[오늘의 공부 내용]

-commit 의 해쉬값을 이용해서 자신이 원하던 시간대의 파일로 돌려보자.

핵심명령어

Git log

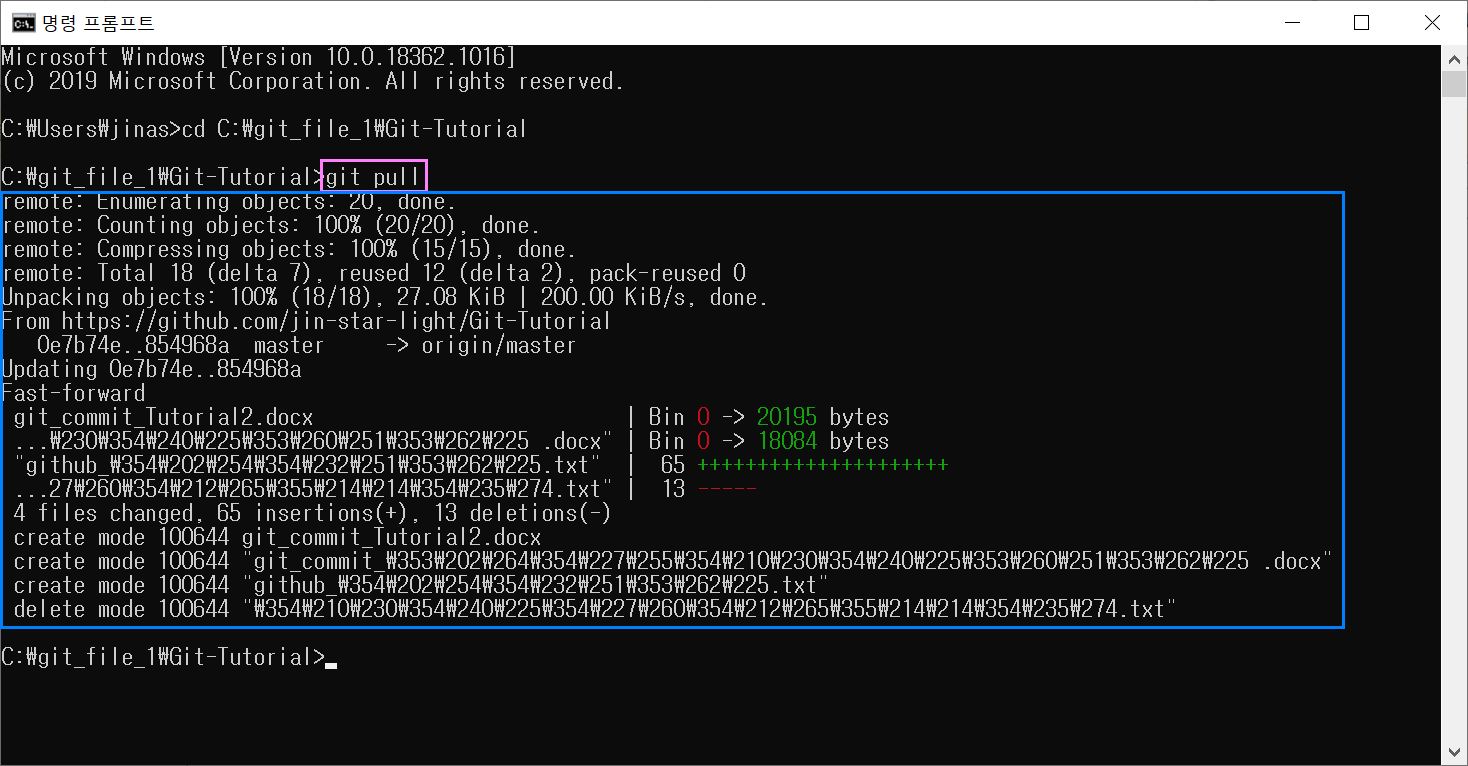
Git reset --hard

Git reset --soft

Git reset --mixed

0.Local Repository와 Remote Repository 파일상태를 동일하게 만들기

git pull을 사용하면 Local Repository와 Remote Repository를 동일한 파일상태로 만들 수 있다.



핑크색 네모 : git pull 명령어를 침.

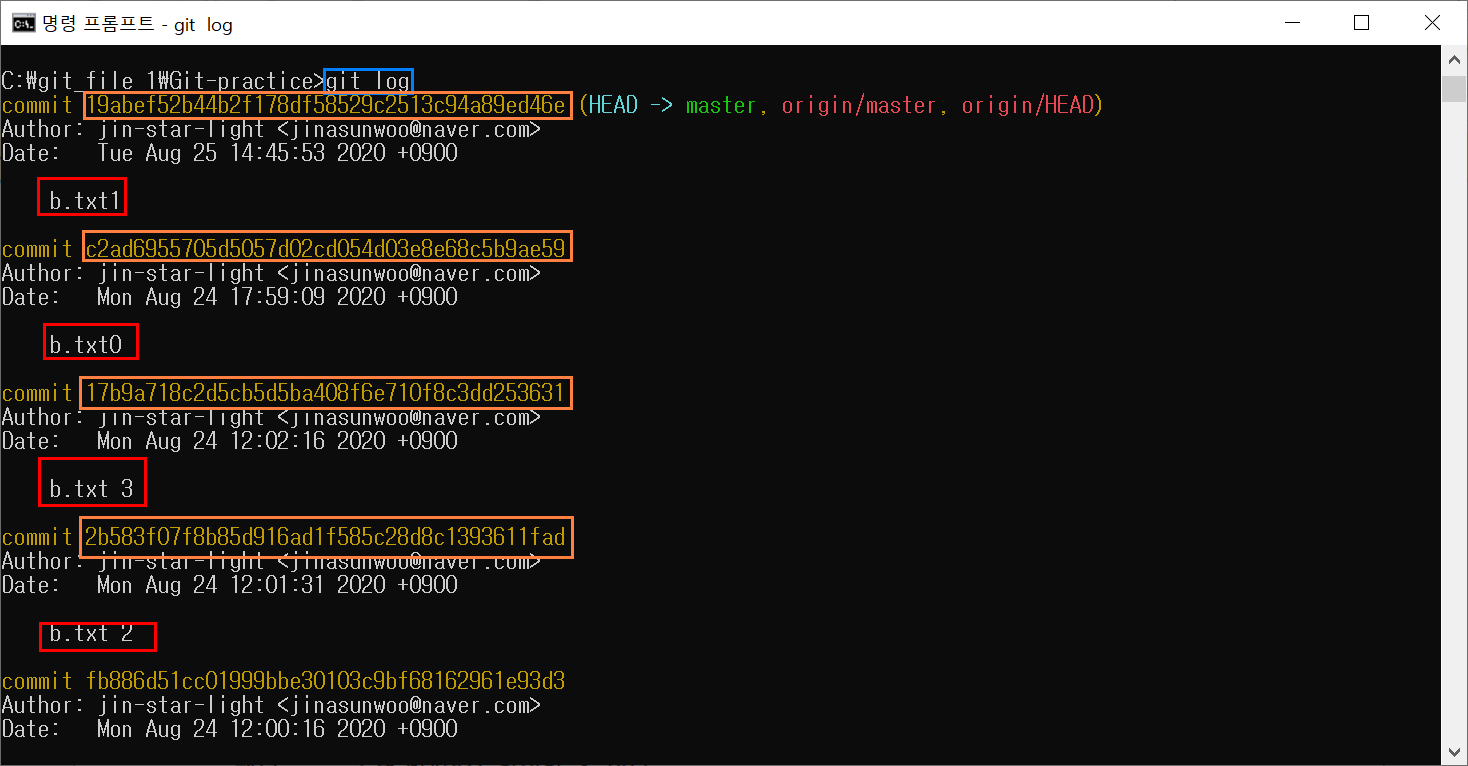
파란색 네모 : git pull로 Local Repository와 Remote Repository의 서로다른 내역을 Remote Repository에서 다운로드 받아온 것으로 두 저장소 모두 같은 파일을 갖기위해 올라간 업데이트 사항이 적혀있다.

[이 명령어가 필요한 이유]

-내가 갖고있는 파일 보다 Remote Repository에 있는 파일이 더 최신파일이라면 추후에 파일을 올릴 때 Local Repository와 Remote Repository의 파일 상태가 다르기 때문에 충돌이 일어난다.

그렇기 때문에 이를 방지하기위해서 혹은 최신버전의 파일 을 사용하기 위해서 git pull을 이용하여 최신버전으로 업데이트하는것이다.

1.Git commit의 반영된 내역을 git log를 통해서 확인할 수있다.



빨간 상자 : commit

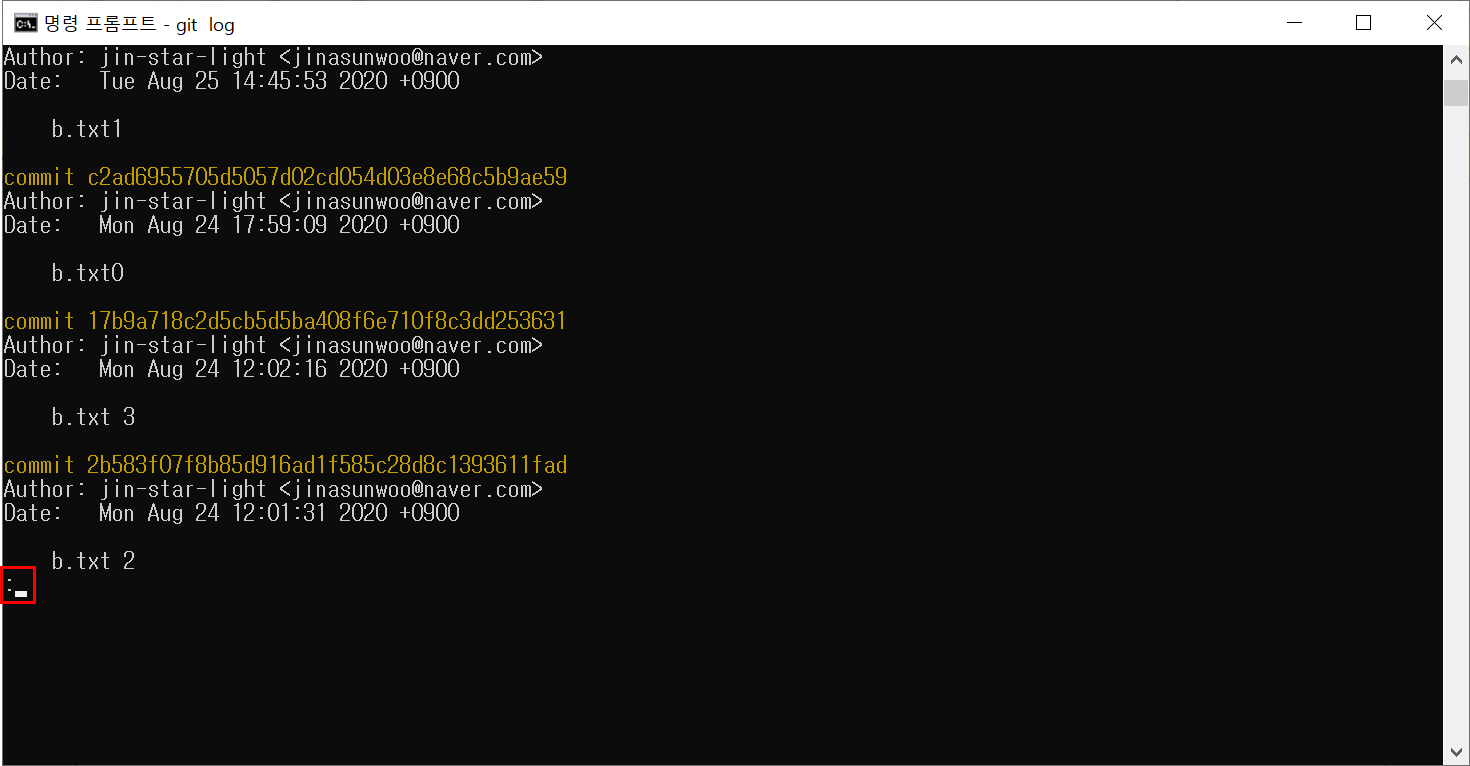
주황상자 : 바로 및 commit에 대한 해쉬값들.

-이 주황상자 ( 해쉬값)을 이용해서 전 버전 commit으로 접근할 수 있다.

모든 commit 변경사항을 알 수 있다.

각각의 commit 내용들은 각각의 해쉬 값으로 매칭이 되어있다.

이 해쉬값들을 사용함으로써 해당 commit 값을 지칭할 수 있다.



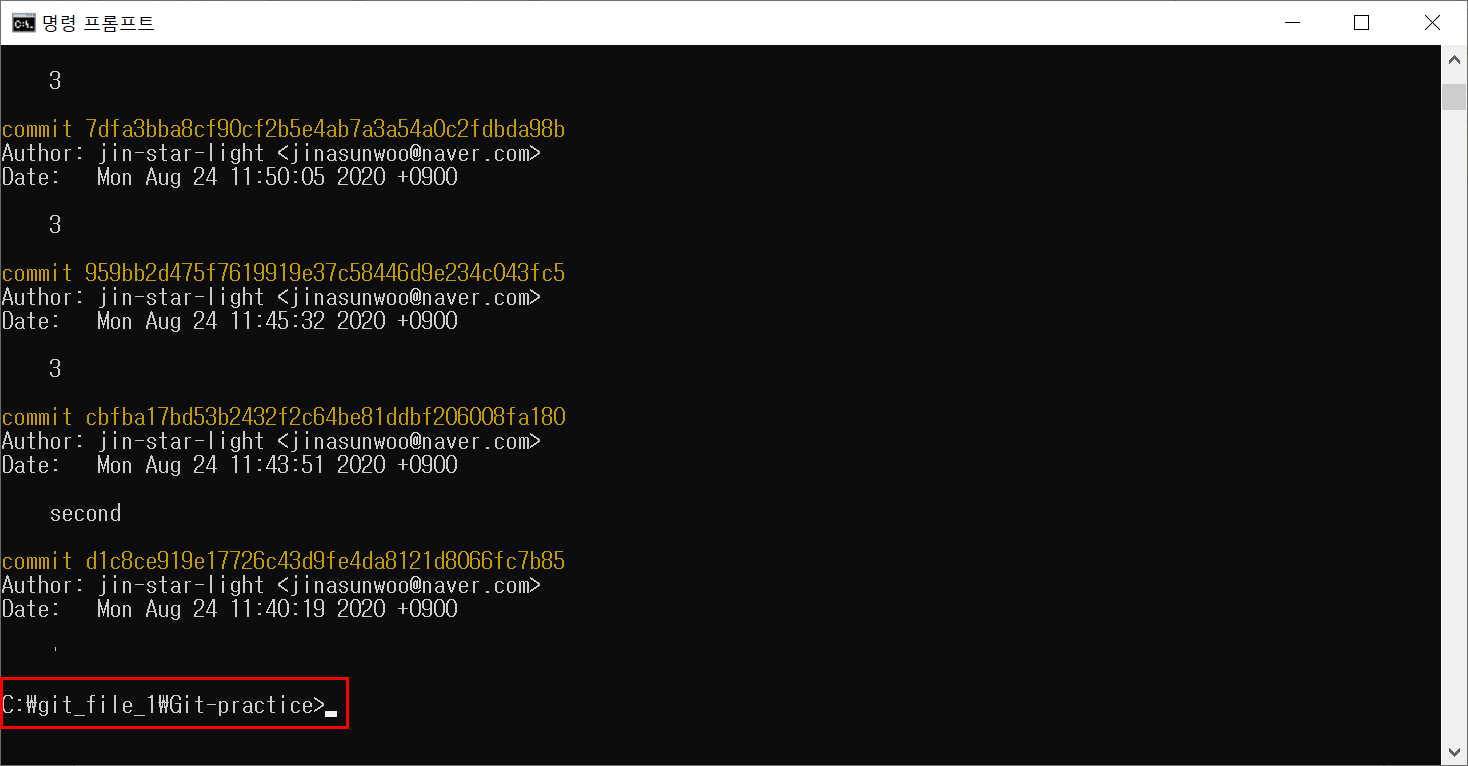
잘보면 맨 밑에 콜론(:)만 있는데 이뜻은.

우리가 실행하고 있는 명령프롬포트 창보다 더 들어갈 내용이 많기 때문에 뜨는 현상으로

더 추가적인 정보를 보고싶다면, enter를 쳐서 추가적으로 볼 수 있다.



만약 끝까지 엔터를 치게 된다면 밑에 (END)라는 창이 뜨는데 이것은 모든 정보를 제공해주었다는 뜻이다. [enter을 쳐도 정보가 더 나오지 않음.]



여기서 git log를 그만 보고싶다면 Q를 눌러서 빠져나올 수 있다.

[END] -> 입력창으로 바뀐 것을 볼 수 있다.

2. Commit 값 되돌려보기.

-cmd창을 보면 git log는 가장 최근에 변경되었던 commit이 위에 있고 내려갈수록 옛날에 변경했던 commit 변경내역을 확인할 수 있다.

-내가 원하던 시점의 (당연히 과거 시점의)commit 의 해쉬값과 git reset을 이용해서 되돌린다.

-여기서 알아두면 좋은점은 git reset [해쉬값]만 실행한다면 git reset --mixed 가 생략 되어있다고 한다. 한마디로 기본 git reset [해쉬값]은 mixed라는 뜻이다.

3.Commit 변경 명령어



git reset --hard [해쉬값]

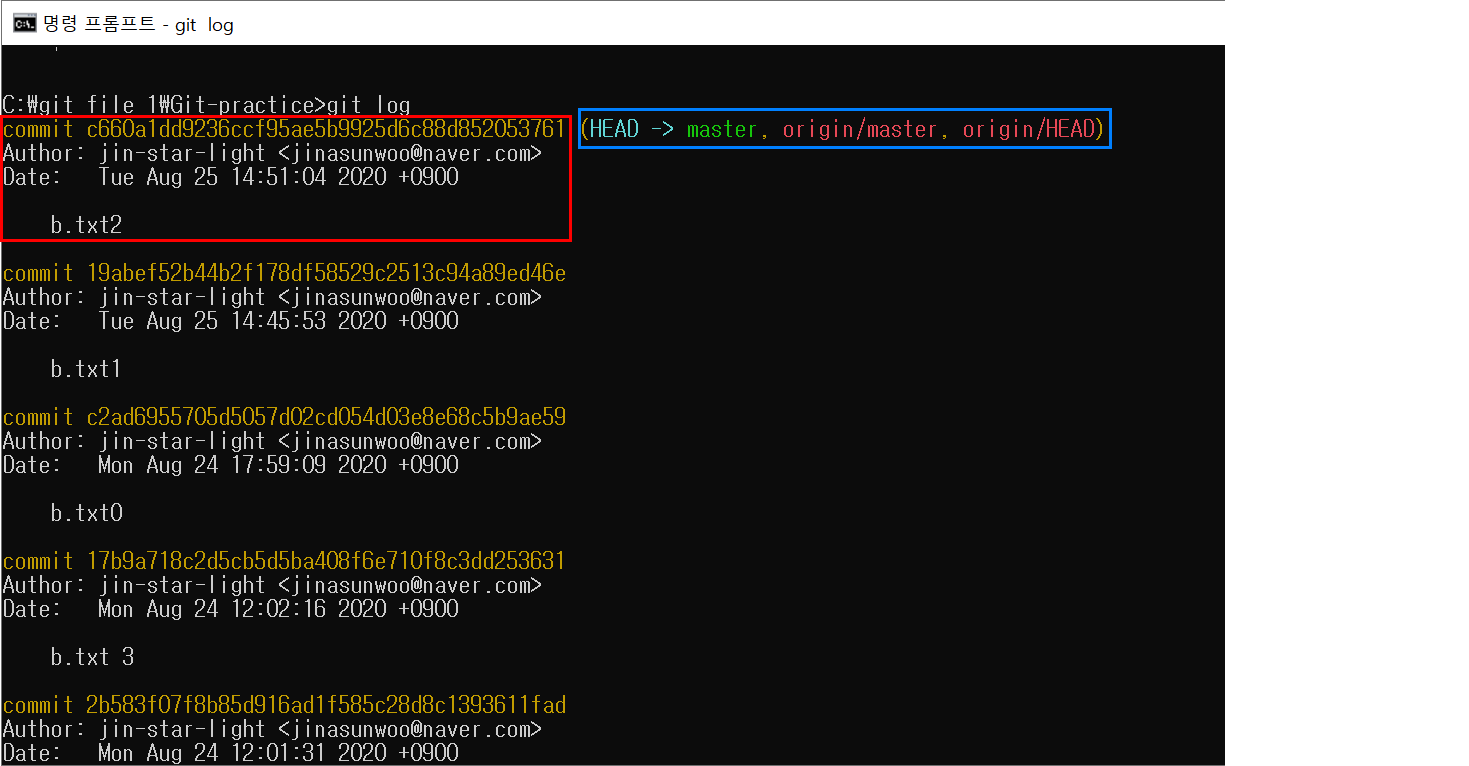
돌아가려는 이력이후의 모든 내용을 지워 버립니다.

hard의 경우 Repository와 stage area, working dir 모두 같은 상태로 돌아간다. 3개의 저장소가 모두 되돌아가기 때문에 따로 저장된 곳은 없게 된다.

-hard를 사용할 때 : 최근작업 내용을 다지워버리고 최신 버전으로 돌아가고 싶을 때 사용한다.

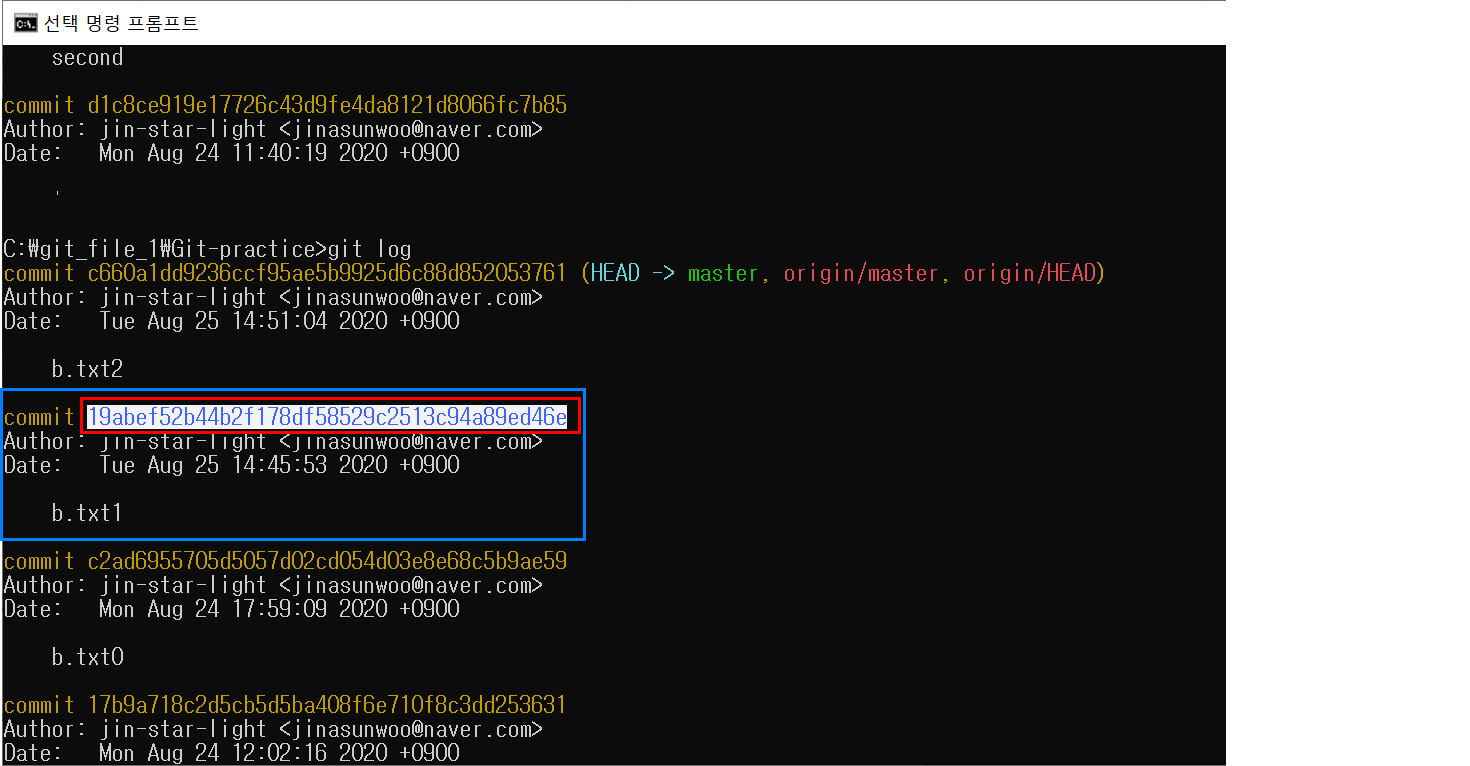
(미련없이 현재의 상태를 버리고 과거로 돌아가고 싶을 때)

특정 commit 지점 이후의 모든 commit값을 지워버리겠다는 뜻이다.

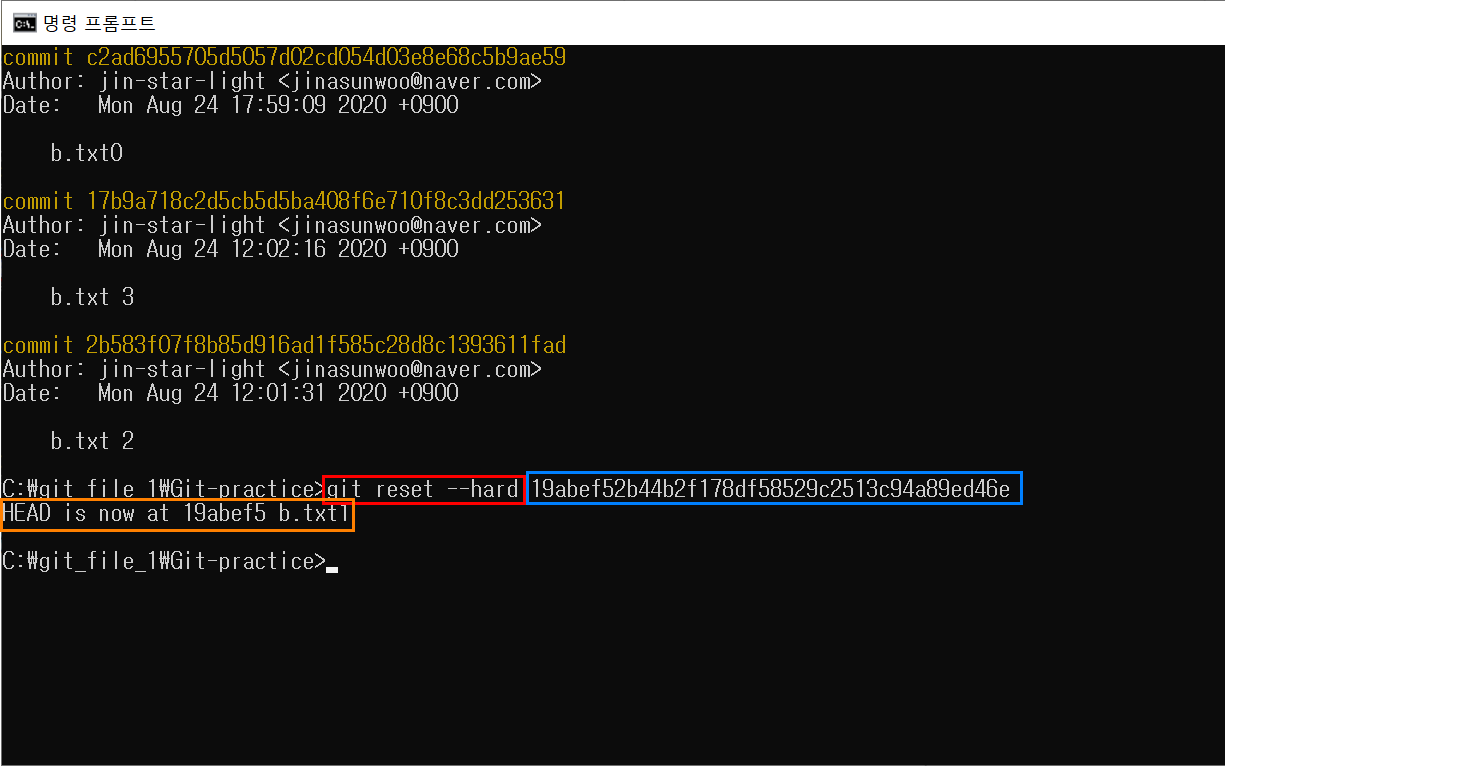


-빨간상자 : 현재 상태의 commit

-파란상자 : 현재 마스터로 지정된 파일

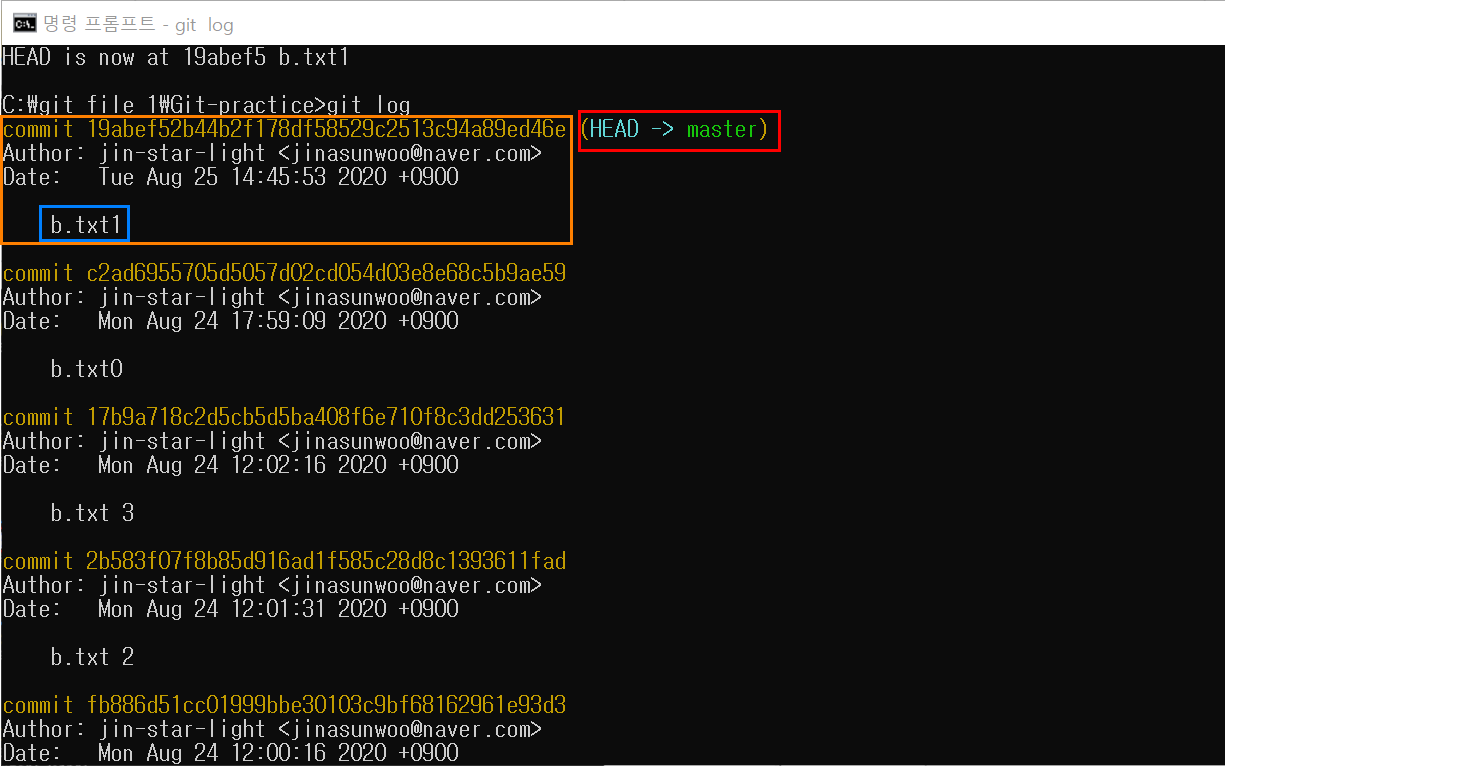


바로전 파일인 commit b.txt의 해쉬값(빨간상자)을 복사해서.



Git reset --hard 뒤에 넣었다.

주황상자를 보면 성공적으로 반영되었다.



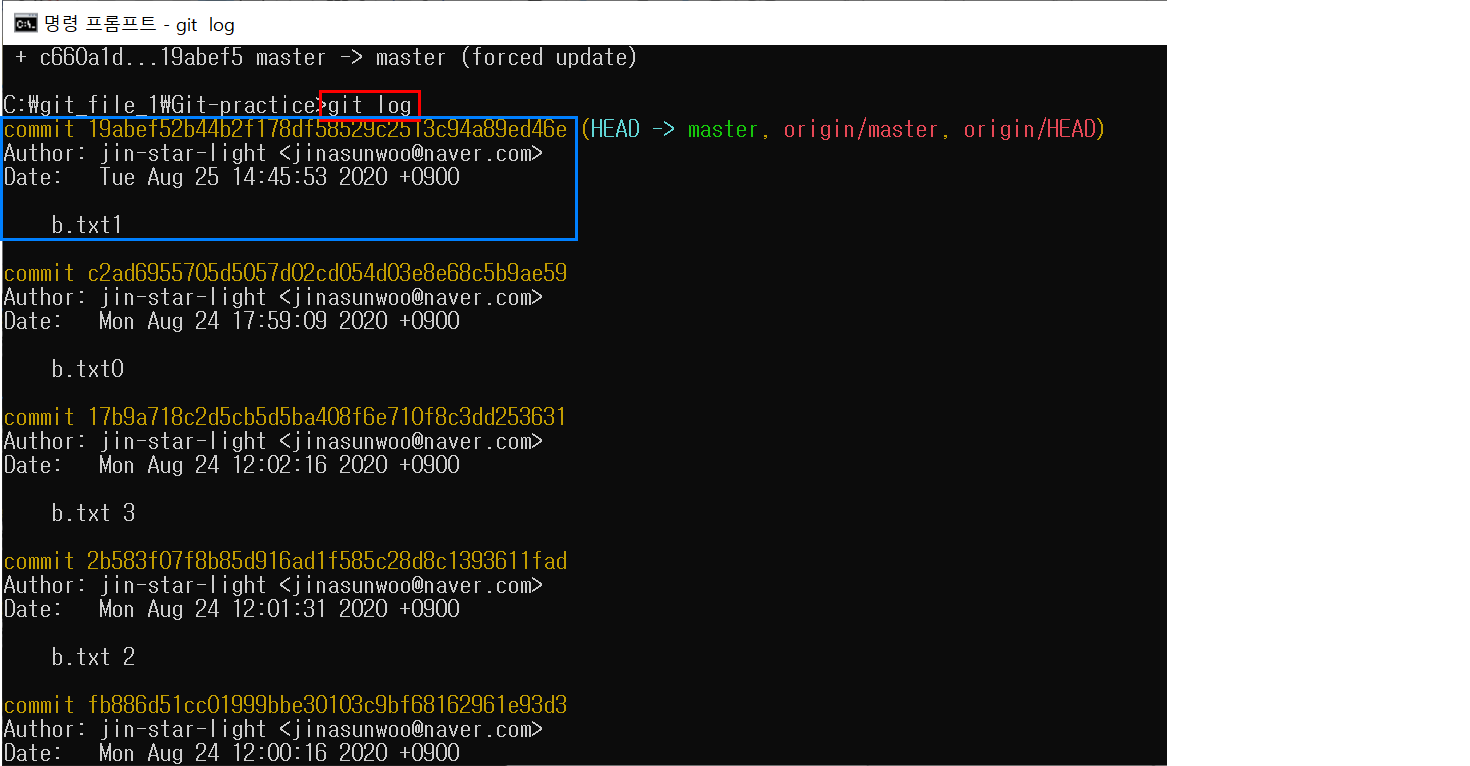
다시 git log를 확인해보면 맨처음 master가 b.txt2가 아닌 b.txt1으로 바뀐 것을 알 수 있다.

Git reset --hard 파일 복구하기.

--hard의 사용으로 가장 최근의 변경사항이 3가지의 저장소에서 모두 사라진 상태.

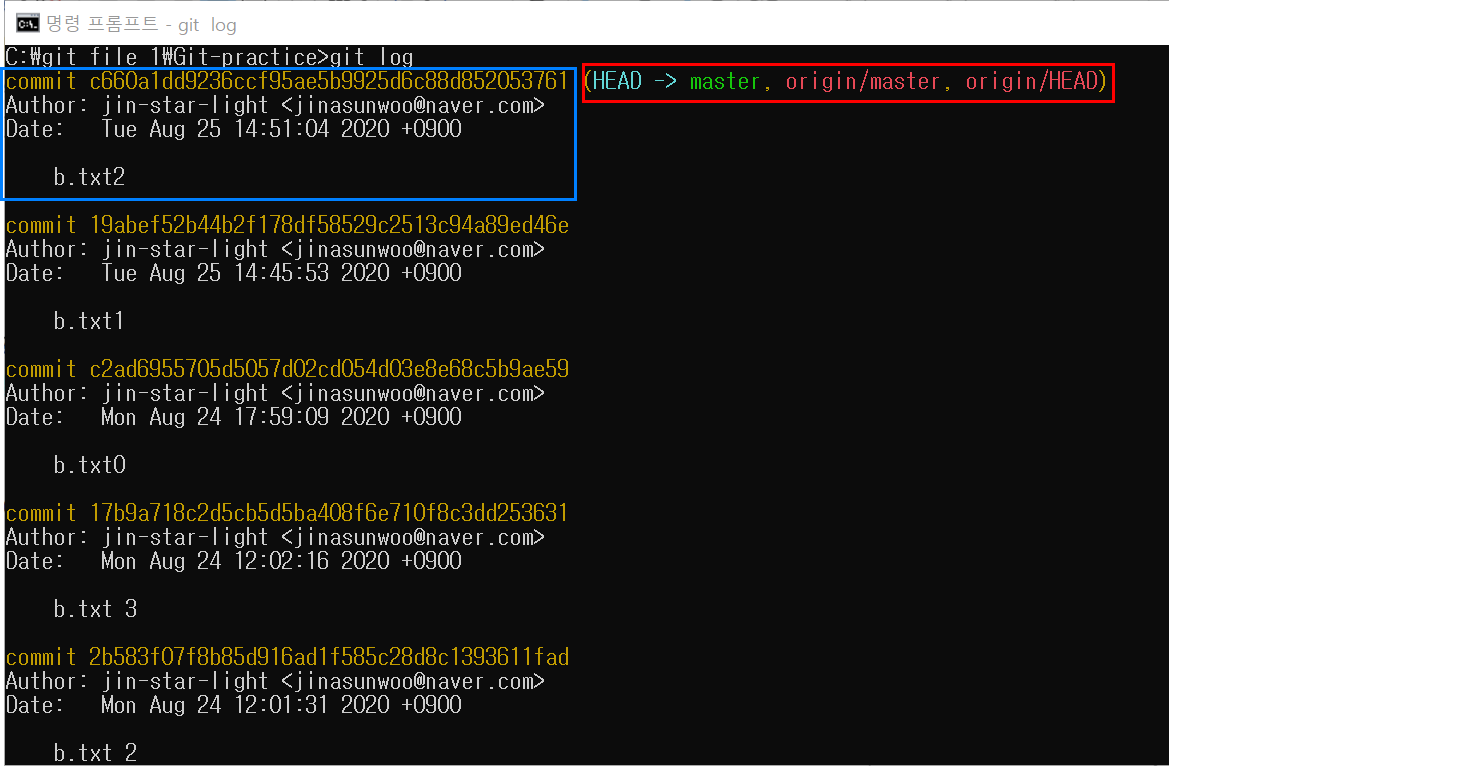
영원히 복구할 수 없는건가? 라고 생각을 했지만 아직 cmd창에 있는 해당 최신변경파일의 해쉬값이 있다면 다시 복구 시킬수 있다고 한다.

아까 위에서 git reset --hard를 이용하여 b.txt2에서 b.txt1으로 바꾼상황 이번에는 다시 b.txt2로 바꾸어보겠다.



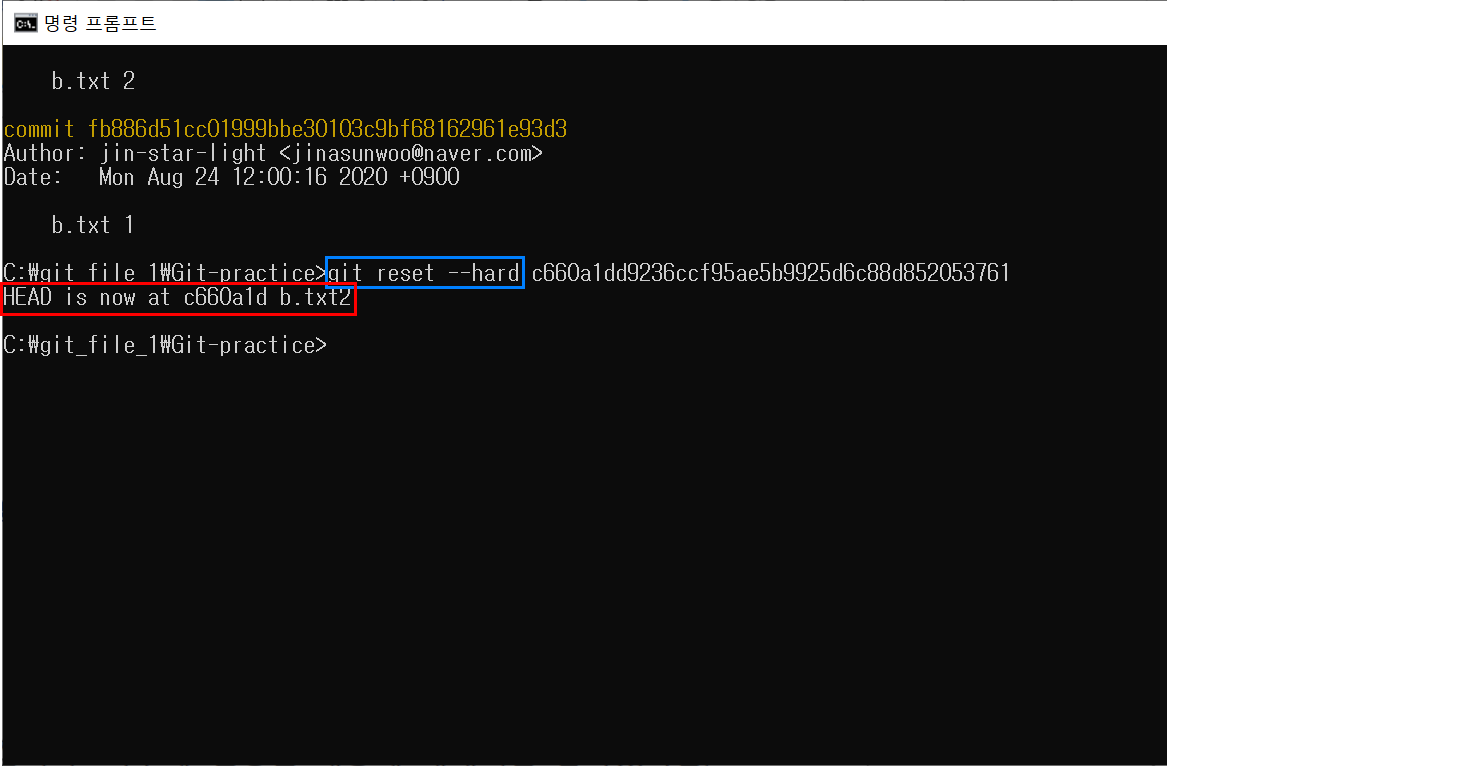
Git log를 다시 쳐봐도 b.txt1 부터나온다.

하지만

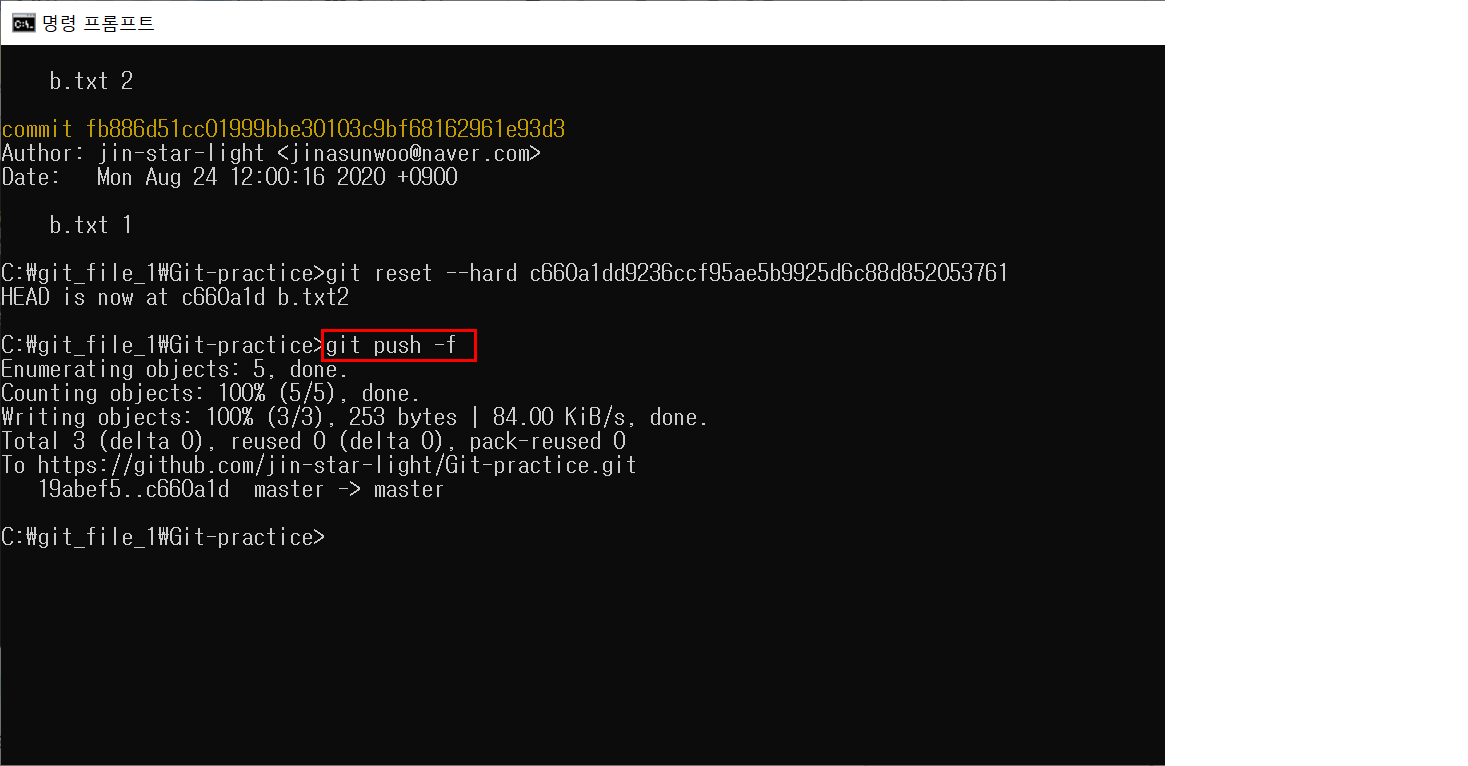


아직 cmd창을 새로 고침하거나 다시 실행시키지 않았다면 위에 log의 흔적이 남아있다.

위사진은 b.txt2가 있던 시절 git log를 한 창이다.

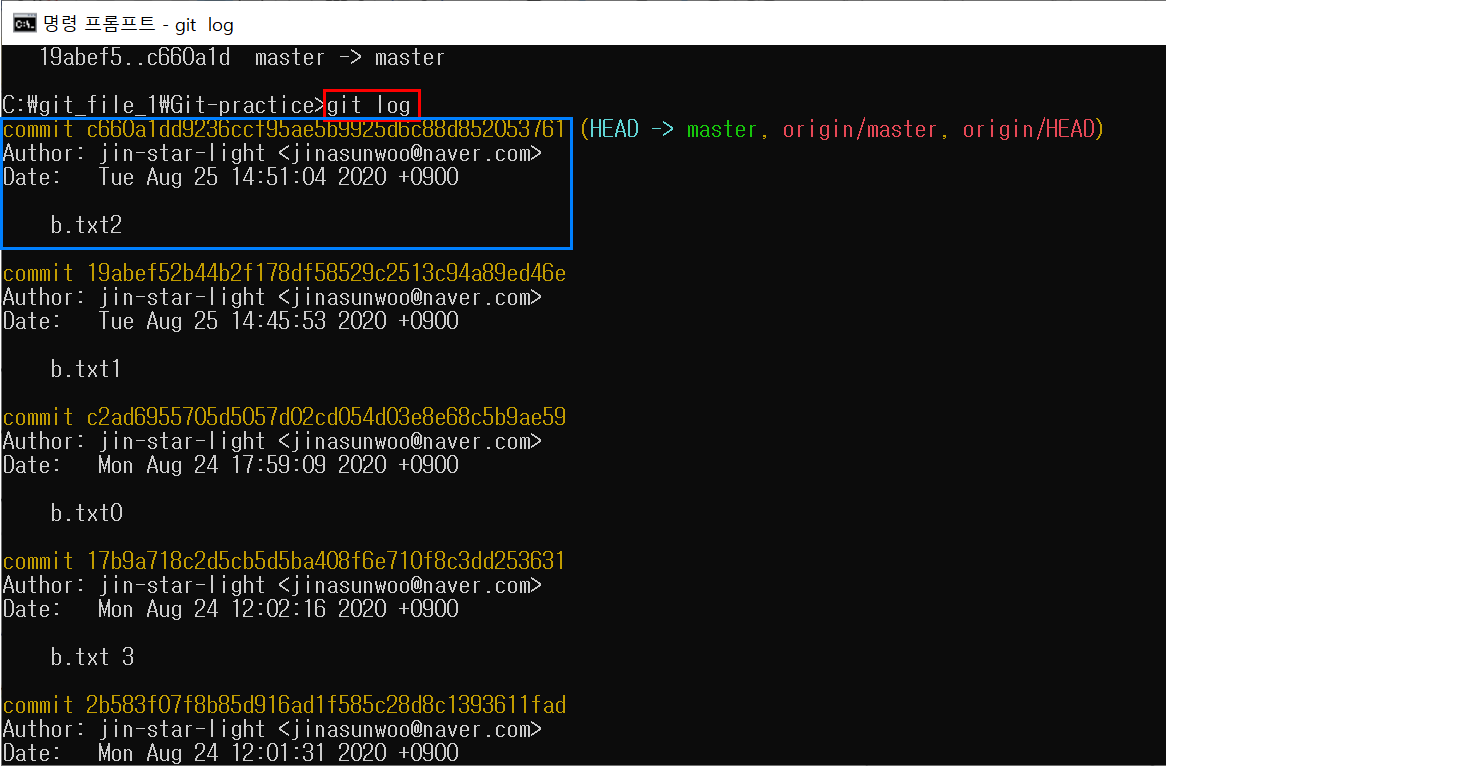


b.txt2 해쉬값을 넣어 실행시키면 이렇게 성공적으로 반영이 되었다고 뜬다.



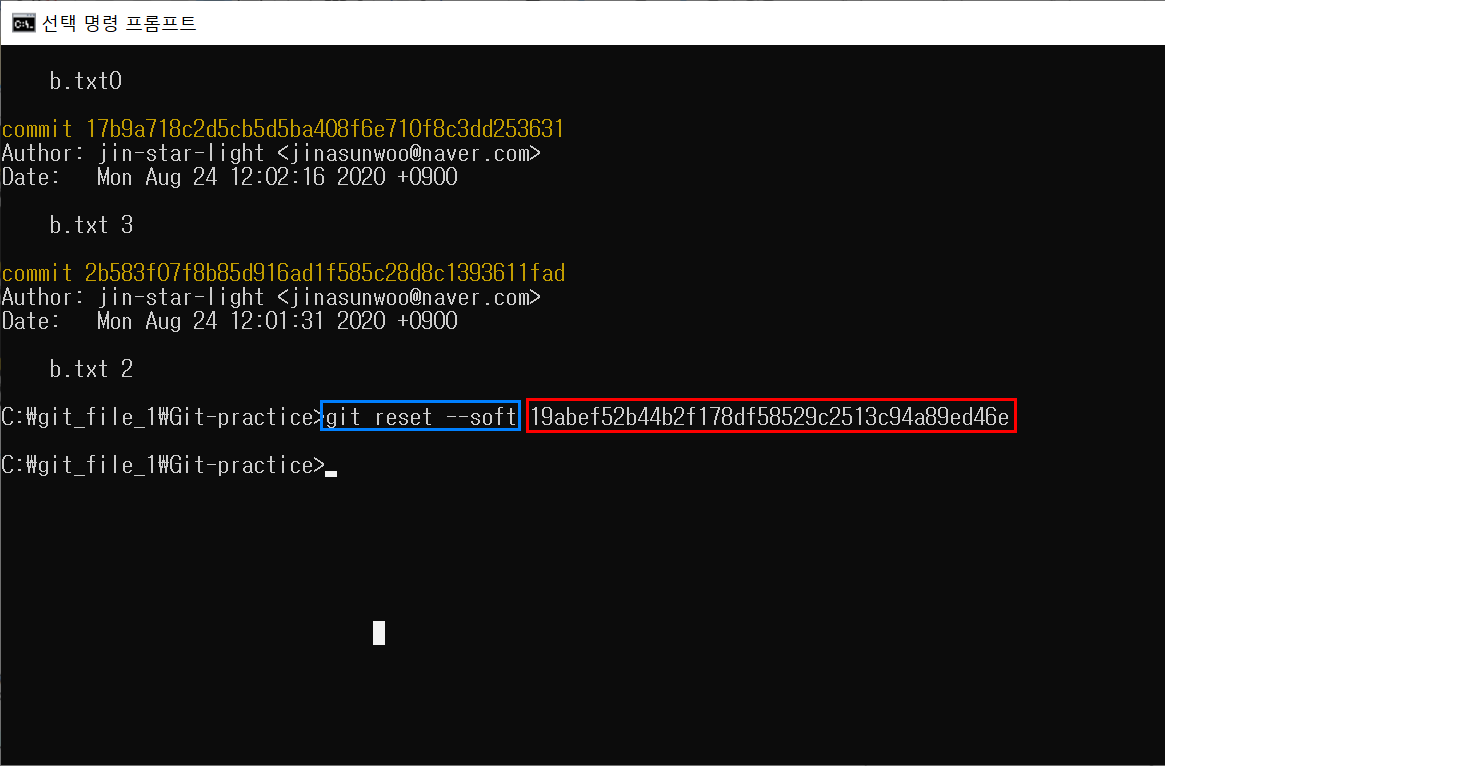
물론 반드시 git push를 해주어야한다는 것도 잊어서는 안된다.

이렇게 다시 복구시킬수는 있지만 git과 함께 쓸 수 있는 유용한 프로그램과 함께 쓰면서 익히는 것도 좋다. 이런 방법보다 좋은 방법이 많다.

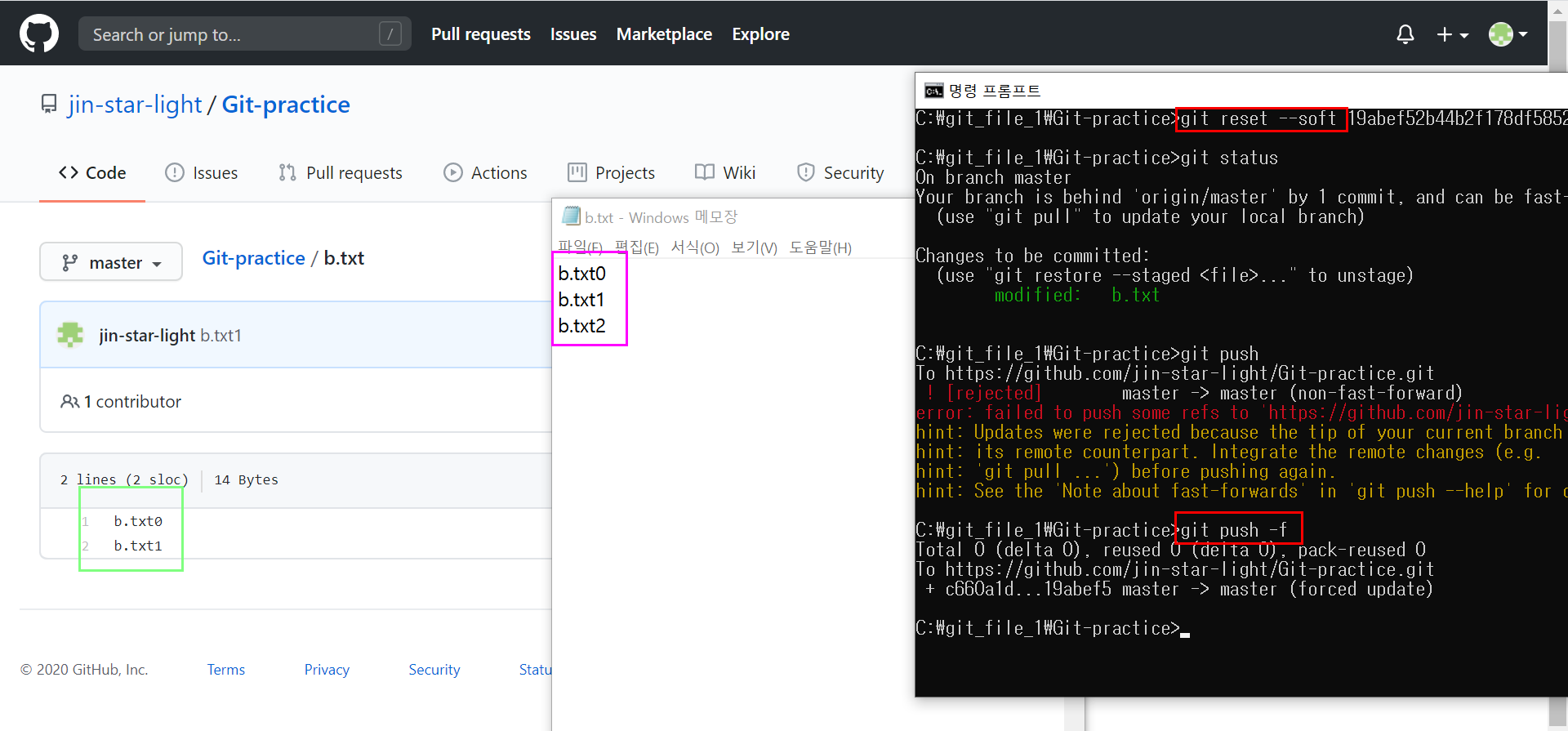


복구후 git log를 들어가보면 b.txt2가 있음을 확인 할 수 있다.

Git reset --sorf [해쉬값]



soft의 해쉬값은 b.txt1이다.



상황 : git reset --soft [b.txt1 해쉬값] > git push -f (빨간상자)를 적용한 상태에다.

분홍상자 - soft를 적용한후 b.txt의 파일(Local Repository)을 확인결과 b.txt2의 형태를 유지하고 있다.

초록상자 - Remote Repository라고도 할 수 있는 git 사이트속 b.txt파일을 올린곳을 확인결과 b.txt1의 형태를 띄고 있음을 확인하였다.



이표를 보면 알수있듯이, hard와 다른점으로는 soft의 변경사항은 오직 Repository에서만 적용된다는 것이다. 이뜻은 내가 작업하는 working dir와 stage area는 변경되지 않음으로써 3이라는 파일을 보존하는 동시에 2라는 파일은 git에 올릴 수 있다는 이야기이다.

이로써 soft의 경우는

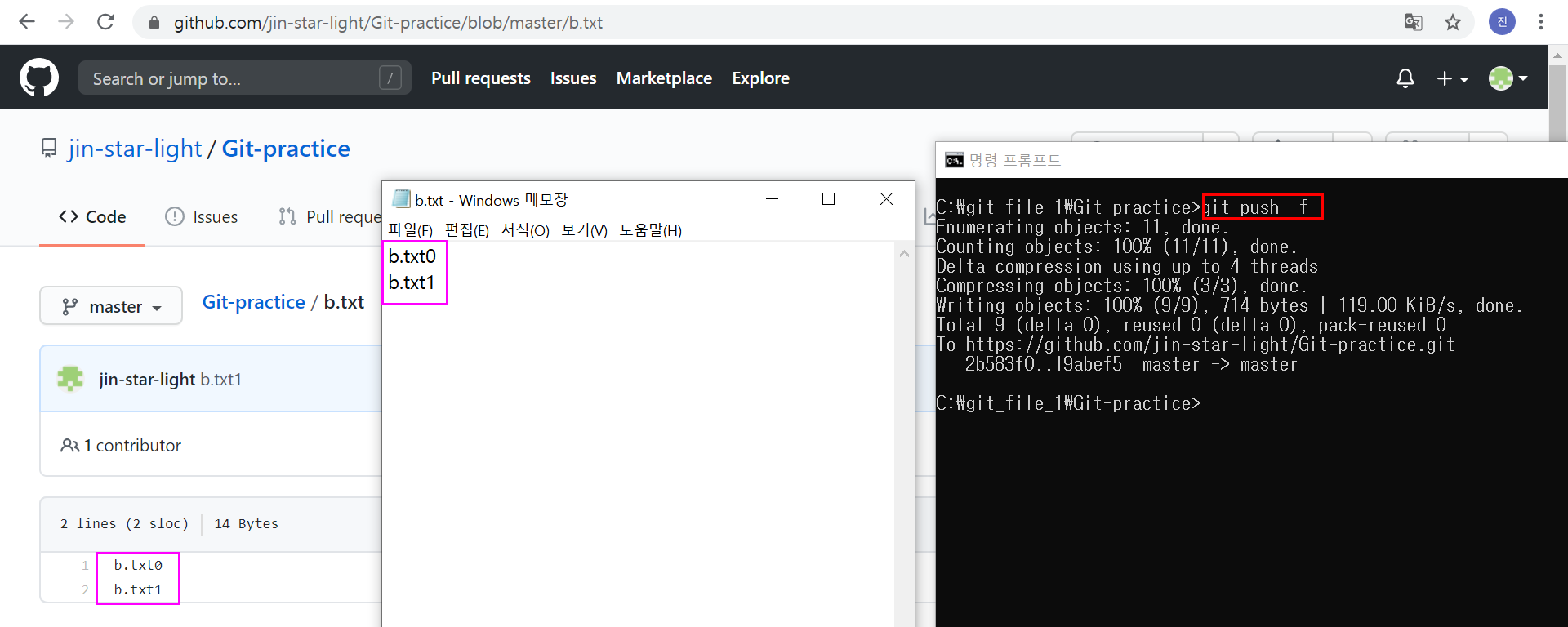
1. 최신 변경파일은 보존하고 싶지만 git에는 올리고 싶지않은 파일들?

2. 프로젝트를 진행하다 앗! 뭔가 새로운 아이디어가 떠올랐어!! 브랜치를 하나 만들어볼까?  
  
최근 작업 내용을 그대로 갖고와서 브랜치를 만들 고 싶을 때

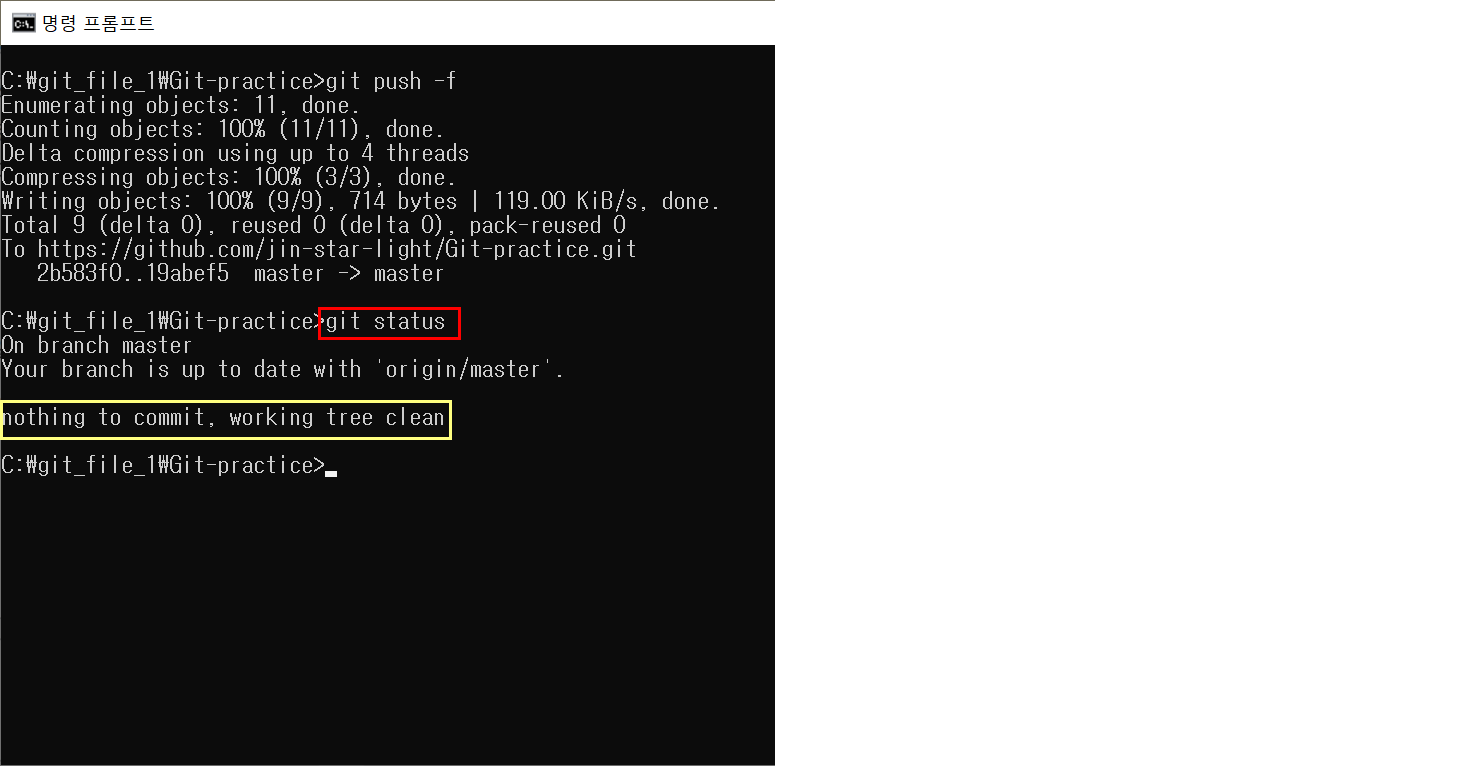
3. reset한 버전과 현재까지의 작업을 합쳐 새로운 버전 만들 때

사용하면 된다.

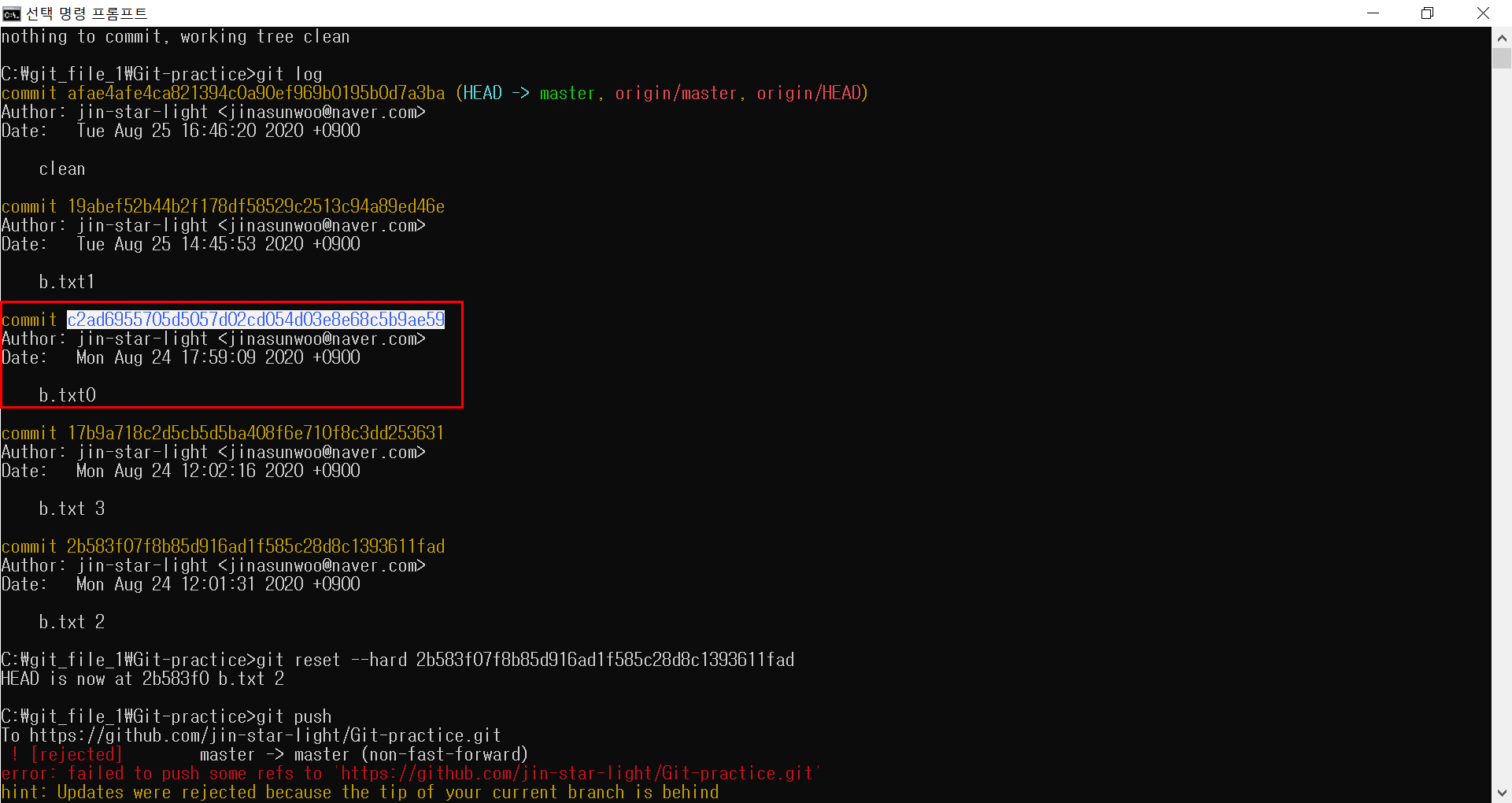
Git reset --mixed [해쉬값]



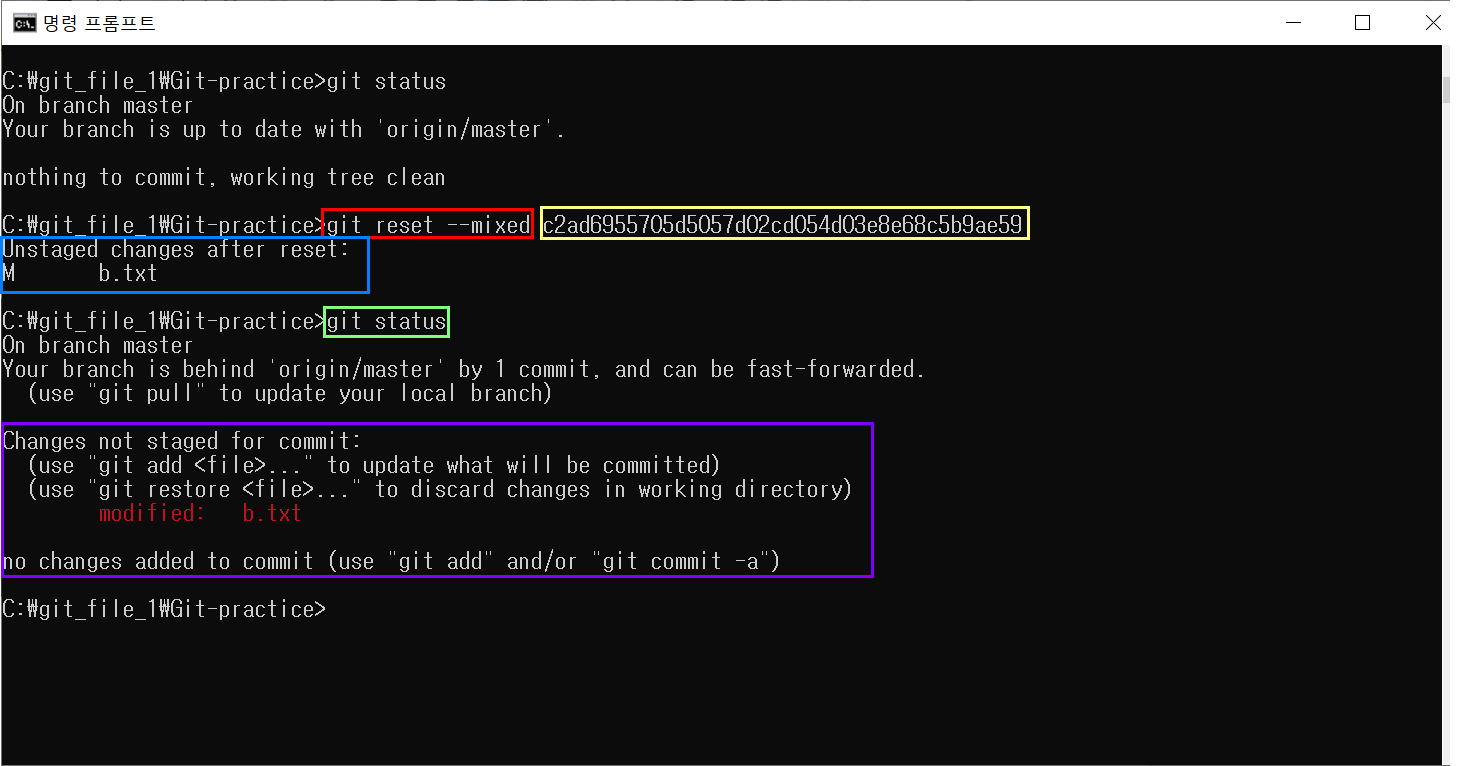
현재상황 : Remote Repository과 Local Repository를 git reset --hard (b.txt1해쉬값)으로 맞추었다.



status에서는 아직 아무것도 없다고 나오는 상황.



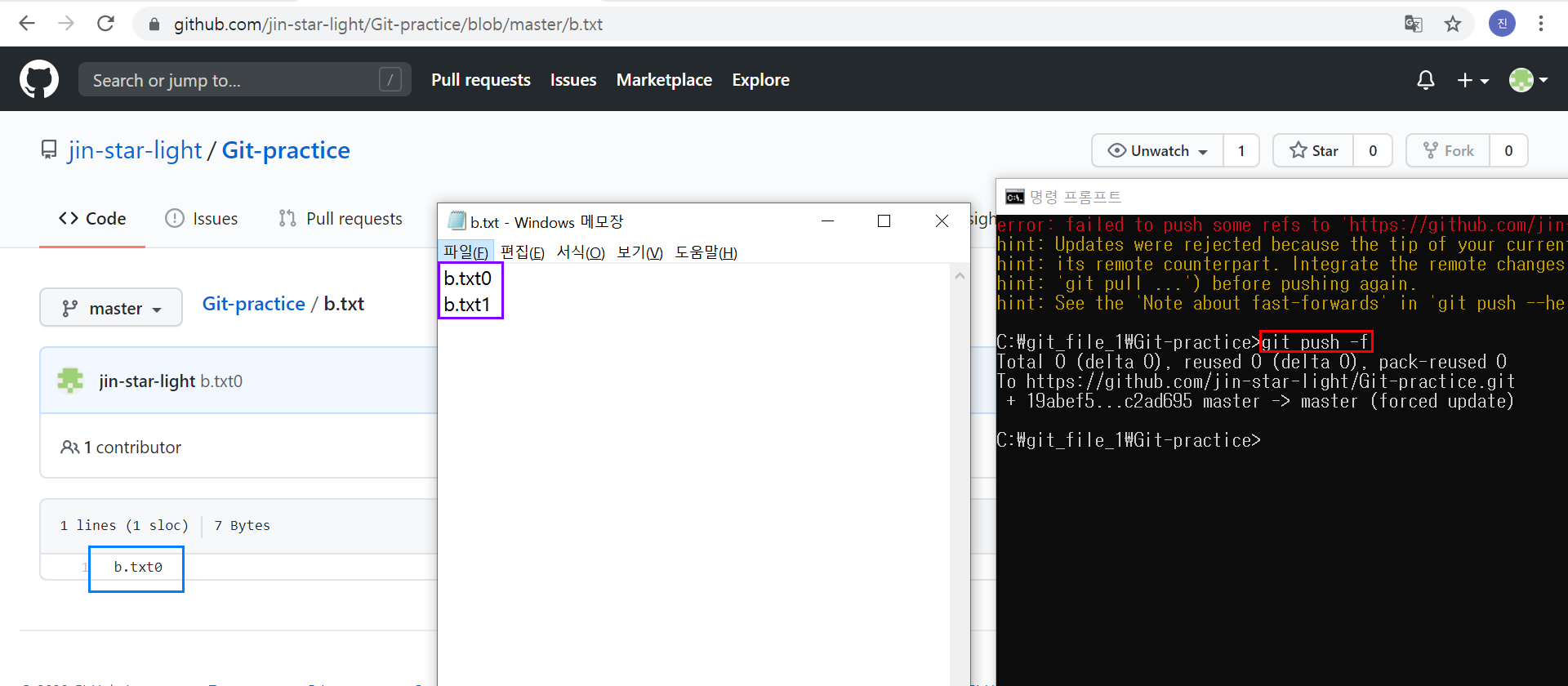
이번에는 b.txt0으로 해쉬값을 적용하여 mixed를 사용해볼 것이다.

git reset --mixed (b.txt0 해쉬값) [빨강,노랑]을 적용하고 나서

Git status(초록상자)를 실행시켜보니

잘보면 아까와 다르게 (아까는 status영역이 비어있었음) 새롭게 status에 등록되지 않은 파일이 있다고 나온다. (보라색박스)

심지어 밑에는 git add혹은 git commit을 눌러 적용시키라는 내용이 있다.



Git push -f를 하여 git 저장소에 올렸더니 (빨간상자)

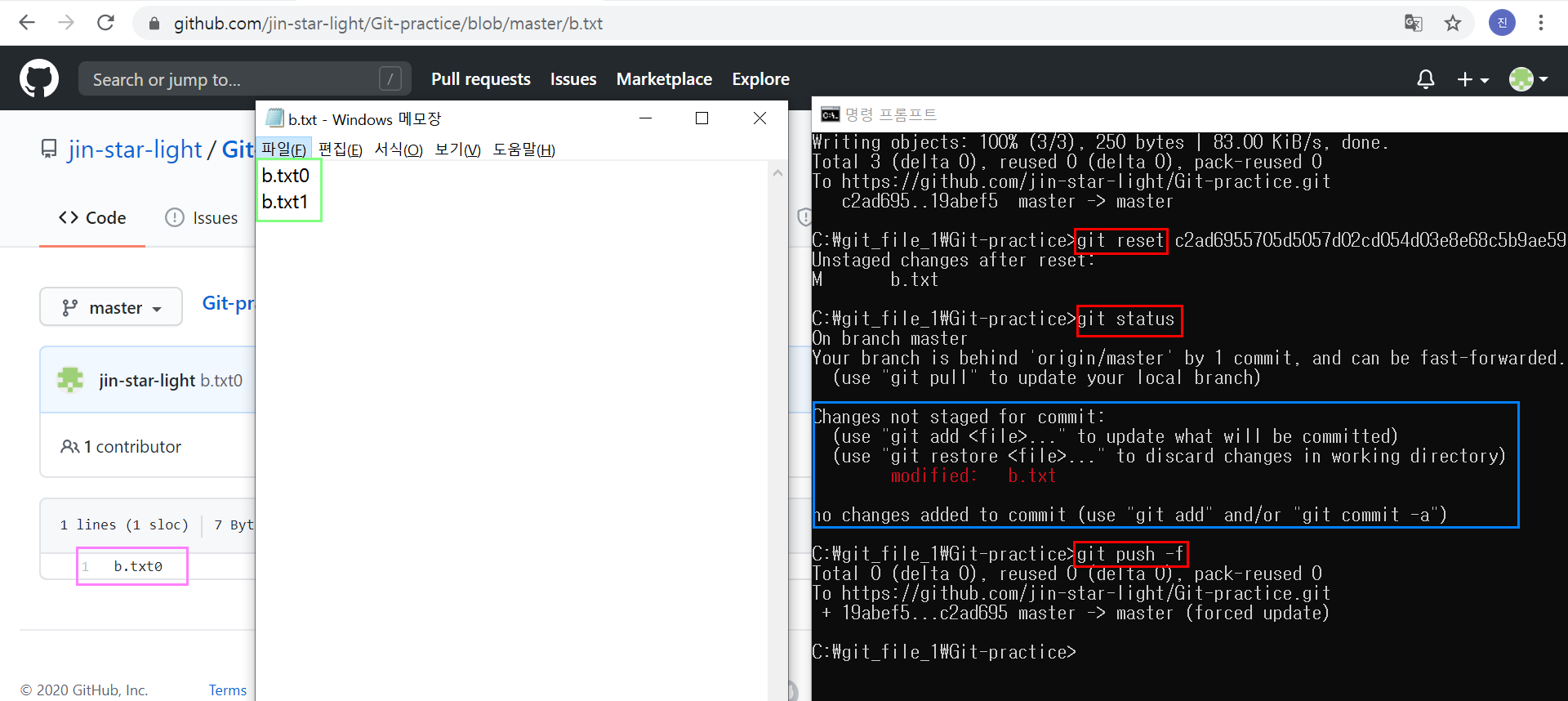
Remote Repository 에는 b.txt0이 적용된 모습을 보여준다.

또한 아까 status 즉 stage area에도 새로 적용된 모습을 볼 수있었다.

하지만

Local Repository 에서는 b.txt1의 모습을 모여준다.

이뿐만이아니라



Git reset --mixed [해쉬값]은 git reset [해쉬값]으로 생략이 가능하며 위에서 언급했듯이

기본값으로 적용된다. 위 사진은 git reset [해쉬값]으로 앞에서 언급한 git reset --mixed와 동일한 과정과 결과를 보여준다.

Git reset --mixed [해쉬값] = git reset [해쉬값]



표로 정리하자면 이렇게 나오는데, mixied는 기본값인 만큼 자주 쓰이는것같다.

mixied를 사용하는 경우 :

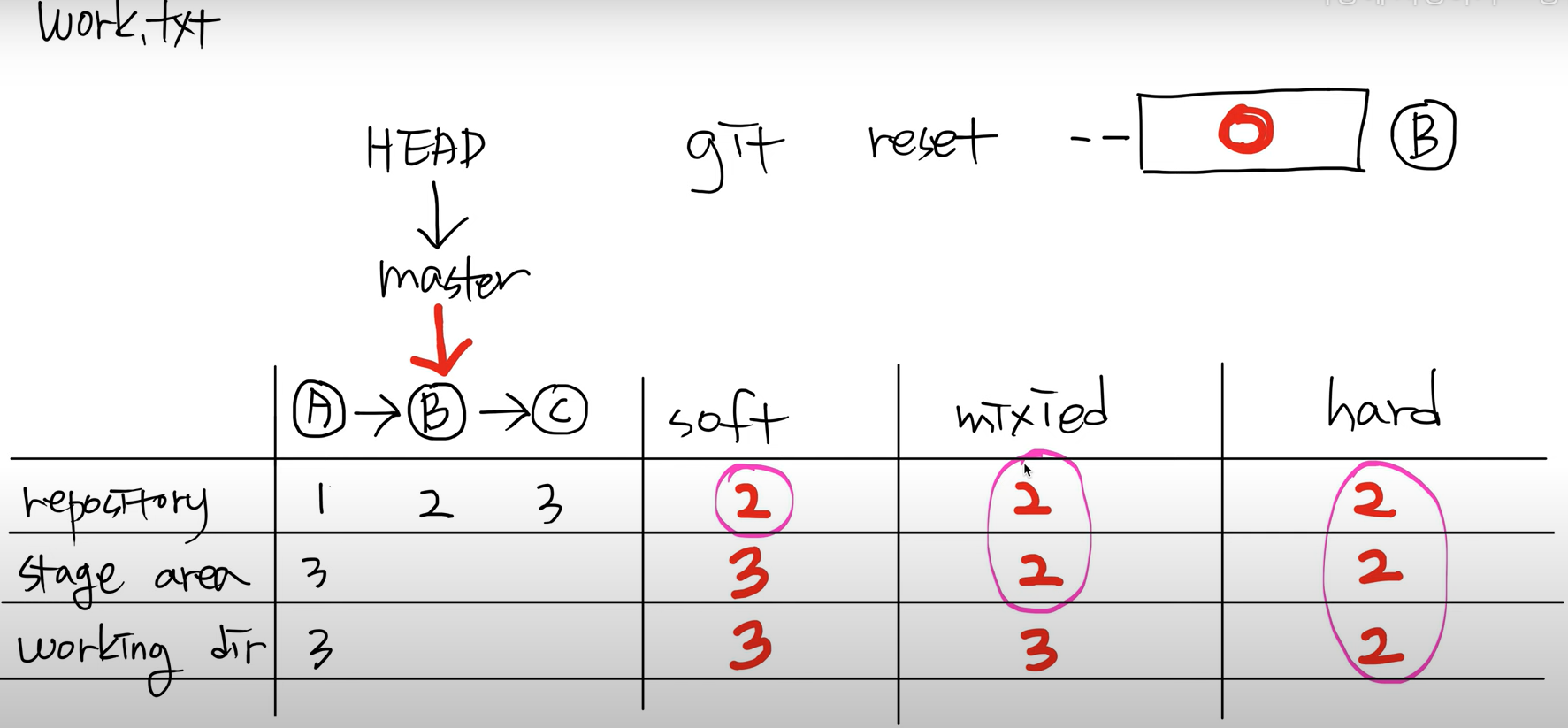
현재 작업물은 지우긴 싫고, 이전 버전으로 돌아가서 add할지 말지 결정해야 할 때

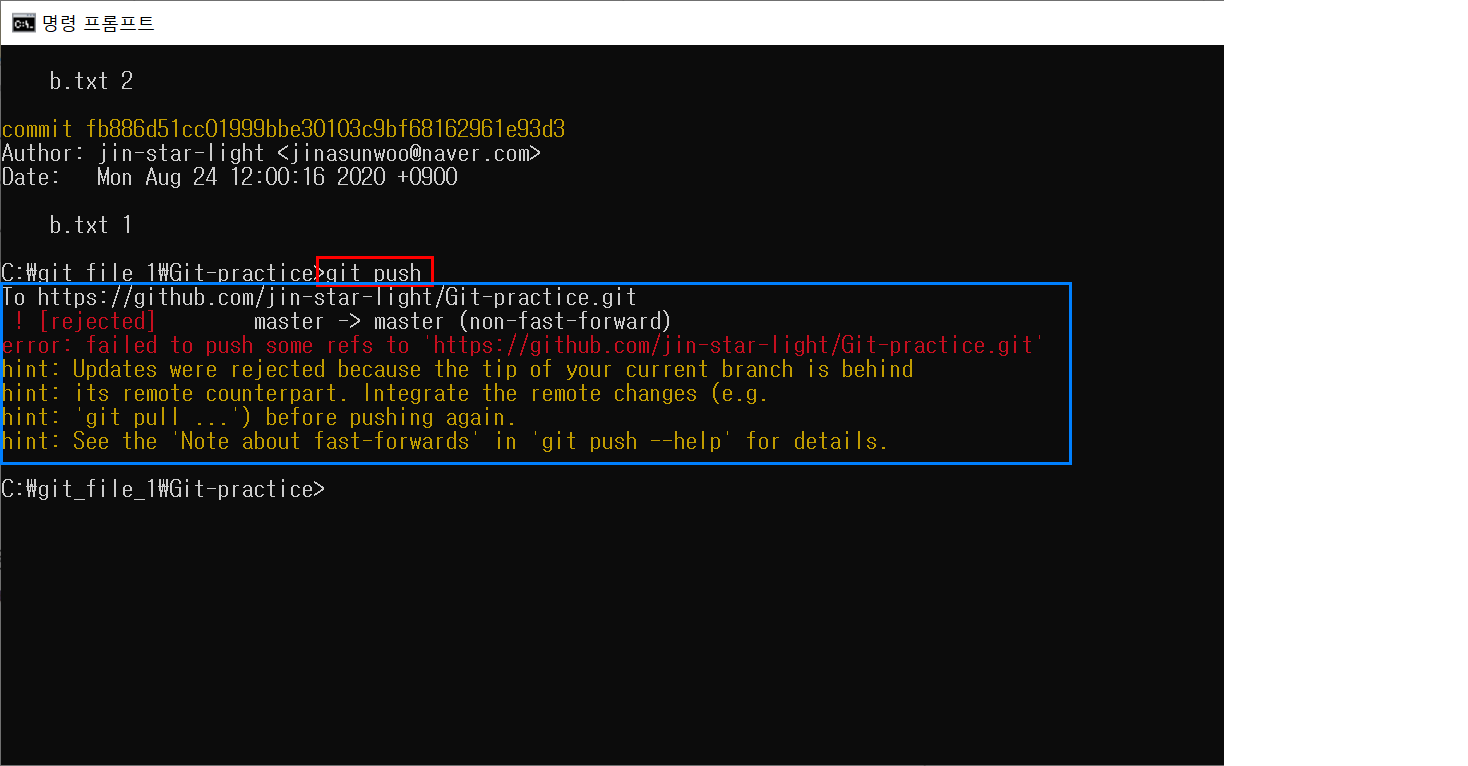
출처

<https://www.devpools.kr/2017/02/05/%EC%B4%88%EB%B3%B4%EC%9A%A9-git-%EB%90%98%EB%8F%8C%EB%A6%AC%EA%B8%B0-reset-revert/>

[git reset의 hard/soft/mixed의 차이점 알아보기.]

https://opentutorials.org/module/4032/24533





Git reset을 이용하여 전 버전의 파일들을 적용한후 확정된 파일을 다시 git push하려고 하지만

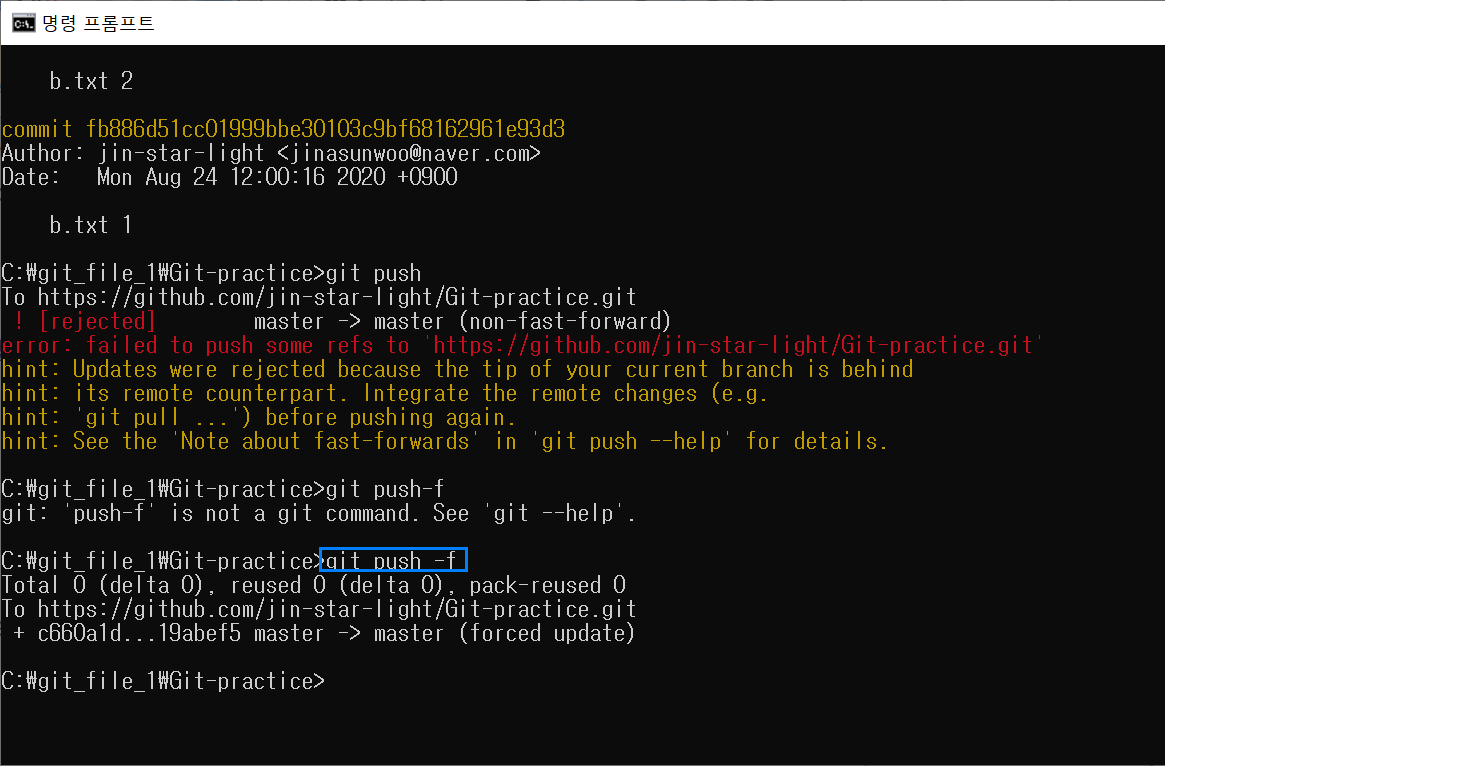
위와 같은 에러가 뜨는 이유는

git reset를 사용하게 되면 변경된 commit시점은 Local Repository와 Remote Repository이 다르게 나타난다.

현재 우리가 변경한 commit부분은 (사용한 명령어들마다 다르겠지만) Local Repository등에서 변경된 사항이고 Remote Repository에서는 적용이 되지 않은상태이다.

여기서 알아야할 것은 git push를 아무리 해봤자 Local Repository에서 적용한 파일 상태를 Remote Repository로 반영할 수 없다는 점이다.

이렇게 오류 에러가 뜨는데 이유는 Local Repository와 Remote Repository의 상태**가 다르기 때문이다.**



그래도 Local Repository의 상태를 그대로 적용하고 싶다면 강제로 넣으면 되는데 명령어는 바로

Git push -f이다.

이 명령어를 사용하면 Local Repository와 Remote Repository 모두 같아진 것을 확인할 수 있다.