

오픈소스sw의 이해

Lecture #6: Collaboration with GitHub

Software Engineering laboratory
Yeungnam University



■ GitHub의 역할

● SW개발자의 SNS

- 동료 개발자, 세계적으로 유명한 개발자, 친해지고 싶은 개발자를 팔로우하여 관계를 형성할 수 있음.
- Instagram, facebook 등에 사진, 동영상을 업로드할 수 있는 것처럼 깃허브에서도 깃 저장소를 만들고 소스 코드를 업로드할 수 있음.
- 마음에 드는 저장소에 '좋아요'를 누를 수도 있음.

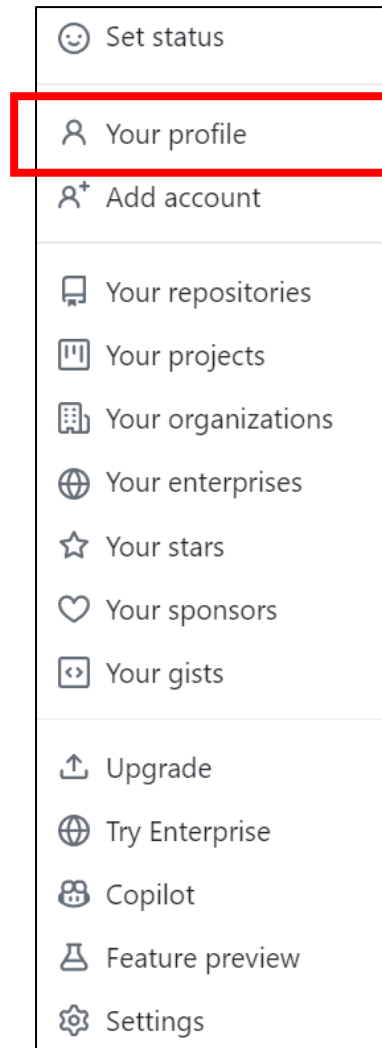
● 원격 저장소 (remote repository) 호스팅 서비스

- 개발자간의 협업 가능.

SW개발자 SNS로서의 GitHub

■ GitHub 사용자 프로필 확인

- 우측 상단의 동그란 아이콘 클릭.
- Your profile 클릭.
- 프로필, 깃 저장소, 활동 내역 등 확인 가능.



SW개발자 SNS로서의 GitHub

■ 다른 사용자의 GitHub URL 접속하기

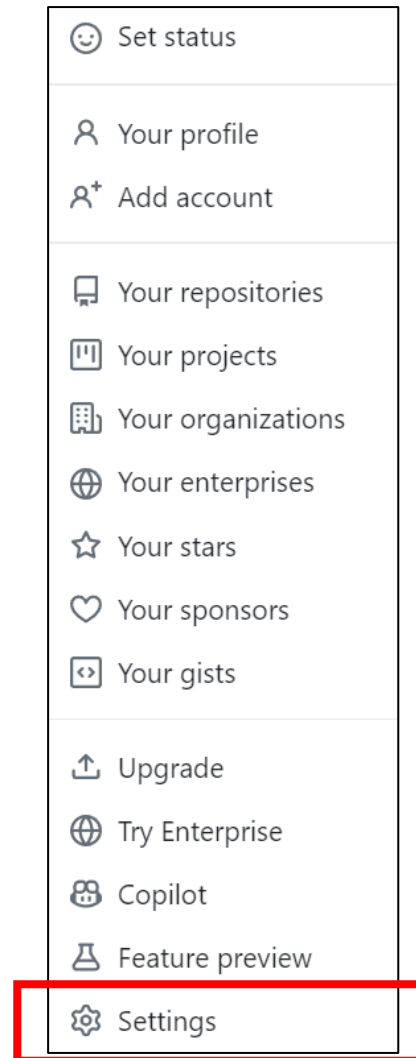
- <https://github.com/계정 이름> 형식으로 되어 있음.
- 위와 같은 URL을 통해 특정 사용자의 프로필을 조회할 수 있음.
- 특정 사용자 프로필 페이지에서 Follow 가능.
 - Follow 한 사용자의 활동 내역 확인 가능.

- 세계적으로 유명한, 이른바 스타 개발자의 프로필을 한번 살펴보자.
 - 깃을 창시한 Linus Torvalds의 계정
 - <https://github.com/torvalds>
 - Linus Torvalds의 깃허브 프로필과 활동 내역 확인 가능.

SW개발자 SNS로서의 GitHub

■ 개인 계정 정보 꾸미기

- 깃허브 첫 페이지에서 우측 상단의 동그란 아이콘을 클릭하고 Settings
- 좌측 메뉴에서 Public profile 클릭
 - 이름, 개인 정보, SNS 계정 등을 작성하고 프로필 사진도 업로드 가능.

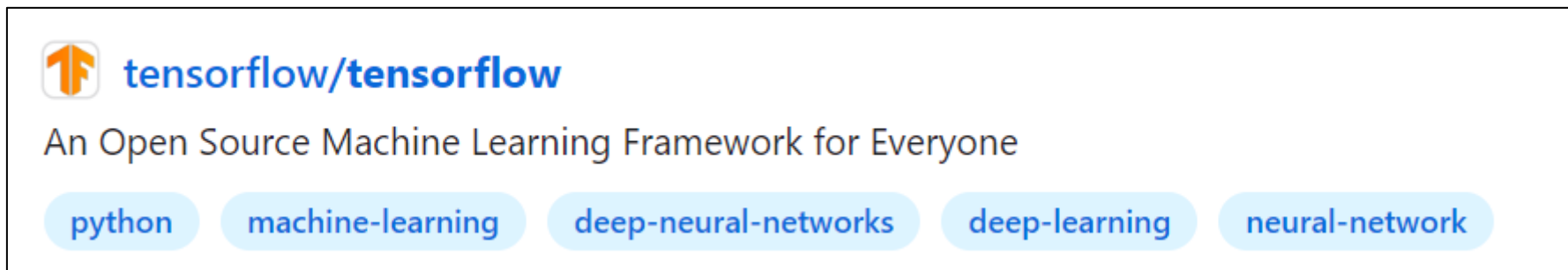


SW개발자 SNS로서의 GitHub

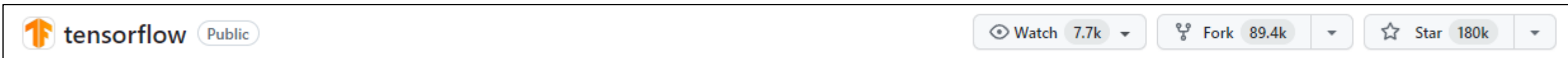
■ GitHub에서 공유되고 있는 유명 repository 탐방

● Tensorflow 방문해보기

- Tensorflow는 오픈 소스로 깃허브에 모든 소스 코드가 공개되어 있음.
- 상단 검색 창에 tensorflow를 입력하고 검색 후 다음 제목을 클릭하여 입장.

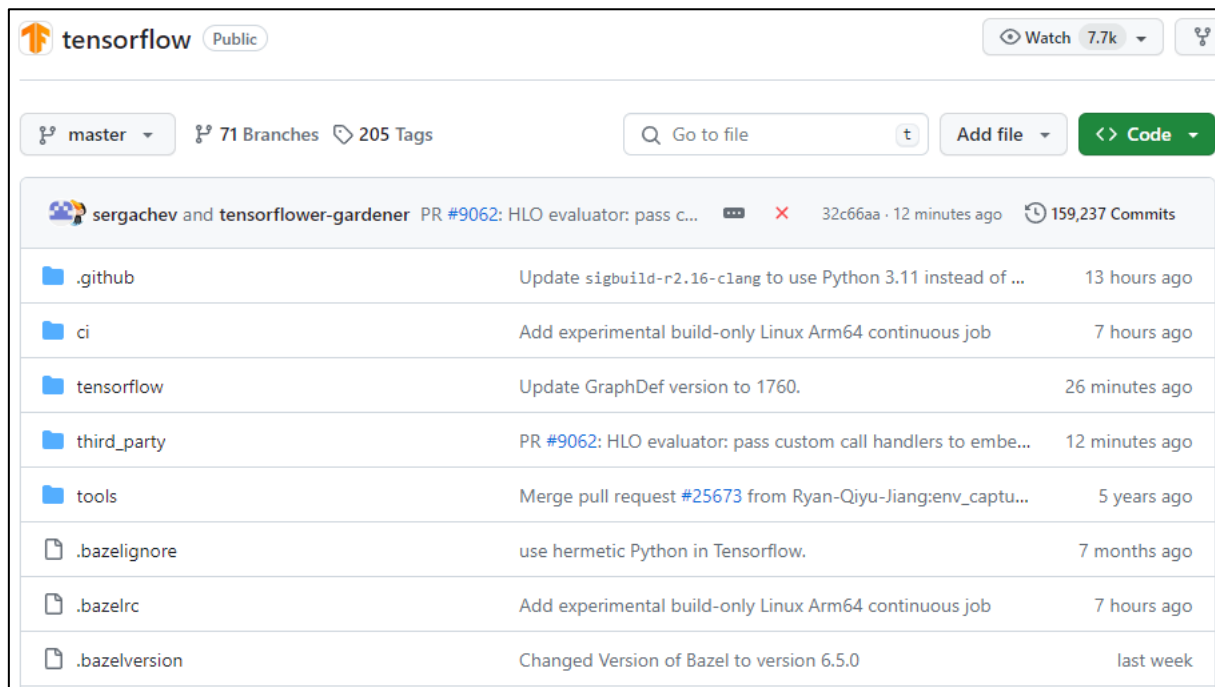


- Tensorflow 저장소 제목 우측 부분에 Star: SNS의 '좋아요'와 같은 역할.
 - 마음에 드는 오픈 소스 프로젝트에 Star를 클릭하여 호감을 표시할 수 있음.



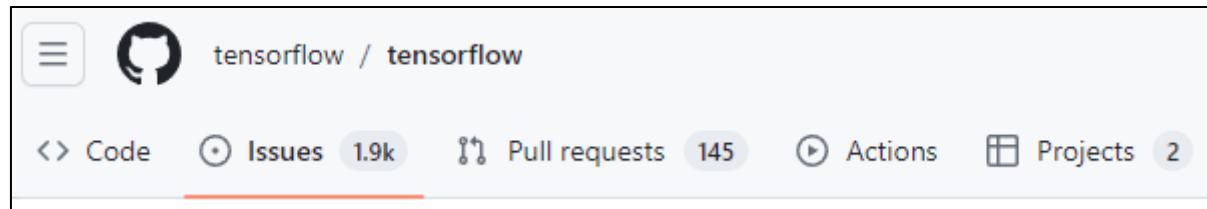
SW개발자 SNS로서의 GitHub

- GitHub에서 공유되고 있는 유명 repository 탐방
 - Tensorflow의 개발 진행 상황 확인
 - 현재 Commit 수 확인
 - 관련 파일 또는 폴더가 마지막으로 변경된 시점 확인
 - 관련 파일 또는 폴더가 마지막으로 변경된 시점의 커밋 메시지 확인



SW개발자 SNS로서의 GitHub

- GitHub에서 공유되고 있는 유명 repository 탐방
 - Issues 확인
 - 오류를 제보하거나, 다양한 새로운 기능을 제안.
 - 이렇게 이슈를 남기면 텐서플로 개발자를 포함한 세계 곳곳의 개발자들이 댓글을 달기도 하고, 이슈를 해결해 주기도 함.



SW개발자 SNS로서의 GitHub

■ Summary

- 여러분만의 프로필을 만들 수 있고, 다른 뛰어난 개발자들을 팔로우할 수 있으며, 여러분의 깃 저장소를 만들고, 다른 저장소에 이슈를 남길 수 있음.
- 다른 뛰어난 개발자들을 팔로우하고 그들의 활동 내역을 확인 가능.
- 다른 저장소(repository)들을 탐방해보고 프로젝트 참여 가능.

원격 저장소 호스팅 서비스로서의 GitHub

■ 저장소 (Repository)

● 로컬 저장소 (Local repository)

- 사용자 개인 컴퓨터 속에만 존재하는 저장소.

● 원격 저장소 (Remote repository)

- 사용자 개인 컴퓨터가 아닌 외부에 있는 저장소.
- 인터넷 세상 어딘가에 있는 다른 컴퓨터 속의 저장소를 의미.
- GitHub의 원격 저장소는 GitHub가 관리하는 컴퓨터 속의 저장소를 의미.

원격 저장소 호스팅 서비스로서의 GitHub

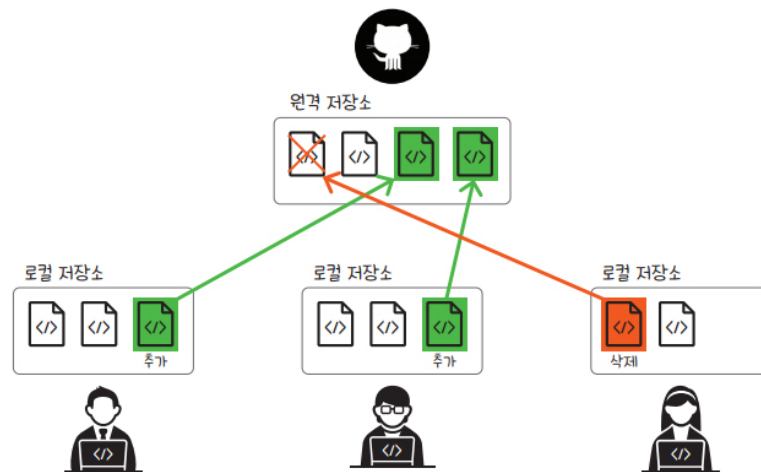
■ 원격 저장소를 활용하는 이유

● 백업 (Backup)

- 로컬 저장소의 프로젝트를 원격 저장소에 백업해두고, 언제든지 원격 저장소에서 프로젝트를 내려받아 사용할 수 있다면 여러분의 컴퓨터가 망가지거나 로컬 저장소가 삭제되어도 전혀 문제가 되지 않음.

● 협업 (Collaboration)

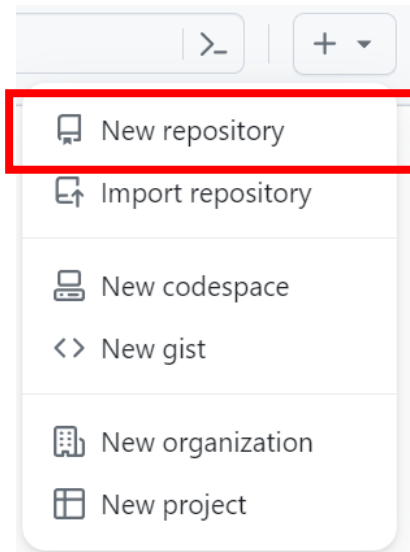
- 모든 개발자가 이해하는 공통 코드를 원격 저장소에 업로드하고, 이를 내려받아 새로운 커밋들을 추가한 뒤 이를 다시 원격 저장소에 추가하는 방식으로 작업한다면 로컬 저장소에서만 작업하는 것보다 훨씬 효율적으로 협업할 수 있음.



(실습) GitHub에서 원격 저장소 만들어보기

■ 원격 저장소 만들기

- 우측 상단의 +를 클릭한 뒤 New repository를 클릭.
 - Owner: 원격 저장소의 소유자
 - Repository name: 원격 저장소의 이름 (ex. test-repo)
 - Description: 생성할 원격 저장소에 대한 설명을 적는 공간 (옵션)
 - Public과 Private 중 하나를 선택
 - Public은 모두에게 공개된 저장소를 의미
 - Private은 모두에게 공개하지 않고 여러분(또는 여러분이 지정한 일부 사용자)만 볼 수 있는 저장소
 - "Add a README file" 선택
- 생성한 저장소 URL은 `https://github.com/계정 이름/저장소 이름 형식`
- Settings-Your profile로 이동해보면 생성한 저장소 확인 가능

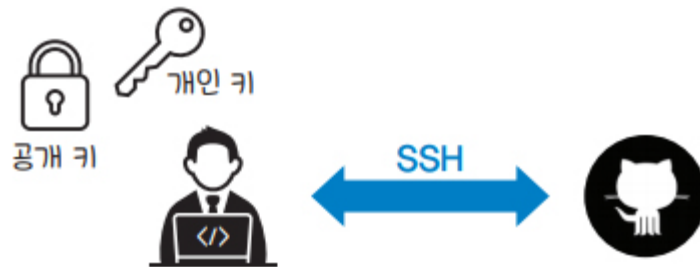


원격 저장소와 상호작용 하기 위한 연동

■ SourceTree와 GitHub 연동하기

- SSH(Secure Shell) 통신할 수 있도록 연동
 - SSH는 원격 호스트에 접속하기 위해 사용되는 보안 프로토콜
 - SourceTree와 GitHub가 서로 SSH 통신이 가능하도록 연동하면 사용자의 컴퓨터(로컬 저장소)와 깃허브(원격 저장소)는 서로 안전하게 정보를 주고받을 수 있음.
- SSH 통신하려면 먼저 여러분의 컴퓨터에서 키(key) 두 개를 생성
 - 공개 키(public key): 모두에게 공개된 키
 - 개인 키(private key): 사용자 개인만 알고 있어야 하는 키

• Key: 여기서는 암호, 또는 암호화된 문자열이라고 생각해도 좋음

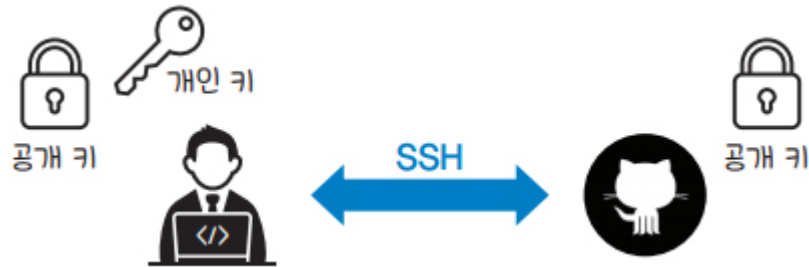


원격 저장소와 상호작용 하기 위한 연동

■ SourceTree와 GitHub 연동하기

● 통신 방법

- 공개 키와 개인 키를 생성한 뒤 통신하려는 대상(GitHub)에게 공개 키를 건네줌.
- SSH 통신은 암호화된 통신 방법이므로 여러분과 (공개 키를 전달받은) 깃허브 사이에 주고받는 대화는 암호화되어 전송.
Data 보안 *인증 과정*
- 다른 누군가가 여러분과 GitHub 사이에 주고받는 대화를 엿듣는다고 해도 전혀 이해할 수 없음.
- 다만, 여러분은 개인 키가 있어서 깃허브에 여러분임을 증명하고, 깃허브와 주고받는 내용을 이해할 수 있음.



원격 저장소와 상호작용 하기 위한 연동

■ (1) SSH key 생성하기

- SSH key는 ssh-keygen이라는 간단한 명령으로 생성할 수 있음.
- git bash 창을 열고 ssh-keygen을 입력하면 개인 키(id_rsa)를 저장할 경로를 선택하라는 문구가 나옴
- 기본 설정된 경로는 다음 붉은 박스 친 경로
- 아무것도 입력하지 않고 Enter 키를 눌러 해당 경로에 key를 지정

```
minchul@DESKTOP-9KULGUE MINGW64 ~
```

```
$ ssh-keygen
```

```
Generating public/private rsa key pair.
```

```
Enter file in which to save the key (/c:/Users/minchul/.ssh/id_rsa):
```

Enter 입력

← 개인 키 저장할 경로

원격 저장소와 상호작용 하기 위한 연동

■ (1) SSH key 생성하기

- 'Enter passphrase (empty for no passphrase)'가 출력
 - 여러분이 사용하려는 암호를 입력한 뒤 Enter를 누르거나, 암호를 사용하지 않으려면 아무것도 입력하지 않고 Enter를 누름
- 일단 아무것도 입력하지 않고 Enter
- 'Enter same passphrase again'이 출력
 - 암호 없이 사용하기로 했다면 그냥 Enter
 - 앞서 암호를 입력한 경우, 입력했던 암호를 한 번 더 입력한 후 Enter

```
minchul@DESKTOP-9KULGUE MINGW64 ~
```

```
$ ssh-keygen
```

```
Generating public/private rsa key pair.
```

```
Enter file in which to save the key (/c/Users/minchul/.ssh/id_rsa):
```

```
Created directory '/c/Users/minchul/.ssh'.
```

```
Enter passphrase (empty for no passphrase):
```

```
Enter same passphrase again: .....
```

Enter 입력

← 사용하려는 암호 입력

← 입력했던 암호 입력

원격 저장소와 상호작용 하기 위한 연동

■ (1) SSH key 생성하기

- SSH key 생성완료

- 개인 키는 C:\Users\사용자명\.ssh 경로에 저장된 id_rsa라는 파일.
- 공개 키는 같은 경로에 저장된 id_rsa.pub라는 파일임.

- 해당 경로로 이동해 2개의 키 확인해보기!

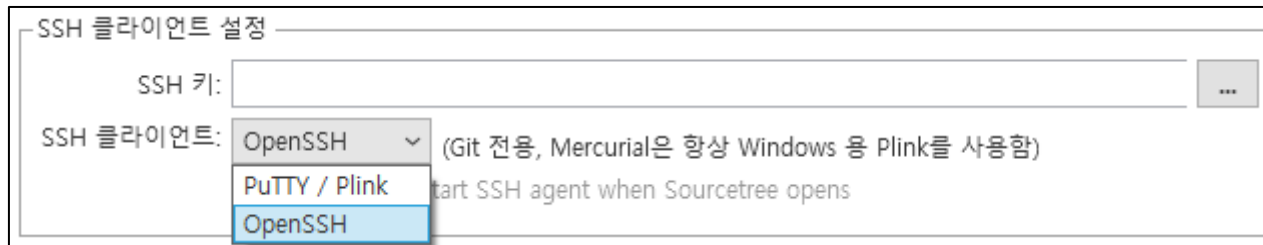
```
minchul@DESKTOP-9KULGUE MINGW64 ~  
$ ssh-keygen  
Generating public/private rsa key pair.  
Enter file in which to save the key (/c/Users/minchul/.ssh/id_rsa):  
Created directory '/c/Users/minchul/.ssh'.  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /c/Users/minchul/.ssh/id_rsa  
Your public key has been saved in /c/Users/minchul/.ssh/id_rsa.pub  
The key fingerprint is:  
SHA256:b8GJrBBFddcw55XaBho+P63BC3dpNshM5ZviYJED9To minchul@DESKTOP-9KULGUE
```

개인 키
공개 키
확인

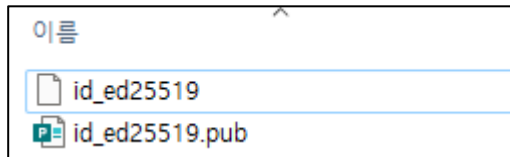
원격 저장소와 상호작용 하기 위한 연동

■ (2) SourceTree에서 생성한 private key 등록하기

- [도구]-[옵션]
- OpenSSH 선택 후 하단에 확인 버튼 클릭



- [도구]-[SSH 키 추가...] 클릭 후 개인 키 선택 후 열기 클릭

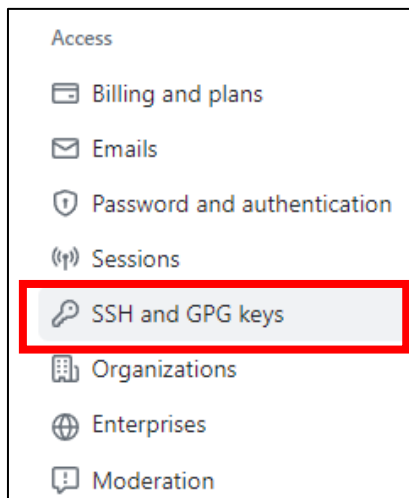


- [도구]-[옵션] 클릭하여 개인 키가 잘 등록되었는지 확인 가능

원격 저장소와 상호작용 하기 위한 연동

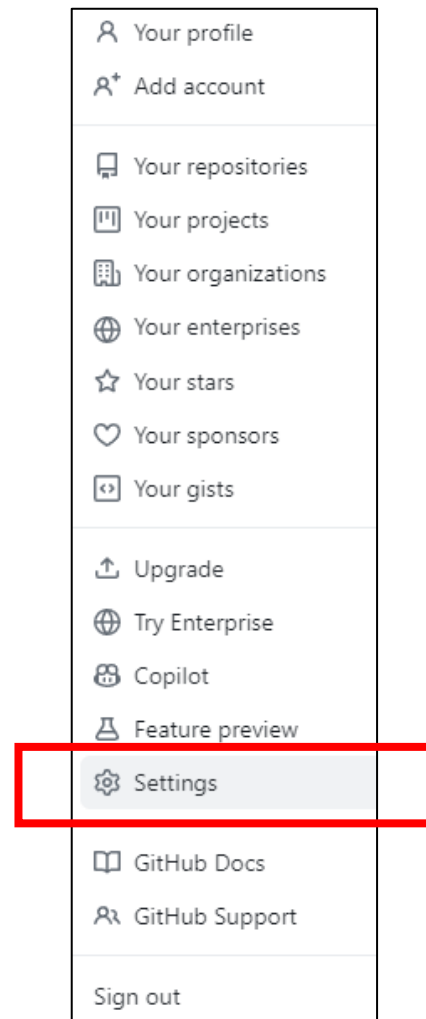
■ (3) GitHub에게 public key 건네주기

- GitHub에 접속한 뒤 프로필 이미지를 클릭해 Settings 선택
- 좌측 메뉴에서 SSH and GPG keys를 클릭



- New SSH key 클릭 후 공개 키 등록

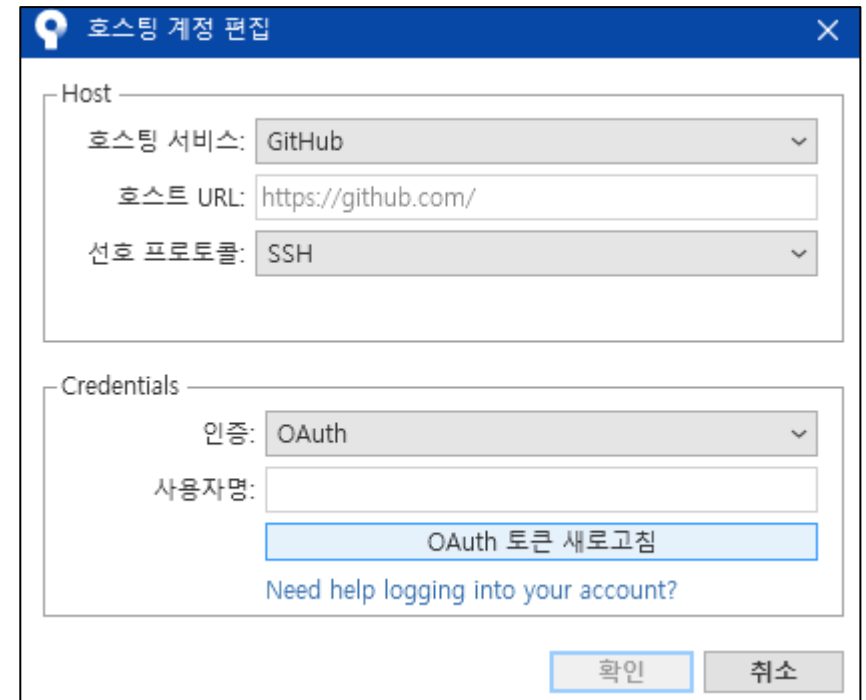
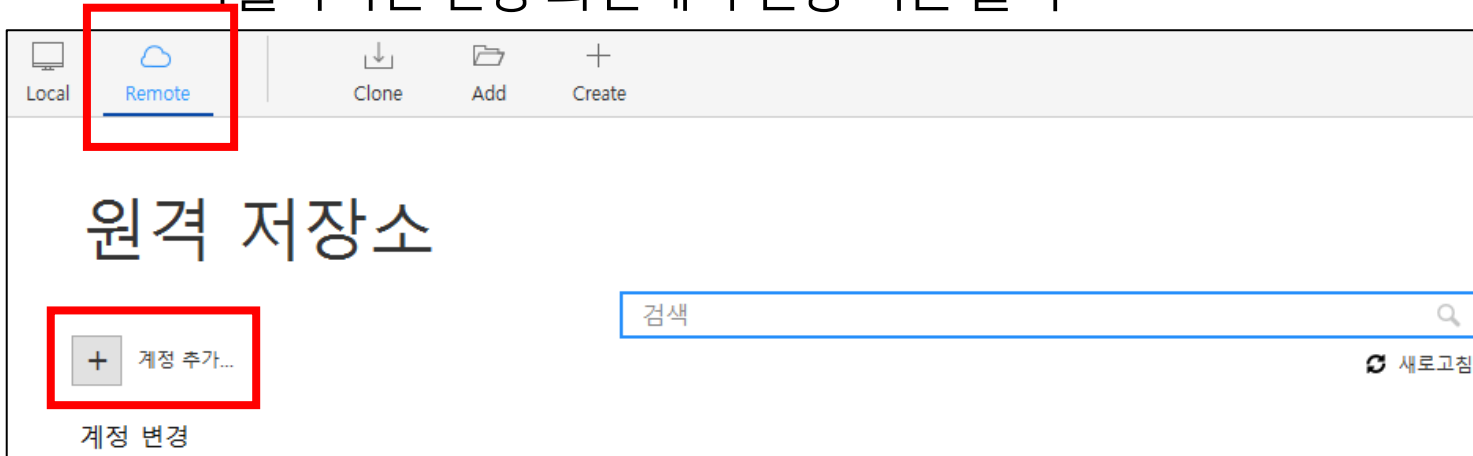
- Title: 키의 제목
- Key: 항목에는 공개 키 파일(***.pub) 안에 적힌 내용 복사 후 붙이기,
- Add SSH key 클릭



원격 저장소와 상호작용 하기 위한 연동

■ (4) SourceTree에서 Remote repository 등록하기

- SourceTree에서 방금 연결한 GitHub에 있는 remote repository를 등록
 - 새 탭을 열어 remote 클릭 - 계정 추가 클릭
 - 호스팅 서비스: GitHub
 - 선호 프로토콜: SSH
 - OAuth 토큰 새로고침 클릭
 - 아틀라시안 인증 화면에서 인증 버튼 클릭



원격 저장소와 상호작용 하기 위한 연동

■ (4) SourceTree에서 Remote repository 등록하기

● 인증 성공 체크 확인

- GitHub 계정 인증 성공
- SourceTree에서 여러분의 GitHub 계정이 보인다면 성공
 - 만든 원격 저장소 ((예)test-repo)도 볼 수 있음
- 혹시 원격 저장소가 뜨지 않는다면 “새로고침”을 클릭

The screenshot shows the '호스팅 계정 편집' (Edit Hosting Account) dialog box in SourceTree. It is configured for GitHub with the following details:

- Host:**
 - 호스팅 서비스: GitHub
 - 호스트 URL: https://github.com/
 - 선택 프로토콜: SSH
- Credentials:**
 - 인증: OAuth
 - 사용자명: ysseo29
 - Buttons: OAuth 토큰 새로고침, Need help logging into your account?

At the bottom left, a green checkmark icon is next to the text '인증 성공' (Authentication Successful), which is highlighted by a red rectangular box. At the bottom right, there are two buttons: '확인' (OK) and '취소' (Cancel).

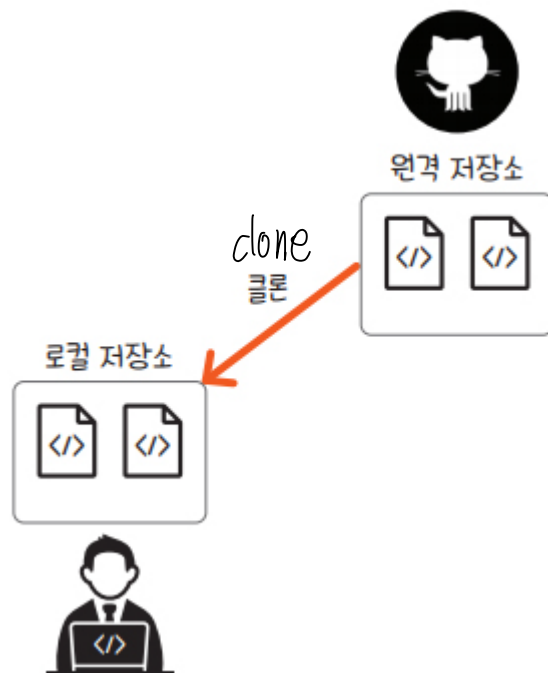
원격 저장소와 상호작용 하기

- (1) clone (클론): 원격 저장소 복제하기
- (2) push (푸시): 원격 저장소에 밀어넣기
- (3) fetch (패치): 원격 저장소를 일단 가져만 오기
- (4) pull (풀): 원격 저장소를 가져와서 합치기 *fetch + merge*

원격 저장소와 상호작용 하기

■ (1) clone (클론)

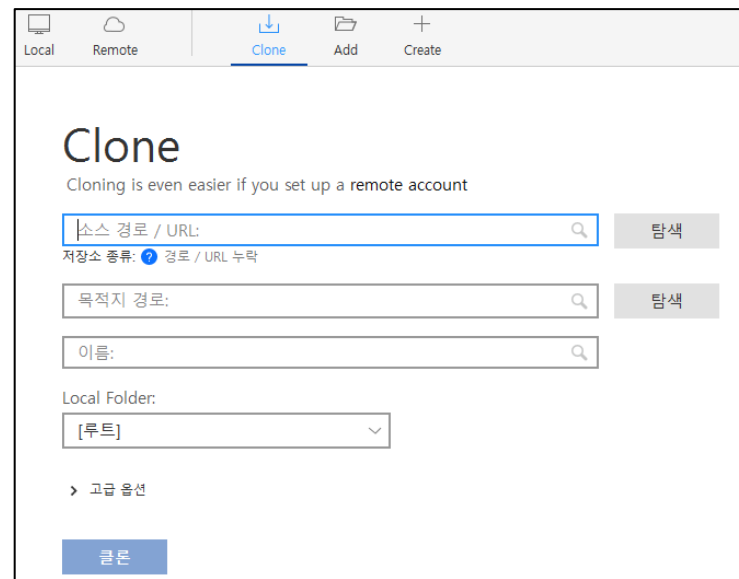
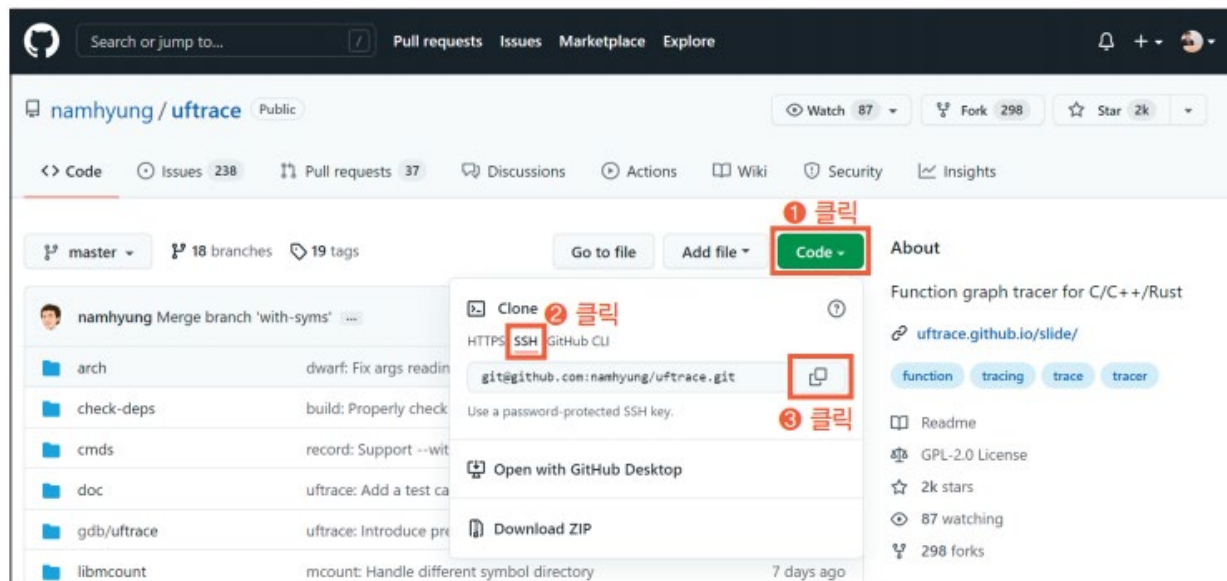
- 원격 저장소 복제하기
- GitHub상에 존재하는 원격 저장소를 여러분의 컴퓨터, 즉 로컬로 복사하여 가져오는 방법
 - 사용자가 직접 만든 원격 저장소 및 GitHub상에 공개된 모든 원격 저장소를 클론하여 가져올 수 있음.



(실습) clone

■ (1) 임의의 Remote repository clone 하기

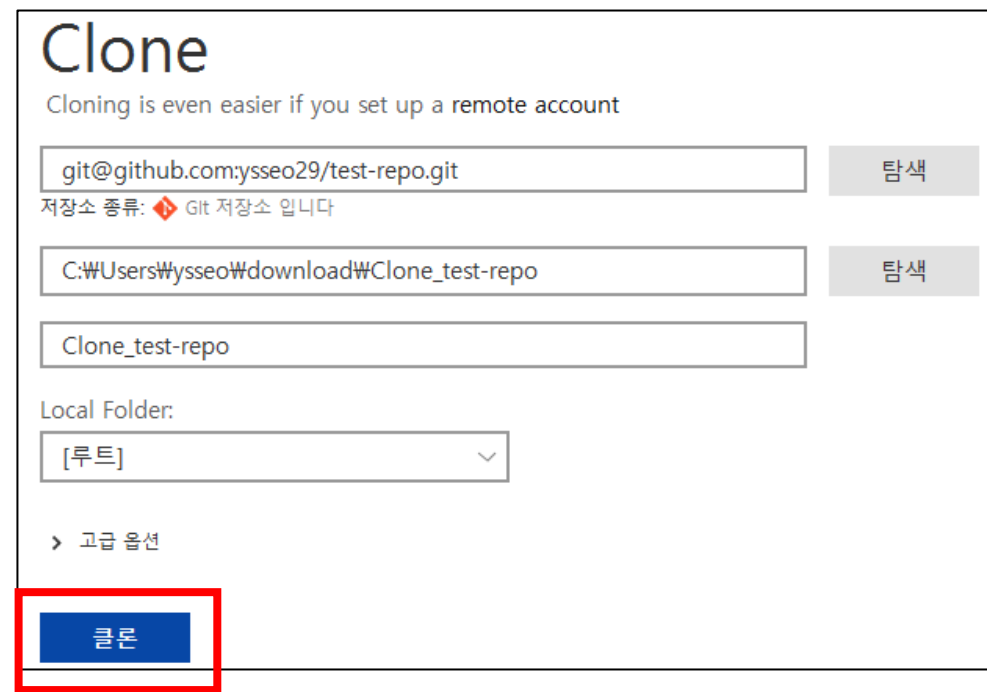
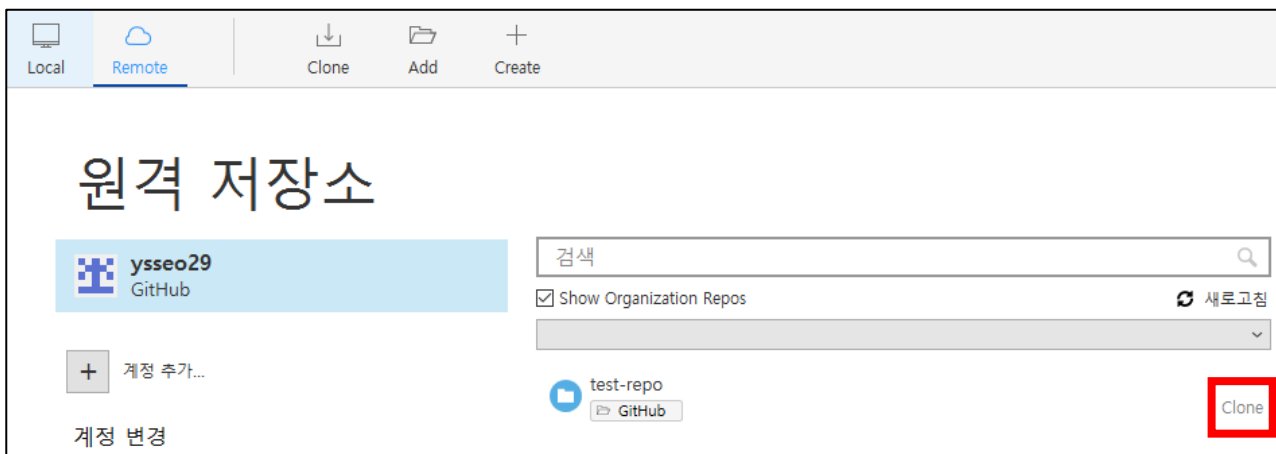
- Remote repository의 SSH 경로 카피
- SourceTree에서 메뉴 상단에 Clone 클릭 후 주소 붙여넣기
 - 목적지 경로: 사용자 컴퓨터 속 클론한 경로를 기입
 - 이름: 복사할 원격 저장소의 이름



(실습) clone

■ (2) 본인의 Remote repository clone 하기

- 새로운 탭 생성 후 Remote 클릭
 - 본인이 생성한 원격 저장소 목록이 표시
 - 표시되지 않을 경우 새로고침 클릭
- 원격 저장소 목록에서 옆에 있는 Clone을 클릭



(실습) clone

■ (주의) Git 주소 연결시 오류

- 방화벽 문제 발생 가능
 - 오류메시지 ssh: connect to host github.com port 22: Connection refused
- HTTPS를 통해서 SSH 연결을 활성화하는 방법
 - SSH 설정을 재정의하여 모든 연결이 해당 서버 및 포트를 통해 실행되도록 함.

- C:\W[사용자]\W사용자ID\W.ssh 에 config 파일 생성하여 다음처럼 편집

```
Host github.com
Hostname ssh.github.com
Port 443
```

(실습) clone

■ (2) 본인의 Remote repository clone 하기

- Clone이 완료되면 다음과 같은 3개의 브랜치 생성

- ^{github}main : ^{git}master 브랜치와 같음 (git에서는 master, GitHub에서는 main)

- origin/main : 원격 저장소 origin의 기본 브랜치

- origin/HEAD : 원격 저장소 origin의 HEAD

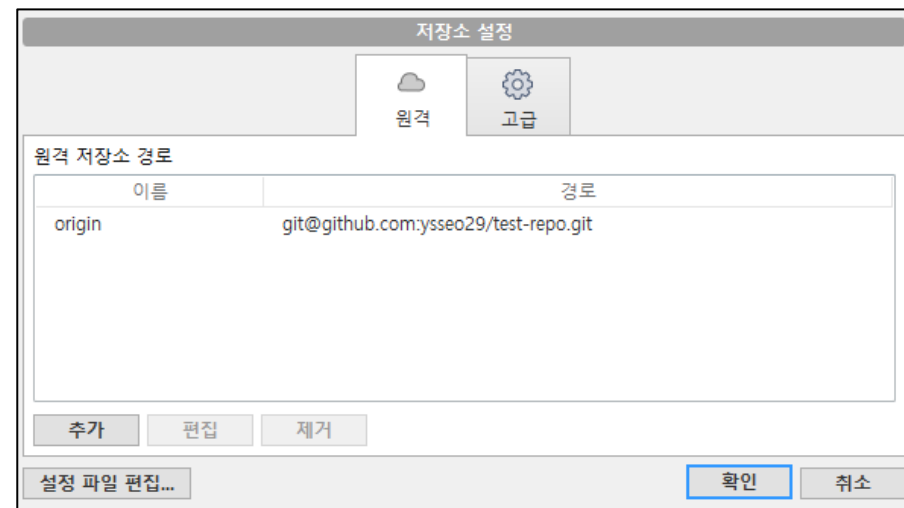
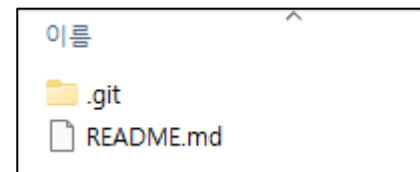
- 우측 상단에 [탐색기] 클릭하여 clone한 파일 확인

- 우측 상단에 [설정]을 클릭

- 원격 저장소를 지칭하는 이름으로 origin 사용 확인.

- 원격 저장소를 지칭할 때 매번 경로(git@github.com:...)를 사용하는 것은 번거로우니 단순히 origin이라고 부르기로 한 것임.

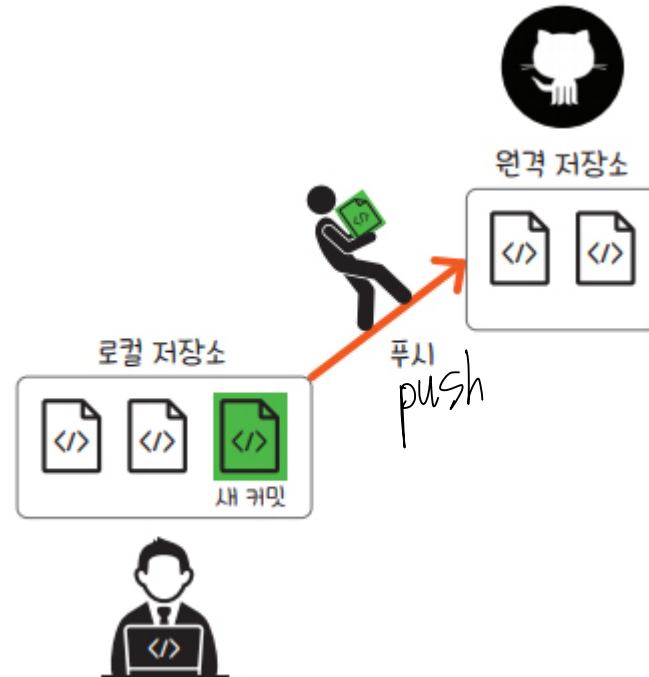
- origin은 원격 저장소에 붙은 이름일 뿐이므로 언제든지 수정 가능.



원격 저장소와 상호작용 하기

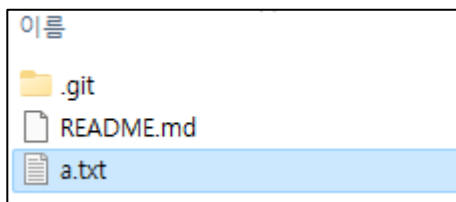
■ (2) push (푸시)

- 원격 저장소에 로컬 저장소의 변경 사항을 밀어넣는 것(push)을 의미
 - 로컬 저장소(로컬 분기)에서 만든 커밋을 원격 저장소로 푸시



(실습) push

- 로컬에서 커밋을 추가한 뒤 이를 원격 저장소에 푸시하기
 - 로컬 저장소 (ex. test-repo)에 문자 A가 담긴 a.txt 파일을 추가



- 위 변경사항을 commit 하기
 - 커밋메시지: add a.txt



(실습) push

■ 로컬에서 커밋을 추가한 뒤 이를 원격 저장소에 푸시하기

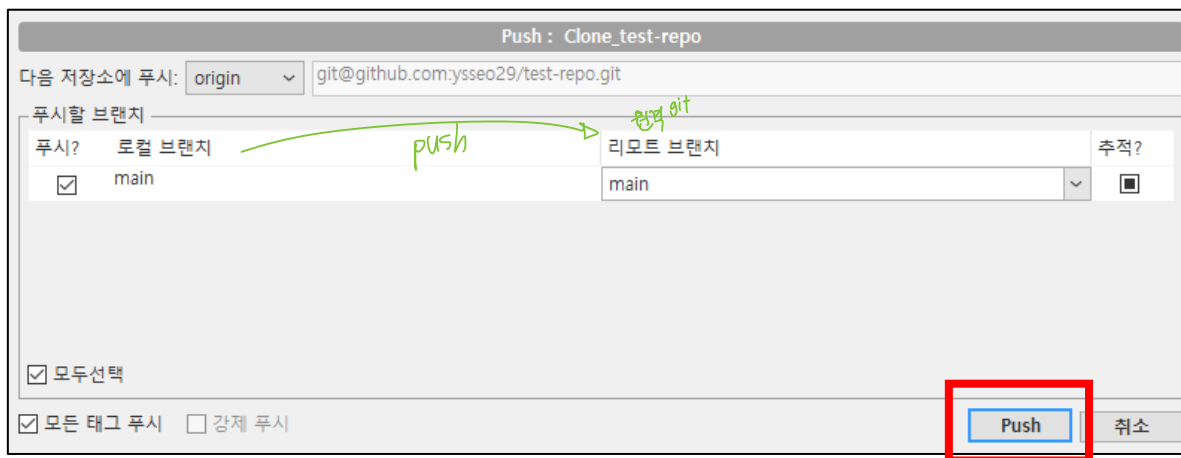
● 상단에 Push 클릭

- A.txt 파일을 원격 저장소에 추가



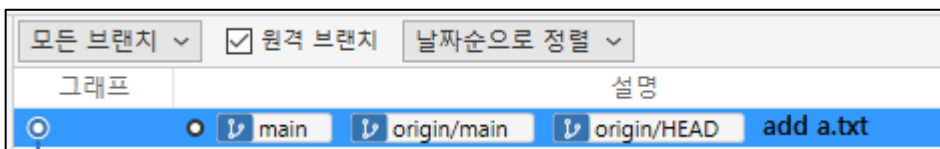
● 팝업창에서 Push 클릭

- 로컬 저장소의 main 브랜치를 원격(리모트) 저장소의 main 브랜치로 푸시한다는 것을 의미



(실습) push

- 로컬에서 커밋을 추가한 뒤 이를 원격 저장소에 푸시하기
 - Push 완료 확인



- 원격 저장소인 GitHub에도 push 결과 확인해보기
 - a.txt 파일이 잘 추가됐을 뿐 아니라 커밋 수가 늘었음.
 - 박스 친 부분을 클릭해 보면 커밋의 세부내역을 확인 가능.



(실습) push

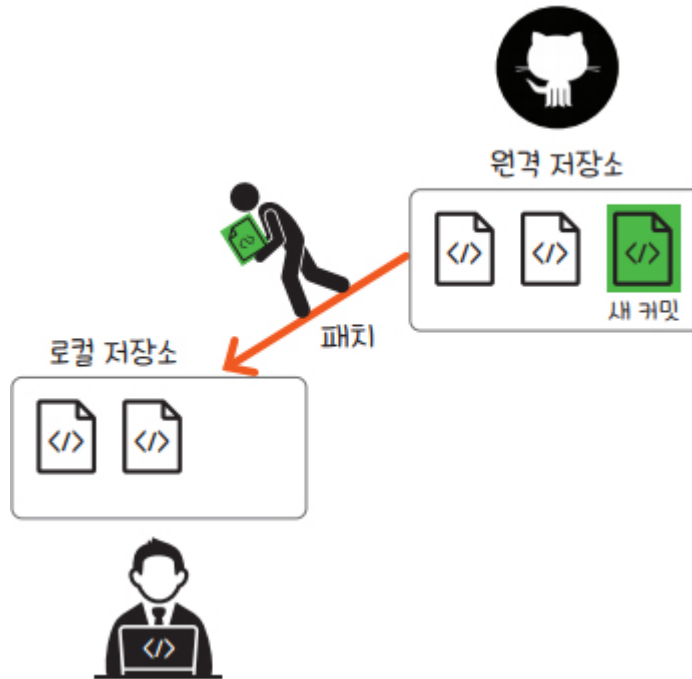
■ 직접 실습하기

- a.txt 파일을 삭제하고, B가 저장된 b.txt 파일을 추가해보기
 - 커밋 메시지는 delete a.txt and add b.txt
 - Push 완료 후 원격 저장소인 GitHub에도 push 결과 확인해보기
 - 커밋 수 증가 확인
 - 커밋 세부 내역 확인

원격 저장소와 상호작용 하기

■ (3) fetch (패치)

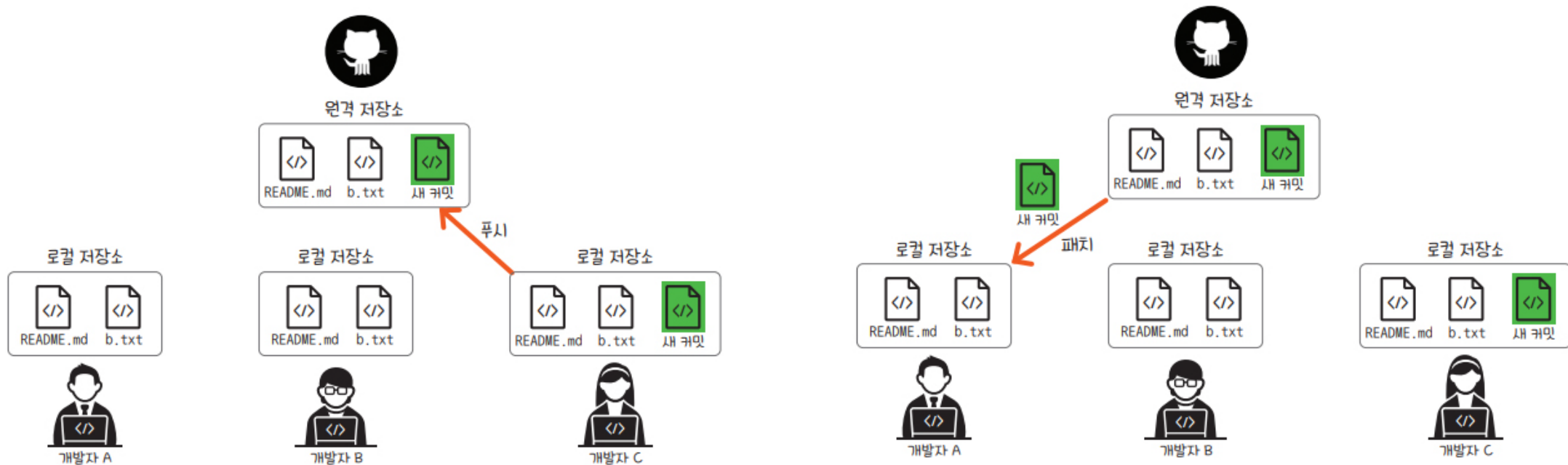
- 원격 저장소의 변경 사항들을 일단 가져만 오는 방법



원격 저장소와 상호작용 하기

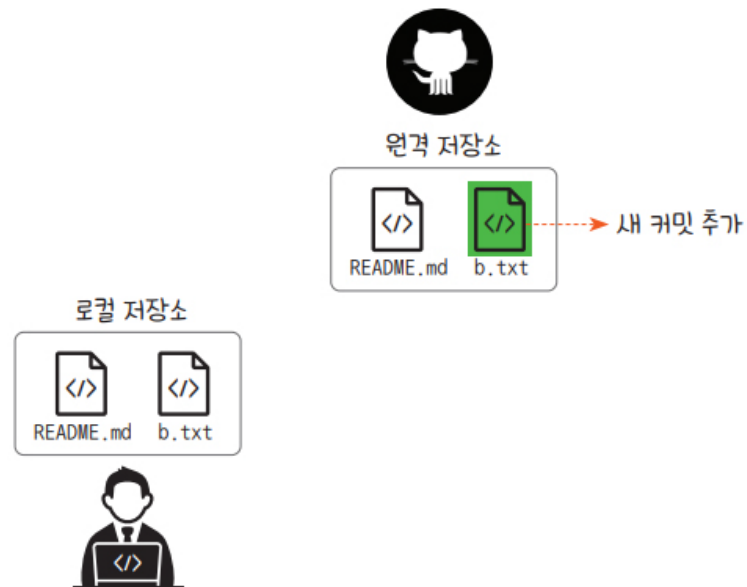
■ (3) fetch (패치)

- 다른 개발자가 푸시한 내용을 여러분의 로컬로 가져오고 싶을 때 사용
 - 원격 저장소를 여러 SW개발자가 협업하여 개발하고 있는 경우, 각자 새로운 변경 사항을 추가할 수 있기 때문에 원격저장소는 시시각각 변할 수 있음.



(실습) fetch

- GitHub에서 파일 수정하고 본인의 원격저장소로 fetch 해보기
 - 본인의 GitHub 원격 저장소에서 파일을 직접 수정하고 커밋을 추가
 - (예) 앞서 실습에서 추가된 b.txt 파일을 더블클릭 후 우측에 연필아이콘 클릭하여 직접 수정하고, 커밋을 추가
 - This is written in GitHub. 문장 추가
 - Commit changes... 클릭
 - 커밋 메시지는 Update b.txt in GitHub 적고 Commit changes 클릭

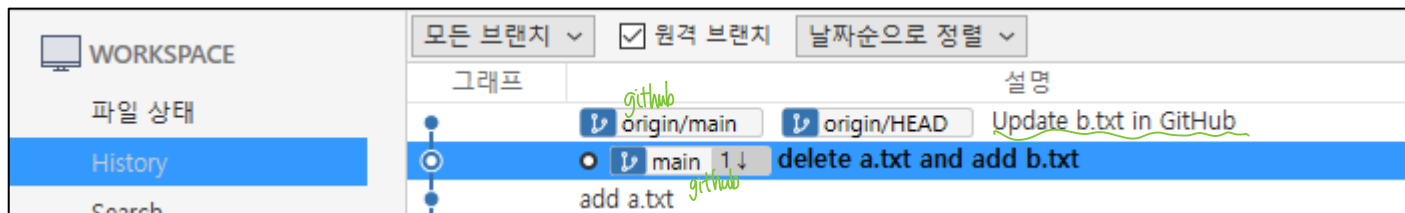
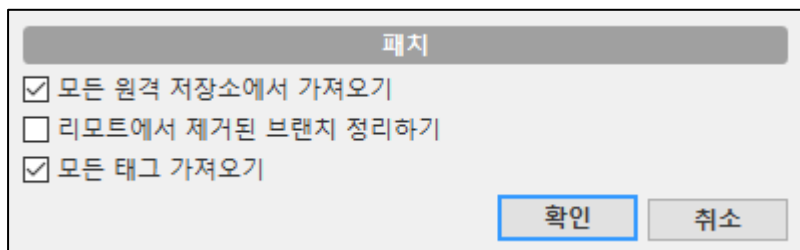


(실습) fetch

■ GitHub에서 파일 수정하고 본인의 원격저장소로 fetch 해보기

● SourceTree에서 원격 저장소 상황 fetch하기

■ 상단에 [패치] 클릭 - 확인 클릭



■ 체크!

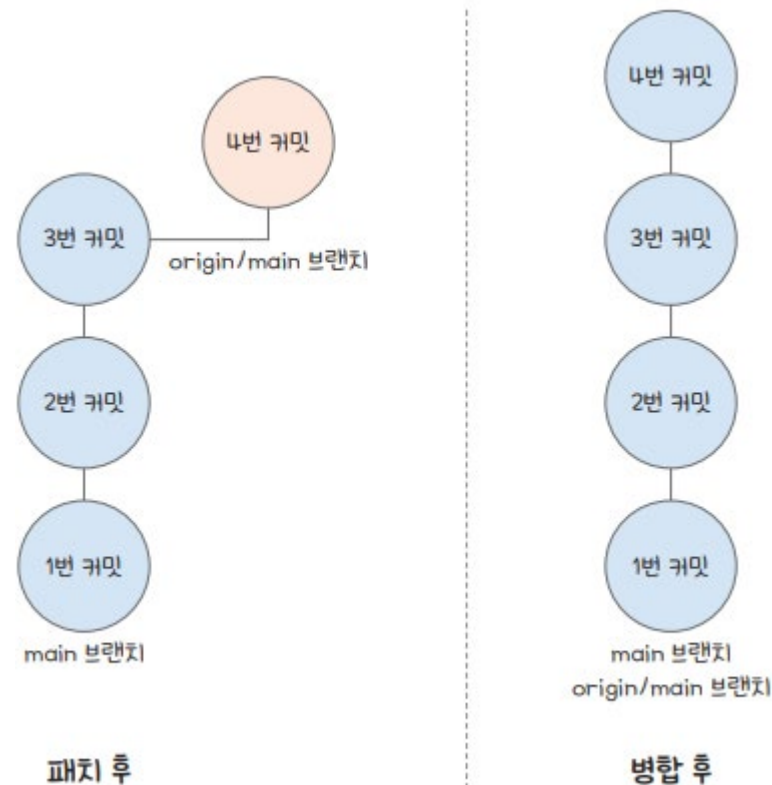
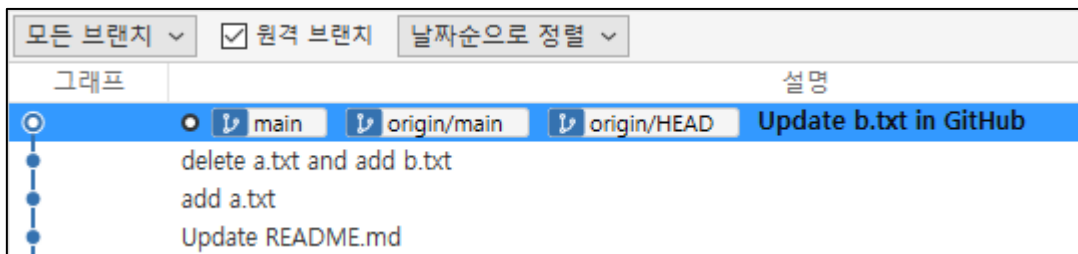
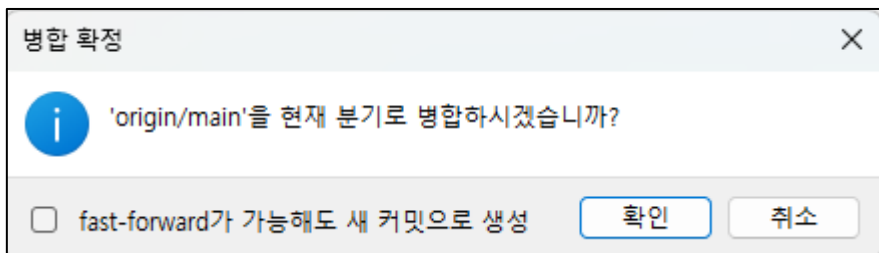
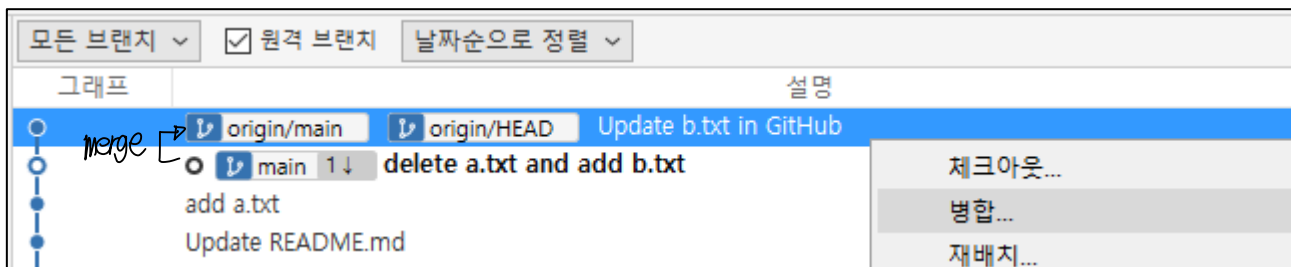
- 로컬 저장소의 브랜치가 변경되지 않았음
- 로컬의 main 브랜치는 여전히 세 번째 커밋을 가리키고, origin/main 브랜치는 네 번째 커밋을 가리킴
- 즉, 패치해도 원격 저장소의 내용이 로컬 저장소에 병합되지 않음

(실습) fetch

■ GitHub에서 파일 수정하고 본인의 원격저장소로 fetch 해보기

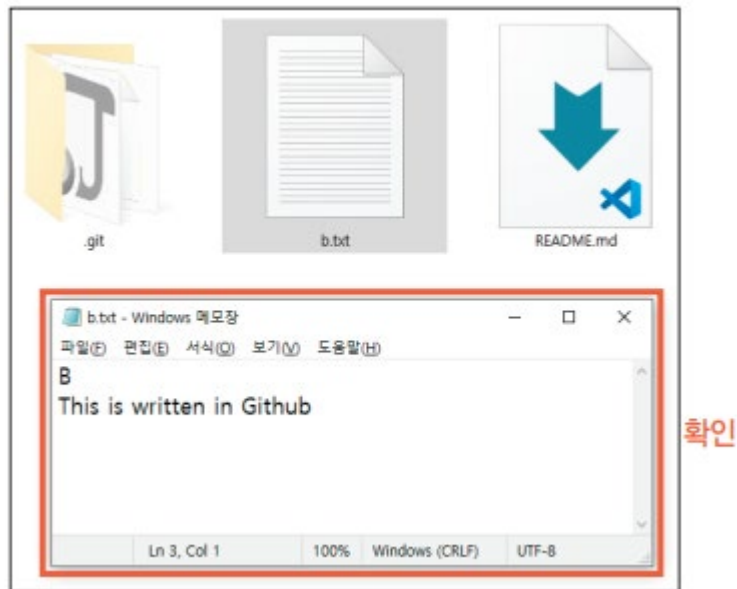
● fetch이후 merge하기

■ origin/main 브랜치에 마우스 오른쪽 버튼 - 병합



(실습) fetch

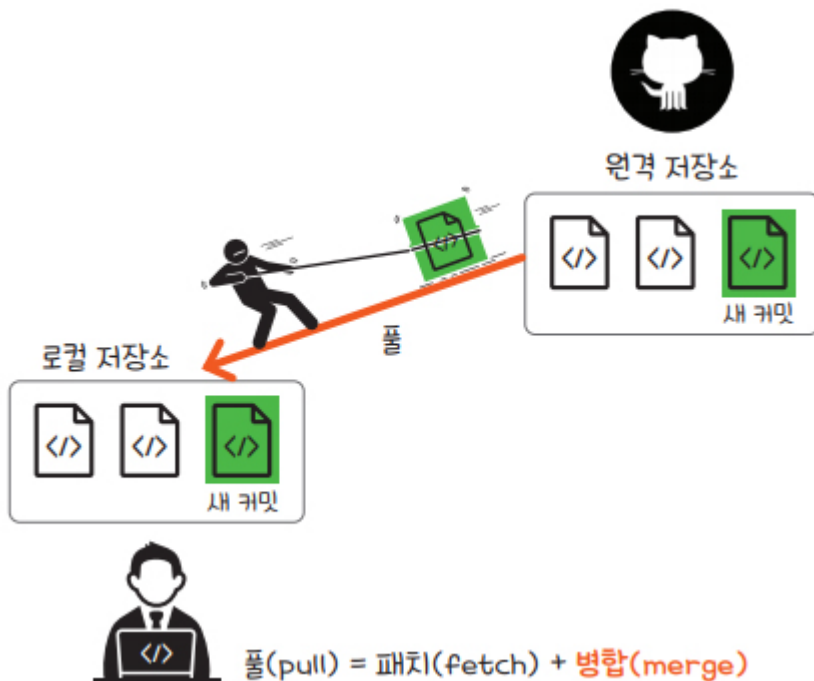
- GitHub에서 파일 수정하고 본인의 원격저장소로 fetch 해보기
 - fetch이후 merge하기
 - 병합 후 탐색기를 열어 로컬 저장소의 b.txt 파일을 클릭하면 깃허브에서 작성한 커밋 확인 가능



원격 저장소와 상호작용 하기

■ (4) pull (풀)

- 원격 저장소를 가져와서 합치는 방법
 - pull = fetch + merge

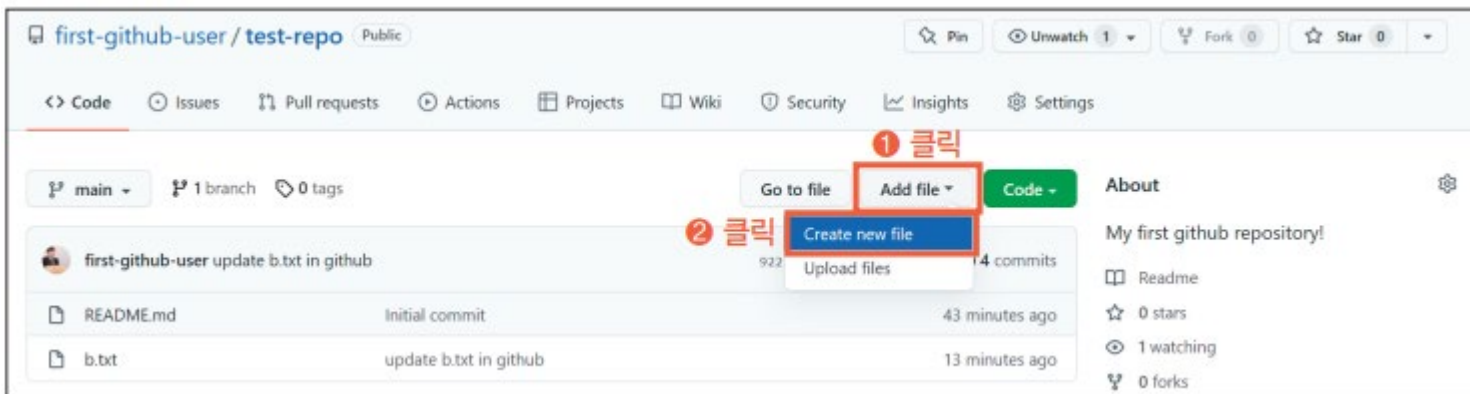


(실습) pull

■ GitHub에서 파일을 생성하고 본인의 원격저장소로 pull 해보기

● GitHub에서 파일 생성하기

■ [Add file] - [Create new file]



■ (예) c.txt 파일을 만들고, 안에는 This file is created in github.를 작성

■ (예) 커밋 메시지로 create c.txt in github를 작성하고 Commit changes을 클릭

Name	Last commit message
README.md	Update README.md
b.txt	Update b.txt in GitHub
c.txt	Create c.txt in github

(실습) pull

- GitHub에서 파일을 생성하고 본인의 원격저장소로 pull 해보기

- 결과

- 로컬 저장소가 모르는 사이에 원격 저장소에 변경 사항이 발생

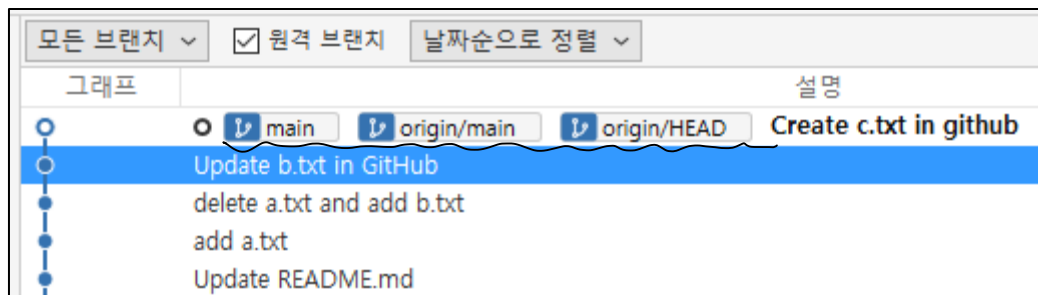


(실습) pull

- GitHub에서 파일을 생성하고 본인의 원격저장소로 pull 해보기

- SourceTree에서 원격 저장소 상황 pull 하기

- 상단에 [Pull] 클릭 - 확인 클릭



- 체크! (fetch와이 차이점)

- 로컬 저장소의 브랜치가 바로 변경됨.

- 즉, pull하면 원격 저장소의 내용이 로컬 저장소에 병합됨.

fetch + pull

(실습) pull

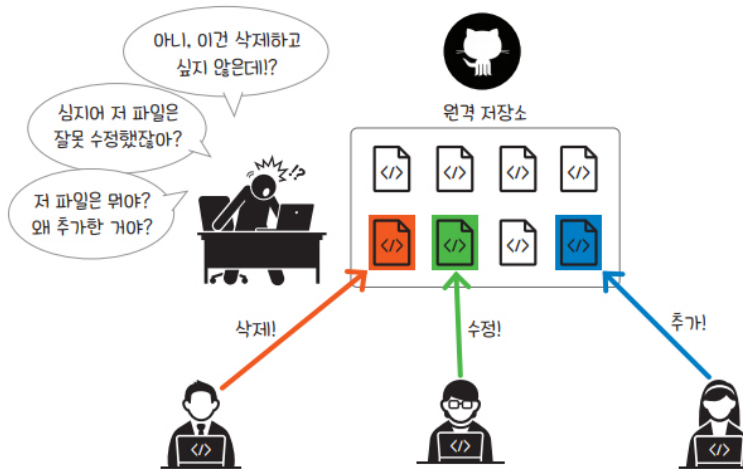
- GitHub에서 파일을 생성하고 본인의 원격저장소로 pull 해보기
 - SourceTree에서 원격 저장소 상황 pull 하기
 - 병합 후 탐색기를 열어 로컬 저장소의 c.txt 파일을 클릭하면 깃허브에서 작성한 커밋 확인 가능



pull request (PR)

■ 실무 개발 상황의 이해

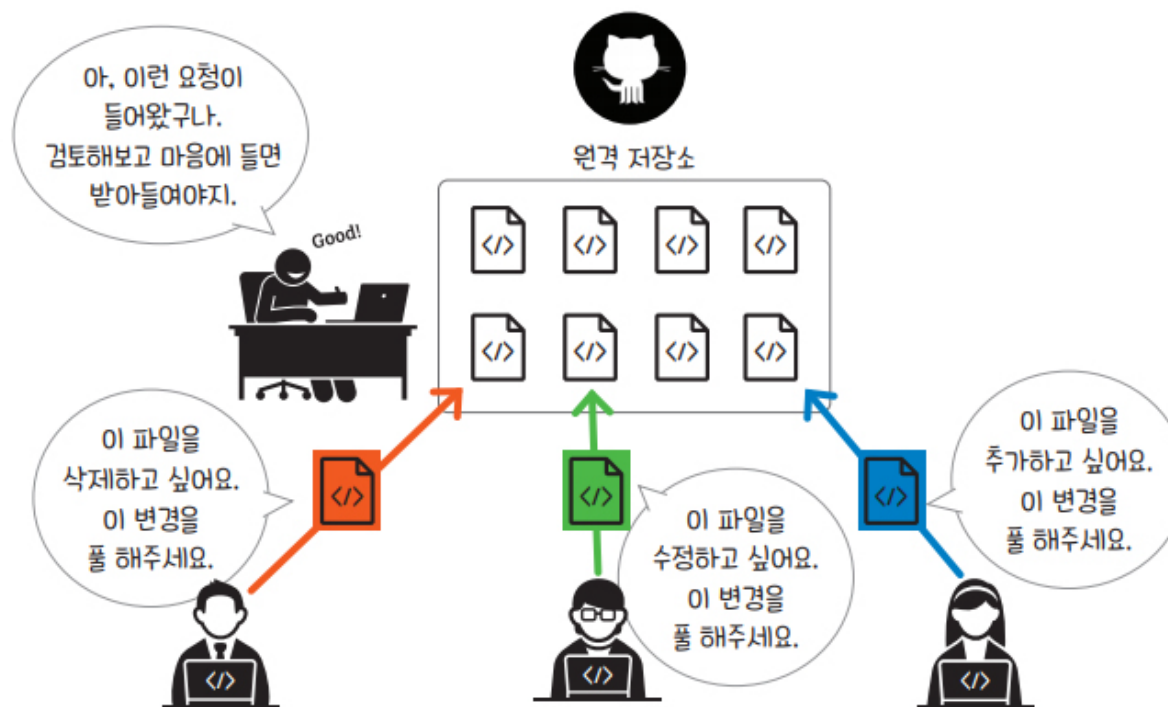
- 실무에서는 여러 개발자가 한 원격 저장소를 두고 개발하는 것이 일반적임
 - 하나의 원격 저장소에 여러 개발자가 코드를 기여할 수 있음.
- 본인이 직접 만들지 않은 원격 저장소에도 push할 수 있을까?
 - 일반적으로 그렇지 않음.
 - 만일 누구나 허락없이 소유하지도 않은 원격 저장소에 푸시할 수 있다면 원격 저장소의 소유자가 원하지도 않는 변경 사항들이 원격 저장소에 마구 추가되는 불상사가 발생할 수 있음.
 - 푸시 권한이 없는 원격 저장소에 어떻게 코드를 밀어넣을 수 있을까?
 - 어떻게 다른 원격 저장소에 변경 사항을 추가할 수 있을까?



pull request (PR)

■ PR의 사용

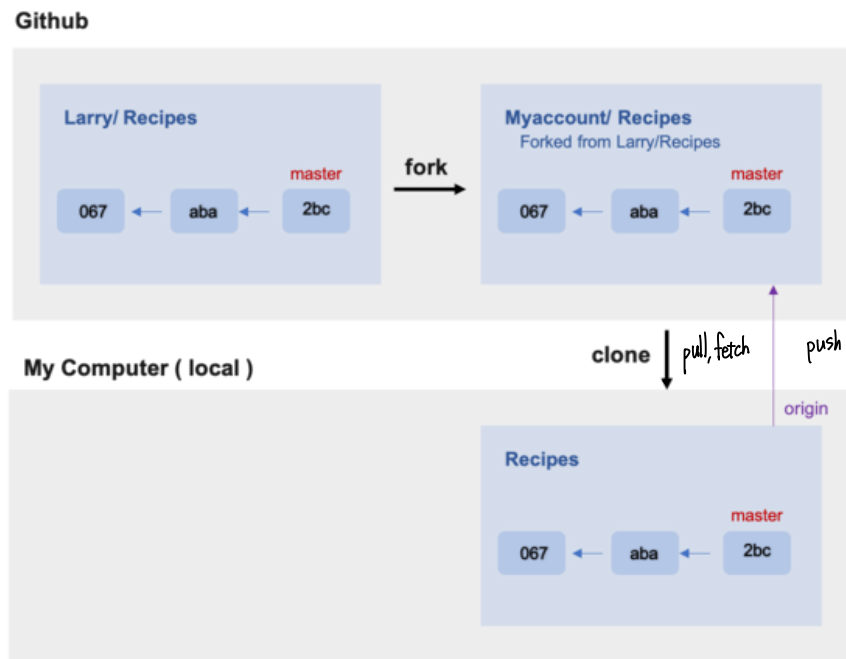
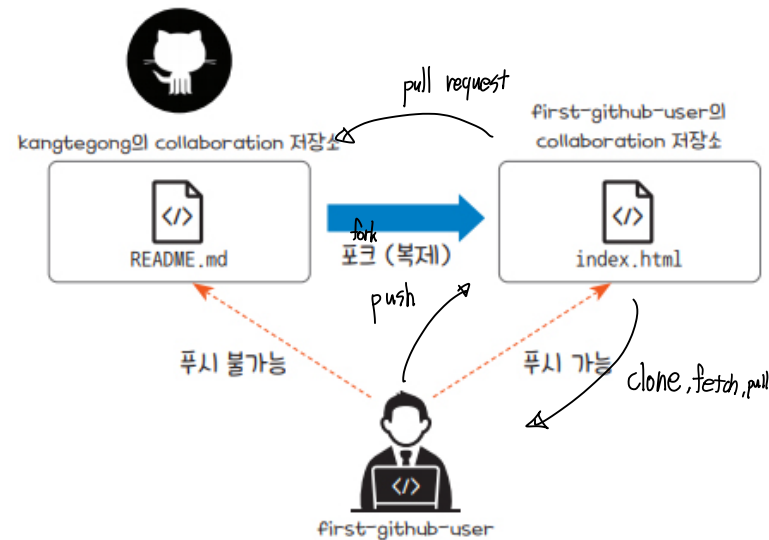
- PR: 원격 저장소가 내 변경 사항을 풀(pull)하도록 요청(request)하는 방법.
 - ‘내가 당신의 원격 저장소를 이렇게 변경하고 싶은데, 이 변경을 당신의 저장소로 풀 해주세요!’ 하고 부탁하는 방법.



GitHub로 다른 개발자와 서로 협업하기

■ 다른 개발자와 서로 협업 위한 PR 수행 단계

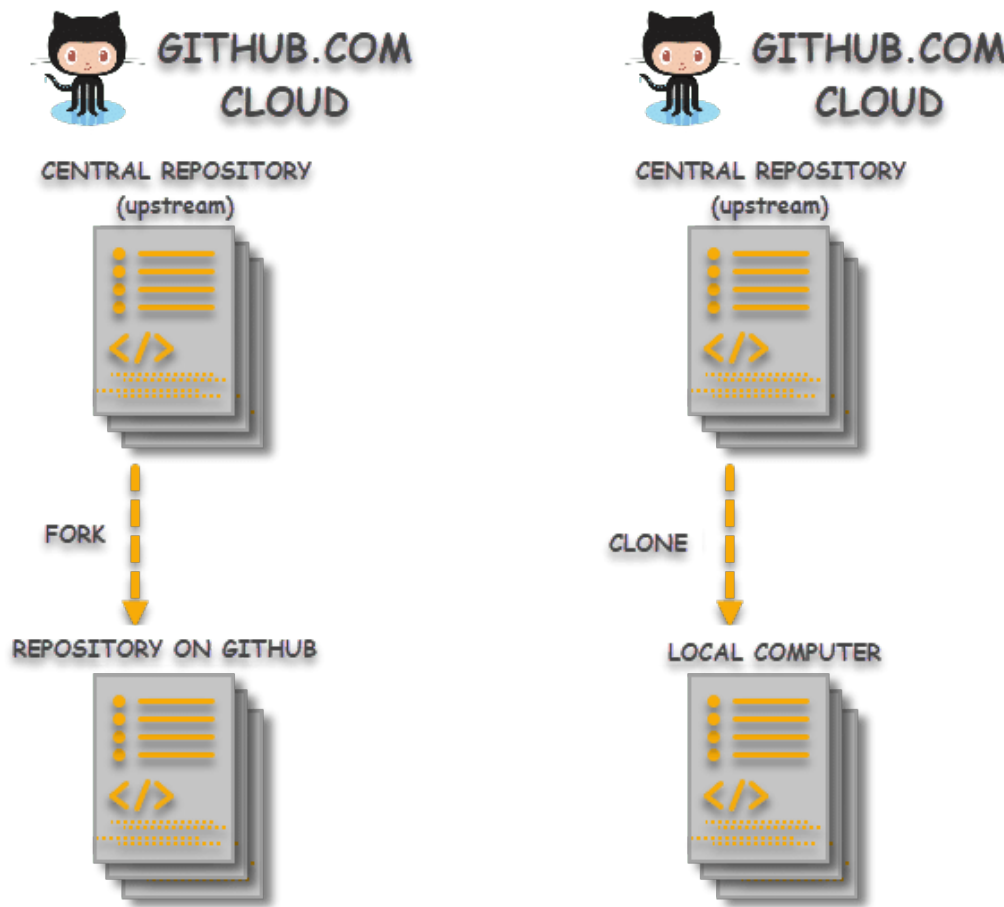
- (1) 기여하려는 저장소를 본인 계정으로 fork하기
 - fork: 원격 저장소를 여러분의 계정으로 복제하는 방법
- (2) fork한 저장소를 clone하기
- (3) branch 생성 후 생성한 branch에서 작업하기
- (4) 작업한 브랜치 push하기
- (5) pull request 보내기



GitHub로 다른 개발자와 서로 협업하기

■ fork와 clone 비교

- fork는 GitHub 계정에서 수행
 - fork한 repository는 GitHub 계정에 생성
- clone은 git을 사용하여 수행
 - clone 한 repository는 로컬컴퓨터에 생성



GitHub로 다른 개발자와 서로 협업하기

■ fork와 clone 비교

● Change가 발생할 경우

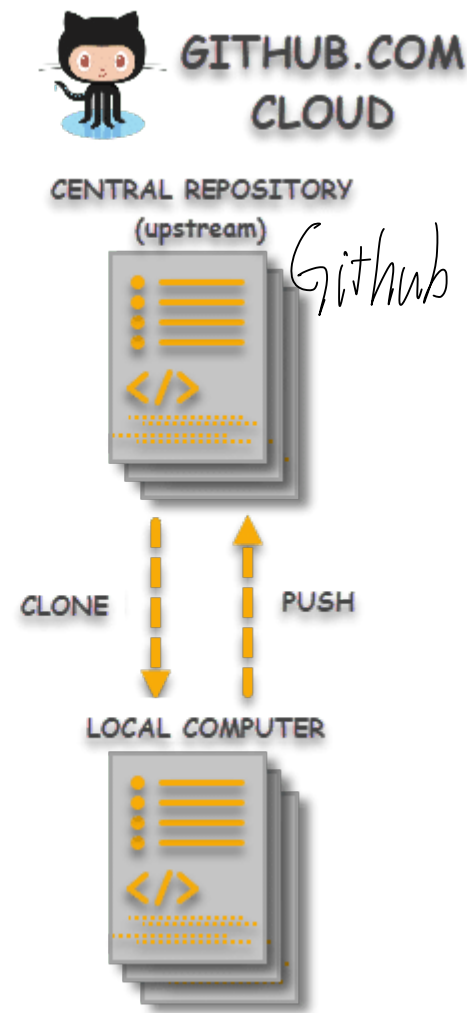
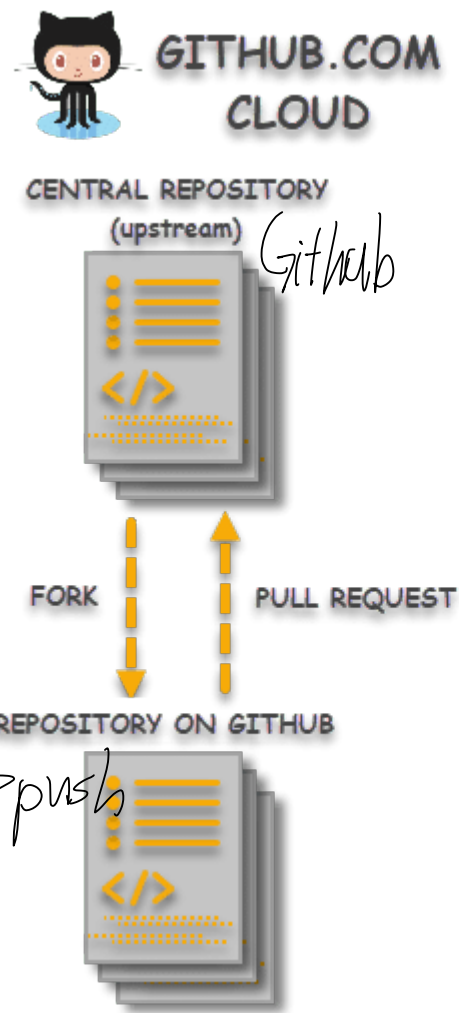
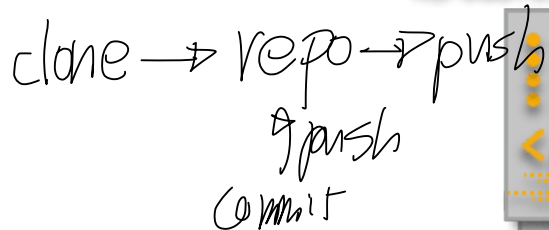
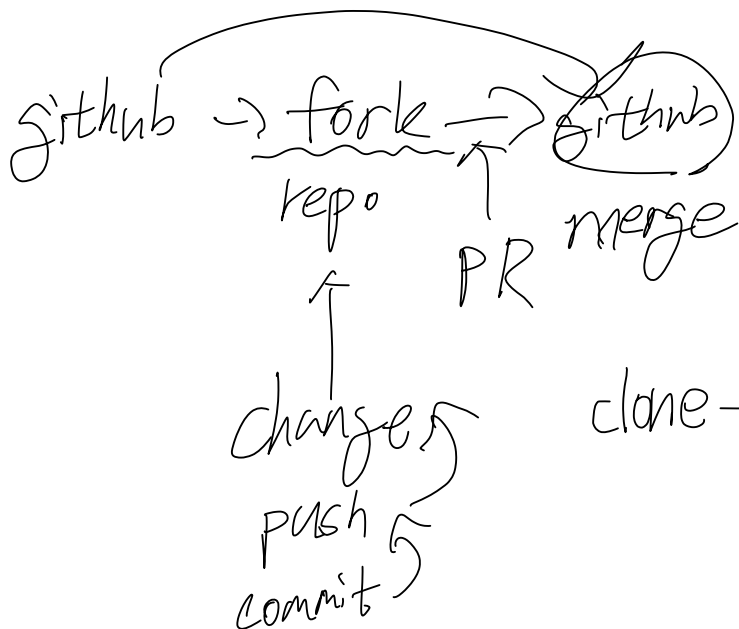
권한 없는 사용자

■ fork는 PR로 원저장소에 merge 요청함.

권한 있는 사용자

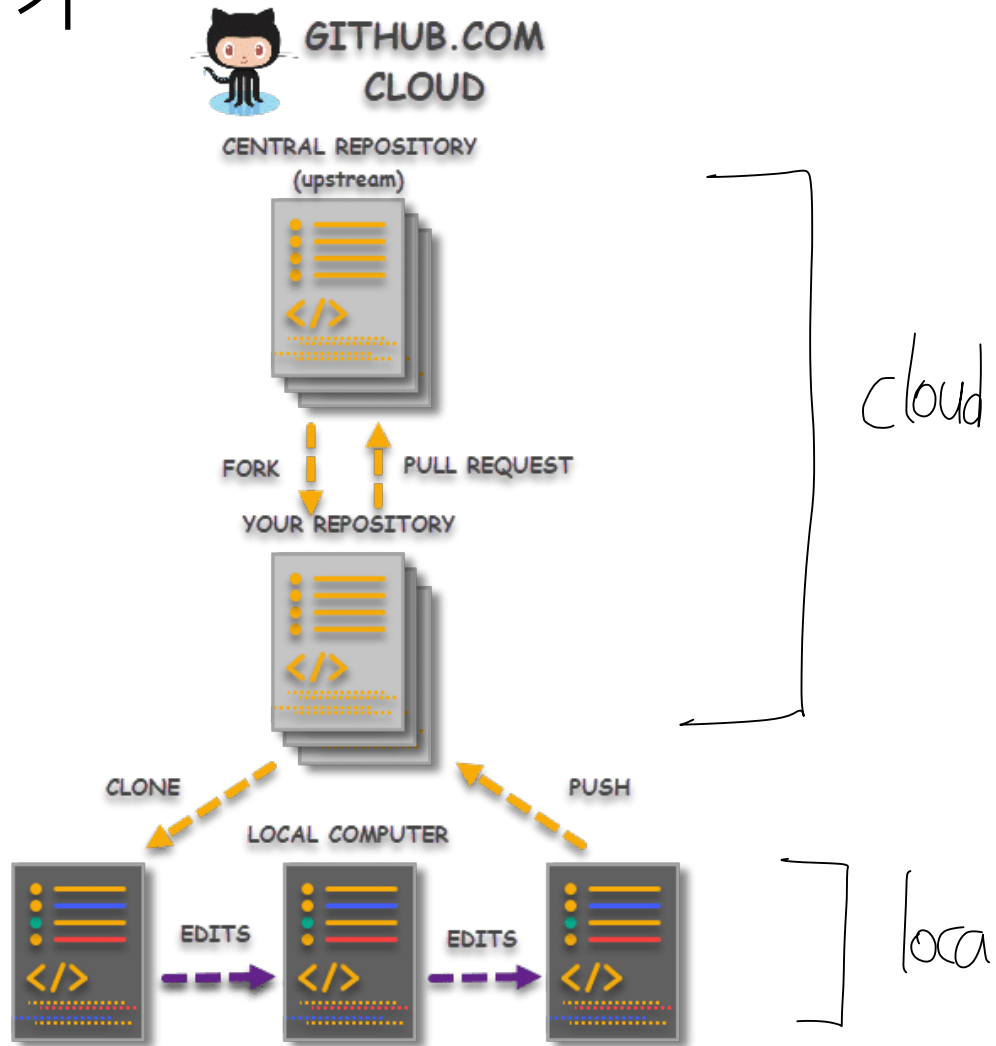
■ clone은 push로 원저장소에 바로 밀어넣음.

- 사용자가 직접 push할 권한이 있어야 함.



GitHub로 다른 개발자와 서로 협업하기

- 다른 개발자와 서로 협업하기




(실습) pull request

■ 원격저장소에 pull request 해보기

- https://github.com/ysseo29/OSS_YU_test_repo
- index.html 파일에 여러분의 GitHub 계정 이름을 적어 넣기
 - 방법: index.html 파일을 열어 ``의 윗 줄에 `여러분 계정 이름`을 적어넣기

● PR 보내기

- 위 링크는 타인의 소유한 저장소이므로
여러분은 이 저장소에 직접 푸시할 수 없음
- “index.html을 변경한 내용을 풀 해주세요!”
하며 PR 보내기



```
Code Blame 24 lines (23 loc) · 532 Bytes Code 55% faster with GitHub Copilot
1 <!DOCTYPE html>
2 <html>
3 <body>
4 <h1>Pull Request Succeeded Members</h1>
5 <p>
6   아래에 본인의 계정 이름이 적힌 li 태그를 추가하고 Pull Request를 보내세요.
7 </p>
8 <ul>
9   <li>YU</li>
10  <li>good</li>
11  <li>kang</li>
12  <li>ghdlend</li>
13  <li>lee</li>
14  <li>sfg486222</li>
15  <li>ghdlend123</li>
16  <li>minpinetree</li>
17  <li>again_yeongwol</li>
18  <li>yeongwol</li>
19  <li>JeongDaiHyun</li>
20  <li>leejiho12</li>
21
22 </ul>
23 </body>
24 </html>
```

출처: 강민철, 모두의 깃&깃허브

(실습) pull request

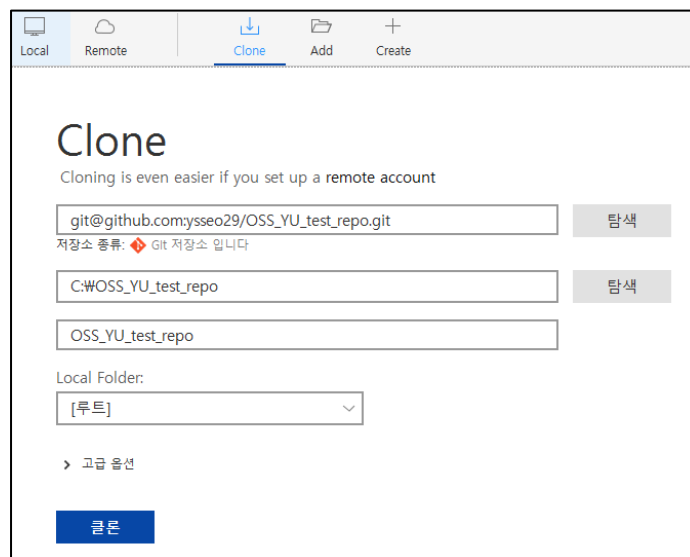
■ (1) 기여하려는 저장소를 본인 계정으로 fork 하기

- 기여하려는 원격 저장소에 접속하여 상단에 fork 클릭
- 사용자의 계정으로 복제: 복제된 저장소에 push 가능



■ (2) fork한 저장소를 clone 하기

- SourceTree에서 상단 remote 클릭 후, 방금 fork한 저장소를 clone 하기



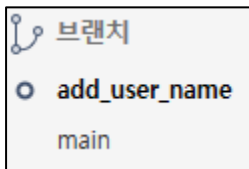
(실습) pull request

■ (3) branch 생성 후 생성한 branch에서 작업하기

- 새로운 branch 생성하기

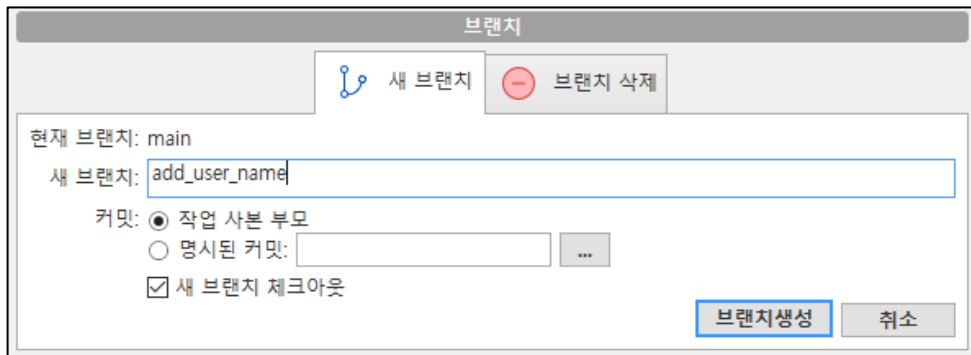
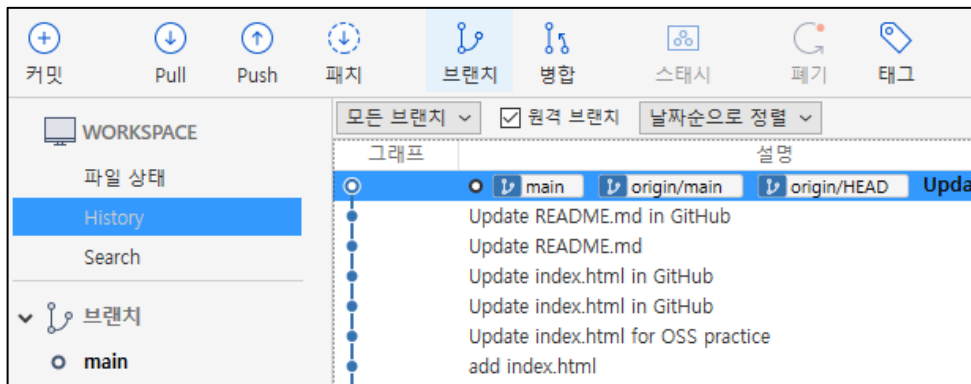
 - (예) add_user_name

- 생성된 branch로 checkout



- 해당 branch에서 작업하기

 - index.html 파일을 열어 학번_계정 추가

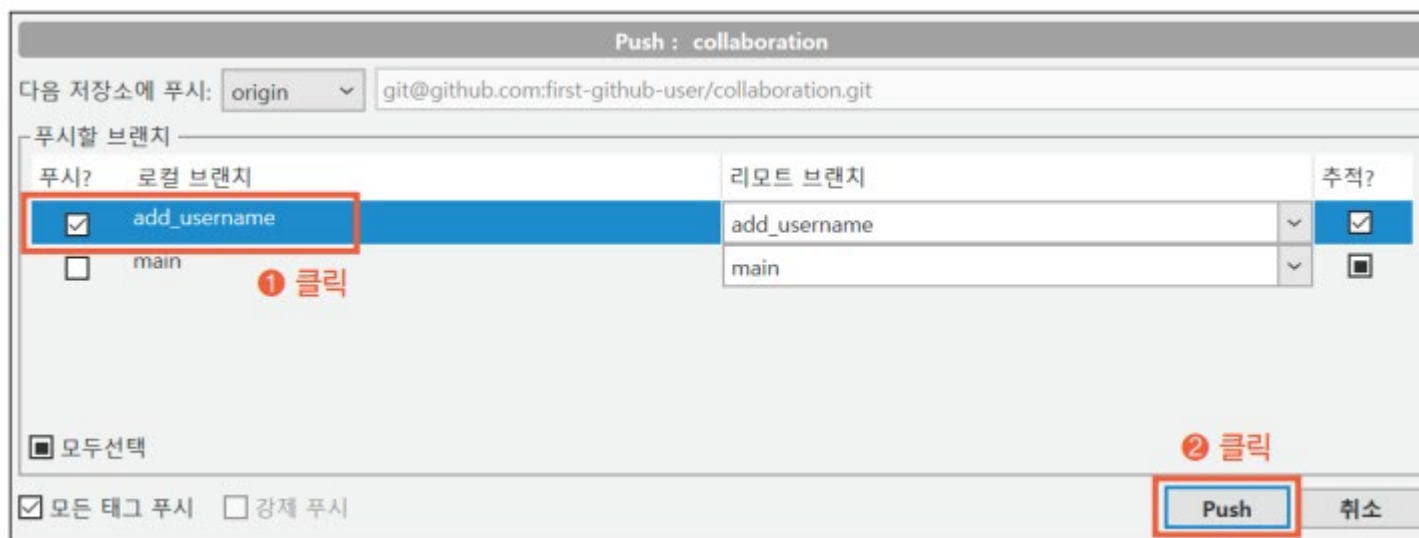
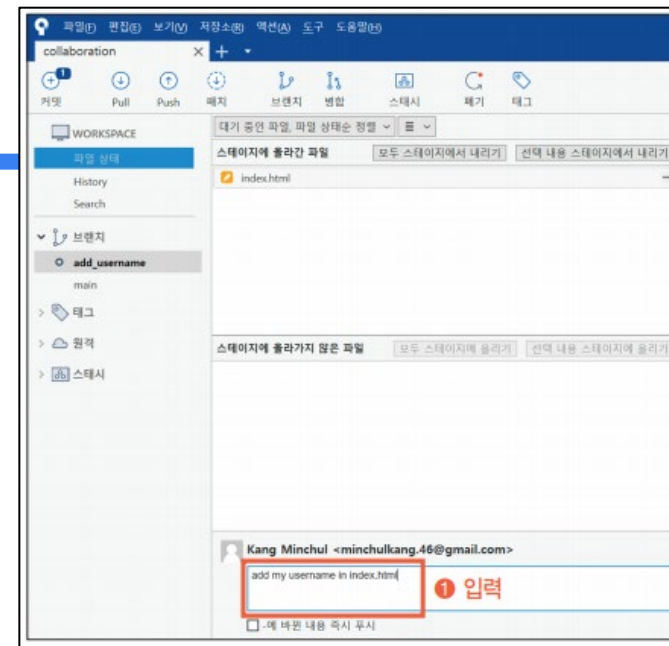


(실습) pull request

■ (4) 작업한 브랜치 push하기

- 수정한 index.html을 add_username 브랜치에 커밋
- 커밋 메시지: add my username in index.html
- 메뉴 상단에 push 클릭

- 작업한 add_username 브랜치를 푸시하기 위해 add_username을 클릭한 뒤 Push를 클릭
- 새로 생성하고 작업한 브랜치를 푸시한다는 것에 유의

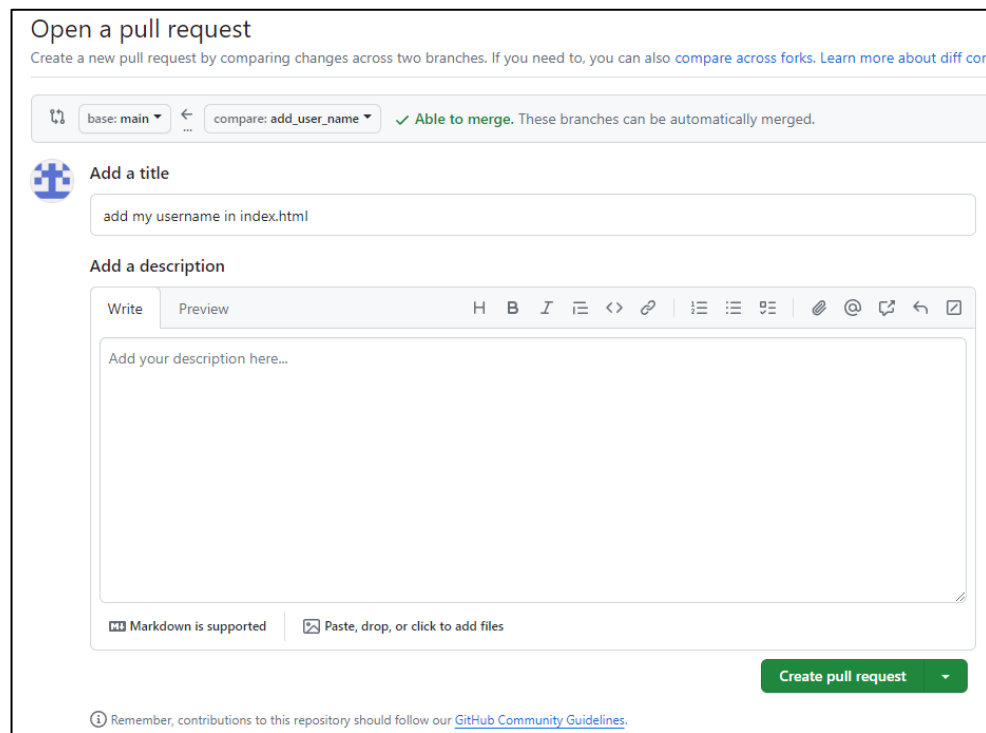


출처: 강민철, 모두의 깃&깃허브

(실습) pull request

■ (5) pull request 보내기

- GitHub에 fork한 저장소로 돌아오면 상단에 Compare & pull request 클릭
 - 제목: [학번] add my username in index.html
 - 커밋메시지: 제목과 동일
- 하단에 Create pull request 클릭



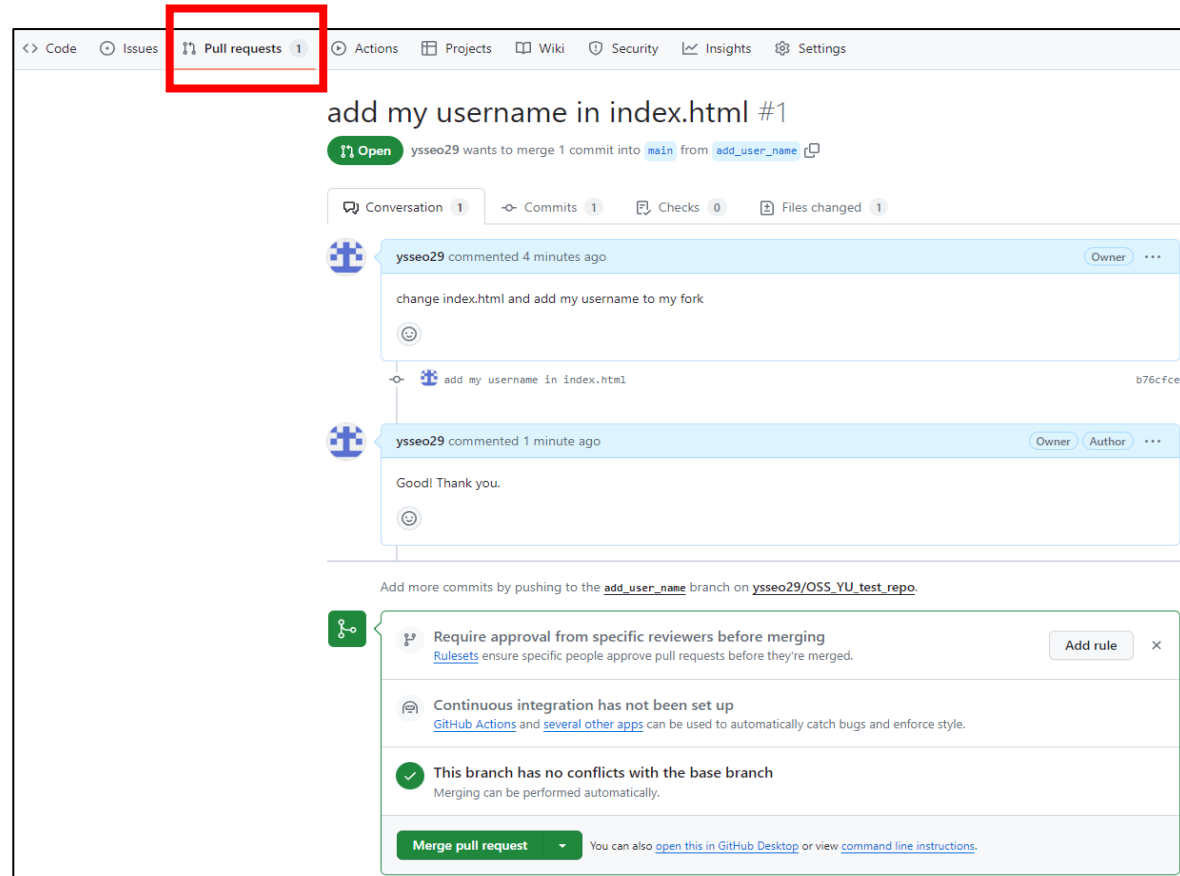
The screenshot shows the GitHub interface for creating a pull request. At the top, it says 'Open a pull request' and provides a brief instruction. Below this, there's a section for selecting the base and compare branches. The base is set to 'main' and the compare branch is 'add_user_name'. A green checkmark indicates 'Able to merge'. The next section is 'Add a title', with the text 'add my username in index.html' entered. Below that is 'Add a description', which has a rich text editor with a toolbar and a large text area. At the bottom right, there is a green button labeled 'Create pull request'. A footer note mentions the GitHub Community Guidelines.

출처: 강민철, 모두의 깃&깃허브

(실습) pull request

■ 원격 저장소 소유자의 pull request 반영

- 원 소유자는 PR 검토 후 merge 함.
- 올라온 PR에 댓글을 남길수도 있음.
 - 댓글로 여러분의 작업에 대한 의견을 남기거나 부족한 내용을 보충해 주기도 함.
 - 코드 논의의 장이 되기도 함.
- Merge가 완료되면 해당 저장소에서 여러분이 index.html에 추가한 이름 확인 가능.



Summary (1/2)

■ Pull request를 통한 협업 및 오픈소스 기여 과정

● 기여하려는 저장소를 본인 계정으로 fork 하기

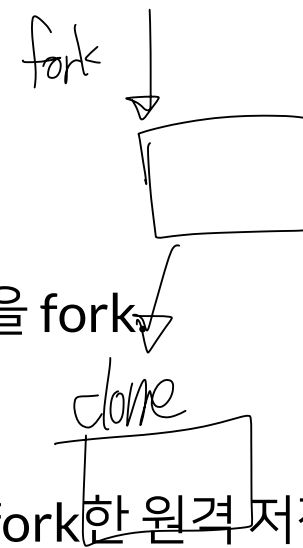
- 일반적으로 여러분이 소유하지 않은 원격 저장소에는 푸시할 수 없음.
- 그렇기에 여러분의 계정으로 원격 저장소를 복제해와야 하는데, 이 과정을 fork

● fork한 본인 계정의 저장소를 clone하기

- 여러분이 소유하지 않은 원격 저장소에 push하기는 불가능할지 몰라도 fork한 원격 저장소에는 push 가능.
- 따라서 fork한 저장소를 clone.

● branch 생성 후 생성한 branch에서 작업하기

- 새로운 branch를 생성한 후 해당 branch에서 변경 사항을 만들고 commit 하기.



Summary (2/2)

- Pull request를 통한 협업 및 오픈소스 기여 과정
 - 작업한 브랜치 push하기
 - 생성한 브랜치를 푸시.
 - GitHub에 pull request 버튼이 생성.
 - pull request 보내기
 - 마지막으로 pull request 보내기.
- 원 저장소 관리자가 pull request를 분석후, 유효한 변경일 경우 merge 함.