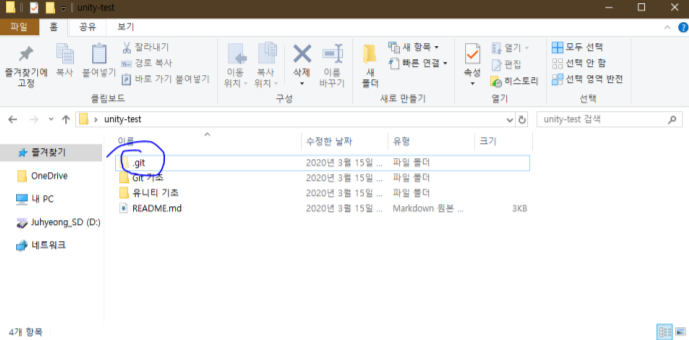
**소스트리 사용법 정리**

**용어**

* 저장소(Repository): 내가 Git 으로 관리할 폴더(디렉터리)이다
* 커밋(Commit): Git에서 버전을 기록하는 단위이다
* 커밋한다: Git에 하나의 버전을 기록한다
* 로컬(Local): 네트워크 없이 접속 가능한 것
* 로컬 컴퓨터 = 내가 바로 사용하고 있는 컴퓨터라고 할 수 있다
* 원격(Remote): 네트워크를 접속해서 사용해야 하는 것
* 클라우드(Google Drive 같은 것): Git 저장소를 올려서 공유하는 곳이다
  + (ex) GitHub, Bitbucket
* 푸시(Push): 로컬에 있는 저장소를 인터넷(Remote)에 올리는 것
  + 푸시를 하면 .git 폴더를 포함해서 나의 모든 파일과 변경 과정(커밋)들을 업로드 한다
* 복제(Clone): 인터넷에 있는(원격) 저장소를 로컬에 다운 받는 것

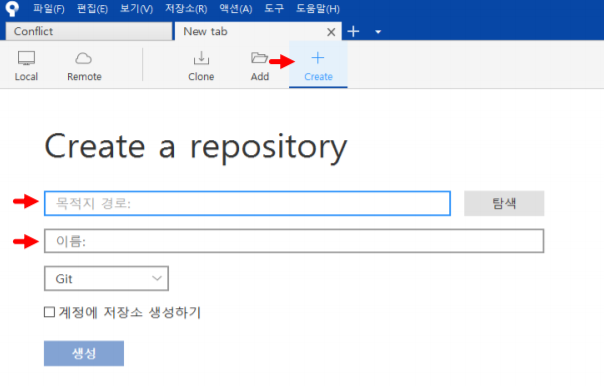
**저장소 선언하기**

저장소 선언한다는 것은 이 폴더를 Git이라는 툴로 관리하겠다는 것

저장소를 선언한 후

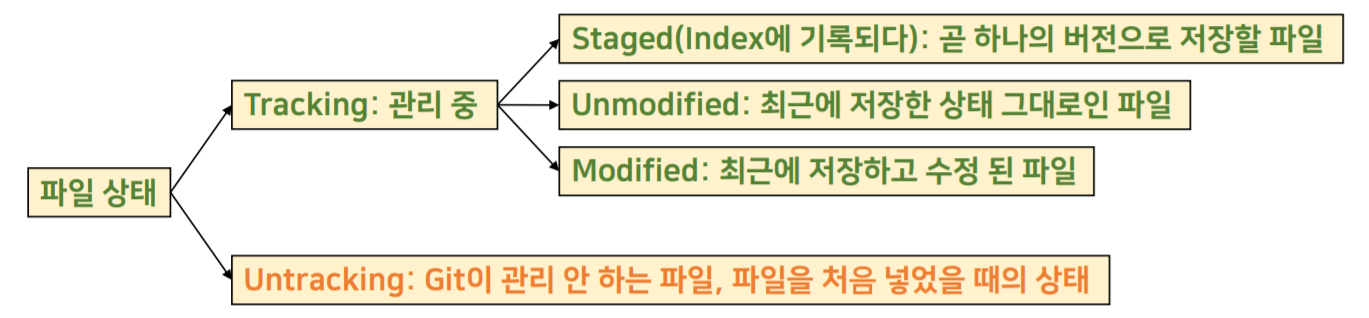
* .git 숨김 폴더가 생겼는지 확인해보자
* 이 폴더에 우리의 버전들이 기록된다. 우리는 건들 일이 없다!
* 이 파일을 삭제하면 우리가 Git으로 저장한 기록들이 다 같이 삭제되기 때문에 절대 삭제하면 안 된다

**저장소 선언을 해보자**

Create a repository

1. 상단의 Create 탭을 누른다
2. 목적지 경로: 어떤 폴더를 git으로 관리할지 설정
3. 이름: 저장소 명 설정

**Workspace - 파일 상태**

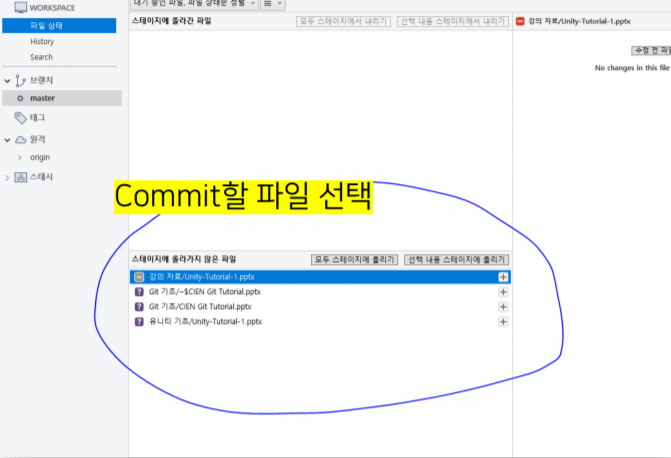
Git의 파일 상태 분류법

* 파일 상태: 내 파일의 상태를 확인할 수 있는 창
* 새로운 Text파일을 만들고 파일 상태를 확인해보자
* Git의 파일 상태 구분법

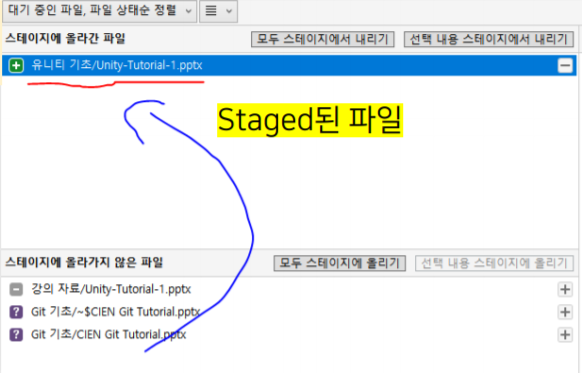
**Stage에 올리기 – Commit할 파일 선택**

Commit하기 전 단계  
Commit 할 파일들을 선택하며 이것을 Stage한다라고 한다

Stage 전

Stage 전

Stage 후

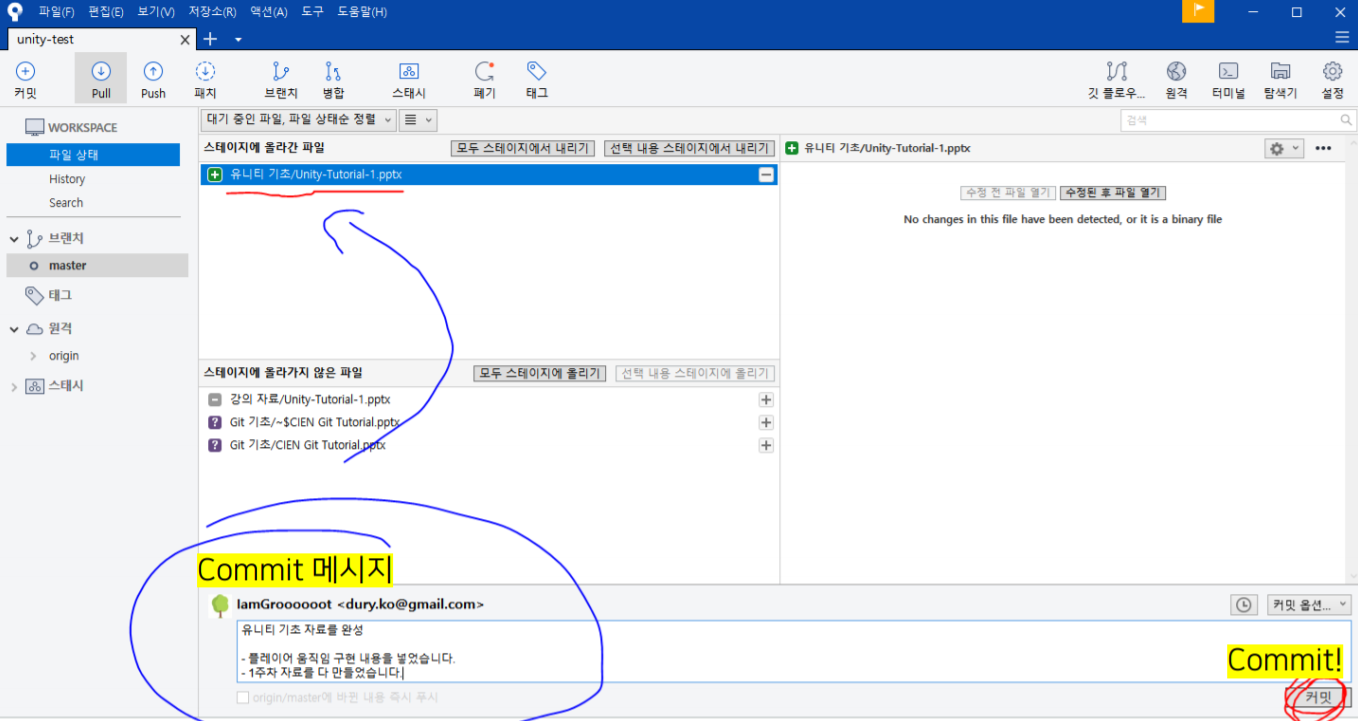
Stage 후

**Commit하기 – Git의 실질적인 기록**

커밋한다는 것은 하나의 버전으로 저장한다는 뜻이다

* 무조건 Stage(Index)에 올라간 파일들만 기록된다
* 어떤 것이 변했는지 알 수 있도록 메시지를 남겨줘야 한다

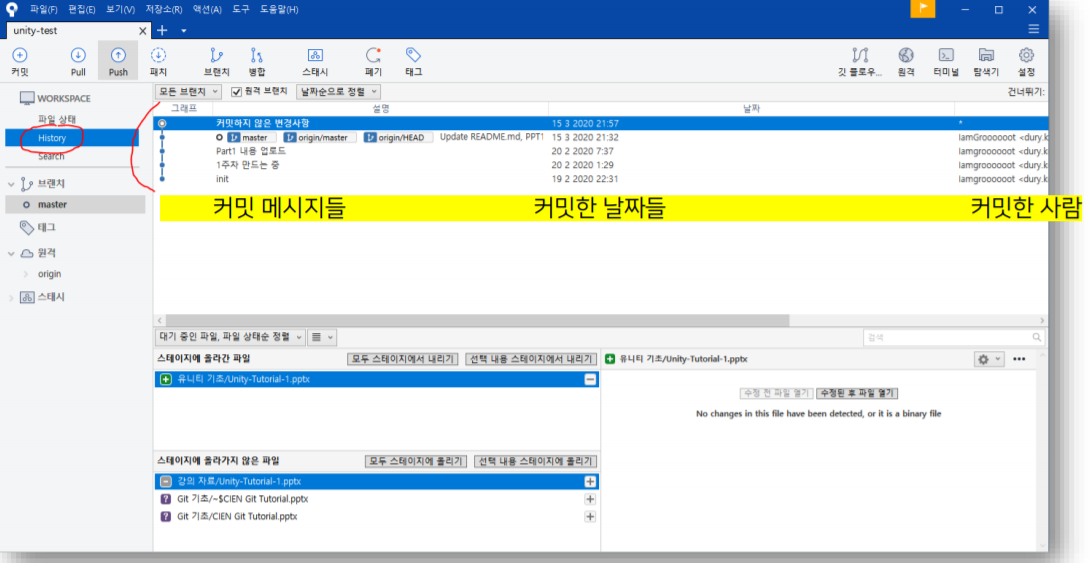
메시지를 작성한 후 커밋을 클릭해보자

Commit

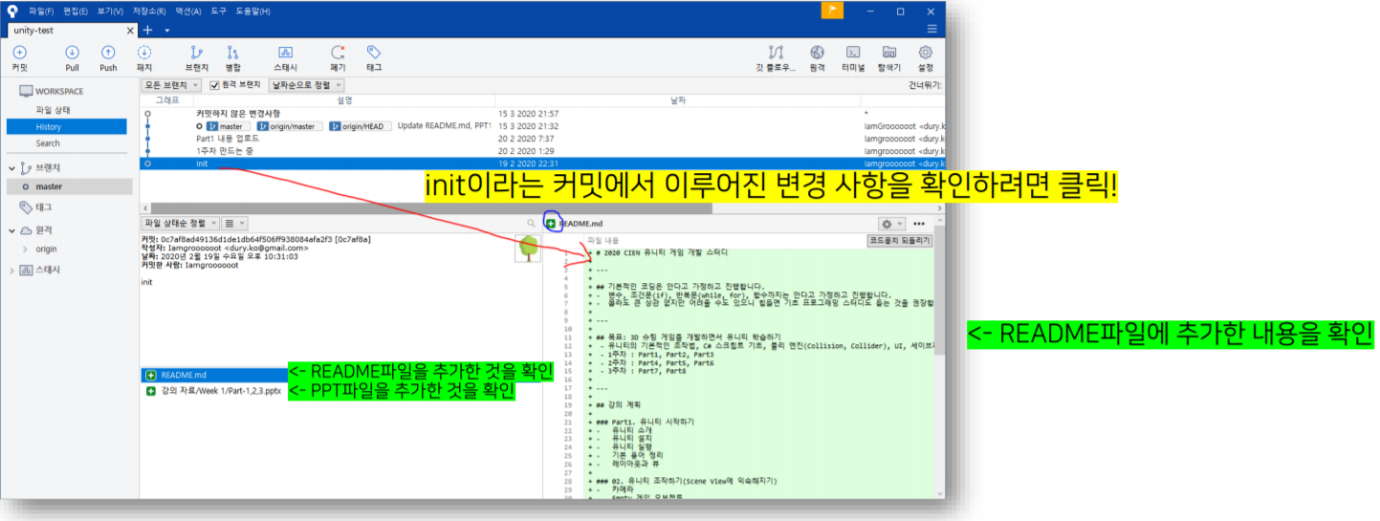
**Workspace – History에 대해서 알아보자**

Workspace – History

저장된 commit들의 기록을 확인하는 창

History

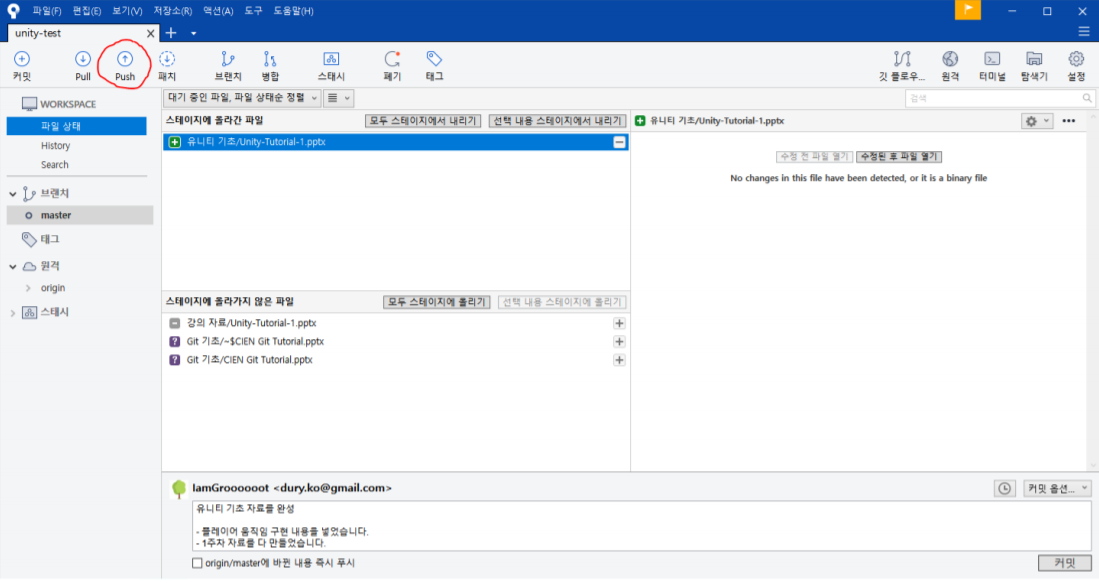
Workspace – History – Show

History – Show

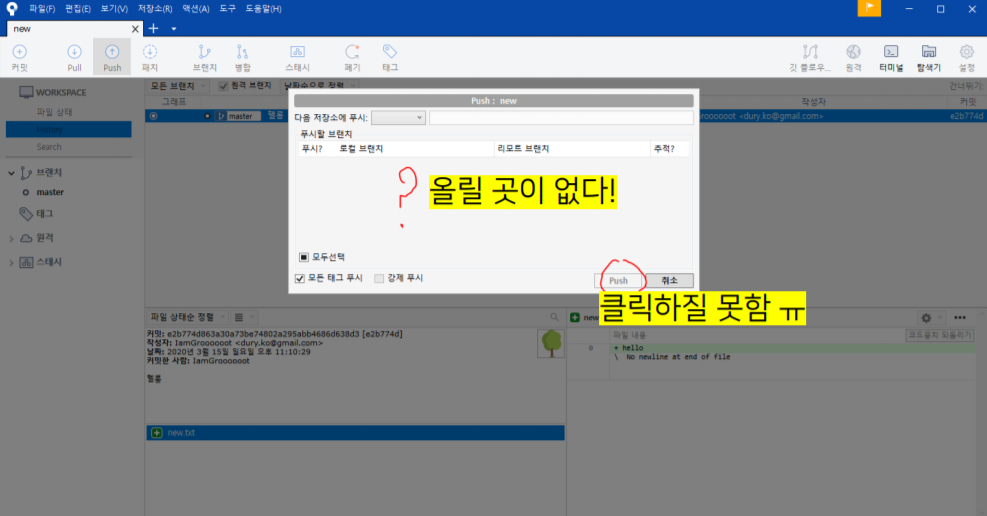
**내 컴퓨터에서 만든 저장소를 공유해보자**

**Push – 인터넷에 저장소 올리기**

Push 버튼 클릭해보자

푸시

Push를 하지 못할 것이다

푸시 실패!

원격을 추가해주지 않고 Push를 누르면 올릴 곳이 없으므로 Push를 못한다.  
우리처럼 로컬에서 저장소를 생성한 경우가 이와 같은 경우이다

**해결해보자**

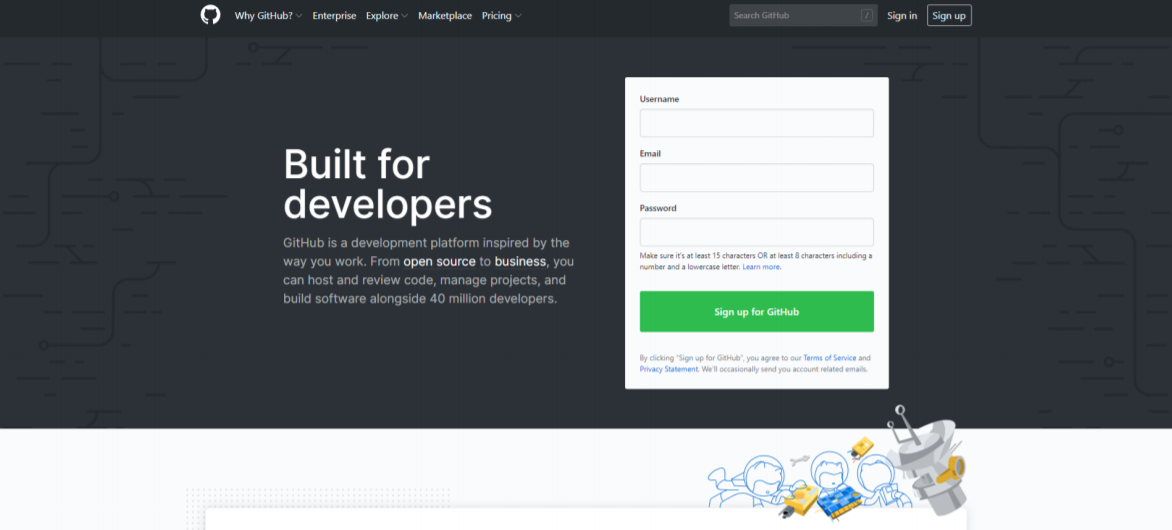
더보기

GitHub에 Push할 곳을 만들자

* 우리 컴퓨터에 있는 Git 저장소를 올리기 위해 비어 있는 GitHub 저장소를 만들자
* GitHub 저장소 = 우리의 원격 저장소

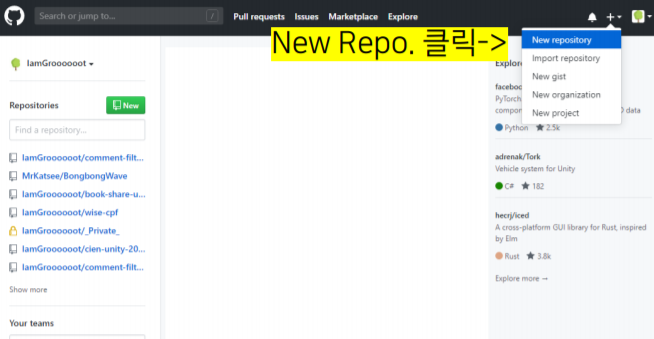
먼저 GitHub에 가입하자.

<https://github.com/> 여기서 가입 한 후, 이메일 인증을 하면 된다.

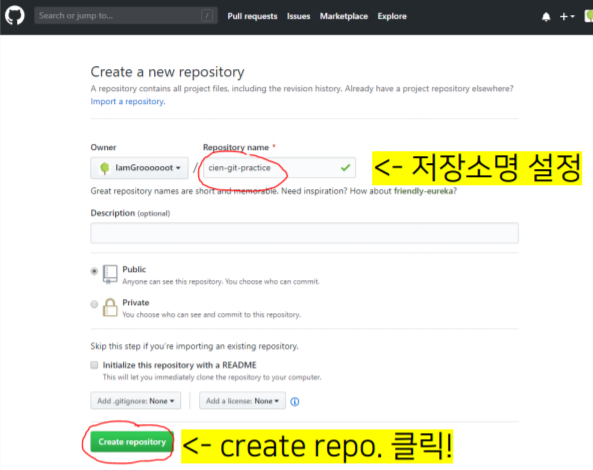
GitHub.com

GitHub에서 원격 저장소를 만들어 보자

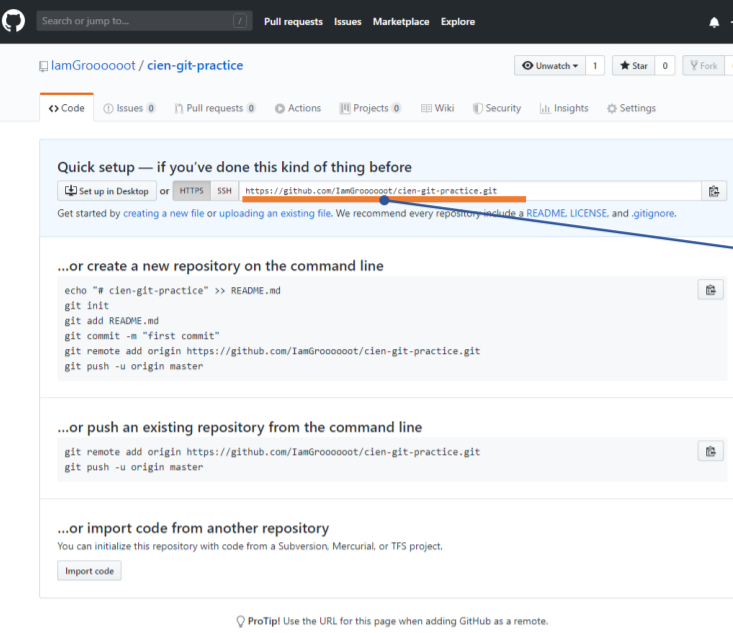
1. New repository 클릭

New repository

1. 저장소 명을 설정하고 Create repository 클릭

Create Repository

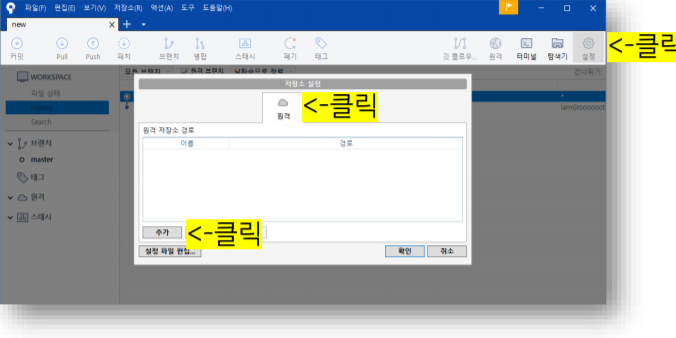
1. 방금 생성한 원격 저장소의 주소를 복사하자

방금 생성한 원격 저장소의 주소(링크)를 복사하자

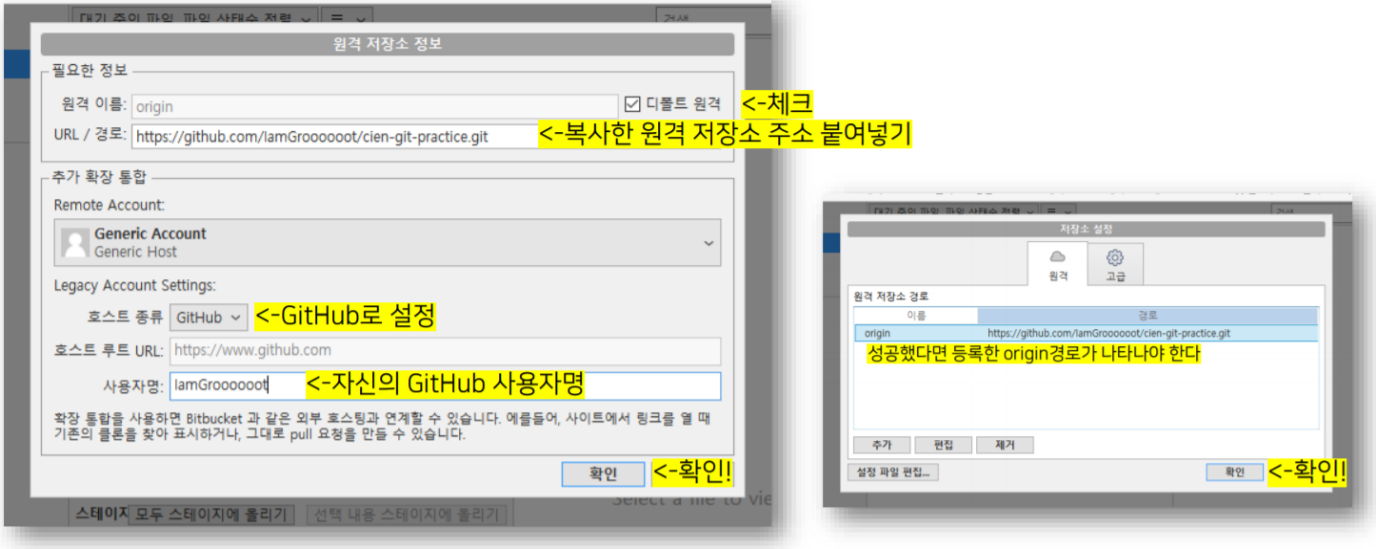
* 이 주소가 바로 내 원격 저장소의 링크이다!
* 이 주소로 추가하면 여기에 Push할 수 있다
* 이 주소를 공유해서 다른 사람들과 협업한다

1. Source Tree로 로컬 저장소와 GitHub 원격 저장소를 연결해보자

상단의 설정 버튼 > 원격 탭 > 추가 클릭

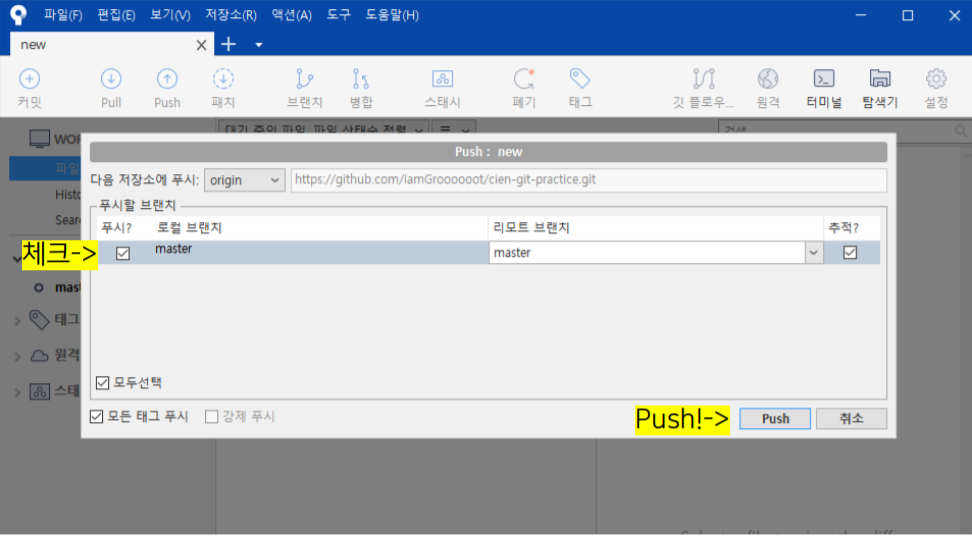
원격 저장소의 주소를 추가하는 창

내 기본(origin) 원격을 방금 만든 GitHub 저장소로 설정해주자

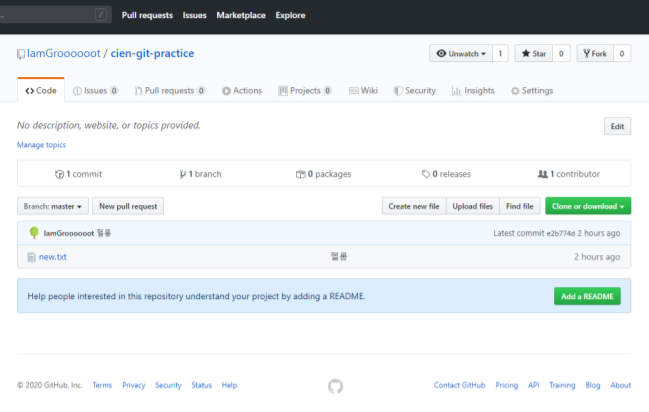
방금 복사한 원격 저장소 주소를 넣어주자

1. 다시 Push를 해보자

다시 Push클릭 후, main(또는 master) 브랜치 체크

Push해보자! (사진과 달리 신버전 Git을 사용하고 있다면 master가 아니라 main으로 나올 수도 있다)Push 성공!

1. GitHub 원격 저장소에 가서 잘 올라갔는지 확인해보자

GitHub에 만든 원격 저장소에 접속해서 확인해보자

매번 원격 저장소를 지정해줘야 할까?

아니다, 처음 한번만 origin 원격을 설정하면 .git 폴더에 저장된다

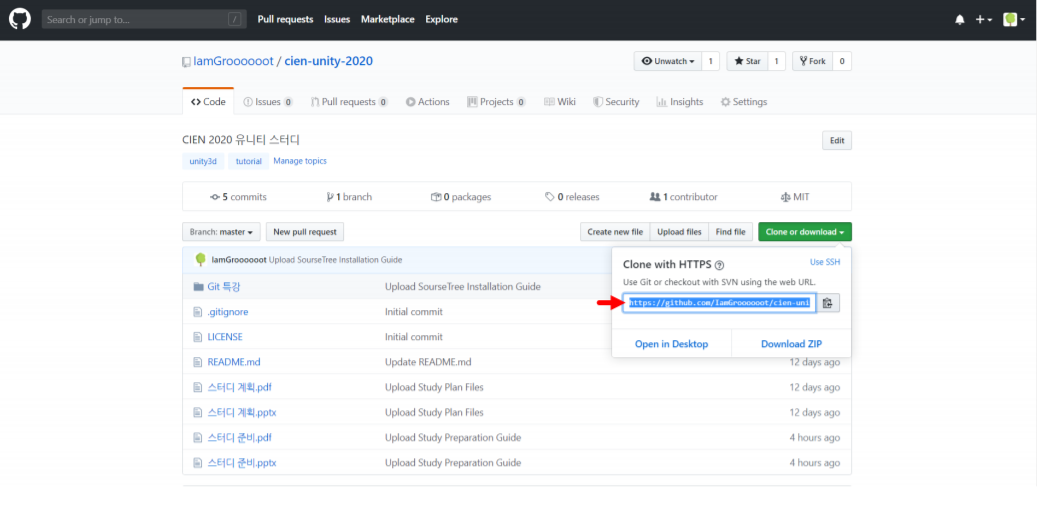
원격 저장소에서 다운로드한 저장소도 지정해주지 않아도 된다

우리가 방금은 로컬에서 저장소를 만들었다  
그렇기 때문에 Git은 원격 저장소의 주소를 몰랐다!  
저장소의 주소를 파악하고 오기 때문에 바로 Push 할 수 있다!  
그래서 처음부터 GitHub에서 만든 저장소 다운 받아서 쓰면 덜 귀찮다.  
이것을 복제(Clone)라고 부른다.

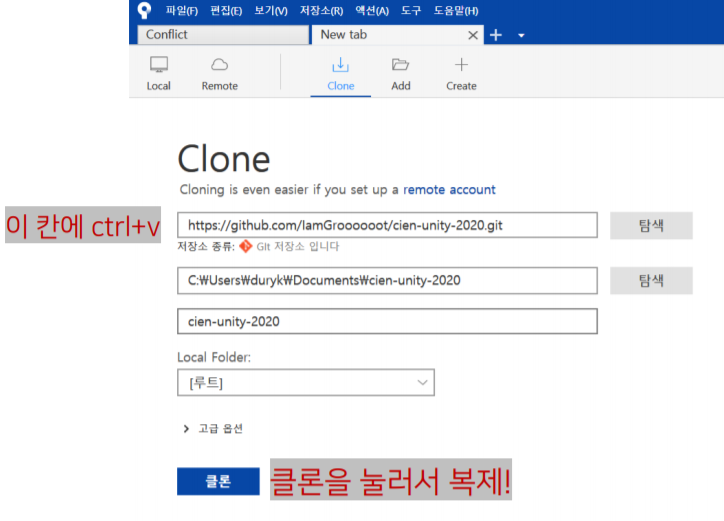
**원격 저장소를 복제해보자**

GitHub 저장소를 로컬에 복제해보자

1. 복제하고 싶은 GitHub 저장소로 이동한다
2. Clone or Download 클릭 후 HTTPS 링크 복사한다

원격 저장소의 주소(링크)를 Clone하기 위해 복서하자

1. 소스트리에 복제(Clone)하기

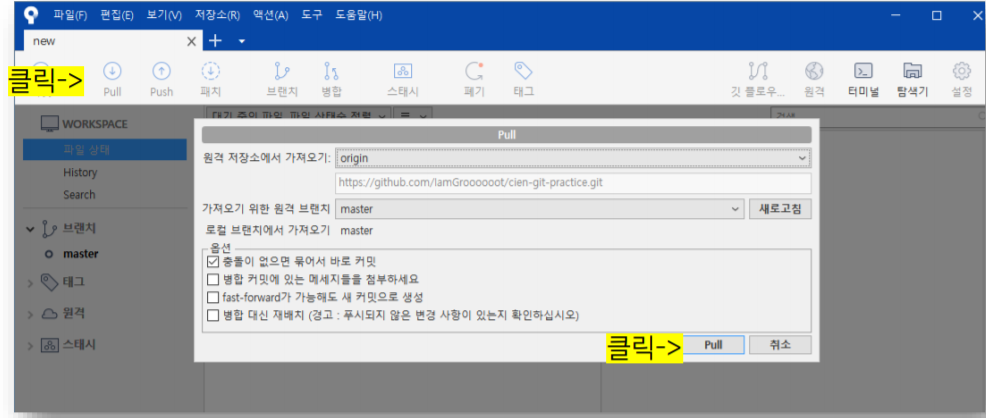
소스 트리 복제(Clone) 기능을 이용해서 다운 받아보자

새로운 탭에서 복제(Clone)를 누른 후 링크를 붙여 넣는다  
붙여 넣어도 클론 버튼이 비활성화되어 있다면 한번 클릭하면 활성화 된다

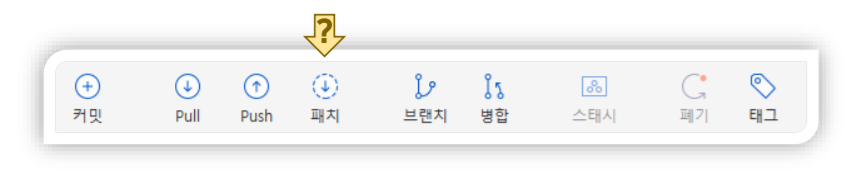
**Pull – 인터넷에서 새로 올라온 커밋 가져오기**

다른 동료가 Push를 했다면 내 로컬의 Git 저장소는 알려주기 전까지 새로 업로드된 파일들을 가지고 있지 않는다

* 변경된 사항은 원격 저장소로부터 다운 받아야 알 수 있다
* 그것이 바로 Pull이다  
  Pull 버튼을 클릭해보자

어떤 Commit이 새로 올라왔는지 확인하려면 Pull해보자

**Fetch란?**

Fetch 기능

Pull은 변경 사항을 다운받고 내 로컬 저장소에 적용까지 한다  
다운하고 자동으로 Merge까지 해준다 (뒤에서 함)

Fetch는 다운 받기만 하고 적용은 안 한다  
실제 파일은 변하지 않는다

따라서, Pull = Fetch(다운만 함) + Merge(합침)

실수를 할 수 있기 때문에 Fetch하고 Merge하는 것을 권장한다

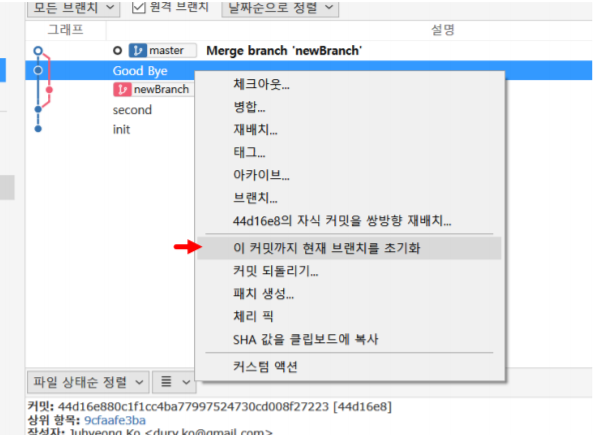
**Commit 취소하기 (주의)**

두 가지 방법을 소개하겠다

1. Revert
2. Reset

주의: 절대로 Push한 커밋은 수정하면 안 된다  
무조건 해야 한다면 팀원들에게 모두 알리고 Force Push하자

**Reset (초기화)**

선택한 커밋까지 Reset!Reset의 종류

Soft: HEAD만 옮긴다

* git reset --soft [커밋]

Hard: [커밋] 이후로 완전히 지운다

* 제일 강력: HEAD, Working Directory, Index(Stage Area)를 싹 다 지움
* git reset --hard [커밋]

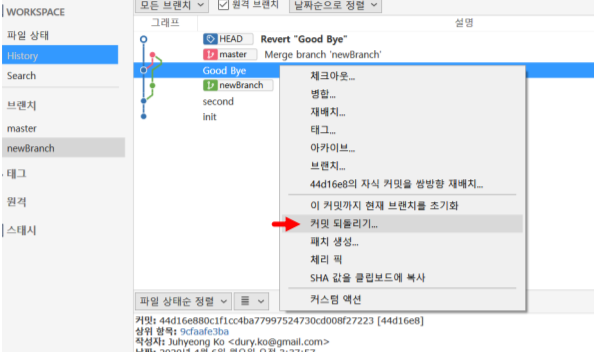
Mixed(기본): HEAD도 옮기고 Index도 바꾼다

* 기본값
* git reset --mixed [커밋]

**Revert (되돌리기)**

파일들을 이전 [커밋]인 상태로 되돌린 새로운 커밋을 만든다

* 이전 커밋을 수정하지 않고 그대로 남겨두기 때문에 안전하다.  
  Revert를 하고 난 다음에는 새로운 커밋이 생기기 때문에 바로 충돌 없이 Push할 수 있다
* git revert [커밋]
* 바로 커밋해주는게 싫다면 git revert -n [커밋]

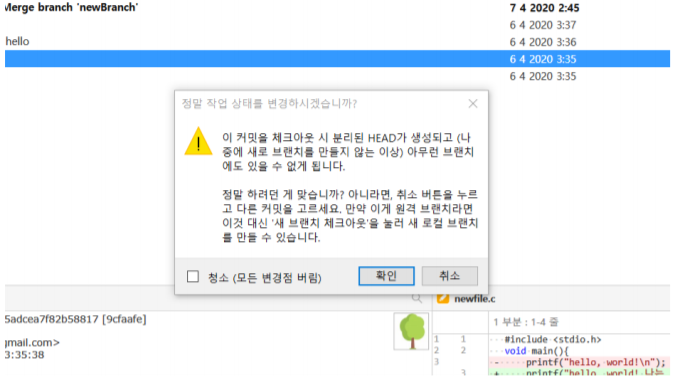
실수한 커밋도 남기는 Revert 기능

**Checkout Commit - 예전 커밋 구경하기**

Commit은 다른 Commit끼리 구별하기 위한 SHA값을 부여받는다

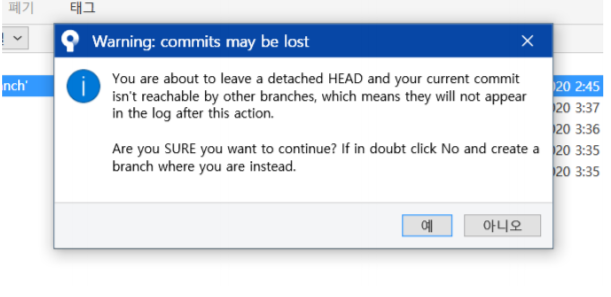
SHA 값으로 예전 커밋을 가져와서 확인할 수 있다

* 소스 트리는 더블 클릭하면 알아서 해준다
* Detached HEAD 상태가 되는 것을 주의해야 한다



**Detached Head 상태**

HEAD가 분리됐을 때이다



HEAD가 커밋을 가리키므로 브랜치에 커밋 할 수 없게 된다. 이때는 일시적인 커밋만 가능하다.  
브랜치가 가리키는 최신 커밋을 더블 클릭함으로써 해결 가능하며 그렇게 되면 일시적인 커밋은 다 날아가게 된다.

**이전 Commit 메시지 수정**

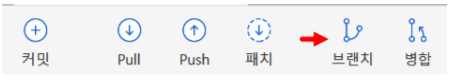
git commit --amend or git reset --soft

(커밋 버튼 위) "커밋 옵션..."에서 "마지막 커밋 정정" 클릭

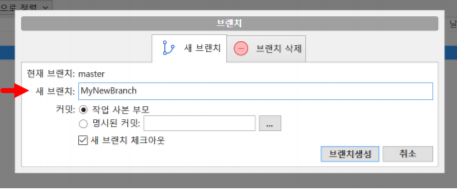


**작업 환경 나누기(Branch)**

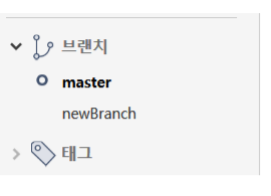
브랜치 생성 클릭



새 브랜치 이름 설정하자



옆에 브랜치 목록에서 원하는 브랜치를 클릭해서 이동할 수 있다  
main(또는 master) 브랜치는 처음에 자동 생성되는 기본 브랜치이다. (예전에는 기본 브랜치의 이름이 master였다)

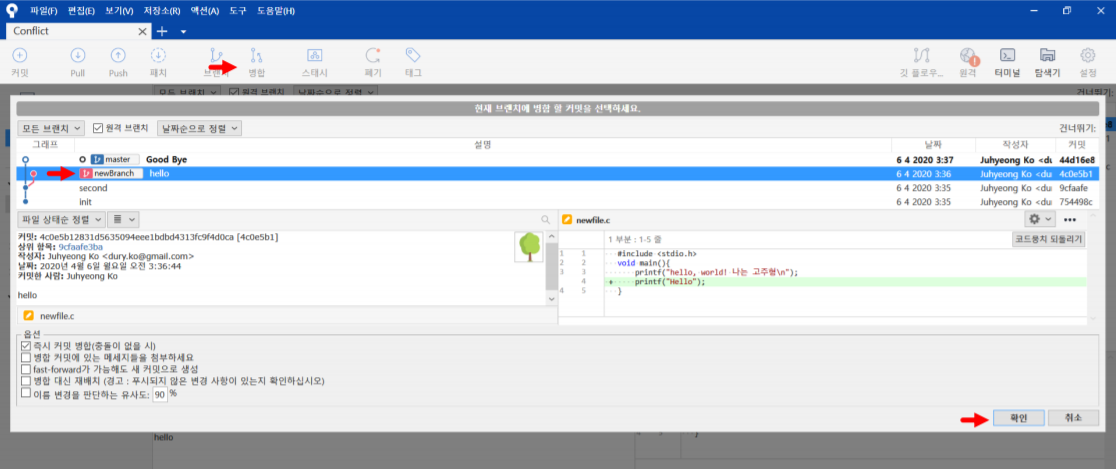


내가 브랜치를 쓸 때는 아래와 같다.

* Bug Fix
* 새로운 기능을 추가할 때
* 내 컴퓨터에서만 테스트할 코드
  + 보통 이런 코드는 푸쉬하지 않는다
* …

**작업한 것 합치기(Merge)**

병합 클릭 -> 현재 병합할 브랜치 선택  
주의! 병합할 브랜치랑 병합될 브랜치를 절대 혼동하지 말자



**Merge에 관해서**

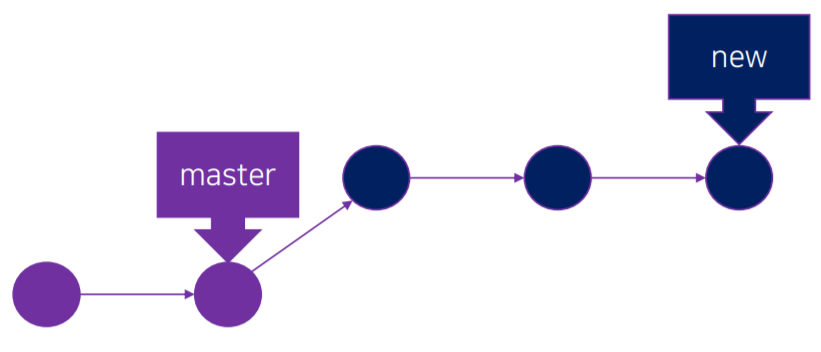
두 종류의 Merge

1. Fast Forward Merge
2. True Merge (흔히, 3-Way Merge라고 부른다)

Fast Forward Merge?

Merge긴 Merge인데 뭔가 인정해주고 싶지 않은 가짜 Merge

커밋이 한 라인에 있을 때 합쳐 주는 것을 Fast Forward Merge라고 부른다.  
기존 커밋에 새로운 기능들만 추가되거나 삭제 됐으니 다른 코드와 충돌할 일이 없다.

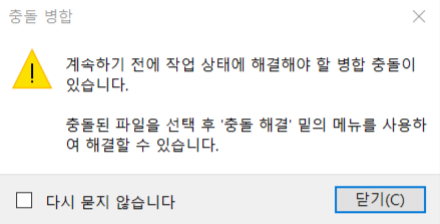


True Merge (3-way merge)?

진정한 Merge

Base, 브랜치1, 브랜치2  
=> 이렇게 3가지를 비교하면서 동작하여 3-Way Merge라고 흔히 부른다

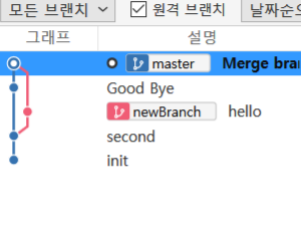
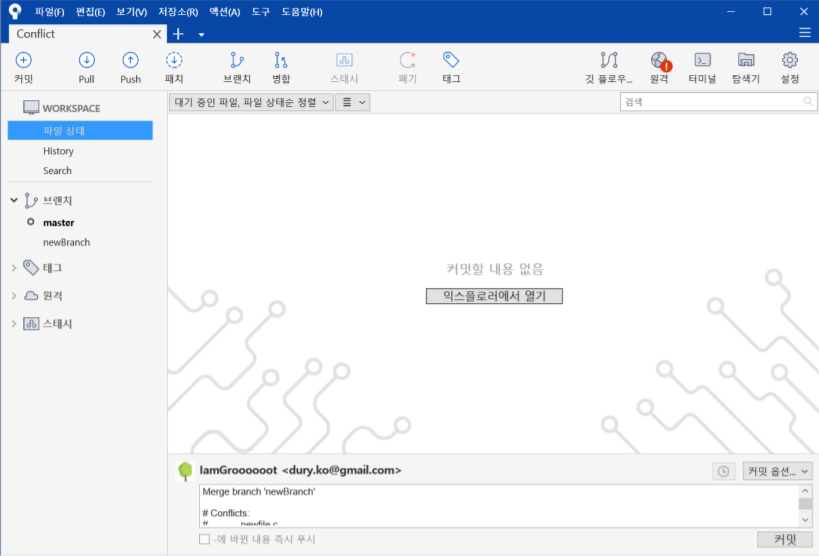
충돌(Conflict)가 발생할 수 있다  
충돌은 Git이 알아서 코드들을 합쳐주지 못 할 때 발생한다. 이럴 때는 내가 직접 무슨 코드를 남겨둘지 결정해줘야 한다.



**충돌(Conflict) 해결하기**

개인적으로 Source Tree의 충돌(Conflict) 해결 기능은 별로라고 생각한다...

Source Tree로 돌아와서 병합한 파일을 커밋해서 Conflict 해결



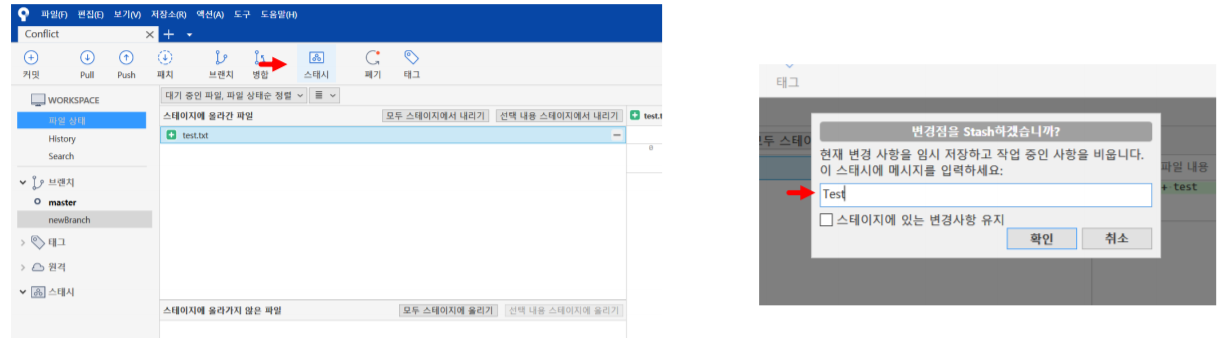
**Stash 사용법 - 흔히 발생하는 문제들을 해결해보자**

Stash는 Git에서 제공해주는 임시 인덱스 저장소

Stage Area(Index)에 있는 파일을 잠시 저장해 둘 수 있는 저장소이다  
Stack이다

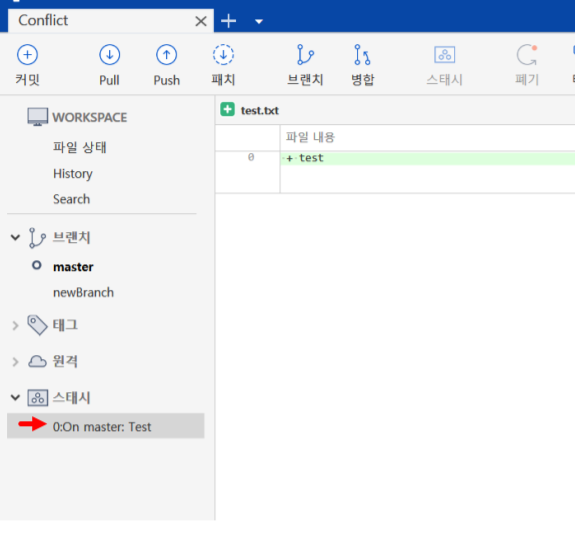
* Push, Pop
* LIFO(Last In First Out)

1. Stash 저장  
   스태시 누르고 저장 메시지 남기기



1. 저장한 것 확인

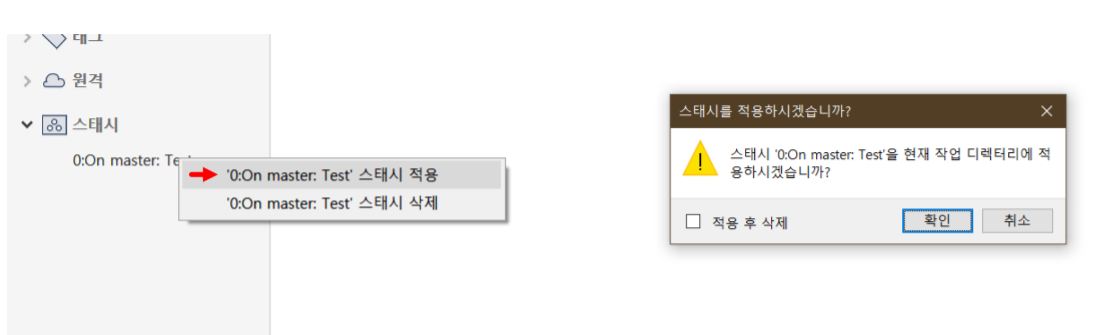
왼쪽 스태시 메뉴에서 확인



1. 적용(불러오기)

적용하고 싶은 스태시 왼쪽 클릭 후 스태시 적용

* 적용하고 삭제하고 싶은 경우 "적용 후 삭제" 체크 (= POP)



**문제 1. Pull할 때 가끔 발생하는 오류**

|  |  |
| --- | --- |
|  | Error: Your Local Changes to the following files would be |
|  | overwritten by merge. |
|  | Please commit your file or stash them before you can merge. |

예를 들어서 내가 만료된 옛날의 저장소에서 Pull하려고 한 경우가 있다.

Stash를 사용한 간단한 해결법이다.

1. 현재 Index를 Stash에 저장  
2. Pull한다  
3. Stash 적용  
4. Commit  
5. Push

**문제 2. 다른 브랜치에서 작업한 것 가져오기**

Feature 브랜치에 커밋해야 할 것을 Master 브랜치에서 작업한 경우 Feature 브랜치에 작업한 커밋을 Stash를 이용해서 Master 브랜치로 가져올려면 어떻게 해야 할까?  
  
0. (커밋한 경우) Mixed Reset  
1. Stash에 저장  
2. Commit할 브랜치로 이동  
3. Stash 적용  
4. Commit