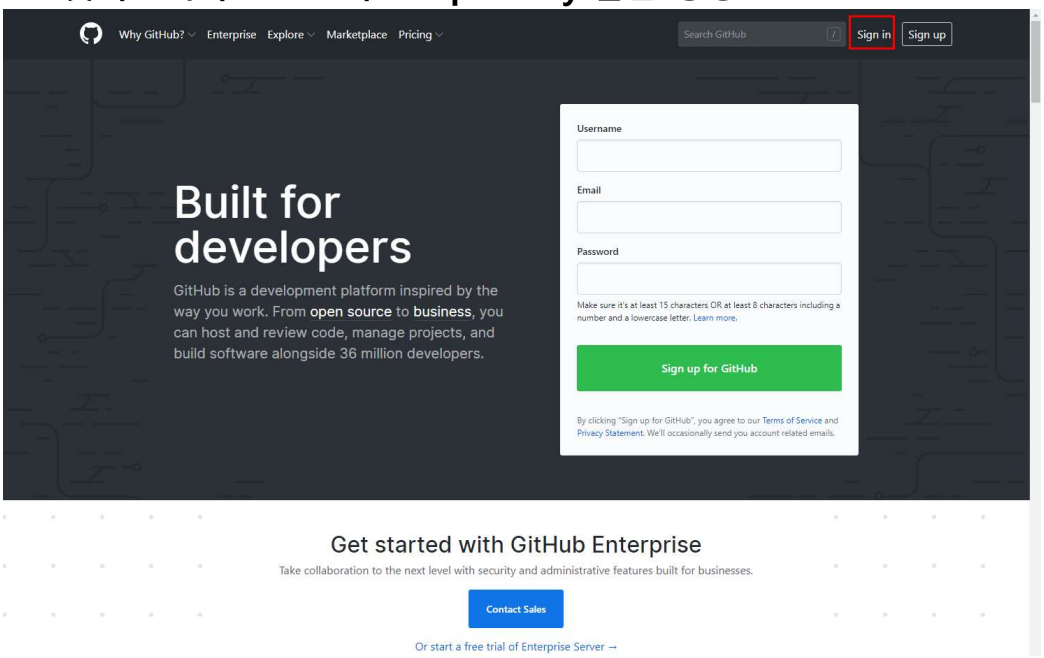
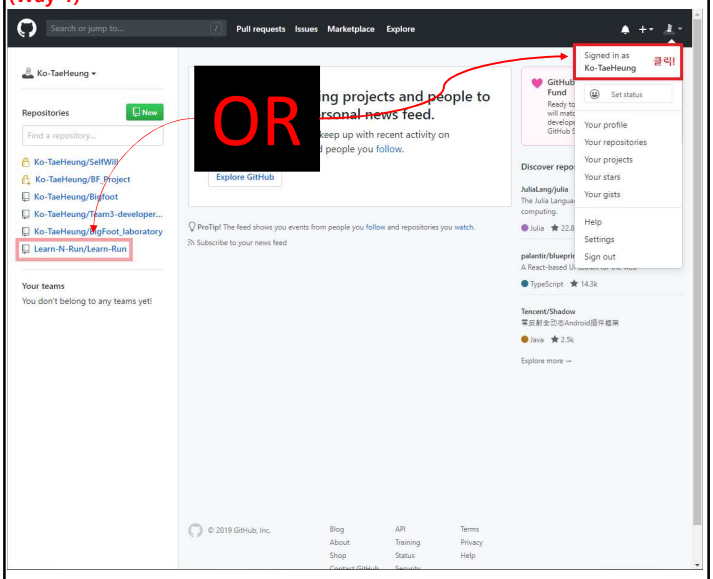
Contents List

Ⅰ. 깃허브에서 소스트리로 repository 클론 생성

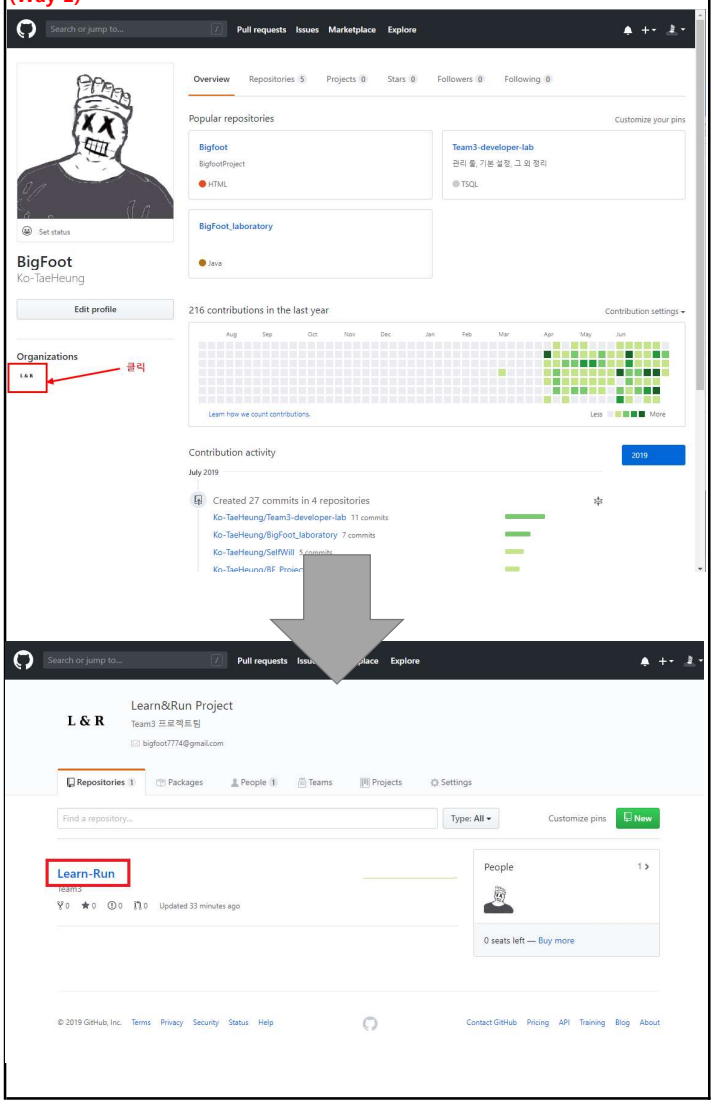


▲ 1. 깃허브 사이트에 접속하여 로그인 해주세요

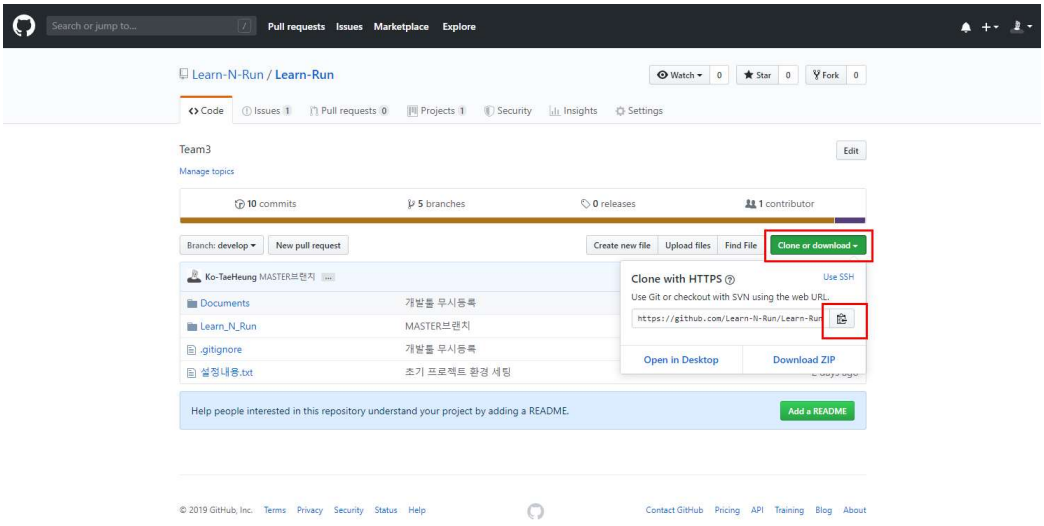
(Way 1)



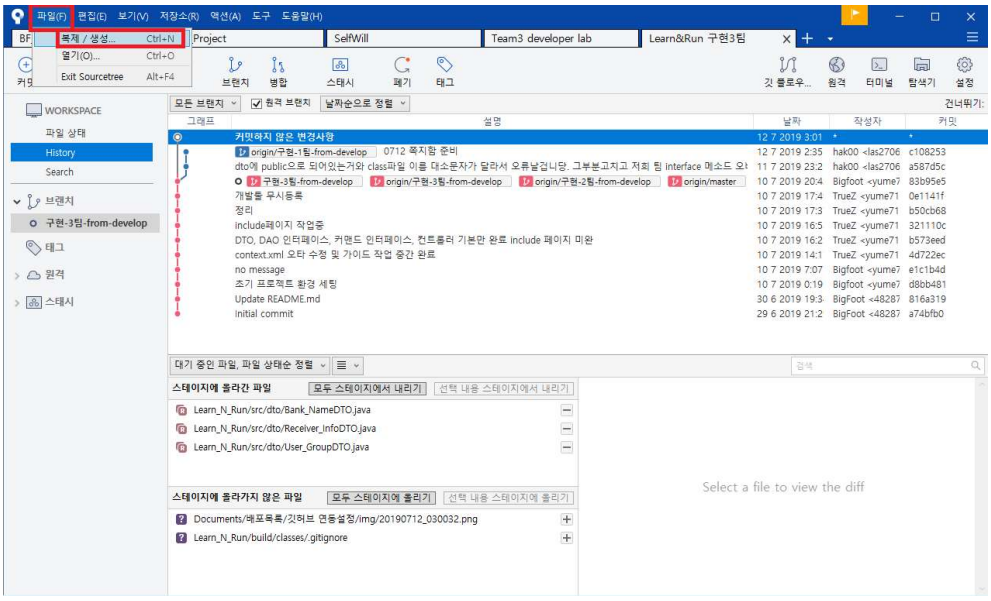
(Way 2)



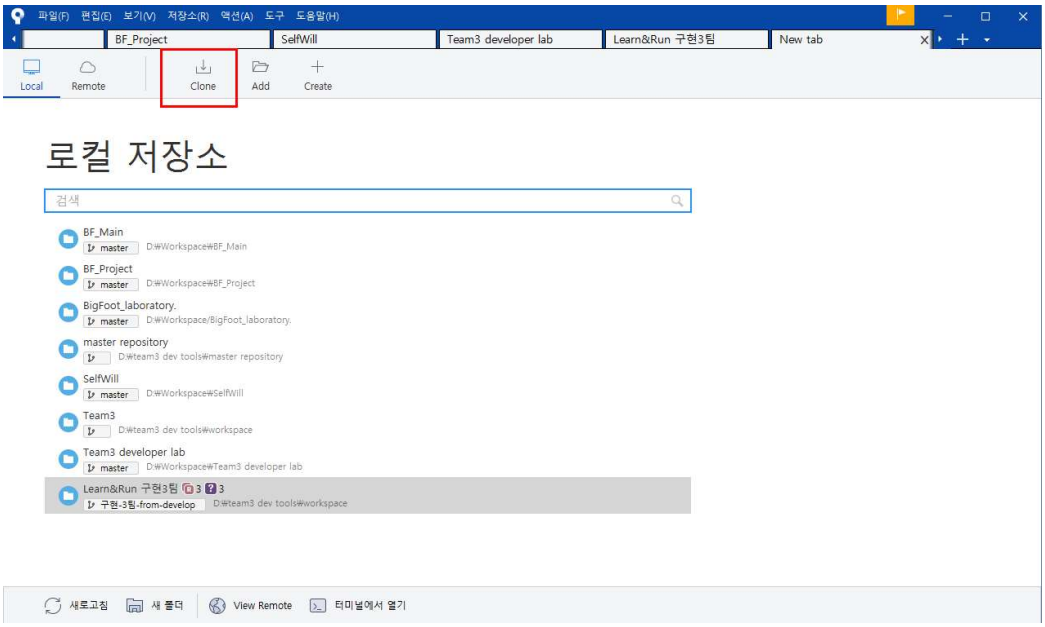
▲ 2. 가입하신 Learn-N-Run조직의 Learn-Run레파지토리로 접속해주세요!
이미 조직멤버로 초대가 다 끝났기 때문에 왼쪽의 레파지토리에서 Learn-N-Run/Learn-Run으로 바로 들어가주셔도 되시고
오른쪽 상단을 클릭하셔서 그림과 같이 조직을 클릭하셔서 접근하셔도 됩니다.



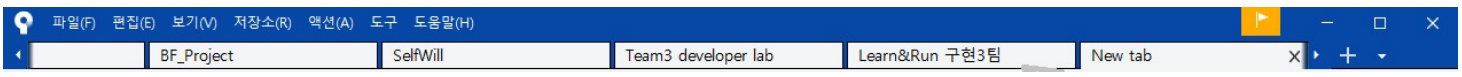
▲ 3. 접속하신 레파지토리의 우측에서 Clone or download버튼을 눌러 현 레파지토리의 url주소를 복사해줍니다.

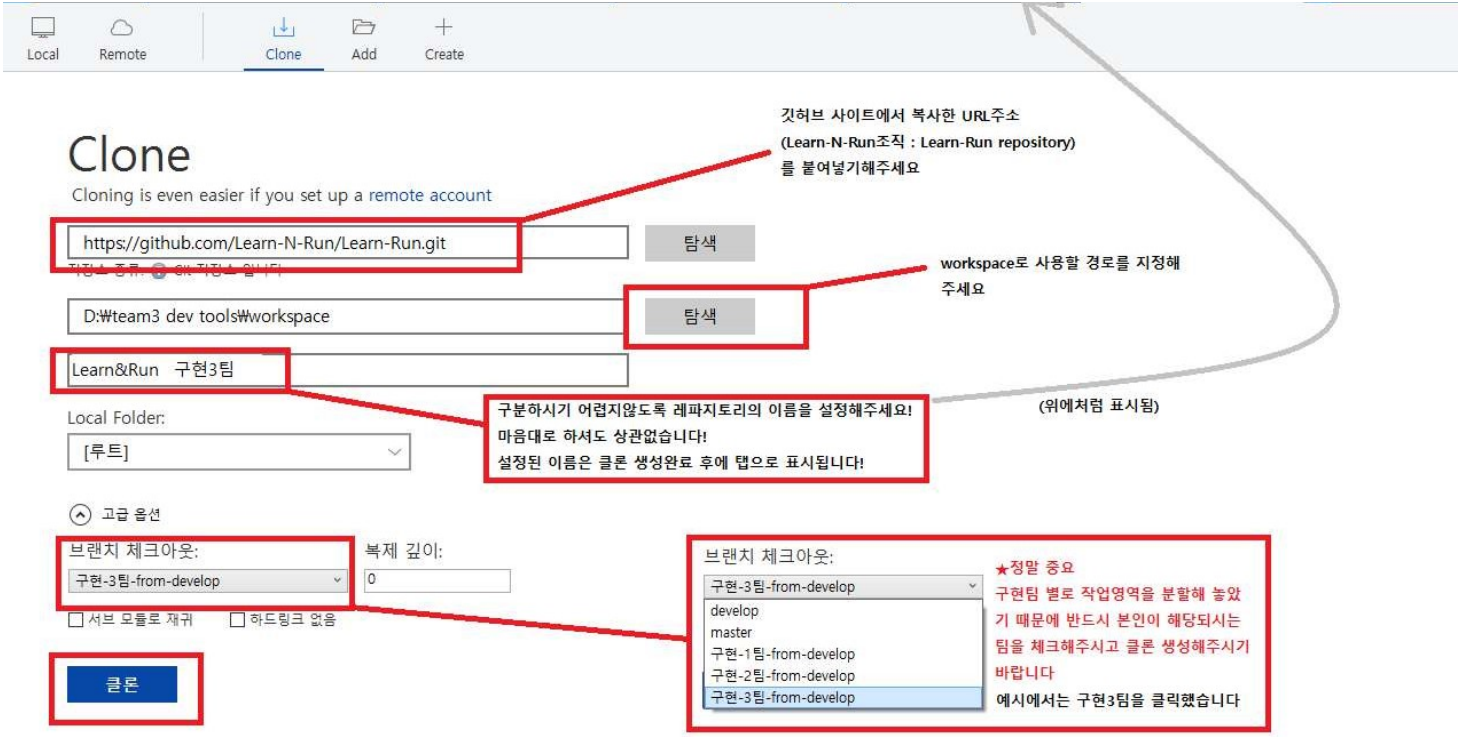


▲ 4. 다운로드하신 소스트리의 좌측 상단에 파일→복제/생성 버튼을 눌러줍니다.

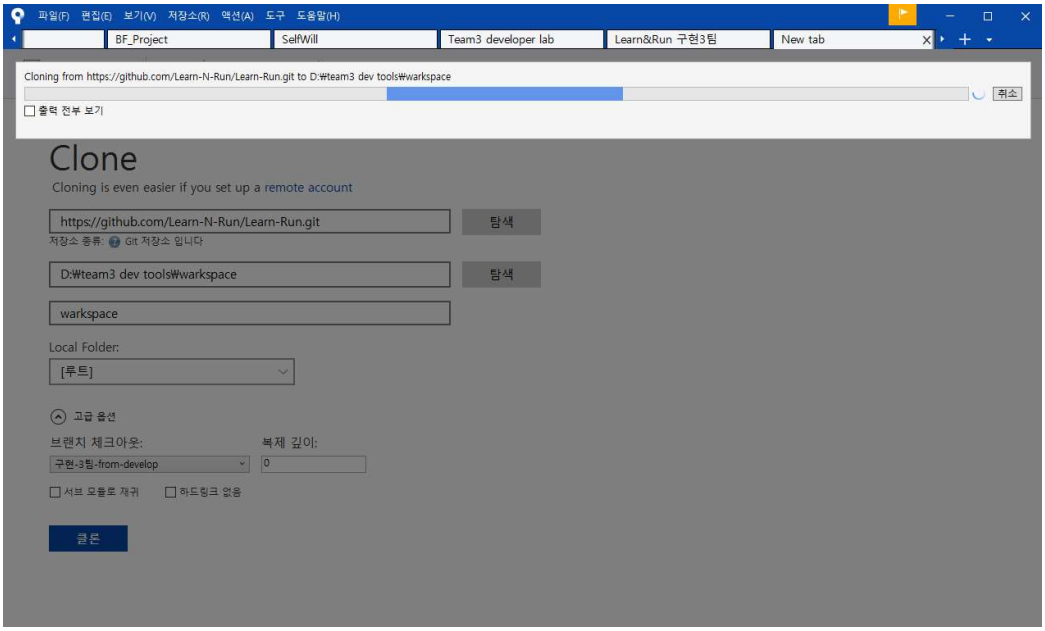


▲ 5. 이어서 상단의 클론버튼을 클릭해주세요!



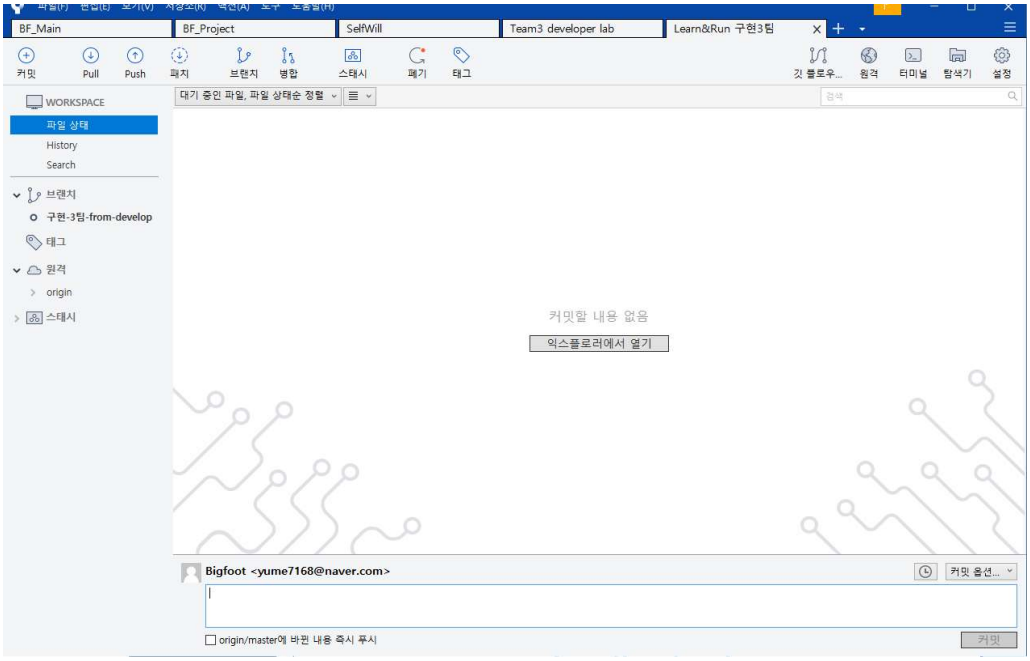


▲ 6. 깃허브 레파지토리에서 클론생성시 설정해주셔야 될 기본 항목들입니다.
(1) 복사하신 url을 붙여넣어주시고, (2) 클론으로 복제할 본인 컴퓨터의 경로를 지정해주시고(프로젝트 진행 때 워크스페이스로 쓰일예정),
(3) 고급옵션 항목에 브랜치체크아웃을 본인팀에 맞게 꼭 설정해주시기바랍니다.(정말 중요)



▲ 7. 설정을 완료 후 클론 버튼을 눌러주시면 이제 복제가 시작됩니다.
실행환경에 따라서 30분, 길게는 1시간까지도 걸리는 것 같습니다. 저희 집은 1분만에 바로 되던데...(핑계)
복제가 완료될 때까지 식사도 하시고 연락 못한 애인이랑 통화도 하시고
저는 그 동안 연락하지 못했던 친구 혹은 부모님께 연락을 드려보며 피곤한 마음을 달래보았습니다.

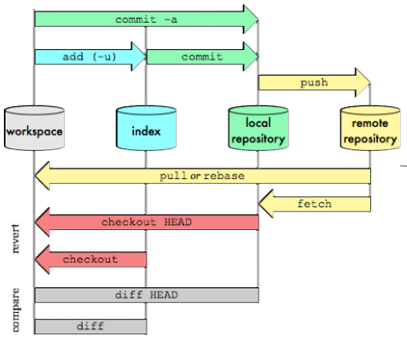
▲ 8. 심심해서 넣어보았습니다



▲ 9. 드디어 완료가 되었습니다!

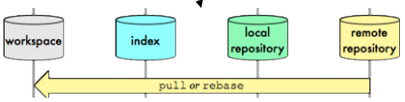
그럴일은 없지만 혹시 오류가 났는지 확인을 위해 소스트리 좌측의 history내역을 확인해보시고
클론생성에 지정했던 경로로 접근하여 디렉토리에 파일이 제대로 들어갔는지 한번 확인해 주시면 클론생성이 끝납니다
수고하셨습니다!

Ⅱ. PULL



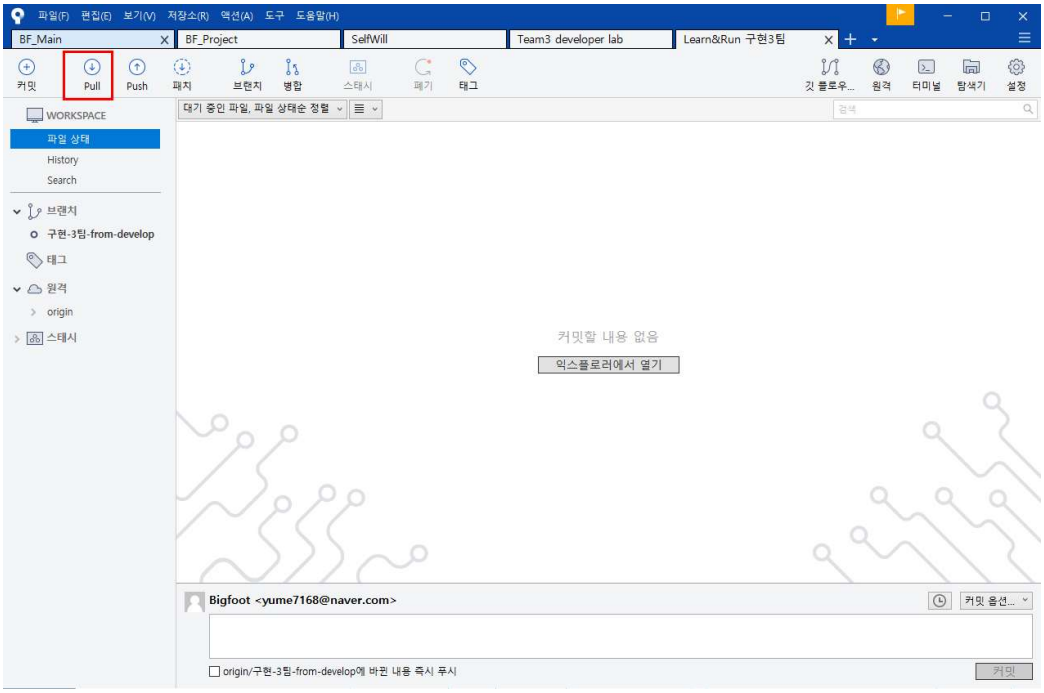
▲ process of git

이해하시기에 이 이미지와
매칭이 어려운 부분이 있을 수 있습니다.
넓은 아량으로 개략적인 과정만 살펴봐주시고
이해해주시면 감사하겠습니다. 죄송합니다(ㅠㅠ)

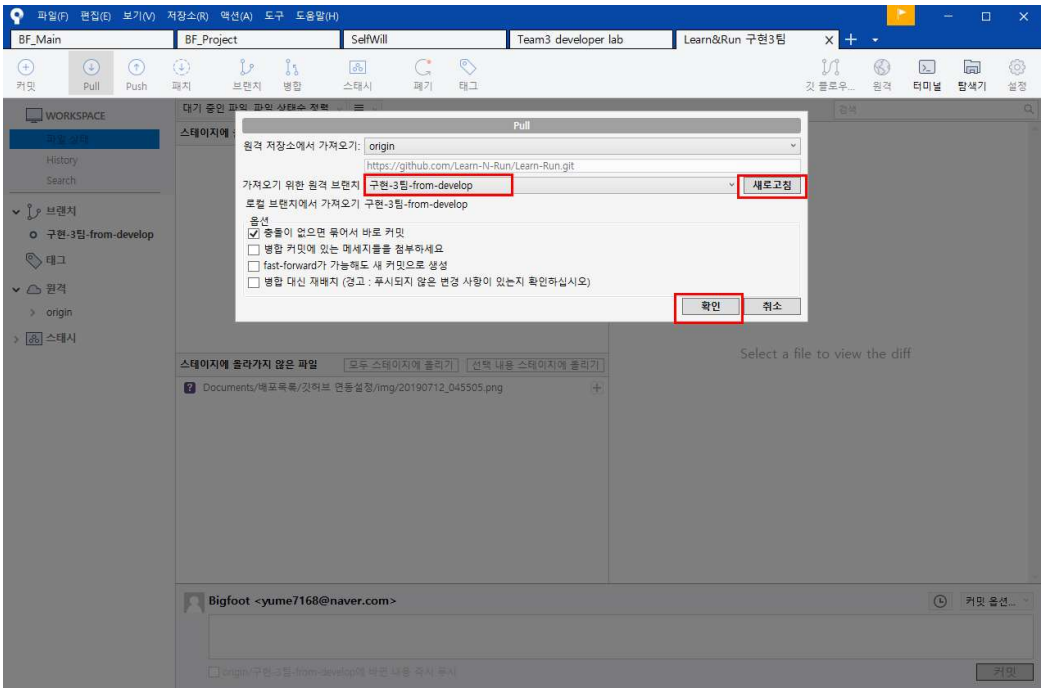


▲ PULL

- 원격 레파지토리(remote repository = git, 여기서는 git을 github와 같다봐도 무방합니다)에서 현재 컴퓨터로 레파지토리 파일을 내려받는 것이 PULL입니다.
여기서는 PULL방법에 대해 기술하겠습니다.

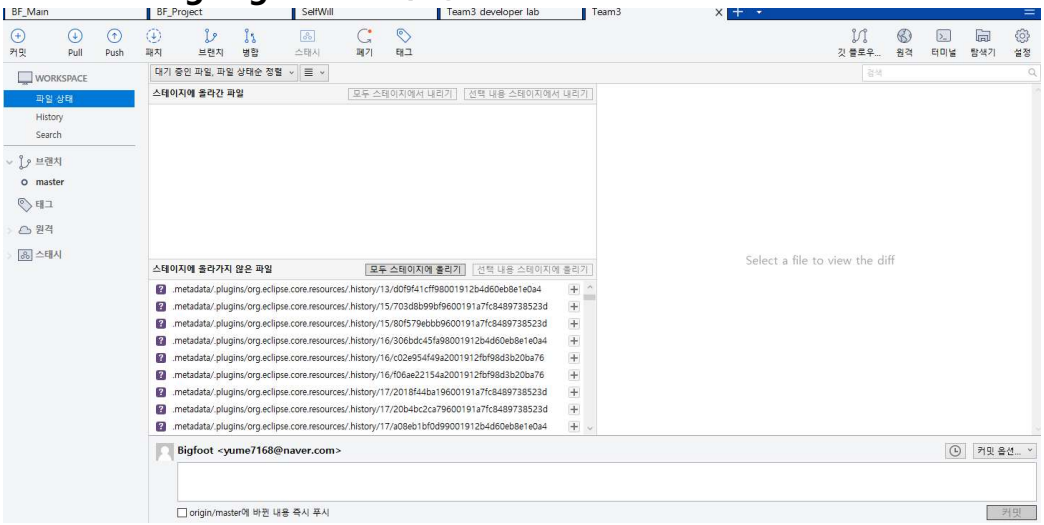


▲ 1. 좌측 상단의 PULL버튼을 눌러주세요

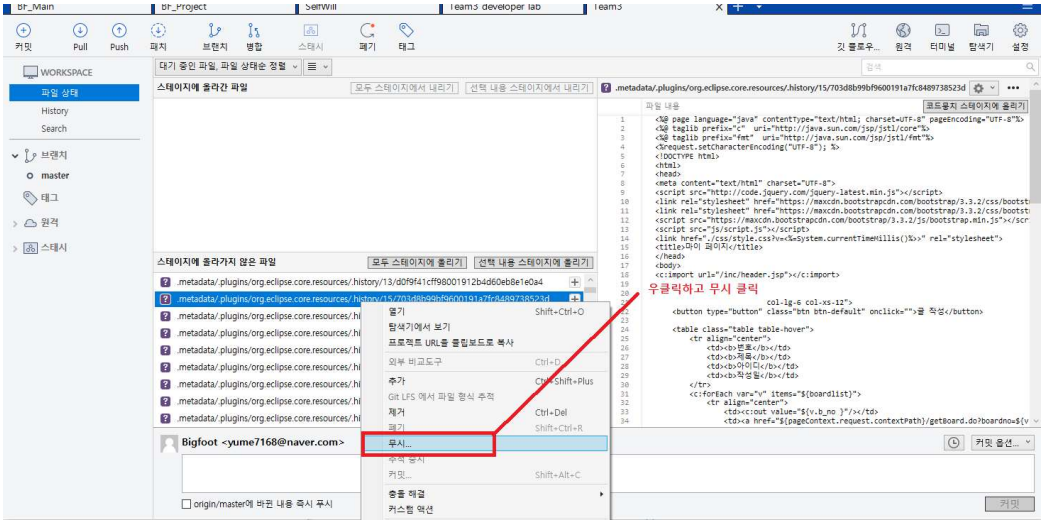


▲ 2. 클릭하시면 창이 하나 뜨는데 한 번 정상적으로 이미 클론을 생성했던 상황이라면 원격브랜치와 옵션이 자동으로 지정됩니다.
혹여나 "가져오기 위한 원격 브랜치"의 커서가 본인이 속한 구현팀이 아닐 경우에는
우측의 새로고침을 누르신 후 커서의 드롭박스를 활성화하여 다시 지정해주시고 확인해주시면됩니다.
끝입니다. 수고하셨습니다!

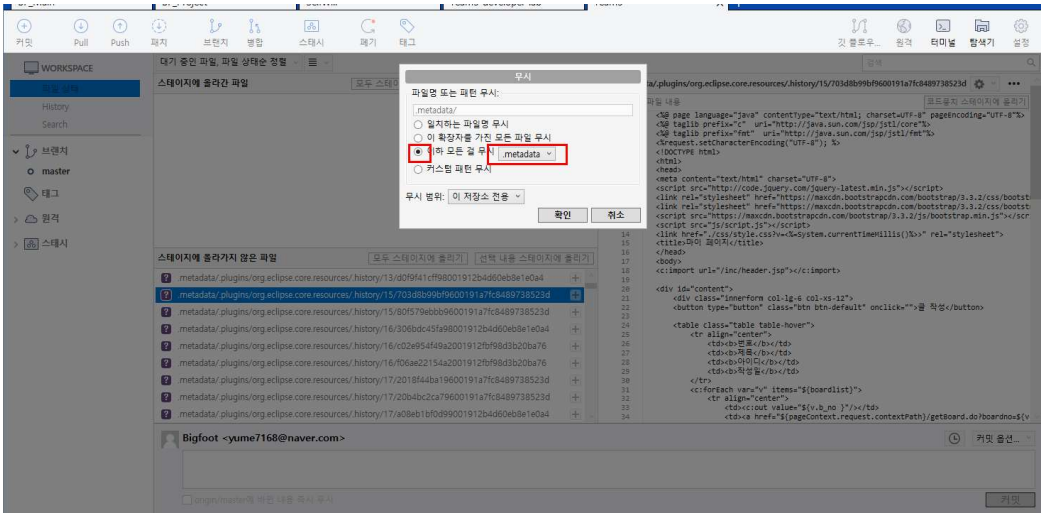
Ⅲ. PUSH - git ignore 등록 방법



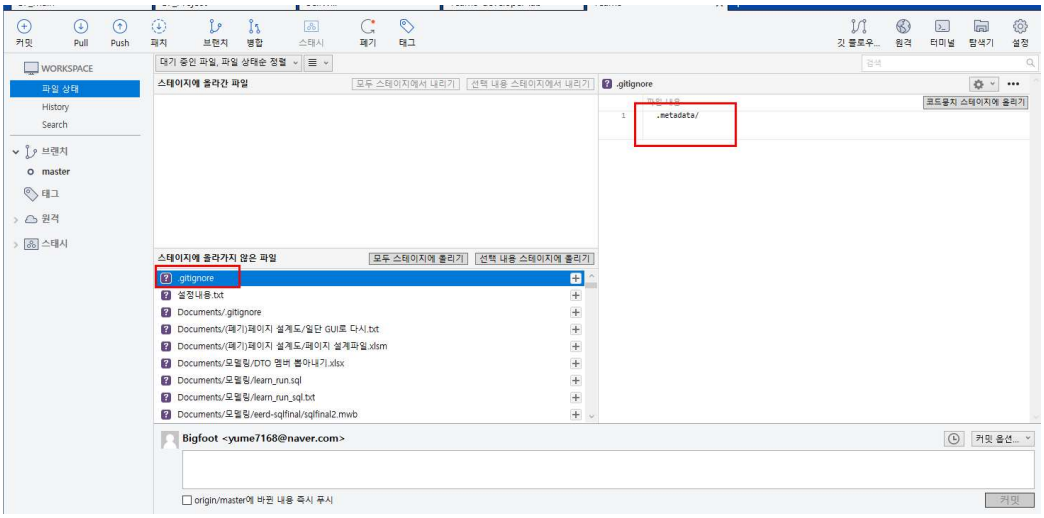
▲ 1. 프로젝트를 시작하면 파일에 코드가 추가되고 그 추가된 코드들은 우리가 git에 PUSH할 파일들이 됩니다.
하지만 코드가 추가되는 건 우리가 작업하는 java혹은 컴파일 된 class파일뿐만 아니라
테스팅을 위해 연결한 톰캣 서버파일, 이클립스 IDE에서 프로젝트마다 자동생성되는 메타데이터도 똑같이 해당됩니다.
그렇기 때문에 PUSH를 할 때 위 이미지와 같이 굳이 필요 없는 목록들이 뜨는데 이 목록들만 제외해 줄 수 있게 하는 것이
git ignore입니다.
여기서는 git ignore의 간단한 설정방법을 기술하겠습니다.



▲ 2. 필요없는 파일항목을 오른쪽 마우스 클릭해주시고 무시버튼을 눌러주세요
예시에서는 IDE에서 자동으로 생성되는 metadata 디렉토리를 예시로 들었습니다.

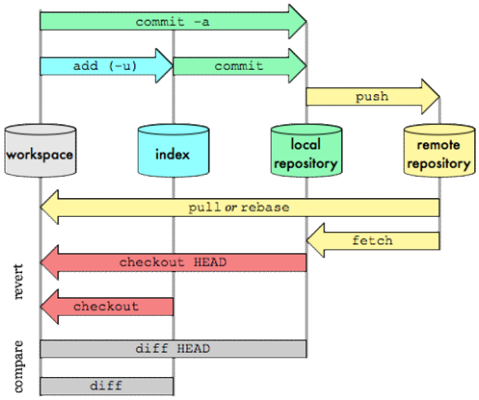


▲ 3. 소스트리 내부에 창이 하나 팝업되는데 예시의 metadata같은 경우에는 3번째 "이하 모든걸 무시"로 설정하여 드롭박스를 체크해주었습니다.
원하시는 설정이 끝나시면 확인을 눌러주세요

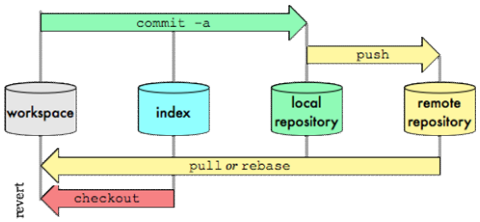


▲ 4. 확인을 누르시면 레파지토리 내에 git ignore 파일이 하나 생성되며
그렇게 많아서 골치였던 metadata가 싹다 사라졌습니다.
이렇게 한 번 설정해주시면 다음에 커밋할 때 다시 설정할 필요가 없으니 꼭 설정해주시기바랍니다
수고하셨습니다!

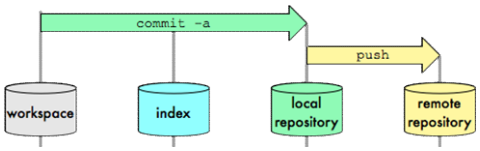
IV. PUSH - COMMIT(커밋)과 PUSH(푸쉬)



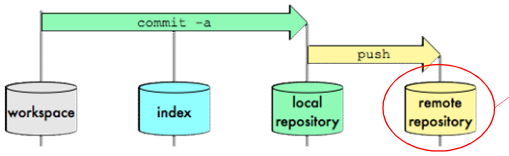
▲ process of git
git의 기본 프로세스입니다.
위 내용을 다 아시면 좋지만 전부 공부하시기에는
공부할 수록 내용이 너무 많고 복잡합니다



▲ 저희 프로젝트에서는
몇몇 특수한 상황을 제외하고는 이 부분만 활용합니다

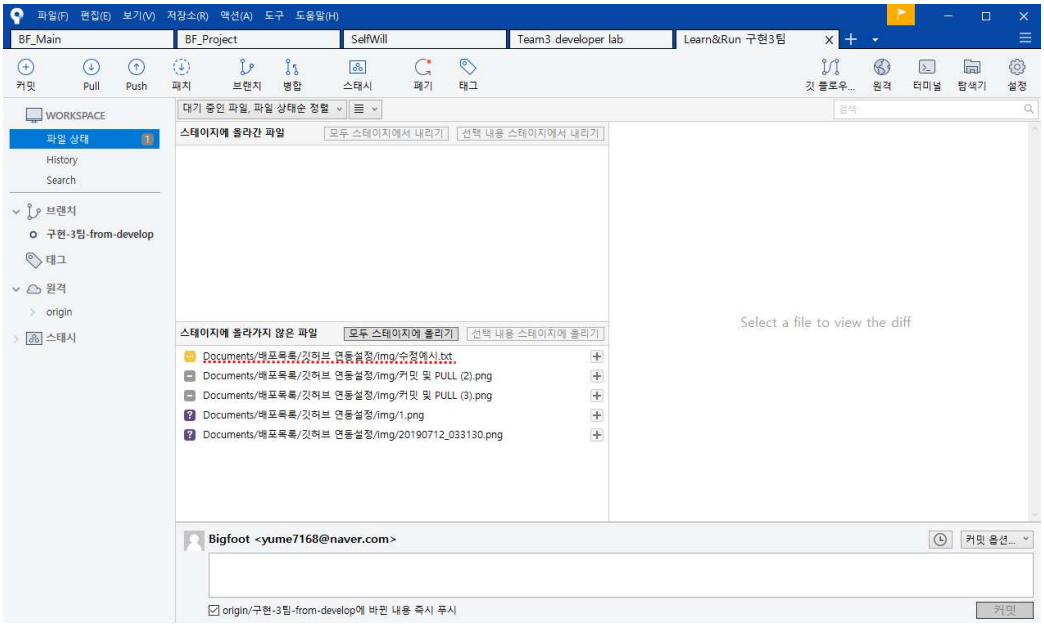


▲ 그 중에서도 이번에는 버전을 저장하고,
원격저장소에 저장한 버전을 업로드(갱신)하는
COMMIT과 PUSH방법에 대해 기술해보겠습니다.

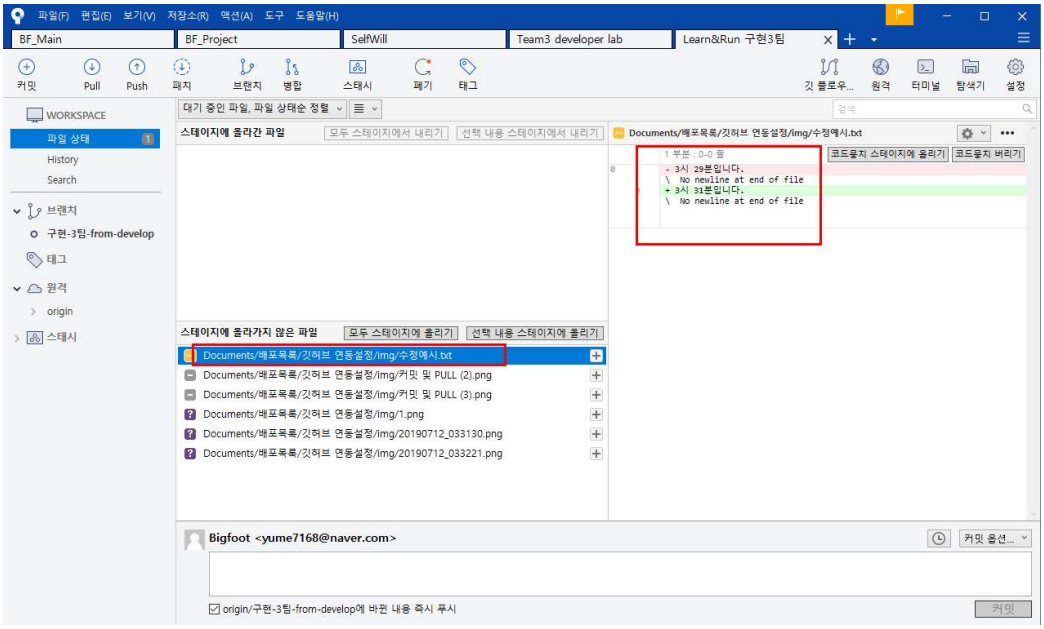


▲예시 1. 예시에서 현재 remote repository(원격 저장소 = git)에
"3시 29분입니다"라는 내용의 "수정예시.txt"라는 텍스트파일이 저장되어있음
2. 이 파일의 내용을 3시 31분입니다라고 수정 후 원격저장소에 이 내용을 업로드할 예정

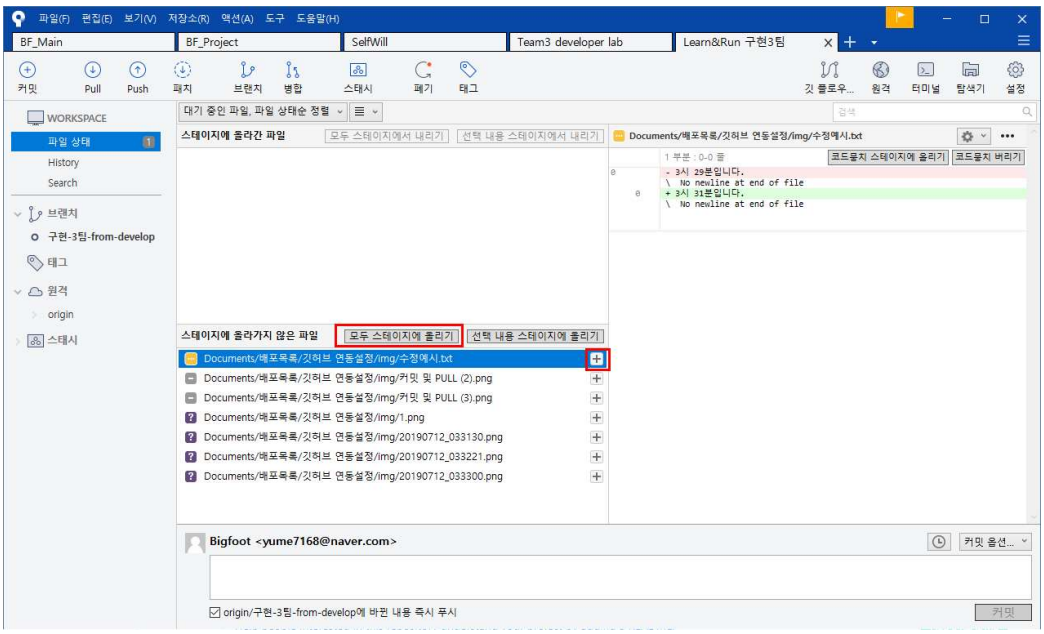
▲ 1. 파일의 내용을 수정하였습니다
(예시에서는 텍스트파일이지만 텍스트파일이 아닌 java파일 혹은 기타 jsp파일로 봐주시면 좋겠습니다)



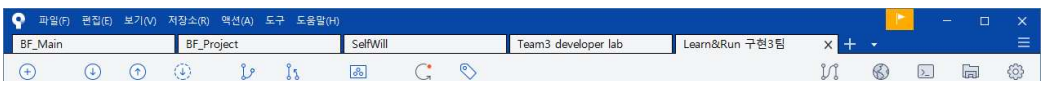
▲ 2. 수정한 파일의 버전 업데이트(커밋)를 위해 소스트리를 실행하여 해당하는 레파지토리(Learn&Run)탭을 활성화 시켜주었습니다.
위와 같이 "스테이지에 올라가지 않은 파일"목록에 수정예시파일을 포함하여 해당되는 파일을 확인하실 수 있습니다.

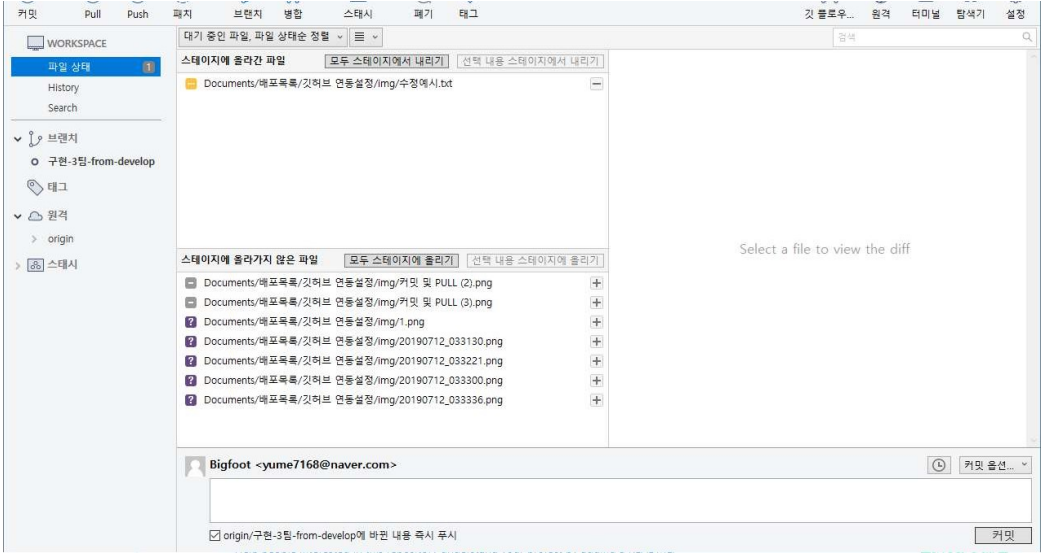


▲ 3. 클릭하시면 수정내역도 우측에서 확인하실 수 있습니다.(물론 만능은 아니고 뷰에서 지원이 안되는 파일확장자명도 있습니다)



▲ 4. 현재 예시에서는 "수정예시.txt" 파일만을 버전 업데이트(커밋)하기 위해 해당되는 파일의 우측 +표시를 눌러줬습니다.
파일을 한 번에 모두 올리실 경우에는 "모두 스테이지에 올리기"버튼을 눌러주시면 됩니다.

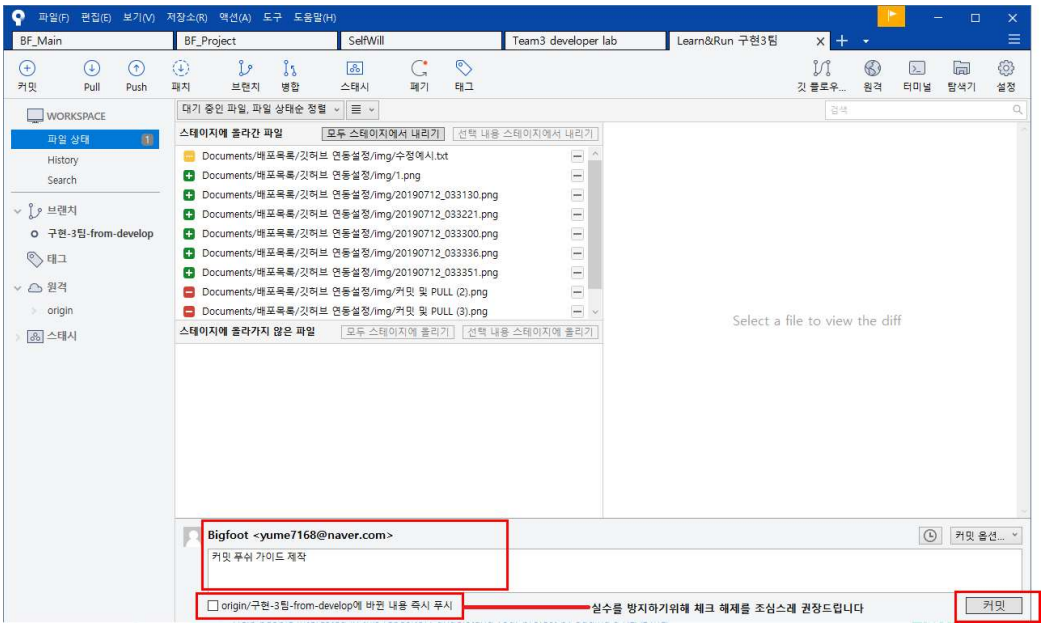




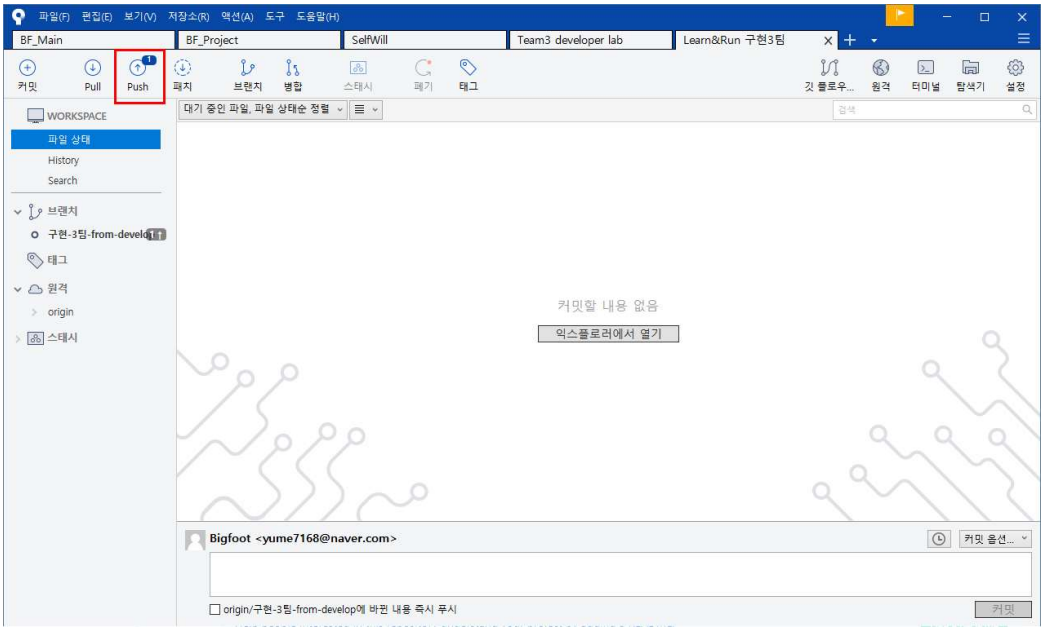
이 부분에 알아주셨으면 하는 점은
여러 파일이 전체적으로 수정되었더라도
특정 파일만 커밋을 주어
버전을 분류할 수 있다는 점입니다.

커밋 내용은 해당 소스를 이용하는 사용자,
혹은 협업하는 동료에게도 유용한 정보이니
구분하셔야 될 상황이 발생하신다면
이 점 참고해서 활용해주시면
정말 감사하겠습니다.

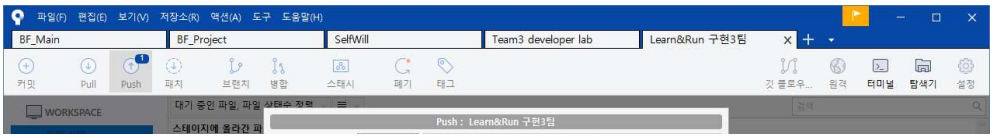
▲ 5. 우측 +버튼을 클릭하니 이렇게 해당파일만 "스테이지에 올라간 파일"의 목록으로 올라갔습니다.



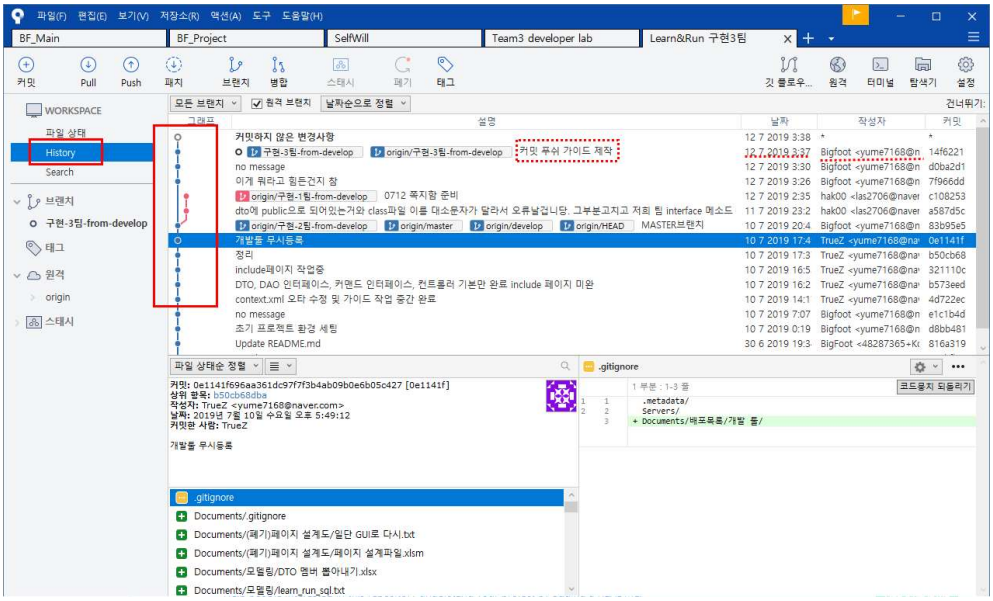
▲ 6. 근데 하나씩 올리기 귀찮아요. 어짜피 다 커밋해야되서 방금 예시로 하나 올려봤고 현재 파일을 전체 다 올렸습니다.
버전 업데이트하실 파일을 스테이지에 올리신 후에 하단에 해당 커밋(버전 업데이트)에 대한 설명을 적어주시면됩니다.
그 후 커밋버튼을 눌러주세요



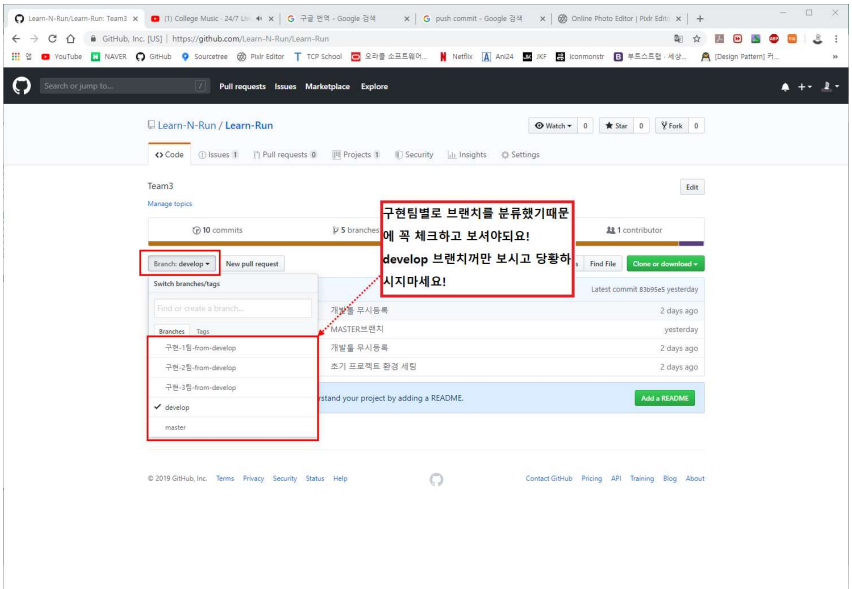
▲ 7. 정상적으로 커밋이 완료되었습니다.
상단 좌측 push란에 1이라는 아이콘이 같이 표시되는데 여기서는 보통 아시겠지만 커밋을 했지만 push가 되지않은 '수'를 표시합니다.
push를 위해 저 버튼을 눌러보겠습니다.



▲ 8. 클릭하시면 창이 이렇게 뜨는데 리모트 브랜치(원격 저장소 = git)에서 해당되는 팀이 맞는지 확인하시고
push버튼을 누르시면 push가 완료됩니다.
(간혹 에러가 발생하며 push가 진행되지 않는다면
작업 직전 PULL이 정상적으로 되지않은 상태에서 PUSH를 진행하였거나
구현팀 내의 팀원 분과 작업 부분이 겹쳐 충돌이 발생하는 경우가 많습니다.
팀원 분과 같이 상이해보시고 그래도 찜찜하다하시면 저한테 연락주시면 감사하겠습니다)

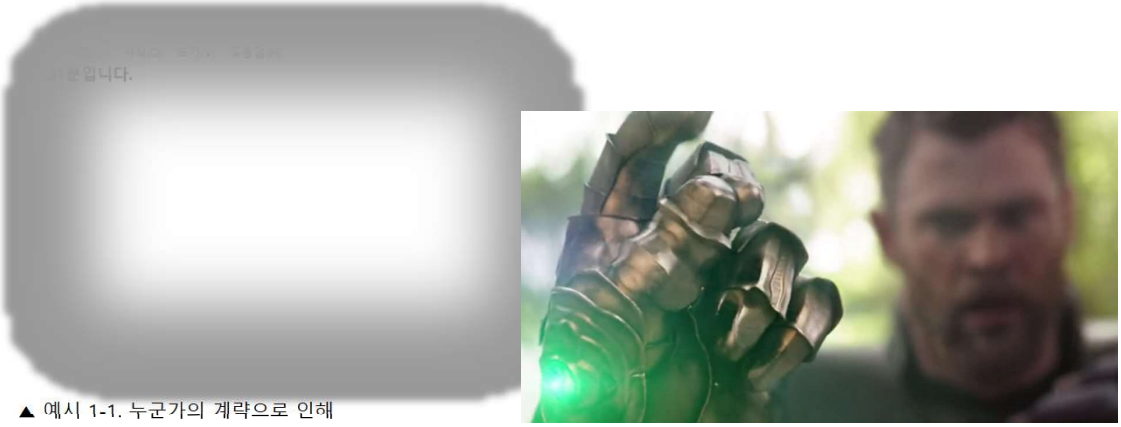


▲ 9. PUSH버튼을 누르고 정상적으로 완료가 되었습니다!
소스트리의 history에서 커밋하셨던 작업내용을 확인하시고

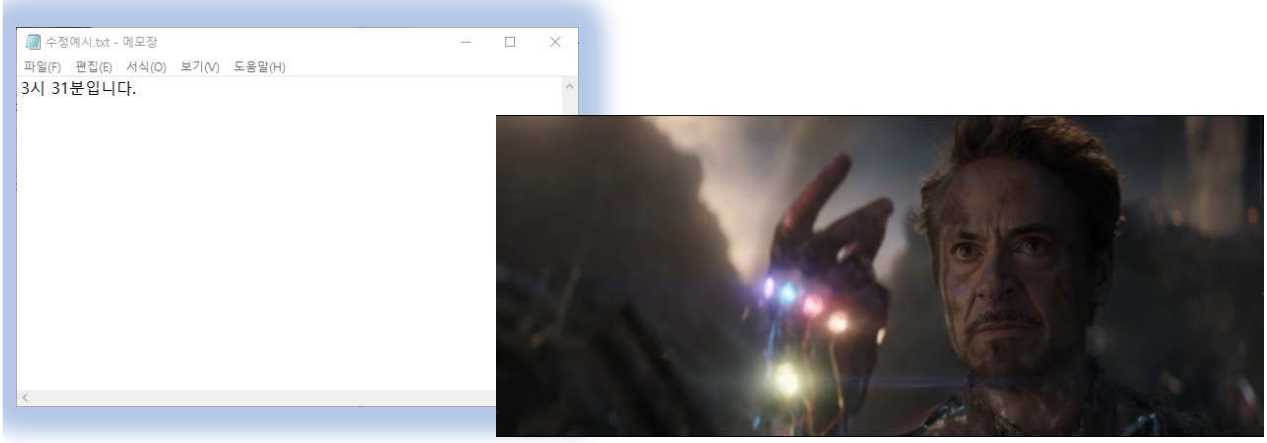


▲ 10. github의 저희 조직 레파지토리 해당 브랜치에서 정상적으로 PUSH 완료가 확인되신다면 commit과 push는 여기서 끝입니다.
수고하셨습니다.

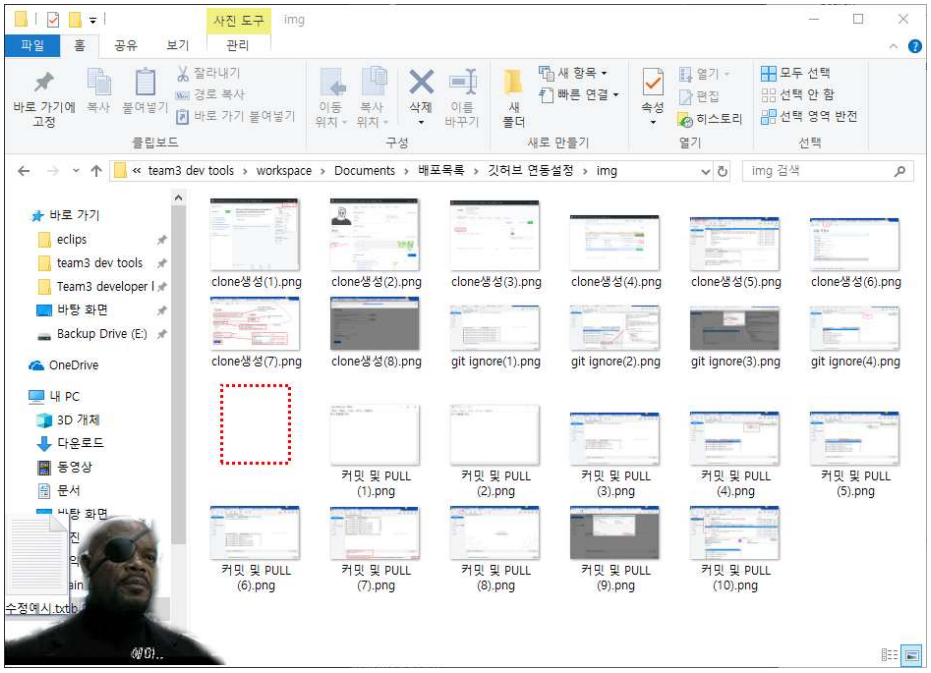
V. FILE RECOVERY - 커밋했던 예전 버전의 파일로 복구



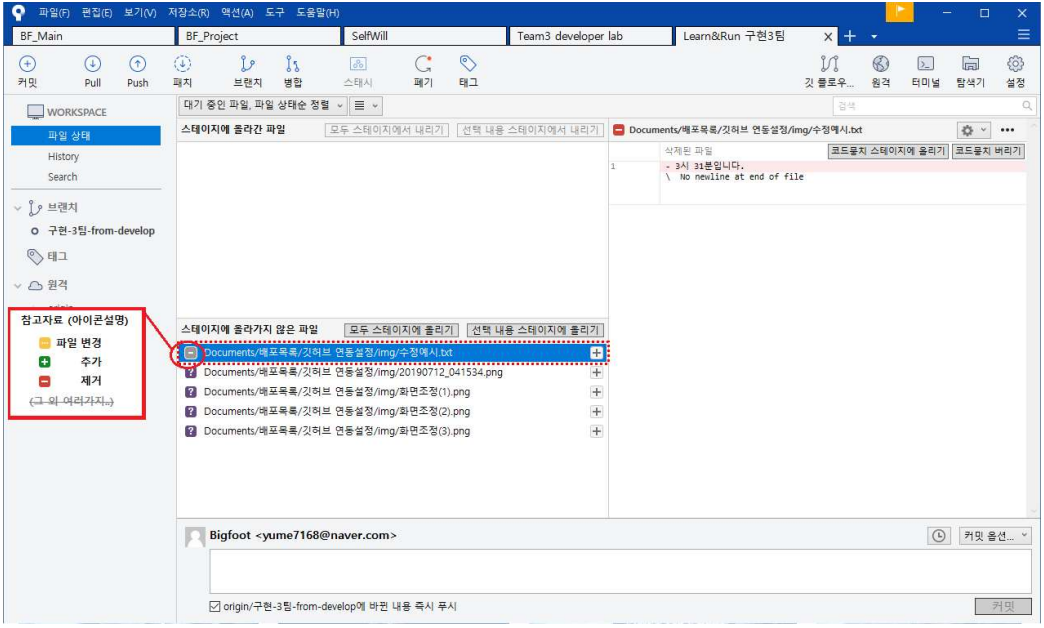
▲ 예시 1-1. 누군가의 계략으로 인해
4장에서 업로드했던 "수정예시.txt"파일이 삭제되었습니다.



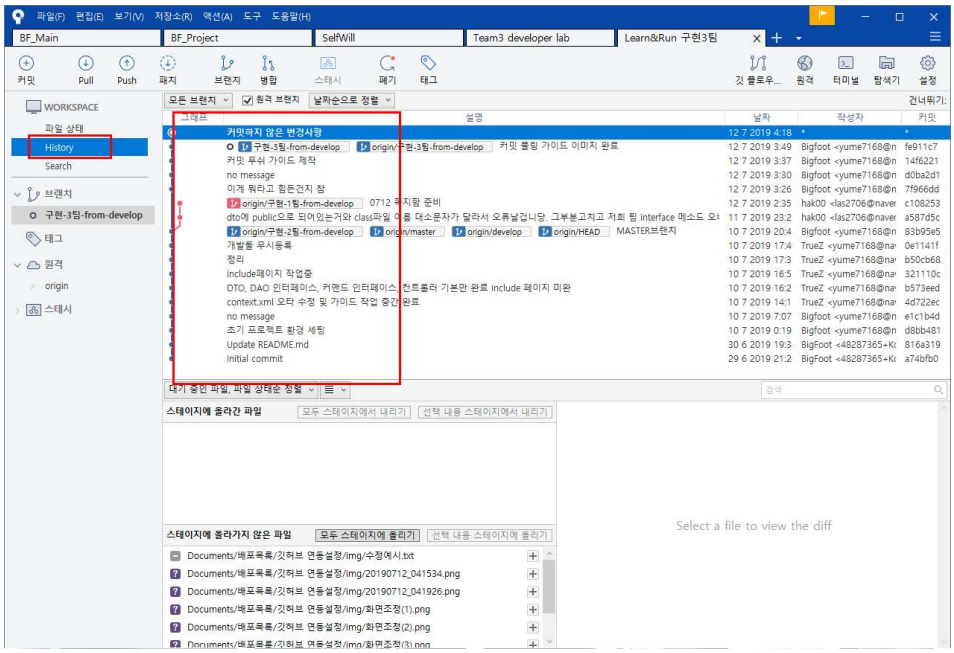
▲ 예시 1-2. 이번에는 삭제(혹은 의도치않게 수정이 되버린) 파일의 복구방법에 대해 기술하겠습니다.



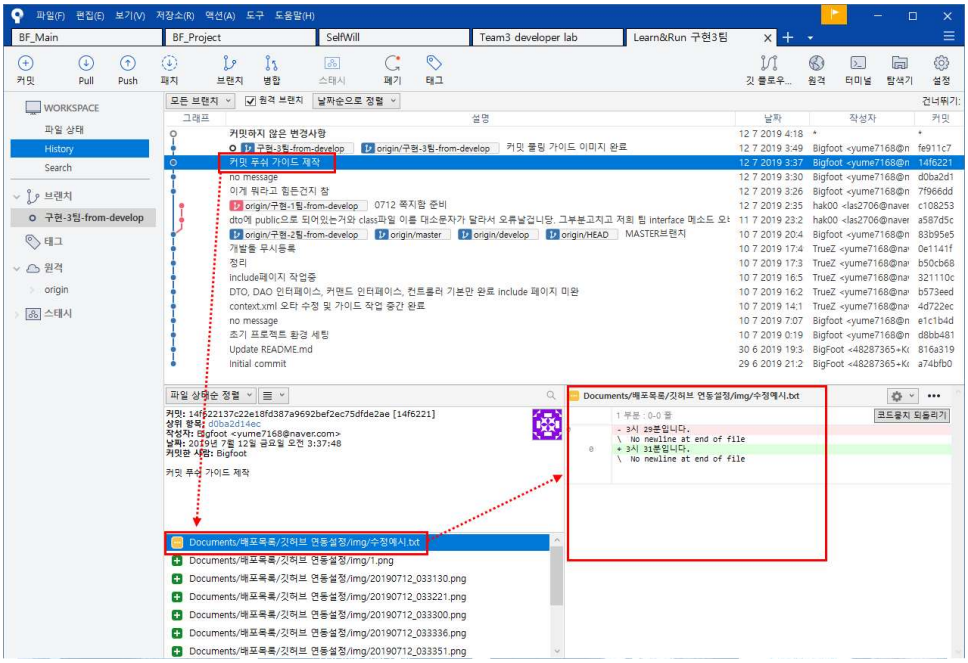
▲ 1. 위에서 삭제 당한 "수정예시.txt"의 디렉토리폴더입니다.



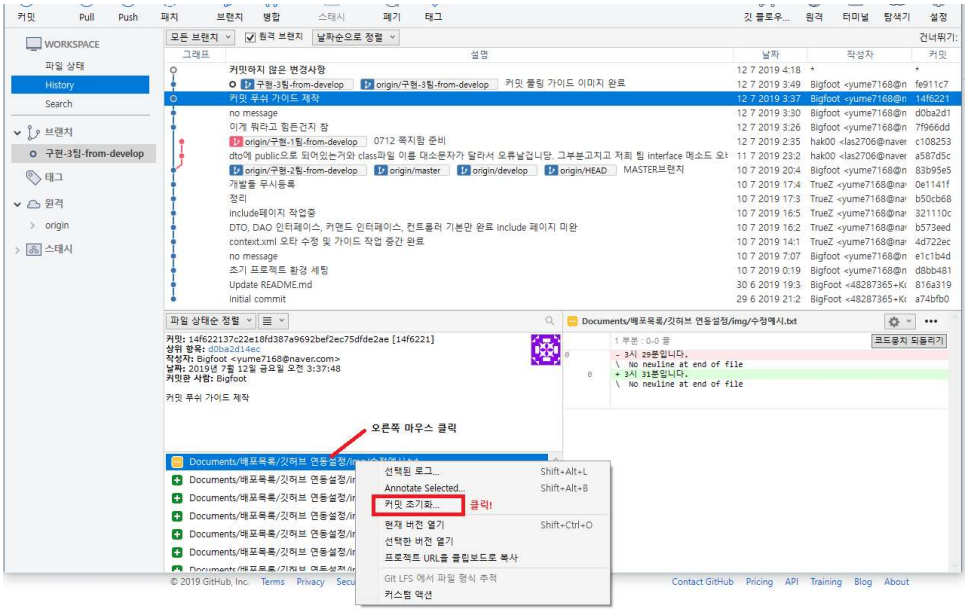
▲ 2. 소스트리에서 해당 레파지토리 탭을 누르시면 "스테이지에 올라가지 않은 파일"에서 그 내역을 확인 하실 수 있습니다.
(소스트리에서 해당파일의 저 화면이 보인다는 것은 이전 버전에는 있었는데 현재는 없어졌다는 것을 의미한다고 아시면 될 것 같습니다)



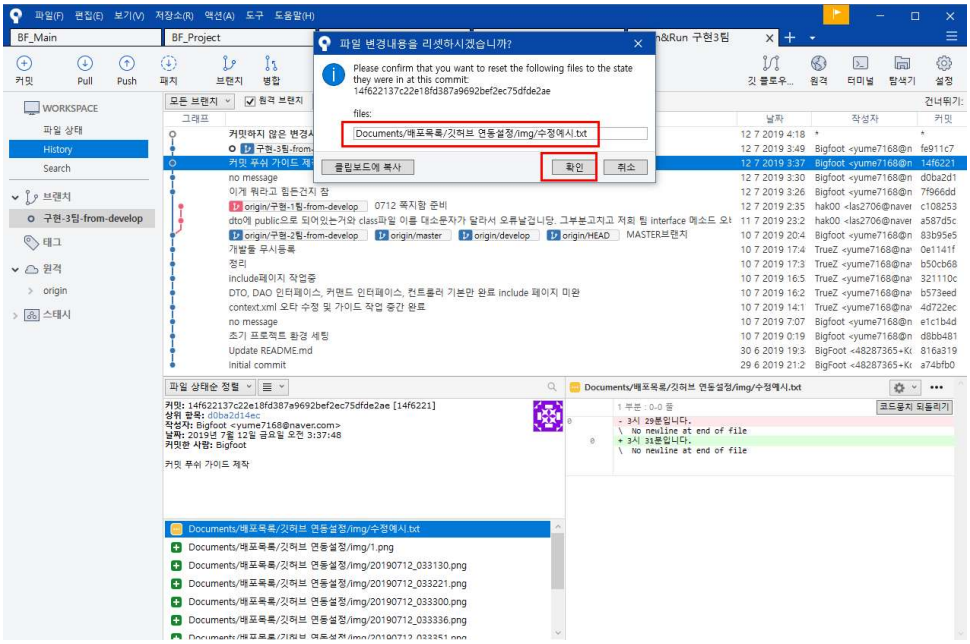
▲ 3. 좌측 사이드바의 history를 눌러서 커밋내역에 진입합니다.



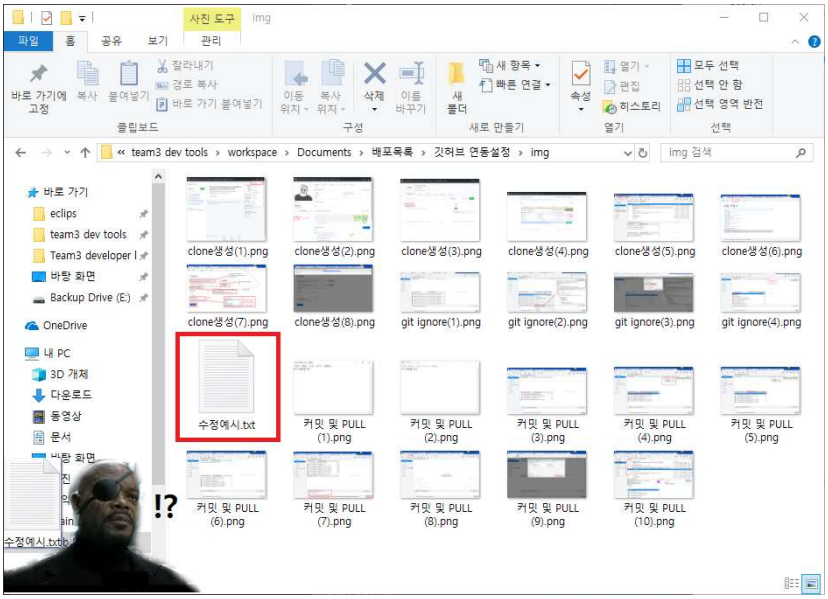
▲ 4. history내역을 살펴보시면 삭제되었던 파일을 확인하실 수 있습니다.
예시에서는 위 그림과 같이 나왔습니다.



▲ 5. history 해당파일에 마우스 오른쪽버튼을 누르시고 커밋 초기화를 눌러주었습니다.



▲ 6. 위와 같이 소스트리에 창이 하나 팝업되는데 저 입력란에 파일을 복구하실 해당 경로를 입력하신 후 확인을 눌러주시면됩니다.
(경로는 보통 최근 작업경로로 자동으로 입력이 되어있습니다.)



▲ 7. 확인을 누른 후 해당 경로로 다시 들어가보니 정말 마법같이 복구가 되었습니다.
이렇게 파일복구를 해주시면 될 것 같습니다 수고하셨습니다