할 수 있음
- 이제 [Merge pull request] 버튼을 눌러 pull request를 병합함. [Confirm merge]를 클릭함
- 아래 그림과 같이 "Merged"라고 뜨고 그 아래에 "...merged commit ..."이라고 뜨면 잘 병합된 것임



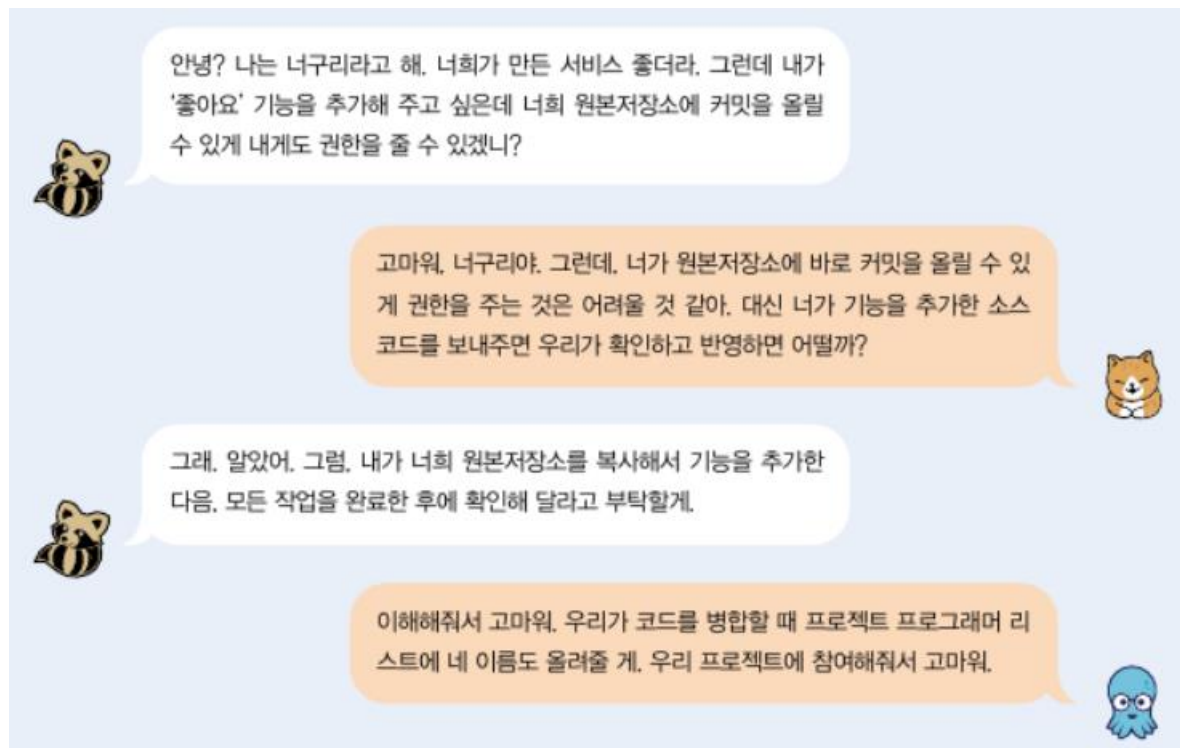
Pull Requests 실습 (Fork)

- 원본저장소 GitRepository1에서 확인하면 pull request를 요청했던 Commit4가 잘 들어간 것을 알 수 있음
- GitFile1.txt의 내용은 "Contents4"로 잘 바뀌어있음



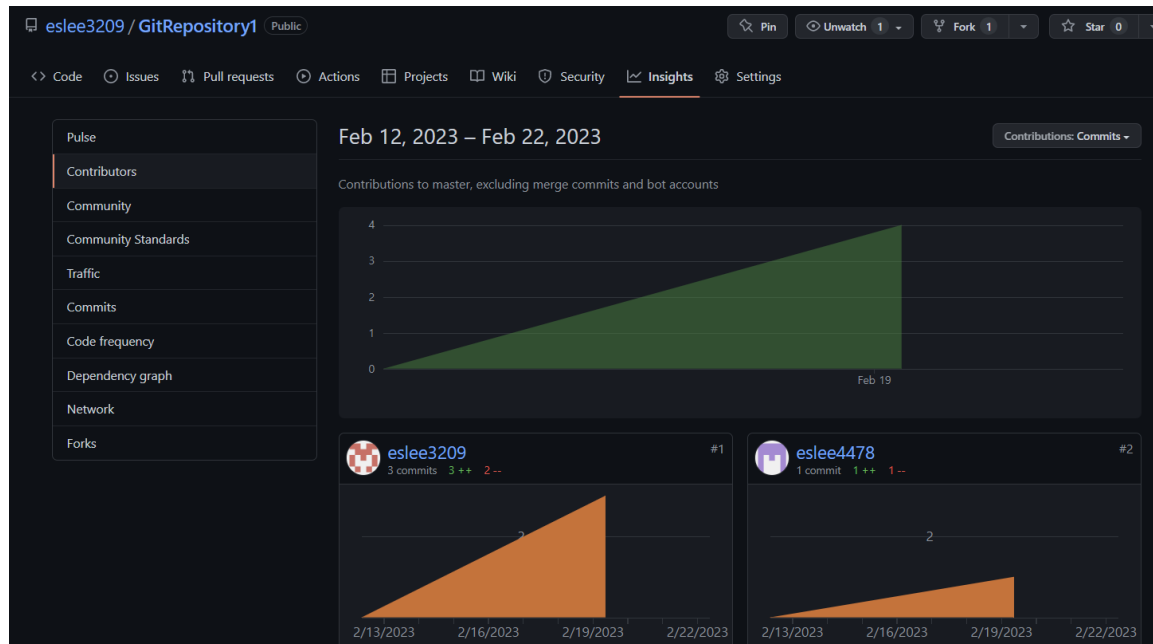
Contributor

- GitHub에 공개되어있는 어떤 저장소(GitRepository1)를 보고 관심을 갖고 기능을 추가하고 싶은 경우 contributor로 참여할 수 있음



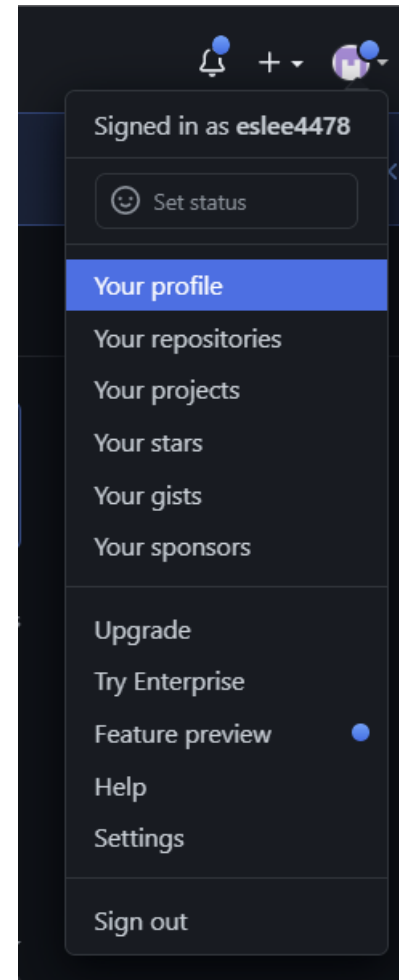
Contributor

- 원본저장소의 contributor를 조회하는 경우?
- GitHub 홈페이지에서 Insights>Contributors에 들어가면 주인 계정(eslee3209)와 외부 계정(eslee4478)이 contributor에 포함된 것을 확인할 수 있음



Contribution 조회

- 자신이 contribution한 것을 조회하는 방법?
- 자신의 계정으로 GitHub에 로그인함
 - 예) eslee4478
- 우측 상단의 역삼각형 버튼을 통해 [Your profile]을 클릭함
- 프로필 페이지가 열리면 [Customize your pins] 텍스트를 클릭함
- 여기서 내 프로필 첫 페이지에 노출되는 원본저장소 목록을 지정할 수 있음.
[eslee3209/GitRepository1]을 선택하고
[Save pins] 버튼을 클릭함



Contribution 조회

- Contribution 활동 내역이 나옴

